An Intelligent Hybrid System for Financial Decision Support

Suran Goonatilake

Thesis submitted for the degree of
Doctor of Philosophy
of the University of London.

April 1994

Department of Computer Science
University College London

ABSTRACT

Recently there has been much research in using intelligent techniques to assist decision making. It is noted, however, that most intelligent techniques have limitations, and that they are not universally applicable to all decision making tasks. Each intelligent technique has particular computational properties making them suitable for certain tasks over others. Against this backdrop, this thesis puts forward a novel intelligent hybrid systems approach for supporting financial decision making.

An important contribution of the thesis is the formulation of a new classification scheme for intelligent hybrid systems, which takes into account factors such as functionality, processing architecture and communication requirements. The three proposed classes of hybrid systems are Function-Replacing, Intercommunicating and Polymorphic. This classification provides a mechanism to make qualitative assessments of existing hybrid systems and also helps to guide the development of new hybrid architectures.

The key requirements of a financial decision support system have been analysed in-depth. It is argued that an ideal decision support system for financial decision making should satisfy a range of criteria including: the ability to induce decision making knowledge from domain data; the ability to process fuzzy relationships; the ability to adapt to changes in the decision making environment; the ability to provide explanations and the ability to allow decision makers to change and add new knowledge. A hybrid system which demonstrates these computational concepts, INTENT, is developed and its effectiveness demonstrated in the complex decision making task of foreign exchange trading. INTENT combines expert systems, fuzzy systems, genetic algorithms and neural networks to produce an effective decision support system.

All decision models derived from the system have been critically evaluated by a domain expert. This provides a mechanism to assess the effectiveness of the approach in terms of its explanation capabilities and the ease of judgmental revisions to decision models. The quantitative performance of the approach has been assessed against the decision-making performance of a human trader, and has also been compared with results in the trading systems literature.

With respect to data pre-processing, a novel clustering-based method for converting raw domain data into *symbolic* linguistic descriptions is developed. This method is further extended to convert raw domain data into fuzzy logic descriptions by finding appropriate fuzzy membership functions. A method of using genetic algorithms to induce fuzzy models is also developed and its effectiveness is demonstrated.

Based on the experiences obtained during this project, a preliminary scheme for hybrid systems development is proposed. This scheme draws upon our three hybrid classes and offers a set of guidelines based on the problem domain and assessments of computational properties of different intelligent techniques.

ACKNOWLEDGEMENTS

This work wouldn't have been possible without the support of various people. Firstly, I am very grateful to Professor John Campbell for the invaluable encouragement and guidance received throughout the period under his supervision.

Secondly, I would like to express my deep appreciation to Cal Futures for the financial support, without which I could not have accomplished this work.

I would also like to thank all my colleagues in the Computer Science Department at UCL for their friendship and advice. In particular, I am grateful to Sukhdev Khebbal, Jason Kingdon, Konrad Feldman, Ugur Bilge, Laura Dekker, Jose Ribeiro and Prof. Philip Treleaven for their advice and assistance during this project. I am also grateful to Dr. Chris Satchwell and Klaus Hebben of Cal Futures for introducing me to financial trading methods and systems. Special thanks to Joaquin Castillo Arminan of Oracle Trading Systems for the many hours spent during the knowledge elicitation and validation exercises. I am also indebted to Dr. Nesar Ahmad of the Indian Institute of Technology, New Delhi, for introducing me to Fuzzy Logic, and to Dr. Amala De Silva of the University of Colombo for introducing me to economic modelling.

My sincere thanks go also to Archana Srivastava and Dr. Mike Luck for their helpful comments on the written work. Special thanks to Dr. Paulo Rocha for providing Latex advice and macros.

Finally, I would like to thank my parents, and all my friends including Tanya Coelho, Nilanthi Swampillai and Gaya Ranawaka for their moral support over the last (long) four years.

CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1
Introduction

This chapter presents: a brief introduction to AI and decision making; the research goals of the thesis; an overview of the hybrid decision support systems and the contributions of this work to intelligent decision support systems and intelligent hybrid systems.

1.1 AI and Decision Making

Most people regardless of whether they are doctors, investment analysts, urban planners or bank managers, spend a large proportion of their time making decisions. Many years of education and training at universities and workplaces have equipped these professionals with knowledge and skills to make effective decisions. Decision theory [76], has attempted to understand and formalise decision processes with the aim of improving the quality of decision making. Increasingly, Artificial Intelligence (AI) methods are being used to understand, model, and build systems to aid decision making. This thesis, as discussed in detail in section 1.4 & 1.5, investigates the use of hybrid artificial intelligence methods to support decision making. We have investigated the combination of Expert Systems, Fuzzy Systems, Genetic Algorithms and Neural Networks for making effective decisions.

The interaction between artificial intelligence and decision making is almost as old as the AI discipline itself. The work of one of the earliest pioneers of AI, Herbert Simon, is still after 30 years widely discussed in the management literature [87]. His 1960 book 'The new science of management decision' [113] still provides a cohesive framework to understand and model decision processes. Simon's work in decision making developed from his research into computational models of human cognitive processes. These computational models of human problem solving later led to the development of *expert systems* that now support a variety of decision making activities in domains ranging from medicine to finance. We begin our discussion on the nature of decision making by introducing Simon's classification of decision making [113].

1.1.1 Programmed and Non-programmed decision making

Simon [113] classifies all decision making into two categories: *programmed decisions* and *non-programmed decisions.* Programmed decisions are routine, repetitive, well-structured decisions for which definitive procedures exist. Examples of such decision making include calculating salary payments, re-ordering office supplies and numerous factory floor decisions. A conventional computer program can easily be written to automate this type of decision process.

Non-programmed decisions are by contrast, novel, unstructured, and ill-defined decisions with no clear-cut procedure for automation. This could be either because a particular decision situation has not previously arisen or because the precise structure of the problem is elusive. Examples of such decisions include diagnosing of diseases, investing in securities and evaluating company loans.

Mitra [88] observes that programmed decision making is typically made by relatively low-paid workers at the bottom end of the corporate hierarchies, while non-programmed decision making is done by senior executives. In addition, the frequency of decision making decreases on moving up the decision making hierarchy (see figure 1.1) [88]. While programmed industrial control decisions are made as frequently as once every 10 seconds, for example, strategic decisions can be made once every year.

Automating Programmed Decision Making

Computers have been used in organisations primarily to automate programmed decision making. Tasks that clerks used to perform routinely have been automated with programs for inventory control, payroll calculations and customer supply ordering. On advanced factory floors, robotic manufacturing units have also automated much programmed decision making. In all of these situations, the tasks or decision making procedures, can be clearly specified as a straightforward algorithm which can then be implemented in a computer program.

Automating Non-programmed Decision Making

Simon's recipe for automating non-programmed decision making is to use *heuristic* AI methods [113]. He observes that programs such as the General Problem Solver

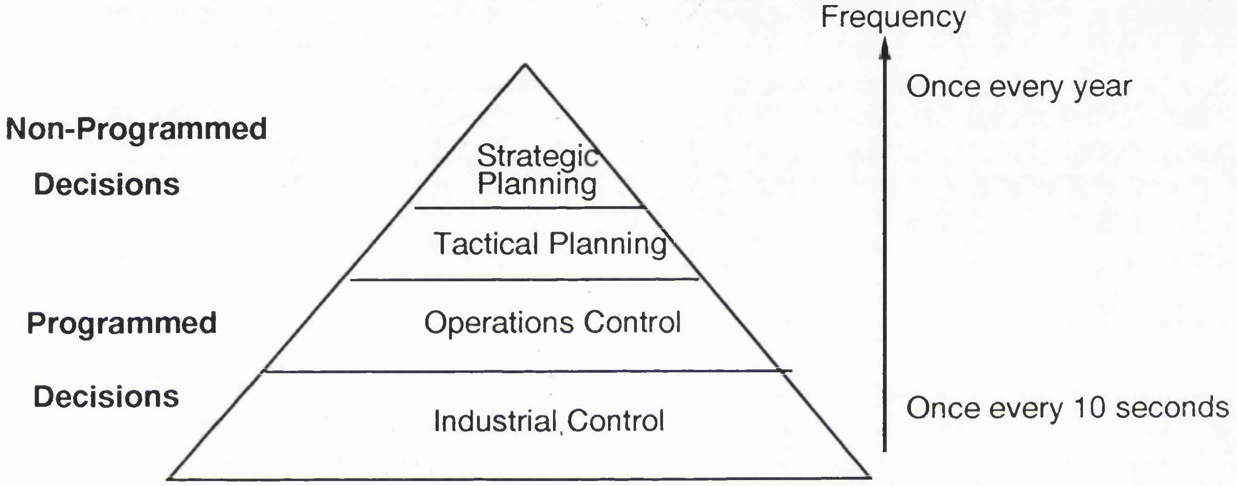


Figure 1.1: Frequency of Decision Making (Adapted from Mitra [1986])

(GPS) [91] which incorporate expert 'rules of thumb' can reduce seemingly intractable problems into solvable sub-problems.

As further merit for the heuristic approach to problem solving, Simon [113] cites the failure of operations research methods, (such as linear programming), to solve large optimisation problems (such as job shop scheduling), and the success of heuristic methods in handling such problems. Expert systems, first introduced in the 1970's [100], use some of these same heuristic problem solving methods. These systems are built by eliciting knowledge from domain experts concerning particular tasks, and have been used in a variety of decision making tasks ranging from medical diagnosis to credit evaluation. The expert knowledge embodied in heuristics provides shortcuts in searching large problem spaces, thus enabling the solution of problems that are intractable by exhaustive search methods [100].

In the 1980s, expert systems were seen as providing an opportunity to automate a large proportion of non-programmed decision making, thus potentially automating a large number of white-collar professional tasks [36]. However, this did not occur for several reasons. A key issue was the difficulty in designing, implementing and maintaining expert systems.

Knowledge elicitation for expert systems is time consuming, expensive and potentially unreliable [43]. Experts find it difficult to articulate particular types of 'intuitive' knowledge [94] and are sometimes unwilling to participate in lengthy knowledge elicitation exercises. Further problems include the possible existence of *gaps* in an expert's knowledge and the correctness of an expert's knowledge [63].

Furthermore expert systems do not have mechanisms to deal with any changes in their decision making environment — they cannot adapt and learn from changes in their operating environment. Thus the *maintenance of knowledge* in expert systems is also time consuming and expensive.

Due to these problems *inductive* techniques, including neural networks and genetic algorithms, have recently gained popularity. In expert systems, the decision boundaries — the bounds used to make particular decisions — are specified by a domain expert, while in inductive systems these decision boundaries are *learned* [127]. Changes in the operating environment cause the decision boundaries to be shifted or changed. Inductive systems can detect and adapt to these changes.

1.2 Intelligent Hybrid Systems

There is now a full repertoire of intelligent techniques, inductive and non-inductive, that can be applied for automating decision making. These techniques range from expert systems at the symbolic end to neural networks and genetic algorithms at the sub-symbolic end (see figure 1.2). However not all techniques are equally applicable to all types of decision making tasks. Each intelligent technique has particular strengths and weaknesses. The main motivation for creating intelligent hybrid systems is to combine the computational properties of different techniques to create superior reasoning systems [48].

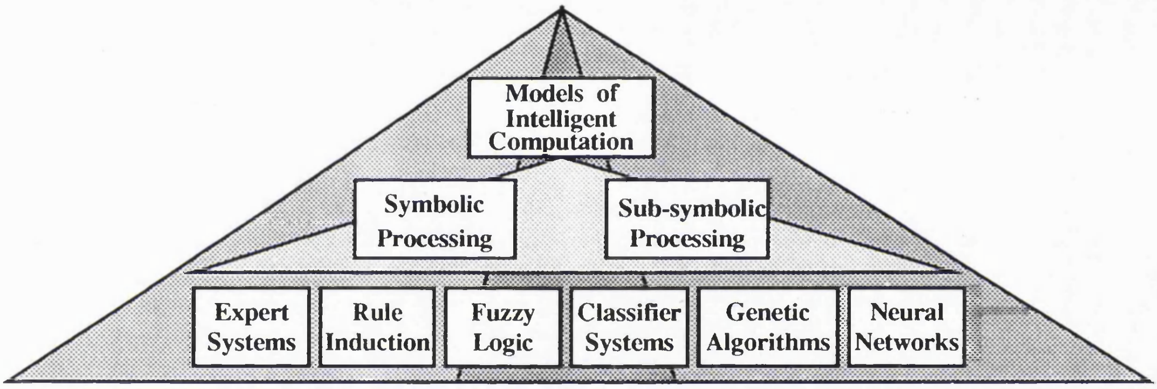


Figure 1.2: A Spectrum of Intelligent Computation

For example, expert systems and fuzzy systems require the explicit coding of knowledge by an expert. In contrast, neural networks and genetic algorithms are able to learn to perform tasks they model directly from domain data. While expert systems, rule induction and fuzzy systems provide clear explanations of their reasoning process, neural networks do not. Expert systems and rule induction methods have difficulties in dealing with imprecise data while fuzzy systems, neural networks and genetic algorithms are exceedingly good in dealing with such data.

There are considerable advantages in combining different intelligent techniques for two principal reasons (see Chapter 3 for further details).

Firstly, *Technique Enhancement*: integrating different intelligent techniques can help to overcome the weaknesses of individual techniques. For example by creating hybrids of fuzzy logic and neural networks, one can combine the ability of fuzzy systems to deal with imprecise data, and the ability to generate explanations with a neural network's ability to learn the fuzzy decision making rules.

The second argument for creating hybrids is the *Multiplicity of Application subdomains*. Many complex domains have many different component problems each of which may require different types of intelligent processing. If there is a complex application which has two distinct sub-problems, say a 'low-level' signal processing task and a serial reasoning task, then a neural network and an expert system respectively can be used for solving these two separate tasks.

Recently there have been many applications of hybrid systems in areas such as industrial control [129], veterinary diagnosis [109], legal reasoning [103] and cognitive modelling [6]. However currently there is no coherent method to make qualitative assessments between different hybrid systems and there are no guidelines for their development. In Chapter 3 we present a novel classification scheme for hybrid systems which provides a framework for qualitative comparisons and takes into account factors such as functionality, processing architecture and communication requirements. In Chapter 9 we present a sketch of a methodology for hybrid systems development based on concepts introduced in the classification scheme.

1.3 Knowledge Based Financial Decision Making

There is currently an upsurge of interest in applying AI techniques for financial decision making [122], [99], [65]. A significant number of these studies have concentrated on financial trading and portfolio management. Further, there have been several applications of AI techniques in mortgage evaluation, credit authorisation and marketing of financial products. All of these applications are in areas where non-programmed decision making is employed.

Trading in financial markets is a complex decision making activity. Principally, there are three types of decisions to be made: the decision to buy securities, the decision of how long to hold these securities, and the decision to sell them. Market analysts and traders use a wide variety of techniques and tools to help them make these decisions. There are two main types of knowledge that market professionals use for trading: fundamental analysis knowledge [80] and technical analysis knowledge [38].

1.3.1 Fundamental Analysis Knowledge

In the fundamental analysis approach to financial trading, an attempt is made to calculate the 'intrinsic' value of a financial instrument by using information published in sources such as company annual reports, and by making assessments on future supply and demand factors [80]. For example, when analysing individual company stocks, the future earnings can be seen as dependent on factors such as the company debt proportion, the value of assets, and the market for its products. A commonly used method in choosing stocks for investment is the examination of the price/earnings (P/E) ratios of stocks [80]. Fundamental analysts believe that a low price/earnings ratio is an indication of a company which is 'healthy', and is likely to be a good performing stock.

The prediction of commodities and currencies using fundamental analysis also follows the same approach of modelling factors of supply and demand. For example, in a forecasting model of the Yen/Dollar exchange rate the following variables may be included: the amount of foreign reserves, balance of payments, overseas investment, industrial production and interest rates. In a fundamental model to forecast cocoa prices, variables such as weather prospects, fertiliser price changes, past demand for cocoa and anticipated future demand for cocoa may be used. Linear regression is by far the most common method for modelling these relationships in fundamental analysis models [119].

1.3.2 Technical Analysis Knowledge

Technical analysts study the *dynamics of the market* rather than the factors that affect the supply and demand for financial instruments [13],[111]. The philosophy behind this approach is that all demand and supply dynamics are *reflected* in market indicators such as price, volume, and open interest, and the extraction of patterns from these movements alone is sufficient for making trading decisions.

The majority of the technical analysis techniques can be categorised into two broad categories, price pattern recognition and trend-following [119]. Practitioners of price pattern recognition, popularly known as 'chartists', claim that they can predict the future movement of financial markets by identifying simple geometric formations such as triangles, rectangles, and 'head-and-shoulder' patterns in price movements [38]. It is claimed for example that if the market 'breaks out' of a head-and-shoulder pattern, then a market will rally in the direction of the breakout [119]. There is a strong element of subjective judgment in interpreting these pattern formations which can require a long training period, and people trained in different 'schools' may arrive at different interpretations of the same data [119].

The trend-following form of technical analysis involves tracking the movements of *trading indicators* [67]. Indicators are typically transformations of the original price series such as moving averages. These indicators are combined to construct rules for making trading decisions. Example of such rules are: 'if the 10 day average becomes larger than the 20 day average then BUY (the price is likely to move up)', 'If the price is *declining* and open interest is *increasing* and volume is *increasing* then SELL (the market is likely to fall)', and 'If open interest is *high*, volume is *low* and volatility is *high* then BUY (market is likely to rise)'.

The linguistic categories in these rules such as 'high', 'low', 'increasing' do not have precise meanings but are typically defined subjectively by domain experts. The use of subjective linguistic categories is common in many areas of expertise [56], and this imposes heavy demands on expert assistance in the development of systems to automate non-programmed decision making tasks. In Chapter 5 we address this issue by introducing an automated clustering based method to produce linguistic descriptions from domain data.

Another issue that needs to be addressed in automating financial decision making is the dynamic nature of the operating environment. As with many complex decision making environments the arena of financial markets is one that is constantly changing and evolving and usually particular trading methods have only limited life-spans. Any decision making approach that is static would not survive in the long-run, and it is therefore vital to have mechanisms to generate new trading rules, to constantly monitor them, and to adapt them to new trading conditions.

1.4 Aims and Motivations

The principal aim of this research is to design, implement and evaluate intelligent hybrid systems for financial decision making. The domain of financial trading is chosen because of several reasons. These are:

- availability of objective criteria for performance evaluation

- availability of domain data
- access to domain expertise
- and the complexity of decision making process.

Although the effectiveness of our approach is demonstrated in financial trading decision making, it is sufficiently general to be applied to other areas of financial decision making such as loan evaluation and is also potentially applicable in areas such as medical diagnosis and urban planning.

There are several goals that the techniques designed and implemented in this research aims to satisfy:

Learn the Decision Making Knowledge: The system should not rely totally on elicited expert knowledge. It should be able to induce decision making knowledge from domain data.

Brittleness: The decision system should be able to handle many types of data. In particular, it should be able to cope with noisy data which is characteristic of much real-world data.

Explanation of the reasoning process: As decision makers typically need to defend their decisions, the reasoning process underlying machine generated decisions must be understood. In some areas such as loan granting, for example, providing explanations of the reasoning process can be a legal requirement [60].

Accessibility to Decision Makers: It is important that decision makers can easily understand the final results and recommendations of the system. Ideally, therefore, the system should convey its results in a format closer to natural language than an obscure mathematical format.

Model changes by Decision Makers: A decision maker should be able to modify any machine generated knowledge, and also add new knowledge. This is important in increasing the effectiveness of decision making in many domains [24],[23].

Adapt to changes in the decision making environment: If there is a change in the decision making environment, the system should be able to detect such changes and adapt accordingly.

In the course of this research it was realised that the above requirements could not be satisfied by the use of a single intelligent technique. Therefore a hybrid approach combining several intelligent techniques was developed and its effectiveness successfully demonstrated.

A central contribution of this thesis is the development of a theoretical framework to view hybrid systems. A classification scheme for hybrid systems which takes into account functionality and processing architecture is developed. This allows a qualitative evaluation of existing hybrid systems and provides a frame of reference to relate this research to other work. By using this framework, an attempt is also made to sketch a methodology of intelligent hybrid systems development.

The implementation of the above concepts is done in a program called INTENT (**IN**telligen**T** **DE**cisio**N** Suppor**T** Hybrid System). The INTENT system consists of six main modules: the Expert System, the Fuzzy System, the Genetic Algorithms, the Neural Networks, Decision Combination and the Decision Evaluation system.

- The **Expert System** module stores and manipulates trading knowledge elicited from an expert trader. A production rule format is used to represent and activate the technical analysis trading knowledge. This module also contains a clustering mechanism which converts raw data into a *symbolic* format that corresponds to the expert descriptions.
- The **Fuzzy System** module contains an algorithm to derive fuzzy membership functions automatically from raw domain data. This is an extension of the method used for converting raw data into symbolic descriptions. Once the fuzzy memberships are obtained, fuzzy rules obtained from expert traders are used to infer trading decisions. The fuzzy rules are represented using the same syntactic structure as the production rules used in the expert system.
- The **Genetic Algorithm** module induces decision making rules in two modes: *symbolic* and *fuzzy*. In the symbolic mode it will induce rules and perform inferences using symbolic data analogous to the operation of an expert system. In the fuzzy mode it will induce rules and perform fuzzy inferences using fuzzified data analogous to the operation of a fuzzy system. The genetically induced rules have the same syntactic structure as the rules in expert and fuzzy systems thus allowing a range of feedback mechanisms between these modules.
- The **Neural Network** module learns decision patterns using either symbolic data or fuzzified data. Selection of variables for the neural network is done by the genetic algorithm. That is, variables that feature in highly 'fit' decision rules induced by the genetic algorithm are used as inputs for the neural networks.

- The **Decision Combination** module aggregates the decisions of all the different decision modules and produces composite decisions. The mechanisms in this module are motivated by recent research suggesting the superiority of consensus decision making over individual models and approaches.
- The **Decision Evaluation** module evaluates the decisions made by the different modules using domain specific performance evaluation criteria. In the case of financial trading decisions, several criteria are used including the percentage of correct trading decisions, average gain per trade and the volatility of returns. Similarly, if the system is used for medical decision making, then this module will include measurements of the effects of decisions in terms of patient recovery times and infection rates.

The contribution of this thesis can be stated as follows:

- A classification scheme for Intelligent Hybrid Systems is proposed. This classification scheme takes into account factors such as functionality, processing architecture and communication requirements. This provides a coherent framework to view existing hybrid systems and allows the possibility of qualitative comparisons of this research.
- A sketch of a methodology for the development of hybrid systems is presented. This is based on our classification scheme and attempts to provide developers of hybrid systems with a set of guidelines to help them make choices in using different hybrid architectures.
- The demonstration of the applicability of using genetic algorithms to induce fuzzy models. This method provides an automated way to construct fuzzy rule bases, which is otherwise a time consuming, trial and error process, typically done by a domain expert.
- A new data pre-processing method for converting raw data into *symbolic* descriptions has been developed and its effectiveness demonstrated. This provides an automated mechanism for producing linguistic descriptions of data (e.g. low, medium, high) using raw domain data. This clustering based method is evaluated by producing symbolic descriptions of trading data which corresponds to descriptions used by trading experts.
- The symbolic data-pre-processing method is extended to pre-process the raw data into *fuzzy* descriptions. The fuzzy memberships derived by this process

have also been demonstrated using trading data and subsequently verified by domain experts.

- The demonstration of the advantages of using production rules to communicate knowledge between expert systems, fuzzy systems and genetic algorithms. The benefits include: the comprehensibility of the decision models, the allowance of judgmental revisions to models, and the augmentation of machine generated knowledge with expert specified knowledge.
- A multiple-model approach for producing decisions by combining different intelligent decision making modules is developed. It is shown that this method produces superior results than in the case of using individual techniques. These results have confirmed the usefulness of multiple-model approach in intelligent systems which have been used elsewhere in applications such as protein structure prediction [130].

1.5 Thesis Organisation

The remainder of this thesis is organised into nine chapters, covering: a review of intelligent decision systems for financial trading, an examination of the arguments for intelligent hybrid systems and a new classification scheme for hybrid systems, the design of the INTENT system, the expert system and decision evaluation, the fuzzy system module, genetic algorithms, neural networks, decision combination and evaluation, and conclusions.

- Chapter 2 begins by examining the different types of trading decision making knowledge. Recent efforts in using intelligent techniques for financial trading decision making are then reviewed. The use of expert systems, rule induction, fuzzy systems, genetic algorithms and neural networks for financial trading are examined and their relative merits are discussed.
- Chapter 3 examines the arguments for intelligent hybrid systems and introduces a new classification scheme for these systems. It is argued that the combination of different intelligent techniques can both help to overcome the weaknesses of individual techniques and also help to solve complex problems that would otherwise be unsolvable.

- Chapter 4 discusses the design considerations of the INTENT system. The rationale for adopting the particular hybrid architecture is discussed and the functioning of each module within this framework is described.
- Chapter 5 describes the expert system and decision evaluation. It examines knowledge elicitation from experts, the representation of the expert knowledge, the use of the clustering method to derive linguistic descriptions of knowledge and the design of the production rule system. Details of the validation of the linguistic descriptions by the domain expert are also presented. This chapter also discusses the criteria used for evaluating the effectiveness of decisions. The decision evaluation module applies several industry standard criteria to evaluate trading decisions generated by the expert system and other modules in INTENT. Finally, simulated trading results for the British Pound and the German Deutschmark are presented.
- Chapter 6 describes the Fuzzy System module. It examines the use of clustering techniques to obtain fuzzy membership functions, the representation of fuzzy rules, and fuzzy reasoning methodologies. The validation of the fuzzy membership functions by the domain expert is also discussed. Simulated trading results for the British Pound and the German Deutschmark are presented.
- Chapter 7 describes the Genetic Algorithm modules. It examines the use of genetic algorithms to induce rules, the representation of these rules, the use of clustered and fuzzified data for rule induction, and the feedback mechanisms to the expert and fuzzy knowledge bases. The genetic algorithms are tested on the same financial market data used to test the expert and fuzzy systems. The advantages of the hybrid genetic algorithm approach in terms of explanation capabilities, and the assessment induced genetic rules by domain experts with respect to their quality of decision making is also discussed in detail.
- Chapter 8 describes the Neural Networks. It examines the choice of neural network architectures and parameters, and the use of symbolic and fuzzy data. This chapter also describes the variable selection mechanism using genetically induced rules. The results of the neural network module are then discussed with respect to other approaches. Expert evaluation of the results and the limitations of the approach in terms of explanation capabilities are also discussed.
- Chapter 9 describes further integration of the modules and overall evaluation of performance. A multiple-model approach for making decisions, by combining decisions of all different modules is presented. The method and results of

an empirical investigation into the effectiveness of actual human trader performance is also reported. The overall performance of all the decision modules is then compared with published trading systems results and the performance of the expert human trader. Finally a sketch of a methodology of hybrid systems development is presented and its strengths and weaknesses are evaluated.

- Chapter 10 presents the conclusions drawn from the work and discusses future research.

Chapter 2

Intelligent Systems for Financial Trading

This chapter presents: an introduction and evaluation of the different intelligent techniques used for financial decision making, and a survey of the application of these systems in financial trading.

2.1 Introduction

This section reviews the applications of intelligent techniques in financial trading. Techniques include expert systems, rule induction systems, fuzzy systems, neural networks and genetic algorithms. In each section we briefly introduce the technique and then review the financial trading applications. We begin by examining expert systems which is a well understood and widely applied technique in financial decision making.

2.2 Expert Systems

The roots of Expert Systems can be traced to models of human expertise first developed in the early 1970's [17]. Underlying many expert systems are knowledge bases that are typically represented as *production systems*. A production-system is a set of logically independent condition-action rules, or *productions*. A production, is an *if-then* pair: *if* the condition is satisfied, *then* the action is executed. Currently there are many commercially successful expert systems in domains ranging from banking to manufacturing to education [36] [55][125]. We now briefly review the main concepts in expert systems.

- **Knowledge acquisition**

  All expert systems embody *knowledge* either obtained from human experts or other knowledge sources such as books. The process of acquiring the knowledge from an expert — knowledge acquisition — typically involves a series of interviews and the careful recording of observations when the expert is performing tasks [68].

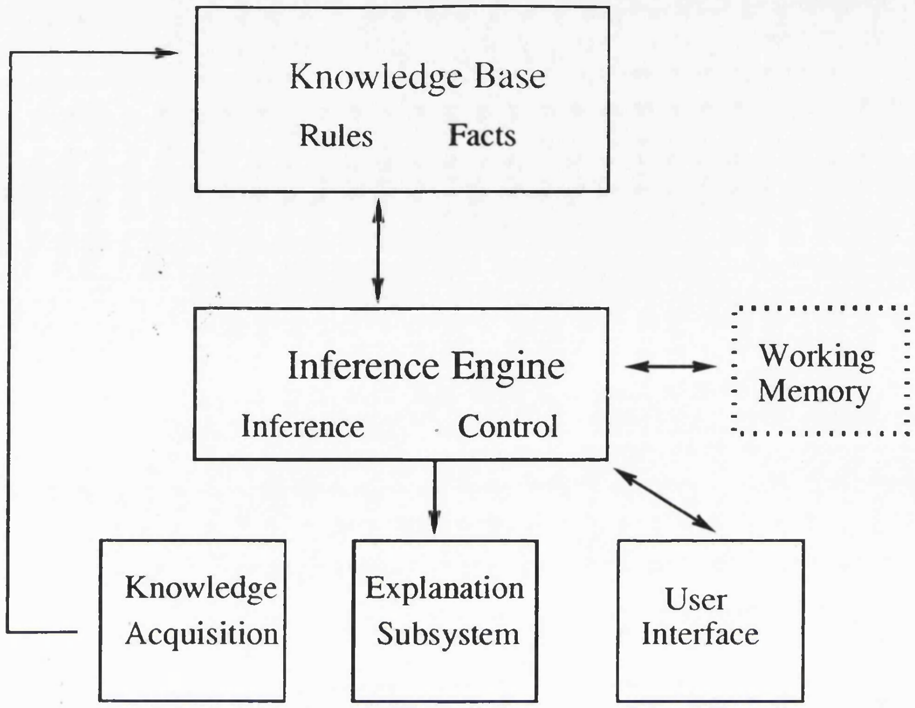


Figure 2.1: Expert System Organisation (Adapted from Harmon and King [1985])

- **Knowledge representation**

  Once the knowledge is acquired from an expert, it is *represented* in a structured form that enables reasoning operations to be performed. A wide range of knowledge representation schemes have been developed including rules, frames and semantic networks, but production rules are by far the most common format.

- **The Interpreter and Control Mechanisms**

  The *inference engine* or interpreter is the control mechanism that drives the reasoning process. When the conditions of the rules are matched against inputs from the operating environment, the interpreter selects and executes the appropriate actions of the rules. If more than one rule is applicable for a given input, a 'conflict resolution' mechanism is used to select a single rule [63]. A commonly used conflict resolution strategy is the use of *specificity* [63] where a rule which has more matching conditions is chosen in preference of rules which have lesser matching conditions.

- **Explanation**

  Most expert systems have rudimentary explanation capabilities which provide 'traces' of rule firings. The provision of such explanations, is crucial both for user acceptance of results and also to aid refining the expert system knowledge base. Some systems [112] have simple natural language interfaces which allow users to interrogate the reasoning process by issuing simple commands. A limitation of producing rule traces is that they can be verbose and sometimes difficult to understand [63].

2.2.1 Expert Systems and Financial Trading

The majority of expert systems used for financial trading use production rules as the chosen knowledge representation scheme [65],[78], [40]. There are two main advantages in using rules in this domain: (1) experts can easily specify and understand the rules and (2) explanations of the reasoning process can be given to justify any investment decisions made.

Lundberg and Barna [78] provide one of most comprehensive examples of the use of expert systems for trading commodity markets. Their system which is used for trading soybeans is based solely on the use of technical indicators. The forty relevant technical indicators and rules were elicited from a single trading expert. Protocol analysis [39] was used to facilitate this knowledge elicitation process.

The technical analysis knowledge Lundberg and Barna used can be separated into three categories: moving averages (averages of price), oscillators (indicators tracking the 'overbought' status of the market) and single step changes (large one day movements in price or volume). The reasoning is performed in three stages, each stage comprising a higher level of abstraction. At the first level the rules process the raw market data to detect short-term up or down trends. The second level tries to infer the longer term movement of the market, and the final level provides the user with advice on buying and selling with an explanation of the associated risks. However, Lundberg and Barna's study, characteristic of many of the expert system studies in this area, provides little detail on its performance. No results on historical data are provided and they do not comment on the stability of the rules. Other expert systems which use technical analysis knowledge for making trading decisions can be found in [131] [53].

Kandt & Yuenger [65] describe an expert system which uses both technical and fundamental analysis. The fundamental analysis part of their system is based on theories of the business cycle — the expansion and contraction of the economy and its effect on interest rates. A chain of reasoning in the system proceeds as follows: business contraction reduces the demand for funds and consequently interest rates fall. The lower cost of borrowing then encourages consumers to spend and a resurgence of business investment occurs. This lifts the economy out of a recession into expansion; the stock market rises. A booming economy, however, places a great demand on funds which cause interest rates to rise. When the interest rates rise, business and consumers are discouraged from spending. This results in a slowing economy; company earnings will fall, and the stock market falls.

In addition to the above fundamental analysis rules, Kandt & Yuenger also use a range of technical rules which include 10 day and 28 day moving averages.

In this study too, however, there are no details of any performance tests on historical or live data. Kandt & Yuenger also do not discuss the issue of knowledge maintenance — the modification of rules by domain experts to account for any changes in market conditions.

2.3 Rule & Tree Induction Systems

Rule induction systems generate decision rules from domain data. The induction process is a process where a given data set is subdivided with respect to a set of *features* (and logical operations on these features). Tree structures from tree induction systems can be converted easily into rules [96], which can then in turn be used as rule bases in expert systems.

A key advantage that rule induction offers over other inductive methods is the transparency of its rules. As they are in a form that can be easily understood by humans they are more likely to be accepted than numeric or statistical explanations given by a model [127]. Rule induction systems have had many successes in a variety of classification tasks, including the diagnosis of soybean diseases [85] and fault diagnosis [77].

A critical part of tree induction systems is the evaluation of features for building classification trees. Typical systems such as Quinlan's ID3 system [96] use measures based on entropy to select features. This involves an assessment of the degree to which a particular feature will be able to split the data set most effectively [127]. The ID3 algorithm iteratively subdivides a given data set with respect to the given features until all data items are correctly classified. Sometimes this process results in trees that are overfitted to the training data, so measures to stop the tree generation process at appropriate points have to be adopted for real world practical learning tasks.

2.3.1 Rule Induction and Financial Trading

Braun and Chandler [19] used the ID3 algorithm to learn the weekly market movements of Dow Jones industrial average using data from 20 technical analysis variables including the 10 day moving average of the Dow Jones, put-call ratios (options market related variables) and the volume of transactions. The task was to classify the movements of the market in the following week — whether it would be higher, lower, or at the same level as the previous week.

The program induced rules from data over a period of 54 weeks, and the generated rules were then tested on unseen data over a period of another 54 weeks. The decision trees that were generated had an average length of 22 nodes and were able to predict the market 64.4% of the time. The system performed slightly better than the human stock market expert (predictions were 60.2% correct) who had helped to identify the relevant variables to be used for the rule induction system.

However it must be noted that the test sample used was quite small (1 year) and it is difficult to generalise on the effectiveness of this approach from this experiment alone. The trading systems literature suggests that a testing period of approximately 5 years is needed to have sufficient confidence in the results [10]. This is the reason why our INTENT system (see Chapters 4–9) is tested over a five year period.

Another difficulty with Braun and Chandler's results is the use of single partition of the data. That is, they separate all the data into a training set and test set — induction of trees is done using the training data and then they are applied to the test data. However it is not clear how many times the parameters of the algorithm were adjusted based on the performance on the training data as well as the test data, thus 'indirectly' training on the test data [127]. A more credible approach would have been to partition the data into three sets - training, validation and testing, where the algorithm parameters are adjusted based on the performance of the training and validation sets and the where the testing data is used only after all parameters are fixed.

2.4 Fuzzy Systems

Fuzzy Logic provides a means for reasoning with imprecise linguistic concepts such as 'small', 'big', 'young', 'old', 'high' or 'low'. The central concept behind this form of information processing is the notion of a fuzzy set which was first developed by Zadeh in 1965 [132]. Fuzzy sets can be contrasted with conventional Boolean sets which have 'crisp' or 'clean' demarcations. Instead their *membership* to a particular set is a *gradation function*. Once the membership function is defined, fuzzy logic operations can be applied on these data representations. Recently this approach of decision making has attracted attention in the area of control engineering, handwriting recognition and loan evaluation [83].

There are four main features of a fuzzy reasoning system.

- **Defining membership functions** involves the conversion of a given set of values into fuzzy membership functions which corresponds to a given linguistic variable (e.g. high). Typically these linguistic variables have a range between 0 and 1. The shapes of these functions are usually specified by a domain expert. Data that have been converted into fuzzy membership functions are referred to as 'fuzzified' data.

- **Fuzzy inference rules** specify the relationships among the fuzzy variables. The fuzzy rules are in the production rule (If/Then) format. They too are usually specified by a domain expert.

  An example rule is: IF interest rates are *high* AND industrial production is *low* THEN the market falls

- **Fuzzy reasoning** is the process of deriving conclusions from a given set of fuzzy inference rules acting on fuzzified data. In contrast to conventional expert systems where only one rule is activated in response to its antecedents being true, in fuzzy inferencing all rules whose antecedents are true or partially true will contribute to the final result. Because of this result aggregation process, collections of fuzzy rules (fuzzy rule bases) can contain partially contradictory rules and facilitates an inferencing process where the hypotheses of all rules are considered, even if the relevance of a particular rule in a given situation is small.

- **Defuzzification** is the process of converting the result inferred by fuzzy reasoning system back into the range of the original data values. There are several popular methods for defuzzification, the 'centre of area' [11] and 'mean of maximum' [11] being two of the most widely used.

2.4.1 Fuzzy Systems and Financial Trading

At present there are very few reported applications of fuzzy logic in financial trading. Yuize et al [52] describe a fuzzy reasoning system for foreign exchange trading (the Yen against the Dollar). This system is primarily based on fundamental analysis, and is unusual compared with other systems in that it makes inferences based on economic news events that affect the currency markets.

The system is divided into several levels of processing. The lowest level processes the numerical data (interest rates, stock prices etc.) of the US and Japan into fuzzy values (*very low, low, more or less high, high, very high*). The shapes and ranges of the fuzzy membership functions are specified by domain experts. At the next level of the system, *news reports* are converted into a fuzzy format for reasoning. This conversion process is done manually by domain experts. For example the news report,

> "The US Secretary of State announced in the afternoon that economic growth, fuelled by moderate inflation, is expected to continue and that there is a conservative attitude towards any increases in official interest rates."

can be converted into the following fuzzy variables,

official discount rate = 0.6/no pressure, 0.4/some pressure to rise

US. price trend = 0.3/low, 0.7/reasonable level

(0.6/no pressure indicates that there is a 0.6 confidence for no pressure to increase the official discount rate)

Both the fuzzified news reports and price data are used by the fuzzy rules to make the final predictions. These rules are also elicited from domain experts. Examples of such rules are:

Rule 1:

IF the US official discount rate has a high pressure to rise, AND the financial policy of the federal reserve board is tight THEN official discount rate will rise

Rule 2:

If the US official discount rate has high pressure to fall, AND the financial policy of the Federal Reserve Board is loose, THEN the US official discount rate has a high pressure to fall

Yuize et al [52] say that the system is being used to forecast currencies daily, but still on a trial basis. They do not provide results of any trading recommendations of the system.

2.5 Neural Networks

Neural networks [7], [12] are a class of computing techniques which have some similarities to the function of nerve cells in the brain. They are composed of many parallel, interconnected computing units. Each of these performs a few simple computations and communicates the results to its neighbouring units. Neural networks can learn to recognise patterns by a process of iterative training and have been used successfully in a variety of classification, prediction and optimisation problems [29].

There are four main considerations in the design and execution of a neural network [99]:

- **Network Architecture** specifies how processing units are connected to each other. Processing units are typically organised into several layers. In each layer the units can be connected to other processing units in one or several other layers.
- **Learning algorithm** determines how the connection strength (weights) between units is changed to learn the relationships in the training data sets. The Backpropagation algorithm [105] and its variants are a popular choice of learning algorithms.
- **Threshold functions** are applied at each processing unit after all the inputs to that unit have been summed. Typically, sigmoid or hyperbolic tangent functions

are used. There is some empirical evidence to suggest that hyperbolic tangent functions can yield faster convergence compared with sigmoid functions [74].

- **Gradient descent parameters** determine the rate at which the network will converge to a possible solution. The momentum and learning rate are two main parameters which influence the magnitude of weight change in the network [99].

Neural networks provide a relatively easy way to model and forecast non-linear systems [73], [126]. This gives them an advantage over many current statistical modelling methods used in economics and finance which are primarily linear. They are also very effective in learning patterns in data that is noisy, incomplete and which even has contradictory examples [7]. The ability to handle such imprecise data makes them very effective in financial information processing.

2.5.1 Neural Networks and Financial Trading

Recently there has been an explosion of interest in applying neural networks for trading in financial markets [118], [28], [108], [99]. While the majority of these systems are at an experimental stage, a few are reported to be trading on a daily basis [99], [28].

Refenes [99], for example, describes a neural system for trading bonds. The aim of the system is to make monthly asset allocation decisions between the bonds of six different countries (Japan, UK, Germany, USA, France and Canada) in a way that will maximise the total return. There are six neural networks corresponding to each country, each of which makes a prediction for returns of the bonds for that given country. Each neural network is trained using fundamental analysis data such as interest rates, oil prices and the ratio of precious to non-precious metals. The distribution of funds among the markets is proportional to the predicted returns (market A expecting twice the return of market B will be allocated twice as much as B). There are also global constraints on the allocation of assets; the maximum allowed in Japan is 50%, UK 30%, Germany 30%, Canada 20%, France 20% and Australia 10%. The system has been tested with data between 1989 to 1992 and the system is claimed to achieve returns of 125% while a benchmark similar to the JP Morgan index achieves a return of 34% in the same period.

Kimoto et al [118] have used a back propagation algorithm to predict the movements of the Tokyo Stock exchange index, TOPIX. The network was trained on four years of previous data which included technical indicators such as interest rates, foreign exchange levels, and turnover levels. The output of the network was a weekly buy or sell decision. The trading recommendations of the network were tested over a period of three years from January 1987 to December 1989. If the TOPIX index was considered to have a value of 1.0 in January 1987, at the end of 1989 the neural network would have generated a value of 1.98 compared to 1.67 if a buy-and-hold strategy had been used.

A general concern about neural networks in financial trading is the issue of explanation. Due to their poor explanation capabilities some financial decision makers are sceptical of their use because decisions often need to be defended to shareholders and others [60]. Further, as the decision makers cannot understand the decision process, this does not allow any judgmental revisions to the decision models (which is vital in rapidly changing environments). We return to this issue of explanation in decision models in Chapter 4.

2.6 Genetic Algorithms

Genetic Algorithms (GAs) [30], [32], [59] are efficient probabilistic search algorithms inspired by the mechanisms of biological evolution. GAs have produced very good solutions for complex optimisation problems that have large numbers of parameters. Areas where these have been applied include imaging, VLSI circuit layout, gas pipeline control, and job shop scheduling [30], [32].

The main idea behind a genetic algorithm is to start with a population of solutions to a problem, and then attempt to produce new generations of solutions that are better than the previous ones. GAs operate through a simple cycle consisting of the following stages: population creation, reproduction, chromosome alteration and evaluation (see figure 2.2).

Classifier systems [59] are machine learning systems that can be viewed as a combination of production rules and genetic algorithms. In classifier systems, the classifier rules have the same production rule format found in expert systems. These classifier rules are modified by a genetic algorithm which uses crossover and mutation

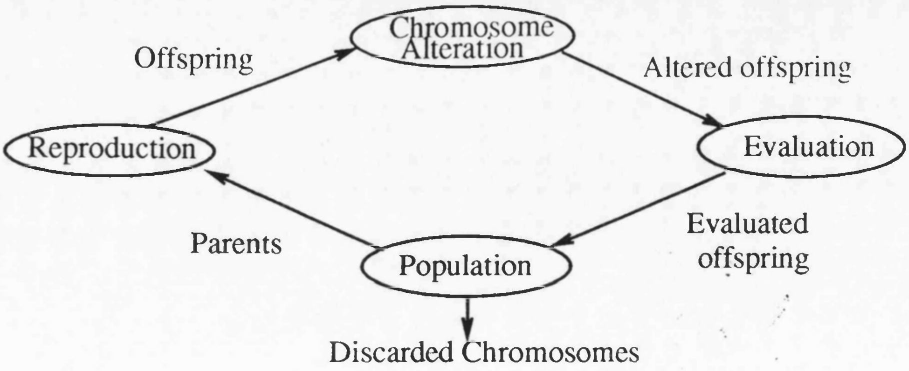


Figure 2.2: The Genetic Algorithm Cycle

operators. In a classifier system, rules which contribute towards solving the problem at hand are 'rewarded', while rules which are unsuccessful are penalised.

There are four main considerations in the design and execution of a genetic algorithm.

- **Chromosome representation**

  Solutions to the current problem are encoded in a structure called a 'chromosome'. Typical chromosome representations use either binary or real-valued fixed-length strings.

- **Chromosome evaluation**

  This involves assessing how 'fit' or good a given solution is at solving the problem at hand. This aspect of the algorithm is always domain specific. For example in the case of financial trading decisions, a chromosome which has a higher correct percentage of trades or has a lower volatility of returns will have a higher fitness than a chromosome which makes less correct trades and has a higher volatility of returns.

- **Chromosome selection**

  This is the process of selecting solutions from the solution population for further breeding. The roulette wheel selection procedure [30] is a commonly used procedure, where the chances of reproduction is directly proportional to the fitness of the solutions.

- **Chromosome alteration**

  There are two main methods for changing the structure of the chromosomes — *crossover* and *mutation*. The *crossover* operator is analogous to sexual reproduction in nature where new chromosomes are created by the swapping of genetic material.

  For example, consider the crossover operation in figure 2.3 where elements of strings A1 and A2 are swapped to produce the new strings, B1 and B2

  The point at which the crossover occurs in the strings (in this example, the fourth element of the string) is chosen randomly.

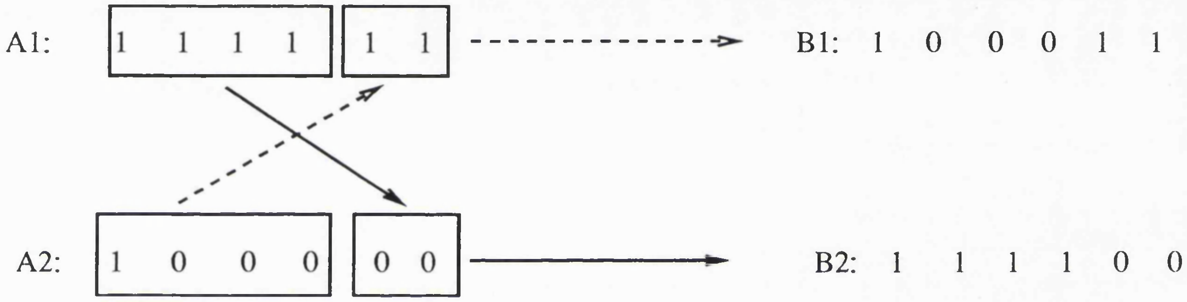


Figure 2.3: The Crossover operation

  The other genetic operator, *mutation*, causes small random changes in the elements of the strings. This causes the algorithm to introduce variation in the search process. In figure 2.4, the mutation operator is applied at the second element of the string. In the second example, it is applied at the fourth element of the string.

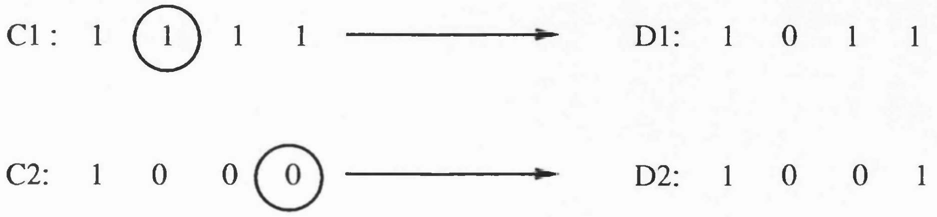


Figure 2.4: The Mutation operation

Recently there have been several new developments in the use of novel representation schemes in genetic algorithms. The *genetic programming* paradigm where symbolic structures such as LISP tree structures are used for representing problems

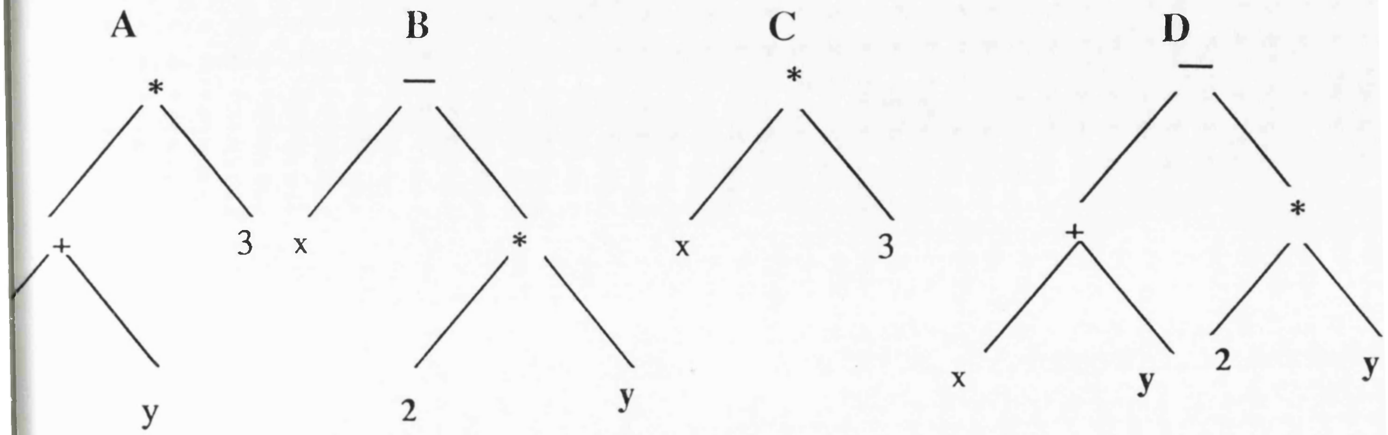


Figure 2.5: Inducing Equations using Genetic Algorithms

is now being applied in problems as diverse as economic forecasting and control problems [72]. In the European Community funded PAPAGENA project, tree structured genetic algorithms have been used to induce complex mathematical formulae [37]. This system devised by Haasdijk et al has been applied in credit evaluation tasks. We are not aware of any applications of this technique in financial trading.

Figure 2.5 shows the crossover operations in this system. The parent tree A represents $(x + y) * 3$ while the tree B represents $x - 2y$, and the resulting *child* trees are C and D.

Apart from novel representation schemes, recently there have also been significant developments in the parallelisation of genetic algorithms. Parallel genetic algorithms where distributed processing is used for implementing genetic algorithm models has two main approaches [79]. These are:

- The *island model* where several isolated sub-populations evolve in parallel periodically migrating their best individuals with the neighbouring sub-populations.
- The *fine grained* model (or neighbourhood model) where a single population evolves, each individual of which is placed in a cell of a planar grid. Selection and crossover are applied only between neighbouring individuals on the grid (according to a pre-defined negihbouring structure).

The UCL genetic algorithm programming environment GAME [44] provides a convenient method to implement parallel genetic algorithms over clusters of workstations and PCs. Such parallel implementations can speed up the execution of genetic algorithms considerably. As discussed in Chapter 7, the genetic algorithm component in our INTENT system is computationally very intensive and takes a very long time to produce good decision models. A genetic algorithm environment such as GAME can be potentially used to parallelise INTENT over a network of machines thus helping to increase its performance in terms of speed. However, due to time limitations we could not explore these possibilities as part of this thesis.

2.6.1 Genetic Algorithms and Financial Trading

One reported application of genetic algorithms is the creation of an 'artificial stock market' by Arthur et al at the Santa Fe institute [9]. In this project, classifier systems were used as 'artificial traders' to learn the dynamics of this artificial stock market. The aim of the classifier systems was to learn how to make the most amount of profit by trading — to buy when the price is low and to sell when the price is high.

The classifier rules were started with random settings and after about 1000 generations (run for about a day on a workstation) they had formed rules about trading. They generated rules such as to buy when 'the price earnings multiple is low', and they also formed simple rules about price trend movements.

An interesting aspect of the simulation is that the classifier rules not only formed rules from market data such as prices and moving averages, but they also evolved rules according to the actions of other artificial traders (other classifiers). It appeared that if a large number of artificial traders believed the fact that 'prices will go up when the market index reaches the 2100 level', this will cause stock prices to rebound at the 2100 price level — it will be a self-reinforcing belief. Similarly, the competing belief, 'prices will fall if the market index reaches 2100' might turn out to be true if it attracted enough followers. These results provide some justification for a variety of technical analysis methods which are widely believed to work because of their self-reinforcing nature.

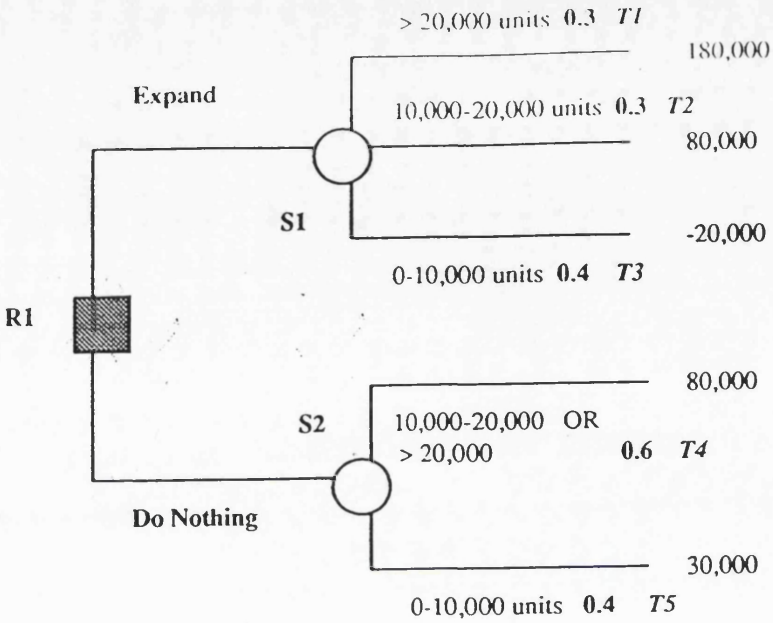


Figure 2.6: Decision Tree for financial planning

2.7 Decision Theoretic Methods and Probabilistic Reasoning

Decision theoretic models are frequently used to support a variety of financial decision making tasks including corporate planning and investment decision making [128], [20], [90]. A common method in decision analysis is the use of decision trees [20]. Miller [86] provides an example of the use of decision trees for making corporate investment decisions.

The decision tree in figure 2.6 represents the possible impacts of a decision on whether or not to expand the production of a new product, a toy called Victor Vole, by its producing company. As a new product, Victor Vole does not have a well established pattern of demand and can only be produced in limited quantities at a small production facility. If demand for Victor is at or above the average for a product of its type, production using the existing facilities will not be able to keep up with demand. Production capacity can be expanded greatly, however, by automating much of the assembly of Victor Vole. The cost of expanding the production facilities is \$50,000. Demand can be characterised as falling into one of three categories 0-10,000 units, 10,000-20,000 units, more than 20,000 units. The first category has a subjective probability of 40 percent, the second category 30 percent and the final category 30 percent.

If the level of demand is less than 10,000, then expansion is not necessary to meet demand and the gross maximum profit generated is \$30,000. When the level of demand is greater than 10,000 units, expansion is necessary if the maximum profits are to be obtained. Without expansion, profits of \$80,000 are all that can be achieved if demand is greater than 10,000 units. With expansion, gross profits are \$230,000 for demand greater than 20,000 units and \$130,000 for demand between 10,000-20,000 units, but remain at \$30,000 for a demand in the range of 0-10,000 units. Considering the expansion cost of \$50,000 gives net profits of \$180,000, \$80,000 and –\$20,000 for the three options respectively. These values are represented alongside the decision paths in figure 2.6.

The mechanism for arriving at the particular decision involves the calculation of the expected monetary values for each of the chance nodes (S1 and S2). At node S1, the expected value is computed as follows:

$$\begin{aligned} V[S1] &= P[T1]\ v[T1] + P[T2]\ v[T2] + P[T3]\ v[T3] \\ &= 0.3\ (180{,}000) + 0.3\ (80{,}000) + 0.4\ (-20{,}000) \\ &= 70{,}000 \end{aligned}$$

where v is a function that gives the value of the tree at a node and P is a function that gives the probability of the node being reached. Similarly, the expected value of node S2 is computed as follows:

$$\begin{aligned} V[S2] &= P[T4]\ v[T4] + P[T5]\ v[T5] \\ &= 0.6\ (80{,}000) + 0.4\ (30{,}000) = 60{,}000 \end{aligned}$$

The value of the root node, R1 is computed as:

$$v[R1] = \mathrm{Max}[\ [v[S1],\ v[S2]] = \mathrm{Max}\ [70000,\ 60000]= 70{,}000$$

which implies that the S1 branch or the expand option is the most profitable.

Similar decision tree methods are also used in other areas of financial decision making such as options trading [128]. However, in our literature survey we did not find any references of their use in foreign exchange trading. Further, we are not aware of any financial trading applications that process technical analysis knowledge using such decision trees.

We now turn to a completely different technique. Bayesian methods are a class of probabilistic reasoning that is used in a variety of expert systems [100]. Bayes' Theorem provides a method of computing the probability of a particular event given a set of observations.

The theorem states:

$$P(H_i|E) = \frac{P(E|H_i) * P(H_i)}{\sum_{n=1}^{k} P(E|H_n) * P(H_n)}$$

where, $P(H|E)$ = the probability that Hypothesis H_i is true given evidence E.

$P(E|H_i)$ = the probability that we will observe evidence E given that hypothesis I is true.

$P(H_i)$ = the a priori probability that hypothesis I is true in the absence of any specific evidence. These probabilities are called prior probabilities.

k = the number of possible hypotheses.

This type of reasoning has been deployed very successfully in the expert system PROSPECTOR which was used for mineral prospecting [101]. In this system, inferences on the likelihood of finding minerals are made with the Bayes formula using the prior probabilities of finding various minerals and the probabilities of a particular mineral being present given certain observed physical characteristics.

In our literature survey we have not come across the use of Bayesian reasoning for trading decision making.

In one part of our INTENT system, the genetic algorithm in the symbolic mode (see section 7.2.3), a simple probabilistic approach is used to compute the 'goodness' of the decision models. This is Packard's [93] genetic model fitness estimation procedure. In other parts of the INTENT system, as in the genetic algorithm in the fuzzy mode (see section 7.5), fuzzy reasoning is used for making trading decisions. A primary reason for adopting fuzzy descriptions and fuzzy reasoning is its inherent use of linguistic descriptions such as *low*, *medium* and *high*. This provides us with a mechanism to construct models that use the bf same linguistic descriptions that experts use to describe trading decisions. This in turn makes the validation of the

INTENT models by an expert trader much easier. Section 7.8 describes the validation of genetic-algorithm-induced decision models by an expert trader. The other main consideration for inducing fuzzy decision models was the availability of other case studies of using fuzzy systems for trading foreign exchange [52]. This allows us to make qualitative comparisons between the decision support mechanisms in the INTENT system with that of other fuzzy decision support systems used for financial trading.

2.8 Intelligent Hybrid Systems

As briefly mentioned in Chapter One, there are two primary reasons for using hybrid systems: (a) *Technique Enhancement*: integrating different intelligent techniques can help to overcome the weaknesses of individual techniques and (b) *Multiplicity of Application sub-domains*: using different intelligent techniques to handle different aspects of a complex problem. In Chapter Three we will examine these issues in detail and evaluate several hybrid systems. The next section briefly reviews financial trading applications of hybrid systems.

2.8.1 Intelligent Hybrid Systems and Financial Trading

There are very few reported applications of hybrid systems for financial trading. Bergerson & Wunsch [14] describe a commodity trading model where the neural network is used to learn the patterns of trading, and an expert system is used to manage the investments according to rules. The neural network used in this system, in contrast with other neural financial trading systems, is trained on data items that have been marked as profitable situations by an expert trader. In other words the network is emulating the decision processes of the expert rather than trying to extract a model from the raw domain data. The expert system contains encoded knowledge of risk management which enables one to limit losses from incorrect predictions and let profits accrue if correct predictions are made.

The system has been used to predict the S & P 500, a US stock market index, using training data from 1980–1988. The system was tested from 1989 to 1991, and Bergerson and Wunsch [14] claim that a capital growth of 660% would have resulted if the system had traded during the two test years.

An example of a genetic algorithm and neural network hybrid in financial prediction is a neural network used for trading in a simulated double auction market. A double auction is a type of trading institution which is used by, among others the New York Stock Exchange and the Chicago Board of Trade. In 1989, a tournament was organised by the Santa Fe institute where thirty computer programs traded with each other in a computerised double auction market [62]. Dallaway and Harvey's [62] entry for this tournament was a recurrent neural net which made decisions to buy or sell the 'tokens' in this artificial market. A genetic algorithm was used to evolve the connection strength parameters in the network, based on how well the network was performing over a series of trading periods. Simulated trading performance figures for this hybrid system are not given.

2.9 Conclusions

In this chapter we have reviewed the application of intelligent techniques in financial trading. These techniques have been applied in many of the major markets including government bonds, currencies and stock index futures. There are probably many other pieces of work on building systems for financial trading where similar experience exists but because commercial secrecy their experience has not been published.

One of the main points that emerges from the literature is that the majority of the systems use technical analysis knowledge. A possible explanation for this is that most systems discussed are for short term trading and the fact that technical analysis is biased towards such trading. An additional explanation may be the difficulties in obtaining appropriate data for fundamental analysis and the complexity of mechanisms needed to process such data. As much of fundamental analysis information is reported through news reports with associated subjective assessments, it is difficult to produce automated methods to construct decision models. Yuize's study [52] using fuzzy logic is the only attempt to use such news data, but here it is difficult to make an assessment of the method due to the lack of any performance results.

A major limitation of many of the studies is that they do not provide performance figures for either simulated or real trading. This is most probably due to the fact that many of the published studies relate to prototype systems.

For the purposes of selecting an optimum intelligent technique for trading, the literature also has drawbacks in that there are no comparisons of the different techniques using the *same data.* There are significant differences in the character of different markets with respect to variables such as volatility and trading volumes which makes the comparison of trading systems across markets very difficult.

Another concern with the literature is the possibility of researchers overfitting the decision models in order to obtain good results. The majority of the studies which detail results appear not to keep the test data strictly 'hidden', but instead appear to use the test set for model parameter adjustments as well. As Weiss [127] points out, this is a common error in many modelling exercises which tends to produce errors that are lower than the true error rate for that given sample. Ideally all parameters of the model should be adjusted using the training data and a validation data set and the test data should be presented to the decision model only when all the parameters are fixed.

Further, the literature does not discuss the effort required to build these different systems (in terms of man-months), thus making it difficult for researchers to choose a particular technique on the basis of time and cost. An assessment of the scale of involvement of expert traders is very much needed as expertise in this domain is generally very expensive.

It is only Braun & Chandler's rule induction study that compares system performance with that of expert trader performance. It is desirable to have comparisons with expert performance for several reasons. Firstly, from a trading organisation viewpoint it provides an objective measure for assessing the usefulness of these new technologies for automating tasks that are currently performed by humans. Secondly through the analysis of the relative performance of the expert and the intelligent systems, it may be possible to identify periods and conditions under which each approach has particular merits.

It is also important to have domain experts to evaluate decision models to guard against the possibility of induced models capturing behaviour of the operating environment from past data that is no longer valid. Further, if the induced decision model does not follow a particular 'logic' or line of reasoning that the expert can understand and rationalise it may not be accepted for use in that organisation. Drawing a parallel from the application of induction methods in medicine, Evans [42] reports that in the domain of dysmorphology different experts can hold completely different interpretations of the same data depending on whether (say) the expert is a clinical geneticist or a specialist in radiology. The acceptance of a particular decision making system would therefore depend both on its 'objective' performance as well as its conformance to a particular set of theories or ideologies that are in place in that organisation or discipline.

As will be discussed in detail in Chapter 4, this research adopts an intelligent hybrid approach to financial trading and gives performance comparisons of different techniques using the same data. Technical analysis knowledge is used in this research, and the techniques are demonstrated using foreign exchange trading data. In Chapter 9 we present results of an empirical investigation into expert trading decision-making and evaluate our INTENT system results (presented in Chapters 5, 6, 7, 8 and 9) against the expert trader performance.

Chapter 3

Intelligent Hybrid Systems : Issues and Classifications

This chapter presents: a critical examination of the computational properties of different intelligent systems; arguments for the need for hybrid systems; and a novel classification scheme for intelligent hybrid systems with example systems.

3.1 Properties of Intelligent Reasoning Systems

The previous chapter introduced different intelligent techniques in the context of decision support systems for financial trading. Each individual intelligent technique has particular strengths and weaknesses, and cannot be applied universally to every type of problem. The central justification for trying to create hybrid systems is that by integrating different techniques we may be able to overcome the limitations of individual techniques.

In order to discuss how such integration can be fruitfully made, we need to examine more closely the key factors that characterise different intelligent systems. These factors are brittleness, knowledge acquisition, higher and lower level reasoning, and explanation. We evaluate these factors with respect to the intelligent techniques introduced in the previous chapter and then examine hybrid systems which attempt to overcome the limitations of individual techniques.

3.1.1 Brittleness

Although there are notable successes of expert systems, almost all of these systems operate in very narrow domains under limited operational conditions. Holland [58] calls this phenomena *brittleness*:

> 'The systems are *brittle* in the sense that they respond appropriately only in narrow domains and require substantial human intervention to compensate for even slight shifts in domain.'

An operational view of the brittleness problem can be seen as the inability of an intelligent system to cope with inexact, incomplete or inconsistent knowledge. Causes of this *Brittleness Problem* are twofold — inadequate representation structures and reasoning mechanisms. In symbolic systems, knowledge is represented in atomic symbolic constructs and reasoning consists of *logical* operations on these constructs.

In contrast to symbolic reasoning where only one operation is active at a given moment, neural networks have many computational units that process information. *Reasoning* in neural networks involves the numeric aggregation of representations over the whole network [12]. As Smolensky [114] points out,

> "Each connection represents a soft constraint; the knowledge contained in the system is the set of all such constraints. If two units have an inhibitory connection, then the network has the knowledge that when one is active the other ought not be. Any of the soft constraints can be overridden by others, they have no implications singly; they only have implications collectively. "

This distributed representation and reasoning allows these systems to deal with incomplete and inconsistent data and also allow the systems to *gracefully degrade* [104]. That is, even if parts of the network are made non-operational, the rest of the network will function and attempt to give an answer. This type of inherent fault tolerance strongly contrasts with symbolic processing systems which usually fail to function even if a single processing part is non-operational.

At present most expert systems are susceptible to the brittleness problem because of their strict, logical reasoning structures as well as their symbolic representations. This manifests itself in many ways, particularly as problems in the maintenance of large, complex knowledge bases [82].

Symbolic rule induction systems such as ID3 are also susceptible to the problems of brittleness. They have difficulties in inducing relationships in data which has contradictory and incomplete examples [46]. However, recent enhancements of the ID3 algorithm have overcome some of the limitations with respect to its handling of noisy data [97].

Fuzzy logic deals with the problem of brittleness by adopting novel knowledge representation and reasoning methods. Fuzzy sets, the form in which knowledge is represented, diffuses the boundaries between concepts. There are no sharp divisions where one concept ends and the next begins. This fuzzy data representation in conjunction with fuzzy reasoning mechanisms allows a reasoning mechanism that can process data which are non-exact or *partially correct* [135].

The key argument for Genetic Algorithms being able to cope with brittleness is put forward by Holland [58]. Holland argues that it is the maintenance of a ***population of solutions*** which makes genetic algorithms and classifier systems non-brittle. Each rule in the classifier system population contains a relationship describing the system being modelled. The system's flexibility arises from the rules representing a wide range of competing, conflicting hypotheses. The selection of the appropriate rule to fire is dependent on its past performance — a statistical aggregation of its 'correct' performance. Similar to neural networks, it is this ***statistical reasoning*** property, based on past performance that gives genetic algorithms their ability to cope with brittleness.

3.1.2 Knowledge Acquisition

Knowledge acquisition is a crucial stage in the development of intelligent systems. As a process, it involves eliciting, interpreting and representing the knowledge from a given domain [68].

As mentioned in Chapter 1 expert systems suffer from problems in knowledge elicitation and representation. These tasks are often time consuming, expensive, and potentially unreliable [43].

Because of these problems, several symbolic learning systems have been applied to the automation of the knowledge acquisition process. Rule induction systems as introduced in Chapter 2 have been used to learn rules and decision trees from 'raw' domain data and have been applied successfully in industrial and commercial domains [96], [85]. Fuzzy systems, similar to Expert Systems, require a domain expert to acquire the necessary domain knowledge.

Neural networks and genetic algorithms have the ability to learn from domain data, and to form their own 'internal model' of that domain [26]. Through the processes of gradient descent learning in the case of neural networks and stochastic, randomised search in the case of genetic algorithms, they induce relationships in the environment they operate in.

3.1.3 Higher and Lower Level Reasoning

For a complete theory of cognition, there need to be explanations of how we can do 'low level', parallel, pattern recognition tasks as well as how we perform sequential, 'high level' cognitive tasks.

Although symbolic techniques have provided plausible models to describe 'high level' cognitive tasks such as language generation and comprehension [18], and goal directed reasoning [18] , their ability to explain 'low level' pattern recognition type tasks has been limited. In contrast, sub-symbolic techniques such as neural networks offer the exact complementary ability. They are very good at modelling pattern recognition tasks such as visual processing and motor control but are not well equipped to model sequential, high level cognitive tasks. Because of these limitations, some researchers have criticised neural networks as inadequate models of cognition [45].

An analogue to cognitive modelling can be found in complex real world industrial and commercial problems. That is, most complex real world tasks are easily decomposed into logical, sequential operations and parallel, pattern recognition operations [54]. For example in industrial robotics, neural networks are good candidates for performing low-level signal processing tasks, while symbolic systems are good candidates for higher-level path planning tasks [120]. Similar tasks which have mixed signal processing/high-level reasoning sub-problems can be found in many industrial control and financial data analysis.

3.1.4 Explanation

The ability to provide users with explanations of the reasoning process is an important feature of intelligent systems [27]. Explanation facilities are required both for user acceptance of the solutions generated by an intelligent system, and for the purpose of understanding whether the reasoning procedure is sound [31]. Good examples of this requirement can be found in medical diagnosis, loan granting, and legal reasoning.

There have been fairly successful solutions to the *Explanation Problem* by symbolic systems such as expert systems and case based reasoning systems. In expert systems and rule induction systems explanations are typically provided by tracing the 'chain of inference' during the reasoning process [115]. In Fuzzy Logic, although one cannot trace a 'chain of reasoning' as the effects of all the rules are aggregated, one can nevertheless inspect the rules and understand the hypotheses the rules hold.

Genetic Algorithms, especially in the form of Classifier Systems, can build reasoning models in the form of rules [59]. As in the case of Expert Systems, it is possible to trace a chain of reasoning and provide some degree of explanation of the reasoning process.

Explanation in Case-Based Reasoning systems is provided by retrieving cases (from case databases) that are most similar to the current set of observations or inputs from the operating environment [70]. In areas such as legal reasoning, explanations in terms of past cases or histories is commonly used by domain experts.

In contrast, in neural networks it is difficult to provide adequate explanation facilities. This is due to neural networks not having explicit, declarative knowledge representation structures but instead having knowledge encoded as *weights* distributed over the whole network [33]. It is therefore more difficult to find a 'trace of reasoning' which can be used for producing explanations.

There is now a small but growing number of researchers who are attempting to provide neural networks with explanation facilities by extracting rules from their internal weight structures [47]. Additionally there has also been progress in using 'feature significance estimation' methods that identify the most important variables in a given neural network model [110].

3.2 The Case for Hybrid Systems

We now put forward three central arguments for producing intelligent hybrid systems: *Technique enhancement*, the *Multiplicity of Application sub-domains* and the *Demonstration of Computational principles.*

(1) *Technique Enhancement*: this is the integration of different techniques to overcome the limitations of each individual technique. Creating this type of hybrid can be viewed as taking a technique that scores low in a particular property (see table 3.1) and combining it with a technique that scores high on that particular property. For example, fuzzy systems score high in their ability to deal with brittleness and in their ability to provide explanations, but score low in their ability to acquire knowledge

(the membership functions and rules have to be manually specified). A neural network that scores high on knowledge acquisition can therefore be potentially used to create a hybrid with fuzzy systems to overcome the limitation of fuzzy systems. Recently, there have been several examples [71], [83] of such fuzzy logic/neural network combinations used in this fashion.

Computational Properties

Technologies	**Knowledge Acquisition**	**Brittleness**	**High-level Reasoning**	**Low-level Reasoning**	**Explanation**
Expert Systems	✓	✓	✓✓✓✓✓	✓	✓✓✓✓✓
Rule Induction	✓✓✓✓	✓✓	✓✓✓	✓✓	✓✓✓
Fuzzy Systems	✓	✓✓✓✓✓	✓✓✓	✓✓✓✓✓	✓✓✓
Neural Networks	✓✓✓✓✓	✓✓✓✓✓	✓	✓✓✓✓✓	✓
Genetic Algorithms	✓✓✓✓✓	✓✓✓	✓✓✓	✓✓✓	✓✓✓

Figure 3.1: The Properties of Different Intelligent Techniques

(2) *Multiplicity of Application sub-domains*: The reason for creating these hybrid systems is because no single technique is applicable to the many sub-problems that a given application may have. Most real world domains have both 'logical', static components (that can be easily handled by symbolic techniques) and fuzzy, dynamic, poorly understood components (which can be handled by sub-symbolic techniques).

The aim therefore is to match techniques that score high on particular properties with tasks that may require such properties. For example if a particular application has say two distinct sub-problems, a 'low-level' signal processing task and a serial reasoning task then a neural network and an expert system respectively can be used for these tasks. Many complex real world tasks can be broken down into distinct sub-problems each of which can be solved by distinct intelligent techniques.

(3) *Demonstration of Computational principles*: The motivation for creating these hybrid systems is to demonstrate particular computational principles, mainly cognitive mechanisms, where a particular technique is used to *emulate* a different intelligent

technique. The prime example of such systems is the use of neural networks for symbol processing, demonstrating the cognitive principle that connectionist networks can be used for serial reasoning [6], [117].

3.3 Three Hybrid Classes

In this section we propose a classification scheme for intelligent hybrid systems, drawing from the above analysis. This classification scheme (see figure 3.2) takes into account factors such as functionality, processing architecture and communication requirements.

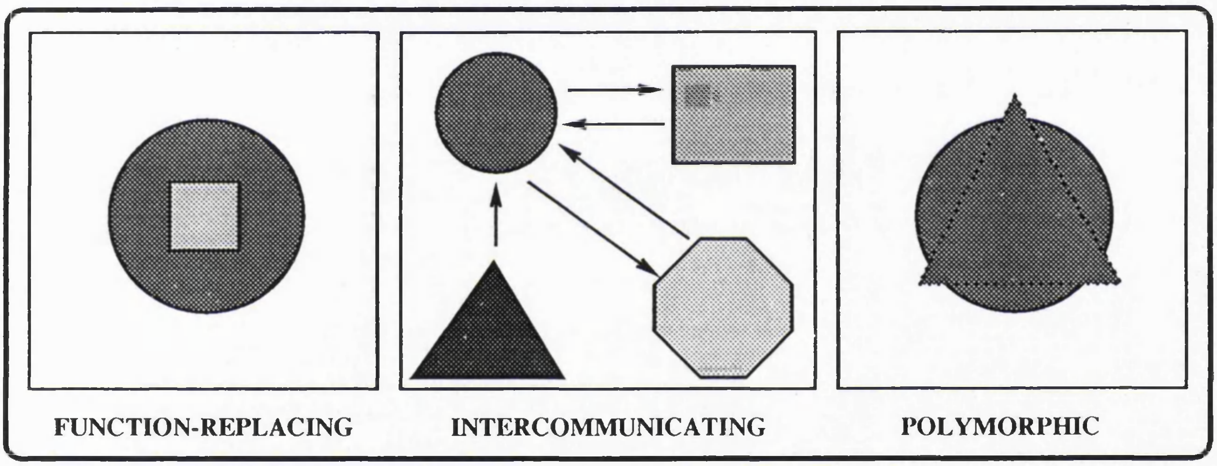


Figure 3.2: Three Proposed Hybrid Classes

Although we have limited our discussion to hybrids of intelligent techniques, the proposed hybrid classification is also applicable to hybrids of other numeric and symbolic techniques (e.g. statistical clustering, regression techniques, linear programming etc.).

3.3.1 Function-Replacing Hybrids

Function-Replacing Hybrids address the functional composition of a *single* intelligent technique. In this hybrid class, a *principal function* of the given technique is replaced by *another* intelligent processing technique. The motivation for these hybrid systems, is the *technique enhancement* factor discussed above.

The motivation for replacing these *principal functions* could either be to increase execution speed or enhance reliability. Examples of *principal functions* include pattern matching in an expert system, weight changing in a neural network and crossover operations in a genetic algorithm.

Montana and Davis [89] provide an example of a *function-replacing hybrid.* They replace the backpropagation weight-changing mechanism of a neural network with genetic algorithm operators. The genetic algorithm takes the existing weights of the neural network and then applies mutation and crossover operators on these weight values to obtain new weights. The weights of the network are encoded in a list structure (see figure 3.3). Crossover and mutation operators are performed on these lists representing the weights. The performance of this genetic weight updating method was compared with back-propagation [104] on a sonar pattern recognition task. The hybrid method outperformed back propagation by taking a much smaller number of iterations to converge to a good solution [89].

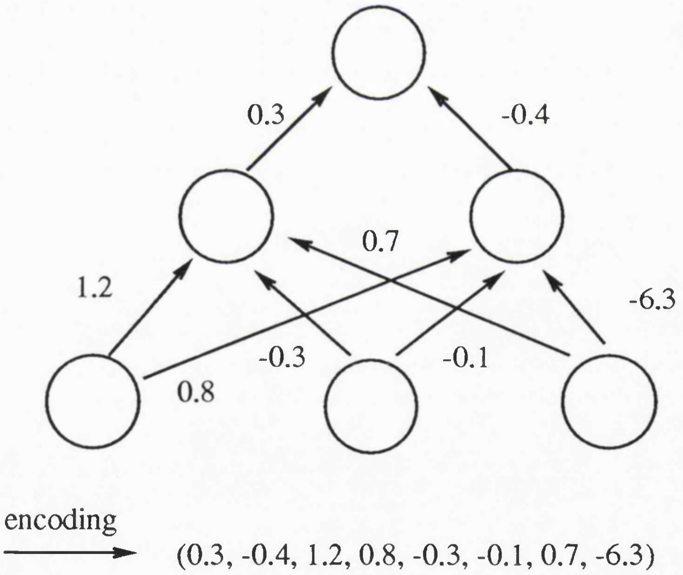


Figure 3.3: Montana & Davis [1989] Neural Network Chromosome Encoding

Another good example of a function replacing hybrid is the Karr's [66] work on using a genetic algorithm for replacing the task of manually defining fuzzy membership functions. The definition of a membership function is usually done manually by a domain expert who uses his or her subjective judgement to specify the shapes and values that the membership function may take. Typically this is a very time consuming trial and error process that takes the largest amount of time in the development of fuzzy systems.

Karr uses a genetic algorithm to design fuzzy membership functions in a fuzzy system for controlling cart-pole balancing. There are four condition variables each having three linguistic variables. Karr uses triangular shapes in the fuzzy membership definition (see figure 3.4), and the aim of the genetic algorithm is to find the optimal 'anchor points' in these triangles. Bit strings were used to represent the possible positions of these anchor points. After having viewed only a small portion of the search space (approximately 32000 of the 2^{132} possible points), the genetic algorithm was able to learn membership functions that were much better at controlling the cart-pole than the membership functions defined by the author.

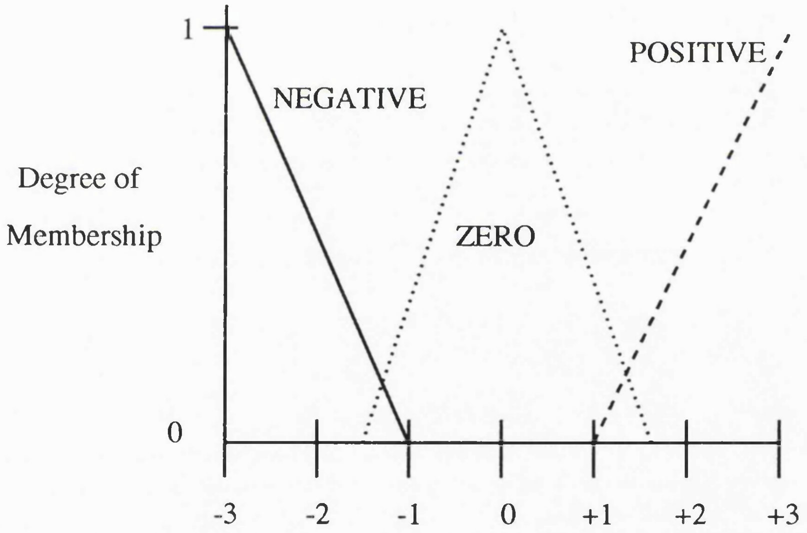


Figure 3.4: Karr [1991] Fuzzy Membership Definitions

3.3.2 Intercommunicating Hybrids

Intercommunicating Hybrids are *independent*, self contained, intelligent processing modules that exchange information and perform separate functions to generate solutions.

If a problem can be subdivided into distinct processing tasks, then different independent intelligent modules can be used to solve the parts of the problem that they are best at. These independent modules which collectively solve the given task are co-ordinated by a *control mechanism*.

For example, if a particular task has sub tasks of pattern recognition, serial reasoning and optimisation, then a neural network, an expert system and a genetic algorithm can be made to perform these respective tasks. These independent modules can process sub-tasks either in a sequential manner (where the control can be provided in the rules of the expert system) or in a competitive-co-operative framework such as a blackboard problem solving architecture (where the control is in a distinct control component).

An example of an *intercommunicating hybrid* is Schreinemakers and Touretzky's system for veterinary diagnosis [109]. In their system, a rule based expert system performs serial inferences, and calls neural networks to find patterns in the data as part of the diagnosis. The system uses a particular set of OPS5 data structures called Working Memory Elements (WMEs) for communication between the neural network and the expert system.

For example a typical rule in the system is,

```
(p FEED-COW-TO-MASTITIS-NETWORK
 ( subgoal (goal type diagnose-cow number n )
  ( cow-wme (cow-descriptor number n)
  → (call invoke-network mastitis-net cow-wme)
  (remove subgoal))
```

If the neural network mis-diagnoses a particular case, then the expert system acts as 'knowledge manager' by allowing a domain expert to inspect the mis-classification and to change the associated training data if such data is found to be incorrect. The authors claim that the system achieves a classification rate of 87%, which is comparable to the performance of an expert veterinarian.

A further example of this type of intercommunicating hybrid is given by Dunker et al [35] who describe a co-operative environment for integrating neural networks and knowledge based systems. The principal aim of this system is to decompose large complex reasoning tasks into manageable sub-problems and to process them via a blackboard architecture. The agents of the blackboard system are several 'neural problem solvers', each concerned with modelling a particular aspect of the problem domain. A neural problem solver consists of a neural network and a symbolic control mechanism. Each of these neural problem solvers pass and receive messages to and from the central blackboard. If all the conditions for a neural problem solver are met (checked by the symbolic control mechanism) then that particular neural network agent will be used to process the data. Once the processing has been completed, it will pass information to the blackboard indicating that the sub-task is finished and will post the relevant processed results. Another neural problem solver now can use these results for further processing. Dunker et al are planning to use this system in banking for the analysis of investment opportunities.

3.3.3 Polymorphic Hybrids

Polymorphic Hybrids are systems that use a *single processing architecture* to achieve the functionality of *different* intelligent processing techniques. The broad motivation for these hybrid systems is the *demonstration of different computational principles* within particular computational architectures. These systems can *functionally mimic* or emulate different processing techniques. These might be viewed as being 'chameleon-like' systems which can change their functional form.

Examples of Polymorphic hybrids are neural networks that attempt to perform symbolic tasks such as step-wise inferencing and also neural networks which function as if they were doing genetic search.

Ajjanagadde and Shastri [6] demonstrate a *Polymorphic hybrid system* where a neural network achieves the functionality of a symbolic reasoning system. This system addresses the symbolic variable binding problem (how to dynamically represent variables and their role bindings) by propagating *rhythmic* patterns of activity wherein dynamic bindings are represented as *in-phase* firings of appropriate nodes in the network. This allows a rule-like chain of inference and reasoning to be present within a neural architecture. The primary motivation for this model is to demonstrate the cognitive principle of how serial processing can be achieved with a neural processing mechanism.

3.4 Discussion

In this chapter a novel classification scheme for hybrid systems has been introduced and example systems discussed. Currently the word 'hybrid' has acquired several interpretations within the context of intelligent systems [48]. This confusion has been clarified by this scheme and since this work was first reported several researchers in the field have adopted it to describe their own research [50] [107] [106]. Madey [50] classifies his hybrid of a neural network and an expert system based simulation system (used to model a manufacturing task) as an intercommunicating hybrid. Scherer et. al. [107] also use the intercommunicating classification to describe their blackboard system based neural network and expert system hybrid. This hybrid is being applied in the area of investment decision analysis.

However as with any classification scheme there are certain limitations. The difficulties are in what may be termed as 'border-line' cases which can appear to fall into more than one category. For example there can be particular hybrid systems which are function-replacing but which nevertheless contain quite distinct, separate processing modules that intercommunicate their results thereby appearing to satisfy the definition of an intercommunicating hybrid. For example, a genetic algorithm that replaces the function of finding the weights in a neural network where the genetic algorithm and neural network intercommunicate their results (say running on two separate machines) may appear as being both intercommunicating and function-replacing.

In order to resolve these situations, emphasis must be shifted from the communication of results to the nature of the application domain. As detailed in the definition of intercommunicating hybrids, they are characterised by both the multiplicity of the application domain (use of different techniques for different parts) as well as their intercommunication of results. The above mentioned example of conflicting categories only arises if the communication aspects of the intercommunicating hybrids is emphasised. If on the other hand it is viewed from the viewpoint of examining the characteristics of the application domain, then the above mentioned genetic-neural system is clearly not an intercommunicating hybrid as it does not solve different parts of the application. This hybrid is a function-replacing hybrid despite the fact that there is a strong element of intercommunication of results between the two techniques. Although the majority of hybrid systems clearly fall into one of three classes, in a small number of cases such a detailed analysis of the defining characteristics of the classes will be required.

A general comment regarding the proposed classification scheme is the limited number of examples in the Polymorphic category. The majority of the Polymorphic hybrids are examples of neural networks used for symbolic reasoning and there is one example of using neural networks for imitating genetic search procedures [4]. However it is possible to speculate the possibility of Polymorphic hybrids of say fuzzy systems behaving as if they were neural networks, and neural networks as if they were fuzzy systems because of the similarities in their processing of imprecise concepts. It is likely that many more examples of such Polymorphic hybrids will be developed as more researchers studying different intelligent techniques begin to understand the parallels between the different approaches.

To our knowledge there is one other hybrid systems classification scheme which has been developed by Medsker and Bailey [84]. This scheme was published after we reported our classification scheme [49] and is concerned with classifying expert systems and neural network hybrids. There are five categories in Medsker and Bailey's scheme: stand-alone models, transformational models, loose-coupling models, tight-coupling models and full-integration models.

Stand alone models are defined as systems which have neural networks and expert systems solving a particular task completely independent of each other without any exchange of information. Medsker and Bailey cite cases where one system maybe used as a parallel 'backup' system in case the other intelligent system breaks down. However whether this type of system can be regarded as a hybrid system is debatable. We believe that some notion of information exchange has to be present in systems that are regarded as hybrid systems. Medsker and Bailey also seem to share our reservations on this category by declaring "stand-alone models are a degenerate case for integration purposes" [84].

Medsker and Bailey's second category is transformational models. There are two forms of transformational models: expert systems that are transformed into neural networks, and neural networks that metamorphose into expert systems. They say, " a neural network can be developed to identify trends and relationships within sales data and then the neural network can be used as the *basis* for an expert system. An expert system is useful to verify the knowledge and to provide justification capabilities".

They say that expert systems can also be transformed into neural networks by "using an expert system to set the initial conditions for the neural network learning".

Medsker and Bailey provide no example systems for this category and it is not difficult to see why. In order for such neural network/expert systems transformations to occur, there needs to be significant development in our understanding of knowledge representation and manipulation in neural networks. The closest to such transformations is the use of neural networks to extract rules [47] which can then be used by expert systems. However the issue of transforming an expert system into a neural network appears to be well outside the capability of current technology. Medsker and Bailey themselves acknowledge the difficulty in this category by saying, "Limitations to transformational models are significant. There is no fully automated way of transforming an expert system into a neural network and vice versa. In fact, there is no known method for accurately and completely performing the transformation".

The third Medsker and Bailey category is loosely-coupled models which is defined as separate neural network and expert system components that communicate via *data files*. Typical examples cited are when neural networks are used as pre-processors or post-processors for expert systems. This category on first inspection appears to be similar to our intercommunicating class but there are differences which we will discuss after detailing the next category.

The only difference between the fourth category, tightly-coupled models, and the loosely-coupled models is that the tightly coupled models use resident data structures as opposed to external data files. Similar to loosely-coupled models, tightly-coupled models also consist of separate processing models that exchange information and typical examples include neural networks as pre-processors for expert systems.

We believe that making a distinction between two hybrid classes purely on the basis of using files as opposed to resident memory structures for information exchange is not very useful. We believe that the defining properties of hybrid systems should be made on the basis of broad *functional* properties and benefits, and not on the basis of implementation details. It is only by having a functional viewpoint of hybrid systems that users can conceptualise and understand the benefits of such systems and begin to ask relevant questions (e.g. What limitations of the technique can be overcome by hybridisation ? Are there different sub-problems that can be solved by separate techniques ?)

Both the Medsker and Bailey loosely-coupled and tightly-coupled categories fall into the intercommunicating hybrid class in our scheme. A key definition of the intercommunicating class was the multiplicity of application sub-domains, and both of the above categories fall into this category regardless of whether they use files or memory structures for exchanging information.

The only Medsker and Bailey category that is congruent to one of our classes is their fifth category — fully-integrated models. The fully-integrated models are expert system/neural network models that share data structures and knowledge representations. Communication between the different components is accomplished via the dual nature (symbolic and neural) of the structures. This category is the same as our polymorphic category.

Medsker and Bailey have completely missed the notion of a function-replacing hybrid. This is because their definition of hybrids has been heavily biased towards implementation and communication issues (e.g. file handling) and less towards functional distinctions.

In conclusion, our classification scheme is simpler (3 categories, as opposed to 5) and it does not contain redundant classes (e.g. transformational category) thereby helping users of the scheme to classify systems more easily. We also believe our scheme is more useful to users as it defines systems based on the functional aspects of hybrid systems (e.g. benefits of using one technique to overcome the limitations of another) rather than making distinctions based on detailed implementation issues. Finally, in our opinion, it is the functional definition of our classification that allows the possibility of formulating a general methodology for hybrid systems. A sketch of a methodology for hybrid systems based on our classification scheme is presented in Chapter 9.

Chapter 4
The INTENT System

This chapter presents: the requirements of our decision support system for financial trading; the design considerations of the INTENT system; the INTENT architecture; and the functioning of the different knowledge processing components.

4.1 Requirements of our decision support system for financial trading

Before we outline the design of our hybrid system we detail the specific properties of the financial trading domain. The properties that our system attempts to satisfy are: the ability to acquire trading knowledge, the ability to cope with brittleness, continuous learning, transparency of decision model and the ability to incorporate human knowledge. As we have discussed the issue of knowledge acquisition previously, we start with the brittleness criteria specific to financial trading.

- **Ability to cope with Brittleness**

  Financial market data are typically very noisy [67]. A good financial decision support system should be robust enough to find patterns in such data. In financial systems it is rare that a particular set of conditions (say, particular values of market indicators), if observed again, will produce the same future market behaviour. Because of this non-deterministic nature of financial markets, strict deterministic rules will fail to perform well in this domain. For example, if by observing past data one induces a rule that if the Pound/Dollar exchange rate is 1.546 and the volume of contracts traded is 23898 then the market will rise, it is probable that because of these very specific conditions the future market data will never match these conditions exactly. It is therefore vital that a good trading decision system should have the capability to induce patterns in a fuzzy or statistical manner. The patterns they induce should be robust enough so that they perform well even when there are slight shifts in the operating environment; in other words to be able to cope with *brittleness* as discussed in the previous chapter.

- **Continuous Learning**

  A key feature of financial markets is that they evolve over time [15]. As financial traders know, the characteristics that define a particular market can change significantly over a relatively short time period. For example, for a few months, rises in interest rates may strengthen a currency, while it is possible that in following months, rises in interest rates may actually weaken a currency [92]. Market practitioners are very aware of such character changes in markets and can quickly adjust to these.

  A good example of changes in financial systems behaviour is the recent failure of 'value investing techniques' in the US stock market [2]. Buying 'value' shares - those with low prices relative to, say, earnings and dividends - has until recently produced returns that beat the stock market average [61]. This technique has worked well in all of the major stock markets (US, UK, Japan, France) for the last thirty years. Recently, however, there are signs that this strategy is failing in the United States due to the Clinton Administration's interventionist policies in the health care, tobacco and energy industries [2]. That is, even shares which appear to have good 'value' have recently found their share prices declining dramatically due to US Government policies. A prime example of this phenomenon is the collapse of health care stocks due to the proposed reforms of the industry.

  Because of these changes of character in financial systems, a trading decision support system should, ideally, have the ability to adapt to such changes and be able to make successful trading recommendations before and after such changes. In order for this to happen, the trading decision support system should have the capability to learn continuously from the market. That is, it should constantly induce new relationships, test them, and then present them to the user.

- **Transparency of Decision Model**

  When financial trading decisions involve large amounts of money, reassurance as to the soundness of the decision-making procedure is needed. The ability to cite the exact conditions and reasoning of a trading decision, is often required by senior managers in financial organisations. As computer assisted decision making is still relatively uncommon, most managers do not entirely 'trust' machine generated trading decisions and require an understanding of the decision process in a format that they can understand. Because of this factor, some investment managers remain suspicious of the use of neural networks and other 'black box' techniques for making trading decisions [1].

It is also important to have an understanding of the reasoning process in order to improve the model. For example, if a trading model ceases to work for some reason, it can only be corrected if the reasoning processes are understood. On the other hand, if a black-box trading system ceases to make good trading decisions, then it will be very difficult to understand as to what is causing the system to make incorrect decisions.

- **Ability to incorporate human knowledge**

Judgmental forecasting, where domain experts use their experience alone to make predictions, is widely used in a variety of domains ranging from weather forecasting to predicting sales figures [24]. There is now evidence to suggest that sometimes expert or judgmental forecasting has particular advantages over quantitative methods [24]. These advantages are particularly evident under special circumstances such as in the case of extraordinary competitive developments [64].

Judgmental forecasting procedures are also used as a method of *enhancing* predictions derived from quantitative models [24]. The predictions of macroeconomic models are often subjected to such revisions by domain experts [41].

Bunn and Wright [24], in their review of judgmental forecasting, identify two reasons for such revisions in econometric models.

1. *Specification Error*: If the model has not been performing well recently then it is quicker to perform an *ad hoc* adjustment to the output rather than to re-specify the model.

2. *Structural Change*: Some external factor or a background assumption is likely to influence future events. These adjustments are also known as 'broken-leg cues' an analogy which says that one might adjust a statistical model of a person's mobility on learning that the person has broken his leg.

In our financial trading domain we would also like to be able to have the ability to perform judgmental revisions on our models. Financial markets frequently undergo significant character changes and often have 'temporary shocks'. Judgmental forecasting can therefore potentially play a role in increasing the accuracy of trading decisions in such situations.

4.2 The System Design Considerations

The main goal in the design of our decision support system was to satisfy as many of the above outlined requirements as possible. In order to construct our 'ideal' system, we started with a systematic evaluation of the many available intelligent techniques. Parts of the results of this survey and evaluation have been presented in the two previous chapters, and we now briefly summarise these findings in the context of choosing techniques for our system. The resulting evaluation suggests that there is no *single* technique that has all the desired properties for a decision support system for financial trading.

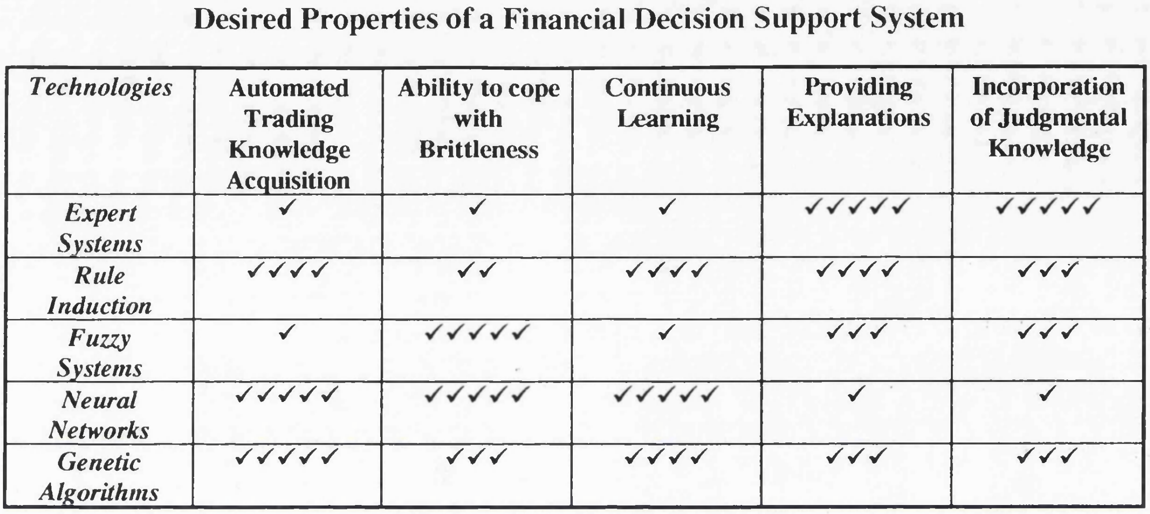
As discussed in chapters 2 and 3, Expert Systems have limitations of not being able to automatically acquire trading knowledge, not being able to cope with brittleness, and not being able to adapt to changing market conditions as they cannot learn continuously. On the positive side, their reasoning is transparent and they allow the incorporation of judgmental knowledge.

Rule Induction systems have the ability to automate the knowledge acquisition process, can learn continuously and humans can inspect and understand induced rules. Because of their rule-based representation, judgmental knowledge can also in principle be incorporated. However, they usually have problems with inducing relations in noisy data. This is a main consideration in the case of financial trading.

Fuzzy Systems cannot acquire trading knowledge in an automated fashion, and cannot learn continuously. But they are exceedingly good at coping with brittleness, and their knowledge bases, which in a rule-based format can be examined and understood by a human expert. The rule based format also facilitates the incorporation of judgmental knowledge.

Neural Networks can learn the trading knowledge, can learn continuously and can cope with brittleness. However they are 'black box' reasoning systems that provide little in terms of explanation of trading decisions. It is also difficult to see how any explicit judgmental knowledge can be incorporated into a neural network.

Genetic Algorithms can automatically acquire the trading knowledge and can learn continuously. They are good in coping with brittleness and decision models derived using GAs are transparent.



Desired Properties of a Financial Decision Support System

Technologies	Automated Trading Knowledge Acquisition	Ability to cope with Brittleness	Continuous Learning	Providing Explanations	Incorporation of Judgmental Knowledge
Expert Systems	✓	✓	✓	✓✓✓✓✓	✓✓✓✓✓
Rule Induction	✓✓✓✓	✓✓	✓✓✓✓	✓✓✓✓	✓✓✓
Fuzzy Systems	✓	✓✓✓✓✓	✓	✓✓✓	✓✓✓
Neural Networks	✓✓✓✓✓	✓✓✓✓✓	✓✓✓✓✓	✓	✓
Genetic Algorithms	✓✓✓✓✓	✓✓✓	✓✓✓✓	✓✓✓	✓✓✓

Figure 4.1: Evaluating Intelligent Techniques for Decision Support Systems

We summarise this assessment in table 4.1. This table has similarities with the table in the Chapter 3 which evaluated general computational properties but is specific to the properties of the trading domain.

After this evaluation was completed, we assessed ways of combining different techniques to obtain all of our desired properties. We wanted to have,

- the knowledge acquisition and continuous learning abilities of either rule induction, neural networks or genetic algorithms
- the ability to cope with brittleness like fuzzy logic or neural networks
- the decision model transparency of expert systems or rule induction systems
- the ability to incorporate judgmental knowledge like fuzzy systems, expert systems or rule induction systems

4.2.1 Knowledge Communication and System Design

After assessing the desired property list, we decided to create a hybrid of expert systems, fuzzy systems and genetic algorithms. Between these three techniques, they seem to have all the desired properties (automated knowledge acquisition, ability to cope with brittleness, ability to generate transparent decision models, and the ability to incorporate judgmental knowledge). However the crucial question was how to combine them in a way that knowledge could be *communicated* between the systems in a manner that satisfied our goals. If one wants to integrate different techniques, then one needs to find a common *knowledge communication protocol* or platform that allows the communication of results and partial results between the different intelligent techniques. If for example a Frame system [100] was used to represent knowledge in the expert system, then it is not clear how a genetic algorithm can fruitfully use that knowledge, or how a genetic algorithm can induce knowledge in a frame format in order to create the expert system knowledge base.

A solution to this problem was to adopt *production rules* as our common knowledge representation format. Both expert systems and fuzzy systems use production rules and we saw no problems in implementing both systems having rules with the same *syntactic* structure. Classifier systems, genetic algorithms that induce production rule systems, were also a potential candidate for the remaining part of the hybrid system.

There was also another strong argument from the financial trading literature to adopt rules as our base knowledge representation scheme. As discussed in chapter 2, there is a large body of literature supporting the use of technical analysis knowledge for trading decision making, and most of this knowledge is inherently rule-based.

However, there was a problem in choosing a representation scheme for the genetic algorithm. Most classifier systems and genetic algorithms are represented using binary strings [59], and it was not clear how such a representation scheme would fit well with the expert system and fuzzy system rule representations. We therefore decided to use production rules, encoded as symbolic list structures, as the representation scheme for solutions in our genetic algorithm. The genetic production rules in our system have the same syntactic structure as the rules in the expert and fuzzy systems.

Recently similar symbolic representation schemes (as opposed to binary bit string representations) have become popular with Koza's [72] Genetic Programming paradigm where LISP list structures are used to derive computer programs. Due to the heavy reliance on list processing in our system, we have used POP-11 [98], an AI list processing language, to implement our system. POP-11 provides a variety of in-built pattern matching mechanisms which helps users to construct symbolic reasoning systems fairly quickly.

4.3 The INTENT Architecture

Our decision support system, INTENT (INtelligenT DEcisioN SupporT Hybrid System), consists of three separate hybrid systems, which we refer to as different *strands* of the system.

The first strand is called the **expert-fuzzy-genetic strand** and is a *function-replacing hybrid* according to our classification scheme. This integrates an expert system, a fuzzy system, and genetic algorithms for making trading decisions. As we shall explain below, this combination of techniques allows us to satisfy all of the previously specified requirements.

The second strand is called the **genetic-neural strand** and it is an *intercommunicating hybrid.* This hybrid combines neural networks and genetic algorithms.

The third strand, the **multiple model strand**, is also an *intercommunicating hybrid.* Here the results of all of the different modules (expert system, fuzzy system, genetic algorithms and neural networks) are combined.

We now provide a very brief introduction to the three different strands of processing. The design, implementation, experiments and results of each processing module will be presented in detail in the following five chapters.

4.3.1 The Expert - Fuzzy - Genetic Strand

This strand is a *function replacing hybrid system.* Here intelligent techniques are combined with the aim of overcoming the limitations of individual techniques. The intelligent techniques that are combined are an expert system, a fuzzy system and a genetic algorithm (see figure 4.2). The genetic algorithm acts as a rule induction mechanism for the expert and fuzzy systems, thereby replacing the task of manually specifying rules by a domain expert.

Symbolic data Pre-processing

The starting point for this strand of processing is the pre-processing of raw market data. The raw market data consists of three variables — price, open interest and volume. As discussed earlier in the chapter we do not want to induce rules from

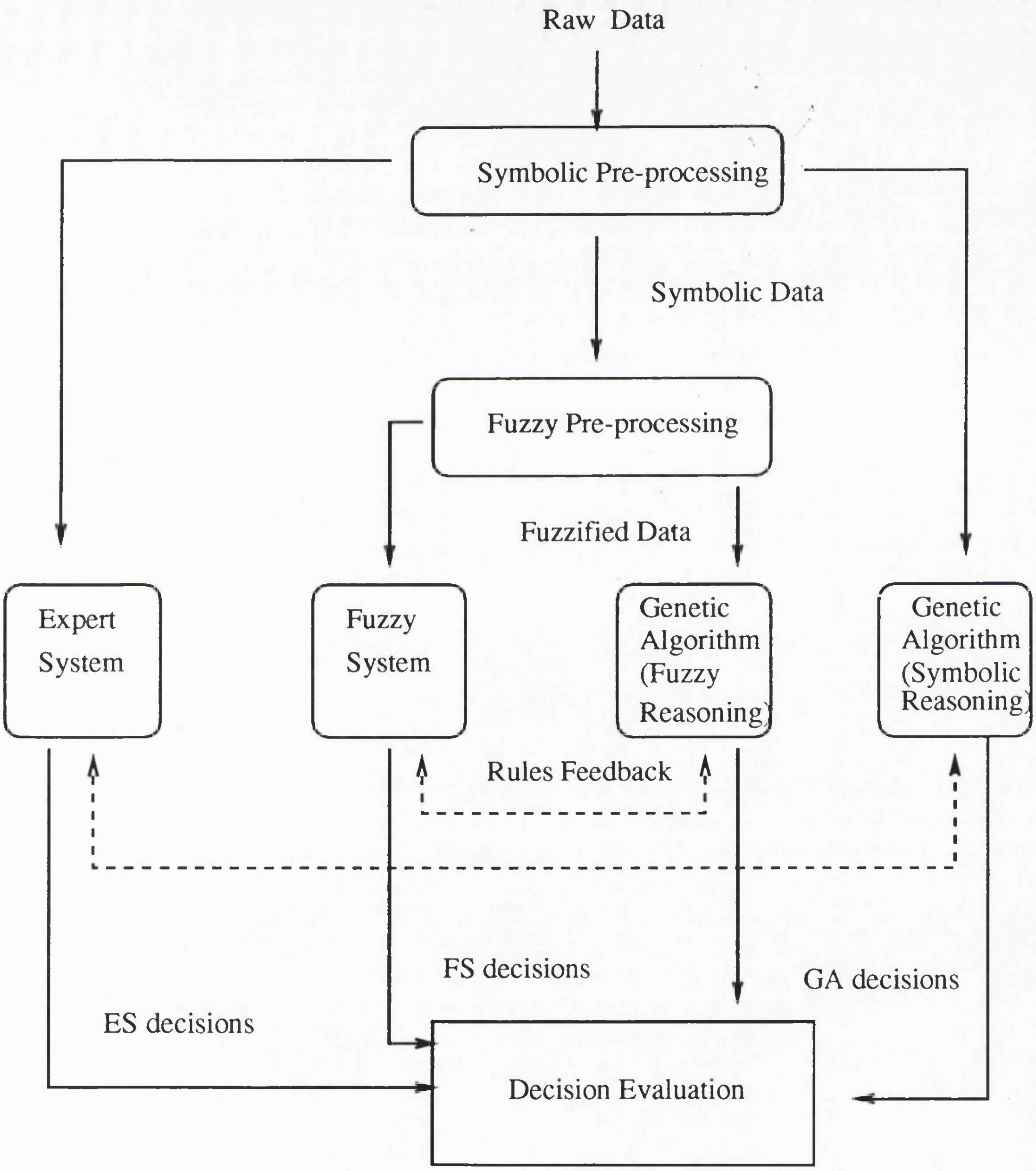


Figure 4.2: The Expert-Fuzzy-Genetic Strand

raw market data (e.g. price 1.54, volume 456875 and open interest 67678), as rules induced will be so specific that these conditions may not recur. Additionally, trading rules elicited from domain experts typically contain linguistic conditions such as *price high, volume low* and not specific numeric price values.

In order to overcome this problem we have developed a method based on a clustering algorithm (the Single Linkage algorithm) to 'symbolise' the raw data. That is, we convert the raw data into groupings of data, where the boundaries of these groupings are determined by the distribution of past data (details are given in chapter 5). The clustering method thus transforms raw market data (e.g. 78789 open interest, 20,000 volume) into 'symbolic' linguistic categories such as '*high* open interest' and '*low* volume' (see figure 4.3).

This cluster selection algorithm also forms the basis of the fuzzy data preprocessing operations which are discussed later.

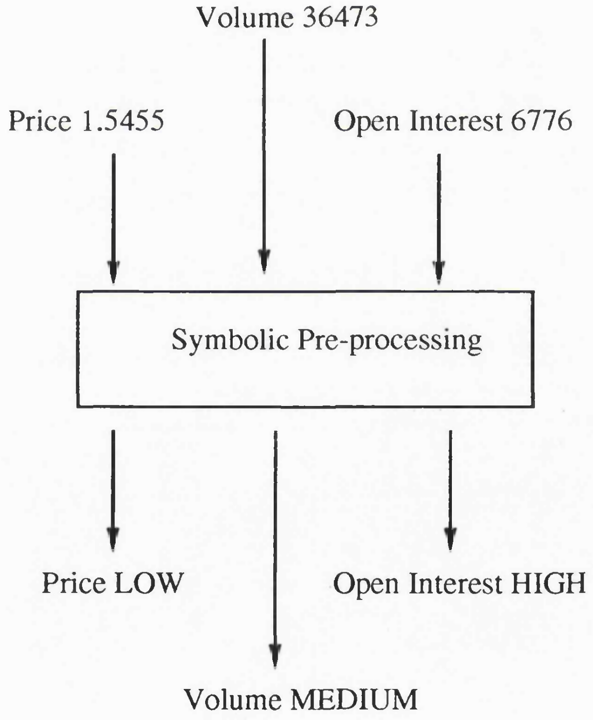


Figure 4.3: Symbolic Pre-processing

The Expert System

The Expert System module stores and manipulates technical trading knowledge obtained from two sources: knowledge elicited from an expert trader and knowledge obtained from books in the application domain. Knowledge elicitation from the trading expert was conducted primarily through a series of interviews which were recorded. The knowledge elicited from both the trader and knowledge contained in technical trading books, falls into six types of technical trading strategies. These are: open interest and volume methods, trading range breakout strategies, moving average strategies, oscillator methods and volatility methods (for further details see chapter 5).

A production rule format was used to represent all trading knowledge in the expert system. A typical technical trading rule in the expert system knowledge base is:

IF the open interest is *high*, the volume is *medium* and the price is *increasing* THEN Buy.

The corresponding representation of this rule in our system is a list structure which contains a list of lists (in POP-11). The above rule will be represented as,

```
[[E17 [open-interest [high]] [volume [medium]] [price [increasing]] [action
[BUY]]]
```

Variables within a rule are always connected by a conjunction (AND) operation, while disjunction (OR) operations apply over the whole rule base. The item 'E17' is an identifier indicating that it is the 17th rule in the expert system knowledge base. The antecedent variables of the rule (e.g. price) are represented as lists and each contains a condition that needs to be satisfied (e.g. increasing). A condition containing an empty list (denoted as []) indicates that a particular variable is not being considered in the evaluation of the rule. The consequent in the rule always has the 'action' string followed by the specified decision in a list.

The Decision Evaluation system

Once the expert system module makes a decision, this decision is then sent to the decision evaluation system to assess its effectiveness. This involves evaluating the buy and sell decisions that the expert system makes. The same decision evaluation module is also used to evaluate decisions made by all of the other processing modules in the INTENT system.

The effectiveness of the decisions will be evaluated against the same criteria traders use to evaluate the effectiveness of their trading decisions. These criteria include the financial gains/losses of each trade, the volatility of the gains/losses, the longest string of good/bad trading decisions, largest gains and largest losses etc. (details of the evaluation process can be found in chapter 5). The adoption of such evaluation criteria allows us to compare the performance of our system with that of human traders and published studies that assess the effectiveness of technical trading strategies.

We assess the performance of the different decision modules by making trading decisions using past price data for two currencies - the British Pound and the Deutschmark. As the actual outcome of the past currency movements are known, the effectiveness of the trading decisions can be evaluated in financial terms and the stability of the decision procedures can be assessed.

Fuzzy data Pre-processing

In order for fuzzy reasoning operators to be applied, the data has to be converted into a suitable format. We have developed a method to produce such 'fuzzified' data by building upon the previously mentioned clustering algorithm that produces symbolic descriptions of data. The fuzzification process can be thought as 'diffusing' the *hard* boundaries that were produced by the symbolic clustering algorithm (for details refer to chapter 6). Now instead of having sudden jumps of memberships between class or linguistic categories, one has a *gradation* function that smoothes the membership from one class to another.

We adopt a normalised fuzzy representation format [5] so that the sum of all the membership values is 1. This allows the subsequent computations to be implemented in an easier manner.

An example of a 'fuzzified' data item is the following representation of volume — the raw figure is 36473 contracts, the symbolic representation is *medium* and the fuzzy representation is,

```
[volume [low 0.2] [medium 0.7] [high 0.1]
```

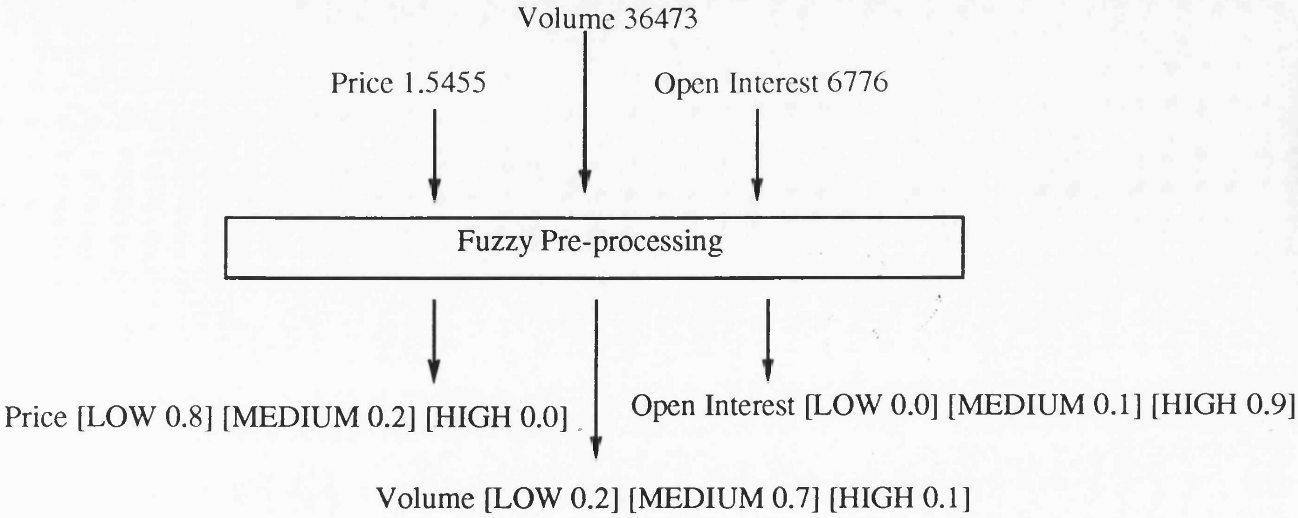

Volume 36473
Price 1.5455
Open Interest 6776
Fuzzy Pre-processing
Price [LOW 0.8] [MEDIUM 0.2] [HIGH 0.0]
Open Interest [LOW 0.0] [MEDIUM 0.1] [HIGH 0.9]
Volume [LOW 0.2] [MEDIUM 0.7] [HIGH 0.1]

Figure 4.4: Fuzzy Pre-processing

This means that this particular value of the volume variable is low with 0.2 confidence, medium with 0.7 confidence and high with 0.1 confidence. Each fuzzified data item is represented as a list of lists where the first item is the name (e.g. volume) followed by the lists which contain the membership name (e.g. medium) and degree of membership (e.g. 0.7).

The Fuzzy System

The fuzzy system is composed of the fuzzy knowledge base containing decision making knowledge, and fuzzy reasoning procedures that act on the fuzzified data. The decision making knowledge is the same technical trading knowledge that is represented in the expert system which was obtained from expert traders and technical books. This knowledge is represented using production rules that have the same syntactic structure (a list of lists in POP-11) as the expert system rules.

Although the decision making knowledge is the same as the expert system, the use of a different representation scheme (fuzzified data as opposed to symbolic data) and a different reasoning scheme (fuzzy inferencing as opposed to symbolic reasoning) produces quite different results.

An example of a fuzzy knowledge base is,

```
[ [F10 [ open-interest [high] ] [price [increasing] ] [action [BUY] ] ]
[ [F11 [ open-interest [low] ] [price [decreasing] ] [action [SELL] ] ]

[ [F12 [ volume [medium] [price [neutral] ] [action [BUY]]]

[ [F13 [ volume [high] [price [increasing] ] [action [SELL]]]
```

The rule F10 means that if open interest is high and the price is increasing then buy. 'F10' is an identifier indicating that it is the 10th rule in the fuzzy knowledge base. The independent variables within a rule are connected by conjunction (AND) relations, and the disjunction (OR) relations apply over the whole set of rules.

We follow Mamdani and Assilian [81], and use their compositional rule of inference to perform the fuzzy reasoning operations. We also use the *centre of area* method [11] as the defuzzification procedure to obtain the final result. The detail description of the implementation of these operations can be found in Chapter 6.

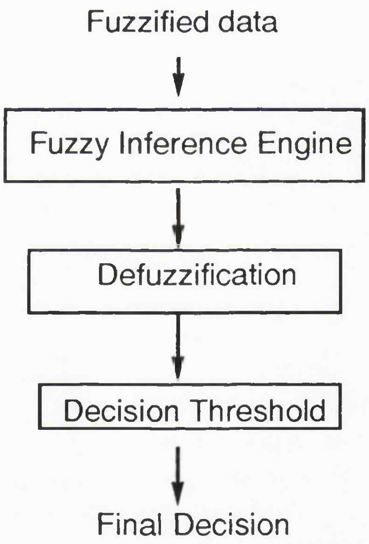


Figure 4.5: Fuzzy Processing Architecture

Once the fuzzy reasoning algorithm produces the final result, a threshold is applied to obtain the final trading decision. Trading decisions generated by the fuzzy system are evaluated by the decision evaluation module applying the same criteria used to evaluate expert system decisions.

The Genetic Algorithm

The genetic algorithm is the primary mechanism for *acquiring knowledge* in the INTENT system. The genetic algorithm induces decision making knowledge in the form of rules obeying the same syntactic format used in the expert and fuzzy systems. It induces rules in two modes: symbolic and fuzzy. In the symbolic mode it induces rules and performs symbolic inferences using the symbolic data analogous to the operation of the expert system. In the fuzzy mode it will induce rules and perform fuzzy inferences using fuzzified data analogous to the operation of the fuzzy system. As the genetic algorithm replaces the *principal function* of *knowledge acquisition* with respect to the expert and fuzzy systems, this hybrid, according to our classification is a function-replacing hybrid.

The genetic algorithm we use is based on Packard's genetic algorithm for complex data analysis [93]. Packard's genetic algorithm attempts to solve complex problems by searching a large space of models to find a set that captures the underlying relationships of the data being analysed. The models are represented by the use of a particular set of *codes*. We substitute a production rule representation scheme for Packard's code representation.

We also extend Packard's system to induce production rules that can operate using symbolic data and reasoning, and also to induce rules that operate using fuzzified data and fuzzy inferencing. In the symbolic mode, the data is pre-processed using the clustering algorithm described earlier. This data is of the same type as used in the expert system. When the genetic algorithm is inducing fuzzy models, it uses data obtained from the fuzzification process described in the fuzzy data-pre-processing section.

The genetic algorithm maintains a population of production rules. The aim of the genetic algorithm is to search through the large space of possible production rules to find good decision making rules. The genetic algorithm, as described in Chapter 2, is an iterative algorithm where each cycle consists of evaluating the models, selecting good models for alteration, and applying the genetic operators crossover and mutation to create new models. This process is repeated until a satisfactory set of decision making rules is discovered.

In the trading decision making application, the task is to find decision rules that will make good trading decisions. This process can be viewed as an optimisation procedure where one attempts to find relationships between combinations of technical indicators (independent variables) and good buy/sell decisions (dependent variables).

In the symbolic mode, the genetic algorithm operates in the following way. Raw trading data and derived trading indicators are firstly 'symbolised' using the above described clustering method. The task of the genetic algorithm is to use these symbolic representations of past trading data to *learn* good trading rules.

Firstly a population of rules is initialised with random values for the production rule conditions. The genetic algorithm evaluates the decisions made by each production rule *separately* — the decisions made by the rules are compared with the known market behaviour and the fitness of each rule is calculated accordingly. The rules are then ranked in terms of their fitness, and mutation and crossover operations are performed to produce new rules (after retaining a small number of the fittest rules). After a number of iterations, the majority of the population will consist of highly effective decision rules.

At each iteration, we select a particular number of the fittest rules, which we refer to as our genetic (symbolic) rule base. This rule base is then sent to the expert system interpreter to perform inferences on the symbolic data. If there is more than one applicable rule for the given data, then this conflict is resolved using the *specificity* strategy mentioned previously (for details see Chapter 7).

An example of a genetically derived rule base is,

```
[[ma-diff-1-20 [positive]] [oi-rsi-14 [high]] [action [UP]]

[volume-diff-1-10 [positive]] [oi-diff-14 [high]] [action [UP]]

[ma-diff-1-20 [negative]] [oi-rsi-14 [low]] [action [DOWN]]

[volume-diff-1-20 [negative]] [oi-diff-14 [low]] [action [DOWN]]

[ma-diff-1-20 [positive]] [oi-rsi-14 [medium]] [action [UP]]

[ma-diff-1-20 [neutral]] [oi-rsi-14 [medium]] [action [DOWN]]

[volume-diff-1-20 [positive]] [oi-diff-14 [medium]] [action [UP]]

[volume-diff-1-20 [neutral]] [oi-diff-14 [medium]] [action [DOWN]] ] ]
```

In the fuzzy mode, the genetic algorithm is used to induce good fuzzy rule bases. The data used for this operation are data that have been 'fuzzified' using the fuzzy data pre-processing algorithm. The same fuzzy reasoning procedures that are used in the fuzzy system are used for evaluating the genetically induced fuzzy models.

In contrast to the symbolic mode where the population members consist of individual rules, in the fuzzy mode the population members are *rule bases*, where each rule base consists of an R number of rules where R is specified by the user. The genetic algorithm therefore evaluates *combinations* of rules rather than individual rules. The rationale for optimising fuzzy rule bases as opposed to individual fuzzy rules is because it is the *interaction* of several fuzzy rules that gives the fuzzy approach its advantage in dealing with brittleness [11]. The Crossover and Mutation operators are modified to account for operations across rule bases (see Chapter 7 for details).

The following is a part of a fuzzy rule base; it consists of three population members each of which contain four fuzzy rules.

```
[ [[ma-diff-1-20-fuzzy [positive]] [oi-rsi-14-fuzzy [high]] [action [UP]]
[ma-diff-1-20-fuzzy [negative]] [oi-rsi-14-fuzzy [low]] [action [DOWN]]
[ma-diff-1-20-fuzzy [positive]] [oi-rsi-14-fuzzy [medium]] [action [UP]]
[ma-diff-1-20-fuzzy [positive]] [oi-rsi-14-fuzzy [high]] [action [UP]] ]

[[ma-diff-1-20-fuzzy [neutral]] [oi-rsi-14-fuzzy [high]] [action [DOWN]]
[ma-diff-1-20-fuzzy [negative]] [oi-rsi-14-fuzzy [high]] [action [DOWN]]
[ma-diff-1-20-fuzzy [neutral]] [oi-rsi-14-fuzzy [high]] [action [UP]]
[ma-diff-1-20-fuzzy [neutral]] [oi-rsi-14-fuzzy [high]] [action [UP]]]

[[ma-diff-1-20-fuzzy [negative]] [oi-rsi-14-fuzzy [high]] [action [UP]]
[ma-diff-1-20-fuzzy [negative]] [oi-rsi-14-fuzzy [low]] [action [DOWN]]
[ma-diff-1-20-fuzzy [negative]] [oi-rsi-14-fuzzy [low]] [action [DOWN]]
[ma-diff-1-20-fuzzy [negative]] [oi-rsi-14-fuzzy [low]] [action [DOWN]]] ]
```

The genetic algorithm cycle in the fuzzy mode operates as follows. The population is firstly initialised with a random collection of fuzzy rule bases. At each iteration, each fuzzy rule base will make fuzzy inferences on fuzzified data. The final results are then passed through a threshold and the final trading decisions are obtained. These decisions are then compared with the known 'best' decisions using the past trading data, and the fitness of rule bases are calculated accordingly. The fuzzy rule bases are ranked in terms of their fitness, and afterwards mutation and crossover operations are performed to produce new rule bases. Over time this procedure produces a collection of highly effective fuzzy rule bases.

The Seeding Mechanism

The genetic algorithm also allows the use of 'seeds' to start off the genetic search process. That is, an expert derived rule or rules (in either symbolic or fuzzy mode) can be used as starting points for the genetic rule discovery process. If one considers the very large search space of possible decision making rules, given the set of variables and all of their possible states, then one can hypothesise that expert specified rules may be *near* a good region of this large search space. It can be argued that the success of experts in a given field is the finding of such good patches or regions in a large decision space through trial and error over a very long period of training and experience. Therefore, by using expert trading rules as 'seeds' to start off the genetic search process, one may arrive at better solutions quicker than by using a random starting point.

Feedback to Permanent Knowledge Bases

We have also implemented a simple feedback mechanism whereby the genetically derived rules can be added to the expert trader specified expert and fuzzy knowledge bases. Such an augmentation of machine derived knowledge with expert derived knowledge is possible because of the common knowledge representation scheme that is being used.

The mechanism we have proposed to do this is as follows: firstly the genetic algorithm induces rules either in the *symbolic* or in the *fuzzy* mode. The rules can then be tested over a test data sample (data outside the training set). Rules that perform well in this test set are then selected for possible feedback to the permanent expert or fuzzy rule bases. Rules from this set that are most *dissimilar* to rules that already exist in the permanent expert or fuzzy rule base are given a higher chance of being added. The heuristic rationale behind such a selection is that it encourages a diverse rule base which may respond well to a variety of different conditions (see Chapter 7 for further details).

4.3.2 The Genetic - Neural Strand

The main knowledge processing components in this strand of hybrid, are the genetic algorithms and neural networks (see figure 4.6). In this hybrid system the genetic and neural components perform distinct tasks and intercommunicate their results, without any function replacement. This hybrid is considered an *intercommunicating hybrid system* according to our classification scheme.

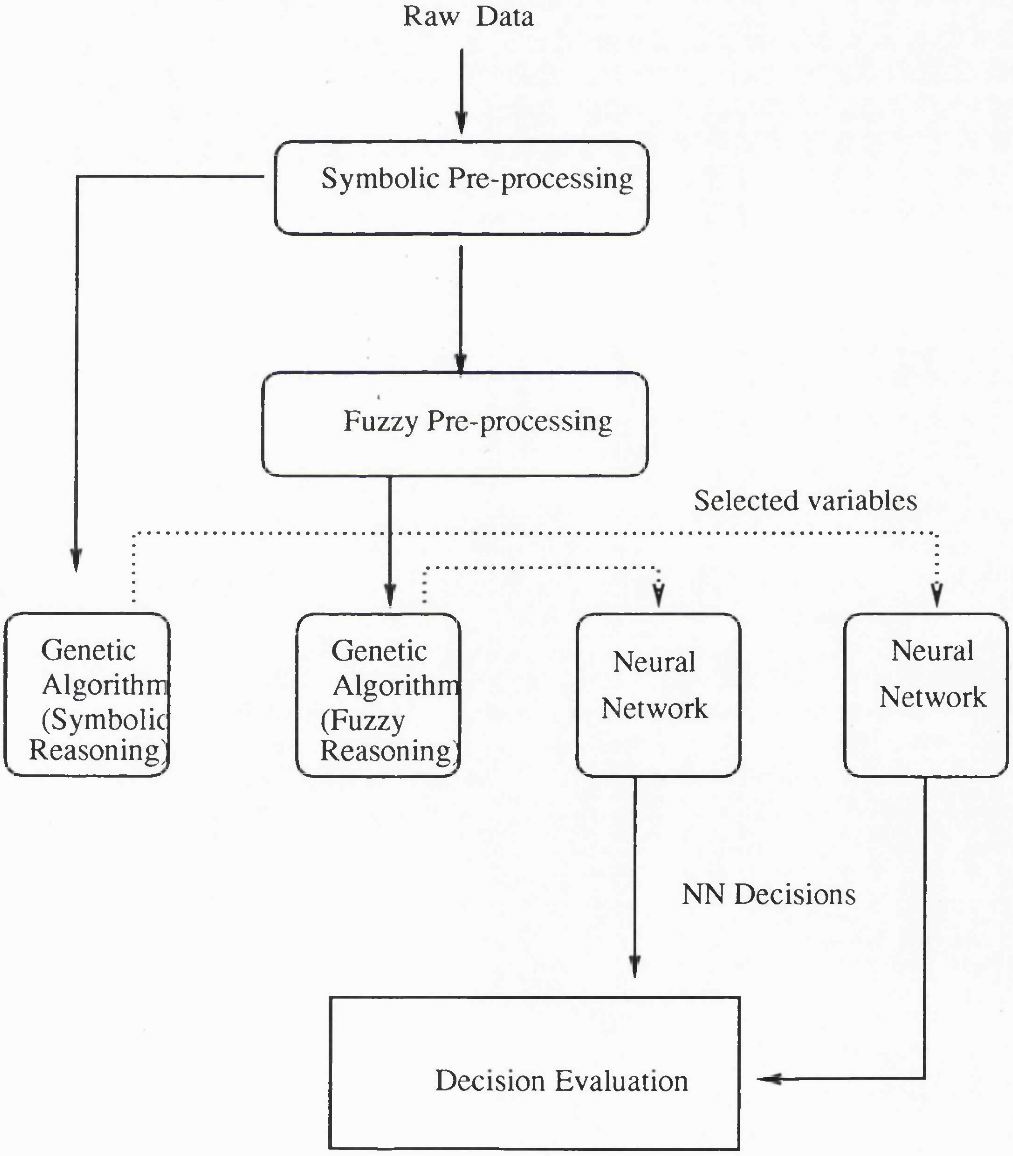


Figure 4.6: The Genetic - Neural Strand

The starting point for this strand of processing is the pre-processing of raw data using the symbolic and fuzzy data pre-processing modules. These stages of processing are the same as described above under the expert-fuzzy-genetic strand.

A central issue in the training of neural network models is the selection of good inputs for the network. Most successful applications of neural networks have involved either judgmental selection of variables or the use of techniques such as principal components analysis to select predictive variables [7], [124]. We use the genetic algorithm in our system as a mechanism for selecting inputs for the neural network. We make an assumption that the variables present in the highly fit rules or rule bases have predictive value as they have been validated against past data and hence are good candidate variables for neural network model building (for details of this selection algorithm refer to chapter 8).

The neural networks, as in the case of the genetic algorithms, are used in two modes : symbolic and fuzzy. In the symbolic mode, the data are in a symbolic format (as used by the expert systems), and the relevant variables are identified from the genetically induced 'fit' symbolic rules. In the fuzzy mode, the data are in a fuzzy format (as used in the fuzzy systems), and the relevant variables are identified from the genetically induced fuzzy rule bases.

As in the case of the genetic algorithm, the neural network is used to learn the mapping between the independent variables (technical indicators) and the optimum decisions (buy/sell) using past trading data. We use a feed-forward multilayer perceptron which uses a backpropagation algorithm [104] to learn the weights in the network.

4.3.3 The Multiple Model Strand

This hybrid strand can be considered as a superset of the two previous strands. In this hybrid model the decisions of all the modules are aggregated (see figure 4.7). That is, it combines the decisions made by the expert system, fuzzy system, genetic algorithm (symbolic), genetic algorithm (fuzzy), neural network (symbolic) and neural network (fuzzy). In our classification scheme this is an *intercommunicating* hybrid as there are separate intelligent processing modules that communicate their results to a central (decision combination) module.

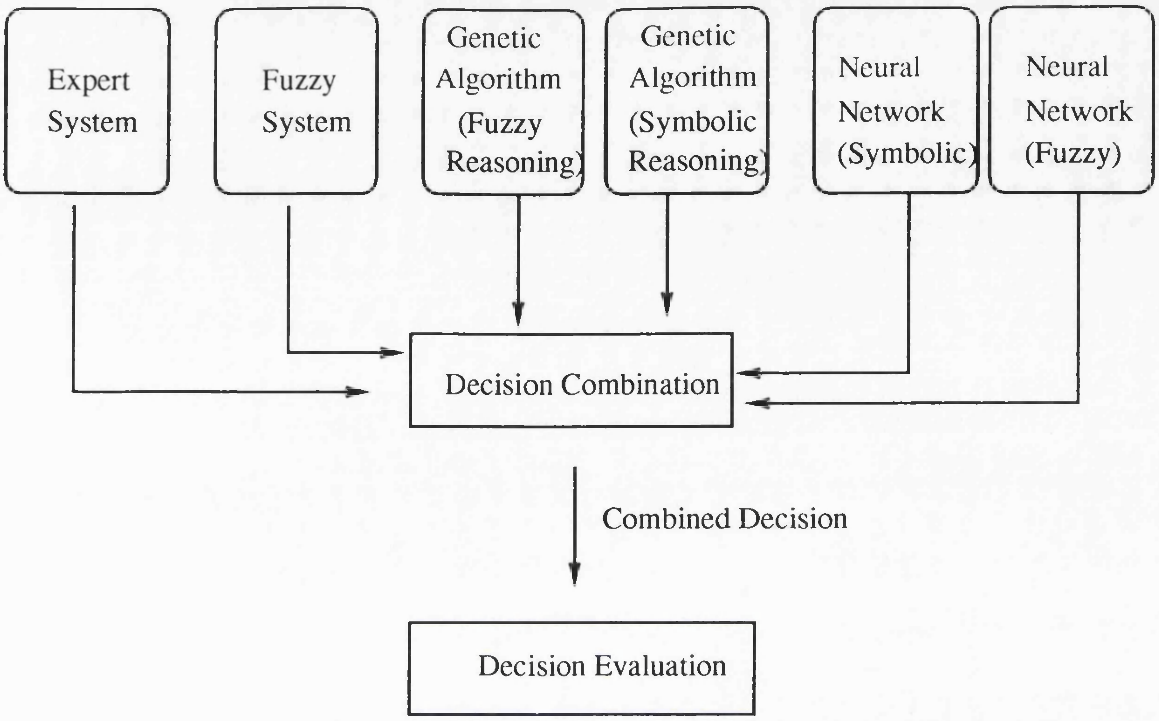


Figure 4.7: The Multiple Model Strand

This hybrid is inspired by the recent evidence suggesting that combining forecasts of several models can yield better results than using results of a single forecasting model alone [3]. The advantages of such combinations is particularly apparent when there is a negative correlation between the models [22].

We use a very simple approach to combining the decisions. The user specifies a *consensus threshold*, C, and only if there are greater than C identical decisions being made by the different modules is that decision executed (see Chapter 9 for details). For example if C is set to 4, then only if at least four of the modules make the same decision is that decision implemented. The higher the value of C the more stringent the criteria becomes with the result of less trades being made for a given data set. We have empirically evaluated the effect of increasing the value of the consensus threshold and its effect on the quality of decisions. In Chapter 9 we also discuss the implications of this decision combination method with respect to the quality of explanation of decisions.

We have been able to use a simple approach for decision combination because the outputs of the different decision modules is of the same type (i.e. either buy or sell). In applications where the modules contain knowledge of different sub-domains and have different types of outputs, more complex approaches to decision combination will be required. In such situations, *knowledge negotiation* methods [16] where intelligent agents inter-communicate their results to arrive at consensus decisions can potentially be very useful.

4.4 Summary

This chapter identified key properties of an 'ideal' decision support system for financial trading and outlined the INTENT system design that attempts to satisfy these requirements. The overall system consists of three strands of hybrid systems which correspond to different types of hybrids according to our classification scheme.

A key consideration in producing a successful hybrid system consisting of expert systems, fuzzy systems and genetic algorithms is the choice of a common knowledge communication platform. A list based production rule format that has a common syntactic structure is used thus facilitating knowledge interchange between the modules.

In the expert-fuzzy-genetic strand, the expert system and the fuzzy system contains rules that have been elicited from expert traders and technical books in the area. While the expert system uses symbolic data and symbolic reasoning, the fuzzy system uses fuzzified data and fuzzy reasoning. The genetic algorithm acts as a knowledge induction system where it induces rules using either symbolic data and symbolic reasoning, or fuzzy data and fuzzy reasoning.

The second strand of the INTENT module, the genetic-neural strand, has been motivated by the need for automated mechanisms for selecting variables for neural decision systems. In this intercommunicating hybrid, variables that appear in highly 'fit' rules induced by the genetic algorithm are used as inputs for the neural networks.

The third strand of the system, the multiple model strand, combines the results of all the different decision modules in the system. This is motivated by recent results indicating the advantages of the multiple model combinations for forecasting tasks. This intercommunicating hybrid aggregates the results of the expert system, fuzzy system, genetic algorithms and the neural networks to produce composite trading decisions.

It has to be stressed that the above hybrid architectures that will be detailed in the next five chapters are general purpose architectures that can be applied to automate many types of non-programmed decision making tasks. For example, it has clear applications in the area of medical diagnosis where linguistic categories are used to describe medical conditions and where the application of both inductive techniques and domain expertise is required. For example in the field of cardiology, Roiger [102] reports the use of linguistic categories such as *moderate, normal, mild* for describing readings of cholesterol, blood pressure, maximum heart rate and others. For all these variables raw data in terms of readings from measuring instruments are also available (e.g. max-heart-rate 180, blood pressure 130, cholesterol 219). The previously mentioned symbolic pre-processing mechanism (detailed in the next Chapter), provides a mechanism to convert such raw instrument readings into descriptions that are congruent to expert medical descriptions.

Once data has been converted into this format then the above described genetic algorithm can be used to induce decision making rules from known past medical case histories. Such an inductive approach also has particular advantages in the medical field as experts with different training backgrounds can often come to completely different conclusions by examining the same data [42]. Rules which have been induced by this genetic algorithm approach can be examined by medical experts and can then be subsequently modified if they do not completely agree with the hypotheses the rules hold.

Hart [56] provides further examples of domains where experts use linguistic concepts such as *good, fair or poor* in their descriptions. In the domain of undergraduate admissions assessment, the domain experts used such qualitative references and they found it difficult to precisely articulate any quantitative definitions of these attributes. In Hart's case study which involves the use of ID3 to derive decision rules, no attempt is made to derive the qualitative descriptions from raw domain data such as exam marks. Instead, the raw data is pre-processed by an expert manually labelling such categories. The architecture we have presented provides a mechanism for both such symbolic pre-processing and the subsequent induction of decision rules.

Chapter 5

The Expert System and Decision Evaluation

This chapter presents: the knowledge elicitation methods used to construct the expert system; the types of trading decision making knowledge elicited and represented; a novel method to pre-process data into symbolic representations; the architecture and conflict resolution strategies of the expert system; the criteria to evaluate performance of trading decisions; and finally test results for simulated trading in two currency markets.

5.1 Knowledge Elicitation

The decision making knowledge for the expert system was elicited from two sources: from an expert trader, and from technical trading books. There was some overlap in the knowledge contained in both sources.

The trader is at a medium-sized securities firm which specialises in trading futures. They trade in a variety of futures contracts including foreign currencies, bonds, stock index futures and oil futures. Trading is conducted primarily in the US, at the Chicago futures exchange, while a smaller proportion of trades are executed through the London Futures exchanges.

The knowledge elicitation from the expert trader was conducted primarily through a series of interviews that were recorded and subsequently transcribed. The knowledge elicitation process was conducted over a period of six months. When we first started the knowledge elicitation process, the first month or so was spent in understanding the basics of trading - the types of contracts traded, how a trade is conducted and who buys and sells, and for what purpose etc. General knowledge of this sort was also obtained from books in the area.

The next stage was an attempt to formulate an approach for making automated trading decisions. The trader uses technical trading knowledge as his primary method to make trading decisions. He believes that all information related to the supply and demand for a commodity is reflected in market indicators of a commodity such as price, volume and open interest.

We therefore examined technical analysis approaches to making trading decisions, and conducted a series of interviews with the expert trader. There were several groups of technical trading strategies, and the expert trader was primarily interested in a group of methods known as open interest and volume methods. In the following sections we detail the different types of trading knowledge that are represented in the expert system.

5.2 Technical Trading Strategies

The expert system contains several commonly used technical trading strategies. These strategies comprise open interest and volume methods, moving average methods, trading break methods, relative strength index methods and volatility based methods. We first describe these trading strategies and then describe the data preprocessing methods used to convert the raw market data into a 'symbolic' format that is congruent to expert descriptions of market behaviour.

5.2.1 Open Interest and Volume Methods

In technical analysis, there is a group of methods which attempt to make trading decisions based on the movement of open interest and volume of commodity [119]. The descriptions of these trading strategies were obtained from the expert trader. They were found to be consistent with descriptions of these strategies in two technical trading books [13], [111].

In the futures markets *Volume* refers to the aggregate number of contracts traded in a given period; it is a measure of the combined market supply and demand for that period [119]. *Open Interest* is the total of purchase commitments outstanding. At any time, the purchase commitments or number of contracts "long" is equal to the sale commitments or number of contracts "short". (see figure 5.1 indicating price, volume and open interest)

The expert trader believes that the changes in the levels of open interest and volume acts as a barometer of the current 'mood' of the market, and can be used to forecast the future price changes.

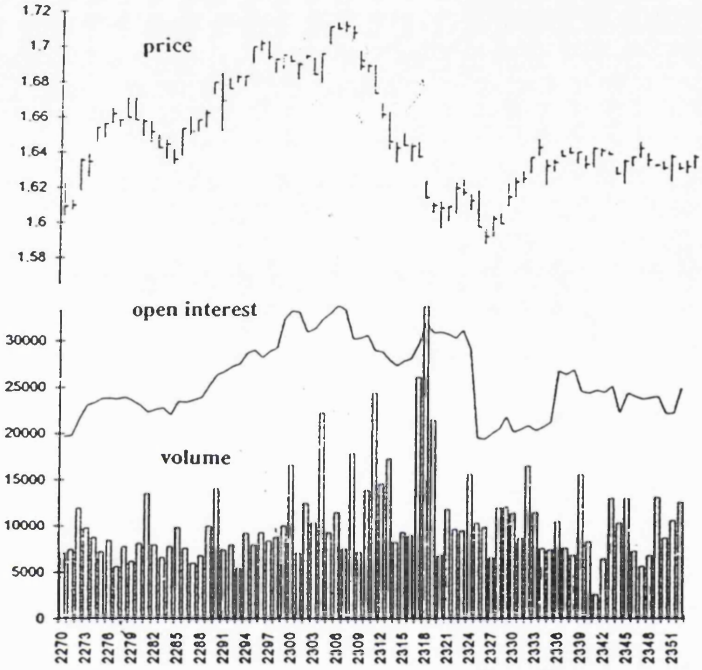


Figure 5.1: Price, Volume and Open Interest for the British Pound

These relationships between price and volume and open interest movements are summarised in a set of rules which the expert trader refers to as the 'rules of open interest and volume'. There are six principal rules. All of these are represented in our expert system using the production rule format (in POP-11) that was introduced in the previous chapter. For a listing of all rules contained in the expert system, refer to Appendix B.

The six rules of open interest and volume are:

(E1) If **Price** is *high* and **Open Interest** is *high* then the market is likely to **rise** (action : BUY)

(E2) If **Price** is *high* and **Open Interest** is *low* then the market is likely to **fall** (action: SELL)

(E3) If **Price** is *low* and **Open Interest** is *high* then the market is likely to **fall** (action: SELL)

(E4) If **Price** is *low* and **Open Interest** is *low* then the market is likely to **rise** (action: BUY)

(E5) If **Price** is *high* and **Open Interest** is *high* and **Volume** is *high* then the market is likely to **rise** (action: BUY)

(E6) If **Price** is *low* and **Open Interest** is *high* and **Volume** is *high* then the market is likely to **fall** (action: SELL)

A complete set of experiments investigating the effectiveness of these open interest and volume strategies to make simulated trading decisions is described in section 5.6.

Note that the expert trader has used linguistic terms such as 'low' and 'high' in describing his decision making strategies. When asked for an exact quantitative definition of these terms, he was unable to provide a precise definition and said that the reference to a particular amount of volume or price being 'high' or 'low' is relative to the past values in price and volume. Our clustering technique provides a method for converting raw market data (e.g. 1.787 and 20,7980) into symbolic terms such as 'low', 'medium' and 'high' (details of this procedure are presented later in the chapter in section 5.3).

5.2.2 Trading Range Breakout

A key idea in technical analysis is the notion of 'support levels' and 'resistance levels' of prices [123]. Technical analysts believe that when a price rises above or 'breaks out' of a particular resistance level then prices will continue to rise. Similarly if there is a 'support level' in the trading range and the price falls through this level, then the price is expected to fall. A simple trading strategy based on this concept is to calculate the maximum (minimum) of the last N trading days, and predict a rise (fall) in the market if the current price is greater (lesser) than the maximum (minimum) [123]. The rules corresponding to these tactics are:

(E7) If the current price $>$ N day maximum then price is likely to *rise* (action : BUY)

(E8) If the current price $<$ N day minimum then price is likely to *fall* (action : SELL)

5.2.3 Moving Average Strategies

Another popular group of technical trading strategies involves tracking the movements of *moving averages* of the prices.

The moving average of prices is given by

$$MA_t = \frac{1}{N}\sum_{i=0}^{N-1} P_{t-i}$$

where N is the number of days, P is the price and MA_t is the moving average on day t.

Generally two moving averages are used - a long period (e.g. the moving average of the last 200 days' prices) and a short period (e.g. the moving average of the last 10 day's prices). The general idea behind computing the moving averages is that they smooth the generally volatile time series, and provide an indication of the general trend of the market [67].

There are several ways [123], [67] of using moving averages for making trading decisions. One type of trading strategy is to execute BUY trades when the short moving average is higher than the long moving average, and to execute SELL trades when the short moving average is lower than the long moving average.

The rules corresponding to these hypotheses are:

(E9) If the **short moving average** is higher than the **long moving average** then the market is likely to **rise** (action : BUY)

(E10) If the **short moving average** is lower than the **long moving average** then the market is likely to **fall** (action : SELL)

A variation of this approach is to execute trades when the moving averages cross each other. With this strategy a BUY trade is executed at the point when the short moving average becomes higher than the long moving average, and a SELL trade is executed at the point when the short moving average becomes lower than the long moving average.

The rules corresponding to these hypotheses are:

(ER11) If the **short moving average** crosses the **long moving average** from *below* then the market is likely to **rise** (action: BUY)

(ER12) If the **short moving average** crosses the **long moving average** from *above* then the market is likely to **fall** (action : SELL)

All the above moving average schemes are essentially based on measures reflecting the *difference* between the values of the two moving averages. We devise a simple measure of this difference,

$$MAdiff = \frac{SMA - LMA}{SMA}$$

where SMA is the short moving average, LMA is the long moving average and *MAdiff* is the measure of difference between the two moving averages. We devise the following trading strategy based on this difference measure:

If the MAdiff is **positive** then the price is likely to **rise** (action : BUY)

If the MAdiff is **negative** then the price is likely to **fall** (action : SELL)

A variation of this rule would be to make trading decisions only when the difference between the moving averages is large. The corresponding rules for this variation are:

(E13) If the MAdiff is **LARGE positive** then the price is likely to **rise** (action : BUY)

(E14) If the MAdiff is **LARGE negative** then the price is likely to **fall** (action : SELL)

5.2.4 Oscillator Methods

It is generally accepted in the technical trading literature that while moving average methods work well when a market is in a definite trend, they perform less well when the market enters an oscillatory state [67], [99]. Techniques known as oscillator methods are generally thought to perform well in such volatile markets. The Relative Strength Index (RSI) is an example of an oscillator type technical indicator. The RSI index is defined as,

$$RSI = \frac{UP.moves.in.N.trades}{N}$$

The RSI index measures whether a market has been 'overbought' or 'oversold'. The principle here is that in an oscillatory state the price will oscillate around a mean value and that if the RSI index is very high, then the market has probably reached its maximum and can be expected to revert to the mean value, i.e. fall. Similarly if the RSI is very low, then the market has probably reached the minimum and probably will rise again.

The trading rules based on the RSI index are:

(E15) If RSI $>$ 0.75 then the market is likely to **fall** (action: SELL)

(E16) IF RSI $<$ 0.25 then the market is likely to **rise** (action: BUY)

5.2.5 Volatility Methods

As described above, the technical trading literature suggests that two different types of methods should be used for dealing with different types of market behaviour - in trending markets to use moving average methods, in oscillatory markets to use oscillator methods [67], [119]. A method to characterise the oscillatory state of a market is the *volatility* of price movements [67]. We define volatility as the

N day standard deviation of the maximum price movement in a day ($high - low$)

We can now devise rules that combine the moving average methods with volatility and also rules that combine oscillator methods with volatility.

The rules that incorporate volatility are:

(E17) If volatility is **low** and short moving average > long moving average then the market is likely to **rise** (action : BUY)

(E18) If volatility is **low** and short moving average < long moving average then the market is likely to **fall** (action : SELL)

(E19) If volatility is **high** and RSI index is high then the market is likely to **fall** (action : SELL)

(E20) If volatility is **high** and the RSI index is low then the market is likely to **rise** (action: BUY)

5.3 Symbolic Data Pre-processing

Many of the previously described trading rules, obtained both from the expert trader and from technical books, use linguistic categories (e.g. low, high, large) to describe market behaviour. We therefore need a mechanism to convert 'raw' market data — price, volume and open interest — into such linguistic symbolic descriptions. We have developed a novel, and relatively simple method based on the use of a clustering algorithm to convert such data into symbolic linguistic descriptions. The algorithm is demonstrated using foreign exchange trading data, but is sufficiently general to be applied to any univariate data set. Data pre-processed through this mechanism are used in the expert system and genetic algorithm in the INTENT system.

The starting point for the symbolic pre-processing method is for the user to specify linguistic 'labels'. These labels are for the symbolic categories into which the algorithm will subsequently classify raw data. Examples of these labels are low, medium, high and small, moderate and big. The linguistic categories should be specified in an increasing order e.g. low - medium - high.

Once the order of the labels are specified, a clustering algorithm is applied to the raw market data. The clustering algorithm used is the Single Linkage Clustering Method (SLINK). The SLINK clustering algorithm is in the family of ***nearest neighbour*** techniques, which iteratively group data points that are nearest to each other into clusters. We chose the SLINK algorithm because it is a computationally efficient clustering procedure [57], especially compared with neural network clustering methods [69], [12]. The issue of computational efficiency is important, as the system needs to cluster a large number of data points (approx. 5,000) relatively quickly. A public domain implementation of the SLINK algorithm written by Stolcke [116] is used for all clustering operations.

A cluster selection algorithm that we have devised (see section 5.3.2) is then used to search the cluster tree for clusters which may correspond to the linguistic categories. Once these clusters are found and their data ranges are obtained, unseen data items can then be classified.

5.3.1 The SLINK Clustering Algorithm

The following is a description of the Single Linkage Clustering algorithm [57] that is used for clustering the data.

The M objects will be arranged in order so that each cluster is a contiguous sequence of objects. The Ith object in this new order will be denoted by $O(I)$ and distance is denoted by D. A *gap* $G(I)$ is associated with the Ith object in the order. These gaps determine the boundaries of the clusters.

Step 1. Let $O(1)$ be any object. Let $G(1) = \infty$.

Step 2. Let $O(2)$ be the object closest to $O(1)$. Let $G(2)$ be the distance between $O(2)$ and $O(1)$.

Step 3. For each $I(3 \leq I \leq M)$ let $O(I)$ be the object, not among $O(1), O(2), \ldots, O(I-1)$. That is, for some $K(1 \leq K \leq I-1)$

$D[O(I), O(K)] \leq D(J, L)$,

where J ranges over $O(1), \ldots, O(I-1)$ and L ranges over the remaining objects. The gap $G(I)$ is set equal tc this minimum distance $D[O(I), O(K)]$.

Step 4. The cluster $O(L1) - O(L2)$, containing objects $O(L1), O(L1+1), \ldots, O(L2-1), O(L2)$ is associated with gap $G(I)$, where $(L1, L2)$ is the maximal interval including I, such that $G(J) \leq G(I)$ for all J with $L1 < J \leq L2$.

5.3.2 Cluster Selection

The clustering algorithm returns a tree structure of the clustered items as a list of lists. See Appendix A for an example of the cluster formation using British Pound volume data. The next task is to select clusters that correspond to the linguistic items. The cluster selection algorithm operates using two main heuristics.

1. The clusters that define the linguistic categories will have a larger number of data points than clusters that do not correspond to the linguistic categories.
2. The clusters corresponding to the linguistic categories will roughly divide the total number of data points among them.

The algorithm for selecting the clusters is as follows:

1. Order the linguistic items in terms of increasing value (e.g. low medium high).
2. Create a *slots list* (a list of lists) where the number of elements equals the number of linguistic categories and initialise these lists with zero elements.
3. Calculate the 'ideal' cluster length. This is computed based on the heuristic that the clusters corresponding to the linguistic categories equally divide all the data points.

   ideal cluster length = total number of data points / number of linguistic categories
4. Start from the root of the cluster tree and compare the slot lists with the clusters,

   IF the cluster has common elements with any of the slots in the slots list, AND the cluster is closer to the ideal cluster length, THEN replace the contents of the slot with the current cluster

   ELSE

   IF there are no common slots THEN,

   (a) Find the worst slot from the slots list (the worst slot is the slot which has a maximal length difference from the ideal cluster length)

(b) IF the current cluster is closer to the ideal cluster than the worst slot, THEN replace the contents of that slot with the current cluster

5. Repeat 4 until the leaves of the cluster tree are reached.

Defining Class Boundaries and Classification

The above described cluster selection process produces a list of clusters which corresponds to the linguistic categories. The numerical ranges of the clusters corresponding to the linguistic categories that will be later used to classify unseen data items.

For example, in the case depicted in figure 5.2, the algorithm has chosen the *medium* cluster with a range between 3100 and 4300, and the *high* cluster with values between 4700 and 5000. An unseen data-item which has a value of 3300 will be classified as being medium and a value of 4900 will be classified as being high.

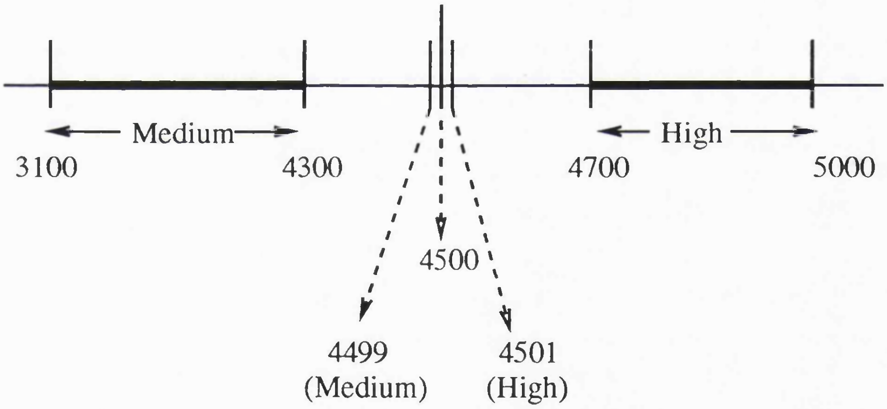


Figure 5.2: The Class Boundaries

For data items falling between the class boundaries we take the following approach: compute the distance between the data-item to be classified and the ranges of the clusters, and assign the linguistic category of the cluster to which it is nearest.

The Symbolic Data File

All raw data that have been converted into symbolic representations are written to a file that we refer to as the **symbolic data file**. The expert system interpreter performs all reasoning operations using data items contained in this file, and not on the items contained in the original 'raw' market data file.

Each entry in the symbolic data file contains the date followed by the symbolic transformations of the price, open interest and volume variables. The last field of a particular entry is called the **current index** which is used in addition to the date as a unique identifier of that day's trading indicator values.

An example of a section of the symbolic file is:

840104 negative negative negative neutral low low DOWN 759
840105 negative negative negative positive low medium UP 760
840106 negative negative negative positive low medium UP 761

Expert Verification of Symbolic Pre-processing

In order to assess the quality and plausibility of the symbolic data pre-processing scheme, the sample results from the clustering method were verified by the expert trader. Data from two variables (British Pound price 100 day moving average, British Pound open interest 10 day moving average) were presented to the expert trader. The samples were chosen from data between 1982 and 1985. For each variable, fifty randomly chosen samples consisting of the 'raw' data items (e.g. 1.432) and the corresponding symbolic representation of that variable (e.g. high) were presented. At each presentation of the samples the expert was asked to comment whether the symbolic representation was accurate.

For the price moving average data, the expert claimed that 44 out of the 50 samples (88%) were ones that he considered as being plausible representations of the data. There were six samples where the expert interpretation differed from the categories produced by our algorithm. From these six samples four were categories that were *adjoining* (e.g. medium as opposed to high, low as opposed to medium).

The open interest data had a similar expert approval where 42 out of the 50 samples (84%) were same as the expert classification. Here all the mis-classified eight samples were from adjoining categories.

However, the expert did have a reservation regarding the usefulness of the symbolic pre-processing method over long time periods. He explained that the linguistic categories induced from a given data set would only be valid for a given number of trading years as the characteristics of the markets change over time. For example, he said that if the linguistic categories such as low, medium and high were induced using volume data over a 3 year period where it had a range between 10,000 and 50,000 contracts then subsequent classifications of data, say over the next 3 years, has to assume that there is a continuity in this volume data range. However, if after (say) 6 years, the range of volume is between 60,000 and 100,000, then the previously induced linguistic categories are not useful in describing the current data.

A potential solution that we proposed is to update the derived linguistic categories periodically using the most recent data. For example, if a trading system based on INTENT is used in real-trading then a suggestion is to update all linguistic classifications every year or so. This simply involves clustering the most recent X years of data and deriving the linguistic categories. Again it is advisable to verify the derived ranges and samples of symbolic categories from these periodic updates through inspection by a domain expert.

5.4 The Expert System Architecture

We adopt the simplest expert system architecture, with two main components: the knowledge base and the interpreter (see figure 5.3). Other architectures are possible, but this was chosen for simplicity and efficiency.

The knowledge base consists of rules representing the previously described technical trading knowledge (see Appendix B for a full listing of the knowledge base).

The interpreter is a simple pattern matcher which matches the conditions contained in the rules in the knowledge base with data values contained in the symbolic data file. In many cases, however, there will be more than one expert system rule whose conditions match the symbolic data. Therefore a *conflict resolution strategy* has to be employed for selecting a single rule for execution.

The conflict resolution strategy we use is the heuristic of 'specificity' [63]. Rules that have a greater number of conditions are more difficult to satisfy, and are therefore preferred to more general rules with fewer conditions. Thus given that one could fire either $P \ \& \ Q \ \& \ R \rightarrow S$ or $P \rightarrow S$, one chooses the former, because it takes more of the current data into account.

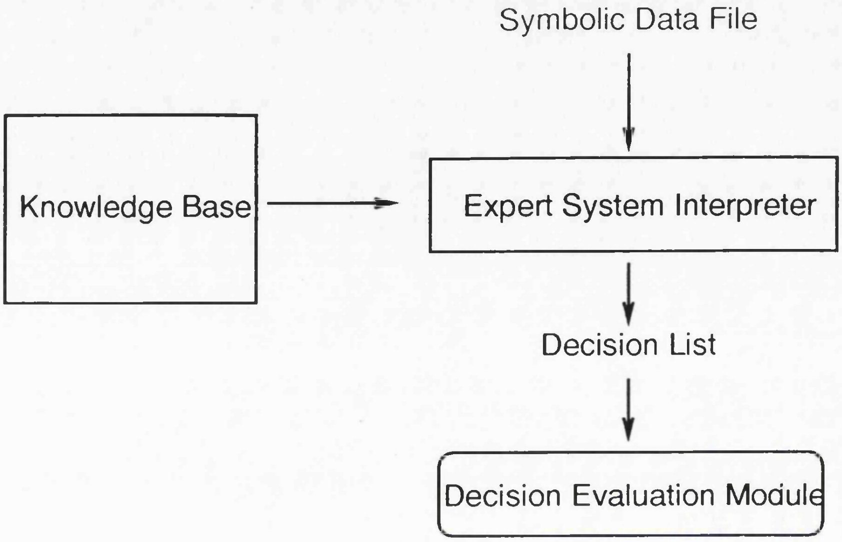


Figure 5.3: The Expert System Architecture

In the following example if both rules A and B fire as their conditions are satisfied, then the rule A is selected as it contains more matching conditions than rule B.

[[A [open-interest [high]] [volume [medium]] [price [increasing]] [action [BUY]]]

[[B [open-interest [high]] [price [increasing]] [action [BUY]]]

If there are two rules with equal specificity, then a rule is chosen randomly.

Finally when a rule is selected, the action specified in the consequent along with its current index identifier is added to a structure called the **decision list**. These decisions are subsequently analysed by the decision evaluation module.

An example of a decision list is:

[[2018 E1 BUY] [2019 E3 SELL] [2020 E1 BUY] [2025 E2 SELL] [2027 E3 SELL]]

5.5 Decision Evaluation

The decisions made by the expert system and other decision modules are evaluated by the decision evaluation module. When using the INTENT system for making financial trading decisions, this module contains several criteria for measuring the effectiveness of trading decisions.

As Babock [10] notes, there is no single measure of evaluating the effectiveness of trading decisions:

> "Like beauty, successful performance of trading systems is in the eye of the beholder. Trader A may prefer the system that makes the most total profits. Trader B may prefer the system that has the highest average profit per trade. Trader C may prefer the system with the highest percentage of winning trades. Trader D may prefer the system with the lowest maximum loss."

We therefore provide a range of performance criteria from Babock [10] and Kaufman [67] to measure the effectiveness of the trading decisions. All simulated trading decisions are made using past data, and the effectiveness of these decisions in financial terms is calculated. When the trading decision is to BUY, a user specified amount of currencies or other traded commodity is 'bought' and is held for user specified holding period. On the date at the end of the holding period, any profits (if the price went up) or losses (if the price went down) are calculated. The reverse is true for SELL decisions.

In order to make the trading decisions more realistic, a commission of 0.01% of the size of the trade is deducted as transaction charges. This figure is comparable to the commissions charged by most trading houses [10] [123].

The following are the trade statistics and evaluation criteria used to evaluate trading decisions. Most of the criteria are self explanatory.

Trading Period: The total length of simulated trading in years.

Total Number of Closed Trades: The total number of trading decisions made in the trading period.

Total Number of Profitable Trades

Total Number of Losing Trades

Percentage of Profitable Trades

Total Profit or Loss: This is total profits or losses made during the whole trading period.

Initial Capital: This is the total capital invested at the *beginning* of the trading period.

Capital after trading: This is the total capital at the end of the trading period.

Average Gains (P/L) per Trade: This is the total profit or loss at the end of the trading period divided by the total number of trades.

Largest Profitable Trade

Largest Losing Trade

Average Profitable Trade

Average Losing Trade

Maximum Drawdown: This has been defined as the total amount lost in the maximum number of consecutive losses.

Percentage of Buy Signals: This is the percentage of Buy trades executed from the total number of trades.

Percentage of Profitable Buy trades

Percentage of Profitable Sell trades

Total Rate of Return (per year): This is annual rate of return on the initial capital.

Standard Deviation of the gains per trade

5.6 Simulated Trading

We assess the performance of the expert system (and other decision modules in INTENT) by performing *simulated* trades using past trading data. Trades are performed using data from two currency markets - the British Pound (US dollars against the Pound) and the Deutschmark (US dollars against the Deutschmark). The raw data for each market consists of the opening price, highest and lowest price for the day, and the closing price. These prices are for the 'spot' cash prices and not the price of the futures contract for these currencies. The total open interest and total volume for the futures contracts trading in these currencies are also used. We used data for both currencies from 1982 to 1992.

Trading indicators (moving averages etc.) are firstly calculated from the data between 1982 to 1985 (see figure 5.4). These calculated indicators are then sent to the clustering algorithm and the ranges for the linguistic categories are obtained. Trading indicators from all data from the beginning of 1985 to the beginning of

1992 are then calculated and then subsequently classified into linguistic categories (symbolic descriptions).

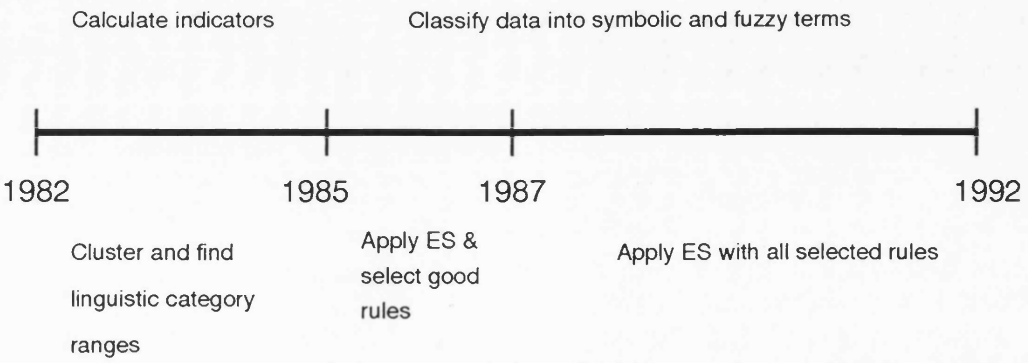

Calculate indicators
Classify data into symbolic and fuzzy terms
1982
1985
1987
1992
Cluster and find linguistic category ranges
Apply ES & select good rules
Apply ES with all selected rules

Figure 5.4: Data Ranges for Processing

Symbolic data from the beginning of 1985 to the beginning of 1987 are used to make the expert system trading decisions. All rules are *individually* used to make trading decisions, and the performance of each rule is evaluated. Rules that have performed well in this period are then selected (by selection criteria described below) and are collectively applied to data between the beginning of 1987 and the beginning of 1992.

5.6.1 Rules and Symbolic Pre-processing

There are a total of forty rules in the Expert System consisting of transformations of the twenty expert system rules (E1-E20) described earlier in this chapter. A full listing of the rules can be found in Appendix B.

From these twenty rules there are an infinite number of rules that can be derived depending on the number of days that are used to calculate the indicators. For example, in Rule E9,

(E9) If the **short moving average** is higher than the **long moving average** then the market is likely to **rise** (action : BUY),

there is a variety of legitimate days that one can use to compute the 'short' and 'long' Moving Averages (MA). One, five, and ten days are commonly used to calculate the short moving average while fifty, one hundred and two hundred days are commonly used to calculate the long moving average. Similar choices have to be made in calculating all indicators featured in the rules E1 to E20.

We have selected a set of configurations for these indicators which were considered reasonable by the domain expert. The full listing of the days used to calculate the indicators and their possible (symbolic) states are given in section B.1 in Appendix B.

5.6.2 Executing Trades

Simulated trading is performed in the following manner. When the expert system makes a trading decision, the trade type (BUY or SELL) and an identification index (current index) is added to the **decision list**. All trades are executed at the opening of the day and the data used for model inputs consists of information up to the previous day's close. [1]

The decision list is then sent to the Decision Evaluation module for evaluation. The holding period for each trade is ten days. All trading calculations start from the initial capital of $100,000, and all returns are re-invested for further trading. In order to make the simulations more realistic, a commission of 0.01% for each trade is deducted. If another trading signal occurs while there is a trade in progress, then any such signal is ignored. This is common practice in the assessment of trading systems [10].

5.6.3 Rule Selection

The criteria for selecting rules from the initial test period (1985–1987) are quite straightforward. There are two criteria which a successful rule must satisfy:

- More than 55% of all trades should be profitable.
- The average gains per trade should be greater than $200.

[1] Open interest and volume are usually reported one day later than the price information. In order to make our trading more realistic, the last open interest and volume figures we use are the ones reported one day before the trading day.

The criterion of at least 55% of trades being correct was suggested to us by the expert trader who thought that this figure, as a rule of thumb, was the minimum for an acceptable trading system. The criterion of the gain of an average trade being at least \$ 200 is suggested by Babock [10] and is determined by considering administrative costs of trading organisations.

5.6.4 Results

British Pound

Simulated trading statistics for all the forty rules (E1 – E40) for trading the British Pound between 1984 to 1987 are displayed in tables C1 to C8 in Appendix C. Rules that have no associated results have not been fired during this period of simulated trading.

The following rules passed the criteria for selection: E1, E10, E19, E20, E22, E25, and E38. That is, only 17.5 % of the elicited rules were valuable enough for further consideration (a more complete discussion of the results is presented in the next section).

These seven selected rules have then been applied *collectively* to unseen data between 1987 and 1992. The results of applying these rules in this period are presented in Table 5.1. The Equity Graph for the selected Expert rules between 1987 and 1992 is displayed in figure 5.5.

Deutschmark

The simulated trading statistics for all the forty rules (E1 – E40) for trading the Deutschmark between 1984 to 1987 are displayed in Tables C9-C16 in Appendix C.

The following rules passed the criteria for selection: E1, E9, E15, E17, E19, E25, E27, E38. That is, only 20 % of the elicited rules were accepted for further consideration

The selected eight rules have then been applied to unseen data between 1987 and 1992. The results are presented in table 5.2 and the corresponding equity graph is presented in figure 5.6.

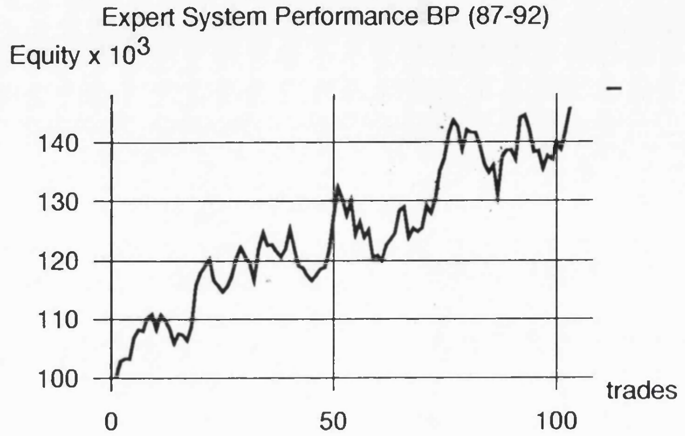


Figure 5.5: BP Expert- Equity Curve Selected Rules (87-92)

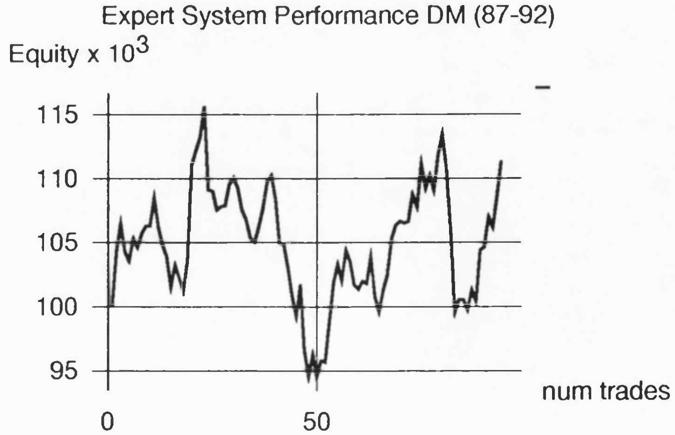


Figure 5.6: Deutschmark Expert - Equity Curve Selected Rules (87-92)

5.6.5 Discussion

Of the seven selected expert system rules trading the British Pound (between 1984 and 1987) the best performing rules (E1, E10, E25) are the strategies that use the moving average ***differences*** as described in section 5.2.3. (see Appendix C for the rule performance tables). The two best rules (E1 and E10) incorporate all three types of technical information of price, open interest and volume. Most Oscillator rules and volatility rules performed poorly apart from one single rule (E38). None of the Trading Break strategies appeared to work with this particular data set.

In trading the Deutschmark (between 1984 and 1987), half of the best performing rules (E1, E9, E25, E27) are, in effect tactics that use the moving average differences. Unlike the case of the British Pound trading, here one Trading Break strategy (E15) performs well, achieving roughly the same percentage of correct trades (68%), as the methods of moving average differences. Here too, on average, the oscillator rules and volatility rules performed poorly.

The important comparison of results should however be made on the performance of all the *selected* rules on *unseen* data from 1987 to 1992. As presented in Table 5.1, results for the British Pound are good with 57.8 % of trades being correct and the average gains per trade being $ 449.5. The results for the Deutschmark using all the selected rules on unseen data (1987-1992) are not as good with 53% of trades being correct and the average gains per trade being $122.2.

We presented these results to the expert trader and asked for any possible explanations of the differences in performance between the British Pound and the Deutschmark. An explanation he offered was that the Deutschmark has been extremely volatile during particular periods in the testing period (1987–1992) as a result of German unification, and that this may have had caused particular indicators to be less effective in predicting the future direction.

There are difficulties in comparing our results with other published studies because other studies have used different time periods for testing. We nevertheless discuss these as it may give a general indication of the general performance of trading systems for foreign exchange trading.

Kaufman [67] obtains an average of 53% correct trades by trading the British Pound (1970–1979) using a modified moving average crossover approach. This same method produces about 52.2 % correct trades when used with Deutschmark data.

Our results for the British Pound are about 4% better and are roughly the same for the Deutschmark.

Babock [10] has tested eight different technical trading methods for trading the Swiss Franc and the Japanese Yen. The primary methods he uses are moving average methods and trading break methods. He does not discuss any experiments on trading of the British Pound or the Deutschmark. Babock's systems produce between 38% and 47% profitable trades for the Swiss Franc and between 33% and 50% profitable trades for the Japanese Yen.

Although Babock's results are very poor with a decision making quality of less than tossing a coin, Babock [10] qualifies these results by referring to the general nature of trend following systems. The claim is that even though trend following systems have low percentages of correct trades, the correct trades they make have significantly larger gains than the small losses that incorrect trades produce. This is typically achieved by using 'stop-loss' mechanisms. An example of a simple stop-loss mechanism is the cancelling of a trade when more than 1% of the equity has been lost. When a trend-following system makes a correct trade the profits are left to accrue, until the trade appears not to make any further progress.

It must be noted that the trading systems methodology to achieve the above quoted results are publicly available. Because of this reason these studies may not be totally representative as there are hundreds of proprietary trading systems whose performance figures and methodologies are not disclosed.

We therefore ideally need further criteria to assess our system's performance. A good criterion of evaluation is a comparison with the performance of a human trader making trading decisions. It is likely that a human trader will draw upon a large reservoir of experience and knowledge to make his decisions, and may use a number of checks and methods that he would not disclose. In Chapter 9 we describe the method and results of such an empirical investigation into human trading decision making. The expert trader is asked to make trading decisions using the same data used by the INTENT system and his performance is measured using the same evaluation criteria. The above expert system results are then discussed with respect to the expert trader performance.

There are several proposed extensions to the symbolic pre-processing mechanism detailed in this chapter. As evident from the expert verification of symbolic data items, in a small number of cases the algorithm does not produce classifications that are congruent to the expert descriptions. In order to overcome these situations the expert can be provided with a 'linguistic data range editor' where the data ranges found by the algorithm can be presented to expert and he or she can adjust the ranges if it is felt necessary. Such a modification facility will be especially useful in areas such as medical diagnosis where experts often disagree on qualitative descriptions of domain data [42].

Another approach to modify the data ranges would be to use inductive techniques to find the optimum data ranges for linguistic categories. As detailed in Chapter 7, a genetic algorithm can be used to induce decision making rules using past (symbolic) data. The genetic algorithm could also be set up to search for optimal adjustments to the symbolic data ranges produced by the clustering algorithm. Such a two level search, adjustments to the definition of the symbolic terms and decision-making rules based on these adjusted terms would, however, be computationally very expensive.

With respect to our three strands of hybrid processing, the expert system module features in two strands: the *expert-fuzzy-genetic* and *multiple model.* In the expert-fuzzy-genetic strand the genetic algorithm is used to induce rules that the expert system can use. A detailed description of the links between the expert system and the genetic algorithm will be presented in Chapter 7.

In the multiple model strand, the expert system results are combined with the results of other modules to produce aggregated decisions. Details of this intercommunicating hybrid and comparisons of expert system performance with other modules in the multiple model strand are presented in Chapter 9.

Trading-Period-Years	5.04
Total-Closed-Trades	102.0
Total-Proft-Trades	59.0
Total-Losing-Trades	43.0
Profitable-Trades-pcn	**57.84**
Total-Gains-$	45856.8
Initial-Capital-$	100000.0
Capital-After-Trading-$	145857.0
Av-Gains-per-Trade-$	**449.58**
Max-Proft-Trade-$	7083.35
Max-Losing-Trade-$	-5780.96
Av-Proft-Trade-$	2292.87
Av-Losing-Trade-$	-2079.59
Maximum-Drawdown-$	-8806.43
Buy-Signals-pcn	71.56
Profit-Buy-Trades	58.90
Profit-Sell-Trades	55.17
Return-per-year-pcn	7.77
Std-Dev-of-PL	2697.15

Table 5.1: British Pound Expert All Selected Rules (87 - 92)

Trading-Period-Years	5.04
Total-Closed-Trades	93.0
Total-Proft-Trades	50.0
Total-Losing-Trades	43.0
Profitable-Trades-pcn	**53.76**
Total-Gains-$	11366.3
Initial-Capital-$	100000.0
Capital-After-Trading-$	111366.0
Av-Gains-per-Trade-$	**122.22**
Max-Proft-Trade-$	7475.17
Max-Losing-Trade-$	-6530.46
Av-Proft-Trade-$	1742.32
Av-Losing-Trade-$	-1761.62
Maximum-Drawdown-$	-13945.4
Buy-Signals-pcn	100
Profit-Buy-Trades	53.76
Profit-Sell-Trades	0
Return-per-year-pcn	2.15
Std-Dev-of-PL	2247.71

Table 5.2: Deutschmark Expert All Selected Rules (87 - 92)

Chapter 6

The Fuzzy System

This chapter presents: the arguments for using fuzzy processing; a clustering based method to derive fuzzy membership functions; the trading knowledge represented as fuzzy rules; the fuzzy reasoning procedures; the defuzzification procedures; and finally, test results for simulated trading in two currency markets.

6.1 The Rationale for Fuzzy Processing

In the previous chapter we discussed a symbolic pre-processing scheme based on clustering to derive linguistic classifications. However, one of the problems that is present in this method of classifying data items is the *sudden shift of class membership*. In the fuzzy logic parlance, the boundaries between the classes is sharp or 'crisp'.

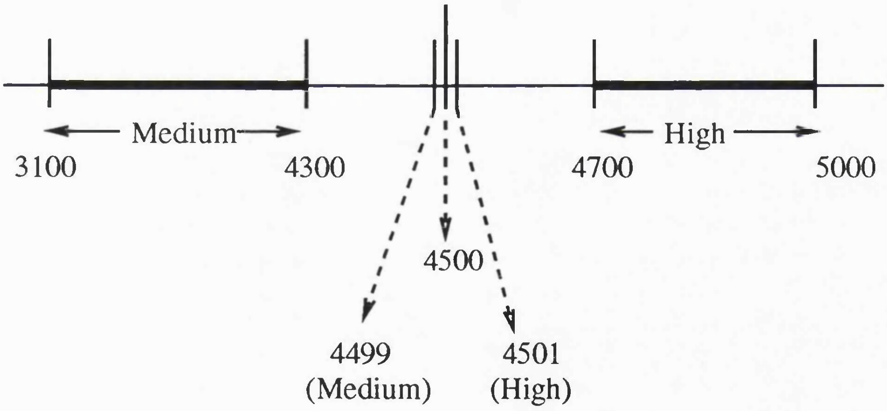


Figure 6.1: The Class Boundaries

To illustrate this, consider the following example where the class **medium** has a range between 3100 and 4300 and the class **high** has a range between 4700 and 5000. Let us consider the situation where a data item falls between these classes. If the data item has a value of 4499 then this item is closer to the 'medium' cluster, and therefore is classified as medium. If it has a value of 4500 then the classification is either medium or high as it is equidistant from both classes. A random choice will have to be made in this case. If value is 4501, then it is closer to the high cluster and therefore it will be classified as high.

Thus, although the data points 4499 and 4501 are very close to each other, they have distinct class memberships. However, from a perspective of expert knowledge, the expert finds no such artificial jumps between these close values. If the expert considers 4499 to be a *medium* value for open interest, the values 4500 and 4501 will also be considered as medium values.

Such 'crisp' boundary definitions are common to almost all expert systems. For example in a medical expert system a production rule may say,

IF body temperature rises > 39 C THEN prescribe the new medicine

For this rule to fire, the condition specified in the antecedent has to be satisfied exactly — even if the body temperature is 38.999 the rule will not fire. This property of requiring all input data to have 'exact' matches with the stored production rules is a manifestation of the brittleness problem described in Chapter 3.

Apart from this anomaly of sudden class-membership jumps between clusters, there is also a mis-match with expert experience in the case of class membership *within* the cluster ranges. Let us take the cluster which has the range from 4700 to 5000 and has been identified as corresponding to the class *high*. In this case a data point which is at the beginning of the cluster range, say 4702, is of the same *intensity* of high as a data point which is 4999. However from an expert's perspective the two data items do not have the same weighting — one data item is *more high* than the other.

Conventional expert systems do not have any machinery to deal with such 'nearby' or 'fuzzy boundary' situations. We therefore turn to fuzzy systems, which can deal with information that has poorly defined boundaries, and investigate their effectiveness in financial decision-making.

Fuzzy Sets, as introduced in Chapter 2, were invented by Zadeh as a response to the exact type of problem described above; to extend clearly demarcated sets to have intermediate or graded memberships. Fuzzy systems provide a mechanism to assign meaning to 'non-exact' information, and a method to reason with such information.

The starting-point for our fuzzy system is the clustering process introduced in the previous chapter, which produces clusters corresponding to specified linguistic categories. The aim of the fuzzy system module is to "fuzzify", or blur the boundaries of these linguistic categories, and to perform fuzzy reasoning on this "fuzzified" knowledge.

6.2 Defining the Fuzzy Sets

The process of defining fuzzy sets is an extension of the symbolic pre-processing procedure described in the last chapter. As stated in Section 5.3.2, the cluster selection algorithm selects clusters that correspond to specified linguistic categories. However as pointed out earlier in this chapter, these linguistic boundaries are 'crisp' and one needs a mechanism to effectively *smooth* these linguistic boundaries. We achieve such smoothness by defining triangular fuzzy membership functions using the cluster ranges as 'anchor points' (described in detail below).

We will illustrate our fuzzy membership definition process starting with cluster ranges obtained from the cluster selection algorithm corresponding to low, medium and high derived from volume (V) data (see figure 6.2).

Let the fuzzy sets of volume, V, be V_1 = *low*, V_2 = *medium*, V_3 = *high*. We define the range of the fuzzy membership functions to be between 0 and 1.

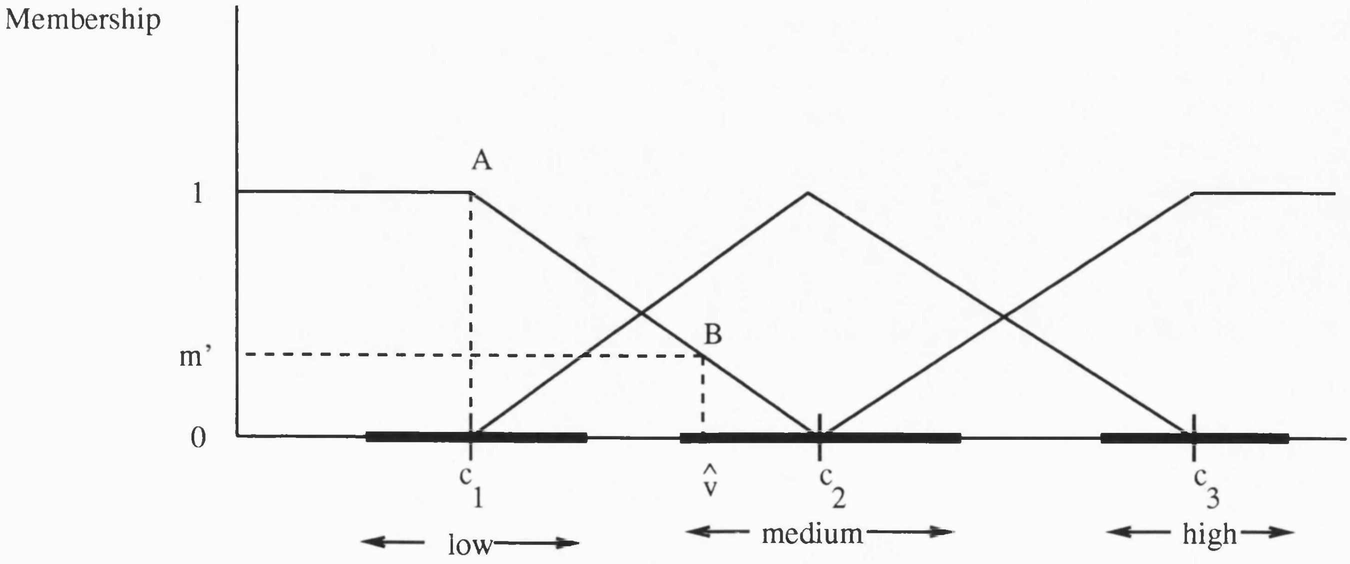


Figure 6.2: The Membership Function Definitions

Let c_1 be the midpoint of the first set V_1 (low), let c_2 be the midpoint of the middle set V_2 (medium), and c_3 the midpoint of the set V_3 (high).

We use the following algorithm for defining the shapes of the fuzzy sets:

If it is the first set, it has a trapezoidal shape where the line opposite the base and the line opposite the right angle take values corresponding to fuzzy memberships. The line opposite the base has a membership $\mu = 1$ where $\hat{v} <= c_1$.

The line opposite the right angle joins the base at the mid point of the next set (c_2). The membership in this section $\mu = (c_2 - \hat{v})/(c_2 - c_1)$.

If the set is a middle set then the membership will be defined as a triangle where the vertex has a membership μ=1 at a point perpendicular to the midpoint. The two sides of the triangle join the midpoints of the two adjacent fuzzy sets. The upward slant is calculated by $\mu = (\hat{v} - c_1)/(c_2 - c_1)$. The downward slant is calculated by $\mu = (c_3 - \hat{v})/(c_3 - c_2)$.

If the set is the last set, then the membership has a trapezoidal shape where the line opposite the right angle joins the base at the midpoint of the previous set and joins the line perpendicular to the base at the mid-point of the set. The membership in this section is $\mu = (\hat{v} - c_2)/(c_3 - c_2)$. The line opposite the base has a membership $\mu = 1$ where $\hat{v} >= c_3$.

6.2.1 The Universe of Discourse and Fuzzification

The *universe of discourse* in fuzzy systems is the total range of values that a particular linguistic variable can take. It is over this range that fuzzy sets are defined. In our method, the minimum value of the universe of discourse corresponds to the minimum value in the cluster with the lowest values and the maximum corresponds to the maximum value in the highest cluster.

We discretise the universe of discourse (UoD) into equal segments to simplify the computations involved in fuzzy reasoning. In the following applications we have discretised the UoD into an arbitrary number of (13) equal segments or bins.

For example, if the variable volume, V, has chosen clusters with a range between 911 contracts to 4491 contracts, then the thirteen corresponding bin ranges are;

```
[911.0--1186 1186--1461 1461--1737 1737--2012 2012--2287 2287--2563 2563--2838
2838--3114 3314--3389 3389--3664 3664--3940 3940--4215 4215--4491]
```

Referring to the fuzzy set definition in figure 6.2, and the above universe of discourse, a volume value of 912 will have a membership of 1 for the *low* fuzzy set, and a membership of 0 for the fuzzy sets *medium* and *high*. Similarly a volume value of 4490 will have a membership of 1 for the *high* fuzzy set. An example of definitions of the degree of memberships for all three fuzzy sets for the above discretised thirteen ranges are:

```
[[low     [1.0 1.0 1.0 0.8 0.6 0.5 0.3 0.1 0.0 0.0 0.0 0.0 0.0]]

[medium   [0.0 0.0 0.0 0.1 0.3 0.4 0.6 0.8 0.9 0.6 0.4 0.1 0.0]]

[high     [0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.3 0.5 0.8 1.0]]]
```

Once the definitions for the fuzzy sets are obtained, classifying a new raw data item into fuzzy values is straightforward. Classifying a new value involves looking up the corresponding range (bin) in the discretised universe of discourse and then looking up the corresponding fuzzy values from the defined fuzzy sets. For example taking the above Volume data and fuzzy definitions, suppose that we need to find the fuzzy values for the data item of 2000 contracts. First the corresponding bin is found in the UoD — the bin range is the 1737–2012 bin (the fourth bin). Then the fuzzy values are obtained by looking up fuzzy sets corresponding to this bin range. These fuzzy values are: 0.8 low, 0.1 medium and 0.0 high. These fuzzy values are represented in INTENT as:

```
[[low 0.8] [medium 0.1] [high 0]]
```

Expert Verification of Fuzzy Values

Similar to the verification of the symbolic data pre-processing scheme by the domain expert, the fuzzy data pre-processing schemes were also verified. Data from the same two variables used in the expert system verification (British Pound price 100 day moving average, British Pound open interest 10 day moving average) were presented to the expert trader. The samples were chosen from data between 1982 and 1985. For each of the two variables, fifty randomly chosen samples consisting of the 'raw' data items (e.g. 1.432) and the corresponding fuzzy representation of that variable (e.g. [[low 0.0] [medium 0.1] [high 0.9]]) were presented to the expert.

The expert remarked that he found the verification of the fuzzy variables more difficult than the verification of symbolic data items. He remarked that although the higher resolution of having weightings in several categories (e.g. [[low 0.0] [medium 0.1] [high 0.9]]) was technically more accurate than having a single category (e.g. high), he was more used to classifying market behaviour using the simpler single category classification.

For each sample presented the expert was asked to make subjective classifications of the raw data into the three fuzzy categories with appropriate weightings. Of the fifty samples of price moving average data, 23 samples had identical weightings to ones produced by our algorithm (in all three categories). Because of the higher resolution in the fuzzy classifications such a relatively low hit rate can be expected. If the tolerance of comparison is increased to within 0.2 of fuzzy memberships, then the expert approval rises to 45 samples out of 50 (90%). Using the open interest data the expert approval (using the 0.2 tolerance) was 41 samples out of 50 (82%).

6.2.2 The Fuzzy Knowledge Base

The fuzzy knowledge base consists of the decision making knowledge that is represented as If–Then rules.

A rule $R_{j,l}$ in the fuzzy knowledge base may be expressed as,

$R_{j,l}$: If v is V_j and p is P_l then cp is $TD_{j,l}$

where $j = 1, ..m, l = 1, ..n$

A typical collection of rules in the fuzzy knowledge base are;

F1. If Volume(V) is high AND Price(P) is low THEN the trading decision (TD) is BUY

F2. If Volume(V) is low AND Price(P) is high THEN the trading decision (TD) is BUY

F3. If Open Interest (OI) is low AND price(P) is low THEN the trading decision (TD) is SELL

We represent such rules using the same syntactic structure used to represent the rules in the expert system. The above rules will be represented as:

```
[ [F1 [ volume [high] ] [price [low]] [action [BUY] ] ]

[ [F2 [ volume [low] ] [price [high]] [action [BUY] ] ]

[ [F3 [ open-interest [low] ] [price [low]] [action [SELL] ] ]
```

The identifier 'F' denotes that the rule is a fuzzy rule.

Defining Membership Functions for the Rule Consequents (decisions)

The fuzzy membership functions for the antecedents (e.g. volume is high, price is low) of the above rules is derived from the clustering-based method described earlier.

The membership functions for the antecedents, the decisions, (buy or sell) are defined heuristically. The trading decisions are defined as having a range of $[-3,+3]$ where the negative values indicate a SELL decision while the positive values indicate a BUY decision (see figure 6.3). A membership function, DO-NOTHING, reflecting the decision not to trade has also been defined. The numerical values indicate the level of confidence of the decision (e.g. -2.9 a definite SELL decision, -0.8 a less definite SELL decision).

The Membership function for Trading Decision (TD) is TD_i:

	UoD	[-3.0 -2.0 -1.0 0.0 +1.0 +2.0 +3.0]
TD_1	SELL	[1.0 0.8 0.2 0.0 1.0 2.0 3.0]
TD_2	DO-NOTHING	[0.0 0.2 0.8 1.0 0.8 0.2 0.0]
TD_3	BUY	[0.0 0.0 0.0 0.0 0.2 0.8 1.0]

6.2.3 Fuzzy Reasoning and Defuzzification

We follow Mamdani and Assilian [81], and use their compositional rule of inference to perform the fuzzy reasoning operations. We also use the *Centre of Area* method [11] as the defuzzification procedure to obtain the final result. These particular methods were chosen because of their success in modelling a range of decision making tasks [83].

The top level algorithm for fuzzy reasoning and defuzzification is as follows:

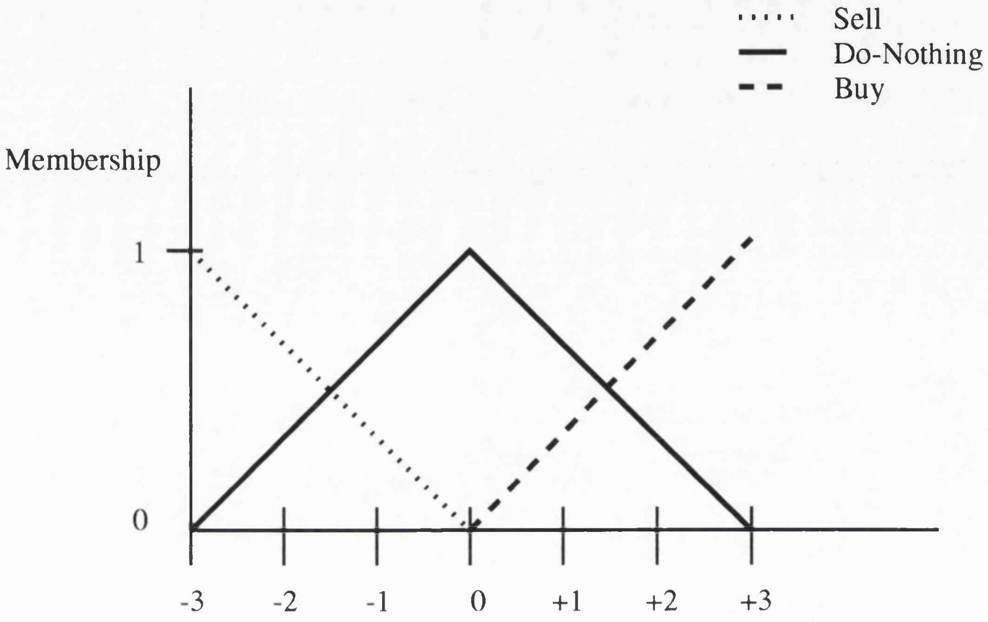


Figure 6.3: Trading Decision Membership Function

1. For a new unclassified data item find the fuzzy membership in the different linguistic terms.
2. Compute the firing strength (α) of each fuzzy rule by finding the minimum membership of the antecedents of the rule.
3. Compute the *rule effectiveness* for each rule by taking the pairwise minimum between the firing strength of the rule and the membership of the rule consequent.
4. Repeat steps 2 and 3 for all rules and create the rule effectiveness matrix.
5. Compute the fuzzy union over the rule effectiveness matrix by finding the maximum value over each column.
6. Defuzzify and obtain the final result by the centre of area method.

The antecedents of rules in the fuzzy system have expressions which are connected through the logical connective AND. The calculation of the 'firing strength' (α) of a fuzzy rule involves finding the fuzzy intersection (AND) of the membership functions of the antecedents. Let μ be the membership of a fuzzy set. The fuzzy intersection (AND) operation of the fuzzy sets V_j and P_l is defined as,

Fuzzy AND = min (μ_{V_j}, μ_{P_l})

The firing strength of a rule α is therefore,

$\alpha_{j,l}$ = min (μ_{V_j}, μ_{P_l})

Let us take an example to illustrate this operation;

F1: IF volume(V) is medium AND price(P) is high THEN trading decision price(TD) is BUY

Let the new data items be $\hat{v} = 2000$ and $\hat{p} = 1.34$. Let their corresponding fuzzy membership values, V_j, P_l be;

V_j, `[low 0.87] [med 0.12] [high 0]`

P_l, `[low 0.1] [med 0.8] [high 0.3]`

The firing strength, *alpha* is,

α_1=`min ([medium 0.12], [high 0.3])` = 0.12

The next step is to compute the effectiveness of the rule, TD_i', by taking the pairwise minimum between the firing strength of the rule (α) and the elements in the membership of the corresponding rule consequent.

Taking the rule F1, the corresponding rule consequent membership function is the SELL membership function. Therefore,

The Rule effectiveness = min(α_1, TD_3)

= min(0.12, TD_3)

= `min(0.12, up [0 0 0 0 0 .2 .8 1])`

= `[0 0 0 0 0 0.12 0.12 0.12]`

The above process of finding the rule effectiveness is repeated for all the three rules in the fuzzy rule base forming the matrix of rule effectiveness.

The rule effectiveness matrix = `[0.0 0.0 0.0 0.0 0.1 0.12 0.12]`

`[0.0 0.2 0.0 0.4 0.0 0.12 0.3]`

`[0.0 0.0 0.3 0.0 0.0 0.5 0.12]`

Each rule in the fuzzy rule base is connected to other rules by a fuzzy union (OR) connective. The fuzzy union operation is defined as:

Fuzzy OR $= \max\ (\ \mu_{V_j},\ \mu_{P_i})$

Therefore the fuzzy OR operation is performed on the rule effectiveness matrix by taking the maximum over each column. Let TD' be the result of this operation.

$TD' = \max(\mu TD) =$ `[0 0.2 0.3 0.4 0.1 0.5 0.3]`

Defuzzification

Since the consequent for the complete set of rules TD' is a fuzzy subset which assigns different membership values to all Trading Decisions (TD), it is necessary to get the crisp value of trading decisions to know whether it is positive (BUY) or negative (SELL). The commonly used ways to get a crisp output value are the i) max criterion, ii) the mean of maximum, and (iii) the centre of area method [75]. We have used the centre of area method which is analogous to finding the centre of mass in a compound object. The algorithm takes the summation of all the fuzzy sets (the rule effectiveness matrix) and finds centre of influence. The centre of area method is defined as:

$$TD_k = \frac{\sum_{i=1}^{n} TD'TD_i}{\sum_{i=1}^{n} TD'}$$

where n is the number of discrete ranges (bins) of the universe of discourse. As mentioned before we have used $n = 13$ in our experiments. The final result is a scalar value which has a range corresponding to the universe of discourse of TD (-3 to $+3$). If the final scalar value is between $+1$ and $+3$ the recommended decision is to Buy , for values around 0.0 do nothing, and between -1 and -3 the recommended decision is to Sell.

Users can define their specific cut-off points. For example, a conservative user may specify values only above $+2.8$ as Buy decisions, while another user may specify values above $+1.5$ as Buy decisions.

In our example, taking the above TD' vector, and the Uod for the consequent TD, the final crisp result (TD_k) will be;

$$TD_k = \frac{(0 * -3) + (0.2 * -2) + (0.3 * -1) + (0.4 * 0) + (0.1 * 1) + (0.5 * 2) + (0.3 * 3)}{0 + 0.2 + 0.3 + 0.4 + 0.1 + 0.5 + 0.3}$$

$$= 0.72$$

6.3 Simulated Trading

We assess the performance of the fuzzy system in the same way as the expert system was assessed — by performing 'simulated' trades using past trading data. We used the same trading data — the British Pound and the Deutschmark — and the same periods for evaluation.

Trading indicators (as in the expert system) are firstly calculated from the data between 1982 and 1985. These indicators are next used by the clustering algorithm to obtain ranges for the linguistic variables. Trading indicators from data from the beginning of 1985 to the beginning of 1992 are then converted to fuzzy values. That is, their fuzzy memberships are calculated using the fuzzy membership algorithm described previously.

Fuzzified data from the beginning of 1985 to the beginning of 1987 are used to make the fuzzy system trading decisions. Twenty eight fuzzy rules are *individually* used to make trading decisions, and the performance of each rule is evaluated. Fuzzy rules which have performed well in this period are selected (using the same criteria used to select expert system rules described in Chapter 5 section 5.5) and applied to data between the beginning of 1987 and the beginning of 1992. In this test period, the fuzzy knowledge base consists of *all* the selected fuzzy rules. That is, during this period the effects of all the selected fuzzy rules are aggregated to produce the trading decisions.

6.3.1 Rules and Fuzzy Pre-processing

There are a total of 28 rules in the Fuzzy System consisting of transformations of the twenty rules E1–E20 described earlier in the previous chapter. A full listing of the fuzzy rules can be found in Appendix B.

6.3.2 Executing Trades

A Buy trade is initiated when the final defuzzified value is greater than +2.2 (the maximum is +3.0). A Sell trade is initiated when the final defuzzified value is less than -2.2 (the minimum is -3.0). These values have been chosen arbitrarily and due to time constraints we have not experimented with different values for these. A possible extension would be to automate the selection of these values by the use of a genetic algorithm using past data.

As in the expert system, when a trade is initiated it is added to the trades list. All trades are executed at the opening of the day and the decision models use information until the previous days' close as inputs. Each trade in the trades list is kept open for ten days and the resulting financial gain or loss on the capital invested is calculated. If another trading signal occurs while there is a trade in progress, then such signals are ignored. For each trade a commission of 0.01% of the capital is deducted.

The criteria for selecting rules from the initial test period (1985–1987) are the same as used to select rules in the expert system which was described in the previous chapter. All trading calculations are done using a $100,000 initial capital, where all returns are re-invested for further trading.

6.3.3 Results

Simulated trading statistics for all the 28 fuzzy rules (F1 - F28) for trading the British Pound between 1984 to 1987 are displayed in tables D.1 to D.6 in Appendix D. The corresponding results for the Deutschmark are displayed in tables D.7 to D.12 in Appendix D.

The following rules passed the above criteria using British Pound data from 1985 to 1987: F1, F5, F6, F10, and F12. That is, only 17.8 % of the elicited rules passed the evaluation criteria. For the Deutschmark data 14.2% of the elicited rules (F9, F13, F15, F26) were acceptable by the selection criteria.

These selected rules have then been applied *collectively* to unseen data between 1987 and 1992. The results of applying the British Pound selected rules in this period are presented in table 6.1 and the Deutschmark results are presented in table 6.2.

The corresponding Equity Graph that can be drawn after applying the selected rules to British Pound data between 1987 and 1992 is presented in figure 6.4 and the Deutschmark graph is presented in figure 6.5.

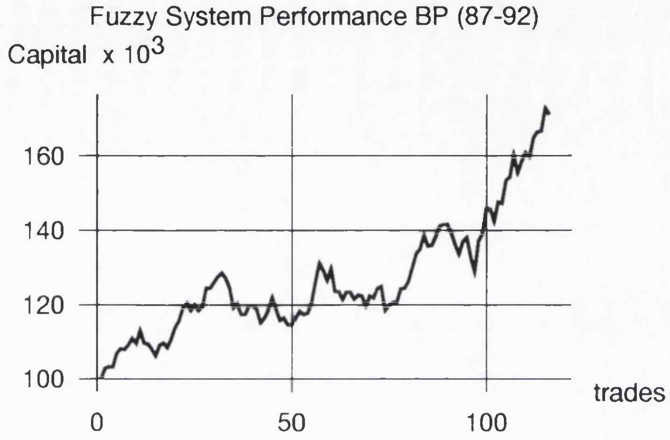


Figure 6.4: British Pound Equity Curve Selected Fuzzy Rules (87-92)

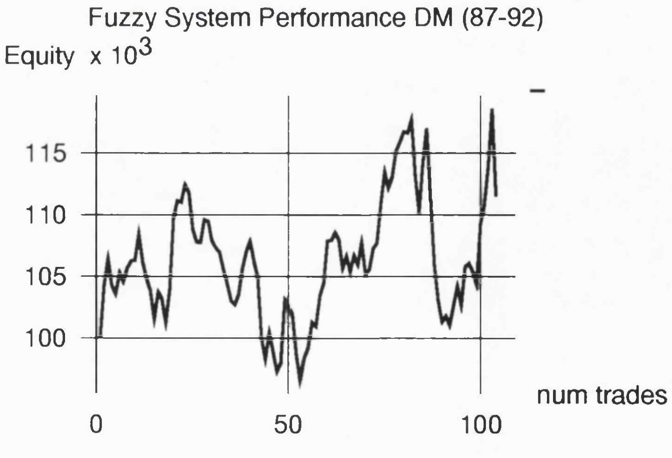


Figure 6.5: Deutschmark Equity Curve Selected Rules (1987–1992)

6.3.4 Discussion

In contrast to the best performing expert system rules, the majority of best performing fuzzy rules (3 out of 5) trading the British Pound (between 1984 and 1987) are rules that use the oscillator type trading approach. There were two selected fuzzy rules (F1 and F10) which were fuzzy versions of the best performing rules presented in the expert system module (E1 and E10). None of the fuzzy rules that incorporate volatility appeared to work with this particular data set. (see Appendix B for a full listing of the selected fuzzy rules)

In trading the Deutschmark, 75% of the selected fuzzy rules (F9, F13, F15) were fuzzy versions of selected expert system rules. And all these three rules used the moving average differences tactic. Here too, on average, the volatility rules performed poorly.

As in the case of assessing the expert system performance, the important comparison of the fuzzy system results should be made on the performance of *all* the *selected* rules on unseen data from 1987 to 1992. As presented in table 6.1, results for the British Pound are very good with 64.34% of trades being correct and the average gains per trade being $618.21. The results for the Deutschmark using all the selected fuzzy rules on unseen data (1987-1992) are less good with 55.2% of trades being correct and the average gains per trade being $140.3. This follows a similar pattern as the expert system results where the British Pound results were generally better than the Deutschmark results. As stated in the previous chapter, a possible explanation offered by the expert trader for these differences was the extreme volatility in the Deutschmark in the test period.

The selected fuzzy rule results were better than the selected expert system results, confirming our hypothesis that the fuzzy system may yield better trading results due to its ability to deal with 'brittleness' as mentioned in Chapter 4. The fuzzy system on average (taking both the British Pound and Deutschmark) was 59.77% correct compared with an average of 55.8% for the expert system. On the average gains per trade the fuzzy system on average yielded $379.25 compared with $285.85 for the average expert system performance.

Unfortunately we cannot compare our fuzzy system results with published studies on fuzzy trading, as there is a scarcity of such studies. The only study we know of on fuzzy trading is by Yuize et al [52], and they do not discuss any performance figures.

However, comparing with Kaufman's [67] expert system studies on foreign exchange trading that were mentioned in the previous chapter, the fuzzy system is better than the 53% rate for the British Pound and the 52.2% rate for the Deutschmark. In Chapter 9 we will again discuss these fuzzy system results in comparison with the performance of a human trader trading currencies over the same test period.

As with the symbolic data pre-processing scheme, the fuzzy pre-processing scheme also has limitations with regard to the usefulness of the fuzzy categories over time. If over time the ranges of the underlying raw data change, then the derived fuzzy categories would not be valid to describe the current data. As with the symbolic data pre-processing scheme a possible solution is to periodically derive new fuzzy membership functions using the most recent data. In order to increase the confidence in these derived ranges these should be ideally verified by a domain expert.

Further, if the expert is provided with a 'fuzzy membership editor' modifications to ranges of the membership functions can be easily made if it is felt necessary. Another extension may be to allow the expert to modify the smoothness of the fuzzy membership functions. We have used triangular functions in our simulations but a choice of Gaussian functions could be provided to an expert within a fuzzy membership editor.

Further research is needed to evaluate whether domain experts are comfortable with the concept of fuzzy membership functions thus allowing the possibility of judgmental revisions. As previously mentioned, our domain expert was not that comfortable in classifying raw data items into weightings of three different linguistic categories. It appears as if the expert has a feel for when a data item is less of (say) *low* and more of *medium* but finds it difficult and tiresome to articulate such descriptions. Further, it is not clear whether experts may have difficulties or preferences in specifying other parameters such as the choice of (say) triangular functions over (say) Gaussian functions.

As with the expert system, the fuzzy system features in two out of the three strands of hybrid processing. In the expert-fuzzy-genetic strand, the *function-replacing hybrid*, the genetic algorithm is used to induce rules that the fuzzy system can use. These mechanisms will be discussed in detail in the next chapter.

The fuzzy system also features in the multiple model hybrid which is an *intercommunicating hybrid* in our classification scheme. Details of this hybrid are presented in

Chapter 9. Here decisions from all the different modules (expert system, fuzzy system, genetic algorithms and neural networks) are aggregated to produce composite decisions.

Trading-Period-Years	5.04
Total-Closed-Trades	115.0
Total-Proft-Trades	74.0
Total-Losing-Trades	41.0
Profitable-Trades-pcn	**64.34**
Total-Gains-$	71094.4
Initial-Capital-$	100000.0
Capital-After-Trading-$	171094.0
Av-Gains-per-Trade-$	**618.21**
Max-Proft-Trade-$	8238.33
Max-Losing-Trade-$	-6494.82
Av-Proft-Trade-$	2252.69
Av-Losing-Trade-$	-2331.81
Maximum-Drawdown-$	-9370.45
Buy-Signals-pcn	75.65
Profit-Buy-Trades	65.51
Profit-Sell-Trades	60.71
Return-per-year-pcn	11.24
Std-Dev-of-PL	2826.26

Table 6.1: British Pound Fuzzy All Selected Rules (1987 – 1992)

Trading-Period-Years	5.04
Total-Closed-Trades	98.0
Total-Proft-Trades	54.0
Total-Losing-Trades	44.0
Profitable-Trades-pcn	**55.2**
Total-Gains-$	13749.4
Initial-Capital-$	100000.0
Capital-After-Trading-$	113749.4
Av-Gains-per-Trade-$	**140.3**
Max-Proft-Trade-$	5768.46
Max-Losing-Trade-$	-7124.83
Av-Proft-Trade-$	1842.57
Av-Losing-Trade-$	-1796.73
Maximum-Drawdown-$	-15748.9
Buy-Signals-pcn	100.0
Profit-Buy-Trades	55.2
Profit-Sell-Trades	0
Return-per-year-pcn	2.5
Std-Dev-of-PL	2320.97

Table 6.2: Deutschmark All selected Fuzzy rules (1987–1992)

Chapter 7
The Genetic Algorithms

This chapter presents: Packard's system for complex data analysis; implementation and extensions to Packard's system; the representation of rules and the genetic algorithm cycle; the symbolic and fuzzy modes of operation; expert knowledge as seeds for the genetic algorithm; feedback to permanent expert and fuzzy rule bases; and finally test results for simulated trading in two currency markets.

7.1 Introduction

In chapter 4, we discussed the required properties of an intelligent trading system; automated knowledge acquisition, ability to cope with brittleness, continuous learning, the ability to produce transparent decision models, and the ability to incorporate expert knowledge. The expert system module described in chapter 5 can provide an explanation of its reasoning and can incorporate expert knowledge. The fuzzy system satisfies three requirements: ability to cope with brittleness, ability to explain and the ability to incorporate human knowledge. The outstanding requirements are the ability to automate the knowledge acquisition process and to enable the system to learn continuously while it is in operation.

The aim of any machine learning procedure is to search a given feature space to find relationships between the *classification variable* (dependent variable) and the *features* (independent variables) [127]. For many real world problems the search space is so large that 'brute-force' exhaustive search techniques will take an unacceptably large amount of time to find a good solution. For example, in our financial trading problem, if one uses an exhaustive search technique, the number of states one will have to explore is 3^{50}. This is calculated by taking 50 variables (50 technical indicators), each having 3 states (e.g. low, high, medium).

A possible route to tackling such combinatorial search problems is to devise techniques that attempt to find *approximate* solutions rather than attempting to find the *optimal* solutions [134]. We have used an approximate reasoning technique, a genetic algorithm, to induce the trading decision-making knowledge. The genetic algorithm essentially finds relationships between the states of the technical trading indicators and 'good' trading decisions.

The genetic algorithm we use is an extension of an algorithm developed by Packard [93] for complex data analysis. We have extended Packard's scheme to allow the induction of production rules and to allow the discovery of fuzzy rule bases (see section 7.3 for more details). Before discussing our extensions, we use Section 7.2 to introduce and detail Packard's original algorithm.

7.2 Packard's System for Complex Data Analysis

Packard's genetic algorithm [93] can be viewed as a *model searching mechanism* which searches a very large space of possible models to find a good set of models that can capture underlying regularities of the given system being modelled. Packard has used this algorithm to optimise the allocation of resources in administrative decision making and has also used the algorithm for forecasting chaotic time series [93].

There are three main components in Packard's modelling system.

1. A code or representation scheme for the models
2. A mechanism to evaluate the usefulness (fitness) of the models
3. A mechanism to generate new models

7.2.1 Data

Let us assume the data to be a collection of pairs $(\vec{x}, y)$,

$((x_1, \ldots, x_n), y) = (\vec{x}, y)$

where each $\vec{x}$ is a set of independent variables (*features*) and where y is the corresponding dependent variable (*classification variable*). Both the independent and dependent variables have to have discrete states, and if the source is continuous the values have to be discretised or 'binned'.

The aim of the algorithm is to search for states of the independent variables, $\vec{x}$, which on the average have a high correlation with particular desired states of y, the dependent variable. The induced patterns will take the form of a set of hypotheses or models, each of the form, *"when some subset of the independent variables satisfy particular conditions, a certain behaviour of the dependent variable is to be expected."* In a market forecasting context, the dependent variable will typically be a future (discretised) state of the system such as 'the market in 10 days (UP or DOWN)' and the independent variables will be (discretised) states of technical trading indicators such as 'Relative Strength Indicator *low*' and 'Volume *high*' etc.

When the algorithm is used in a forecasting context, Packard describes the algorithm as 'searching for pockets of predictability'. It is assumed that the majority of the search space is in fact non-predictable, and the algorithm attempts to fit models to the predictable parts (pockets) of the search space. This non-predictability can be due to external (measurement) noise, dynamical noise (chaotic behaviour), or lack of sufficient data [93]. In contrast to most other learning techniques such as regression, neural networks or other machine learning approaches, this algorithm does not fit a *global function* to the data. Instead it is finding models that work well 'locally' in particular parts or pockets of the search space. The models derived by the algorithm are typically 'incomplete' in the sense that *a prediction is made only if the conditions of the model are satisfied.* Therefore it is possible that for most inputs (values of $\vec{x}$), there may be no prediction as the *conditions of the induced models* would not have been satisfied (see figure 7.1).

In the trading context, one is using the model to discover BUY, SELL trading decisions depending on the states of technical trading indicators. The models or conditional patterns will be derived from using past trading data. Once these models are induced, current market data will be matched against them and only if the conditions specified by the models are satisfied will a trading decision be made. Thus, in fact, most of the time a trading decision-making model derived by this means will not make any decision to trade, but will trade only when it matches a pattern discovered from past trading data.

7.2.2 Representation of Models

The representation of models or *conditional sets* in Packard's system is in the familiar *disjunctive normal form* [127], which specifies relationships between entities in terms of AND, OR relations. A conditional set or model contains as many 'condition positions' as there are independent co-ordinates, n, identifying each of them with one of the co-ordinates. Each 'condition position' will be allowed to take on either a value

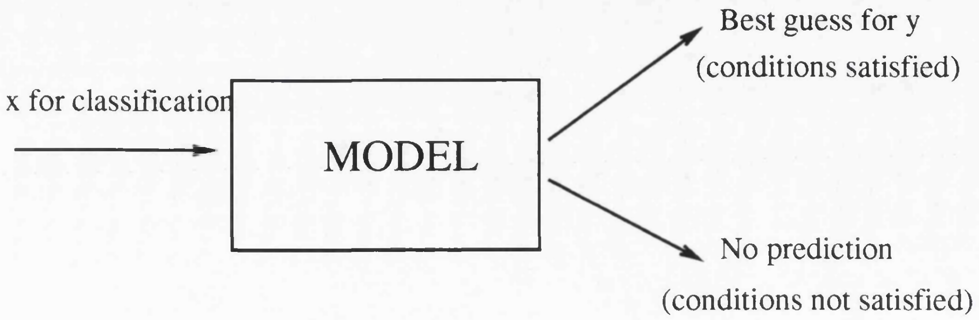


Figure 7.1: Model for Classification

of *, indicating no condition is set for the corresponding co-ordinate, or a sequence of numbers $(c_1, \ldots, c_k)$ indicating OR'ed values of the corresponding co-ordinate. For example,

$$(*, (5,9), *, *, 7, *, *) \sim X_c$$

indicates that the conditional set X_c will be true if the second co-ordinate has a value of either 5 or 9, and the fifth co-ordinate has a value of 7. It will ignore the values of the other co-ordinates.

7.2.3 Searching for Pockets of Predictability

If the aim of the algorithm is to find *good models* or conditional sets, then there must be a mechanism for evaluating the *goodness* or 'fitness' of a given model. This means to find the level of correlation between the states of the independent variables and the *target* dependent variable.

Let N_c be the total number of points in the conditional set X_c (the set of points that satisfy all the specified conditions). We then construct our empirical estimate of the conditional probability distribution of y values given the values of $\vec{x} \in X_c$,where N_c is the number of points in X_c. $\delta(y - y')$ is 1 if $y = y'$ and 0 otherwise.

$$P_c(y) = \frac{1}{N_c} \sum_{(x,y) \in X_c} \delta(y - y')$$

Packard [93] also introduces a 'devaluing' operator to guard against the building of conditional sets that have very small numbers of data points in them, and hence to reduce the effects of statistical flukes. A term proportional to $\frac{-1}{N_c}$ is introduced to achieve this devaluation.

The fitness $F(c)$ of a model or conditional set with the devaluation operation is therefore defined as

$$F_c(y) = P_c(y) - \frac{\alpha}{N_c}$$

where α is a parameter to adjust the dependence on N_c.

7.2.4 The Genetic Algorithm

The mechanism to generate new conditional sets or models in Packard's system is via a standard kind of genetic algorithm. The genetic cycle is:

1. Initialise the population with a random set of conditional sets.
2. Calculate the fitness of each conditional set via the fitness evaluation procedure described above.
3. Discard a fraction of the population with low fitness, and replace the deleted members with alterations of the remaining population, using the genetic operators.
4. Repeat 2 and 3 until good conditional sets are found.

The mutation operators Packard uses are;

1. Picking a new co-ordinate * → (a1,a2)

   $(*,(5,9),*,*,7,*,*) \rightarrow (*,(5,9),*,*,7,*,3)$
2. Deleting a co-ordinate

   $(*,(5,9),*,*,7,*,*) \rightarrow (*,(5,9),*,*,*,*,*)$
3. Changing a co-ordinate

   $(*,(5,9),*,*,7,*,*) \rightarrow (*,(5,9),*,*,2,*,*)$

Crossover operations are performed preserving the positions of the conditional sets. For example, the two conditional sets C1 and C2 may produce the new conditional sets C3 and C4.

$$\begin{array}{ccc} C1(*,(5,9),*,*,7,*,*) & & C3(*,(2,3),*,3,7,*,*) \\ & \rightarrow & \\ C2(*,(2,3),*,3,*,*,4) & & C4(*,(5,9),*,*,7,*,4) \end{array}$$

7.3 Implementation and Extensions to Packard's System

Although Packard has not referred to his system as a Classifier System, architecturally it is very similar to a Classifier System. If one replaces the term 'conditional codes' with *production rules*, one essentially gets a classifier system as described in Chapter 2. A departure from the traditional classifier system [59] is the lack of chains of inferences and the associated credit assignment mechanisms [59]. In contrast, Packard's system evaluates each rule individually, thus shifting the emphasis from finding a '*set* of good classifiers' to finding 'good *individual* classifiers'.

The first extension to Packard's system we have made is to implement it in a production rule form. In implementing we use the same production rule structure used to represent knowledge in the Expert System and Fuzzy System modules. The adoption of a common syntactic structure to also represent the Genetic Algorithm knowledge allows a variety of flexible *knowledge feedback* mechanisms (discussed in detail in Section 7.6).

The second extension to Packard's system is the discovery of *fuzzy* models. The definition of fuzzy rules in fuzzy systems, like the construction of expert systems, is time consuming and error prone. We therefore use the genetic algorithm not only to find good sets of 'symbolic' rules (analogous to expert system rules), but also to find good sets of fuzzy rules. As mentioned in Chapter 6, it is the interaction of *several* fuzzy rules that give the fuzzy approach its power, and therefore we use the genetic algorithm not to find individual rules but to find good *rule bases* or collections of fuzzy rules.

The third extension to Packard's system is the use of the genetic algorithm to augment existing knowledge in permanent Expert or Fuzzy rule bases. Rules which have been induced by the genetic algorithm are compared with existing rules in the expert or fuzzy rule bases, and if the rules are sufficiently *dis-similar* to existing rules then the new rules are added to the respective knowledge bases. This approach can be viewed as enhancing existing expert or fuzzy systems by the addition of new knowledge to the knowledge specified by an expert. This method can be used as a process to add new knowledge (possibly due to changing operating conditions) to an existing expert or fuzzy system while it is in operation.

7.3.1 Rule Representation

An example of our representation is as follows; let our dependent variable (target variable) be *price-in-5-days* with two possible decision states: buy or sell.

For example if we choose three independent variables to be used for the genetic search process — *oi-rsi-14* , *ma-large-1-20* and *price volatility 20* (see Appendix B for an explanation of the abbreviated variable names). *Oi-Rsi-14* has three possible states, [low medium high], *ma-large-1-20* has two possible states, [yes no] and *price volatility* has three possible states, [low medium high].

An example of a representation of a production rule is,

```
[ [oi-rsi-14 [medium]] [ma-large-1-20 [no]] [price-volatility [high]
[action [BUY] ],
```

which says "*If* the oi-rsi-14 is *medium* and ma-large-1-20 is *no* and the price-volatility is *high THEN* BUY."

In order to be consistent with the knowledge representation scheme in the Expert and Fuzzy Systems, each term is connected via a conjunctive (AND) relation only. Disjunctive relations (OR) are not allowed *within* a rule, but are assumed over the set of rules. Again as in the expert and fuzzy systems, if a term has no expression within it, e.g. oi-rsi-14 [], then that term is not evaluated.

7.3.2 The Genetic Algorithm

Our implementation of the genetic algorithm cycle has the following seven standard steps.

1. Initialisation of a population of (random) Rules.
2. Evaluation of fitness of each Rule in the population.

3. Selection of parent Rules for alteration.
4. Creation of new Rules by Crossover and Mutation operators.
5. Deletion of the old rule population.
6. Creation of a new population by inserting altered rules and the fittest rules.
7. Go to 3 until a satisfactory rule(s) is found or a specified number of iterations have been completed.

Rule Initialisation

The variables for each rule in the population (each represented as a list) are initialised with randomly chosen values which are looked up in a reference table (feature-list) which stores each variable's permissible states. For example for the variable oi-rsi-14, the reference table (feature-list) is looked up and from the three permissible states [low medium high], one state is randomly chosen. (Refer to Appendix B for the full listing of the `feature-list`.)

Rule Evaluation

Each rule in the population is evaluated using Packard's fitness evaluation function $F_c(y)$ described in Section 7.2.3. We set the term α to be 15. This means that any rule that has a 'catchment area' of less than 15 data points will be penalised notably more strongly than one with support from more points. This reduces the chances of the algorithm 'mining' small statistically insignificant data pockets.

The following is an example of the fittest rules. Each population member is followed by the number of the member and the fitness value.

```
    [[[price-up-down []] [oi-up-down []] [current-oi-class [low]] [action [BUY] ] 3 0.51935

    [[price-up-down [notrend]] [oi-up-down []] [current-oi-class [low]] [action [SELL]]
30 0.499432]

    [[price-up-down []] [oi-up-down []] [current-oi-class [high]] [action [BUY]] 6 0.486024

    [[price-up-down [notrend]] [oi-up-down []] [current-oi-class [low]] [action [SELL]]
21 0.480996]
[[price-up-down [up]] [oi-up-down []] [current-oi-class [low]] [action [SELL]] 17
0.477792] ]]
```

At this stage of rule evaluation we also select a portion of the *fittest members* from the current population for inclusion in the next new population. That is, all the members are ranked in terms of their fitness and a small number of the fittest rules (without duplicates) are selected.

Parent Selection

Once the fitness for each rule has been established, we use the roulette wheel selection procedure [30] to select the best 'parent' rules for alteration. We have not experimented with any other parent selection schemes, and this particular scheme was chosen because of its proven effectiveness in solving a variety of real world problems [30].

The purpose of this scheme is to give more reproductive chances, on the whole, to those population members that are the most fit. Although this selection procedure is random, each parent's chance of being selected is directly proportional to its fitness. This process over a number of generations will drive out the least fit members and contribute to the spread of the genetic material of the fittest population members. As we later delete the old population, the number of rules we select via this procedure is the total population size (typically 50) minus the number of fittest members chosen earlier.

There are three steps in the Roulette Wheel Parent Selection Procedure [30]:

1. Sum the fitness of all the population members.
2. Generate N, a random number between 0 and the total fitness.
3. Return the first population member whose fitness, added to the fitness of the preceding population members, is greater than or equal to N.

Rule Alteration: Crossover and Mutation

The rules that are returned by the Roulette Wheel procedure are then altered via the crossover and mutation operators to create new rules.

The crossover operation swaps the conditions of rules with other conditions of other rules, at the same conditional locations. The crossover rate (Cr) which is set by the users determines the probability of a crossover operation occurring at a particular conditional point.

If there are two population members (chromosomes) c1, c2,

```
c1 :  [oi-rsi-14 [high]] [ma-large-1-20 [yes]] [action [BUY] ]

c2:  [oi-rsi-14 [low]] [ma-large-1-20 [no]] [action [SELL ]
```

the effect of crossover can result in forming the following two new rules C3 and C4;

```
c3:  [oi-rsi-14 [low]] [ma-large-1-20 [yes]] [action [SELL] ]

c4:  [oi-rsi-14 [high]] [ma-large-1-20 [no]] [action [BUY] ]
```

There are three mutation operators in the system. The probability of a mutation operation being applied is determined by the user-specified mutation rate M. The three mutation operators are:

1. Picking a new co-ordinate

   `ma-diff-1-20 []` → `ma-diff-1-20 [negative]`

2. Deleting a co-ordinate

   `ma-diff-1-20 [positive]` → `ma-diff-1-20 []`

3. Changing the value of a co-ordinate

   `ma-diff-1-20 [negative]` → `ma-diff-1-20 [neutral]`

7.4 The Symbolic Mode

Our genetic algorithm operates in two modes: the symbolic mode and the fuzzy mode. We shall now discuss the way the symbolic mode operates and Section 7.5 will detail the operation of the fuzzy mode.

In the symbolic mode, the data that are used for inducing decision-making rules are 'symbolic' data as was used in the Expert System. These symbolic data items were derived by the clustering algorithm explained in Chapter 5.

The reasoning method used in the symbolic mode of the genetic algorithm is also the same as used in the expert system. That is, *all* the conditions present in the antecedent of a rule have to be matched **exactly** with domain data (data in the symbolic file) for the rule to be fired and for the decision specified in the rule consequent to be executed.

7.4.1 Complexity Fit Estimation

As with any rule induction algorithm, the appropriate *complexity fit* [127] for the genetic algorithm has to be found. Typically in decision tree and rule induction algorithms, too many nodes or rules with too many conditions may overfit the data, leading to low training set errors but high true error rates. Similarly, in the case of neural networks if there are too many hidden units there is an increased flexibility in the network to fit the data. The aim should therefore be to select the right complexity of the rule base or network which has captured the characteristics of the data but has not overfitted the training data.

The genetic algorithm finds new 'fitter' rules as it proceeds through the processing cycle. As the algorithm iterates, the rules it finds will be increasingly specific to the training set and will perform poorly on data outside the training set. We follow Weiss [127] and use a train-and-test approach to find the correct complexity fit. Weiss suggests dividing an available sample into three data sets: train, validation and testing. At each iteration, classification error rates (training set error and validation set error) for the decisions made by the whole rule base is obtained. The error measure we use is the percentage of correct trading decisions.

As the rules get increasingly specific to the training set, the error on training set continues to fall, while the error in the validation set falls and at some point begins to increase again. Weiss [127] identifies such turning points as having the right level of complexity for solving the problem at hand. Figure 7.2 contains a graph depicting the errors on the training and validation sets. Although the error rates should be ideally calculated using a N-fold cross-validation approach, we have used a single train-and-test partition because the genetic learning procedure is very time consuming.

The selected rule base at error turning points is subsequently tested on completely unseen data. Decisions made by this rule base is then evaluated by the decision evaluation module.

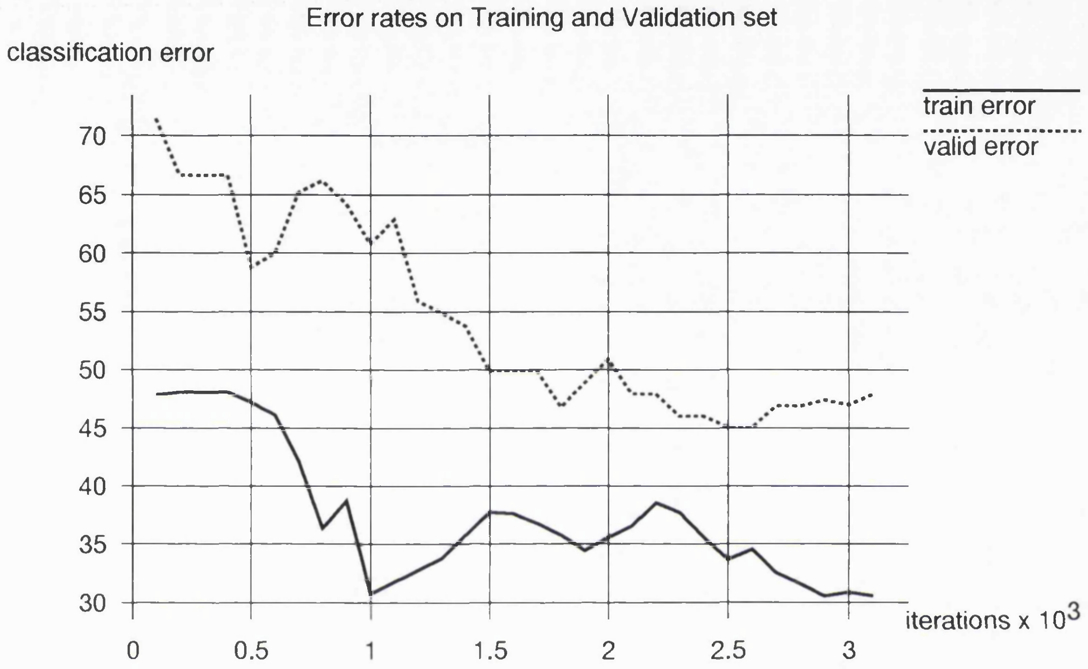


Figure 7.2: Classification error on the training and validation set

However, it must be noted that this method for determining the stopping condition is not perfect. If one looks at figure 7.2 one can see that the turning point is not entirely clear. There are at least two points in the graph where there are such turning points. We have used an entirely 'visual' approach to selecting such turning points after producing graphs of these error curves. As the genetic rule discovery approach is computationally intensive, one does not have the luxury (at least with the current implementation and hardware) to run for a much larger number of cycles than we have explored in this part of the work.

7.5 The Fuzzy Mode

In Chapter 6 we have examined the use of fuzzy reasoning mechanisms for trading decision making. As we mentioned in that Chapter, the primary method for constructing fuzzy rule bases is by eliciting rules from domain experts, which is expensive and error prone. Recently there have been several investigations into the use of neural networks for learning fuzzy rules from domain data [83]. Our approach can be viewed as a similar fuzzy rule extraction method, but with the difference of using genetic algorithms. This we believe is an important extension to Packard's system.

The data the system uses in this mode is 'fuzzified' data (using the algorithm in section 6.2) and the reasoning procedure it uses is the fuzzy reasoning procedure introduced in Chapter 6 (the compositional rule of inference). There is a very large number of possible fuzzy models that can be constructed using a given set of variables. We use the genetic algorithm as a mechanism to search through the very large space of possible fuzzy models (rule bases) to find a set of good fuzzy models (rule bases). As described in Chapter 6, it is the *aggregation* effect of **all fuzzy rules** (all of which may partially match the data) that results in the fuzzy system's ability to deal with brittleness. Therefore, unlike in the case of the previous symbolic mode where we use the GA to find good *individual* rules, we now use the GA to find good *collections of rules* (*fuzzy rule bases*).

The genetic algorithm representation is different in this mode in that it now has a population of *rule bases* as opposed to a population of rules. The aim of the algorithm is to find good rule bases consisting of good predictive rules.

An example of two members , GF1 and GF2 (each rule base having 4 rules) of a rule population are :

```
[GF1 [ma-diff-1-20-fuzzy [positive]] [oi-rsi-14-fuzzy [high]] [action [UP]]
[ma-diff-1-20-fuzzy [negative]] [oi-rsi-14-fuzzy [low]] [action [DOWN]]
[ma-diff-1-20-fuzzy [positive]] [oi-rsi-14-fuzzy [medium]] [action [UP]]
[ma-diff-1-20-fuzzy [positive]] [oi-rsi-14-fuzzy [high]] [action [UP]] ]

[GF2 [ma-diff-1-20-fuzzy [positive]] [oi-rsi-14-fuzzy [high]] [action [DOWN]]
[ma-diff-1-20-fuzzy [negative]] [oi-rsi-14-fuzzy [high]] [action [DOWN]]
[ma-diff-1-20-fuzzy [neutral]] [oi-rsi-14-fuzzy [low]] [action [UP]]
[ma-diff-1-20-fuzzy [neutral]] [oi-rsi-14-fuzzy [high]] [action [UP]]]
```

The evaluation of each member is performed as follows. Each rule base is firstly sent to the fuzzy reasoning algorithms in the fuzzy system. There the compositional rule of inference is applied and the final crisp result is obtained (refer to Chapter 6 for details). As defined by the universe of discourse of trading decisions, this result will have a range $[-3, +3]$ (values close to -3 indicate a SELL decision, values closer to $+3$ indicate a BUY decision). We then apply the same threshold as used in the fuzzy system to infer the final decision (values < -2.2 SELL, values $> +2.2$ BUY).

After this, the Packard's fitness evaluation procedure (as detailed in section 7.2.3) is applied and the fitnesses of the rule bases are computed. That is, for each fuzzy rule base the fitness of its decisions are calculated where N_c are data entries that return values beyond the specified fuzzy threshold (values < -2.2 SELL, values $> +2.2$ BUY).

We use the same complexity fit estimation procedure used in the symbolic mode, where errors in a validation set are monitored. At the error turning points the fittest fuzzy rule base is selected and then applied on unseen data. The resulting decisions are then evaluated by the decision evaluation module.

The crossover and mutation operators have been modified to take account of the different structure of the population members. Crossover can occur both within an individual rule base as well as with members that are in a different rule base. The mutation operation is unchanged with the difference that the mutation probability rate is now applied to a rule base rather than an individual rule. The fitness ranking and parent selection mechanisms are the same as in the Symbolic Mode detailed in Section 7.4.

7.6 Feedback Mechanisms

There are two main methods by which the genetic algorithm interfaces with the expert and fuzzy systems (see figure 7.3). The first is the use of genetically derived rules as additions to the expert and fuzzy systems. The second involves presenting expert or fuzzy rules as 'seeds' for the genetic algorithm.

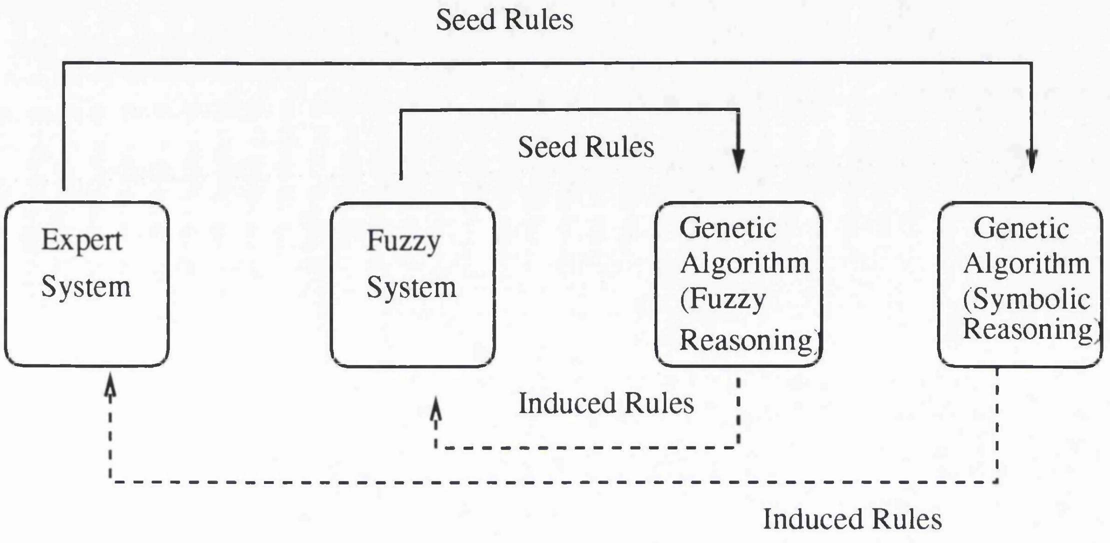

Seed Rules
Seed Rules
Expert System
Fuzzy System
Genetic Algorithm (Fuzzy Reasoning)
Genetic Algorithm (Symbolic Reasoning)
Induced Rules
Induced Rules

Figure 7.3: Genetic Feedback Mechanisms

7.6.1 Feedback to Expert and Fuzzy Systems

The first feedback mechanism is the transfer of genetically derived rules to the expert and fuzzy systems. If one views the expert and fuzzy systems as being 'permanent' knowledge bases, then this method opens a way to add new genetically discovered knowledge. In this sense the mechanism can be seen as a method that can augment human derived knowledge with machine generated knowledge. From our hybrid classification perspective (set out in Section 3.3) this mechanism is a *function replacing hybrid* where the genetic algorithm replaces the function of domain experts in specifying knowledge for expert and fuzzy systems.

This process can be used in either the symbolic mode or fuzzy mode. If the mode is symbolic, then rules which are selected (the selection criteria is described below) augment the expert system knowledge base. In the fuzzy mode, genetically derived fuzzy rule bases are used to augment the knowledge base in the fuzzy system.

The mechanism of rule selection is, however, common to both modes. Firstly, GA induced rules are selected by the same cross-validation method described earlier. The rules are then tested over unseen data. The rules that perform well in this test set are then collected for possible feedback to the permanent expert or fuzzy knowledge bases. The additional criterion for selecting rules from this set of rules is that rules that are most 'dis-similar' to the permanent expert or fuzzy rule base should have the highest chance of being added. The heuristic rationale behind such a selection is that it encourages a diverse permanent rule base which may respond well to a variety of different conditions.

The mechanism for detecting similarity is implemented as follows:

1. Take two rules, one from the permanent expert or fuzzy rule base and the other a genetically induced one.
2. Consider conditional positions in the two rules. If the conditions match, increase the similarity count by one.
3. Repeat Step 2 for all the rules in the permanent rule base and sum all the similarity counts for all rules.
4. Repeat Steps 2 and 3 for all the selected genetic rules.
5. Put the genetic rules into a decreasing order of similarity as assessed by our counting scheme for determining similarity.
6. Add the rule(s) with the lowest similarity counts to the permanent knowledge base.

A full example of this rule comparison mechanism is presented in section E.1 in Appendix E.

7.6.2 Expert Knowledge as Seeds for the Genetic Algorithm

A mechanism that can potentially speed up the genetic search processes in general is the introduction of good 'seeds' or starting points for the algorithm. Powell [34] have demonstrated this in the domain of engineering design optimisation where they have used expert knowledge (in rules) as starting points for the algorithm.

We take a similar approach to Powell and use expert rules as seeds for the genetic algorithm. Our genetic algorithm is essentially searching through a very large decision space, and if expertise corresponds to good solutions, i.e. good regions of this space, then a search process starting from such regions may have a better chance of finding good solutions.

As we have used the same syntactic rule structure throughout the system, it is easy to use either expert system rules (when the GA is in the Symbolic Mode) or fuzzy system rules as seeds (when the GA is in the Fuzzy Mode).

This process of using expert derived knowledge as seeds for the GA can also be used as a *knowledge refinement* process. That is, an expert decision maker can input knowledge in the form of rules and allow the GA to find rules that are similar (near the same region in the decision space) but which are potentially better than the original rules.

This type of facility can be very useful if the decision maker wants to refine 'hunches' about possible relationships. The decision maker can input a particular rule and the genetic algorithm will then evaluate it and modify it over many cycles and may produce a rule that does a better job of capturing the behaviour that the decision maker is looking for.

The exact mechanism of introducing an expert rule into the GA is very simple. It is to include the expert rules as a member(s) of the initial population. If there are N population members and E expert rules, then the initial population consists of $(N - E)$ randomly initialised rules plus E expert rules.

7.7 Simulated Trading

As in the previous modules, simulated trading is carried out using data from two currency markets — the British Pound and the Deutschmark.

The training data sets consist of Symbolic and Fuzzy data from the beginning of 1984 to the beginning of 1987 (84/01/05 - 87/01/02). Rules and Rule bases induced from this period are tested on the validation set which consists of data from the beginning of 1987 to the beginning of 1988 (87/01/03 - 88/01/04). The selected rules and rule bases (at the point of overfitting) are then tested on unseen data from the beginning of 1988 to the beginning of 1992 (88/01/05 - 92/01/02).

7.7.1 The Symbolic Mode

We have conducted two sets of experiments in using the genetic algorithm in the symbolic mode. We have separated all the symbolic variables into two sets (**A** and

	BP-Symbolic-(Set A)
Trading-Period-Years	4.03
Total-Closed-Trades	91.0
Total-Proft-Trades	52.0
Total-Losing-Trades	39.0
Profitable-Trades-pcn	**57**
Total-Gains-$	14278
Initial-Capital-$	100000.0
Capital-After-Trading-$	114278
Av-Gains-per-Trade-$	**156.9**
Max-Proft-Trade-$	6617.47
Max-Losing-Trade-$	-6186.19
Av-Proft-Trade-$	1872.11
Av-Losing-Trade-$	-2126.55
Maximum-Drawdown-$	-8864.71
Buy-Signals-pcn	26.37
Profit-Buy-Trades	62.5
Profit-Sell-Trades	47.76
Return-per-year-pcn	3.4
Std-Dev-of-PL	2531.67

Table 7.1: British Pound Symbolic (A) (1988–1992)

B). The set A roughly corresponds to the moving average difference variables, while the set B corresponds to the oscillator type variables (see Section 5.2). In each set the genetic algorithm is used to find rules in the solution space spanned by the variables given in that set. For a full listing of variables in set A and set B, refer to appendix E.

The rules discovered by the genetic algorithm for the set A experiments (at the point of overfitting) using British Pound data from 1984 to 1987, can be found in appendix E. These rules are then applied collectively to unseen data between 1988 and 1992. The results of the application in this period are presented in table 7.1.

The corresponding equity graph that can be drawn after applying these genetically induced rules to data between 1988 and 1992 is presented in figure 7.4.

The rules discovered by the genetic algorithm using the Set B variables (oscillator indicators) using data from 1984 to 1987 can be found in Appendix E. These rules are then applied collectively to unseen data between 1988 to 1992. The results are presented in table 7.2.

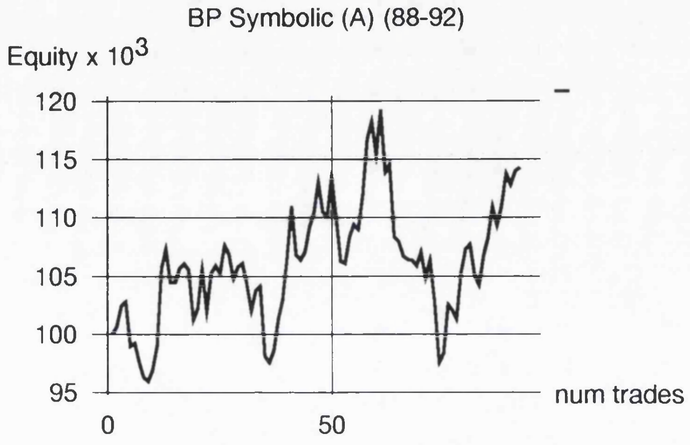


Figure 7.4: British Pound (Symbolic A) Equity Curve (1988–1992)

The corresponding equity graph that can be drawn after applying these Symbolic Set B rules to data between 1988 and 1992 is presented in figure 7.5.

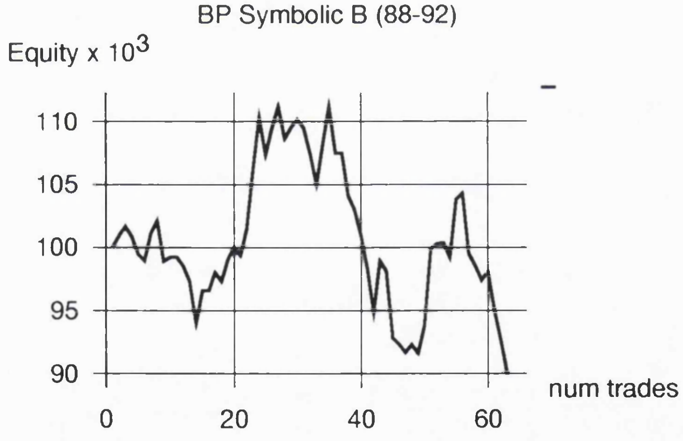


Figure 7.5: British Pound (Symbolic B) Equity Curve (1988–1992)

We have repeated the above procedures for the Deutschmark data. Again the symbolic variables are separated into two sets (A and B), and the variables are the same as used for the British Pound experiments. The induced rules at the points of overfitting can be found in Appendix E.

The results of applying the induced Set A rules to data between 1988 and 1992 are presented in table 7.3, and the corresponding equity graph is presented in 7.6.

	BP - Symbolic (B) (88-92)
Trading-Period-Years	4.03
Total-Closed-Trades	62.0
Total-Proft-Trades	30.0
Total-Losing-Trades	32.0
Profitable-Trades-pcn	**48.38**
Total-Gains-$	-10092.7
Initial-Capital-$	100000.0
Capital-After-Trading-$	89907.3
Av-Gains-per-Trade-$	**-162.79**
Max-Proft-Trade-$	6106.08
Max-Losing-Trade-$	-5297.34
Av-Proft-Trade-$	1797.48
Av-Losing-Trade-$	-2000.53
Maximum-Drawdown-$	-12712.5
Buy-Signals-pcn	0.0
Profit-Buy-Trades	0.0
Profit-Sell-Trades	48.38
Return-per-year-pcn	-2.6
Std-Dev-of-PL	2396.9

Table 7.2: British Pound Symbolic (B) (1988–1992)

Similarly, the results of applying the induced Set B rules to data between 1988 and 1992 are presented in table 7.4 and the corresponding equity graph is presented in 7.7.

The above presented results will be further discussed in detail in Section 7.8. All the rules induced by the genetic algorithm were presented to the domain expert and his assessments of the quality of the rules were recorded. These discussions will also be detailed in Section 7.8.

7.7.2 The Fuzzy Mode

As mentioned earlier in the chapter, the genetic algorithm in the fuzzy mode operates on fuzzified data using fuzzy rules and fuzzy reasoning. Each member of the genetic algorithm population is a *collection* of rules or a fuzzy rule base, as opposed to individual rules in the symbolic mode. The full list of variables that were used to induce fuzzy rule bases can be seen in Appendix E. As in the symbolic mode, we evaluate this approach to trading decision-making by simulated trading using the

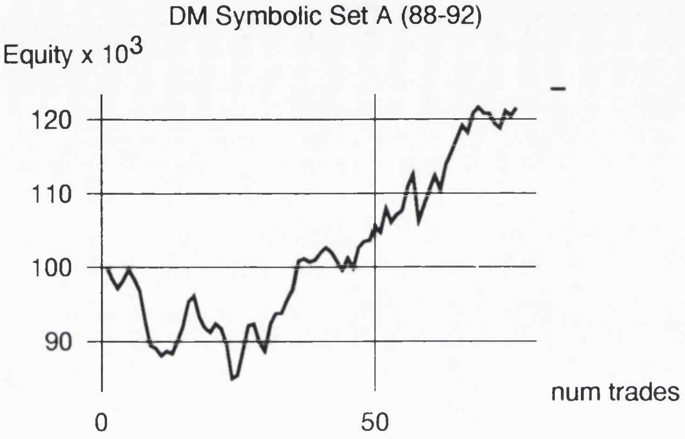


Figure 7.6: DM Genetic (symbolic) Set A

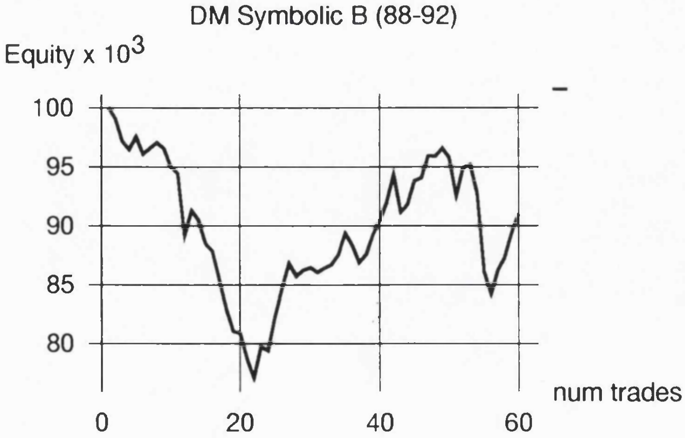


Figure 7.7: DM Genetic (Symbolic) Set B - Equity Curve (88-92)

	DM - Symbolic - Set A
Trading-Period-Years	4.03
Total-Closed-Trades	75.0
Total-Proft-Trades	42.0
Total-Losing-Trades	33.0
Profitable-Trades-pcn	**56**
Total-Gains-$	21277
Initial-Capital-$	100000.0
Capital-After-Trading-$	121277.0
Av-Gains-per-Trade-$	**283.6**
Max-Proft-Trade-$	3966.09
Max-Losing-Trade-$	-6456.9
Av-Proft-Trade-$	1617.95
Av-Losing-Trade-$	-1899.28
Maximum-Drawdown-$	-14104.5
Buy-Signals-pcn	98.66
Profit-Buy-Trades	56.75
Profit-Sell-Trades	12
Return-per-year-pcn	5.0
Std-Dev-of-PL	2187.88

Table 7.3: Deutschmark Symbolic (A) (1988–1992)

British Pound and Deutschmark data. Again the same periods of the data as used in the symbolic mode are used for training, validation and testing (1984–1987 training, 1987–1988 validation, 1988–1992 testing). The same procedure of selecting rule bases at the point of overfitting is used to select fuzzy rules for testing on un-seen data.

The full listing of the fuzzy rule bases induced by the genetic algorithm (at the point of overfitting) using British Pound and Deutschmark data from 1984 to 1987 can be found in Appendix E. The results of applying induced fuzzy rules using British Pound data from 1988 to 1992 is presented in table 7.5. The corresponding equity graph that can be drawn after applying the genetically induced fuzzy rule base is presented in figure 7.8. The same above procedure is repeated for the Deutschmark data, and trading results are presented in table 7.6 and the corresponding equity graph is displayed in figure 7.9

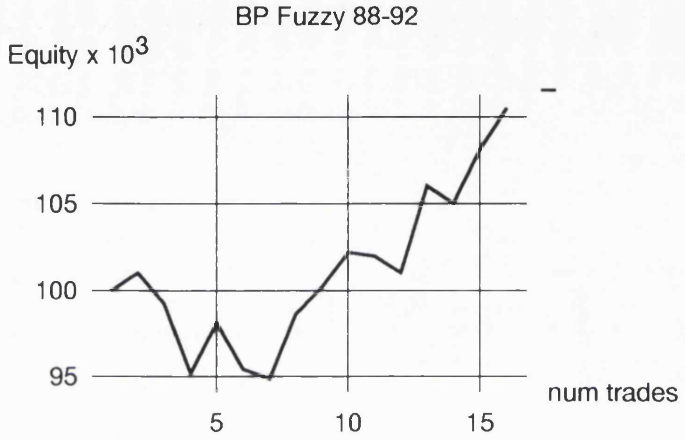


Figure 7.8: BP Genetic (Fuzzy) - Equity Curve (88-92)

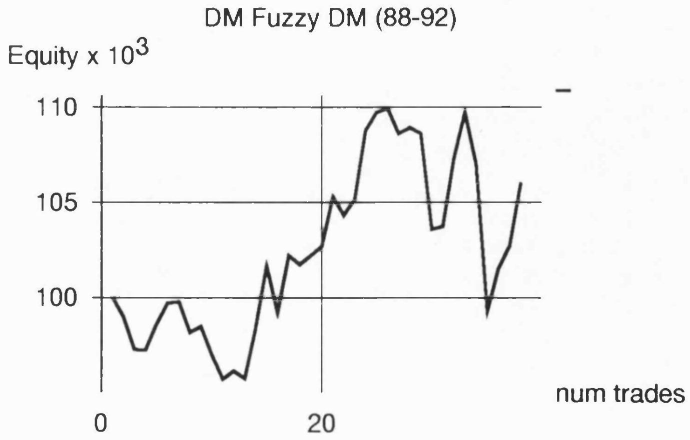


Figure 7.9: DM Genetic (Fuzzy) - Equity Curve (88-92)

	DM Symbolic (Set B)
Trading-Period-Years	4.032
Total-Closed-Trades	59.0
Total-Proft-Trades	31.0
Total-Losing-Trades	28.0
Profitable-Trades-pcn	**52.54**
Total-Gains-$	-9036.41
Initial-Capital-$	100000.0
Capital-After-Trading-$	90963.6
Av-Gains-per-Trade-$	**-153.16**
Max-Proft-Trade-$	2865.79
Max-Losing-Trade-$	-6705.38
Av-Proft-Trade-$	1318.13
Av-Losing-Trade-$	-1782.09
Maximum-Drawdown-$	-14236.2
Buy-Signals-pcn	100
Profit-Buy-Trades	52.54
Profit-Sell-Trades	0
Return-per-year-pcn	-2.32
Std-Dev-of-PL	1939.84

Table 7.4: Deutschmark Symbolic (B) (88-92)

7.8 Discussion

When assessing the results of the genetic algorithm in the Symbolic mode, a key apparent feature is the difference in performance between the two sets of experiments, A and B. On average the set A experiments which induced rules using moving averages yielded better results than the set B experiments which used oscillator indicators. The set A experiments on average achieved 56.5% correct trades while the set B experiments on average produced only 50% correct trades. The average gain per trade for the set A was $ 221 while the set B actually loses money having an average loss per trade of $ -157.97.

These results confirm the usefulness of the moving average differences approach which was also found to be superior to other approaches in the expert system experiments described in Chapter 5. As the genetic algorithm in the symbolic mode uses the same symbolic data and symbolic reasoning mechanisms as used in the expert system, it is helpful to analyse the nature of the rules it induces.

	British Pound (Fuzzy)
Trading-Period-Years	4.032
Total-Closed-Trades	15.0
Total-Proft-Trades	9.0
Total-Losing-Trades	6.0
Profitable-Trades-pcn	**60**
Total-Gains-$	10494.8
Initial-Capital-$	100000.0
Capital-After-Trading-$	110495.0
Av-Gains-per-Trade-$	**699.65**
Max-Proft-Trade-$	4998.59
Max-Losing-Trade-$	-4046.21
Av-Proft-Trade-$	2715.05
Av-Losing-Trade-$	-1603.67
Maximum-Drawdown-$	-5823.26
Buy-Signals-pcn	0.0
Profit-Buy-Trades	0.0
Profit-Sell-Trades	60
Return-per-year-pcn	2.5
Std-Dev-of-PL	2475.04

Table 7.5: British Pound - Fuzzy (88-92)

From the nine rules that were induced using Set A variables using British Pound data, 22% of the rules (G4 and G8 in Appendix E) were the same as the best performing expert system rules specified by the domain expert. Using Deutschmark data, rules induced with Set A variables had 11% rules in common with the best expert specified rules. None of the rules induced using the Set B variables in using either the British Pound or Deutschmark data was common with expert specified rules.

In comparing the overall performance of the genetically induced symbolic rules with the expert system performance, genetic symbolic rules on average produced 53.25% correct trades and had an average gain per trade of $31.50 while the expert system on average produced 55% correct trades and an average gain per trade of $285.89. Although the expert system appears to be better on average than the genetic algorithm in the symbolic mode, the average performance of the genetic algorithm is diluted by the poor performance of the set B oscillator-type variables. If one excludes the set B results then the average performance of the genetic algorithm in the symbolic mode rises to 56.5% correct trades and $221.00 gain which is comparable to the expert system results.

	Deutschmark-Fuzzy
Trading-Period-Years	4.032
Total-Closed-Trades	37.0
Total-Proft-Trades	23.0
Total-Losing-Trades	14.0
Profitable-Trades-pcn	**62.16**
Total-Gains-$	6029.16
Initial-Capital-$	100000.0
Capital-After-Trading-$	106029.0
Av-Gains-per-Trade-$	**162.95**
Max-Proft-Trade-$	3678.91
Max-Losing-Trade-$	-7731.64
Av-Proft-Trade-$	1504
Av-Losing-Trade-$	-2040.25
Maximum-Drawdown-$	-10459.2
Buy-Signals-pcn	100
Profit-Buy-Trades	62.16
Profit-Sell-Trades	0
Return-per-year-pcn	1.46
Std-Dev-of-PL	2325.97

Table 7.6: Deutschmark - Fuzzy (88-92)

All the genetically induced rules were presented to the domain expert in order to obtain a qualitative assessment of the rules. This was the same expert who specified the trading knowledge in the expert and fuzzy systems. Commenting on the induced rules using British Pound Set A data, he said that the majority of the induced rules were in his opinion variations of good trading rules. The expert said that only two rules - G2 and G3 - did not contain sufficient trading knowledge. He said that they excluded price information and relied on volatility and volume information alone. He commented that although these two rules had been predictive in the given sample, he believed that they would not yield good results over the long term. Commenting on the induced rules G1, G4, G5, G6, G7, G8, G9 which he regarded as good trading rules, he said that all of them had used long moving averages (100 day and 200 days) in combination with volume and volatility information which he believed would make them robust trading rules.

After inspecting the induced rules using Deutschmark Set A data, he said that the induced rules were generally more comprehensive than rules induced using British Pound data in that they had more pre-conditions. For example, he commented that the rule G2 had used price information, open interest information, and two types of volatility information. In his opinion this rule was better than others because it agreed with more of his theories of market movements. Similarly, rule G5 was also considered more comprehensive because it imposed conditions on price, open interest and also on volatility.

The expert found one rule in this set (G8) incoherent as it held an opposite hypothesis to another rule, G1. He said that for the rule to work more effectively the open interest condition in G8 should be changed from a *positive* term to a *negative* term. This he said makes the rule more compatible with established rules that explain the effect of open interest on price movement.

A general observation was that the expert found the interpretation of the induced rules quite easy. He commented that the rule syntax was clear and unambiguous.

The other important comparison of results refers to the performance of the genetic algorithm in the fuzzy mode. Using British Pound data the genetic algorithm in the fuzzy mode produced 60% correct trades and an average gain of $699.60. With Deutschmark data the genetic algorithm induced fuzzy rule bases that produced 62 % correct trades and an average gain of $ 162.95. These results make the average performance of the genetically induced fuzzy rule bases (61% correct trades, $431.20 gain per trade) better than the expert-specified fuzzy system which has an average performance of 59% correct trades and an average gain per trade of $364.70.

Half of the rules in the genetically-induced fuzzy rule base using British Pound contained rules that were common with fuzzy rules specified by the domain expert. With the Deutschmark data, 25% of the rules in the genetically-induced fuzzy rule base were common with fuzzy rules specified by the domain expert.

The induced fuzzy rule bases were also presented to the domain expert for evaluation. He commented that the rules induced using British Pound data were not as varied as rules induced in the symbolic mode. The majority of the rules induced used the price moving average differences and open interest information alone. He commented that only one induced rule had used volatility information and thought that more rules should have used such information. After examining the fuzzy rule bases induced using Deutschmark data he remarked that they were of a good quality utilising price, open interest and volume information. He suggested minor changes to conditions in two rules (in rules Gl and G2, the ma-diff-1-50-fuzzy values condition) by changing the *neutral* conditions to *positive* ones. This is a particular illustration of our automated approach for helping to elicit knowledge or refinement of knowledge from an expert.

The domain expert was clearly impressed with the quantitative performance results of the induced fuzzy rule bases. He said that a trading system achieving a percentage of correct trades around 60% would be considered a very good trading system (the average induced fuzzy system performance was 61%). He said that in practice he would use a variety of money management [10] strategies for incorrect trades which were not used in our simulations. The symbolic mode performance on the other hand was described as having a too 'irregular' trading profile. He believed that some of his suggested changes to the conditions might help these symbolic rules to produce more stable results.

In this chapter we have demonstrated the use of the genetic algorithm as a means for inducing both symbolic expert system rule bases as well as for inducing fuzzy rule bases. From a hybrid systems architectural perspective this can be viewed as a function replacing hybrid where the principal function of specifying domain knowledge is replaced by the use of the genetic algorithm. In this sense it is similar to Montana's work of replacing the weight changing mechanism of a neural network by a genetic algorithm [89] and Tirri's research on replacing the pattern matching mechanism of an expert system with a neural network [121]

Weiss [127] discusses the issue of merging inductive knowledge and expert specified knowledge. He says that many advances in knowledge representation and knowledge manipulation are needed to enable such combinations. In our hybrid system we have addressed this issue of 'knowledge merging' by adopting the above described common production rule structure. We believe the knowledge merging mechanisms are sufficiently general to be applied to many decision-making tasks including medical diagnosis and direct marketing. For example in the area of medical diagnosis there are already several operational expert systems and this type of genetic mechanism can help to induce relationships from past data and then augment the new knowledge with the existing knowledge. The genetic algorithm can be used periodically so that any changes in the operating environment will be detected in the form of new rules.

There are many possible extensions to the genetic algorithm experiments that we performed. Firstly, due to time limitations, there have been little experimentation with values for the crossover rate and the mutation rate. A systematic recording of the performance of the genetic algorithm with respect to changes in these two parameters may help to identify optimal values. A second extension is to use additional criteria in the evaluation of fitness of the rules. In the experiments that have been performed, the only criteria for fitness is the amount of correct trades (given a holding period of 10 days). However, further criteria such as rules having a low volatility of returns can easily be set as additional conditions that need satisfying in the dependent variables.

A clear advantage that has been demonstrated in this chapter is that the decision models that are induced are 'transparent' and comprehensible to human users. All induced models consist of production rules and contain linguistic terms that are easily understood by users. Because of this, the genetically induced models can be easily changed by a user if he or she wishes to. As described in Chapter 4, this ability to make judgmental revisions to decision models is important in many situations, but we have not fully evaluated these possibilities. In order to evaluate this facility fully, we would have to have the whole hybrid system running in a real-world operating environment for some time with an expert interacting with the system. Ideally the system would be linked to a 'live' data-feed enabling it to continuously induce new rules that reflect current trading conditions. Then if the expert wished to modify any induced models because of extraordinary changes in the operating environment (e.g. new trading conditions) or new perceptions of the nature of the trading problem, this would be easy to do, and the consequences (e.g. an expert's iterated tuning or adjustment of subsets of the rules to compensate for the modifications) could be logged and inspected. But, to obtain enough observations of this kind of expert behaviour to draw general conclusions is evidently well outside the time-scale of a single thesis project.

7.9 Limitations of the Genetic Algorithm

A main limitation of the genetic algorithm is the bias associated with the expert evaluation of the induced rules. Because we had access to only one financial trading expert, it is the same expert that specified the expert system and fuzzy system rules that was involved in the evaluation process to judge the quality of the rules induced by the genetic algorithm. Ideally we would have liked to have used several experts to evaluate the quality of the genetically induced rules. As there is considerable evidence to suggest the usefulness of 'committees' for decision making as detailed in Chapter 9, a committee composed of several experts would significantly help to increase the quality of the validation process. However, as mentioned in Chapter 4, financial traders are highly paid, very impatient professionals with very little spare time to spend on lengthy knowledge validation exercises which makes it very difficult to carry out a more extensive validation.

Another key limitation of the genetic algorithm part of the INTENT system is that it takes a very long time to induce good decision models, especially in the fuzzy mode where there are a large number of computationally intensive operations. The genetic algorithm in the fuzzy mode required about 1 1/2 days of CPU time on a SUN Sparc station to produce the given results. While POP-11 has provided a convenient way to implement the rules in INTENT, it is a very inefficient language for executing mathematical operations. An implementation of INTENT in a language such as C++ will help to overcome this problem. Further, as discussed in Chapter 2 parallelising INTENT using a environment such as GAME would also help to increase the overall performance.

Another limitation of the genetic algorithm is that the system can induce several rules which are very similar but it does not give any indication of their similarity to the user. However, for users of INTENT a mechanism to automatically 'group' similar rules would help in any evaluation process. A method for such groupings can be potentially based on the mechanism detailed in Section 7.6.1 where an assessment of the similarity of rules is made before transferring genetically induced rules to the permanent expert and fuzzy rule-bases.

Chapter 8
The Neural Networks

This chapter presents: the genetic mechanism for selecting variables for the neural network; the details of the architecture and learning algorithm of the neural network; the preparation of data for the neural network; the determination of network topologies and parameters; and finally test results for simulated trading in two currency markets.

8.1 The Genetic – Neural Strand

The performance of neural networks crucially depends on the selection of good variables [7]. Most of the successful applications of neural networks in finance and in other domains have been ones which have used domain experts or techniques such as principal component analysis to select predictive variables for model building [7], [124], [118].

In our trading application we have a very large number of variables which have been transformed using either the symbolic clustering or fuzzification process as described in Chapter 5 and Chapter 6. Therefore it is desirable to have an automated mechanism to select a subset of predictive variables from this large number of variables as inputs for the neural network. We use the genetic algorithm in the INTENT system as a mechanism for selecting predictive variables for the neural network. We make the assumption that the variables that appear in the fittest rules discovered by the genetic algorithm are variables that have predictive abilities (because our criteria for fitness assesses predictions against actual results), and are therefore suitable for the neural network.

From our hybrid classification perspective, this is an example of an *intercommunicating hybrid* where the results of one intelligent technique are being communicated to another technique for further processing. As shown in figure 8.1, variables that appear in 'fit' genetically induced rules in both modes (symbolic and fuzzy) are used as inputs to neural networks.

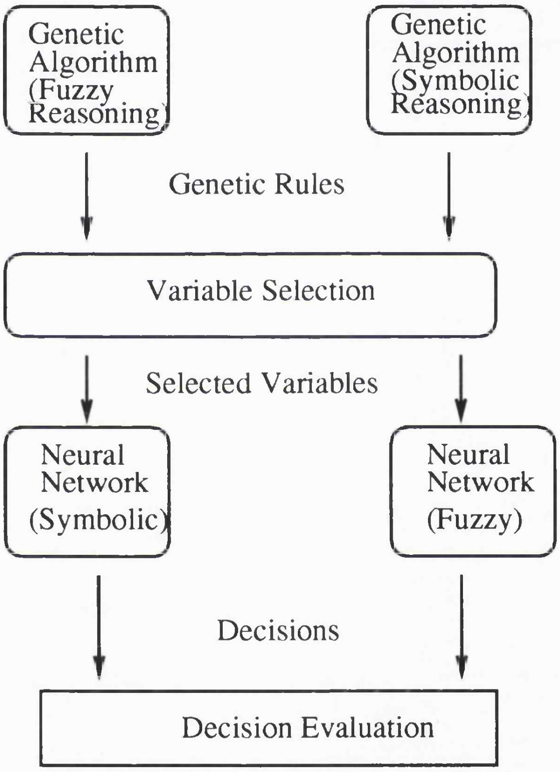


Figure 8.1: Genetic Selection of Variables

Name	pr-7	oi-rsi-7	oi-rsi-14	pr-vola-10	pr-vola-20	pr-vola-50	pr-vola-100
Freq.	4	6	8	1	10	2	7
%	10.5	15.7	21	2.6	26.3	5.2	18.4

Table 8.1: Variable Frequency Table

The procedure for variable selection is as follows:

1. Select all the variables that appear in the genetically induced fittest rules at the point of an overfit occurring using the training and validation data.
2. Rank the variables in terms of their frequency.
3. Select all variables above a user define percentage threshold, P.

Table 8.1 displays the occurrences and frequencies of the variables appearing in the following genetically induced rules.

```
   [[[oi-rsi-7 [medium]] [oi-rsi-14 [medium]] [pricevola-10 [medium]] [price-vola-20
[high]] [price-vola-100 [high]] [action [DOWN]] 18 0.65]

   [ [oi-rsi-14 [medium]] [price-vola-20 [medium]] [price-vola-100 [medium]] [action
[DOWN]] 1 0.564103]

   [[oi-rsi-7 [medium]] [oi-rsi-14 [medium]] [price-vola-20 [medium]] [price-vola-100
[high]] [action [DOWN]] 2 0.557377]

   [[oi-rsi-14 [medium]] [price-vola-20 [high]] [price-vola-50 [medium]] [price-vola-100
[high]] [action [DOWN]] 3 0.545455]

   [[oi-rsi-14 [medium]] [price-vola-20 [medium]] [action [DOWN]] 4 0.541176]

   [[price-rsi-7 [low]] [oi-rsi-7 [medium]] [oi-rsi-14 [medium]] [price-vola-20 [high]]
[action [DOWN]] 5 0.537879]

   [[price-rsi-7 [low]] [oi-rsi-7 [medium]] [oi-rsi-14 [medium]] [price-vola-20 [high]]
[price-vola-100 [high]] [action [DOWN]] 6 0.537879]

   [[price-rsi-7 [low]] [oi-rsi-7 [medium]] [price-vola-20 [high]] [action [DOWN]] 7
0.537879]

   [[price-rsi-7 [low]] [oi-rsi-7 [medium]] [price-vola-20 [low]] [price-vola-100 [high]]
[action [DOWN]] 8 0.537879]]
```

Let the user defined cut-off frequency, P, be 10%, where all variables which have a frequency of 10% or greater will be selected. In this case the following variables are selected as inputs for the neural network:

```
pr-7 oi-rsi-7 oi-rsi-14 pr-vola-20 pr-vola-100
```

These variables are symbolic variables (e.g. values such as medium, positive, low) which were selected from the genetic algorithm operating in the symbolic mode. The same selection mechanism applies for selecting fuzzy variables (e.g. values such as low 0.9 medium 0.1 high 0.0) from the genetic algorithm operating in the fuzzy mode.

8.2 The Neural Network Design

A supervised learning procedure, the back-propagation algorithm [126], is used to learn the mapping between the states of input variables and trading decisions. The main reason for adopting this particular algorithm is its proven ability to successfully model financial systems [99].

All our neural network experiments have been conducted within a publicly available neural network environment called NevProp developed by the University of Nevada [51]. This environment provides a convenient method to change different network parameters and to log network performance measures such as classification errors.

After inputs have been selected for a neural network, there are still many design choices that one has to make. These are the topology of the network, the weight ranges, learning rates, transfer functions and momentum terms. In section 8.2.3 we detail these choices.

The next section describes the detailed functioning of the back-propagation neural network. This section follows a straight-forward 'textbook' application of the back-propagation algorithm without any claim of originality. It was important to conduct the neural network experiments for two principal reasons. Firstly, as the majority of intelligent systems applications in trading use neural networks it provides a method for comparing our other expert, fuzzy and genetic approaches. Secondly, it is necessary to run the neural network experiments as its inputs are used in the multiple-model strand described in the next chapter.

8.2.1 The Back-propagation Network

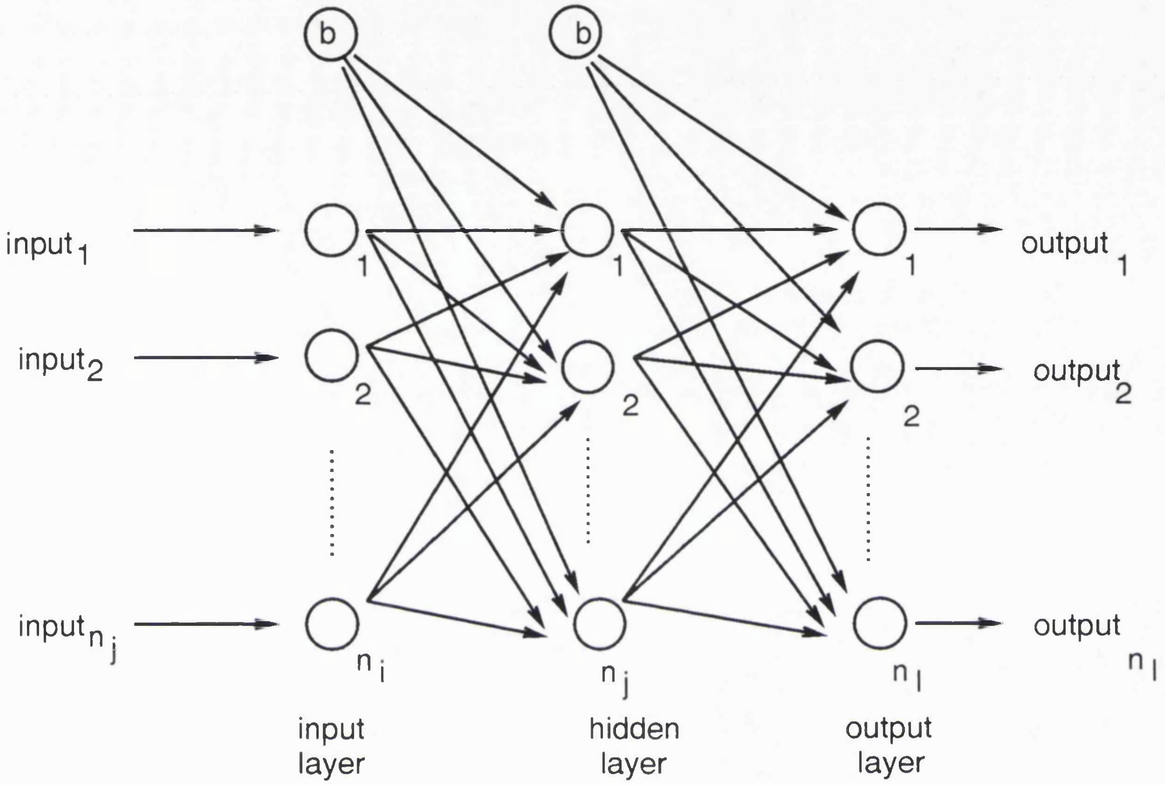


Figure 8.2: Back Propagation Network Structure

Referring to figure 8.2, i denotes an input, o an output, w a connection weight, and n the number of nodes in a layer. The subscripts i, j and l refer to the input, hidden and output layers respectively.

The bias units are indicated by b in figure 8.2. These units always have an output of 1. They serve as threshold units for the layers to which they are connected, and the weights from the bias units to each of the units in the following layer is adjusted exactly like other processing units.

The net input to a processing unit is,

$$net.input_j = i_j = \sum_i w_{ji} o_i$$

(8.1)

This weighted sum, i_j is then sent through an *activation function*, also known as a 'squashing function'. We use the sigmoid function in all our experiments.

The output of a processing unit, o_j, using a sigmoid function is,

$$o_j = \frac{1}{1 + exp(-i_j)}$$

(8.2)

Error Back-Propagation

The neural network is initialised with random weights, typically in the range [-0.5, +0.5]. Input vectors are then presented to the network and their output is calculated using the above equations. The aim of the back-propagation algorithm is to find a set of weights that minimises the error between the input and output vectors. This is achieved by iteratively adjusting the weights in the network in the direction that minimises this global error measure.

A commonly used measure of error, E_p is,

$$E_p = 0.5 \sum_{l=1}^{n_l} (t_{pl} - o_{pl})^2$$

(8.3)

where t_l is the target value and o_l is the value actually produced by the network as a result of the feed-forward calculations. The error term is defined for a given pattern and summed over all output units for that pattern. The subscript p denotes what the value is for a given pattern. We implement the error calculation over the entire set of (epoch) patterns, rather than on a pattern-by-pattern basis.

The error minimisation is achieved by repeatedly changing the weights by an amount proportional to the derivative $\frac{dE}{dW}$ denoted by δ_l.

$$\delta_l = f'(i_l)(t_l - o_l)$$

(8.4)

where $f'(i_l)$ is the first derivative of the output of a unit in the output layer which is a function of its input, or $o_l = f(i_l)$.

For the sigmoid activation function of equation 8.2, the first derivative is just $o_l(1 - o_l)$. In the case of the sigmoid function, therefore the output of a unit in the output layer is,

$$\delta_l = (t_l - o_l)o_l(1 - o_l)$$

(8.5)

There are two ways to propagate this error value back and perform appropriate weight adjustments. One way involves propagating the error back and adjusting weights after each training pattern is presented to the network, called *on-line* or *single pattern*, training. The other way is to accumulate the $\delta's$ for each unit for the entire training set, add them, and propagate back the error based on the grand total δ called *batch* or *epoch* training. We use epoch training in all our simulations.

The weight update procedure for weights that feed the output layer w_{lj} is,

$$w_{lj}(new) = w_{lj}(old) + \eta\delta_l o_j$$

(8.6)

Here, η is defined as the learning coefficient. It can be assigned values between 0 and 1.

This type of weight updating sometimes gets caught in local *energy minima* [12]. If one visualises a bowl-shaped surface with many little bumps and ridges, then the error minimisation procedure is analogous to minimising the energy of our position in this bumpy ridge lined bowl. Ideally one would like to move our position to the bottom of the bowl where the error is at a minimum - a globally optimal solution. Depending on how much or how little we can move at one time, we might get caught in a little depression or ridge that one cannot get out of when simple optimisation methods are used. This situation is most likely with small limits on each individual movements which correspond to small values of η. A *momentum term*, is usually added to the weight update procedure in the hope that it will help escape such local minima. The momentum term is the product of the momentum factor, α, and the previous weight change.

The weight update procedure for the output units with momentum is,

$$w_{lj}(new) = w_{lj}(old) + \eta\delta_l o_j + \alpha[\Delta W_{lj}(old)]$$

(8.7)

In order to calculate the weight update for the hidden units, we calculate the error term δ as described by Rumelhart and McClelland [105].

$$\delta_h = f'(i_h) \sum_{l=0}^{n_l} w_{lh} \delta_l$$

(8.8)

As was the case in the output layer, the output of a unit in the hidden layer is a function of its input, or $o_h = f(i_h)$. For the sigmoid transfer function, this derivative is $o_h(1 - o_h)$ resulting in the hidden node error term define by Eq. 8.9.

$$\delta_h = o_h(1 - o_h) \sum_{l=0}^{n_l} w_{lh} \delta_l$$

(8.9).

The weight changes for the connections feeding the hidden layer from the input layer are now calculated in a manner analogous to those feeding the output layer:

$$w_{ji}(new) = w_{ji}(old) + \eta \delta_j o_i + \alpha[\Delta W_{lj}(old)]$$

(8.9)

For each hidden node, the subscript i takes on values of 0 to n_i, the number of input units. As before, the bias units are represented in the calculations by the n_ith value.

8.2.2 Data Preparation and Network Inputs and Outputs

Variables which are selected from the genetic algorithm have to be converted into a suitable format for the neural network. If one uses a sigmoid transfer function then all inputs have to be scaled into continuos real values in the range [0,1]. The table 8.2 presents an example set of symbolic values that have been converted to a format suitable for the neural network. Such look-up tables containing the binary representations for the linguistic categories (i.e. low is 001, medium is 010) are constructed for each different linguistic representation (e.g. increasing is 1000, decreasing is 0001).

For the fuzzy variables the conversion is more straightforward. Here as the linguistic categories are already scaled into a range [0,1] there is no need to define a separate

Before conversion	After conversion
low	0 0 1
medium	0 1 0
high	1 0 0

Table 8.2: Symbolic Variable Conversion

Before conversion	After conversion
[[low 0] [medium 0.288463] [high 0.711536]]	0 0.288463 0.711536
[[negative 0.182451] [neutral 0.817549] [positive 0]]	0.182451 0.817549 0

Table 8.3: Fuzzy Variable Conversion

look-up table. The data preparation, as presented in table 8.3 simply involves the removing of the symbolic labels from the fuzzified data values.

Once the data is converted to a format suitable for the neural network, calculating the number of input nodes is straightforward. The number of input units is the total number of distinct data items in the range [0,1] that correspond to the selected (by the genetic algorithm) symbolic or fuzzy variables. For example if there are 2 selected variables from symbolic data having ranges of low, medium and high as defined in table 8.2, then there will be six input units in the neural network.

The determination of the number of output nodes is also straight-forward. In this trading application we use one node (values closer to 0 indicating a Sell, values closer to 1 indicating a Buy).

8.2.3 Network Complexity – Determining Hidden Units

The complexity fit for the neural network – the number of hidden units – is determined by the same procedure used for the genetic algorithm. This procedure as described by Weiss [127] involves monitoring the errors on the training set and the validation set as the number of hidden units is increased. The network at the error turning point is chosen and then applied to unseen data. The resulting trading decisions are evaluated by the decision evaluation module.

Data from the beginning of January 1984 to the beginning of January 1987 were used to train the neural networks (the same data set that was used to train the genetic algorithms). Data from January 1987 to January 1989 were used for validation.

Hidden Units	2	4	8	10	12
Iterations	2000	4000	3000	4000	2000
Error (train/valid)	43/77	33/65	29/53	28/44	28/51

Table 8.4: Varying Number of Hidden Units

Experiments with 2, 4, 8, 10 and 12 hidden units were conducted and the topology at the error turning point was selected.

For each network, the classification errors at every 500 epochs were recorded. If the errors for the most recent 500 epochs were not better than the previous 500, then we concluded that no progress was being made and halted training at that point.

Table 8.4 illustrates the effect of varying the number of hidden units using British Pound Symbolic Data. In the first cell of iterations row the number 2000 indicates the number of epochs trained before being stopped by the above error progress criterion. In the first cell of the error row the number classification error 45 indicates the error rate on the training data and 77 is the error rate on the validation data.

In the above example the error on the validation set increases when more than 10 hidden units are used. It is concluded that at this point the network has the right complexity to solve the problem at hand and all further experiments using that particular data-set are done keeping this number of hidden units.

This network is then applied to completely unseen data. The selection of hidden units is done individually for all the four data sets (British Pound using symbolic data, British Pound using fuzzy data, Deutschmark using symbolic data, Deutschmark using fuzzy data).

The following other network parameters were used for all experiments : Initial weights range: $[-0.5, +0.5]$, Learning rate : 0.5 and Momentum : 0.9. We have not experimented with other values for these parameters and these particular values were chosen because other researchers have empirically found these configurations to produce good network results [127] [99].

8.3 Simulated Trading

The table 8.5 presents the selected number of hidden units for the different data sets using data from January 1984 to January 1987. Data from January 1987

Currency	Data Type	Hddn. Units
British Pound	Symbolic	10
Deutschmark	Symbolic	12
British Pound	Fuzzy	8
Deutschmark	Fuzzy	10

Table 8.5: Selected Network Parameters

to January 1988 were used as the validation set. The trained networks were then presented with unseen data from January 1988 to January 1992.

The outputs of the network are interpreted in the following manner: a value $>= 0.7$ is considered a Buy decision while outputs $<= 0.3$ are considered as a Sell decision. These values have been arbitrarily chosen and because of time limitations we have not had the opportunity to conduct experiments using different values.

As in the previous modules the final decisions are sent to the **trades list** which contains information about the decision (buy or sell) and an index indicating the date of the trade being made. This is then subsequently sent to the decision evaluation module.

The simulated trading statistics for the neural network using symbolic data for trading the British Pound between 1988 to 1992 are displayed in table 8.6 and for the Deutschmark in table 8.7. The corresponding equity graph for the British Pound is presented in figure 8.3 and the Deutschmark graph in 8.4.

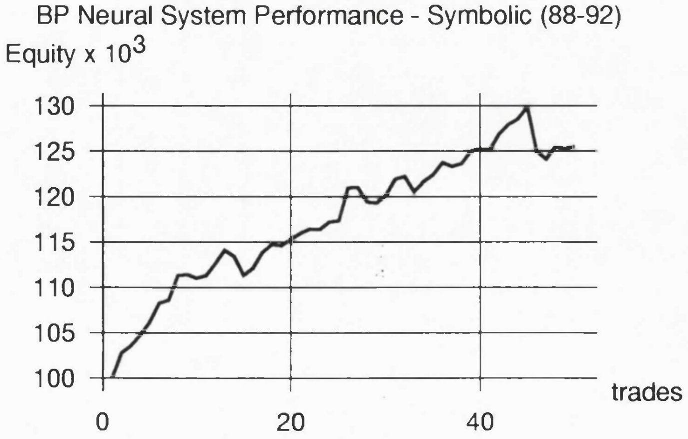


Figure 8.3: Neural Network (Symbolic) - British Pound Equity Curve (88-92)

	British Pound - Symbolic
Trading-Period-Years	4.03
Total-Closed-Trades	49.0
Total-Proft-Trades	32.0
Total-Losing-Trades	17.0
Profitable-Trades-pcn	**65**
Total-Gains-$	25529
Initial-Capital-$	100000.0
Capital-After-Trading-$	125529
Av-Gains-per-Trade-$	**521.0**
Max-Proft-Trade-$	8314.21
Max-Losing-Trade-$	-3105.38
Av-Proft-Trade-$	3148.43
Av-Losing-Trade-$	-1131.23
Maximum-Drawdown-$	-3105.38
Buy-Signals-pcn	10.20
Profit-Buy-Trades	54
Profit-Sell-Trades	59
Return-per-year-pcn	5.80
Std-Dev-of-PL	2696.14

Table 8.6: British Pound - Neural Network Symbolic Data (1988–1992)

The simulated trading statistics for the neural network using fuzzy data for trading the British Pound between 1988 to 1992 is displayed in table 8.8 and the Deutschmark trading figures is presented in 8.9. The equity graph for the British Pound trading is displayed in figure 8.5 and the Deutschmark equity graph is presented in 8.6.

8.4 Discussion

The neural networks detailed in this chapter have produced results superior to all the other modules in the INTENT system. The neural networks that used symbolic variables (selected from the genetic algorithm) on average produced 62.5% correct trading decisions and average gains per trade of $461.5. The neural networks which used fuzzy data performed better than the networks that used symbolic data. The neural networks that used fuzzy data on average produced 65% correct trading decisions with an average gains per trade of $580.

When comparing these results with other modules in the INTENT system, the nearest performing module was the genetic algorithm in the fuzzy mode where the

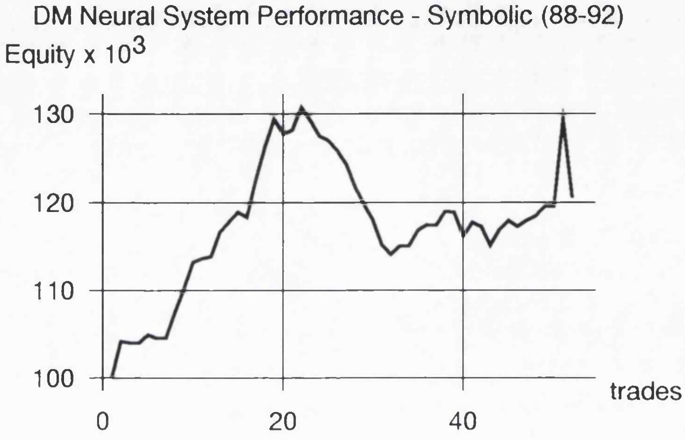


Figure 8.4: Neural Network (Symbolic) - Deutschmark Equity Curve (88-92)

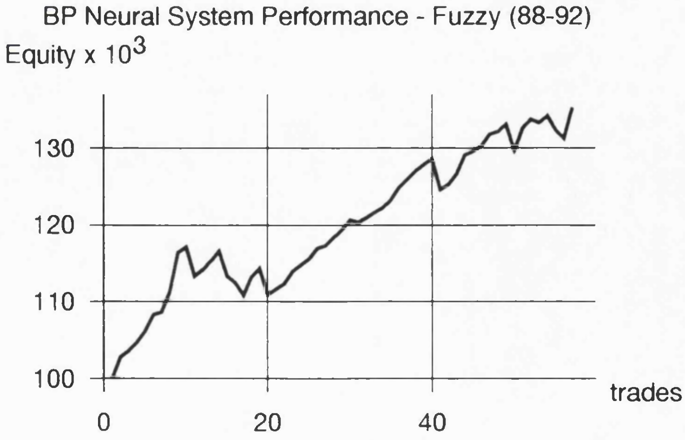


Figure 8.5: Neural Network (Fuzzy) - British Pound Equity Curve (88-92)

	Deutschmark - Symbolic
Trading-Period-Years	4.032
Total-Closed-Trades	51.0
Total-Proft-Trades	31.0
Total-Losing-Trades	20.0
Profitable-Trades-pcn	**60**
Total-Gains-$	20502
Initial-Capital-$	100000.0
Capital-After-Trading-$	120502
Av-Gains-per-Trade-$	**402.0**
Max-Proft-Trade-$	8268.35
Max-Losing-Trade-$	-2305.55
Av-Proft-Trade-$	2316.44
Av-Losing-Trade-$	-958.87
Maximum-Drawdown-$	-2305.55
Buy-Signals-pcn	13.72
Profit-Buy-Trades	60.2
Profit-Sell-Trades	52.00
Return-per-year-pcn	4.7
Std-Dev-of-PL	2255.77

Table 8.7: Deutschmark - Neural Network Symbolic Data (1988–1992)

average gains per trade was $431.30 and the average correct trading decisions was 61%.

Colin [28] reports a 55% correct decision rate using a backpropagation neural network for trading the British Pound and the Deutschmark. The test period that was used, between 1988 and 1991, is included in the test period that we have used. However as Colin has conducted his research within a commercial organisation he does not give details of the variables he has used which are considered as proprietary variables. Because of this situation these results are not reproducible and are of little scientific value. Further, our results are also quantitatively better than the results that Colin reports.

As in the other modules the results and procedures of the neural network simulations were presented to the domain expert. Firstly our domain expert was invited to comment on the quality of the variables that were selected by the genetic algorithm for the neural networks. Commenting on the variables selected using British Pound (Symbolic data), he said that from the six selected variables (ma-diff-200, vol-ma-diff-10, vola-10, vola-20, vola-100, oi-rsi-7) all except one variable (oi-rsi-7)

	British Pound - Fuzzy
Trading-Period-Years	4.03
Total-Closed-Trades	56.0
Total-Proft-Trades	38.0
Total-Losing-Trades	18.0
Profitable-Trades-pcn	**67**
Total-Gains-$	35280
Initial-Capital-$	100000.0
Capital-After-Trading-$	135280.0
Av-Gains-per-Trade-$	**630.0**
Max-Proft-Trade-$	9457.59
Max-Losing-Trade-$	-2274.68
Av-Proft-Trade-$	3288.38
Av-Losing-Trade-$	-852.43
Maximum-Drawdown-$	-2513.73
Buy-Signals-pcn	30.3
Profit-Buy-Trades	58.23
Profit-Sell-Trades	76.9
Return-per-year-pcn	7.9
Std-Dev-of-PL	2651.48

Table 8.8: British Pound - Neural Network Fuzzy Data (1988-1992)

were good variables for model building. He said that the oi-rsi-7 variable which calculates a relative strength index of the open interest variable was less effective than computing a moving average of that particular variable. He was of the view that the neural network might produce very similar results even if this particular variable was dropped. For the variables selected using Deutschmark symbolic data he had a similar comment where he approved four (ma-diff-200, ma-diff-50, oi-ma-diff-10, vola-100) out of six variables and said that two variables (vol-rsi-7, vol-rsi-28) were probably redundant because of their method of calculation. However due to our time constraints we have not conducted experiments to evaluate these claims.

In the fuzzy data sets the selected variables were fewer — three in the case of the British Pound (ma-diff-fuzzy-200, oi-ma-diff-fuzzy-20, vola-fuzzy-100) and four in the case of the Deutschmark (ma-diff-fuzzy-50, ma-diff-fuzzy-200, vol-ma-diff-fuzzy-20, vola-fuzzy-20). The expert commented that all of these selected variables were ones that he considered as having significant predictive power.

The quantitative results of the neural network performance were also presented to the domain expert. Although he remarked that the results were clearly good, he

	Deutschmark - Fuzzy
Trading-Period-Years	4.03
Total-Closed-Trades	57.0
Total-Proft-Trades	36.0
Total-Losing-Trades	21.0
Profitable-Trades-pcn	**63.0**
Total-Gains-$	30210
Initial-Capital-$	100000.0
Capital-After-Trading-$	130210
Av-Gains-per-Trade-$	**530.0**
Max-Proft-Trade-$	7830.12
Max-Losing-Trade-$	-9508.33
Av-Proft-Trade-$	3090.87
Av-Losing-Trade-$	-1961.91
Maximum-Drawdown-$	-1961.91
Buy-Signals-pcn	43.85
Profit-Buy-Trades	51
Profit-Sell-Trades	55.0
Return-per-year-pcn	6.8
Std-Dev-of-PL	2912.1

Table 8.9: Deutschmark - Neural Network Fuzzy Data (1988–1992)

wanted to know more about the exact decision process that was employed. He said he wanted to know whether the network had derived an understanding of the combinations of the variables in a manner similar to the way he uses them. When explained that this was difficult because of the 'black-box' nature of the neural networks, he said that it is vital to get 'some handle on what it does' because under certain trading conditions particular indicators can lose their importance. If the reasoning process in the model is known then in such circumstances the expert can upgrade or downgrade the model predictions using the behavioural knowledge of the trading indicators.

One possible solution to get 'some handle' on the neural network results would be the use of techniques such as sensitivity analysis [25] on the neural networks to determine the inputs that contribute the most to the final decisions. Although one will still not have a complete understanding of the decision procedures, an understanding of the significance of the variables may be useful to the expert to hypothesise relationships among them.

A further extension to the neural network presented here would be to use a genetic algorithm to search for the optimal parameters for learning rates, momentum,

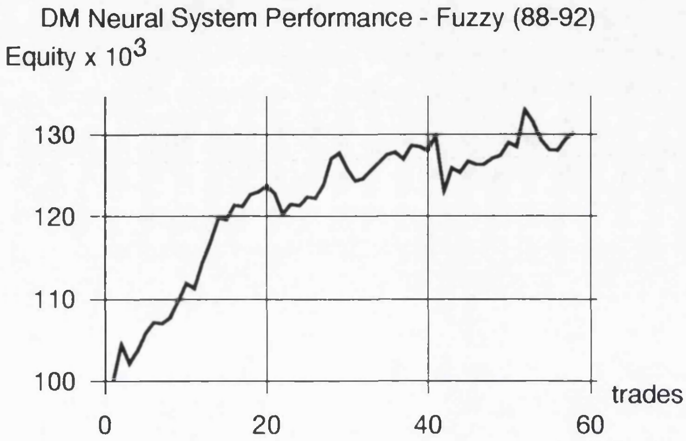


Figure 8.6: Neural Network (Fuzzy) - Deutschmark Equity Curve (88-92)

and the cut-off values for interpreting the network outputs. One can adopt a similar representation scheme used by Montana [89] where the network parameters are represented in a list structure and are subsequently manipulated by a genetic algorithm. The genetic algorithm presented in the previous chapter can be easily used for such a task.

Considering the hybrid architecture of the genetic-neural strand, it is similar to other intercommunicating hybrids such as Schreinmaker's system for veterinary diagnosis [109] where an expert system and neural network solve different, separate aspects of the problem. The tasks performed in our hybrid are serial and are not similar to the competing agent or blackboard style of problem solving adopted in other intercommunicating hybrids such as Dunker's system for integrating expert systems and neural networks [35].

Although this hybrid strand clearly outperforms other approaches in INTENT in terms of quantitative performance, as evident from the reservations of our domain expert, it has limitations in its ability to explain the decision models it produces. Because of this limitation it is doubtful whether such an approach will be general enough to be applied to other domains such as medical decision making. In the area of medical decision making the need for clear explanations is even more paramount than in the domain of financial trading, as lives may be at risk if an incorrect decision is made. Although there could be incorrect decisions even with the expert-fuzzy-genetic strand, the decisions it induces can be verified by (say) a consultant, thus potentially minimising such risks.

Chapter 9
Decision Combination and Evaluation

This chapter presents: combination of decisions in INTENT; an evaluation of the decision making capability of the expert trader; summary of performance of all modules in INTENT; an evaluation of INTENT performance system with respect to other published studies and finally a sketch of a methodology for hybrid systems.

9.1 The Multiple Model Strand

There has been evidence suggesting that combining forecasts of several models can yield better results than using a single model alone [22]. Combining predictions of different models is usually performed by taking a simple average of the predictions or by the use of simple weighting schemes [3].

The advantages of such consensus forecasting is particularly apparent when there is a negative correlation between the models [22]. In two negatively correlated models, Bunn [21] obtained 80% reduction in mean squared error through combining. Armstrong [8] has conducted eight studies and has obtained 6% reduction in mean absolute percentage error through the combination of different models.

In economics, the use of consensus forecasting is now routine. The primary form of reporting economic forecasts, in the *Economist*, for example, is the calculation of the average of forecasts of twenty independent economic forecasting units [3].

McNees [3] has studied the performance of 22 economic forecasters since 1977. He concludes that for a single variable, such as inflation, about a third of the forecasters are consistently more accurate than the consensus. However, each forecaster has strengths and weaknesses. He or she may be very good at forecasting GNP, but may grossly misjudge inflation. Although almost every forecaster can claim to be more accurate than the consensus for at least one variable, none of the forecasters studied were more accurate on all seven indicators covered. Taking all the indicators together, the consensus forecasts had a smaller average error over time.

Another piece of evidence for consensus forecasting is the performance of the winners of an annual contest for economic forecasting conducted by the Sterling

National Bank of New York [3]. In the ten years to 1990 the winners were not, on average, any more accurate than the consensus.

The consensus forecasting approach has very recently started to enter the domain of intelligent systems for forecasting. Zhang et al. [130] have reported exceptional results for protein secondary structure prediction using a *committee* of intelligent modules. They have used a neural network, a statistical model, and a memory-based reasoning module in the committee. All three models make predictions of the protein structures independently, and these predictions are used by another neural network (called the Combiner) which learns the mapping between the model predictions and the target protein structures thus further minimising the prediction error. Zhang et al. claim that this hybrid method outperforms all other reported methods of protein structure prediction.

Pearlmutter [95] has argued that the use of a committee approach for neural modelling has particular advantages as it can potentially deal with noise in the data, sampling error, approximation error and randomness in the classifier itself. The argument concerning randomness is particularly relevant to neural networks and genetic algorithms. For instance, if backpropagation is run on the same data twice, with the same architecture and all the other parameters held the same, it will still typically produce different answers, due to differences in the random initial weights. Pearlmutter argues that averaging this out through the use of multiple models can help to increase the accuracy of results.

The third hybrid strand in the INTENT system, *the multiple model strand*, consists of combining decisions of all the different modules in the system. The decisions made by six different methods — the expert system, the fuzzy system, genetic algorithm (symbolic mode), genetic algorithm (fuzzy mode), neural network (symbolic data), and neural network (fuzzy data) — are aggregated by the decision combination module (see figure 9.1).

As mentioned in the previous chapters, the output of each of the different modules is either a BUY, or SELL decision. The decision combination is performed in the following manner. The user specifies a *consensus threshold*, C, and only if there are greater than C identical decisions being made by the different modules is that decision executed. For example if C is set to 4, then only if at least four of the modules make the same decision is that decision implemented.

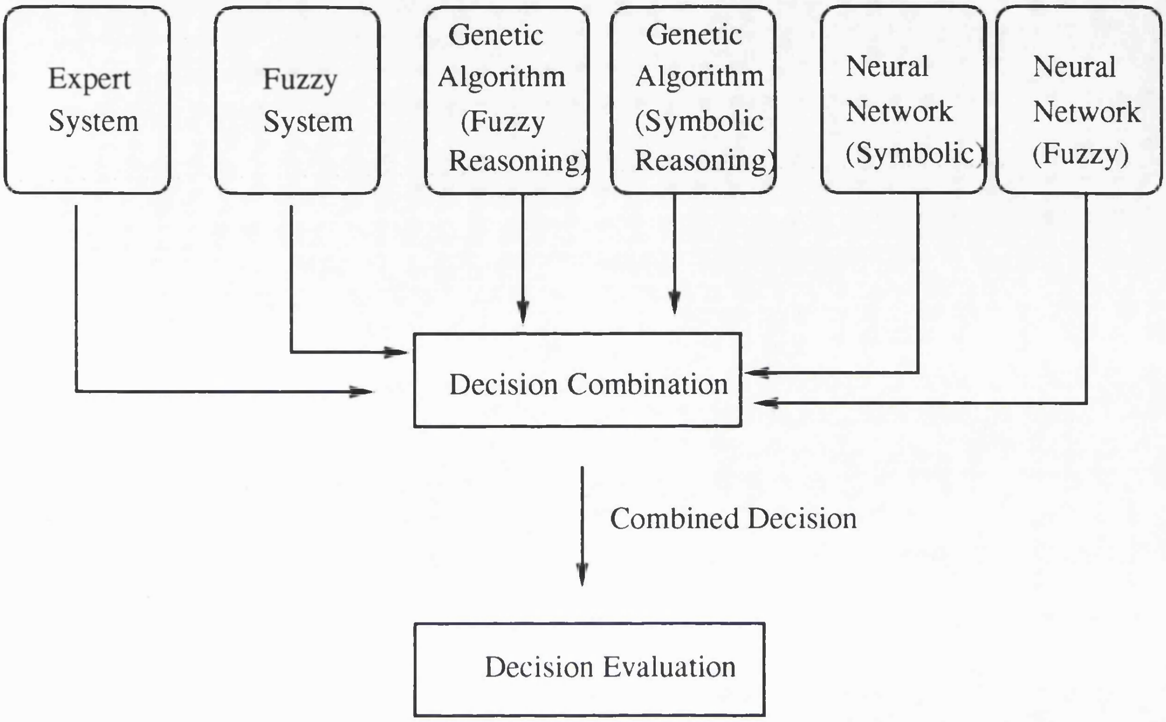


Figure 9.1: Decision Combination using Multiple Models

In our trading simulations we have conducted experiments using $C = 3$, and $C = 4$. The simulations were performed using the same data series as the other simulations (British Pound and Deutschmark) during the same trading periods (1988–1992)

The performance of consensus decision making for the British Pound, $C = 3$ and $C = 4$ are detailed in tables 9.1 and 9.2 respectively. The corresponding equity graphs are shown in figures 9.2 and 9.3.

These simulations are repeated with Deutschmark data. The results of $C = 3$ and $C = 4$ for the Deutschmark are presented in tables 9.3 and 9.4 and the equity graphs are presented in figures 9.4 and 9.5.

The decision combination approach has produced the best results in the INTENT system. The three consenting modules ($C = 3$) on average has produced 73.87 % correct decisions with an average gain per trade of \$708. The four consenting modules ($C = 4$) which imposes a more stringent threshold produces even better results of 74% correct trades and an average gain per trade of \$812. We shall discuss these results further with respect to other approaches in section 9.3.

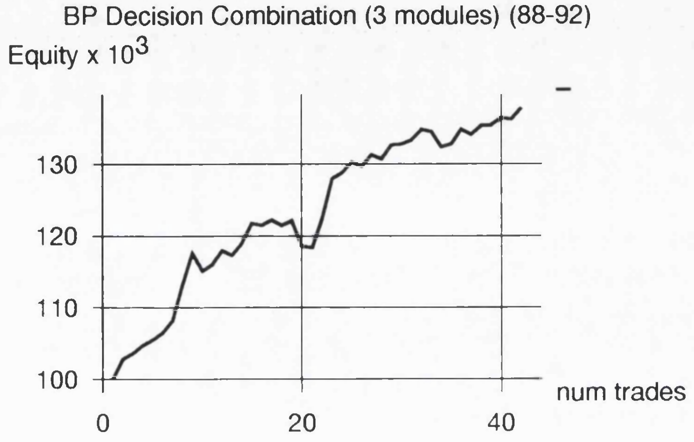


Figure 9.2: British Pound - Decision Combination ($C = 3$) - Equity Curve (1988–1992)

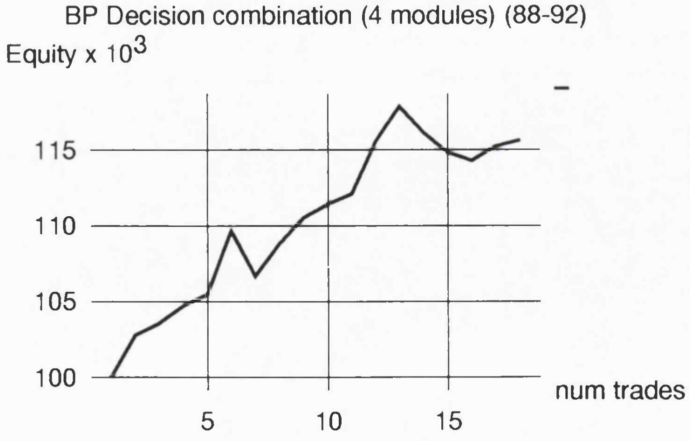


Figure 9.3: British Pound Decision Combination ($C = 4$) (1988–1992)

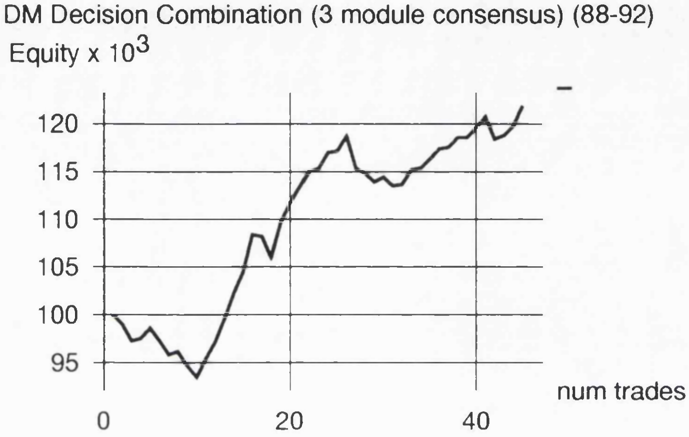


Figure 9.4: Deutschmark Decision Combination ($C = 3$) – Equity Curve (1988–1992)

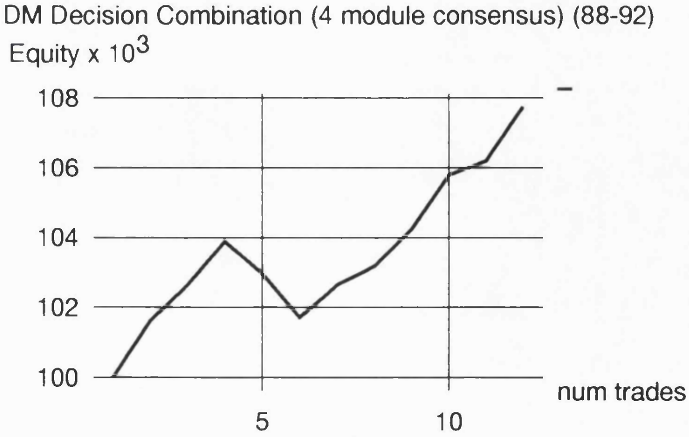


Figure 9.5: Deutschmark Decision Combination ($C = 4$) (1988–1992)

	British Pound ($C = 3$)
Trading-Period-Years	4.03
Total-Closed-Trades	41.0
Total-Proft-Trades	28.0
Total-Losing-Trades	13.0
Profitable-Trades-pcn	**68.29**
Total-Gains-$	37720
Initial-Capital-$	100000.0
Capital-After-Trading-$	137720
Av-Gains-per-Trade-$	**920.0**
Max-Proft-Trade-$	7119.68
Max-Losing-Trade-$	-3531.4
Av-Proft-Trade-$	2473.17
Av-Losing-Trade-$	-1528.43
Maximum-Drawdown-$	-6289.04
Buy-Signals-pcn	24.3
Profit-Buy-Trades	59
Profit-Sell-Trades	67.5
Return-per-year-pcn	8.3
Std-Dev-of-PL	2465.01

Table 9.1: Decision Combination ($C = 3$) - British Pound (1988–1992)

A general observation is that while the quality of decisions increases as the consensus threshold is increased, the frequency of trading decisions also falls. For example in the case of the Deutschmark data, a consensus threshold of four produces only a quarter as many decisions as are produced by using a threshold of three. This aspect of the decision combination method can potentially pose problems because if the threshold is set too high then there might be a situation where there would be no decisions produced at all. In some domains such as medical diagnosis it may be better to have very high consensus thresholds which leads to very few, but high quality, decisions. On the other hand in the trading domain there is a trade-off between the frequency of decisions and the quality of decisions. For example, in a given year it will be more profitable to have 20 trades with an average gain of $200 than to have 10 higher quality trades with an average gain of $350. Because of these reasons the consensus threshold will have be set by a user depending on his or her domain requirements. It is also advisable for the user to experiment with many different thresholds using past data.

A possible extension to the scheme is the allowance of *preference votes* defined by a user or domain expert. That is, the user can decide to give a particular module in

	British Pound ($C = 4$)
Trading-Period-Years	4.03
Total-Closed-Trades	17.0
Total-Proft-Trades	13.0
Total-Losing-Trades	4.0
Profitable-Trades-pcn	**76**
Total-Gains-$	15640
Initial-Capital-$	100000.0
Capital-After-Trading-$	115640
Av-Gains-per-Trade-$	**920.0**
Max-Proft-Trade-$	5652.78
Max-Losing-Trade-$	-2998.65
Av-Proft-Trade-$	2102.69
Av-Losing-Trade-$	-2155.36
Maximum-Drawdown-$	-2998.65
Buy-Signals-pcn	0.0
Profit-Buy-Trades	0.0
Profit-Sell-Trades	76
Return-per-year-pcn	3.7
Std-Dev-of-PL	1976.7

Table 9.2: British Pound ($C = 4$) (1988–1992)

the INTENT system, say the fuzzy system, more weighting in the decision combination process by giving it more than one vote in the votes required by the consensus threshold. For example, if the fuzzy module is given 2 preference votes, then in the case of a three consensus threshold ($C = 3$) only one other module has to produce the same decision as the fuzzy system (one vote) for the threshold condition to be satisfied. The rationale for such a facility is that it allows users to give more weight in the decision combination process to modules that they subjectively feel are better than others. Users may also make such judgments by objectively evaluating the performance of the different individual modules attempting to solve the same problem. For example in this research we have previously evaluated different individual modules (expert system, fuzzy system etc.) on the same data and on the basis of (say) the characteristics of the equity curve an expert may decide to give the fuzzy system twice as much preference as decisions produced by the expert system.

From our hybrid classification viewpoint this multiple model hybrid is an intercommunicating hybrid where the results of different modules are communicated to a central module. However in this case we are stressing the communication aspects of the intercommunicating class and not the aspects that define solving different parts of

	Deutschmark ($C = 3$)
Trading-Period-Years	4.03
Total-Closed-Trades	44.0
Total-Proft-Trades	35.0
Total-Losing-Trades	9.0
Profitable-Trades-pcn	**79.54**
Total-Gains-$	21824.0
Initial-Capital-$	100000.0
Capital-After-Trading-$	121824.0
Av-Gains-per-Trade-$	**496.00**
Max-Proft-Trade-$	10045.1
Max-Losing-Trade-$	-2220.38
Av-Proft-Trade-$	2528.74
Av-Losing-Trade-$	-1271.31
Maximum-Drawdown-$	-2781.66
Buy-Signals-pcn	70.45
Profit-Buy-Trades	74.0
Profit-Sell-Trades	82.30
Return-per-year-pcn	5.0
Std-Dev-of-PL	2348.68

Table 9.3: Deutschmark – Decision Combination ($C = 3$) (1988–1992)

a given problem. The central collation of decisions in this strand has similarities with Dunker's blackboard architecture based hybrid system [35] (discussed in Chapter 3) for combining neural networks and expert systems.

9.2 Evaluating Expert Trader Performance

As discussed in the previous chapters, there are difficulties in assessing the results of INTENT due to the lack of comparative studies. In Chapter 5 we mentioned that the published trading systems studies on the British Pound and the Deutschmark have been conducted using different test periods from the periods that we have used. Because of these difficulties, we undertook an empirical assessment of the performance of the expert trader in making decisions using the same data and test period used by INTENT.

The method for evaluating the performance of the expert trader was as follows. We firstly printed out the **daily** graphs of the prices, open interest and volume of

	Deutschmark ($C = 4$)
Trading-Period-Years	4.03
Total-Closed-Trades	11.0
Total-Proft-Trades	8.0
Total-Losing-Trades	3.0
Profitable-Trades-pcn	**72.0**
Total-Gains-$	7744.0
Initial-Capital-$	100000.0
Capital-After-Trading-$	107744.0
Av-Gains-per-Trade-$	**704.00**
Max-Proft-Trade-$	3802.47
Max-Losing-Trade-$	-752.57
Av-Proft-Trade-$	2489.5
Av-Losing-Trade-$	-605.54
Maximum-Drawdown-$	-752.5
Buy-Signals-pcn	100
Profit-Buy-Trades	72.0
Profit-Sell-Trades	0
Return-per-year-pcn	1.9
Std-Dev-of-PL	1567.29

Table 9.4: Deutschmark Decision Combination ($C = 4$) (1988–1992)

the British Pound and the Deutschmark for the same period as was used to test the INTENT system (1988–1992).

The daily price graph was then covered by a sheet of paper and shifted one day at a time (see figure 9.6, which is a copy of the worksheet used by the expert). At each such shift the trader was given the option of making one of three decisions — he could buy, sell or do nothing. The decisions taken by the trader at each entry were noted down and were later evaluated by the same criteria used to evaluate the other modules of the INTENT system. [1]

[1] Although we intended to do the experiments with both data series, due to circumstances beyond our control we could only complete the British Pound experiments with the expert trader.

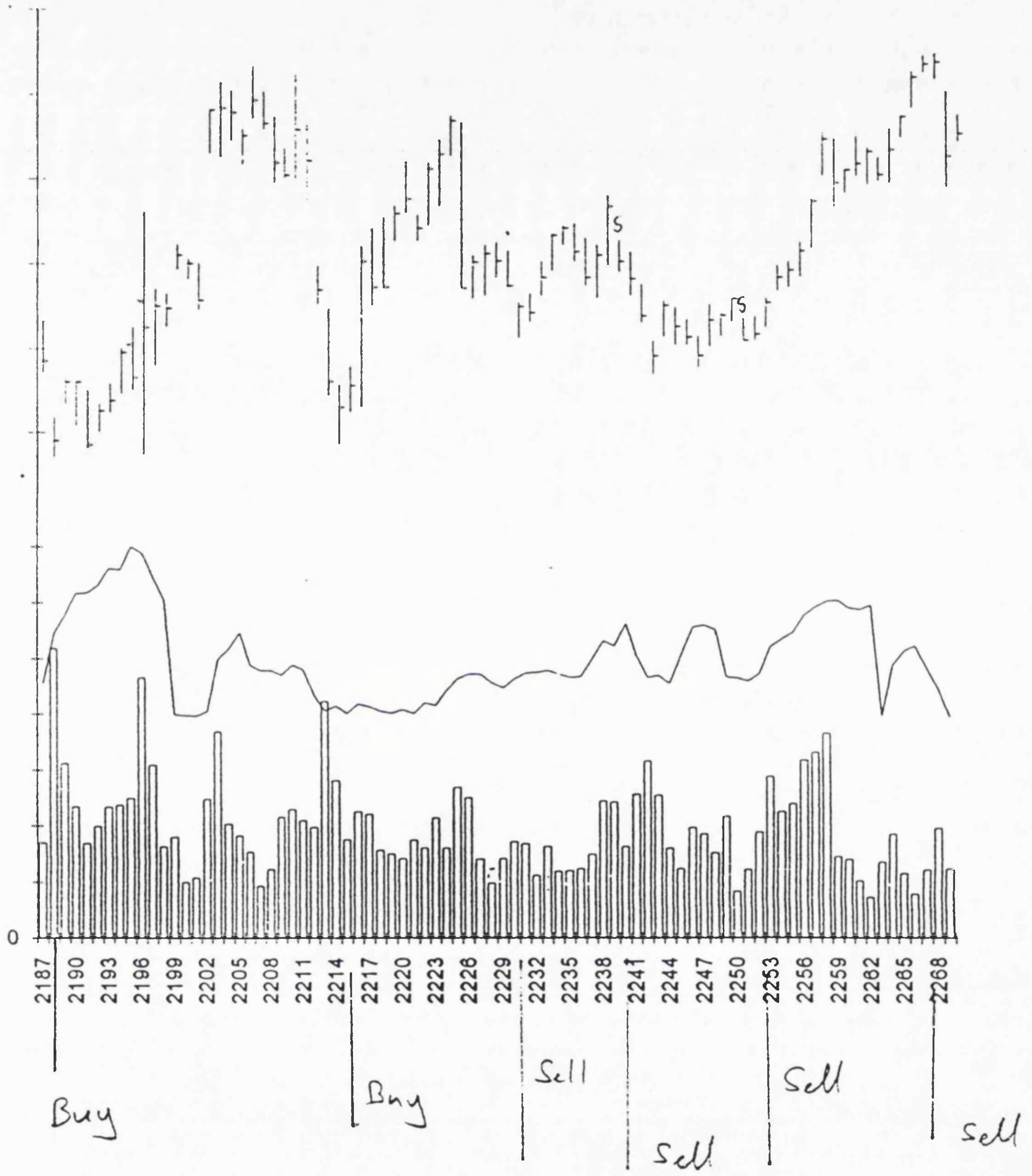


Figure 9.6: Expert Trading Worksheet

Particular precautions were taken to maximise the chances of the experiment being a fair one. The names of the markets were not disclosed to the trader, and we blanked the values of the prices, open interest and volume. This is because traders are known to have good memories for price values and subsequent events. (If the prices are disclosed to the trader, the trader has a good chance of guessing the type of financial market.) For example if the price is in the range of 2.1238 and 2.8989, the trader can probably deduce that the market is the British Pound and if the range is between 20,000 and 30,000 the chances are that it is the Japanese Nikkei stock index. And, if the prices are displayed, the trader can probably remember particular events near certain periods, say when the British Pound was trading near the 1.99 level against the dollar.

The results of the expert trader's trading performance is summarised in table 9.5 and the corresponding equity curve is given in figure 9.7. The expert achieved 64.2% correct trading decisions and an average of $633.35 per trade. These results will be discussed further in the next section.

This same procedure of evaluating human expert performance can be used in most projects that attempt to automate particular decision-making tasks. In the case of loan evaluation, for example, a similar experiment can easily be conducted by presenting the human expert with randomly-chosen samples of cases and evaluating his or her performance.

In domains where there can be significant variations in judgement among experts, as for example in specialist areas of medical decision making, then an average of several experts' decisions should be taken. An extension of such a consensus expert evaluation is to adopt particular weighting schemes such as the ones discussed in the multiple-model strand. Here the decisions of particular experts, say of a senior consultant, can be given a higher weighting than (say) the decisions of a general practitioner.

9.3 Performance Evaluation

In the previous chapters we have presented the results of the different modules in the INTENT system. We shall now briefly summarise these results, make comparative assessments between the different modules and then finally assess these results against other published studies.

	Expert Trader Performance
Trading-Period-Years	4.03
Total-Closed-Trades	14.0
Total-Proft-Trades	9.0
Total-Losing-Trades	5.0
Profitable-Trades-pcn	**64.28**
Total-Gains-$	8866.94
Initial-Capital-$	100000.0
Capital-After-Trading-$	108867.0
Av-Gains-per-Trade-$	**633.35**
Max-Proft-Trade-$	4903.21
Max-Losing-Trade-$	-2911.8
Av-Proft-Trade-$	1872.6
Av-Losing-Trade-$	-1597.29
Maximum-Drawdown-$	-5610.11
Buy-Signals-pcn	42.85
Profit-Buy-Trades	83.33
Profit-Sell-Trades	50.0
Return-per-year-pcn	2.12
Std-Dev-of-PL	2239.11

Table 9.5: Expert Trader Performance – British Pound (1988-1992)

As discussed in Chapter 5, the expert system yielded 57.8% correct trades and an average gain per trade of $449.5 for the British Pound and 53% correct trades and gain per trade $122.2 for the Deutschmark. The fuzzy system described in Chapter 6 produced 64.34% correct trades and an average gain per trade of $618.21 using British Pound data while Deutschmark data produced 55.2% correct trades and an average gain of $140.3 per trade.

The genetic algorithms presented in Chapter 7 were used to induce rules in two modes: symbolic and fuzzy. In the symbolic mode two sets of experiments were conducted (A and B) using two different types of variables (oscillators and moving averages). The set A experiments on average produced 56.5% correct trades while the set B produced only 50% correct trades. The average gain per trade was $221 for the set A experiments while the set B had an average loss per trade of $-157.97. In the fuzzy mode the genetic algorithm produced 60% correct trades and an average gain per trade of $699.60 using British Pound data and 62% correct trades and $162.95 using Deutschmark data.

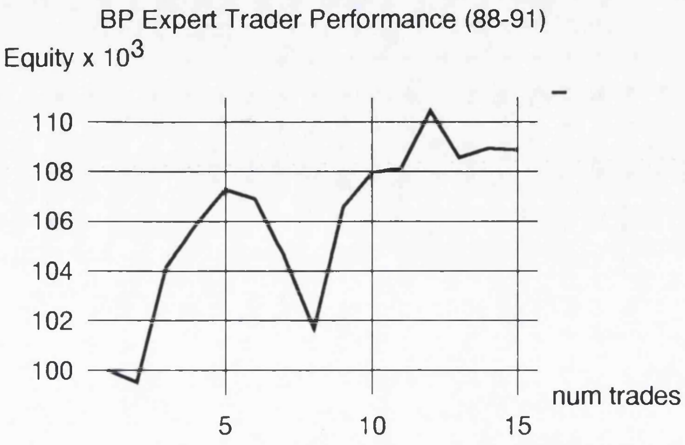


Figure 9.7: British Pound Expert Trader Performance Equity Curve (1988–1992)

	Proft. Trades (%)	Av. Gains per Trade ($)
Expert System	55.8	285.89
Fuzzy System	59.17	364.7
Genetic Algorithm (Symbolic)	53.25	31.0
Genetic Algorithm (Fuzzy)	61.0	431.3
Neural Network (Symbolic)	62.0	461.5
Neural Network (Fuzzy)	65.0	580
Decision Combination ($C = 3$)	73.87	708
Decision Combination ($C = 4$)	74.0	812
Expert Trader	64.2	633.35

Table 9.6: Comparison of all INTENT Methods

The neural networks presented in Chapter 8 use variables selected by the genetic algorithm in two modes — symbolic and fuzzy. The neural networks using symbolic data on average produced 62% correct trades and an average gain of $461.5 while the networks using fuzzy data produced 65% correct trades and an average gain of $580.

The decision combination approach presented earlier in this chapter combines the decisions of all the different modules in INTENT. Experiments were conducted using $C = 3$ and $C = 4$ decision consensus thresholds. A threshold of $C = 3$ on average produced 73.87% correct decisions with an average gain per trade of $708 while a $C = 4$ threshold produces an average of 74% correct decisions and an average gain per trade of $812.

We summarise these results in table 9.6 which presents the **average** performance of all decision modules (trading the British Pound and Deutschmark).

In terms of best percentage of correct decisions as well as the highest gain per trade, the best approach was the decision combination ($C = 4$) method. In terms of the percentage of correct decisions, three out of eight of our decision modules were better than the 64.2% accuracy of the expert. These were the two decision-combination schemes ($C = 3$ and $C = 4$) and the neural network using fuzzy data. In terms of highest average earnings, two of our modules (decision combination $C = 3$ and $C = 4$) were better than the expert's average of $ 633.35 per trade.

The overall INTENT results also compare favourably with published trading systems results although it has to be noted that the test periods are different. All of the modules in the INTENT system produce better results than Kaufman's [67] studies using moving average crossovers for the British Pound and the Deutschmark. He tested his system between 1970 and 1979 and obtained an average of 52.5% profitable trades.

All the INTENT modules apart from the genetic algorithm in the symbolic mode report better average results than Colin's [28] results on using neural networks for trading the British Pound and the Deutschmark where he obtained 55% correct trades between 1988 and 1991.

As discussed in the previous chapters, INTENT results have also been qualitatively assessed by the domain expert. In the expert-fuzzy-genetic strand the rules induced by the genetic algorithm rules were verified by the expert and the majority of them were pronounced as acceptable trading rules. In the case of a few rules the expert suggested minor changes to the conditions in the rules in order to make them more suitable for trading.

In the genetic-neural strand, the variables selected by the genetic algorithm for the neural network were also presented to the expert and his comments were elicited. The majority of the selected variables were ones that he considered useful in predicting future values of the trading data. There were a few selected variables that the expert considered redundant because they were very similar to other selected variables. Due to time limitations we have not validated this assertion (by training neural networks without these particular variables), but ideally such variable validation exercises should be carried out as part of a general method for refining such hybrid decision models.

As mentioned in Chapters 5 and 6 the results of data pre-processing procedures in INTENT were also verified by the domain expert. While the expert broadly agreed with the classifications produced by the pre-processing schemes, he also cautioned against the limited life-time of the derived classes as the characteristics of the variables change over time.

A key qualitative difference between the expert-fuzzy-genetic strand and the genetic-neural strand is in the level of explanation provided to the user. In the expert-fuzzy-genetic strand explicit models of the decision processes are derived while the genetic-neural strand produces decision models that are not transparent. As the expert remarked in the previous chapter, that although the genetic-neural system produces better quantitative results, the ability to understand explicitly and change the decision models is seen as vital for trading applications.

Similarly, although the decision combination method produces results superior to any other approach, its explanation capabilities are limited. This is because the decisions of all the different modules are aggregated thus making it difficult to isolate a single decision reasoning procedure. One suggestion is to analyse the frequency of a particular module contributing to the combined outcome (e.g. the frequency of fuzzy system decisions in C). If it is clear that a particular module(s) is contributing more frequently than others, then its contributing rules can be presented to the user giving an elementary level of explanation. However, if the frequently contributing module is a neural network, then it is not possible to give such details of decision procedures.

From the explanation perspective the unique feature of the expert-fuzzy-genetic strand is the provision of *linguistic* explanations. As discussed in Chapter 7 our domain expert found the induced rules containing commonly used linguistic categories, such as low, medium and high, very easy to understand and modify. A common problem in many organisations is that most decision support models use statistical methods which are applied and understood only by specialist personnel. Most high level decision makers do not have mathematical backgrounds and therefore very rarely have 'hands-on' experience in interacting with data analysis and decision support tools [87]. When analyses are obtained by a statistical model in an organisation, they are often converted into 'common language' by the specialist staff for the benefit of the decision makers.

On the other hand, the decision models induced by the expert-fuzzy-genetic strand do not need to be re-interpreted by an expert in hybrid systems, but instead can be used directly by decision makers at all levels without any specific specialist training. This, we believe, has wide implications in decision support systems known as

Executive Information Systems (EIS) which currently rely on graphical means (animation, 3D graphs etc.) to convey information about an organisation. Graphical methods that are used to present complex relationships in data as pictures are generally accepted as being 'decision-maker friendly'. We believe the decisions models presented in this research are very close to another category of decision-maker friendly information, namely natural language.

In order to assess the benefits of our decision models fully, studies need to be conducted where systems such as INTENT are in daily use in organisations over a period of time. In such a trial study it will be useful to gather answers to the following types of questions. How often do the decision makers judgmentally revise the decision models induced by the system ? Do all decision makers irrespective of their training find the decision models easy to understand ? When conveying decisions (based on models induced by our system) to others in the organisation, do the decision makers use the same linguistic descriptions (e.g. open interest low) that were produced by the system or do they modify them by the use of adverbs ? Answers to such questions will help to identify areas of the hybrid system that needs further refinement.

9.4 Towards a Methodology of Hybrid Systems

In the last five Chapters we have detailed the functioning of the three different hybrid strands of the INTENT system and the intelligent systems modules from which they are composed. In this section we draw upon the experiences of this project and attempt to formulate a preliminary set of guidelines for hybrid systems development.

Our first step in constructing a hybrid system was to list all the computational requirements of our application domain (see Chapter 4). These were automated knowledge acquisition, ability to cope with brittleness, continuous learning, providing explanations, and the incorporation of judgmental knowledge. Secondly, a matrix was constructed by listing known intelligent techniques and their abilities against these domain requirements (see Chapter 4, figure 4.1). In each cell of the matrix a rating scheme was used (we used '$\surd$') to denote the competence of the intelligent technique in dealing with that particular computational property. Let us call this matrix the *property assessment matrix.*

The next step was to analyse the property assessment matrix to see whether there were any intelligent techniques that had a high rating in all properties. If there was such a technique, we could have used that technique alone without any need to consider a hybrid approach.

As there was no such technique, the next step was to identify the combinations of intelligent techniques that had high ratings between them. In our task, genetic algorithms, fuzzy systems and expert systems, collectively had the desired high ratings. The task was then to find a mechanism to combine them in a manner so that the resulting hybrid could contain all the desired properties. In this instance, the genetic algorithm was used to induce the decision-making knowledge for the expert and fuzzy systems, thus *replacing the function* of manually specifying knowledge by a domain expert.

In our particular application, all the desired functions were *overlapping functions* and not *sequential functions.* That is, the structure of the problem was such that all the needed functions were required simultaneously and they were not distinct. However, if the problem had been composed of distinct, sequential functions then different intelligent techniques could have been used to solve the different tasks specified in the property assessment matrix. For example if the desired properties in the matrix consisted of a prediction task (e.g. energy prediction) followed by an optimisation task (e.g. optimisation of the distribution of the energy) then a neural network could be used for the prediction task and a genetic algorithm for the optimisation task. Thus, an *intercommunicating hybrid* could be used in such an application.

Based on our experiences gained from the design of the INTENT, the following contains a set of preliminary steps to be followed in the development of hybrid systems. These steps draw upon the above mentioned property assessment exercises and the hybrid classes introduced in Chapter 3. It has to be stressed that this is only a 'sketch' of a set of guidelines which (with further practical experience in appropriate applications and assessment of any apparent needs for improvement) should eventually lead to a more comprehensive methodology for hybrid systems. These steps are:

1. List all the desired computational properties.
2. List all the properties of available intelligent techniques and construct the *property assessment matrix.*
3. If one intelligent technique has high ratings in all properties, then use that technique alone.

ELSE

4. List the functions that are distinct or sequential.
5. If there are *sequential functions*, then create an *intercommunicating hybrid* which consists of techniques that are capable of solving the different subtasks.
6. If there are no sequential functions then create a *function-replacing hybrid* where different techniques are combined to produce a hybrid that satisfies all the desired properties.

The above scheme is straightforward and we believe is general enough to be applied to most industrial hybrid systems projects. A difficulty that may arise is that some hybrid systems developers may not be aware of all the computational properties of different intelligent techniques available at a given time. This will lead to an incorrect property assessment matrix and a flawed subsequent analysis.

A possible solution to this problem is to produce an *intelligent systems property handbook* that is constantly updated and which is made widely available to researchers and developers. Such a handbook can be an analogue of 'drugs effects handbooks' used in the medical profession which list information such as the types of ailments the drugs are used for, the chemical composition of the drugs and the side-effects of the drugs. In the intelligent systems handbook relevant information will include the previously described computational properties and details of successful hybridisation with other techniques.

A further limitation of the above scheme is that it does not specify methods by which knowledge can be communicated between different intelligent techniques. This research has demonstrated the benefits of using production rules for communicating knowledge between expert, fuzzy and genetic systems. However production rules are probably one of many knowledge representation schemes that can be used for this purpose, and further research is very much needed to evaluate the potential of other schemes such as frames and semantic networks.

Another aspect that is not covered in the above scheme is the involvement of domain experts in the development of hybrid systems projects. As we have seen in our own research, having access to domain expertise has been of paramount importance. Firstly, the domain expertise is needed to identify a possible 'entry' into the problem. In our financial trading example the domain expert explained the two broad approaches to solving the trading problem (fundamental analysis and technical analysis) and helped us to choose the more appropriate (technical analysis). Domain expertise is also needed to specify the broad space of relevant variables for solving the problem at hand. In our research these variables consisted of technical trading indicators. It was from this broad space of variables that the genetic algorithm was able to 'prune' or identify the most important variables for the subsequent decision model building.

Another area of expert involvement in which we have gained benefits is in the verification of hybrid system results. The expert verification of hybrid systems results has helped us to identify the most promising architectures based on criteria such as explanation capabilities. For example, although the genetic-neural hybrid strand produced better quantitative trading results than other hybrids, the expert found the explanation capabilities too limited. Expert interaction has also allowed us to propose extensions to the system such as the creation of adjustments to the decision boundaries (to minimise mis-classifications) in the case of the symbolic pre-processing algorithm.

Finally, we believe that the above mentioned scheme, which has been discussed in the context of hybrid systems between intelligent techniques, can also be extended to hybrids of intelligent techniques and 'conventional' mathematical methods such as linear programming and regression. Similar to intelligent techniques, these different methods have different computational properties which make them suited for particular tasks over others. The construction of computational property tables for these mathematical methods may in fact turn out to be easier than constructing property tables for intelligent techniques as these techniques have been studied in greater depth and their application well understood.

Chapter 10
Conclusions

This chapter presents conclusions of this thesis and suggests avenues for future research.

10.1 Conclusions

This thesis has explored several key issues in intelligent hybrid systems and in intelligent decision support systems. A significant contribution of the thesis is the formulation of a new classification scheme for intelligent hybrid systems which takes into account functionality, processing architecture and communication requirements. The three-fold classification scheme (function-replacing, intercommunicating and Polymorphic) provides a clear mechanism to make qualitative assessments of hybrid systems.

The hybrid system developed as part of this research, INTENT, combines expert systems, fuzzy systems, genetic algorithms and neural networks, and has been demonstrated in the complex decision making task of foreign exchange trading. It consists of three hybrid strands — expert-fuzzy-genetic, genetic-neural and multiple model. While the expert-fuzzy-genetic strand is an function-replacing hybrid according to our classification scheme, the genetic-neural and multiple model strands are intercommunicating hybrids.

We have proposed a sketch of a methodology for hybrid systems development based on our classification scheme and experiences obtained during the construction of INTENT. The scheme involves making assessments of computational requirements of the problem domain and rating available intelligent techniques. We view this as an important initial stage in the development of a comprehensive methodology for hybrid systems.

The trading decision-making task involved constructing a system that had the following properties: the ability to acquire trading knowledge, the ability to cope with brittleness, continuous learning, transparency of decision model and the ability to incorporate human knowledge. After assessing different available intelligent techniques it was clear that no single technique could be used for this task. Therefore a hybrid system combining several intelligent techniques with complementary computational properties was developed.

A key issue that needed to be resolved was the issue of communication of knowledge between the different intelligent systems. By using production rules with the same syntactic structure as the common knowledge representation scheme, we have demonstrated the ability to transfer knowledge between the genetic algorithm, expert and fuzzy systems. The production rule format also facilitates users to easily understand the reasoning process and also allows users to easily change existing knowledge and to add new knowledge. Further research is needed to evaluate the use of other knowledge representation scheme such as frames for such knowledge communication purposes.

In order to capture the meaning of linguistic terms used by domain experts such as *low, medium* and *high*, a novel method for pre-processing data has been developed (see section 5.3) . The algorithm which uses Single Linkage Clustering converts raw data (e.g. volume 15612) into symbolic representations (e.g. volume *high*). An extension of this algorithm is used in the fuzzy system to convert raw data into fuzzy membership functions (e.g. low [0.0] medium [0.2] high [0.8]). The results of these procedures have been verified by the domain expert (see section 6.2). There are many potential applications of this procedure including in the area of cardiology diagnosis where linguistic categories such as *moderate, normal, mild* for describing readings of cholesterol, blood pressure, maximum heart rate and others [102]. Using the above algorithms such linguistic categories can be automatically derived using raw data (e.g. max-heart-rate 180, blood pressure 130, cholesterol 219) from readings obtained from measuring instruments.

Packard's [93] genetic algorithm for complex data analysis has been used for inducing rules in the INTENT system. An important contribution of this research is the extension of Packard's system to induce fuzzy rule bases (see section 7.3). This provides a method to automate the construction of fuzzy rule bases which is usually a time consuming trial and error process. There are many parameters in the hybrid system which have been set arbitrarily (e.g. the cut-off values for the fuzzy system and neural networks) and the genetic algorithm can also be extended to search such parameter spaces.

Another extension to Packard's system is the feedback mechanism to the expert and fuzzy rule bases (see section 7.6). Here an assessment of similarity between existing expert or fuzzy rules and new genetically induced rules is made and only if the rules are very 'dis-similar' are they added to the expert rule bases. This is made possible because of the common knowledge representation schemes used by the different intelligent systems. This mechanism thus provides a mechanism for 'merging' machine generated knowledge with expert specified knowledge. This mechanism may provide particular benefits in domains where there are already operational expert systems such as credit evaluation where this type of genetic mechanism can help to induce relationships from past data and then augment the new knowledge with the existing knowledge.

A further conclusion of this thesis is the confirmation of the effectiveness of decision combination using different intelligent systems (see section 9.1). In the multiple model strand of the INTENT system decisions from the expert system, fuzzy system, genetic algorithms and neural networks are combined. The results have confirmed the usefulness of decision combination approaches in intelligent systems which have been demonstrated elsewhere in applications such as protein structure prediction [130].

A central problem in the trading application domain has been the lack of comparative studies. There are very few published studies on foreign exchange trading performance and most studies do not use the same data sample that we have used. However, it is likely that many trading organisations are conducting similar research, but their experiences are not published because of the competitive nature of the application domain. Because of this reason an empirical investigation of the effectiveness of expert human trader performance was conducted. This provided us with a unique set of measures to compare the performance of INTENT.

The INTENT system was evaluated by making simulated trades for trading the British Pound and the Deutschmark. On the two key measures of trading performance, correct percentage of trades and average gain per trade, the best approach was the decision combination ($C = 4$) method. In terms of the percentage of correct decisions, three out of eight of our decision modules were better than the 64.2% accuracy of the expert. These are the decision combination ($C = 3$ and $C = 4$) and the neural network using fuzzy data. In terms of highest average earnings, two of our modules (decision combination $C = 3$ and $C = 4$) were better than the expert's average of \$633.35 per trade.

All induced knowledge have been evaluated by the domain expert to determine its quality. The general conclusion has been that the induced knowledge is similar to trading knowledge used by our domain expert and that the majority of induced rules can be used without modification. The issue of transparency of decision model has been addressed and the domain expert has concluded that the knowledge representation format is sufficiently general and easy to interpret. The knowledge representation scheme also facilitates the modification of induced knowledge by the domain expert. Further studies of our system being used in decision-making environments are needed to fully assess the receptiveness to the approach among different decision makers. Feedback in the form of the frequency of judgmental revisions to the induced linguistic models and the nature of those revisions will help to guide the further development of the INTENT system.

This thesis also makes a contribution to the general area of *knowledge extraction*. Experts can be helped in the expression of their expertise by providing them with easy to understand linguistic rules (induced from raw data using INTENT) which can then be further refined to suit individual needs. This type of tool can supplement current manual approaches to knowledge extraction (say) based on the analysis of verbal protocols [68]. Another facility in INTENT that can further enhance the knowledge extraction process is the allowance of expert knowledge as 'seeds' for the genetic algorithm (see section 7.6.2). An expert can input a particular set of rules and the genetic algorithm will evaluate them (using past data), and progressively fine-tune them over many cycles, thus providing a mechanism for expert knowledge refinement.

10.2 Future work

An important but relatively simple extension to the INTENT system is the provision of *fuzzy hedges* to the existing system. Fuzzy hedges such as *very, somewhat, rather, extremely* were devised by Zadeh [133] to further qualify linguistic variables in fuzzy systems. These constructs which play a similar role to adverbs in English sentences are constructed by changing the shape of a fuzzy set. The fuzzy hedge *very* (as in *very* low volume), for example, intensifies the fuzzy space by squaring the membership function at each point in the fuzzy set.

The genetic algorithm can be modified to induce relationships with such fuzzy hedges. An example rule that may be induced is 'If the price is *somewhat* **high** AND the open interest is *extremely* **low** THEN Buy'. Users of the system would also be able to change existing knowledge and add new knowledge using such qualifiers. Such extensions would further increase comprehensibility of the decision models and increase the accessibility of the system to decision makers.

In order to check the validity of our ideas independent of particular application details, we propose to apply the INTENT system in other areas such as organisational resource allocation. A common problem in most organisations is the allocation of resources depending on the utility or profitability (or potential profitability) of different type of customers. Our hybrid system can be used in such tasks to uncover relationships in the customer data which can then be used to make strategic decisions.

For example, if a credit card company needs to understand the behaviour of its customers then our system can be used to induce relationships between the behavioural variables characterising customers (income, age, credit balance etc.) and the variables that characterise profitability. The first step, as in the trading application, is the conversion of raw data items (e.g. age 35, income 15,000 etc.) into linguistic categories by the means of the clustering algorithm. Example linguistic categories that can be specified are: for age - young, middle-aged and senior, for income and credit balance - low, medium and high.

The INTENT induction mechanisms can then be applied (with or without expert knowledge) to produce decision rules in the form of,

IF *senior* aged AND the credit balance is *high* THEN the customer profitability is *high*

IF the customer has *low* income AND is *young* AND has a *medium* credit balance THEN the customer profitability is *low*

Such linguistic decision rules will enable most decision makers, regardless of whether they have a technical background or not, to easily understand the underlying relationships and to make judgmental revisions to these models. A variety of resource allocation decisions can then be based on the resulting analysis. For example, a decision maker may decide to allocate less resources (such as advertising) towards customers who are not profitable and to increase resources to the more profitable customers.

As with the trading decision making problem the INTENT system should ideally be set to continuously learn from any changes in the decision making environment. Especially in times of dramatic economic changes such as in the case of a recession, the behaviour of customers (e.g. spending habits) can change dramatically. If the system learns continuously, it will induce new rules that capture such changes. Further, as in the trading domain, the facility to revise decision models judgmentally is a very useful mechanism for dealing with unusual circumstances such as the development of extraordinary competitive conditions.

REFERENCES

[1] Chartists tame chaos. *The Economist*, page 70, Aug 15 1992.

[2] Cheap and cheerful. *The Economist*, page 56, February 27 1993.

[3] Disagreeing about the consensus. *The Economist*, page 77, July 27 1991.

[4] David H. Ackley, Geoffrey E. Hinton, and Terrence J. Sejnowski. A learning algorithm for boltzmann machines. *Cognitive Science*, 9:147–169, 1985.

[5] N. Ahmad. Real-time intelligent distributed control systems. Technical report, Dept of Electrical Engg, Indian Institute of Technology, New Delhi, 1992.

[6] Venkat Ajjanagadde and Lokendra Shastri. Rules and variables in neural nets. *Neural Computation*, 3(1):121–134, 1991.

[7] I. Aleksander and H. Morton. *Introduction to Neural Computing.* North Oxford Press, 1990.

[8] S. Armstrong. *Long-range Forecasting.* John Wiley, New York, 1985.

[9] R. Palmer P. Tayler B. Arthur, J. Holland. Using genetic algorithms to model the stockmarket. In *The proceedings of the Forecasting and Optimisation in Financial Services conference*, London, 1991. IBC Technical Services Ltd.

[10] Bruce Babock. The dow jones-irwin guide to trading systems. pages 34–45. Dow Jones-Irwin, Illinois, 1989.

[11] H. Barenji. Fuzzy logic controllers. In R. Yager and L. Zadeh, editors, *An Introduction to Fuzzy Logic Applications in Intelligent Systems.* Kluwer Acdemic Publishers, 1992.

[12] R. Beale and T. Jackson. *Neural Computing – an Introduction.* Adam Hilger, Bristol, 1990.

[13] L.D. Belveal. Charting commodity market price behaviour. Dow Jones-Irwin, Homewood, Illinois, 1985.

[14] K. Bergerson and D. Wunsch. A commodity trading model based on a neural network-expert system hybrid. In *Proceedings of International Joint Conference for Neural Networks*, Seattle, 1991.

[15] L. Blume and D. Easley. Evolution and market behavior. *Journal of Economic Theory*, 58:9–40, 1992.

[16] D. Bobrow. Dimensions of interaction. *AI Magazine*, Fall 1991.

[17] Margaret A. Boden. *Artificial Intelligence and Natural Man.* MIT Press, 1987.

[18] Margaret A. Boden. *Computer Models of Mind.* Cambridge University Press, 1988.

[19] H. Braun and J. Chandler. Predicting stock market behaviour through rule induction: An application of the learning-from-example approach. *Decision Sciences*, 18:415–29, 1987.

[20] Richard Brealey and Stewart Myers. Principles of corporate finance. page 229. McGraw-Hill Inc, New York, 1991.

[21] D. Bunn. The synthesis of predictive models in marketing research. *Journal of Marketing Research*, 16:280–83, 1979.

[22] D. Bunn. Forecasting with more than one model. *Journal of Forecasting*, 8:161–166, 1989.

[23] D. Bunn. Synthesis of expert judgment and statistical forecasting models for decision support. In G. Wright and F. Bolger, editors, *Expertise and Decision Support*. Plenum Press, 1992.

[24] D. Bunn and G. Wright. Interaction of judgmental and statistical forecasting methods: Issues and analysis. *Management Science*, 37(5), 1991.

[25] J.Y. Choi and C-H. Choi. Sensitivity analysis of multilayer perceptron with differentiable activation functions. *IEEE Transactions on Neural Networks*, 3(1):101–107, January 1992.

[26] P. M. Churchland. *A Neurocomputational Perspective*. MIT Press, 1989.

[27] W.J Clancy. The epistemology of a rule-based expert system: a framework for explanation. *Artificial Intelligence*, 20:215–251, 1983.

[28] A. Colin. Neural networks for exchange rate forecasting. In *The proceedings of the fourth European seminar on Neural Computing*. IBC Technical Services Ltd., London, 1991.

[29] I.F. Croall and J.P. Mason. *Applications of Neural Networks for Industry in Europe 1988-1991 - Project Handbook*. U.K. Atomic Energy Authority, Oxford, 1991.

[30] L. Davis. *Handbook of Genetic Algorithms*. Van Nostrand Reinhold, New York, 1991.

[31] R. Davis, B. Buchanan, and E. Shortcliffe. Production rules as a representation for a knowledge-based consultation program. *Artificial Intelligence*, 8(1):15–45, 1977.

[32] D.E.Goldberg. *Genetic Algorithms In Search, Optimisation and Machine Learning*. Addison-Wesley, 1989.

[33] Joachim Diederich. An explanation component for a connectionist inference system. In *ECAI-90, 9th European Conference on Artificial Intelligence*. Pitman Publishing, London, 1990.

[34] M. Skolnick D.Powell and S. Tong. Interdigitation: A hybrid technique for engineering design optimization employing genetic algorithms, expert systems and numerical optimization. In L. Davis, editor, *Handbook of Genetic Algorithms*. Van Nostrand Reinhold, 1991.

[35] Jurgen Dunker, Andreas Scherer, and Gunter Schlageter. Integrating neural networks into distributed knowledge based systems. *Proc of 12th conf on Artificial Intelligence, Expert Systems and Natural language, Avignon, France*, 1992.

[36] P. McCorduck E. Feigenbaum and H. Nii. *The Rise of the Expert Company.* Macmillan, London, 1988.

[37] D. Barrow M. Gerrets E. Haasdijk, R. Walker. Genetic algorithms in business. In J. Kingdon J. Stender, E. Hillebrand, editor, *Genetic Algorithms in Optimisation, Simulation and Modelling*, pages 172–174. IOS Press, 1994.

[38] R. Edwards and J. Magee. *Technical Analysis of Stock Trends.* John Magee Inc., Boston, 1984.

[39] K.E. Ericsson and Simon H.A. *Protocol Analysis - Verbal Reports as data.* The MIT Press, Cambridge, MA, 1984.

[40] J. Essinger. *Artificial Intelligence: Applications in financial trading and investment management.* Euromoney Publications.

[41] K.F. Wallis et al. *Models of the UK Economy: Reviews 1-5.* Oxford University Press, Oxford, 1988.

[42] C. Evans. A case-based assistant for diagnosis and analysis of dysmorphic syndromes, phd thesis. Technical report, Department of Computer Science, University College London, In preparation.

[43] E. Feigenbaum. Themes and case studies of knowledge engineering. In D. Mitchie, editor, *Expert Systems in the micro-technology age.* Edinburgh University Press, 1979.

[44] J.L. Ribeiro Filho and P. Treleaven. Game's parallel programming model. In J. Kingdon J. Stender, E. Hillebrand, editor, *Genetic Algorithms in Optimisation, Simulation and Modelling*, pages 111–151. IOS Press, 1994.

[45] Jerry A. Fodor and Zenon W. Pylyshyn. Connectionism and cognitive architecture : a critical analysis. *Cognition*, 28:3–71, 1988.

[46] R. Forsyth. *Machine Learning: Applications in expert systems and information retrieval.* Ellis Horwood Ltd., 1986.

[47] Stephen I. Gallant. Connectionist expert systems. *Communications of the ACM*, 31(2):152–169, February 1988.

[48] S. Goonatilake and S. Khebbal. Intelligent hybrid systems. In *Proceedings of the First Singapore Conference on Intelligent Systems.* Singapore Computer Society, 1992.

[49] S. Goonatilake and S. Khebbal. Intelligent hybrid systems. Internal Report 92/2/IN, 1992.

[50] Vijay Shah Gregory Madey, Jay Weinroth. Hybrid intelligent systems: Tools for decision making in intelligent manufacturing. In *Neural Networks for Intelligent Manufacturing.* Chapman Hall, 1993.

[51] Biological Modelling Research Group. Nevprop user manual. University of Nevada.

[52] T. Yoneda Y. Katoh Y S. Tano H. Yuize, T. Yagyu and S. Fukami. Decision support system for foreign exchange trading. In *International Fuzzy Engineering Symposium '91*, pages 971–982, 1991.

[53] C. Hammond. *Computer Systems in Investment Management - Masters thesis.* Dept. of Computer Science, University College London, 1989.

[54] David A. Handelman, Stephen H. Lane, and Jack J. Gelfand. Intergrating knowledge-based system and neural network techniques for robotic skill acquisition. In *Proceedings of IJCAI, Detroit*, pages 193–198. Morgan Kaufmann, August 1989.

[55] P. Harmon and D. King. *Expert Systems: Artificial Intelligence in Business.* John Wiley, 1985.

[56] A. Hart. Role of induction in knowledge elicitation. In A. Kidd, editor, *Knowledge Acquisition for Expert Systems: A practical handbook.* Plenum Press, 1987.

[57] J. Hartigan. *Clustering Algorithms.* John Wiley and Sons, New York, 1975.

[58] J. Holland. Escaping brittleness. In R. Michalski, J. Carbonell, and T. Mitchell, editors, *Machine Learning*, volume 2. Morgan Kaufmann, 1986.

[59] J.H. Holland. Genetic algorithms and classifier systems: Foundations and future directions. In *Genetic Algorithms and their Applications: Proceedings of the 2nd International Conference on Genetic Algorithms*, 1987.

[60] C. Hughes. Neural networks and expert systems - a partnership. In *Proceedings of the IEE Colloquium on the Integration of Neural Networks and Knowledge based systems.* IEE London, 1992.

[61] A. Shleifer J. Lakonishok and R. Vishny. Contrarian investment, extrapolation and risk. Technical report, University of Illinois, 1993.

[62] R. Palmer J. Rust and J. Miller. *Behaviour of Trading Automata in a Computerized Double Auction Market, in The Double Auction Market: Theories, Institutions and Experimental Evaluations.* Addison Wesley, 93.

[63] P. Jackson. *Introduction to Expert Systems.* Addison Wesley, 1986.

[64] J.M. Jenks. Non computer forecasts to use right now. *Business Marketing*, 68:82–84, 1983.

[65] K. Kandt and P. Yuenger. A financial investment assistant. In R. Trippi and E. Turban, editors, *Investment Management: Decision support and expert systems.* Van Nostrand Reinhold, New York, 1990.

[66] C. Karr. Genetic algorithms for fuzzy controllers. *AI Expert magazine*, 6, Feb. 1991.

[67] P.J. Kaufman. *The New Commodity Trading Systems and Methods.* John Wiley, 1987.

[68] A. Kidd. *Knowledge Acquisition for Expert Systems: A practical handbook.* Plenum Press, 1987.

[69] T. Kohonen. *Self-Organization and Associative Memory.* Number 8. Springer-Verlag, Berlin, 1989.

[70] J. Kolodner. *Case-Based Reasoning.* Morgan Kaufmann Publishers, San Mateo, CA, 1993.

[71] S.-G. Kong and B. Kosko. Adaptive fuzzy systems for backing up a truck-and-trailer. *IEEE Transactions on Neural Networks*, 3(2):211–23, March 1992.

[72] J. Koza. *Genetic Programming.* MIT Press, 1992.

[73] A. Lapedes and R.M. Farber. Nonlinear signal processing using neural networks: prediction and system modelling. Technical Report LA-UR-87-2662, Los Alamos National Laboratory, 1987.

[74] Y. le Cun. Generalization and network design strategies. CRG- R-89-4, Connectionist Research Group, University of Toronto, Toronto, June 1989.

[75] C.C. Lee. Fuzzy logic in control systems: Fuzzy logic controller - part i and part ii. *IEEE Trans. on Sys man Cybernetics*, Vol SMC-20(No.2):404–435, 1990.

[76] D.V. Lindley. *Making Decisions.* John Wiley and Sons, 1985.

[77] R.I. Lowe. Artificial intelligence techniques applied to transformer oil dissolved gas analysis, 1985.

[78] G. Lundberg and A. Barna. A knowledge based model of commodity trading expertise. *International Journal of Modelling and Simulation*, 7(4), 1987.

[79] V. Maniezzo M. Dorigo. Parallel genetic algorithms: Introduction and overview of current research. In J. Stender, editor, *Parallel Genetic Algorithms*, pages 5–42. IOS Press, 1994.

[80] B.G. Malkiel. *A Random Walk Down Wall Street.* W. Norton and Company, NewYork, 1990.

[81] E. Mamdani and S. Assilian. An experiment in linguistic synthesis with a fuzzy logic controller. *International Journal of Man-machine studies*, 7, 1975.

[82] D. McDermott. Extracting knowledge from expert systems. In *Proceedings of the 8th International Joint Conference on Artificial Intelligence*, pages 100–107, 1983.

[83] D. McNeill and P. Freiberger. *Fuzzy Logic.* Simon and Schuster, New York, 1993.

[84] L. Medsker and D. Bailey. Models and guidelines for integrating expert systems and neural networks. In A. Kandel and G. Langholz, editors, *Hybrid Architectures for Intelligent Systems.* CRC Press, 1992.

[85] R. Michalski and R. L. Chilansky. Learning by being told and learning from examples. *Journal of Policy Analysis and Information Systems*, 4, 1980.

[86] R. Miller. Computer aided financial analysis. page 117. Addison Wesley, New York, 1990.

[87] H. Mintzberg. *Mintzberg on Management: Inside our strange world of organisations.* Free Press, New York, 1989.

[88] G. Mitra. *Computer Assisted Decision Making.* North Holland, 1986.

[89] D. Montana and L. Davis. Training feedforward neural networks using genetic algorithms. In *Proceedings of 11th International Joint Conference on Artificial Intelligence*, pages 762–767, 1989.

[90] P. Moore. Risk in business decision. pages 121–143. Longman, London, 1972.

[91] A. Newell and H. Simon. GPS - a program that simulates human thought. In E. Feigenbaum and J. Feldman, editors, *Computers and Thought.* McGraw Hill, 1963.

[92] Paul Ormerod, John C. Taylor, and Ted Walker. Neural networks in economics. In Mark P. Taylor, editor, *Money and financial markets*, pages 341–353. Blackwell Ltd, Oxford, 1991.

[93] N.H. Packard. A genetic learning algorithm for the analysis of complex data. *Complex Systems*, 4:543–572, 1990.

[94] D. Partridge. Is intuitive expertise rule based ? Technical report, University of Exeter, UK, 1986.

[95] B. Pearlmutter. Discussion on the connectionist mailing list, 26 July 1993.

[96] J.R. Quinlan. Induction of decision trees. *Machine Learning*, 1:81–106, 1986.

[97] J.R. Quinlan. *C4.5:A Machine Learning Paradigm.* Morgan Kaufmann, 1993.

[98] A. Ramsey R. Barrett and A. Sloman. *POP-11: A practical language for artificial intelligence.* Ellis Horwood Series, Chichester:England.

[99] A.N. Refenes, M. Azema-Barac, and P.C. Treleaven. Financial modelling using neural networks. In H. Liddell, editor, *Commercial parallel processing.* Unicom, 1992.

[100] E. Rich. *Artificial Intelligence.* McGraw-Hill, 1983.

[101] K. Konolige R. Reboh R.O. Duda, P.E. Hart. A computer based consultant for mineral exploration. SRI International, 1979.

[102] R. Roiger. An exemplar-based approach to concept learning. *AISB Quarterly*, 82, 1992.

[103] Daniel E. Rose and Richard K. Belew. A connectionist and symbolic hybrid for improving legal research. *International Journal of Man-Machine Studies*, 35:1–33, July 1991.

[104] David E. Rumelhart and James L. McClelland. In *Parallel Distributed Processing - Explorations in the Microstructure of Cognition*, volume ONE. MIT Press, 1986.

[105] David E. Rumelhart and James L. McClelland. *Parallel Distributed Processing : Explorations in the Microstructure of Cognition Volume 1: Foundations.* MIT Press, Cambridge, Mass., 1986.

[106] A. Scherer and G. Schlageter. A multi agent approach for the integration of neural networks and expert systems.

[107] A. Scherer and G. Schlageter. Embedding neural networks in a cooperative framework: Philiosophy and architecture. In *Proceedings of AAAI Workshop on Integrating Neural and Symbolic Processes.* AAAI Press, 1992.

[108] E. Schoneburg. Stock price prediction using neural networks: a project report. *Neurocomputing*, 2:17–27, 1990.

[109] J.F. Schreinmakers and D.S. Touretzky. Interfacing a neural network with a rule-based reasoner for diagnosing mastitis. In *The proceedings of International Joint Conference for Neural Networks*, Washington, 1990.

[110] J. Severwright. Neural networks for direct marketing. In *The Proceedings of the IBC conference on Data Mining in Finance and Marketing.* IBC Technical Services Ltd., 1992.

[111] K. Shaleen. *Volume and Open Interest.* McGraw-Hill Book Company, London, 1990.

[112] E.H. Shortliffe. *Computer Based Medical Consultations: MYCIN.* Elsevier, New York, 1976.

[113] Herbert Simon. *The New Science of Management Decision.* Prentice Hall, 1960.

[114] P. Smolensky. Connectionist AI, symbolic AI, and the brain. *Artificial Intelligence Review*, 1:95–109, 1987.

[115] Richard Southwick. Explaining reasoning: an overview of explanation in knowledge-based systems. *The Knowledge Engineering Review*, 6(1):1–19, 1991.

[116] A. Stolcke. *Cluster Program Manual.* University of Colorado, 1992.

[117] Ron Sun. On variable binding in connectionist networks. *Connection Science: Journal of Neural Computing, Artificial Intelligence and Cognitive Research*, 4(2):93–124, 1992.

[118] M. Yoda M. Takeoka T. Kimoto, K. Asakawa. Stock market prediction system with modular neural networks. In *Proceedings of the International Joint Conference on Neural Networks*, volume 1, 1990.

[119] R.J. Teweles and F. J. Jones. *The Futures Game.* McGraw-Hill Book Company, 1987.

[120] C. Thorpe. In *Vision and Navigation: The Carnegie Mellon NAVLAB.* Kluwer Academic Publishers, 1990.

[121] H. Tirri. Implementing expert system rule conditions by neural networks. *New Generation Computing*, 10:55–71, 1991.

[122] P. Treleaven and S. Goonatilake. Intelligent financial technologies. In *Proceedings of Parallel Problem Solving from Nature, Applications in Statistics and Econmics.* EUROSTAT, 1992.

[123] B. LeBaron W. Brock, J. Lakonishok. Simple technical trading rules and the stochastic properties of stock returns. Technical report, The Santa Fe Institute, 1991.

[124] P.D. Wasserman. *Neural Computing: theory and practice.* Van Nostrand Reinhold, 1989.

[125] D.A. Waterman. *A Guide to Expert Systems.* Addison-Wesley, 1985.

[126] A. Weigend, B. Huberman, and D. Rumelhart. Predicting the future: a connectionist approach. *International Journal of Neural Systems*, 1(3):193–209, 1990.

[127] S.M. Weiss and C.A. Kulikowski. *Computer Systems that Learn: Classification and Prediction Methods From Statistics, Neural Nets, Machine Learning and Expert Systems.* Morgan Kaufmann Publishers Inc., San Mateo, California, 1991.

[128] F. Weston and T. Copeland. Managerial finance. pages 473–510. Dryden Press, New York, 1992.

[129] D. White and D.Sofge. *Handbook of Intelligent Control: Neural,Adaptive and Fuzzy Approaches.* Van Nostrand, 1992.

[130] J. Mesirov X. Zhang and D. Waltz. Hybrid system for protein secondary structure prediction. *Journal of Moleculer Biology*, 225:1049–1063, 1992.

[131] T. Yamaguchi and Y. Tachibana. A technical analysis expert system with knowledge refinement mechanism. pages 86–91, 1991.

[132] L.A. Zadeh. Fuzzy sets. *Information and Control*, 8, 1965.

[133] L.A. Zadeh. Outline of a new approach to the analysis of complex systems and decision processes. *IEEE Trans. on Systems man and Cybernetics*, SMC-3:28–44, 1973.

[134] L.A. Zadeh. The calculus of fuzzy if/then rules. *AI Expert*, 7(3):22–27, March 1992.

[135] L.A. Zadeh. Fuzzy logic. *IEEE Computer*, pages 83–93, April 1988.

Appendix A
Symbolic Pre-processing Example

A.1 Cluster Analysis Example

The following contains an example of clustering using Stolcke's [116] Single Linkage Algorithm and the Cluster Selection algorithm detailed in Chapter 5.

The example data set is British Pound volume data:

```
[1312 1207 234 415 494 367 222 193 332 314 488 1254 1949 826 421 626 824 466 383
1051 1052 968 455 1128 1267 545 1290 1013 842 1331 522 1157 1203 1550 1698 1949 2561
2903 2259 1515 2813 2338 1604 3905 4225 2691 869 860 1205]
```

Vectors in the output are identified by their input sequence number (Eg. in the above example 1312 is 0, 1207 is 1 etc). The following contains an output all pairs of clusters formed, along with their respective inter-cluster distances. Clusters are given as lists of vectors.

```
    minimum distance = 0.000000 ( 35 ) ( 12 )
minimum distance = 1.000000 ( 20 ) ( 19 )
minimum distance = 2.000000 ( 16 ) ( 13 )
minimum distance = 2.000000 ( 48 ) ( 1 )
minimum distance = 3.000000 ( 1 48 ) ( 32 )
minimum distance = 6.000000 ( 10 ) ( 4 )
minimum distance = 6.000000 ( 14 ) ( 3 )
minimum distance = 9.000000 ( 47 ) ( 46 )
minimum distance = 11.000000 ( 22 ) ( 17 )
minimum distance = 12.000000 ( 6 ) ( 2 )
minimum distance = 13.000000 ( 24 ) ( 11 )
minimum distance = 16.000000 ( 18 ) ( 5 )
minimum distance = 17.000000 ( 28 ) ( 13 16 )
minimum distance = 18.000000 ( 9 ) ( 8 )
minimum distance = 19.000000 ( 29 ) ( 0 )
minimum distance = 23.000000 ( 30 ) ( 25 )
minimum distance = 29.000000 ( 31 ) ( 23 )

minimum distance = 29.500000 ( 26 ) ( 11 24 )

minimum distance = 30.500000 ( 17 22 ) ( 4 10 )

minimum distance = 33.833332 ( 46 47 ) ( 13 16 28 )

minimum distance = 35.000000 ( 7 ) ( 2 6 )

minimum distance = 35.000000 ( 39 ) ( 33 )

minimum distance = 38.500000 ( 27 ) ( 19 20 )

minimum distance = 43.000000 ( 5 18 ) ( 3 14 )

minimum distance = 51.166668 ( 0 29 ) ( 11 24 26 )

minimum distance = 57.750000 ( 25 30 ) ( 4 10 17 22 )

minimum distance = 62.500000 ( 32 1 48 ) ( 23 31 )

minimum distance = 70.666664 ( 19 20 27 ) ( 21 )

minimum distance = 71.500000 ( 42 ) ( 33 39 )

minimum distance = 73.500000 ( 3 14 5 18 ) ( 8 9 )

minimum distance = 79.000000 ( 41 ) ( 38 )

minimum distance = 90.000000 ( 40 ) ( 37 )

minimum distance = 110.800003 ( 23 31 32 1 48 ) ( 11 24 26 0 29 )

minimum distance = 123.000000 ( 4 10 17 22 25 30 ) ( 8 9 3 14 5 18 )

minimum distance = 130.000000 ( 45 ) ( 36 )

minimum distance = 141.666672 ( 33 39 42 ) ( 34 )

minimum distance = 176.800003 ( 13 16 28 46 47 ) ( 21 19 20 27 )

minimum distance = 192.500000 ( 8 9 3 14 5 18 4 10 17 22 25 30 ) ( 15 )

minimum distance = 231.974365 ( 15 8 9 3 14 5 18 4 10 17 22 25 30 ) ( 2 6 7 )

minimum distance = 232.000000 ( 36 45 ) ( 37 40 )

minimum distance = 312.622223 ( 11 24 26 0 29 23 31 32 1 48 ) ( 21 19 20 27 13 16 28 46 47 )

minimum distance = 320.000000 ( 44 ) ( 43 )

minimum distance = 349.500000 ( 38 41 ) ( 12 35 )

minimum distance = 504.434204 ( 21 19 20 27 13 16 28 46 47 11 24 26 0 29 23 31 32 1 48 ) ( 34 33 39 42
)

minimum distance = 618.250000 ( 37 40 36 45 ) ( 12 35 38 41 )

minimum distance = 770.230957 ( 34 33 39 42 21 19 20 27 13 16 28 46 47 11 24 26 0 29 23 31 32 1 48 ) (
2 6 7 15 8 9 3 14 5 18 4 10 17 22 25 30 )

minimum distance = 1573.823730 ( 2 6 7 15 8 9 3 14 5 18 4 10 17 22 25 30 34 33 39 42 21 19 20 27 13 16
28 46 47 11 24 26 0 29 23 31 32 1 48 ) ( 12 35 38 41 37 40 36 45 )

minimum distance = 2938.063721 ( 12 35 38 41 37 40 36 45 2 6 7 15 8 9 3 14 5 18 4 10 17 22 25 30 34 33
39 42 21 19 20 27 13 16 28 46 47 11 24 26 0 29 23 31 32 1 48 ) ( 43 44 )
```

The following represents the hierarchical clusters as a tree lying on its side. The leaves of the tree are formed by vector names, and the horizontal spacing between nodes is proportional to the distances between clusters. The linguistic terms HIGH, MEDIUM and LOW appear beside the clusters that were chosen by the cluster selection algorithm.

```
                                             |---|-> 12
                                  |------|   |-> 35         H
                                  |      |---|-> 38         I
                    |-------------|          |-> 41         G
                    |             |       --|-> 37          H
                    |             |------|  |-> 40
                    |                    |--|--> 36
                    |                       |--> 45
                    |
                    |                          -|-> 2
                    |                      |--| |-> 6
                    |                      |  |-> 7
                    |           |-------|  |--> 15              M
|-------------------|           |       |  |      -|-> 8        E
                    |           |       |  |     | |-> 9        D
                    |           |       |--|  |--|  -|-> 3      I
                    |           |          |  |  |-| |-> 14     U
                    |           |          |  |    |-|-> 5      M
                    |           |          |--|      |-> 18
                    |           |             |     -|-> 4
                    |           |             |  |-| |-> 10
                    |           |             |  | |-|-> 17
                    |-----------|             |--|   |-> 22
                                |                |-|-> 25
                                |                  |-> 30
                                |
                                |             |--> 34
                                |       |-----|   -|-> 33
                                |       |     |--| |-> 39
                                |       |        |-> 42
                                |       |            |-> 21
                                |       |         |--|  -|-> 19
                                |-------|         |  |-| |-> 20
                                        |     |---|    |-> 27
                                        |     |   |     -|-> 13
                                        |     |   |  |-| |-> 16
                                        |     |   |--| |-> 28
                                        |     |      |-|-> 46       L
                                        |-----|                     O
                                              |                     W
                                              |---------|-26
                                                        |   | |-|-> 0
                                                        |---|   |-> 29
                                                            |  -|-> 23
                                                            |-| |-> 31
                                                              |-|-> 32
                                                                |-|-> 1
                                                                  |->48
```

Appendix B
Trading Indicators and Expert and Fuzzy Rules

B.1 Trading Indicators Reference Table

The following contains the variable reference table used in INTENT. This is represented as a list of lists where each list contains the following information: type of variable (S=symbolic, F=fuzzy), name of variable (e.g. `ma-diff-1-5-classified-value`), and the permissible states (e.g. `negative`, `neutral positive`). This table is referred to as the *feature index*.

```
[ [S ma-diff-1-5-classified-value [negative neutral positive] ]

/* the difference between the 1 and 5 day price moving average */

[S ma-diff-1-10-classified-value [negative neutral positive] ]

/* the difference between the 1 and 10 day price moving average */

[S ma-diff-1-20-classified-value [negative neutral positive] ]

/* the difference between the 1 and 20 day price moving average */

[S ma-diff-1-50-classified-value [negative neutral positive] ]

/* the difference between the 1 and 50 day price moving average */

[S ma-diff-1-100-classified-value [negative neutral positive] ]

/* the difference between the 1 and 100 day price moving average */

[S ma-diff-1-200-classified-value [negative neutral positive] ]

/* the difference between the 1 and 200 day price moving average */

[F ma-diff-1-5-fuzzy-values [negative neutral positive] ]

/* the difference between the 1 and 5 day price moving average (fuzzy) */

[F ma-diff-1-10-fuzzy-values [negative neutral positive] ]

/* the difference between the 1 and 10 day price moving average (fuzzy) */

[F ma-diff-1-20-fuzzy-values [negative neutral positive] ]

/* the difference between the 1 and 20 day price moving average (fuzzy) */
```

```
[F ma-diff-1-50-fuzzy-values [negative neutral positive] ]

/* the difference between the 1 and 50 day price moving average (fuzzy) */

[F ma-diff-1-100-fuzzy-values [negative neutral positive] ]

/* the difference between the 1 and 100 day price moving average (fuzzy) */

[F ma-diff-1-200-fuzzy-values [negative neutral positive] ]

/* the difference between the 1 and 200 day price moving average (fuzzy) */

[S vol-ma-diff-1-10-classified-value [negative neutral positive] ]

/* the difference between the 1 and 10 day volume moving average */

[S vol-ma-diff-1-20-classified-value [negative neutral positive] ]

/* the difference between the 1 and 20 day volume moving average */

[F vol-ma-diff-1-10-fuzzy-values [negative neutral positive] ]

/* the difference between the 1 and 10 day volume moving average (fuzzy) */

[F vol-ma-diff-1-20-fuzzy-values [negative neutral positive] ]

/* the difference between the 1 and 20 day volume moving average */

[S oi-ma-diff-1-10-classified-value [negative neutral positive] ]

/* the difference between the 1 and 10 day open interest moving average */

[S oi-ma-diff-1-20-classified-value [negative neutral positive] ]

/* the difference between the 1 and 20 day open interest moving average */

[F oi-ma-diff-1-10-fuzzy-values [negative neutral positive] ]

/* the difference between the 1 and 10 day open interest moving average (fuzzy) */

[F oi-ma-diff-1-20-fuzzy-values [negative neutral positive] ]

/* the difference between the 1 and 20 day open interest moving average (fuzzy) */

[S ma-large-1-20 [yes no] ]

/* is the 1 day price moving average larger than the 20 day moving average */

[S ma-large-1-50 [yes no] ]

/* is the 1 day price moving average larger than the 50 day moving average */

[S ma-large-1-100 [yes no] ]

/* is the 1 day price moving average larger than the 100 day moving average */

[S ma-large-1-200 [yes no] ]

/* is the 1 day price moving average larger than 200 day moving average */

[S ma-cb-1-20 [yes no] ]

/* is the 20 day price moving average cut from below by the 1 day moving average*/
```

```
[S ma-cb-1-50 [yes no] ]
/* is the 50 day price moving average cut from below by the 1 day moving average */
[S ma-cb-1-100 [yes no] ]
/* is the 100 day moving average cut from below by the 1 day moving average */
[S ma-cb-1-200 [yes no] ]
/* is the 200 day price moving average cut from below by the a day moving average */
[S ma-ca-1-20 [yes no] ]
/* is the 20 day price moving average cut from above by the 1 day moving average */
[S ma-ca-1-50 [yes no] ]
/* is the 50 day price moving average cut from above by the 1 day moving average */
[S ma-ca-1-100 [yes no] ]
/* is the 100 day price moving average cut from above by the 1 day moving average */
[S ma-ca-1-200 [yes no] ]
/* is the 200 day price moving average cut from above by the 1 day moving average */
[S tb-50 [up down no] ]
/* is there a 50 day trading break */
[S tb-100 [up down no] ]
/* is there a 100 day trading break */
[S tb-200 [up down no] ]
/* is there a 200 day trading break */
[S price-rsi-7 [low medium high] ]
/* the price 7 day relative strength index */
[S price-rsi-14 [low medium high] ]
/* the 14 day price relative strength index */
[S price-rsi-28 [low medium high] ]
/* the 28 day price relative strength index */
[F price-rsi-7-fuzzy [low medium high] ]
/* the 7 day price relative strength index (fuzzy) */
[F price-rsi-14-fuzzy [low medium high] ]
/* the 14 day price relative strength index (fuzzy) */
[F price-rsi-28-fuzzy [low medium high] ]
/* the 28 day price relative strength index (fuzzy) */
```

```
[S vol-rsi-7 [low medium high] ]

/* the 7 day volume relative strength index */

[S vol-rsi-14 [low medium high] ]

/* the 14 day volume relative strength index */

[S vol-rsi-28 [low medium high] ]

/* the 28 day volume relative strength index */

[F vol-rsi-7-fuzzy [low medium high] ]

/* the 7 day volume relative strength index (fuzzy) */

[F vol-rsi-14-fuzzy [low medium high] ]

/* the 14 day volume relative strength index (fuzzy) */

[F vol-rsi-28-fuzzy [low medium high] ]

/* the 28 day volume relative strength index (fuzzy) */

[S oi-rsi-7 [low medium high] ]

/* the 7 day open interest relative strength index */

[S oi-rsi-14 [low medium high] ]

/* the 14 day open interest relative strength index */

[S oi-rsi-28 [low medium high] ]

/* the 28 day open interest relative strength index */

[F oi-rsi-7-fuzzy [low medium high] ]

/* the 7 day open interest relative strength index (fuzzy) */

[F oi-rsi-14-fuzzy [low medium high] ]

/* the 14 day open interest relative strength index (fuzzy) */

[F oi-rsi-28-fuzzy [low medium high] ]

/* the 28 day open interest relative strength index (fuzzy) */

[S price-vola-10 [low medium high] ]

/* the 10 day price volatility */

[S price-vola-20 [low medium high] ]

/* the 20 day price volatility */

[S price-vola-50 [low medium high] ]

/* the 50 day price volatility */

[S price-vola-100 [low medium high] ]

/* the 100 day price volatility */
```

```
[F price-fuzzy-vola-10 [low medium high] ]

/* the 10 day price volatility (fuzzy) */

[F price-fuzzy-vola-20 [low medium high] ]

/* the 20 day price volatility (fuzzy) */

[F price-fuzzy-vola-50 [low medium high] ]

/* the 50 day price volatility (fuzzy) */

[F price-fuzzy-vola-100 [low medium high] ]

/* the 100 day price volatility (fuzzy) */

[S price-after-1-day [UP DOWN] ]

[S price-after-2-day [UP DOWN] ]

[S price-after-5-day [UP DOWN] ]

[S price-after-10-day [UP DOWN] ]

] → feature-list;
```

The following list is the `feature-struct` which selects a given set of variables from the `feature-list` for the INTENT simulations.

```
[ma-ca-1-100 ma-ca-1-200 tb-50 tb-100 tb-200 price-rsi-7 price-rsi-14 price-rsi-28
vol-rsi-7 vol-rsi-14 vol-rsi-28 oi-rsi-7 oi-rsi-14 oi-rsi-28 price-vola-10 price-vola-20
price-vola-50 price-vola-100 price-after-10-day]→ feature-struct;
```

B.2 Expert System Rules

```
[ E1 [ma-diff-1-20-classified-value [positive] ] [oi-ma-diff-1-20-classified-value
[positive] ] [action [UP]] ]

[ E2 [ma-diff-1-20-classified-value [positive] ] [oi-ma-diff-1-20-classified-value
[negative] ] [action [DOWN]] ]

[ E3 [ma-diff-1-20-classified-value [negative] ] [oi-ma-diff-1-20-classified-value
[positive] ] [action [DOWN]] ]

[ E4 [ma-diff-1-20-classified-value [negative] ] [oi-ma-diff-1-20-classified-value
[positive] ] [action [UP]] ]

[ E5 [price-rsi-14 [high] ] [oi-rsi-14 [high] ] [action [UP]] ]

[ E6 [price-rsi-14 [high] ] [oi-rsi-14 [low] ] [action [DOWN]] ]
```

[E7 [price-rsi-14 [low]] [oi-rsi-14 [high]] [action [DOWN]]]

[E8 [price-rsi-14 [low]] [oi-rsi-14 [low]] [action [UP]]]

[E9 [ma-diff-1-20-classified-value [positive]] [vol-ma-diff-1-20-classified-value [positive]] [oi-ma-diff-1-20-classified-value [positive]] [action [UP]]]

[E10 [ma-diff-1-20-classified-value [negative]] [vol-ma-diff-1-20-classified-value [negative]] [oi-ma-diff-1-20-classified-value [negative]] [action [DOWN]]]

[E11 [price-rsi-14 [high]] [vol-rsi-14 [high]] [oi-rsi-14 [high]] [action [UP]]]

[E12 [price-rsi-14 [low]] [vol-rsi-14 [low]] [oi-rsi-14 [low]] [action [DOWN]]]]

[E13 [tb-100 [up]] [action [UP]]]

[E14 [tb-100 [down]] [action [DOWN]]]

[E15 [tb-200 [up]] [action [UP]]]

[E16 [tb-200 [down]] [action [DOWN]]]

[E17 [ma-large-1-100 [yes]] [action [UP]]]

[E18 [tb-100 [no]] [action [DOWN]]]

[E19 [ma-large-1-200 [yes]] [action [UP]]]

[E20 [ma-large-1-200 [no]] [action [DOWN]]]

[E21 [ma-cb-1-100 [yes]] [action [UP]]]

[E22 [ma-cb-1-200 [yes]] [action [UP]]]

[E23 [ma-ca-1-100 [yes]] [action [DOWN]]]

[E24 [ma-ca-1-200 [yes]] [action [DOWN]]]

[E25 [ma-diff-1-100-classified-value [positive]] [action [UP]]]

[E26 [ma-diff-1-100-classified-value [negative]] [action [DOWN]]]

[E27 [ma-diff-1-200-classified-value [positive]] [action [UP]]]

[E28 [ma-diff-1-200-classified-value [negative]] [action [DOWN]]]

[E29 [price-rsi-14 [high]] [action [DOWN]]]

[E30 [price-rsi-14 [low]] [action [UP]]]

[E31 [price-rsi-28 [high]] [action [DOWN]]]

[E32 [price-rsi-28 [low]] [action [UP]]]

[E33 [ma-diff-1-100-classified-value [positive]] [price-vola-50 [low]] [action [UP]]]

[E34 [ma-diff-1-100-classified-value [negative]] [price-vola-50 [low]] [action [DOWN]]]

[E35 [ma-diff-1-200-classified-value [positive]] [price-vola-100 [low]] [action [UP]]]

[E36 [ma-diff-1-200-classified-value [negative]] [price-vola-100 [low]] [action [DOWN]]]

[E37 [price-rsi-14 [high]] [price-vola-50 [high]] [action [DOWN]]]

[E38 [price-rsi-14 [low]] [price-vola-50 [high]] [action [UP]]]

[E39 [price-rsi-28 [high]] [price-vola-100 [high]] [action [DOWN]]]

[E40 [price-rsi-28 [low]] [price-vola-100 [high]] [action [UP]]]

B.2.1 British Pound Selected Rules

[[E1 [ma-diff-1-20-classified-value [positive]] [vol-ma-diff-1-20-classified-value []] [oi-ma-diff-1-20-classified-value [positive]] [action [UP]]]

[E10 [ma-diff-1-20-classified-value [negative]] [vol-ma-diff-1-20-classified-value [negative]] [oi-ma-diff-1-20-classified-value [negative]] [action [DOWN]]]

[E19 [ma-large-1-100 []] [ma-large-1-200 [yes]] [ma-cb-1-200 []] [action [UP]]]

[E20 [ma-large-1-100 []] [ma-large-1-200 [no]] [ma-cb-1-200 []] [action [DOWN]]]

[E22 [ma-large-1-100 []] [ma-large-1-200 []] [ma-cb-1-200 [yes]] [action [UP]]]

[E25 [ma-diff-1-100-classified-value [positive]] [ma-diff-1-200-classified-value []] [action [UP]]]

[E38 [price-rsi-14 [low]] [price-vola-50 [high]] [action [UP]]]

B.2.2 Deutschmark Selected Rules

[[E1 [ma-diff-1-20-classified-value [positive]] [oi-ma-diff-1-20-classified-value [positive]] [action [UP]]]

[E9 [ma-diff-1-20-classified-value [positive]] [vol-ma-diff-1-20-classified-value [positive]] [oi-ma-diff-1-20-classified-value [positive]] [action [UP]]]

[E15 [tb-200 [up]] [action [UP]]]

[E17 [ma-large-1-100 [yes]] [action [UP]]]

[E19 [ma-large-1-200 [yes]] [action [UP]]]

[E25 [ma-diff-1-100-classified-value [positive]] [action [UP]]]

[E27 [ma-diff-1-200-classified-value [positive]] [action [UP]]]

[E38 [price-rsi-14 [low]] [price-vola-50 [high]] [action [UP]]]

B.3 Fuzzy System Rules

[F1 [ma-diff-1-20-fuzzy-values [positive]] [oi-ma-diff-1-20-fuzzy-values [positive]] [action [UP]]]

[F2 [ma-diff-1-20-fuzzy-values [positive]] [oi-ma-diff-1-20-fuzzy-values [negative]] [action [DOWN]]]

[F3 [ma-diff-1-20-fuzzy-values [negative]] [oi-ma-diff-1-20-fuzzy-values [positive]] [action [DOWN]]]

[F4 [ma-diff-1-20-fuzzy-values [negative]] [oi-ma-diff-1-20-fuzzy-values [negative]] [action [UP]]]

[F5 [price-rsi-14-fuzzy [high]] [oi-rsi-14-fuzzy [high]] [action [UP]]]

[F6 [price-rsi-14-fuzzy [high]] [oi-rsi-14-fuzzy [low]] [action [DOWN]]]

[F7 [price-rsi-14-fuzzy [low]] [oi-rsi-14-fuzzy [high]] [action [DOWN]]]

[F8 [price-rsi-14-fuzzy [low]] [oi-rsi-14-fuzzy [low]] [action [UP]]]

[F9 [ma-diff-1-20-fuzzy-values [positive]] [oi-ma-diff-1-20-fuzzy-values [positive]] [vol-ma-diff-1-20-fuzzy-values [positive]] [action [UP]]]

[F10 [ma-diff-1-20-fuzzy-values [negative]] [oi-ma-diff-1-20-fuzzy-values [negative]] [vol-ma-diff-1-20-fuzzy-values [negative]] [action [DOWN]]]

[F11 [price-rsi-14-fuzzy [high]] [oi-rsi-14-fuzzy [high]] [vol-rsi-14-fuzzy [high]] [action [UP]]]

[F12 [price-rsi-14-fuzzy [low]] [oi-rsi-14-fuzzy [low]] [vol-rsi-14-fuzzy [low]] [action [DOWN]]]

[F13 [ma-diff-1-100-fuzzy-values [positive]] [action [UP]]]

[F14 [ma-diff-1-100-fuzzy-values [negative]] [action [DOWN]]]

[F15 [ma-diff-1-200-fuzzy-values [positive]] [action [UP]]]

[F16 [ma-diff-1-200-fuzzy-values [negative]] [action [DOWN]]]

[F17 [price-rsi-14-fuzzy [high]] [action [DOWN]]]

[F18 [price-rsi-14-fuzzy [low]] [action [UP]]]

[F19 [price-rsi-28-fuzzy [high]] [action [DOWN]]]

[F20 [price-rsi-28-fuzzy [low]] [action [UP]]]

[F21 [price-fuzzy-vola-50 [low]] [ma-diff-1-100-fuzzy-values [positive]] [action [UP]]]

[F22 [price-fuzzy-vola-50 [low]] [ma-diff-1-100-fuzzy-values [negative]] [action [DOWN]]]

[F23 [price-fuzzy-vola-100 [low]] [ma-diff-1-200-fuzzy-values [positive]] [action [UP]]]

[F24 [price-fuzzy-vola-100 [low]] [ma-diff-fuzzy-1-200-fuzzy-values [negative]] [action [DOWN]]]

[F25 [price-fuzzy-vola-50 [high]] [price-rsi-14-fuzzy [high]] [action [DOWN]]]

[F26 [price-fuzzy-vola-50 [high]] [price-rsi-14-fuzzy [low]] [action [UP]]]

[F27 [price-fuzzy-vola-100 [high]] [price-rsi-28-fuzzy [high]] [action [DOWN]]]

[F28 [price-fuzzy-vola-100 [high]] [price-rsi-28-fuzzy [low]] [action [UP]]]

B.3.1 British Pound Selected Rules

[[F1 [ma-diff-1-20-fuzzy-values [positive]] [oi-ma-diff-1-20-fuzzy-values [positive]] [action [UP]]]

[F5 [price-rsi-14-fuzzy [high]] [oi-rsi-14-fuzzy [high]] [action [UP]]]

[F6 [price-rsi-14-fuzzy [high]] [oi-rsi-14-fuzzy [low]] [action [DOWN]]]

[F10 [ma-diff-1-20-fuzzy-values [negative]] [oi-ma-diff-1-20-fuzzy-values [negative]] [vol-ma-diff-1-20-fuzzy-values [negative]] [action [DOWN]]]

[F12 [price-rsi-14-fuzzy [low]] [oi-rsi-14-fuzzy [low]] [vol-rsi-14-fuzzy [low]] [action [DOWN]]]

B.3.2 Deutschmark Selected Rules

[[F9 [ma-diff-1-20-fuzzy-values [positive]] [oi-ma-diff-1-20-fuzzy-values [positive]] [vol-ma-diff-1-20-fuzzy-values [positive]] [action [UP]]]

[F13 [ma-diff-1-100-fuzzy-values [positive]] [action [UP]]]

[F15 [ma-diff-1-200-fuzzy-values [positive]] [action [UP]]]

[F26 [price-fuzzy-vola-50 [high]] [price-rsi-14-fuzzy [low]] [action [UP]]]

Appendix C
Simulation Results - Expert System

C.1 The Expert System Results

	E1	E2	E3	E4	E5
Trading-Period-Years	3.02	3.02	3.02	3.02	–
Total-Closed-Trades	22.0	7.0	23.0	23.0	–
Total-Proft-Trades	14.0	2.0	13.0	10.0	–
Total-Losing-Trades	8.0	5.0	10.0	13.0	–
Profitable-Trades-pcn	63.64	28.57	56.52	43.47	–
Total-Gains-$	9375.84	-3426.09	-5386.38	-149.594	–
Initial-Capital-$	100000.0	100000.0	100000.0	100000.0	–
Capital-After-Trading-$	109376.0	96573.9	94613.6	99850.4	–
Av-Gains-per-Trade-$	426.17	-489.44	-234.19	-6.50	–
Max-Proft-Trade-$	11825.6	2896.6	4477.56	4347.5	–
Max-Losing-Trade-$	-4448.69	-4457.52	-4877.59	-4774.55	–
Av-Proft-Trade-$	2258.94	2075.19	1198.69	1864.2	–
Av-Losing-Trade-$	-2781.16	-1515.3	-2096.94	-1445.51	–
Maximum-Drawdown-$	-5227.76	-5664.85	-7423.39	-6721.58	–
Buy-Signals-pcn	100.0	0.0	0.0	100.0	–
Profit-Buy-Trades	0.63	0	0	0.43	–
Profit-Sell-Trades	0	0.28	0.56	0	–
Return-per-year-pcn	3.00	-1.14	-1.81	-0.04	–
Std-Dev-of-PL	3510.4	2122.19	2135.29	2148.07	–

Table C.1: Expert Rules 1-5 Trading the British Pound (84 - 87)

	E6	E7	E8	E9	E10
Trading-Period-Years	–	–	3.02	3.02	3.02
Total-Closed-Trades	–	–	3.0	13.0	8.0
Total-Proft-Trades	–	–	1.0	6.0	6.0
Total-Losing-Trades	–	–	2.0	7.0	2.0
Profitable-Trades-pcn	–	–	33.33	46.15	75.0
Total-Gains-$	–	–	-3425.31	4488.88	3470.25
Initial-Capital-$	–	–	100000.0	100000.0	100000.0
Capital-After-Trading-$	–	–	96574.7	104489.0	103470.0
Av-Gains-per-Trade-$	–	–	-1141.77	345.29	433.78
Max-Proft-Trade-$	–	–	577.33	13128.8	3378.68
Max-Losing-Trade-$	–	–	-3216.85	-3629.12	-4996.37
Av-Proft-Trade-$	–	–	577.33	3974.06	1691.36
Av-Losing-Trade-$	–	–	-2001.32	-2765.06	-3338.92
Maximum-Drawdown-$	–	–	-4002.65	-6608.04	-6677.85
Buy-Signals-pcn	–	–	100.0	100.0	0.0
Profit-Buy-Trades	–	–	0.34	0.46	0
Profit-Sell-Trades	–	–	0	0	0.75
Return-per-year-pcn	–	–	-1.14	1.46	1.13
Std-Dev-of-PL	–	–	1569.29	4472.23	2457.57

Table C.2: Expert Rules 6-10 Trading the British Pound (84-87)

	E11	E12	E13	E14	E15
Trading-Period-Years	–	–	3.02	3.02	3.02
Total-Closed-Trades	–	–	8.0	17.0	7.0
Total-Proft-Trades	–	–	3.0	9.0	3.0
Total-Losing-Trades	–	–	5.0	8.0	4.0
Profitable-Trades-pcn	–	–	37.5	52.94	42.85
Total-Gains-$	–	–	-4936.13	-944.031	-1199.59
Initial-Capital-$	–	–	100000.0	100000.0	100000.0
Capital-After-Trading-$	–	–	95063.9	99056.0	98800.4
Av-Gains-per-Trade-$	–	–	-617.01	-55.53	-171.37
Max-Proft-Trade-$	–	–	3589.74	4645.56	3730.84
Max-Losing-Trade-$	–	–	-3781.94	-9323.52	-3436.41
Av-Proft-Trade-$	–	–	2615.99	1832.51	2718.82
Av-Losing-Trade-$	–	–	-2556.83	-2179.58	-2339.01
Maximum-Drawdown-$	–	–	-9002.21	-10315.8	-9356.05
Buy-Signals-pcn	–	–	100.0	0.0	100.0
Profit-Buy-Trades	–	–	0.37	0	0.42
Profit-Sell-Trades	–	–	0	0.52	0
Return-per-year-pcn	–	–	-1.66	-0.31	-0.39
Std-Dev-of-PL	–	–	2679.77	2979.38	2664.28

Table C.3: Expert Rules 11-15 Trading the British Pound (84-87)

	E16	E17	E18	E19	E20
Trading-Period-Years	3.02	3.02	3.02	3.02	3.02
Total-Closed-Trades	13.0	34.0	68.0	35.0	40.0
Total-Proft-Trades	7.0	18.0	38.0	20.0	24.0
Total-Losing-Trades	6.0	16.0	30.0	15.0	16.0
Profitable-Trades-pcn	53.84	52.94	55.88	57.14	60.0
Total-Gains-$	6026.06	19482.2	-10012.0	7439.13	9887.53
Initial-Capital-$	100000.0	100000.0	100000.0	100000.0	100000.0
Capital-After-Trading-$	106026.0	119482.0	89988.0	107439.0	109888.0
Av-Gains-per-Trade-$	463.54	573.00	-147.23	212.55	247.18
Max-Proft-Trade-$	4645.56	8959.21	4317.43	8752.3	5749.46
Max-Losing-Trade-$	-2216.05	-7158.8	-10162.2	-4569.21	-9893.48
Av-Proft-Trade-$	1896.75	3260.77	1743.45	2120.3	1853.84
Av-Losing-Trade-$	-1208.54	-2450.73	-2542.1	-2331.12	-2162.79
Maximum-Drawdown-$	-3617.15	-7158.8	-26258.0	-5268.32	-13359.7
Buy-Signals-pcn	0.0	100.0	0.0	100.0	0.0
Profit-Buy-Trades	0	0.52	0	0.57	0
Profit-Sell-Trades	0.54	0	0.56	0	0.6
Return-per-year-pcn	1.95	6.06	-3.43	2.40	3.17
Std-Dev-of-PL	1989.29	3640.97	2863.96	2848.02	2601.46

Table C.4: Expert Rules 16-20 Trading the British Pound (84-87)

	E21	E22	E23	E24	E25
Trading-Period-Years	3.02	3.02	3.02	3.02	3.02
Total-Closed-Trades	6.0	6.0	5.0	6.0	44.0
Total-Proft-Trades	3.0	4.0	2.0	2.0	23.0
Total-Losing-Trades	3.0	2.0	3.0	4.0	21.0
Profitable-Trades-pcn	50.0	66.67	40.0	33.34	52.27
Total-Gains-$	3316.0	4741.75	-11388.3	-5152.56	20868.7
Initial-Capital-$	100000.0	100000.0	100000.0	100000.0	100000.0
Capital-After-Trading-$	103316.0	104742.0	88611.8	94847.4	120869.0
Av-Gains-per-Trade-$	552.67	790.29	-2277.65	-858.76	474.29
Max-Proft-Trade-$	4132.22	4027.85	618.309	3170.21	11418.7
Max-Losing-Trade-$	-1915.72	-2584.65	-8700.88	-5644.5	-6823.77
Av-Proft-Trade-$	2581.22	2472.32	450.392	2565.5	2989.24
Av-Losing-Trade-$	-1475.89	-2573.77	-4096.34	-2570.9	-2280.18
Maximum-Drawdown-$	-2761.5	-2584.65	-12289.0	-10283.6	-10498.2
Buy-Signals-pcn	100.0	100.0	0.0	0.0	100.0
Profit-Buy-Trades	0.5	0.67	0	0	0.52
Profit-Sell-Trades	0	0	0.4	0.34	0
Return-per-year-pcn	1.08	1.54	-3.91	-1.73	6.47
Std-Dev-of-PL	2438.79	2513.22	3386.65	2845.02	3558.53

Table C.5: Expert Rules 21-25 Trading the British Pound (84-87)

	E26	E27	E28	E29	E30
Trading-Period-Years	3.02	3.02	3.02	–	3.02
Total-Closed-Trades	29.0	48.0	16.0	–	20.0
Total-Proft-Trades	15.0	26.0	9.0	–	10.0
Total-Losing-Trades	14.0	22.0	7.0	–	10.0
Profitable-Trades-pcn	51.73	54.16	56.25	–	50.0
Total-Gains-$	-6207.13	22433.1	833.25	–	3091.56
Initial-Capital-$	100000.0	100000.0	100000.0	–	100000.0
Capital-After-Trading-$	93792.9	122433.0	100833.0	–	103092.0
Av-Gains-per-Trade-$	-214.04	467.36	52.07	–	154.58
Max-Proft-Trade-$	4315.94	7615.76	4656.35	–	8372.17
Max-Losing-Trade-$	-16208.1	-5002.21	-9010.64	–	-4661.74
Av-Proft-Trade-$	1968.22	2535.56	2401.58	–	2244.91
Av-Losing-Trade-$	-2552.17	-1976.89	-2968.71	–	-1935.75
Maximum-Drawdown-$	-18979.0	-7649.81	-11660.6	–	-6113.97
Buy-Signals-pcn	0.0	100.0	0.0	–	100.0
Profit-Buy-Trades	0	0.54	0	–	0.5
Profit-Sell-Trades	0.52	0	0.56	–	0
Return-per-year-pcn	-2.09	6.92	0.27	–	1.01
Std-Dev-of-PL	3670.02	2896.76	3475.8	–	2845.34

Table C.6: Expert Rules 26-30 Trading the British Pound (84-87)

	E31	E32	E33	E34	E35
Trading-Period-Years	–	3.02	3.02	3.02	–
Total-Closed-Trades	–	6.0	2.0	3.0	–
Total-Proft-Trades	–	0.0	1.0	0.0	–
Total-Losing-Trades	–	6.0	1.0	3.0	–
Profitable-Trades-pcn	–	0.0	50.0	0.0	–
Total-Gains-$	–	-10254.8	2972.34	-2133.38	–
Initial-Capital-$	–	100000.0	100000.0	100000.0	–
Capital-After-Trading-$	–	89745.2	102972.0	97866.6	–
Av-Gains-per-Trade-$	–	-1709.13	1486.17	-711.125	–
Max-Proft-Trade-$	–	0.0	3212.07	0.0	–
Max-Losing-Trade-$	–	-4439.92	-239.762	-1307.55	–
Av-Proft-Trade-$	–	0	3212.07	0	–
Av-Losing-Trade-$	–	-1709.13	-239.762	-711.115	–
Maximum-Drawdown-$	–	-10254.8	-239.762	-2133.35	–
Buy-Signals-pcn	–	100.0	100.0	0.0	–
Profit-Buy-Trades	–	0.0	0.5	0	–
Profit-Sell-Trades	–	0	0	0.0	–
Return-per-year-pcn	–	-3.51	0.97	-0.71	–
Std-Dev-of-PL	–	1375.5	1725.92	465.684	–

Table C.7: Expert Rules 31-35 Trading the British Pound (84-87)

	E36	E37	E38	E39	E40
Trading-Period-Years	–	–	3.02	–	3.02
Total-Closed-Trades	–	–	16.0	–	6.0
Total-Proft-Trades	–	–	9.0	–	0.0
Total-Losing-Trades	–	–	7.0	–	6.0
Profitable-Trades-pcn	–	–	56.25	–	0.0
Total-Gains-$	–	–	6341.81	–	-10254.8
Initial-Capital-$	–	–	100000.0	–	100000.0
Capital-After-Trading-$	–	–	106342.0	–	89745.2
Av-Gains-per-Trade-$	–	–	396.363	–	-1709.13
Max-Proft-Trade-$	–	–	8636.13	–	0.0
Max-Losing-Trade-$	–	–	-4808.72	–	-4439.92
Av-Proft-Trade-$	–	–	2452.12	–	0
Av-Losing-Trade-$	–	–	-2246.74	–	-1709.13
Maximum-Drawdown-$	–	–	-6874.75	–	-10254.8
Buy-Signals-pcn	–	–	100.0	–	100.0
Profit-Buy-Trades	–	–	0.5625	–	0.0
Profit-Sell-Trades	–	–	0	–	0
Return-per-year-pcn	–	–	2.05417	–	-3.51465
Std-Dev-of-PL	–	–	3139.74	–	1375.5

Table C.8: Expert Rules 36-40 Trading the British Pound (84-87)

	E1	E2	E3	E4	E5
Trading-Period-Years	3.02	3.02	3.02	3.02	–
Total-Closed-Trades	28.0	28.0	3.0	3.0	–
Total-Proft-Trades	17.0	12.0	2.0	1.0	–
Total-Losing-Trades	11.0	16.0	1.0	2.0	–
Profitable-Trades-pcn	60.71	42.85	66.67	33.34	–
Total-Gains-$	22920.6	-10473.4	4532.72	-5112.03	–
Initial-Capital-$	100000.0	100000.0	100000.0	100000.0	–
Capital-After-Trading-$	122921.0	89526.6	104533.0	94888.0	–
Av-Gains-per-Trade-$	818.59	-374.05	1510.91	-1704.01	–
Max-Proft-Trade-$	8469.83	7102.81	3061.11	1296.16	–
Max-Losing-Trade-$	-4802.72	-6285.0	-1496.16	-3254.48	–
Av-Proft-Trade-$	2858.08	2229.26	3014.43	1296.16	–
Av-Losing-Trade-$	-2333.35	-2326.54	-1496.16	-3204.1	–
Maximum-Drawdown-$	-7892.81	-21091.6	-1496.16	-6408.2	–
Buy-Signals-pcn	100.0	0.0	0.0	100.0	–
Profit-Buy-Trades	60.71	0	0	33.33	–
Profit-Sell-Trades	0	42.8571	66.67	0	–
Return-per-year-pcn	7.07	-3.59	1.47	-1.72	–
Std-Dev-of-PL	3194.98	2912.27	2126.66	2121.84	–

Table C.9: Expert Rules 1-5 Trading the Deutschmark (84-87)

	E6	E7	E8	E9	E10
Trading-Period-Years	–	–	3.02	3.02	3.02
Total-Closed-Trades	–	–	5.0	22.0	13.0
Total-Proft-Trades	–	–	3.0	15.0	7.0
Total-Losing-Trades	–	–	2.0	7.0	6.0
Profitable-Trades-pcn	–	–	60.0	68.18	53.81
Total-Gains-$	–	–	13417.6	28231.3	-14000.8
Initial-Capital-$	–	–	100000.0	100000.0	100000.0
Capital-After-Trading-$	–	–	113418.0	128231.0	85999.2
Av-Gains-per-Trade-$	–	–	2683.51	1283.24	-1076.99
Max-Proft-Trade-$	–	–	8582.79	8857.42	2550.05
Max-Losing-Trade-$	–	–	-1059.48	-5017.31	-7853.19
Av-Proft-Trade-$	–	–	5018.15	2984.62	1317.03
Av-Losing-Trade-$	–	–	-818.43	-2362.58	-3870.0
Maximum-Drawdown-$	–	–	-1636.88	-8427.08	-20236.3
Buy-Signals-pcn	–	–	100.0	100.0	0.0
Profit-Buy-Trades	–	–	60.0	68.18	0
Profit-Sell-Trades	–	–	0	0	53.85
Return-per-year-pcn	–	–	4.26	8.58	-4.87
Std-Dev-of-PL	–	–	3553.41	3187.24	3034.96

Table C.10: Expert Rules 6-10 Trading the Deutschmark (84-87)

	E11	E12	E13	E14	E15
Trading-Period-Years	–	–	3.02	3.02	3.02
Total-Closed-Trades	–	–	21.0	11.0	19.0
Total-Proft-Trades	–	–	14.0	5.0	13.0
Total-Losing-Trades	–	–	7.0	6.0	6.0
Profitable-Trades-pcn	–	–	66.67	45.45	68.42
Total-Gains-$	–	–	2129.28	3398.69	12027.0
Initial-Capital-$	–	–	100000.0	100000.0	100000.0
Capital-After-Trading-$	–	–	102129.0	103399.0	112027.0
Av-Gains-per-Trade-$	–	–	101.39	308.97	633.00
Max-Proft-Trade-$	–	–	5558.96	4590.8	6097.71
Max-Losing-Trade-$	–	–	-8775.62	-3700.66	-3094.6
Av-Proft-Trade-$	–	–	1677.49	2338.4	1843.41
Av-Losing-Trade-$	–	–	-3050.79	-1382.23	-1989.55
Maximum-Drawdown-$	–	–	-13294.1	-6355.89	-6641.8
Buy-Signals-pcn	–	–	100.0	0.0	100.0
Profit-Buy-Trades	–	–	66.67	0	68.42
Profit-Sell-Trades	–	–	0	45.45	0
Return-per-year-pcn	–	–	0.70	1.11	3.83
Std-Dev-of-PL	–	–	2923.54	2339.31	2253.94

Table C.11: Expert Rules 11-15 Trading the Deutschmark (84-87)

	E16	E17	E18	E19	E20
Trading-Period-Years	3.02	3.02	3.02	3.02	3.02
Total-Closed-Trades	10.0	48.0	66.0	44.0	29.0
Total-Proft-Trades	5.0	30.0	26.0	31.0	14.0
Total-Losing-Trades	5.0	18.0	40.0	13.0	15.0
Profitable-Trades-pcn	50.0	62.5	39.39	.4546	48.2759
Total-Gains-$	2500.56	41741.0	-45357.3	37607.8	6050.13
Initial-Capital-$	100000.0	100000.0	100000.0	100000.0	100000.0
Capital-After-Trading-$	102501.0	141741.0	54642.7	137608.0	106050.0
Av-Gains-per-Trade-$	250.05	869.60	-687.23	854.72	208.62
Max-Proft-Trade-$	4550.92	8889.45	3621.29	9694.11	4735.74
Max-Losing-Trade-$	-3668.51	-5098.23	-6528.56	-6135.67	-4490.94
Av-Proft-Trade-$	2078.81	2629.29	1399.67	2412.44	2346.02
Av-Losing-Trade-$	-1578.72	-2063.21	-2043.72	-2859.81	-1786.28
Maximum-Drawdown-$	-6300.68	-9504.3	-14991.0	-9773.5	-9529.55
Buy-Signals-pcn	0.0	100.0	0.0	100.0	0.0
Profit-Buy-Trades	0	62.5	0	70.4546	0
Profit-Sell-Trades	50.0	0	39.39	0	48.27
Return-per-year-pcn	0.82	12.24	-18.13	11.15	1.96
Std-Dev-of-PL	2383.98	2944.08	2204.55	3156.57	2453.98

Table C.12: Expert Rules 16-20 Trading the Deutschmark (84-87)

	E21	E22	E23	E24	E25
Trading-Period-Years	3.02	3.02	3.02	3.02	3.02
Total-Closed-Trades	6.0	4.0	5.0	4.0	51.0
Total-Proft-Trades	4.0	1.0	3.0	3.0	34.0
Total-Losing-Trades	2.0	3.0	2.0	1.0	17.0
Profitable-Trades-pcn	66.67	25.0	60.0	75.0	66.67
Total-Gains-$	7777.59	-5407.16	-5938.97	5848.5	39163.9
Initial-Capital-$	100000.0	100000.0	100000.0	100000.0	100000.0
Capital-After-Trading-$	107778.0	94592.8	94061.0	105849.0	139164.0
Av-Gains-per-Trade-$	1296.27	-1351.79	-1187.79	1462.13	767.92
Max-Proft-Trade-$	8621.07	741.22	3071.41	3164.09	9942.47
Max-Losing-Trade-$	-3367.5	-3152.69	-8452.13	-365.83	-6292.87
Av-Proft-Trade-$	3593.53	741.22	2393.5	2071.44	2471.72
Av-Losing-Trade-$	-3298.27	-2049.46	-6559.73	-365.828	-2639.67
Maximum-Drawdown-$	-6596.53	-6148.38	-13119.5	-365.828	-9364.01
Buy-Signals-pcn	100.0	100.0	0.0	0.0	100.0
Profit-Buy-Trades	66.67	25.0	0	0	66.67
Profit-Sell-Trades	0	0	60.0	75.0	0
Return-per-year-pcn	2.51	-1.82	-2.00	1.99	11.56
Std-Dev-of-PL	4078.56	1703.26	4577.7	1684.83	3118.26

Table C.13: Expert Rules 21-25 Trading the Deutschmark (84-87)

	E26	E27	E28	E29	E30
Trading-Period-Years	3.02	3.02	3.02	–	3.02
Total-Closed-Trades	18.0	46.0	23.0	–	13.0
Total-Proft-Trades	11.0	27.0	11.0	–	7.0
Total-Losing-Trades	7.0	19.0	12.0	–	6.0
Profitable-Trades-pcn	61.11	58.69	47.82	–	53.84
Total-Gains-$	2576.28	36481.7	-6743.34	–	3059.16
Initial-Capital-$	100000.0	100000.0	100000.0	–	100000.0
Capital-After-Trading-$	102576.0	136482.0	93256.7	–	103059.0
Av-Gains-per-Trade-$	143.127	793.08	-293.189	–	235.32
Max-Proft-Trade-$	4324.54	9372.75	3672.04	–	8069.13
Max-Losing-Trade-$	-7002.76	-6528.9	-4954.3	–	-6013.4
Av-Proft-Trade-$	1869.63	2767.31	1814.1	–	2743.16
Av-Losing-Trade-$	-2569.95	-2012.4	-2224.86	–	-2690.49
Maximum-Drawdown-$	-7612.18	-10472.1	-10089.8	–	-8511.2
Buy-Signals-pcn	0.0	100.0	0.0	–	100.0
Profit-Buy-Trades	0	58.69	0	–	53.84
Profit-Sell-Trades	61.11	0	47.82	–	0
Return-per-year-pcn	0.84	10.84	-2.28	–	1.00
Std-Dev-of-PL	2827.05	3141.67	2507.62	–	3483.51

Table C.14: Expert Rules 26-30 Trading the Deutschmark (84-87)

	E31	E32	E33	E34	E35
Trading-Period-Years	3.02	–	3.02	3.02	3.02
Total-Closed-Trades	13.0	–	6.0	1.0	3.0
Total-Proft-Trades	7.0	–	3.0	1.0	0.0
Total-Losing-Trades	6.0	–	3.0	0.0	3.0
Profitable-Trades-pcn	53.8461	–	50.0	100.0	0.0
Total-Gains-$	3059.16	–	7428.5	3327.31	-3958.97
Initial-Capital-$	100000.0	–	100000.0	100000.0	100000.0
Capital-After-Trading-$	103059.0	–	107429.0	103327.0	96041.0
Av-Gains-per-Trade-$	235.32	–	1238.08	3327.31	-1319.66
Max-Proft-Trade-$	8069.13	–	8412.04	3327.31	0.0
Max-Losing-Trade-$	-6013.4	–	-1912.6	0.0	-2515.08
Av-Proft-Trade-$	2743.16	–	3926.19	3327.31	0
Av-Losing-Trade-$	-2690.49	–	-1450.03	0	-1319.66
Maximum-Drawdown-$	-8511.2	–	-1912.6	0.0	-3958.99
Buy-Signals-pcn	100.0	–	100.0	100.0	0.0
Profit-Buy-Trades	53.8461	–	50.0	100.0	0
Profit-Sell-Trades	0	–	0	0	0.0
Return-per-year-pcn	1.00	–	2.40	1.08	-1.32
Std-Dev-of-PL	3483.51	–	3639.29	0.0	922.74

Table C.15: Expert Rules 31-35 Trading the Deutschmark (84-87)

	E36	E37	E38	E39	E40
Trading-Period-Years	3.02	–	3.02	–	3.02
Total-Closed-Trades	3.0	–	12.0	–	6.0
Total-Proft-Trades	0.0	–	7.0	–	3.0
Total-Losing-Trades	3.0	–	5.0	–	3.0
Profitable-Trades-pcn	0.0	–	58.33	–	50.0
Total-Gains-$	-5980.81	–	10262.0	–	7428.5
Initial-Capital-$	100000.0	–	100000.0	–	100000.0
Capital-After-Trading-$	94019.2	–	110262.0	–	107429.0
Av-Gains-per-Trade-$	-1993.6	–	855.164	–	1238.08
Max-Proft-Trade-$	0.0	–	8633.09	–	8412.04
Max-Losing-Trade-$	-3197.02	–	-4578.85	–	-1912.6
Av-Proft-Trade-$	0	–	2899.17	–	3926.19
Av-Losing-Trade-$	-1993.61	–	-2006.45	–	-1450.03
Maximum-Drawdown-$	-5980.82	–	-8511.2	–	-1912.6
Buy-Signals-pcn	0.0	–	100.0	–	100.0
Profit-Buy-Trades	0	–	58.33	–	50.0
Profit-Sell-Trades	0.0	–	0	–	0
Return-per-year-pcn	-2.02	–	3.28	–	2.40
Std-Dev-of-PL	1251.04	–	3232.09	–	3639.29

Table C.16: Expert Rules 36-40 Trading the Deutschmark (84-87)

Appendix D
Simulation Results - Fuzzy System

D.1 The Fuzzy System Results

	F1	F2	F3	F4	F5
Trading-Period-Years	3.02	3.02	3.02	3.02	3.02
Total-Closed-Trades	15.0	7.0	23.0	9.0	34.0
Total-Proft-Trades	9.0	3.0	13.0	3.0	19.0
Total-Losing-Trades	6.0	4.0	10.0	6.0	15.0
Profitable-Trades-pcn	60.0	42.85	56.52	33.33	55.88
Total-Gains-$	9485.91	-755.0	-14441.0	-4526.41	15261.8
Initial-Capital-$	100000.0	100000.0	100000.0	100000.0	100000.0
Capital-After-Trading-$	109486.0	99245.0	85559.0	95473.6	115262.0
Av-Gains-per-Trade-$	632.39	-107.85	-627.87	-502.93	448.87
Max-Proft-Trade-$	11616.6	2896.6	4560.35	4029.36	7296.57
Max-Losing-Trade-$	-4370.07	-4457.52	-11500.1	-3168.23	-7533.6
Av-Proft-Trade-$	2719.29	2241.84	1566.72	1732.53	2460.3
Av-Losing-Trade-$	-2497.93	-1870.14	-3480.84	-1620.67	-2098.93
Maximum-Drawdown-$	-6547.28	-5664.85	-20748.7	-9724.04	-14463.6
Buy-Signals-pcn	100.0	0.0	0.0	100.0	100.0
Profit-Buy-Trades	60.0	0	0	33.34	55.89
Profit-Sell-Trades	0	42.86	56.52	0	0
Return-per-year-pcn	3.04	-0.25	-5.02	-1.52	4.80
Std-Dev-of-PL	3812.51	2405.43	3358.91	1968.84	3055.31

Table D.1: Fuzzy Rules 1-5 Trading the British Pound (84-87)

	F6	F7	F8	F9	F10
Trading-Period-Years	3.02	3.02	3.02	3.02	3.02
Total-Closed-Trades	23.0	19.0	8.0	11.0	9.0
Total-Proft-Trades	16.0	10.0	4.0	6.0	7.0
Total-Losing-Trades	7.0	9.0	4.0	5.0	2.0
Profitable-Trades-pcn	69.56	52.63	50.0	54.54	77.77
Total-Gains-$	16488.2	-8769.13	7641.88	-307.656	3625.13
Initial-Capital-$	100000.0	100000.0	100000.0	100000.0	100000.0
Capital-After-Trading-$	116488.0	91230.9	107642.0	99692.3	103625.0
Av-Gains-per-Trade-$	716.87	-461.53	955.23	-27.96	402.79
Max-Proft-Trade-$	7138.08	3659.34	9706.69	3596.76	3383.74
Max-Losing-Trade-$	-5019.95	-8731.18	-4035.96	-3596.56	-5003.84
Av-Proft-Trade-$	2236.73	1901.99	3491.08	2275.09	1473.28
Av-Losing-Trade-$	-2757.07	-3087.67	-1580.61	-2791.64	-3343.92
Maximum-Drawdown-$	-8978.56	-15465.4	-6322.45	-6818.34	-6687.84
Buy-Signals-pcn	0.0	0.0	100.0	100.0	0.0
Profit-Buy-Trades	0	0	50.0	54.54	0
Profit-Sell-Trades	69.56	52.63	0	0	77.78
Return-per-year-pcn	5.17	-2.98	2.46	-0.10	1.18
Std-Dev-of-PL	2870.08	3222.3	3736.78	2746.38	2322.21

Table D.2: Fuzzy Rules 6-10 Trading the British Pound (84-87)

	F11	F12	F13	F14	F15
Trading-Period-Years	3.02	3.02	3.02	3.02	3.02
Total-Closed-Trades	22.0	5.0	42.0	29.0	48.0
Total-Proft-Trades	12.0	4.0	23.0	15.0	26.0
Total-Losing-Trades	10.0	1.0	19.0	14.0	22.0
Profitable-Trades-pcn	54.54	80.0	54.76	51.72	54.17
Total-Gains-$	9473.13	4331.41	20682.8	-6207.13	22433.1
Initial-Capital-$	100000.0	100000.0	100000.0	100000.0	100000.0
Capital-After-Trading-$	109473.0	104331.0	120683.0	93792.9	122433.0
Av-Gains-per-Trade-$	430.59	866.28	492.49	-214.03	467.35
Max-Proft-Trade-$	5785.98	3995.16	11516.1	4315.94	7615.76
Max-Losing-Trade-$	-3283.68	-2785.96	-6881.93	-16208.1	-5002.21
Av-Proft-Trade-$	2070.88	1779.34	2948.65	1968.22	2535.56
Av-Losing-Trade-$	-1537.74	-2785.96	-2480.85	-2552.17	-1976.89
Maximum-Drawdown-$	-8786.78	-2785.96	-9715.55	-18979.0	-7649.81
Buy-Signals-pcn	100.0	0.0	100.0	0.0	100.0
Profit-Buy-Trades	54.55	0	54.76	0	54.16
Profit-Sell-Trades	0	80.0	0	51.72	0
Return-per-year-pcn	3.04	1.41	6.41	-2.09	6.92
Std-Dev-of-PL	2295.52	2196.47	3655.12	3670.02	2896.76

Table D.3: Fuzzy Rules 11-15 Trading the British Pound (84-87)

	F16	F17	F18	F19	F20
Trading-Period-Years	3.02	3.02	3.02	3.02	3.02
Total-Closed-Trades	16.0	50.0	26.0	54.0	17.0
Total-Proft-Trades	9.0	26.0	11.0	27.0	9.0
Total-Losing-Trades	7.0	24.0	15.0	27.0	8.0
Profitable-Trades-pcn	56.25	52.0	42.30	50.0	52.94
Total-Gains-$	833.25	-15200.9	1226.81	-20607.3	5298.28
Initial-Capital-$	100000.0	100000.0	100000.0	100000.0	100000.0
Capital-After-Trading-$	100833.0	84799.1	101227.0	79392.8	105298.0
Av-Gains-per-Trade-$	52.07	-304.01	47.18	-381.61	311.66
Max-Proft-Trade-$	4656.35	6151.03	9161.0	4576.52	8294.59
Max-Losing-Trade-$	-9010.64	-9075.41	-4147.09	-8351.8	-4972.1
Av-Proft-Trade-$	2401.58	1538.08	2878.71	1394.83	2487.09
Av-Losing-Trade-$	-2968.71	-2299.62	-2029.27	-2158.07	-2135.69
Maximum-Drawdown-$	-11660.6	-11928.6	-13942.0	-12781.3	-5381.18
Buy-Signals-pcn	0.0	0.0	100.0	0.0	100.0
Profit-Buy-Trades	0	0	42.30	0	52.94
Profit-Sell-Trades	56.25	52.0	0	50.0	0
Return-per-year-pcn	0.27	-5.30	0.40	-7.34	1.72
Std-Dev-of-PL	3475.8	2655.86	3152.94	2431.91	3348.67

Table D.4: Fuzzy Rules 16-20 Trading the British Pound (84-87)

	F21	F22	F23	F24	F25
Trading-Period-Years	3.02	3.02	–	–	3.02
Total-Closed-Trades	1.0	6.0	–	–	39.0
Total-Proft-Trades	1.0	2.0	–	–	22.0
Total-Losing-Trades	0.0	4.0	–	–	17.0
Profitable-Trades-pcn	100.0	33.33	–	–	56.41
Total-Gains-$	1945.06	-1750.75	–	–	-11623.2
Initial-Capital-$	100000.0	100000.0	–	–	100000.0
Capital-After-Trading-$	101945.0	98249.3	–	–	88376.8
Av-Gains-per-Trade-$	1945.06	-291.792	–	–	-298.03
Max-Proft-Trade-$	1945.06	3242.26	–	–	6267.69
Max-Losing-Trade-$	0.0	-2869.26	–	–	-9247.55
Av-Proft-Trade-$	1945.06	1590.37	–	–	1477.98
Av-Losing-Trade-$	0	-1232.85	–	–	-2596.4
Maximum-Drawdown-$	0.0	-4347.86	–	–	-12154.8
Buy-Signals-pcn	100.0	0.0	–	–	0.0
Profit-Buy-Trades	100.0	0	–	–	0
Profit-Sell-Trades	0	33.33	–	–	56.41
Return-per-year-pcn	0.63	-0.58	–	–	-4.00
Std-Dev-of-PL	0.0	1840.17	–	–	2831.45

Table D.5: Fuzzy Rules 21-25 Trading the British Pound (84-87)

	F26	F27	F28
Trading-Period-Years	3.02	3.02	3.02
Total-Closed-Trades	21.0	52.0	16.0
Total-Proft-Trades	10.0	27.0	7.0
Total-Losing-Trades	11.0	25.0	9.0
Profitable-Trades-pcn	47.619	51.9231	43.75
Total-Gains-$	9697.63	-19899.3	2545.41
Initial-Capital-$	100000.0	100000.0	100000.0
Capital-After-Trading-$	109698.0	80100.8	102545.0
Av-Gains-per-Trade-$	461.79	-382.67	159.08
Max-Proft-Trade-$	9927.6	4617.33	8077.74
Max-Losing-Trade-$	-4494.12	-8426.29	-4842.11
Av-Proft-Trade-$	3373.43	1364.96	2777.75
Av-Losing-Trade-$	-2185.16	-2270.13	-1877.65
Maximum-Drawdown-$	-15108.7	-12895.3	-5240.49
Buy-Signals-pcn	100.0	0.0	100.0
Profit-Buy-Trades	47.619	0	43.75
Profit-Sell-Trades	0	51.92	0
Return-per-year-pcn	3.10	-7.07	0.83
Std-Dev-of-PL	3606.01	2469.08	3337.51

Table D.6: Fuzzy Rules 26-28 Trading the British Pound (84-87)

	F1	F2	F3	F4	F5
Trading-Period-Years	3.02	3.02	3.02	3.02	3.02
Total-Closed-Trades	24.0	28.0	4.0	20.0	37.0
Total-Proft-Trades	13.0	11.0	3.0	10.0	21.0
Total-Losing-Trades	11.0	17.0	1.0	10.0	16.0
Profitable-Trades-pcn	54.16	39.28	75.0	50.0	56.75
Total-Gains-$	11587.1	-22840.8	5813.91	-16988.5	6325.66
Initial-Capital-$	100000.0	100000.0	100000.0	100000.0	100000.0
Capital-After-Trading-$	111587.0	77159.2	105814.0	83011.5	106326.0
Av-Gains-per-Trade-$	482.79	-815.74	1453.48	-849.42	170.96
Max-Proft-Trade-$	7365.62	2913.13	3098.62	2021.79	4975.88
Max-Losing-Trade-$	-4359.9	-5894.88	-1514.5	-4533.58	-5626.13
Av-Proft-Trade-$	2597.79	958.679	2442.8	695.735	2120.74
Av-Losing-Trade-$	-2016.75	-1963.9	-1514.5	-2394.59	-2388.12
Maximum-Drawdown-$	-7165.08	-7347.47	-1514.5	-9692.3	-10954.9
Buy-Signals-pcn	100.0	0.0	0.0	100.0	100.0
Profit-Buy-Trades	54.17	0	0	50.0	56.75
Profit-Sell-Trades	0	39.28	75.0	0	0
Return-per-year-pcn	3.7	-8.2	1.88	-5.97	2.05178
Std-Dev-of-PL	2751.86	1913.42	1868.95	1858.97	2607.21

Table D.7: Fuzzy Rules 1-5 Trading the Deutschmark (84-87)

	F6	F7	F8	F9	F10
Trading-Period-Years	3.02	3.02	3.02	3.02	3.02
Total-Closed-Trades	9.0	3.0	5.0	21.0	12.0
Total-Proft-Trades	3.0	2.0	2.0	12.0	7.0
Total-Losing-Trades	6.0	1.0	3.0	9.0	5.0
Profitable-Trades-pcn	33.34	66.67	40.0	57.14	58.34
Total-Gains-$	-5048.69	2172.56	6074.81	10944.9	-9828.09
Initial-Capital-$	100000.0	100000.0	100000.0	100000.0	100000.0
Capital-After-Trading-$	94951.3	102173.0	106075.0	110945.0	90171.9
Av-Gains-per-Trade-$	-560.96	724.18	1214.96	521.18	-819.00
Max-Proft-Trade-$	3845.95	4566.27	8376.09	5210.4	2052.2
Max-Losing-Trade-$	-4103.36	-3680.88	-2243.06	-4340.95	-8234.23
Av-Proft-Trade-$	2001.63	2926.71	5417.9	2468.14	1290.56
Av-Losing-Trade-$	-1842.27	-3680.88	-1587.0	-2074.75	-3772.4
Maximum-Drawdown-$	-6228.76	-3680.88	-4760.99	-7291.06	-15878.2
Buy-Signals-pcn	0.0	0.0	100.0	100.0	0.0
Profit-Buy-Trades	0	0	40.0	57.1429	0
Profit-Sell-Trades	33.34	66.67	0	0	58.34
Return-per-year-pcn	-1.70	0.71	1.97	3.49	-3.36
Std-Dev-of-PL	2270.56	3390.34	3964.3	2590.86	2985.26

Table D.8: Fuzzy Rules 6-10 Trading the Deutschmark (84-87)

	F11	F12	F13	F14	F15
Trading-Period-Years	3.02	3.02	3.02	3.02	3.02
Total-Closed-Trades	5.0	1.0	48.0	19.0	44.0
Total-Proft-Trades	5.0	1.0	30.0	11.0	31.0
Total-Losing-Trades	0.0	0.0	18.0	8.0	13.0
Profitable-Trades-pcn	100.0	100.0	62.5	57.8947	70.4546
Total-Gains-$	9917.56	2043.06	42015.0	-1166.38	33223.2
Initial-Capital-$	100000.0	100000.0	100000.0	100000.0	100000.0
Capital-After-Trading-$	109918.0	102043.0	142015.0	98833.6	133223.0
Av-Gains-per-Trade-$	1983.51	2043.06	875.313	-61.3882	755.073
Max-Proft-Trade-$	3022.79	2043.06	8906.62	3988.68	9385.22
Max-Losing-Trade-$	0.0	0.0	-5108.08	-6747.25	-5940.16
Av-Proft-Trade-$	1983.52	2043.06	2634.14	1629.92	2338.63
Av-Losing-Trade-$	0	0	-2056.07	-2386.95	-3021.11
Maximum-Drawdown-$	0.0	0.0	-9327.89	-6747.25	-12743.5
Buy-Signals-pcn	100.0	0.0	100.0	0.0	100.0
Profit-Buy-Trades	100.0	0	62.5	0	70.4546
Profit-Sell-Trades	0	100.0	0	57.89	0
Return-per-year-pcn	3.18	0.67	12.31	-0.38	9.96
Std-Dev-of-PL	978.43	0.0	2944.12	2466.92	3107.32

Table D.9: Fuzzy Rules 11-15 Trading the Deutschmark (84-87)

	F16	F17	F18	F19	F20
Trading-Period-Years	3.02	3.02	3.02	3.02	3.02
Total-Closed-Trades	23.0	55.0	13.0	39.0	22.0
Total-Proft-Trades	11.0	25.0	7.0	12.0	12.0
Total-Losing-Trades	12.0	30.0	6.0	27.0	10.0
Profitable-Trades-pcn	47.82	45.45	53.84	30.76	54.54
Total-Gains-$	-6973.56	-28807.1	3059.16	-29170.0	13296.4
Initial-Capital-$	100000.0	100000.0	100000.0	100000.0	100000.0
Capital-After-Trading-$	93026.4	71192.9	103059.0	70830.0	113296.0
Av-Gains-per-Trade-$	-303.19	-523.76	235.32	-747.95	604.38
Max-Proft-Trade-$	3662.98	2995.71	8069.13	5857.48	10740.4
Max-Losing-Trade-$	-4942.07	-4902.71	-6013.4	-7392.56	-4679.26
Av-Proft-Trade-$	1809.62	1390.33	2743.16	2339.49	2997.53
Av-Losing-Trade-$	-2239.94	-2118.85	-2690.49	-2120.14	-2267.4
Maximum-Drawdown-$	-10064.9	-10227.1	-8511.2	-15460.7	-10977.7
Buy-Signals-pcn	0.0	0.0	100.0	0.0	100.0
Profit-Buy-Trades	0	0	53.85	0	54.55
Profit-Sell-Trades	47.83	45.45	0	30.77	0
Return-per-year-pcn	-2.36	-10.64	1.00	-10.79	4.22
Std-Dev-of-PL	2514.19	2088.42	3483.51	2598.74	3499.82

Table D.10: Fuzzy Rules 16-20 Trading the Deutschmark (84-87)

	F21	F22	F23	F24	F25
Trading-Period-Years	–	3.02	–	3.02	3.02
Total-Closed-Trades	–	3.0	–	3.0	51.0
Total-Proft-Trades	–	0.0	–	0.0	26.0
Total-Losing-Trades	–	3.0	–	3.0	25.0
Profitable-Trades-pcn	–	0.0	–	0.0	50.98
Total-Gains-$	–	-3958.97	–	-6212.91	-21474.8
Initial-Capital-$	–	100000.0	–	100000.0	100000.0
Capital-After-Trading-$	–	96041.0	–	93787.1	78525.2
Av-Gains-per-Trade-$	–	-1319.66	–	-2070.97	-421.074
Max-Proft-Trade-$	–	0.0	–	0.0	4681.92
Max-Losing-Trade-$	–	-2515.08	–	-3429.12	-5126.65
Av-Proft-Trade-$	–	0	–	0	1617.38
Av-Losing-Trade-$	–	-1319.66	–	-2070.97	-2541.07
Maximum-Drawdown-$	–	-3958.99	–	-6212.92	-10025.0
Buy-Signals-pcn	–	0.0	–	0.0	0.0
Profit-Buy-Trades	–	0	–	0	0
Profit-Sell-Trades	–	0.0	–	0.0	50.9804
Return-per-year-pcn	–	-1.32	–	-2.10	-7.69
Std-Dev-of-PL	–	922.74	–	1327.89	2441.29

Table D.11: Fuzzy Rules 21-25 Trading the Deutschmark (84-87)

	F26	F27	F28
Trading-Period-Years	3.02	3.02	3.02
Total-Closed-Trades	12.0	36.0	21.0
Total-Proft-Trades	7.0	11.0	11.0
Total-Losing-Trades	5.0	25.0	10.0
Profitable-Trades-pcn	58.33	30.56	52.38
Total-Gains-$	10262.0	-22871.1	13218.4
Initial-Capital-$	100000.0	100000.0	100000.0
Capital-After-Trading-$	110262.0	77128.9	113218.0
Av-Gains-per-Trade-$	855.16	-635.30	629.45
Max-Proft-Trade-$	8633.09	6378.38	10733.0
Max-Losing-Trade-$	-4578.85	-8049.97	-4676.04
Av-Proft-Trade-$	2899.17	2805.32	3261.54
Av-Losing-Trade-$	-2006.45	-2149.19	-2265.84
Maximum-Drawdown-$	-8511.2	-16835.6	-10970.2
Buy-Signals-pcn	100.0	0.0	100.0
Profit-Buy-Trades	58.33	0	52.38
Profit-Sell-Trades	0	30.55	0
Return-per-year-pcn	3.28	-8.24	4.19
Std-Dev-of-PL	3232.09	2899.35	3577.72

Table D.12: Fuzzy Rules 26-28 Trading the Deutschmark (84-87)

Appendix E
Rule Comparisons and Induced Genetic Rules

E.1 Rule Comparisons

This is an example of the rule comparison mechanism introduced in Chapter 7. The rules with E suffixes are existing expert rule bases and the rules with G suffixes are ones that have been induced by the genetic algorithm. The comparison is made by the comparing-rules function.

```
    [[E1 [ma-diff-1-20-classified-value []] [ma-diff-1-50-classified-value [positive]]

    [vol-ma-diff-1-10-classified-value [negative]] [vol-ma-diff-1-20-classified-value
[]]

    [oi-ma-diff-1-10-classified-value []] [oi-ma-diff-1-20-classified-value [neutral]]

    [price-vola-20 [high]] [price-vola-50 []] [action [UP]]]

    [E2 [ma-diff-1-20-classified-value []] [ma-diff-1-50-classified-value [positive]]

    [vol-ma-diff-1-10-classified-value []] [vol-ma-diff-1-20-classified-value []]

    [oi-ma-diff-1-10-classified-value []] [oi-ma-diff-1-20-classified-value []]

    [price-vola-20 [high]] [price-vola-50 []] [action [UP]]]

    [E3 [ma-diff-1-20-classified-value [negative]] [ma-diff-1-50-classified-value [positive

    [vol-ma-diff-1-10-classified-value []] [vol-ma-diff-1-20-classified-value [negative]]

    [oi-ma-diff-1-10-classified-value []] [oi-ma-diff-1-20-classified-value []]

    [price-vola-20 [high]] [price-vola-50 [medium]] [action [UP]]]

    [E4 [ma-diff-1-20-classified-value [positive]] [ma-diff-1-50-classified-value []]

    [vol-ma-diff-1-10-classified-value []] [vol-ma-diff-1-20-classified-value [negative]]

    [oi-ma-diff-1-10-classified-value []] [oi-ma-diff-1-20-classified-value []]
```

```
[price-vola-20 []] [price-vola-50 [low]] [action [DOWN]]]

[E5 [ma-diff-1-20-classified-value [neutral]] [ma-diff-1-50-classified-value []]

[vol-ma-diff-1-10-classified-value []] [vol-ma-diff-1-20-classified-value [neutral]]

[oi-ma-diff-1-10-classified-value [negative]] [oi-ma-diff-1-20-classified-value []]

[price-vola-20 [medium]] [price-vola-50 []]

[action [UP]]] ] → expert-rulebase;

[[G7 [ma-diff-1-20-classified-value [positive]] [ma-diff-1-50-classified-value []]

[vol-ma-diff-1-10-classified-value []] [vol-ma-diff-1-20-classified-value []]

[oi-ma-diff-1-10-classified-value [positive]] [oi-ma-diff-1-20-classified-value []]

[price-vola-20 [low]] [price-vola-50 []] [action [DOWN]]]

[G8 [ma-diff-1-20-classified-value [negative]] [ma-diff-1-50-classified-value [negative

[vol-ma-diff-1-10-classified-value []] [vol-ma-diff-1-20-classified-value []]

[oi-ma-diff-1-10-classified-value []] [oi-ma-diff-1-20-classified-value [neutral]]

[price-vola-20 [high]] [price-vola-50 [high]] [action [UP]]]

[G9 [ma-diff-1-20-classified-value []] [ma-diff-1-50-classified-value [positive]]

[vol-ma-diff-1-10-classified-value []] [vol-ma-diff-1-20-classified-value []]

[oi-ma-diff-1-10-classified-value []] [oi-ma-diff-1-20-classified-value []]

[price-vola-20 [high]] [price-vola-50 []]

[action [UP]]] ] → genetic-rulebase;
```

The following comparing-rules function performs the similarity base comparison and returns the least similar rule (G7 in this example) for addition to the permanent rule base.

comparing-rules(1,expert-rulebase,genetic-rulebase) → rules-to-add;

rules-to-add== →

```
** [[G7 [ma-diff-1-20-classified-value [positive]] [ma-diff-1-50-classified-value
[]]
[vol-ma-diff-1-10-classified-value []] [vol-ma-diff-1-20-classified-value []]

[oi-ma-diff-1-10-classified-value [positive]] [oi-ma-diff-1-20-classified-value []]

[price-vola-20 [low]] [price-vola-50 []] [action [DOWN]]]]
```

Another example where the two least similar rules G7 and G8 are found for addition to the permanent rule base.

comparing-rules(2,expert-rulebase,genetic-rulebase) → rules-to-add;

rules-to-add== →

```
** [[G7 [ma-diff-1-20-classified-value [positive]] [ma-diff-1-50-classified-value
[]]

[vol-ma-diff-1-10-classified-value []] [vol-ma-diff-1-20-classified-value []]

[oi-ma-diff-1-10-classified-value [positive]] [oi-ma-diff-1-20-classified-value []]

[price-vola-20 [low]] [price-vola-50 []] [action [DOWN]]]

[G8 [ma-diff-1-20-classified-value [negative]] [ma-diff-1-50-classified-value [negative

[vol-ma-diff-1-10-classified-value []] [vol-ma-diff-1-20-classified-value []]

[oi-ma-diff-1-10-classified-value []] [oi-ma-diff-1-20-classified-value [neutral]]

[price-vola-20 [high]] [price-vola-50 [high]] [action [UP]]]]
```

E.2 The Symbolic Mode Induced Rules

The following is the rule base induced by the genetic algorithm in the symbolic mode with the *Set A* variables using **British Pound** data from 1984 to 1987. The results from applying this induced rule base to data between 1988 and 1992 was presented in Chapter 7.

In all of the presented rules (and rule bases) the numerical figure at the end indicates the fitness of that rule (rule base).

```
[[G1 [ma-diff-1-5-classified-value []] [ma-diff-1-10-classified-value []]

[ma-diff-1-20-classified-value []] [ma-diff-1-50-classified-value []]
[ma-diff-1-100-classified-value []] [ma-diff-1-200-classified-value [negative]]
[vol-ma-diff-1-10-classified-value []] [vol-ma-diff-1-20-classified-value []]
[oi-ma-diff-1-10-classified-value []] [oi-ma-diff-1-20-classified-value []]
[price-vola-20 [high]] [price-vola-50 []]
[price-vola-100 []] [action [DOWN]] 1 0.541353]

[G2 [ma-diff-1-5-classified-value []] [ma-diff-1-10-classified-value []]
[ma-diff-1-20-classified-value []] [ma-diff-1-50-classified-value []]
[ma-diff-1-100-classified-value []] [ma-diff-1-200-classified-value []]
[vol-ma-diff-1-10-classified-value []] [vol-ma-diff-1-20-classified-value []]
[oi-ma-diff-1-10-classified-value []] [oi-ma-diff-1-20-classified-value []]
[price-vola-20 [high]] [price-vola-50 []]
[price-vola-100 []] [action [DOWN]] 2 0.520505]

[G3 [ma-diff-1-5-classified-value []] [ma-diff-1-10-classified-value []]
[ma-diff-1-20-classified-value []] [ma-diff-1-50-classified-value []]
[ma-diff-1-100-classified-value []] [ma-diff-1-200-classified-value []]
[vol-ma-diff-1-10-classified-value [positive]] [vol-ma-diff-1-20-classified-value
[]]
[oi-ma-diff-1-10-classified-value []] [oi-ma-diff-1-20-classified-value []]
[price-vola-20 [high]] [price-vola-50 []]
[price-vola-100 []] [action [DOWN]] 3 0.503145]

[G4 [ma-diff-1-5-classified-value []] [ma-diff-1-10-classified-value []]
[ma-diff-1-20-classified-value []] [ma-diff-1-50-classified-value []]
[ma-diff-1-100-classified-value [positive]] [ma-diff-1-200-classified-value []]
[vol-ma-diff-1-10-classified-value []] [vol-ma-diff-1-20-classified-value []]
[oi-ma-diff-1-10-classified-value []] [oi-ma-diff-1-20-classified-value []]
[price-vola-20 []] [price-vola-50 []]
[price-vola-100 []] [action [UP]] 4 0.492958]
```

```
    [G5 [ma-diff-1-5-classified-value []] [ma-diff-1-10-classified-value []]
    [ma-diff-1-20-classified-value []] [ma-diff-1-50-classified-value []]
    [ma-diff-1-100-classified-value []] [ma-diff-1-200-classified-value [positive]]
    [vol-ma-diff-1-10-classified-value [neutral]] [vol-ma-diff-1-20-classified-value
[]]
    [oi-ma-diff-1-10-classified-value []] [oi-ma-diff-1-20-classified-value []]
    [price-vola-20 []] [price-vola-50 []]
    [price-vola-100 []] [action [UP]] 5 0.48954]

    [G6 [ma-diff-1-5-classified-value []] [ma-diff-1-10-classified-value []]
    [ma-diff-1-20-classified-value []] [ma-diff-1-50-classified-value []]
    [ma-diff-1-100-classified-value []] [ma-diff-1-200-classified-value [positive]]
    [vol-ma-diff-1-10-classified-value []] [vol-ma-diff-1-20-classified-value [neutral]]
    [oi-ma-diff-1-10-classified-value []] [oi-ma-diff-1-20-classified-value []]
    [price-vola-20 []] [price-vola-50 []]
    [price-vola-100 [high]] [action [UP]] 6 0.488636]

    [G7 [ma-diff-1-5-classified-value []] [ma-diff-1-10-classified-value []]
    [ma-diff-1-20-classified-value []] [ma-diff-1-50-classified-value []]
    [ma-diff-1-100-classified-value []] [ma-diff-1-200-classified-value [negative]]
    [vol-ma-diff-1-10-classified-value []] [vol-ma-diff-1-20-classified-value [neutral]]
    [oi-ma-diff-1-10-classified-value []] [oi-ma-diff-1-20-classified-value []]
    [price-vola-20 []] [price-vola-50 []]
    [price-vola-100 []] [action [UP]] 7 0.488636]

    [G8 [ma-diff-1-5-classified-value []] [ma-diff-1-10-classified-value []]
    [ma-diff-1-20-classified-value []] [ma-diff-1-50-classified-value []]
    [ma-diff-1-100-classified-value [positive]] [ma-diff-1-200-classified-value []]
    [vol-ma-diff-1-10-classified-value [neutral]] [vol-ma-diff-1-20-classified-value
[]]
    [oi-ma-diff-1-10-classified-value []] [oi-ma-diff-1-20-classified-value []]
    [price-vola-20 []] [price-vola-50 []]

    [price-vola-100 []] [action [UP]] 8 0.487805]
```

```
[G9 [ma-diff-1-5-classified-value []] [ma-diff-1-10-classified-value []]

[ma-diff-1-20-classified-value []] [ma-diff-1-50-classified-value []]

[ma-diff-1-100-classified-value []] [ma-diff-1-200-classified-value [positive]]

[vol-ma-diff-1-10-classified-value []] [vol-ma-diff-1-20-classified-value []]

[oi-ma-diff-1-10-classified-value []] [oi-ma-diff-1-20-classified-value []]

[price-vola-20 [high]] [price-vola-50 []]

[price-vola-100 []] [action [DOWN]] 9 0.483645]]
```

The following is the rule base induced by the genetic algorithm in the symbolic mode with the *Set B* variables using British Pound data from 1984 to 1987.

```
[[G1[price-rsi-7 []] [price-rsi-14 []] [price-rsi-28 []]

[vol-rsi-7 []] [vol-rsi-14 []] [vol-rsi-28 []]

[oi-rsi-7 [medium]] [oi-rsi-14 [medium]] [oi-rsi-28 []]

[price-vola-10 [medium]] [price-vola-20 [high]] [price-vola-50 []]

[price-vola-100 [high]] [action [DOWN]] 18 0.65]
```

```
[G2[price-rsi-7 []] [price-rsi-14 []] [price-rsi-28 []]

[vol-rsi-7 []] [vol-rsi-14 []] [vol-rsi-28 []]

[oi-rsi-7 []] [oi-rsi-14 [medium]] [oi-rsi-28 []]

[price-vola-10 []] [price-vola-20 [medium]] [price-vola-50 []]

[price-vola-100 [medium]] [action [DOWN]] 1 0.564103]
```

```
[G3[price-rsi-7 []] [price-rsi-14 []] [price-rsi-28 []]

[vol-rsi-7 []] [vol-rsi-14 []] [vol-rsi-28 []]

[oi-rsi-7 [medium]] [oi-rsi-14 [medium]] [oi-rsi-28 []]

[price-vola-10 []] [price-vola-20 [medium]] [price-vola-50 []]

[price-vola-100 [high]] [action [DOWN]] 2 0.557377]
```

```
[G4[price-rsi-7 []] [price-rsi-14 []] [price-rsi-28 []]
[vol-rsi-7 []] [vol-rsi-14 []] [vol-rsi-28 []]
[oi-rsi-7 []] [oi-rsi-14 [medium]] [oi-rsi-28 []]
[price-vola-10 []] [price-vola-20 [high]] [price-vola-50 [medium]]
[price-vola-100 [high]] [action [DOWN]] 3 0.545455]

[G5[price-rsi-7 []] [price-rsi-14 []] [price-rsi-28 []]
[vol-rsi-7 []] [vol-rsi-14 []] [vol-rsi-28 []]
[oi-rsi-7 []] [oi-rsi-14 [medium]] [oi-rsi-28 []]
[price-vola-10 []] [price-vola-20 [medium]] [price-vola-50 []]
[price-vola-100 []] [action [DOWN]] 4 0.541176]

[G6[price-rsi-7 [low]] [price-rsi-14 []] [price-rsi-28 []]
[vol-rsi-7 []] [vol-rsi-14 []] [vol-rsi-28 []]
[oi-rsi-7 [medium]] [oi-rsi-14 [medium]] [oi-rsi-28 []]
[price-vola-10 []] [price-vola-20 [high]] [price-vola-50 []]
[price-vola-100 []] [action [DOWN]] 5 0.537879]

[G7[price-rsi-7 [low]] [price-rsi-14 []] [price-rsi-28 []]
[vol-rsi-7 []] [vol-rsi-14 []] [vol-rsi-28 []]
[oi-rsi-7 [medium]] [oi-rsi-14 [medium]] [oi-rsi-28 []]
[price-vola-10 []] [price-vola-20 [high]] [price-vola-50 []]
[price-vola-100 [high]] [action [DOWN]] 6 0.537879]

[G8[price-rsi-7 [low]] [price-rsi-14 []] [price-rsi-28 []]
[vol-rsi-7 []] [vol-rsi-14 []] [vol-rsi-28 []]
[oi-rsi-7 [medium]] [oi-rsi-14 []] [oi-rsi-28 []]
[price-vola-10 []] [price-vola-20 [high]] [price-vola-50 []]
[price-vola-100 []] [action [DOWN]] 7 0.537879]
[G9[price-rsi-7 [low]] [price-rsi-14 []] [price-rsi-28 []]
[vol-rsi-7 []] [vol-rsi-14 []] [vol-rsi-28 []]
[oi-rsi-7 [medium]] [oi-rsi-14 []] [oi-rsi-28 []]
[price-vola-10 []] [price-vola-20 [low]] [price-vola-50 []]
[price-vola-100 [high]] [action [DOWN]] 8 0.537879]]
```

Here the rule base which was induced with the Set A variables using the **Deutschmark** data is presented. Again as with the British Pound data the results of applying these rules to unseen data was presented in Chapter 7.

```
[[[G1 [ma-diff-1-5-classified-value []] [ma-diff-1-10-classified-value []]
[ma-diff-1-20-classified-value []] [ma-diff-1-50-classified-value []]
[ma-diff-1-100-classified-value []] [ma-diff-1-200-classified-value [positive]]
[vol-ma-diff-1-10-classified-value []] [vol-ma-diff-1-20-classified-value []]
[oi-ma-diff-1-10-classified-value [negative]] [oi-ma-diff-1-20-classified-value []]
[price-vola-20 [high]] [price-vola-50 []]
[price-vola-100 []] [action [UP]] 1 0.671233]

[G2 [ma-diff-1-5-classified-value []] [ma-diff-1-10-classified-value []]
[ma-diff-1-20-classified-value []] [ma-diff-1-50-classified-value []]
[ma-diff-1-100-classified-value []] [ma-diff-1-200-classified-value [positive]]
[vol-ma-diff-1-10-classified-value []] [vol-ma-diff-1-20-classified-value []]
[oi-ma-diff-1-10-classified-value [negative]] [oi-ma-diff-1-20-classified-value []]
[price-vola-20 [high]] [price-vola-50 []]
[price-vola-100 [high]] [action [UP]] 2 0.666667]

[G3 [ma-diff-1-5-classified-value []] [ma-diff-1-10-classified-value []]
[ma-diff-1-20-classified-value []] [ma-diff-1-50-classified-value []]
[ma-diff-1-100-classified-value []] [ma-diff-1-200-classified-value [positive]]
[vol-ma-diff-1-10-classified-value []] [vol-ma-diff-1-20-classified-value []]
[oi-ma-diff-1-10-classified-value []] [oi-ma-diff-1-20-classified-value []]
[price-vola-20 []] [price-vola-50 []]
[price-vola-100 [high]] [action [UP]] 3 0.642857]
```

```
[G4 [ma-diff-1-5-classified-value []] [ma-diff-1-10-classified-value []]
[ma-diff-1-20-classified-value []] [ma-diff-1-50-classified-value []]
[ma-diff-1-100-classified-value []] [ma-diff-1-200-classified-value [positive]]
[vol-ma-diff-1-10-classified-value []] [vol-ma-diff-1-20-classified-value []]
[oi-ma-diff-1-10-classified-value []] [oi-ma-diff-1-20-classified-value []]
[price-vola-20 [high]] [price-vola-50 []]
[price-vola-100 [high]] [action [UP]] 4 0.642082]

[G5 [ma-diff-1-5-classified-value []] [ma-diff-1-10-classified-value []]
[ma-diff-1-20-classified-value []] [ma-diff-1-50-classified-value [positive]]
[ma-diff-1-100-classified-value []] [ma-diff-1-200-classified-value [positive]]
[vol-ma-diff-1-10-classified-value []] [vol-ma-diff-1-20-classified-value []]
[oi-ma-diff-1-10-classified-value [negative]] [oi-ma-diff-1-20-classified-value []]
[price-vola-20 [high]] [price-vola-50 []]
[price-vola-100 []] [action [UP]] 5 0.641791]

[G6 [ma-diff-1-5-classified-value []] [ma-diff-1-10-classified-value []]
[ma-diff-1-20-classified-value []] [ma-diff-1-50-classified-value []]
[ma-diff-1-100-classified-value []] [ma-diff-1-200-classified-value [positive]]
[vol-ma-diff-1-10-classified-value []] [vol-ma-diff-1-20-classified-value []]
[oi-ma-diff-1-10-classified-value []] [oi-ma-diff-1-20-classified-value []]
[price-vola-20 []] [price-vola-50 []]
[price-vola-100 []] [action [UP]] 6 0.638655]

[G7 [ma-diff-1-5-classified-value []] [ma-diff-1-10-classified-value []]
[ma-diff-1-20-classified-value []] [ma-diff-1-50-classified-value []]
[ma-diff-1-100-classified-value []] [ma-diff-1-200-classified-value [positive]]
[vol-ma-diff-1-10-classified-value []] [vol-ma-diff-1-20-classified-value []]
[oi-ma-diff-1-10-classified-value []] [oi-ma-diff-1-20-classified-value []]
[price-vola-20 [high]] [price-vola-50 []]
[price-vola-100 []] [action [UP]] 7 0.637895]

[G8 [ma-diff-1-5-classified-value []] [ma-diff-1-10-classified-value []]

[ma-diff-1-20-classified-value []] [ma-diff-1-50-classified-value [positive]]

[ma-diff-1-100-classified-value []] [ma-diff-1-200-classified-value [positive]]

[vol-ma-diff-1-10-classified-value []] [vol-ma-diff-1-20-classified-value []]

[oi-ma-diff-1-10-classified-value [negative]] [oi-ma-diff-1-20-classified-value []]

[price-vola-20 [high]] [price-vola-50 []]

[price-vola-100 [high]] [action [UP]] 8 0.636364]

[G9 [ma-diff-1-5-classified-value []] [ma-diff-1-10-classified-value []]

[ma-diff-1-20-classified-value []] [ma-diff-1-50-classified-value [positive]]

[ma-diff-1-100-classified-value []] [ma-diff-1-200-classified-value [positive]]

[vol-ma-diff-1-10-classified-value []] [vol-ma-diff-1-20-classified-value []]

[oi-ma-diff-1-10-classified-value []] [oi-ma-diff-1-20-classified-value []]

[price-vola-20 [high]] [price-vola-50 []]

[price-vola-100 [high]] [action [UP]] 9 0.621176]]]
```

Here is the rule base induced using Deutschmark Set B data.

```
[[[G1 [price-rsi-7 [low]] [price-rsi-14 []] [price-rsi-28 []] [vol-rsi-7 []]

[vol-rsi-14 []] [vol-rsi-28 []] [oi-rsi-7 []] [oi-rsi-14 []]

[oi-rsi-28 []] [price-vola-10 [high]] [price-vola-20 []] [price-vola-50 []]

[price-vola-100 []] [action [UP]] 1 0.634783]

[G2 [price-rsi-7 [low]] [price-rsi-14 []] [price-rsi-28 []] [vol-rsi-7 [medium]]

[vol-rsi-14 []] [vol-rsi-28 []] [oi-rsi-7 []] [oi-rsi-14 []]

[oi-rsi-28 []] [price-vola-10 [high]] [price-vola-20 []] [price-vola-50 []]

[price-vola-100 []] [action [UP]] 2 0.634615]

[G3 [price-rsi-7 [low]] [price-rsi-14 []] [price-rsi-28 []] [vol-rsi-7 [medium]]

[vol-rsi-14 []] [vol-rsi-28 [medium]] [oi-rsi-7 []] [oi-rsi-14 []]
[oi-rsi-28 []] [price-vola-10 [high]] [price-vola-20 []] [price-vola-50 []]
[price-vola-100 []] [action [UP]] 3 0.634615]

[G4 [price-rsi-7 [low]] [price-rsi-14 []] [price-rsi-28 []] [vol-rsi-7 [medium]]
[vol-rsi-14 []] [vol-rsi-28 [medium]] [oi-rsi-7 []] [oi-rsi-14 []]
[oi-rsi-28 []] [price-vola-10 [high]] [price-vola-20 []] [price-vola-50 []]
[price-vola-100 [high]] [action [UP]] 15 0.634615]

[G5 [price-rsi-7 [low]] [price-rsi-14 []] [price-rsi-28 []] [vol-rsi-7 []]
[vol-rsi-14 []] [vol-rsi-28 []] [oi-rsi-7 []] [oi-rsi-14 []]
[oi-rsi-28 []] [price-vola-10 [high]] [price-vola-20 []] [price-vola-50 []]
[price-vola-100 [low]] [action [UP]] 20 0.631579]

[G6 [price-rsi-7 []] [price-rsi-14 []] [price-rsi-28 []] [vol-rsi-7 [medium]]
[vol-rsi-14 []] [vol-rsi-28 []] [oi-rsi-7 []] [oi-rsi-14 []]
[oi-rsi-28 []] [price-vola-10 [high]] [price-vola-20 []] [price-vola-50 []]
[price-vola-100 []] [action [UP]] 4 0.59802]

[G7 [price-rsi-7 []] [price-rsi-14 []] [price-rsi-28 []] [vol-rsi-7 [medium]]
[vol-rsi-14 [medium]] [vol-rsi-28 []] [oi-rsi-7 []] [oi-rsi-14 []]
[oi-rsi-28 []] [price-vola-10 [high]] [price-vola-20 []] [price-vola-50 []]
[price-vola-100 []] [action [UP]] 5 0.59802]

[G8 [price-rsi-7 []] [price-rsi-14 []] [price-rsi-28 []] [vol-rsi-7 []]
[vol-rsi-14 [medium]] [vol-rsi-28 []] [oi-rsi-7 []] [oi-rsi-14 []]
[oi-rsi-28 []] [price-vola-10 [high]] [price-vola-20 []] [price-vola-50 []]
[price-vola-100 [low]] [action [DOWN]] 16 0.597786]

[G9 [price-rsi-7 []] [price-rsi-14 []] [price-rsi-28 []] [vol-rsi-7 []]
[vol-rsi-14 []] [vol-rsi-28 []] [oi-rsi-7 []] [oi-rsi-14 [medium]]
[oi-rsi-28 []] [price-vola-10 [high]] [price-vola-20 []] [price-vola-50 []]
[price-vola-100 []] [action [UP]] 22 0.593807]]]
```

E.3 The Fuzzy Mode Induced Rules

The following are the fittest two fuzzy rule bases (FR1,FR2) induced using the British Pound data.

```
[[FR1[G1[ma-diff-1-50-fuzzy-values []] [ma-diff-1-100-fuzzy-values []]
[ma-diff-1-200-fuzzy-values [negative]] [vol-ma-diff-1-10-fuzzy-values []]
[vol-ma-diff-1-20-fuzzy-values []] [oi-ma-diff-1-10-fuzzy-values []]
[oi-ma-diff-1-20-fuzzy-values [neutral]] [price-fuzzy-vola-20 []]
[price-fuzzy-vola-50 []] [price-fuzzy-vola-100 []] [action [DOWN]]]

[G2[ma-diff-1-50-fuzzy-values []] [ma-diff-1-100-fuzzy-values []]
[ma-diff-1-200-fuzzy-values [negative]] [vol-ma-diff-1-10-fuzzy-values []]
[vol-ma-diff-1-20-fuzzy-values []] [oi-ma-diff-1-10-fuzzy-values []]
[oi-ma-diff-1-20-fuzzy-values [neutral]] [price-fuzzy-vola-20 []]
[price-fuzzy-vola-50 []] [price-fuzzy-vola-100 []] [action [DOWN]]]

[G3[ma-diff-1-50-fuzzy-values []] [ma-diff-1-100-fuzzy-values []]
[ma-diff-1-200-fuzzy-values [negative]] [vol-ma-diff-1-10-fuzzy-values []]
[vol-ma-diff-1-20-fuzzy-values []] [oi-ma-diff-1-10-fuzzy-values []]
[oi-ma-diff-1-20-fuzzy-values [negative]] [price-fuzzy-vola-20 []]
[price-fuzzy-vola-50 []] [price-fuzzy-vola-100 []] [action [DOWN]]]

[G4[ma-diff-1-50-fuzzy-values []] [ma-diff-1-100-fuzzy-values []]
[ma-diff-1-200-fuzzy-values [negative]] [vol-ma-diff-1-10-fuzzy-values []]
[vol-ma-diff-1-20-fuzzy-values []] [oi-ma-diff-1-10-fuzzy-values []]
[oi-ma-diff-1-20-fuzzy-values [negative]] [price-fuzzy-vola-20 []]
[price-fuzzy-vola-50 []] [price-fuzzy-vola-100 []] [action [DOWN]]] 1 0.635294]

[FR2[G1[ma-diff-1-50-fuzzy-values []] [ma-diff-1-100-fuzzy-values []]
[ma-diff-1-200-fuzzy-values [negative]] [vol-ma-diff-1-10-fuzzy-values []]
[vol-ma-diff-1-20-fuzzy-values []] [oi-ma-diff-1-10-fuzzy-values []]
[oi-ma-diff-1-20-fuzzy-values [neutral]] [price-fuzzy-vola-20 []]

[price-fuzzy-vola-50 []] [price-fuzzy-vola-100 []] [action [DOWN]]]

[G2[ma-diff-1-50-fuzzy-values []] [ma-diff-1-100-fuzzy-values []]

[ma-diff-1-200-fuzzy-values [negative]] [vol-ma-diff-1-10-fuzzy-values []]

[vol-ma-diff-1-20-fuzzy-values []] [oi-ma-diff-1-10-fuzzy-values []]

[oi-ma-diff-1-20-fuzzy-values [neutral]] [price-fuzzy-vola-20 []]

[price-fuzzy-vola-50 []] [price-fuzzy-vola-100 []] [action [DOWN]]]

[G3[ma-diff-1-50-fuzzy-values []] [ma-diff-1-100-fuzzy-values []]

[ma-diff-1-200-fuzzy-values [negative]] [vol-ma-diff-1-10-fuzzy-values []]

[vol-ma-diff-1-20-fuzzy-values []] [oi-ma-diff-1-10-fuzzy-values []]

[oi-ma-diff-1-20-fuzzy-values [negative]] [price-fuzzy-vola-20 []]

[price-fuzzy-vola-50 []] [price-fuzzy-vola-100 []] [action [DOWN]]]

[G4[ma-diff-1-50-fuzzy-values []] [ma-diff-1-100-fuzzy-values []]

[ma-diff-1-200-fuzzy-values [negative]] [vol-ma-diff-1-10-fuzzy-values []]

[vol-ma-diff-1-20-fuzzy-values []] [oi-ma-diff-1-10-fuzzy-values []]

[oi-ma-diff-1-20-fuzzy-values [negative]] [price-fuzzy-vola-20 []]

[price-fuzzy-vola-50 []] [price-fuzzy-vola-100 [high]] [action [DOWN]]] 2 0.635294]]
```

The fuzzy rule bases induced using Deutschmark data is presented next.

```
[[FR1[G1[ma-diff-1-50-fuzzy-values [neutral]] [ma-diff-1-100-fuzzy-values []]

[ma-diff-1-200-fuzzy-values [positive]] [vol-ma-diff-1-10-fuzzy-values []]

[vol-ma-diff-1-20-fuzzy-values [positive]] [oi-ma-diff-1-10-fuzzy-values []]

[oi-ma-diff-1-20-fuzzy-values []] [price-fuzzy-vola-20 [low]]

[price-fuzzy-vola-50 []] [price-fuzzy-vola-100 []] [action [UP]]]

[G2[ma-diff-1-50-fuzzy-values [neutral]] [ma-diff-1-100-fuzzy-values []]

[ma-diff-1-200-fuzzy-values [positive]] [vol-ma-diff-1-10-fuzzy-values []]
```

```
[vol-ma-diff-1-20-fuzzy-values [positive]] [oi-ma-diff-1-10-fuzzy-values []]
[oi-ma-diff-1-20-fuzzy-values []] [price-fuzzy-vola-20 []]
[price-fuzzy-vola-50 []] [price-fuzzy-vola-100 []] [action [UP]]]

[G3[ma-diff-1-50-fuzzy-values []] [ma-diff-1-100-fuzzy-values []]
[ma-diff-1-200-fuzzy-values [neutral]] [vol-ma-diff-1-10-fuzzy-values []]
[vol-ma-diff-1-20-fuzzy-values [positive]] [oi-ma-diff-1-10-fuzzy-values [neutral]]
[oi-ma-diff-1-20-fuzzy-values []] [price-fuzzy-vola-20 []]
[price-fuzzy-vola-50 [low]] [price-fuzzy-vola-100 []] [action [UP]]]

[G4[ma-diff-1-50-fuzzy-values []] [ma-diff-1-100-fuzzy-values []]
[ma-diff-1-200-fuzzy-values [positive]] [vol-ma-diff-1-10-fuzzy-values []]
[vol-ma-diff-1-20-fuzzy-values [positive]] [oi-ma-diff-1-10-fuzzy-values [neutral]]
[oi-ma-diff-1-20-fuzzy-values []] [price-fuzzy-vola-20 []]
[price-fuzzy-vola-50 []] [price-fuzzy-vola-100 []] [action [UP]]] 1 0.671875]


[FR2[G1[ma-diff-1-50-fuzzy-values [positive]] [ma-diff-1-100-fuzzy-values []]
[ma-diff-1-200-fuzzy-values [positive]] [vol-ma-diff-1-10-fuzzy-values []]
[vol-ma-diff-1-20-fuzzy-values [positive]] [oi-ma-diff-1-10-fuzzy-values [neutral]]
[oi-ma-diff-1-20-fuzzy-values []] [price-fuzzy-vola-20 [low]]
[price-fuzzy-vola-50 []] [price-fuzzy-vola-100 []] [action [UP]]]

[G2[ma-diff-1-50-fuzzy-values [neutral]] [ma-diff-1-100-fuzzy-values []]
[ma-diff-1-200-fuzzy-values [positive]] [vol-ma-diff-1-10-fuzzy-values []]
[vol-ma-diff-1-20-fuzzy-values [positive]] [oi-ma-diff-1-10-fuzzy-values [neutral]]
[oi-ma-diff-1-20-fuzzy-values []] [price-fuzzy-vola-20 []]
[price-fuzzy-vola-50 []] [price-fuzzy-vola-100 []] [action [UP]]]

[G3[ma-diff-1-50-fuzzy-values []] [ma-diff-1-100-fuzzy-values []]
[ma-diff-1-200-fuzzy-values [positive]] [vol-ma-diff-1-10-fuzzy-values []]
[vol-ma-diff-1-20-fuzzy-values [positive]] [oi-ma-diff-1-10-fuzzy-values [neutral]]
```

```
[oi-ma-diff-1-20-fuzzy-values []] [price-fuzzy-vola-20 []]

[price-fuzzy-vola-50 []] [price-fuzzy-vola-100 []] [action [UP]]]

[G4[ma-diff-1-50-fuzzy-values [neutral]] [ma-diff-1-100-fuzzy-values []]

[ma-diff-1-200-fuzzy-values [positive]] [vol-ma-diff-1-10-fuzzy-values []]

[vol-ma-diff-1-20-fuzzy-values [positive]] [oi-ma-diff-1-10-fuzzy-values []]

[oi-ma-diff-1-20-fuzzy-values []] [price-fuzzy-vola-20 []]

[price-fuzzy-vola-50 []] [price-fuzzy-vola-100 []] [action [UP]]] 2 0.671875]]
```

Appendix F
INTENT Main Programs

This appendix contains a selection of the main POP-11 routines of the INTENT system. The full listings can be found in /cs/academic/phd1/stanley/surang/INTENT/intent There are three main associated files: expert.p, fuzzy.p and gri.p.

```
  --------------   EXPERT SYSTEM - (expert.p) ----------------------------- */


 finding the "best" clusters from the cluster tree */

fine find-best-clusters(ling-terms,clusterlist-lists)->
       best-clusters;


     /*find total number of points - because the final binary partition
       has all the elements, you only need to count all the elements in
       those two*/

  length(clusterlist(1)) + length(clusterlist(2)) -> total-number-points;

  total-number-points / length(ling-terms) ->ideal-cluster-length;

  /* construct a list which has a number of lists corresponding to the
  number of elements in ling-terms */

  []->slots-list;

  repeat length(ling-terms) times
      [~~slots-list []]->slots-list;
  endrepeat;

  for clus in clusterlist do

      /* Checking for common elements - cluster and slots */
      false->common-elements-present;  []->common-elem-slot;0->count-common;

      for slot in slots-list do
          if length(slot) > 0 then
              0->elemcount; false->match-found;

                until match-found = true or elemcount= length(slot) do
                  elemcount + 1 ->elemcount;
                  slot(elemcount)->elem;
                  if member(elem,clus) then
                      /* beacause more than one common element */
                      count-common + 1 ->count-common;
                      slot->common-elem-slot;
                      true->common-elements-present;
                      true->match-found;
                  endif;
              enduntil;
          endif;
      endfor;

      /* checking for a clean common replacement */

      if count-common = 1 then

         /* find if the clus is closer to the ideal-cluster-length
          than the one in the slot*/

          if abs(length(clus) - ideal-cluster-length) <
              abs(length(common-elem-slot) - ideal-cluster-length)
          then
              slots-list-->[??firstbit ~~common-elem-slot ??secondbit];
              /* replace slot*/
              [~~firstbit ~clus ~~secondbit]->slots-list;
          endif;
```

```
        else
            if count-common = 0 then

                []->largest-diff-slot; 0->largest-difference;
                /* find the worst slot in the slots list and replace with clus*/

                 for Aslot in slots-list do
                    slotcount + 1->slotcount;
                    abs(length(Aslot) - ideal-cluster-length)-> difference;

                    if difference > largest-difference then
                        difference ->largest-difference;
                        slotcount->largestslotcount; Aslot->largest-diff-slot;
                    endif;
                endfor;


                if abs(length(clus) - ideal-cluster-length) <
                    abs(length(largest-diff-slot) - ideal-cluster-length)
                then
                    slots-list-->[??firstbit largest-diff-slot ??endbit];
                    clus->slots-list(largestslotcount);
                endif;
            endif;
        endif;
    endfor;
    slots-list->best-clusters;
enddefine;


/* classifying new data items */

define classification(value,cluster-ranges)->classified-value;

/* first see if the value is less than the lowest value if so it is low
   else
   then see if it is higher than the highest then high
   else
   if it is within the values of the clusters - give the appropriate name of clus
   else
   find which cluster it is nearest to and assign the name of that cluster
    */

    vars clus name item Fname Lname Ffirstval Llastval found firstval lastval;
    false->found;

    cluster-ranges(1)-->[?Fname [?Ffirstval ==]];
    last(cluster-ranges)-->[?Lname [== ?Llastval]];

    /* lower than the lowest*/
    if value < Ffirstval then Fname ><'' ->classified-value;
        true->found;
    endif;

    /* higher than the highest */
    if value > Llastval then Lname ><''->classified-value;
        true->found;
    endif;

    if found=false then
        /* within the clusters */
        for clus in cluster-ranges do
            clus-->[?name [?firstval == ?lastval]];
            if value >= firstval and value <=lastval then
```

```
              name ><''->classified-value;
              true->found;
              quitloop(1);
          endif;
      endfor;
  endif;

  /* finding the nearest cluster */

  if found = false then
      vars min-clus-distance nearest-clus-name a-dis b-dis;
      /* getting the first nearest cluster distance */
      cluster-ranges(1)-->[?name [?firstval == ?lastval]];
      abs(value - firstval)-> a-dis; abs(value - lastval) -> b-dis;

      if a-dis < b-dis then a-dis ->min-clus-distance;
      else
          b-dis ->min-clus-distance;
      endif;
      name->nearest-clus-name;

      for clus in cluster-ranges do
          clus-->[?name [?firstval == ?lastval]];
          abs(value - firstval)-> a-dis;
          abs(value - lastval) -> b-dis;

          if a-dis <min-clus-distance then
              a-dis->min-clus-distance;
              name->nearest-clus-name;
          endif;

          if b-dis <min-clus-distance then
              b-dis->min-clus-distance;
              name->nearest-clus-name;
          endif;
      endfor;
      nearest-clus-name >< ''->classified-value;
  endif;
idefine;


 Expert system interpreter */

fine interpreter(feature-index,symbolic-array,trade-stindex,trade-endindex,
oduction-rules,conflict-flag)->trades-list;

  for arrayindex from trade-stindex to trade-endindex do
      []->total-rule-stats;
      false->rule-not-apply; []->rule-stats;
      ''->select-id;

       /* take each rule in the current rule context - i.e. expert, genetic
      or fuzzy rules */

      for rule in production-rules do
          []->rule-stats; false->rule-not-apply;

          /* take the identifier*/
          1->feat-cnt;
          rule(feat-cnt)->rule-identifier;

          /* take each feature of a rule*/

          until feat-cnt=length(feature-struct)  do
                feat-cnt + 1->feat-cnt;
                rule(feat-cnt)->feat-in-rule;
                feat-in-rule-->[= ?sub-features];

                /* take each value (numeric or qualitative) in features*/

                0->num-sub-features-matched;

                for sub-item in sub-features do

                    /* here the check is to see whether the sub-features
                    in the array and of the rule match. */

                    sub-item><''->sub-item;

                   /* feat-cnt -1 is to compensate for the addition of the rule
                    identifier */

                    if sub-item =
                     symbolic-array(feature-index(feat-cnt-1),arrayindex) then

                        num-sub-features-matched + 1 ->num-sub-features-matched;
                    endif;
                endfor;

              /* conflict resolution based on specifity - more items
               matched higher the score, better the rule */

                if length(sub-features) =0 then
                    [~rule-stats 0]->rule-stats;
                else
                    [~~rule-stats ~num-sub-features-matched]->rule-stats;
                endif;

                if num-sub-features-matched = 0 and length(sub-features)/=0
                then true ->rule-not-apply;
                endif;
            enduntil;

             if rule-not-apply=false then
                [~~total-rule-stats [~rule-identifier ~rule-stats]]
                    ->total-rule-stats;
             endif;
        endfor;

        /* conflict resolution - apply rules of specificity to select a single rule */

        if length(total-rule-stats) > 0 then
            conflict-resolution(conflict-flag,total-rule-stats)->select-id;

        /* find the coressponding action of the selected rule */
        production-rules-->[== [~select-id == [= [?act]]]   ==];

        /* add rule number E4  G4 F5 EG5 etc, current index and action into
         trades list*/

         [~~trades-list
         [ ~(symbolic-array(cindex(1),arrayindex))
                        ~select-id ~act] ]->trades-list;
        endif;
    endfor;
enddefine;
```

```
 conflict resolution */

fine conflict-resolution(conflict-flag,total-rule-stats)->select-id;
  vars identifier val-list total val add-rule-stats rlst select-id;
  []->add-rule-stats;0->val;0->total;

  /* if conflict resolution is not on then take first one */

  if  conflict-flag = "no" then
      hd(hd(total-rule-stats))-> select-id;
  else
      for rlst in total-rule-stats do
          rlst-->[?identifier ?val-list];
          0->total;
          for val in val-list do
              val + total ->total;
          endfor;
          [~~add-rule-stats [~identifier ~total]]->add-rule-stats;
      endfor;

      /* sorting */
      sort-by-n-num(add-rule-stats,2)->add-rule-stats;

      /* selection */
      last(add-rule-stats)-->[?select-id =];
  endif;
define;


 Evaluation of Decisions */

ine evaluate-trades(data-array,position-days,trades-list,open-col,initial-capital,tra
-costs,running-positions,numcolumns)->trades-report->nrow;

  0->nrow; 0.0->profit-total;0.0->loss-total; 0->daycount;
  initial-capital->capital; length(trades-list)->leng;
  newarray([1 ~numcolumns 1 ~leng ])->trades-report;

  /* a host of 'conflicting trade actions' will have to be assessed
   - that is if there is a new signal before the current trade 'expires'
     what action is to be taken
      (1) if the new signal says the same action then ignore it
      (2) if it has a different action then assess it's past
          predictive rating and if this rating is very high then
          shift to this new rule */

  0->prev-indx;''->prev-action;''->prev-rule; []->temp-trades-list;

  if running-positions="yes" then

      /* construct the new trades-list which has the running positions */
      [~~temp-trades-list ~(trades-list(1))]->temp-trades-list;

      trades-list(1)-->[?indx ?rule-id ?action];
      rule-id->prev-rule-id; length(trades-list)->len-trades-list;

      for tradenum from 1 to len-trades-list do

          trades-list(tradenum)->trade;
          trade-->[?indx ?rule-id ?action];
          if rule-id /= prev-rule-id then
              [~~temp-trades-list ~trade]->temp-trades-list;
              rule-id ->prev-rule-id;
          endif;
```

```
      endfor;
      temp-trades-list->trades-list;
  endif;
  length(trades-list)->len-trades-list;

  for tradenum from 1 to (len-trades-list-1) do
      trades-list(tradenum)->trade; trade-->[?indx ?rule-id ?action];
      indx + 1 ->indx;

      /* the today's open price - the following 'position-days' the number
      of days you want to be in the market */

      if running-positions /= "yes" then
          data-array(open-col,indx) - data-array(open-col,indx + position-days)
              -> change-in-price;
      else
          /* get the index of the next item */
          /* if tradenum < len-trades-list then */
              trades-list(tradenum + 1)->next-trade;
              next-trade-->[?next-index = =];
              ;;; executing the following morning
              next-index + 1 -> next-index;
              data-array(open-col,indx) - data-array(open-col,next-index)
                  ->change-in-price;
      endif;
      (change-in-price/data-array(open-col,indx)) * capital ->appr-dep;
      capital * (0.1/100) -> trans-costs;

      if (indx - prev-indx) > position-days or prev-indx = 0
      or running-positions = "yes" then
          if action = "DOWN" and change-in-price > 0 then
              abs(appr-dep) + capital - trans-costs ->capital;
              'profit'->profit-loss; profit-total+1->profit-total;
              abs(appr-dep) - trans-costs ->trade-value;
          endif;

          if action = "DOWN" and (change-in-price < 0 or
              change-in-price = 0) then
              capital - abs(appr-dep) - trans-costs -> capital;
              'loss'->profit-loss; loss-total+1->loss-total;
              negate(abs(appr-dep)) - trans-costs ->trade-value;
          endif;

          if action = "UP" and change-in-price < 0 then
              abs(appr-dep) + capital - trans-costs  ->capital;
              'profit'->profit-loss; profit-total+1->profit-total;
              abs(appr-dep) - trans-costs ->trade-value;
          endif;

          if action = "UP" and (change-in-price > 0 or change-in-price=0) then
              capital - abs(appr-dep) - trans-costs -> capital;
              'loss'->profit-loss; loss-total+1->loss-total;
              negate(abs(appr-dep)) - trans-costs ->trade-value;
          endif;

          /* in percentage terms */
          profit-total/(profit-total+loss-total) * 100->trades-report(9,nrow);
          /* including transaction costs */
          capital ->trades-report(10,nrow);
          rule-id->trades-report(11,nrow);
         /* Have introduced the new penalty item suggested by Chris
         - this is to take into account the transaction costs*/

          indx->prev-indx; action->prev-action; rule-id->prev-rule;
      endif;
   endfor;
ddefine;



------------------ FUZZY SYSTEM (fuzzy.p) -----------------------------*/

 Defining fuzzy membership functions from the clusters */

 here you take the cluster-ranges and find the min and max value of
ch range - and then stretch each vector to form a "always membership
 type function. Now the range of low becomes from the
eginning of low until the mid-point of medium. The mid-point of medium
 the mid-point of medium - now stretch it until the mid-point of low and
idpoint of high. (you do this for all functions in the middle Stretch high
ntil the mid-point of medium.  also calculate the mid-point. The mid-point
ll be useful in the next section in the assignment of
embership functions. You put the new range in what is called a
uzzy-vector-range */


fine cluster-ranges-to-stretched-vectors(cluster-ranges)->
       fuzzy-vector-ranges;
  []->clusters-with-mid-points; []->fuzzy-vector-ranges;

  /* getting mid-points */
  for cl in cluster-ranges do
      cl-->[?name ?clrange]; clrange(1) ->minval;
      clrange(length(clrange)) ->maxval; /* getting the mid value */
      (minval+maxval)/2.0->midval;
      [~~clusters-with-mid-points [~name [~minval ~midval ~maxval]] ]
          ->clusters-with-mid-points;
  endfor;

  0->count;
  for cl in clusters-with-mid-points do
      count + 1 ->count;
      cl-->[?name ?clrange];
      /* the first item - like low */
      if count = 1 then
          clrange(1) ->minval; clrange(2) ->midval;
          /* getting the mid-point of the next cluster */
          clusters-with-mid-points(count+1)-->[= ?nextrange];
          nextrange(2) ->maxval;
          [~~fuzzy-vector-ranges [~name [~minval ~midval ~maxval]] ]
              ->fuzzy-vector-ranges;
          minval->prev-minval; midval->prev-midval; maxval->prev-maxval;
      endif;

      /* middle clusters */
      if count /=1 and count /= length(clusters-with-mid-points) then
          prev-midval->minval; clrange(2)->midval;
          clusters-with-mid-points(count+1)-->[= ?nextrange];
          nextrange(2) ->maxval;
          [~~fuzzy-vector-ranges [~name [~minval ~midval ~maxval]] ]
              ->fuzzy-vector-ranges;
          minval->prev-minval; midval->prev-midval; maxval->prev-maxval;
      endif;

      /* end cluster */
      if count = length(clusters-with-mid-points) then
          prev-midval->minval; clrange(2)->midval; clrange(3)->maxval;
          [~~fuzzy-vector-ranges [~name [~minval ~midval ~maxval]] ]
```

```
                  ->fuzzy-vector-ranges;
          endif;
      endfor;
enddefine;


define calc-membership-function(fuzzy-vector-ranges,num-bins)->
         membership-functions->binranges;

    /* get the beginning of the range and the end */
    fuzzy-vector-ranges(1)-->[= [?rangeminval ==]];
    fuzzy-vector-ranges(length(fuzzy-vector-ranges))-->[= [== ?rangemaxval]];

    rangemaxval - rangeminval ->total-range; total-range/num-bins->bin-length;
    /* construct binranges list */ rangeminval -> binpos;

   /* NOTE the num-bins +1, so there will be one more elememnt -
      because of the ranges */

    repeat num-bins+1 times
      [~~binranges ~binpos]->binranges;
      bin-length + binpos->binpos
    endrepeat;

    /* now get the membership for each fuzzy variable*/
    0->count;
    for vector in fuzzy-vector-ranges do
        []->single-mem-func;
        count + 1->count;
        vector-->[?name [?minval ?midval ?maxval]];

       /* now you calculate the membership values according to where the
          cluster/vector is - if it is in the beginning or end it is the
          line and slope shape, if it is in the middle it is the slope and
          slope shape */

        rangeminval->pointing-value;

        /* for low or any linguistic term which is in the begining */
        if count = 1 then
            repeat num-bins times
                /* upto the line */
                if pointing-value <=midval then 1->mem-value endif;

                /* calculating the slope */
                if pointing-value >midval and pointing-value < maxval then
                    (1/(maxval-midval)) * (maxval - pointing-value) -> mem-value;
                endif;

                /* zero values range */
                if pointing-value >=maxval then 0->mem-value endif;

                [~~single-mem-func ~mem-value]->single-mem-func;
                pointing-value + bin-length -> pointing-value;
            endrepeat;
            [~~membership-functions [~name ~single-mem-func]]->membership-functions;

            /* Last linguistic term */
        elseif count = length(fuzzy-vector-ranges) then
            repeat num-bins times
                /* zero values range */
                if pointing-value <minval then 0->mem-value endif;

                /* upward slant*/
                if pointing-value <midval and pointing-value >minval then
                    (1/(midval-minval)) * (pointing-value - minval) -> mem-value;
                endif;

                /* 1 value range */
                if pointing-value > midval then 1->mem-value endif;

                [~~single-mem-func ~mem-value]->single-mem-func;
                pointing-value + bin-length -> pointing-value;
            endrepeat;
            [~~membership-functions [~name ~single-mem-func]]->membership-functions;
        else
            /* all middle upward and downward slanting fucntions */
            repeat num-bins times
                /* upward slant*/
                if pointing-value <midval and pointing-value > minval then
                    (1/(midval-minval)) * (pointing-value - minval) -> mem-value;
                endif;

                /* downward slant */
                if pointing-value >midval and pointing-value <maxval then
                    (1/(maxval-midval)) * (maxval - pointing-value) -> mem-value;
                endif;

                /* zero values range */
                if pointing-value <minval then 0->mem-value endif;
                if pointing-value >maxval then 0->mem-value endif;

                [~~single-mem-func ~mem-value]->single-mem-func;
                pointing-value + bin-length -> pointing-value;
            endrepeat;
            [~~membership-functions [~name ~single-mem-func]]->membership-functions;
        endif;
    endfor;
idefine;


fine classify-new-value(fuzzy-membership-functions,binranges,classifyval)
         ->fuzzy-values;

    until found-range = true do count+1->count;

        /* if even the first one is larger than the one to classify */
        /* takes care of the situation where the value to classify is
         lower than even the first element*/

        if count=1 and binranges(count) >= classifyval then
            1->rangeindex; true->found-range;

       /* the situation where the value to be classified is larger than the
        last element */

        elseif classifyval >= binranges(length(binranges))
             then length(binranges) - 1 ->rangeindex; true->found-range;
        elseif binranges(count) >= classifyval then
            true->found-range; count - 1 ->rangeindex;
        endif;
    enduntil;


    for func in fuzzy-membership-functions do
        func-->[?name ?mem-func];
         mem-func(rangeindex) -> round-fuzzy-val;
        [~~fuzzy-values [~name ~round-fuzzy-val] ]->fuzzy-values;
```

```
    endfor;
enddefine;


/* fuzzy reasoning */

define min-max-antecedents(rule,fuzzy-values) ->rule-strength;
    vars num-antecedents count ante ante-val fval active-value active-value-list;
    vars orlist aval; []->active-value-list; 0->count; length(rule) -1 ->num-antecedents;

    until count = num-antecedents do
        count + 1 ->count;
        rule(count)-->[?ante ?ante-val];

        if length(ante-val)=0 then
            /* just ignore because this part of the rule is redundant */

        elseif length(ante-val) > 1 then
            []->orlist;

             for aval in ante-val do

              /* get the correspoing entry from the fuzzy-values
                 eg.  [low 1 ] [medium 0.3   ] [high 0.1  ] */

                fuzzy-values(count)->fval; fval-->[== [~aval ?active-value] ==];
                [~~orlist ~active-value]->orlist;
            endfor;
            last(sort(orlist)) ->active-value;
            [~~active-value-list ~active-value]->active-value-list;
        else
            /* only one value eg. vol[high] */
            fuzzy-values(count)->fval;
            /* in this situation there is only one member in this list */
            hd(ante-val)->ante-val;
            fval-->[== [~ante-val ?active-value] ==];
            [~~active-value-list ~active-value]->active-value-list;
        endif;
    enduntil;

    /* finding the minimum */

    if length(active-value-list) > 0 then
    hd(sort(active-value-list)) ->rule-strength;
    else 0.0->rule-strength;
    endif;
enddefine;

define min-consequents(rule,fuzzy-values,rule-strength)->cp-vector;

    /* the conseqent is the last element in the rule */
    last(rule)-->[?conseq [?conseq-val]];

    /* get the coresponding fuzzy membership definition*/
    memCP-->[== [~conseq-val ?conseq-memb] == ];

    for item in conseq-memb do
        if item > rule-strength then
            conseq-memb -->[??firstbit ~item ??endbit];
            [~~firstbit ~rule-strength ~~endbit]->conseq-memb;
        endif;
    endfor;
    conseq-memb->cp-vector;
enddefine;
```

```
 Do the above operations for each rules and form a matrix - cpmatrix */

fine top-level-reasoning(fuzzy-rules,fuzzy-values)->cp-matrix;

  for rule in fuzzy-rules do
      min-max-antecedents(tl(rule),fuzzy-values) ->rule-strength;
      min-consequents(rule,fuzzy-values,rule-strength)->cp-vector;
      [~~cp-matrix ~cp-vector]->cp-matrix;
  endfor;
ddefine;

 Squeeze the cp-matrix into a vector by taking the maximum over each
lumn */

tine squeeze-cp-matrix(cp-matrix)->final-cp-vector;
  /* finding the num of rows and cols */
  length(cp-matrix(1))->num-cols; length(cp-matrix)->num-rows;

  /* find the maximum over each column */

  until countcol = num-cols do
      countcol + 1->countcol; 0->countrow; []->col-list;
      until countrow = num-rows do
          countrow+1->countrow;
          cp-matrix(countrow)->row;
          [~~col-list ~(row(countcol))]->col-list;
      enduntil;

      /* get the maximum of the column and make elements of final-cp-vector*/
      [~~final-cp-vector ~(last(sort(col-list))) ]->final-cp-vector;
  enduntil;
ldefine;


 Defuzzify using the Centre of Area method */

ine defuzzify(cp-final-vector, uodCP) ->scaler-value;

  for uodval in uodCP do
      count + 1 ->count; ( uodval * cp-final-vector(count)) + prod-total ->prod-total;
  endfor;

  for item in cp-final-vector do
      item + cp-total ->cp-total;
  endfor;
:define;



 --------------- GENETIC ALGORITHM (gri.p) -------------*/

 -- The Population -- is a list of lists - each gene represented by a
 t - each gene has a list of each feature, which in turn can be a
 t of lists.  Structure of members : -

  [ [ oi [verylow low] ]  [ price [increasing] ]
    [ ma-5 [5000 6000] ]


 Initialising the population */

 ine initial-population(num-members,feature-list,feature-struct,seed-population)
```

```
         ->selected-feature-list->population->fuzzy-feature-list;

    /* select the features from the larger list */
    for feature-name in wy-feature-struct do
        feature-list-->[== [?discrete ~feature-name ??feature-sublist] ==];
        if discrete="F" then [~~fuzzy-feature-list ~feature-name]
                ->fuzzy-feature-list;
        endif;
        [~~selected-feature-list [~discrete ~feature-name ~~feature-sublist]]
            ->selected-feature-list;
    endfor;

/* Here you intialise the members - upto the specified amount.
Initialise each member according to whether it is discrete or non-discrete. */

    ;;; taking the seed population

    num-members - length(seed-population) ->num-members;
    for mem from 1 to num-members do

        []->memb;
        for item in selected-feature-list do
            []->member-sublist;
            item-->[?discrete ?feature-name ?feature-sublist];

            /* Put the check that the action element can only have ONE
               element in the initiated member. Have a simple random number
            flipping between the number of elements */

            if feature-name = "action" then
                [~(feature-sublist(random(length(feature-sublist))))]->member-sublist;
            else
                if discrete = "D" or fuzzy-reasoning = true then
                    random(length(feature-sublist) + 1)->ranum;

           /* there is a random chance snapping it depending on the number of
                    sub-features */
                    if ranum-1 = 0 then
                        []->member-sublist
                        /* further reduction of chances */
                    elseif random(2) = 1

                    then
                        [ ~(feature-sublist(ranum - 1))]->member-sublist;
                    else
                        random(length(feature-sublist))->rnd;
                     []->member-sublist;
                    endif;

                elseif discrete = "N" then
                    /* there is a one in three chance to snap it */
                    if random(3) = 1 then []->member-sublist;
                    else
                        feature-sublist-->[?minval ?maxval];
                        maxval-minval->val-range; minval + random(val-range)->minval; max
val - random(val-range)->maxval;

                        if minval < maxval then
                            [~minval ~maxval]->member-sublist;
                        else
                            [~maxval ~minval]->member-sublist;
                        endif;
                    endif;
```

```
              elseif discrete = "F" and fuzzy-reasoning /=true then
                  []->member-sublist;
                  /* initialising the fuzzy values */
                  0->fuz-initial; vars fuzz-feat-count; 0->fuzz-feat-count; 0->fuz-ini
al-total;

                  for fuz-feat in feature-sublist do

                      fuzz-feat-count + 1 ->fuzz-feat-count;

                      /* the fuzzy feature ranges are always 0-1 */
                     /* IF you want to make the GA do less work then
                        you should make the sum of these different values add
                        up to 1 - because that's the way fuzzy memberships
                      are defined */

                      /* the last one */
                      if fuzz-feat-count=length(feature-sublist) then
                          1.0 - fuz-initial-total ->fuz-initial;
                      else
                          false->add-initial; until add-initial = true do
                              (random(11) - 1)/10.0 ->fuz-initial;
                              if fuz-initial + fuz-initial-total <= 1.0 then
                                  true->add-initial;
                                  fuz-initial + fuz-initial-total -> fuz-initial-total

                              endif;
                          enduntil;
                      endif;
                      [~~member-sublist [~fuz-feat ~fuz-initial]]->member-sublist;
                  endfor;
              endif;
          endif;
          [~~memb [~feature-name ~member-sublist] ]->memb;
      endfor;
      [~~population ~memb]->population;
  endfor;
     [~~seed-population ~~population]->population;
:define;



 take each member of the population,
ıd how many points there are in the symbolic-array which have these conditions
ıly the features which don't have an empty list)
ı from those matching points find the number of dependent-y's which
.ch the target-y. Calculate Pc - alpha/Nc */

 A rule look like :-
  [[price-up-down [down]]
   [oi-up-down [notrend]]
   [oi-class [dontknow]]
   [fuzzy-oi [fuzzy-oi [low 0.2] [medium 0.3] [high 0.5]]]
   [action [DOWN]]]



 ine evaluation(population,feature-index,symbolic-array,alpha,
       eval-stindex,eval-endindex)->evaluated-list;

  /* Take each member of the population */

  for memb in population do
```

```
 mem-cnt + 1->mem-cnt; 0->good-y; 0->Nc; 0.0->fitness;0->no-elem-count;

/*Take each element in the data array and */
    0->feat-cnt;
    until feat-cnt= length(memb)-1 do
        feat-cnt + 1->feat-cnt;
        memb(feat-cnt)->feat-in-memb;
        feat-in-memb-->[?feature-name ?sub-features];
        if length(sub-features)=0 then
           no-elem-count + 1 ->no-elem-count;
        endif;
    enduntil;

for arrayindex from eval-stindex to eval-endindex do
    /* take each feature of a member EXCEPT last action feature*/

    true->all-feat-match;

    until feat-cnt= length(memb)-1 do
        feat-cnt + 1->feat-cnt; memb(feat-cnt)->feat-in-memb;
        feat-in-memb-->[?feature-name ?sub-features];

         /* check for fuzzy feature */
        if not(member(feature-name,fuzzy-feature-list)) then

            false->sub-feature-found;
            for sub-item in sub-features do

            /* here the check is to see whether the sub-features
            in the array and of the members match.
                Not to see whether the dependent-y is the same */

                sub-item><''->sub-item;

                if sub-item =
                    symbolic-array(feature-index(feat-cnt),arrayindex) then
                    /* clever mapping here */
                    true->sub-feature-found;
                endif;
            endfor;

            if sub-feature-found = false and
                length(sub-features)/=0 then
                false->all-feat-match;
            endif;
    enduntil;

    /* total matched points - as in features */

    if all-feat-match = true then Nc + 1.0 -> Nc; endif;

    /* here is the important match - features and the target is true*/
    last(memb)-->[= [?rule-action]];
    rule-action><''->rule-action;

    if all-feat-match = true
    and symbolic-array(last(feature-index),arrayindex) = rule-action
    then
        /* [Good one !]=> */
        good-y + 1.0 ->good-y;
    endif;
endfor;
```

```
      if no-elem-count = length(memb)-1 then ;;;    [vol [] oi [] action [DOWN]
       0.00->fitness;                        ;;;    null rules
      elseif Nc > 0.0 then
          (good-y/Nc) - (alpha/Nc) ->fitness;
      else
          0.00->fitness;
      endif;

      [~~evaluated-list [~mem-cnt ~fitness]]->evaluated-list;
  endfor;
ddefine;


fine fuzzy-rule-bases-evaluation
   (population,feature-index,symbolic-array,alpha,threshold,eval-stindex,
   eval-endindex)->fitness;

  /* Here we are going to evaluate a WHOLE Rule Base at a time */

  []->fuzzy-rules;
  for memb in population do
      memb-->[??mlist];
      [~~fuzzy-rules [ID ~~mlist]]->fuzzy-rules;
  endfor;

      mem-cnt + 1->mem-cnt; 0.0->good-y; 0.0->bad-y; 0.0->Nc;
      /*Take each element in the data array and */

      for arrayindex from eval-stindex to eval-endindex do

          []->fuzzy-values; 0->feat-cnt;
          until feat-cnt= length(memb)-1 do
              feat-cnt + 1->feat-cnt;
              [~~fuzzy-values
                  ~(symbolic-array(feature-index(feat-cnt),arrayindex))]
                  ->fuzzy-values;
          enduntil;

          top-level-reasoning(fuzzy-rules,fuzzy-values)->cp-matrix;
          squeeze-cp-matrix(cp-matrix)->final-cp-vector;
          defuzzify(final-cp-vector, uodCP) ->scaler-value;

          if scaler-value < negate(threshold) or scaler-value > threshold
             then Nc + 1 -> Nc;
          endif;

          if sign(scaler-value) = -1.0 and scaler-value < negate(threshold)
          and symbolic-array(last(feature-index),arrayindex) = 'DOWN'
          then
              /* [Good one !]=> */
              good-y + 1.0 ->good-y;

          elseif
              sign(scaler-value) = 1.0 and scaler-value > threshold
          and symbolic-array(last(feature-index),arrayindex) = 'UP'
          then
              /* [Good one !]=> */
              good-y + 1.0 ->good-y;
          endif;
      endfor;

      if Nc > 0.0 then
      (good-y/Nc) - (alpha/Nc) ->fitness;
```

```
        else
          0.0->fitness;
        endif;
enddefine;


/* here you specify the number of members you want to keep -
that is the fittest members - remove duplicating entries -
if there isn't a sufficient number of solutions then you stop at
the total number of different solutions */

define get-fit-members(num-to-keep, ranked-list,population)->fittest-members->fitstats;
    vars mem item fit-from-population count;
    []->fittest-members; []->fit-from-population; 0->count;
    []->fitstats;

    for item in ranked-list do
        [~~fit-from-population ~(population(hd(item)))]->
        fit-from-population;
    endfor;

    for mem in fit-from-population do
        count + 1 ->count;
        if not(member(mem,fittest-members)) and
              length(fittest-members) < num-to-keep
        then
             [~~fittest-members ~mem]-> fittest-members;
             [~~fitstats [~~mem ~~(ranked-list(count))] ]->fitstats;
        endif;
    endfor;
enddefine;


/* It is here that members for mating are chosen via the
the roulette wheel selection strategy. It chooses the members for
mating in proportion to their fitness. */

define select-members-to-mate(popu-size,evaluated-list,popul)
         ->population-toadd;

    vars num-newmembers i total-fitness ranfit total-sofar population-toadd;
    vars mm abs-fitness;
    popu-size -> num-newmembers;

    0->total-fitness; []->population-toadd;
    /* summing the total fitness */
    for i in evaluated-list do
        i(2) + total-fitness ->total-fitness;
    endfor;

    repeat num-newmembers times
        0->total-sofar;
        if total-fitness=0.0 or total-fitness < 0.0
            then abs(total-fitness) -> abs-fitness;
            random(abs-fitness + 0.00001) ->ranfit;
            ranfit * -1.0 ->ranfit;     ;;; becuase random can't use -9 etc
        else
            random(total-fitness)->ranfit;
        endif;

        for mm in evaluated-list do
            if total-fitness = 0.0 then
                [~~population-toadd ~(popul(mm(1)))]->population-toadd;
                quitloop;
          endif;

          if mm(2) + total-sofar >= ranfit then
              [~~population-toadd ~(popul(mm(1)))]->population-toadd;
              quitloop;
          endif;

          mm(2) + total-sofar ->total-sofar;
      endfor;
  endrepeat;
ddefine;


 Here the strategy is to take the selected members, make two copie
 these and crossover them. Construct the new population with the
ttest members plus the new members upto the specified population.

en you make a call to the mutation procedure and change only the
mbers which have been created - that is you keep the best solutions
thout alteration.


fine crossover(num-members,population-toadd,crossover-rate,
       max-changes)->crossed-members;

  /* make 2 identical copies */
   population-toadd->replica1; population-toadd->replica2;

  []->crossed-members; length(replica2)->len-replica2;

  /* strategy is to take each member of replica1 and a random member of
  replica2 and crossover elements */

  for memb1 in replica1 do
      replica2(random(len-replica2))->memb2;

      /* not to take the target y */
      length(feature-struct) -1 ->num-features;

      /* Producing offspring */
      []->new-member1; []->new-member2;
      for i from 1 to num-features do

          /* the feat-names in both members should be the same */

          memb1(i)--> [?feat-name ?sub-feature1];
          memb2(i)--> [?feat-name ?sub-feature2];

          /* so the higher the crossover-rate is lower the chance of
          crossover*/

          if member(feat-name,fuzzy-feature-list) and fuzzy-reasoning /= true
             then
              length(sub-feature1)->fuz-feature-length;

              /* changing the first of the chosen two */

              if random(crossover-rate)=1 then
                  random(fuz-feature-length)->fuz-feat-index;
                  sub-feature2(fuz-feat-index)->fuz-feat-cross;
                  sub-feature1->copy-sub-feat1;
                  fuz-feat-cross->copy-sub-feat1(fuz-feat-index);

                  [~feat-name ~copy-sub-feat1]->new-feat1;
```

```
                  [~~new-member1 ~new-feat1]->new-member1;

              else
                  [~feat-name ~sub-feature1]->new-feat1;
                  [~~new-member1 ~new-feat1]->new-member1;
              endif;

              /* changing the second of the chosen two */
              if random(crossover-rate)=1 then
                  random(fuz-feature-length)->fuz-feat-index;

                  sub-feature1(fuz-feat-index)->fuz-feat-cross;
                  sub-feature2->copy-sub-feat2;
                  fuz-feat-cross->copy-sub-feat2(fuz-feat-index);

                  [~feat-name ~copy-sub-feat2]->new-feat2;
                  [~~new-member2 ~new-feat2]->new-member2;
              else
                  [~feat-name ~sub-feature2]->new-feat2;
                  [~~new-member2 ~new-feat2]->new-member2;
              endif;
          else
              if random(crossover-rate)=1 then
                  [~feat-name ~sub-feature2]->new-feat1;
                  [~~new-member1 ~new-feat1]->new-member1;
              else
                  [~feat-name ~sub-feature1]->new-feat1;
                  [~~new-member1 ~new-feat1]->new-member1;
              endif;

              if random(crossover-rate) = 1 then
                  [~feat-name ~sub-feature1]->new-feat2;
                  [~~new-member2 ~new-feat2]->new-member2;
              else
                  [~feat-name ~sub-feature2]->new-feat2;
                  [~~new-member2 ~new-feat2]->new-member2;
              endif;
          endif;
      endfor;
      [~~crossed-members ~new-member1 ~new-member2]->crossed-members;
  endfor;
 enddefine;
```

```
/* The inputs to this is the maximum number of members you want
to mutate in one go (it takes a random number from this max-number),
the mutation rate, which specifies the chance of changing a
selected member of mutation and the final input is the
population */

define mutation(max-changes,mutation-rate,crossed-members)->muted-members;

    /* There are five basic mutation operators --
    First is picking a new co-ordinate i.e. price []  -->  price [increasing]

    Second is to zap a co-ordiantes i.e. price [ increasing decreasing]->
                                            price []

    Third is changing the value of a co-ordinate i.e price [increasing]
                                                   --> price [decreasing]

    Fourth is to increase the number of co-ordinates i.e. price [increasing]
                                --> price [increasing decreasing]
```

```
Fifth is to decrease the number of co-ordinates
         i.e: price [increasing decreasing notrend]
                             --> price [decreasing]

FOR the ACTION features take only ONE of the coordinates
FOR FUZZY features only increment or decremnt fuzzy values
*/


crossed-members->muted-members;    ;;; making a copy

repeat max-times times
    /* selecting a member randomly */

    random(length(muted-members))->member-indx;
    muted-members(member-indx)->selected-member;

    /* then you select a feature for mutation randomly */

    if random(mutation-rate) = 1 then
        /* to select the sub-feature for mutation*/
        length(feature-struct) - 1 ->n-features;
        selected-member(random(n-features))->feat-to-mutate;
        selected-member-->[??first-bit ~feat-to-mutate ??end-bit];
        feat-to-mutate--> [?feat-name ?sub-feat];

        /* get corresponding values from feature-list */
        feature-list-->[== [= ~feat-name ?sub-feat-in-feat-list] ==];

        /* [fuzzy-oi [fuzzy-oi [low 0.6] [medium 0.5] [high 0.2]]] */

        /* ------------- Fuzzy feature check --------------- */

        /* ONLY MUTATIONS are increasing and decreasing fuzzy values
           HAVE introduced the check to make sure the values
           of the mutated features addup to 1
        */

        length(sub-feat)->fuz-feat-length;
        if member(feat-name,fuzzy-feature-list)
        and fuzzy-reasoning /= true then
            /* check for price [] type ones */
            []->new-feature;

        /* Now you mutate the sub-feature with random filtering
            along the way */

        /* The only thing to note is that there is no check
        to see whether the mutation ( the attributes for mutation)
            are the same or not from the ones you already have */
             random(2)->rndnum;
            /* for selecting the type of mutation action */
            if length(sub-feat) = 0
            then
                /* pick a random number from the length of feature-list
                and add items */
                []->selected-feature; 1->rnd;     ;;; only one
                []->already-in; /* to check against putting same item
                twice*/
                for item from 1 to rnd do
                    random(length(sub-feat-in-feat-list))->rnd2;
                    sub-feat-in-feat-list(rnd2)->item-to-add;

                    if not(member(item-to-add,already-in)) then
                        [~~selected-feature ~item-to-add]
                            ->selected-feature;
                        [~~already-in ~item-to-add]->already-in;
                    endif;
                endfor;


                [~feat-name [~~selected-feature]] ->new-feature;
                [~~first-bit ~new-feature ~~end-bit]->mutated-member;
                /* put the new member in muted-members*/
                mutated-member->muted-members(member-indx);

            elseif rndnum= 1 and feat-name/= "action" then
                /* zapping co-ordinates */
                [~~first-bit [~feat-name []] ~~end-bit]->mutated-member;
                mutated-member->muted-members(member-indx);

            elseif rndnum = 2 then
                /* change co-ordinates */
                sub-feat->feat-to-change;
                repeat rnd3 times
                    sub-feat-in-feat-list(random(length(sub-feat-in-feat-list)))
                        ->item-change;

                    if not(member(item-change,feat-to-change))  then
                        item-change->feat-to-change(random(length(feat-to-change)));
                    endif;
                endrepeat;

                feat-to-change->selected-feature;
                [~feat-name [~~selected-feature]] ->new-feature;
                [~~first-bit ~new-feature ~~end-bit]->mutated-member;
                mutated-member->muted-members(member-indx);

            /* Have safeguards against mutating the "action" bit
                unlawfully */
            elseif rndnum = 4 and feat-name /= "action" then

                 /* decrease the number of co-ordinates */
                /* put the new member in muted-members*/
            endif;
        endif;
    endif;
endrepeat;
enddefine;



;;; This is to compare GRI rules with existing Expert or Fuzzy rules to add
;;; to the Expert or Fuzzy rule bases
;;; select the rules which are most disimilar to the Expert or Fuzzy rule
;;; bases for addition

define comparing-rules(num-to-select,rule-base,gri-rules)->gri-rules-to-add;
    vars g-rule r-rule count g-name r-name g-value r-value differences;
    vars diff-list g-count identical-rule item diff-count diff-count-list;
    vars ncount grinum;
    []->diff-count-list;[]->gri-rules-to-add;

    if length(rule-base(1)) /= length(gri-rules(1)) then
        [Rule Bases have a different lengths ]=>
    else
        0->g-count;
        for g-rule in gri-rules do
```

```
            g-count + 1-> g-count; []->diff-list;
            for r-rule in rule-base do
                0->differences; 1->count; ;;; ignore id
                until count = length(g-rule) do
                    count + 1 ->count;
                    g-rule(count)-->[?g-name ?g-value];
                    r-rule(count)-->[?r-name ?r-value];
                    endif;

                    if g-value /= r-value then
                        differences + 1 ->differences;
                    endif;
                enduntil;
                [~~diff-list ~differences]->diff-list;
            endfor;

            ;;; check if there is an identical rule in the rule-base

            false->identical-rule;
            for item in diff-list do
                if item = 0 then true->identical-rule; endif;
            endfor;

            if identical-rule = true then 0->diff-count;
            else 0->diff-count;
                for item in diff-list do
                    diff-count + item ->diff-count;
                endfor;
            endif;
            [~~diff-count-list [~g-count ~diff-count]]->diff-count-list;
        endfor;
        sort-by-n-num(diff-count-list,2)->diff-count-list;
        rev(diff-count-list)->diff-count-list;

        if num-to-select <= length(gri-rules) then
            0->ncount;
            until ncount = num-to-select do
                ncount + 1 ->ncount;
                diff-count-list(ncount)-->[?grinum =];
                [~~gri-rules-to-add ~(gri-rules(grinum))]->gri-rules-to-add;
            enduntil;
        else
            [num to select is too large]=>
        endif;
    endif;
enddefine;


define go(cycles,popu-size, fuz-thresh,feature-list,feature-struct,symbolic-array,eval-s
tindex,eval-endindex,seed-population,record,record-file)->final-population;
    vars iteration;
    vars mute-max-changes mute-numbers record-list;
    []->record-list;
    initial-population(popu-size,feature-list,feature-struct,seed-population)
        ->selected-feature-list->population->fuzzy-feature-list;
  [Initial population]=>
  population==>
    /* The main GA loop should start here */

    for iteration from 1 to cycles do
        [Cycle Number ~iteration]=>

        []->ranked-list;[]->evaluated-list;[]->fittest-members;
        []->fitstats;[]->population-toadd;[]->crossed-members;
        []->muted-members;

        /* Problem in the evaluation - that's why the solutions are lost */

        if fuzzy-reasoning = true then
            ;;; alpha = 10
            fuzzy-evaluation(population,feature-index,symbolic-array,10,fuz-thresh,eval-s
tindex,eval-endindex)->evaluated-list;
        else
            ;;; alpha = 10
            evaluation(population,feature-index,symbolic-array,10,eval-stindex,eval-endin
dex)->evaluated-list;
        endif;

        rankmembers(evaluated-list)->ranked-list;

        rev(ranked-list)->ranked-list;
        /* because the sorting procedure puts from low to high */


        /* the number here is the number of the fittest members you want to keep */

        get-fit-members((popu-size/3), ranked-list,population)->fittest-members->fitstats
;

        fittest-members->datafile('good-popu');

         if record = "yes" then
           [~~record-list CYCLE ~iteration ~fitstats ~ranked-list]->record-list;
         record-list-> datafile(record-file);
         endif;

        select-members-to-mate(popu-size,evaluated-list,population)
            ->population-toadd;

        crossover(popu-size,population-toadd, 2, 1)
            ->crossed-members;
       intof(popu-size/2)->mute-max-changes;

        mutation(mute-max-changes,1,crossed-members)->muted-members;

    []->mute-population; []->fit-population;

    /* trim down the list to the require number */
     datafile('good-popu')->good-popu; length(good-popu)->fit-nm; popu-size - fit-nm ->mu
te-numbers;

    for x from 1 to mute-numbers do
        [~~mute-population ~(muted-members(x))]->mute-population;
    endfor;

    [~~good-popu ~~mute-population]->population;
    endfor;
enddefine;
```