An Intelligent Decision Support System for Automated Medication Review

By

Ivan Karl Bindoff, BComp

A dissertation submitted to the
School of Computing
In partial fulfilment of the requirements for the degree of

Bachelor of Computing with Honours

November, 2005

University of Tasmania

I, Ivan Karl Bindoff, declare that this thesis contains no material which has been accepted for the award of any other degree or diploma in any tertiary institution. To my knowledge and belief, this thesis contains no material previously published or written by another person except where due reference is made in the text of the thesis.

Ivan Karl Bindoff

Abstract

Knowledge Acquisition techniques are not historically designed for domains where the expert knowledge being modelled is inconsistent and poorly defined. This study explores the applicability of the MCRDR technique of knowledge acquisition towards a domain of this nature, medication reviews. Through this experimentation it is also sought to improve the quality of service that the medication reviewers can provide to their patients, by reducing the incidence of missed classifications.

To facilitate this study a Knowledge-Based System was developed that employed the MCRDR method and which was capable of both being instructed in the domain of medication review through its routine use by the expert, and acting similarly to the expert when producing its classifications based on genuine medication review cases.

It was found that there was a high incidence of missed classifications by the expert which were automatically repaired by the system, and it was also shown that the incidence of missed classifications reduced dramatically as the systems knowledge of the domain grew more complete. It was also found that the inconsistent nature of the expert's knowledge of the domain did not appear to significantly affect the functioning of the system, with none of the tests performed indicating any deviation from what would be expected in a normal MCRDR system.

Acknowledgements

It's been a big year, and I've worked hard to produce this slim volume, but it was made easy for me by being surrounded by and supported by a wonderful group of people. These lovely specimens of man-kind can be easily divided into two core groups, those who directly helped me produce the thesis, and those who directly helped me procrastinate.

Of those who helped me produce the thesis obviously my supervisor, Byeong Ho Kang, is key. He always had a good suggestion up his sleeve, and a solution to whatever problem arose so a hearty thankyou goes to him. Apart from Byeong was the friendly group of Pharmacists at UMORE, a big thankyou to Gregory Peterson, Omar Hassan, Shane Jackson and Diep Le, with special thanks going to Peter Tenni for being my personal medication review slave (a.k.a. expert), if only for a while. On top of this thanks must go to my worthy proofreaders, Jamie, Mum and Dad, with the latter two also requiring thanks for all that general Mum and Dad style stuff they've done as well.

The list of people who helped me procrastinate is vast indeed, so I shall try and keep their individual salutations brief :-

- Vanessa, my wonderful girlfriend, your presence and comfort was greatly appreciated throughout the year. You've kept me sane in so many ways.
- Jamie, you're a good mate who made sure I had fun outside of the honours rooms occasionally.
- David, Michael, Anthony, Tristan, Hallzy, Dima, Lambert, Emma, Simon, Duncan, Hamid, Bruno and Steve a huge thanks go out to all of you for being such a lovely bunch of coconuts. I've had far too much fun hanging around the honours rooms with all you lot, and I'm quite sure I could have produced something far more profound without the constant footy-o'clocks and general procrastination you offered me. But then, what fun would that have been?

For my grandfather, who was interested until the end.

Table of Contents

List of Figures and Tables

1 Introduction

Sub-optimal drug usage is a serious concern in both Australia and overseas (Bates et al. 1995; Peterson 2004), particularly with the elderly, resulting in at least 80,000 hospital admissions annually - approximately 12% of all admissions and reflecting a cost of about $400 million annually (Peterson 2002). In an attempt to counter this problem Australian government and pharmaceutical officials brought about both the Residential Medication Management Reviews (RMMRs) and the Home Medicines Review (HMR) schemes. The crux of these schemes is that pharmacists are now remunerated via a nationally funded program for conducting medication reviews for nursing home, hostel and selected high risk patients (Peterson 2002). Medication reviews are seen as an effective way to improve drug usage, with medication reviews conducted by pharmacists being found to be better than those conducted by general practitioners, leading to more medication changes and higher savings in medication costs (Anon. 2001; Zermansky et al. 2001). The remuneration offered in the Medication Review schemes is shown to be effective in increasing interest from pharmacists in undertaking medication review roles (Peterson 2002). However, many community-based pharmacists are still unwilling to undertake this new role, citing reasons including fear of error and a lack of confidence, while the quality of service provided by those who have is by no means standardised (Rigby 2004). Further to this, there is anecdotal evidence suggesting an inadequate number of medication reviews are being done, with all residents of a nursing home or aged accommodation facility being entitled to at least one medication review per year. Yet with over 220,000 residents of these facilities in Australia (Australian Bureau of Statistics 2003) and each case taking “between 2 and 4 hours” (Tenni 2005), there are simply not enough pharmacists offering medication reviews to cope with the demand.

Reflecting upon this it seems desirable to produce software solutions which provide both efficient and constructive handling of medication reviews and intelligent

decision support tools to the pharmacist. In fact, some have gone so far as to suggest a focus on Information Technology based solutions (Avorn 2001). A system of this nature is termed an Expert or Knowledge-Based System, and should help to improve consistency and quality of service, reassure reviewers of their conclusions and if possible speed up the process of performing the review. Furthermore, the system should be easily and naturally maintained and alterable, since new information is being added to the domain on a regular basis. To date it is not believed that such a system exists commercially, with medication review software focusing mostly on data-storage and reporting facilities, providing no heuristic support or reasoning capabilities (Kinrade 2003). Even in research, systems of this nature are fraught with problems, classically being extremely difficult to keep reliably maintained for long periods of time over extensive domains (Compton & Jansen 1989). The closest attempt at a heuristic Knowledge Based System in this vein was developed by Classen *et al.* for a hospital in the U.S. showing success in detecting sub-optimal drug usage in the hospital environment, but this system is not considered suitable for the broader task of assisting in the general role of medication review (Jha et al. 1998). However, a large, maintainable Knowledge Based System has been contrived in a related domain in the form of LabWizard, a pathology reporting system. In a personal communication from Pacific Knowledge Systems (PKS) this system was shown to demonstrate "over a 29 month period, over 16,000 rules have been added and 6,000,000 cases interpreted with a correct classification rate in the order of 99%" (PKS 2005). It achieved these figures using the Ripple-Down Rules approach to building Knowledge Based Systems, which allows maintenance to become an unobtrusive part of the process of using the system.

Unfortunately, the domain of medication reviews is considered difficult to model, as the knowledge experts in the field have tends to be poorly structured, as the area is relatively new and insufficiently evolved (Rigby 2004). This results in the process taking longer than necessary, and the reviewer often missing comments relevant to the case. Furthermore, the experts' knowledge cannot be expected to be complete, as the domain operates on data from multiple disciplines (Pathology, General Practitioner, Pharmacy) and yet the reviewer is usually only completely trained in the Pharmacy portion of this field. Knowledge Based System techniques have been designed to handle steadfast, well defined sets of knowledge, and have historically

not been well suited to poorly structured or dynamic sets of knowledge. Newer techniques such as Case Based Reasoning and Ripple Down Rules are seen to be more capable of representing this kind of knowledge, although it remains to be tested.

2 Background

When building a system such as the one developed in this study it is almost invariably possible to separate the background into two distinct parts; the Domain section and the Computer Science section. Subsequently, this chapter has been split into the two parts mentioned above, with a section describing Medication Reviews and a section describing Knowledge Based Systems.

2.1 Medication Reviews

It is well documented that inappropriate drug prescription and underutilisation of drugs is a persistent and significant problem in today's society (Peterson 2004). Estimates place sub-optimal drug usage as resulting in at least 80,000 hospital admissions in Australia annually, which accounts for approximately 12% of all admissions. Figures like these reflect a cost of about $400 million per year, yet about half of these admissions are considered to be avoidable (Roughead 1999; Roughead et al. 1998; Stanton et al. 1994). Perhaps even more alarming is the fact that there is an overwhelming amount of literature suggesting that elderly (65 years and over) patients are at particularly high risk (Peterson 2002), including suggestions that:-

1. comprehensive medication reviews conducted by pharmacists identify potential or actual drug-related issues in at least 80% of elderly patients;
2. up to 46% of drugs prescribed for the elderly are either inappropriate or unnecessary;
3. inappropriate prescribing of drugs such as non-steroidal anti-inflammatory drugs (NSAIDs) and benzodiazepines is common among community-dwelling older people and persists over time;
4. drug-related problems are often significant and have been associated with at least 19% of hospitalisations of the elderly;
5. adverse drug-related events account for at least 10% of hospital emergency department presentations in elderly patients;

6. drug-related problems may contribute to up to 20% of all hospital deaths in the elderly; and
7. there is insufficient treatment of many common medical conditions in elderly patients, including heart failure, depression, and atrial fibrillation.

(Peterson 2002)

Being aware of these issues has prompted the Australian government to act, initiating the Home Medicines Review scheme (HMR) and earlier still (1997), the Residential Medication Management Reviews (RMMRs) scheme in which pharmacists are now remunerated via a nationally funded program for conducting medication reviews (Peterson 2002). However, it is known that despite Residential Medication Management Reviews (RMMRs) being introduced in 1997 they still do not have a conceptual model for delivery, which has resulted in a wide range of differing qualities of service being provided (Rigby 2004). Furthermore, there is considered to be an undersupply of accredited pharmacists[1] to perform medication reviews (Tenni 2005) with over 220,000 residents requiring annual reviews. This is further compounded by an estimated undersupply of around 3000 pharmacists in the next decade (Gadiel 2003). Considering all these factors, it would be reasonable to expect an increased demand for tools with which to more efficiently carry out Medication Reviews, thus providing faster reviews and a more consistent Quality of Service (QoS). It is also anticipated that if these software tools could be made to provide reliable decision support features they might encourage more pharmacists to take up the role of medication reviewer, with many pharmacists citing confidence as a primary reason not to perform medication reviews (Rigby 2004).

2.1.1 Existing Process

To perform a medication review, Pharmacists assess potential Drug Related Problems (DRPs) and Adverse Drug Events[2] (ADEs) in a patient by examining various patient records, primarily their medical history, any available pathology results, and their drug regime (past and current) (Tenni et al. 2005).

[1] pharmacists require accreditation from the Australian Association of Consultant Pharmacists in order to be remunerated for medication reviews (AACP 2005)

[2] defined by the World Health Organisation as being “an injury resulting from medical intervention related to a drug.” (Bates et al. 1995)

The expert looks for a variety of indicators between the case details provided checking for known problems, such as an; Untreated Indication – where a patient has a medical condition which requires treatment but doesn't have the treatment; Contributing Drugs – where a patient has a condition and is on a drug which can cause or exacerbate said condition; High Dosage – where a patient is potentially on a too high dosage because of a combination of drugs with similar ingredients; Inappropriate Drug – where a patient is on a drug that is designed to treat a condition they don't seem to have; and many others besides. Once these indicators have been identified a statement is produced explaining each problem, or potential problem, and often what the appropriate course of action is.

2.1.2 Previous Work

Commercial packages are available to assist in the process of performing MR reports, with a 2003 report reviewing Domiciliary Medication Management Review (DMMR) software identifying products including Mediflags and Miracle MMR, both stand-alone medication review software packages. Other packages identified include Pharmcare and the Health Reference Disc which are modules of a broader pharmaceutical software package. These packages provide data modelling for medication review report generation and modification, but provide no higher level services such as attempting automated expert classification of the cases they store (Kinrade 2003). Another commercial product available in this field, but not reviewed by Kinrade is Cognicare. Cognicare makes attempts to add decision support features by including alerts when certain conditions are met. Cognicare Solutions Ltd. Website states: "CMMS [Cognicare Medication Management System] will accurately analyse the many permutations and combinations of your clients medical conditions and medications: dose checks, side effect profiles, age and sex appropriateness of medications, drug/drug interactions, drug/condition interactions, etc" (Cognicare Solutions Ltd. 2004). The complication with this approach is that it will return many erroneous results, as it does not take into consideration other circumstances. For example, if a patient was suffering from Dizziness it might flag 3 or 4 of the patient's drugs as potentially being the cause of this symptom, while not considering that the patient has Meniere's disease which causes periods of dizziness. This problem is implicit in the techniques Cognicare Solutions Ltd. employed to provide their level of decision support, in which pre-defined basic relationships are

checked from a vast database of possible problems. That is, if they're on a drug, Cognicare will supply a comprehensive list of every possible adverse effect for that drug, but there is no secondary level of checking. This approach is fraught with perils, since more complicated relationships cannot be defined (For example, if a given drug only causes a given problem in patients over the age of 75 who have asthma?). So, not only will this approach flag many erroneous problems in an attempt to "cover its back", it is also likely to miss important problems because it does not have a model in place to handle that level of checking. To solve these problems a more advanced knowledge based approach is required.

An attempt at building a Knowledge-Based System (see *2.2 Knowledge Based Systems* for more information) capable of expert classification in a similar domain was made in the United States by Classen *et al.*, using standard rule-based techniques across patients in a hospital in Utah. This system was compared against the two major manual types of medication review used at that hospital, stimulated voluntary report and chart review, but looked only for significant ADEs, and not more general DRPs. Stimulated voluntary reporting is where nurses and pharmacists on study units were asked daily if any ADEs had occurred, while chart review is where an intensive review is undertaken on each case by a certified pharmacist. Obviously, the chart review identified significantly more ADEs than the stimulated voluntary reporting. With only 52 rules the Classen system was found to detect significantly more ADEs than stimulated voluntary reporting (275 and 2 potential vs. 23 and 61 potential), but still less than a full chart review which managed 398 ADEs and 23 potentials (Jha et al. 1998). This system was successful because it managed a significantly higher rate than would otherwise have been determined, since full chart reviews are not routine, but it would be expected a system that could be more easily maintained - hence gaining a larger rule base - might be more successful still. Furthermore, the system developed by Classen *et al.* was focused on a relatively small domain of high risk incidents (Jha et al. 1998) which is reflected in the fact it found a much lower ratio of ADEs vs. potential ADEs than the full chart review. A more maintainable approach might allow for easier expansion into the broader domain of DRPs as the complications involved in adding new rules are reduced.

2.2 Knowledge Based Systems

Knowledge Based Systems (KBS), also known as Expert Systems (ES) are systems which are designed to perform a task that would otherwise require a human expert, and are a subset of the broader field of Artificial Intelligence (A.I.). KBSs are generally built to enhance quality of service by having an "expert" more readily available to workers when human experts are in short supply, or to assist a human expert by hopefully making their decisions more consistent. KBS's are usually applied in situations where a more heuristic approach to classification is required and conventional software engineering techniques cannot provide the solution. Similar to KBSs are Decision Support Systems (DSS). DSS are systems which assist a human expert in making decisions, but do not by definition need to represent human-like intelligence, although they often do. For example a simple DSS might simply flag potential problems it identifies by traversing a complete problems database. However, for the remainder of this study the term DSS will be treated as having some degree of intelligence in order to provide its support.

2.2.1 Applications for Knowledge Based Systems

KBSs have been applied to a diverse range of problem domains, including, but not limited to; medicine, law, geology, air traffic control, finance and of course computing. However, attempts are usually only made to model a small sub-section of any domain with a KBS, due to the explosion of knowledge that must be represented in order to branch out further into the domain. Various approaches have been taken in attempting to build KBSs which are capable of more general knowledge, rather than domain specific knowledge. One such attempt is the CYC project which is trying to model common sense by building an extraordinarily large knowledge base. This project was first commenced in 1984 and is still underway today (Cycorp 2005; Lenat 1995); although opinions are mixed on how successful it might be (Compton & Jansen 1989).

2.2.1.1 KBS in Medicine

KBS techniques have long been considered in the medicine domain, with many of the earliest such systems being health-based, including Mediphor which screens for drug-drug interactions, INTERNIST which diagnoses complex internal problems (Miller, Pople & Myers 1982), MYCIN (Buchanan & Shortliffe 1984) which is discussed

further later, GARVAN-ES1 which provides diagnostic reports for a pathology lab in the Thyroid domain (Jansen & Compton 1988), PEIRS a broader application of GARVAN-ES1 (Preston, Edwards & Compton 1994), and Thorask which assists in optimizing the triage, diagnosis, and management of non-traumatic chest pain (Federhofer 2005). A range of medical expert systems can be seen at http://www.computer.privateweb.at/judith/name_3.htm (Federhofer 2005).

2.2.2 Techniques

A broad range of techniques have been developed and employed in the development of KBSs and in the acquisition of knowledge. A sample of popular techniques is included below.

2.2.2.1 Traditional Rule Based

Rule Based KBSs are the bread and butter of KBSs, although what exactly "Rule Based" means is debatable. Generally speaking, a Rule Based KBS is built upon a list of IF-THEN rules, which are applied over simple predicate logic constructs which are treated as facts to the system. An example of a simple Rule Base can be seen in Figure 2-2.1, which defines the relationships between parents and children. A list of facts that might be defined before inferencing through the rule base are shown in Figure 2-2.2, these facts would potentially result in the classification of grandparent (Bob, Mary), grandparent(Bob, Charlie), and sibling(Mary, Charlie). However, the classifications that the system would actually make depend on how the system handles the rules. For example it might only fire on the first possible rule, or it may continue through the rule list and fire on all rules that match the facts.

```
IF parent(x, y) THEN child(y, x).
IF child(x, y) THEN parent(y, x).
IF child(x, z) AND child(y, z) THEN sibling(x, y).
IF parent(x, y) AND parent(z, x) THEN grandparent(z, y).
```

```
parent(Joe, Mary).
parent(Joe, Charlie).
parent(Bob, Joe).
```

Forward Chaining

The Figures shown above are an example of Forward Chaining. Forward Chaining KBSs are ones which perform their inferencing over a fully defined set of facts and assume all facts are known before beginning the process.

Backward Chaining

Backward Chaining KBSs contrast to forward chaining by assuming little or no information is known to begin with, and ask questions of the user to learn new facts.

Combined

It is possible to create KBSs which combine both forward and backward chaining styles of inferencing, where rules have additional information defining whether they are forward or backward chaining. A classic example of this kind of system is the MYCIN system which was developed in the 1970-1980s as part of the Stanford Heuristic Programming Project. This system was used to determine which antimicrobial (or antibiotic) should be used for patients with varying bacterial infections. It had a very complete Backward Chaining inference engine, which was not only capable of asking appropriate questions to classify the problem, but which was capable of providing explanations to the user as to why it was asking particular questions or coming up with particular classifications, while only relatively few rules were restricted to forward-chaining mode (Buchanan & Shortliffe 1984).

Rule Based Shortcomings

It was commonly observed among those responsible for maintaining a traditional rule-based ES that the task of continued maintenance of the KB was actually more complex than the initial development of the system in many ways (Compton & Jansen 1989). A classic example of this is the XCON system, designed to analyse the technical correctness of a customer's computer order and provide guidance as to the actual assembly of this order. After four years in routine use it still involved the employ of four full-time knowledge engineers, although this was partly due to the constantly expanding nature of the domain (Bachant & McDermott 1988). This maintainability issue was largely brought about by the basic structure of the rule based expert system, because when new knowledge is added to an existing KB, it must be extensively manipulated in order to maintain the validity and structure of the

existing knowledge, subsuming and conflicting with the existing rules (illustrated in Figure 2-2.3) (Compton & Jansen 1989). There have been attempts at developing tools which check for subsumption and confliction in a KB (Suwa, Scott & Shortliffe 1982), but these tools are incapable of detecting instances where rules which have no logical overlap still satisfy a single data profile (Compton & Jansen 1989). This problem lead Buchanan *et al.* to observe, "Knowledge acquisition is a bottleneck in the construction of expert systems" (Buchanan et al. 1983).

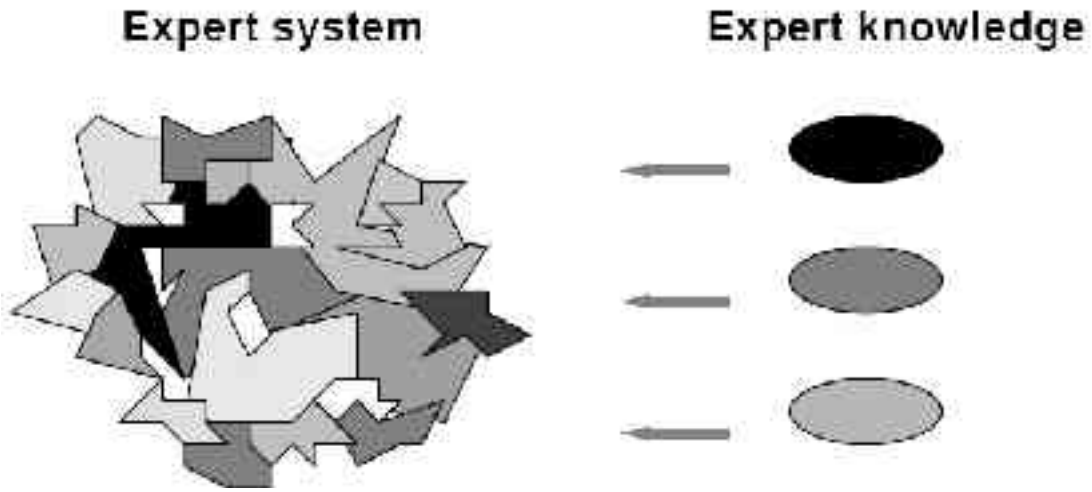


Figure 2-2.3: The difference between knowledge expressed by the expert, and the knowledge as it must be represented in a standard KB (Compton & Jansen 1989)

2.2.2.2 Fuzzy Logic

Fuzzy Logic is another commonly used technique in the development of KBSs. It is basically the addition of confidence factors and degrees of firing to the standard rule-based approach. In short, the rule will fire to a degree calculated based on how confident the system is that the rule should fire (Zadeh et al. 1974). This technique is particularly suited to problems where it makes sense for a rule to fire to different degrees. For example steering a car, which could be represented by the simple rule set shown below in Figure 2-2.4. The steer-left or steer-right classifications will fire to a degree based on how much the system believes the statement "veering-right" or "veering-left". So, if the car is veering heavily to the right the system will try to steer left quite hard. If the car is only slightly off course to the left, it will only steer slightly to the right.

```
IF veering-right THEN steer-left.
IF veering-left THEN steer-right.
```

is possible, but not guaranteed. Different confidence factors can be applied to the rules under different circumstances. For example, the system may be only 20% certain that a classification would be true under some set of facts. However, when another set of facts is added to the facts list that confidence may be increased or decreased based on the new evidence. This kind of classification can be very useful for diagnosis style problems such as the one considered in this study, with the system being varying degrees of "sure" that any particular diagnosis is correct, as was used in the MYCIN system mentioned earlier. Fuzzy reasoning has also been applied to Ripple Down Rule and Multiple Classification Ripple Down Rule (see *2.2.2.5 Ripple Down Rules*) style classifiers, but the implementation issues are non-trivial and will not be considered in this preliminary study (Park, M. et al. 2003).

2.2.2.3 Machine Learning

Machine Learning (ML) is a method commonly considered in the development of KBSs, and has been successfully employed to this end in many notable instances (Aha, Kibler & Albert 1991; Cost & Salzberg 1993; Quinlan 1993). It is specifically concerned with the task of Knowledge Acquisition.

There are many different approaches to ML, but the broad principle is to apply an algorithm to pre-classified data which can extract some form of knowledge which provides future classifications. Some of the more popular techniques include Instance (or Case) Based learners, Artificial Neural Networks, Bayesian Classifiers, and Decision Trees (Witten & Frank 2000).

Obviously, the fact that ML techniques are entirely automated is one of their greatest selling points, but it can also be their greatest flaw, as they often require reasonably simple datasets, and almost invariably need large volumes of pre-classified data to learn from, a problem which makes them unsuitable for many domains including the one being considered in this study. Furthermore, each technique tends to have its own limitations, which will be discussed independently for each section below.

Instance Based Learners

Instance Based Learners, also known as nearest-neighbour methods, attempt to plot classification data in graphical space, then when new inputs are applied they are plotted onto the graph and given the same classification as the nearest pre-classified point. There are many methods employed to enhance the capabilities of this type of learner, but they are generally considered to be unsuitable for datasets that do not have simple sets of discrete values and a small number of possible classifications, making them unsuitable for the domain being considered in this study, which has large and relatively complicated sets of data to model (Aha, Kibler & Albert 1991).

Artificial Neural Networks

Artificial Neural Networks (ANNs) are inspired by the observation that biological learning systems are built from very complex webs of interconnected neurons. It is an entirely automated approach with no human intervention or assistance, but can take a very long time to learn, and the knowledge they model can be extremely difficult to follow. They are considered to be well-suited to problems where the training data corresponds to noisy, complex sensor data, such as inputs from cameras and microphones, as well as problems for which more symbolic representations are often used (Mitchell 1997). Unfortunately they are not well suited to problems with many non-discrete attributes, as their inputs must be entirely numerical in nature, and are noted to be remarkably slow to learn from complex data sets.

Bayesian Classifiers

Bayesian classifiers were inspired by the work on probabilities undertaken by Thomas Bayes in the 1700s. They are a relatively simple idea, where the probability of each classification given the input data is determined based on the likelihood of it happening in the training data with those inputs. It is really only suitable for making very basic predictions across reasonably simple datasets.

Decision Trees

Decision Trees are an approach where a tree structure is built up which can be followed to determine the classification of the data. The tree is built upon training data (pre-classified instances), basically by considering each attribute as a node, and their possible values as branches. Information theory is employed to produce a more efficient tree. Due to the nature of this structure it is generally not suitable for non-discrete values, such as continuous numbers, although techniques have been developed to attempt to classify this kind of data which this method (Quinlan 1993).

It should be mentioned that Decision Trees can be easily converted into a standard rule-based representation, and only slightly less easily converted from a rule-based representation into trees (Witten & Frank 2000). This is because the Decision Tree representation of knowledge is essentially just a different way of structuring the same kind of knowledge.

2.2.2.4 Case Based Reasoning

Case Based Reasoning (CBR) is a technique that employs past experiences (cases) to attempt to classify a current problem. The classification is usually then tested for validity using logical reasoning methods (Kolodner 1991). For example, a doctor might recall that when presented with a patient in the past suffering from the symptoms [very runny nose, sore eyes, and sore throat] they were found to have influenza. This would trigger the doctor to apply a few simple tests to further reinforce this hypothesis, such as checking the throat, ears and lungs of the patient. If, when checking the throat of the patient, the doctor notices white lumps around their tonsils the doctor might revise their conclusion to be that of tonsillitis based on this new symptom (Aamodt & Plaza 1994).

The concept is consistent with concepts that psychologists have observed in the problem-solving people do. It was noted that although people are good at using analogs (past experiences) to solve new problems, they are not necessarily good at remembering which ones to use. Computers are fundamentally good at remembering things, so it is assumed that this approach will show success in improving human decision making when coupled with the memorisation capabilities of computers (Kolodner 1991).

CBR follows a cyclic process which is sometimes referred to as the “four RE’s” (Aamodt & Plaza 1994) and can be seen in Figure 2-2.5:

1. **RETRIEVE** the most similar cases;

   During this process, the CB reasoner searches the database to find the most approximate case to the current situation.

2. **REUSE** the cases to attempt to solve the problem;

   This process includes using the retrieved case and adapting it to the new situation. At the end of this process, the reasoner might propose a solution.

3. **REVISE** the proposed solution if necessary;

   Since the proposed solution could be inadequate, this process can correct the first proposed solution.

4. **RETAIN** the new solution as a part of a new case.

   This process enables CBR to learn and create a new solution and a new case that should be added to the case base.

(Aamodt & Plaza 1994)

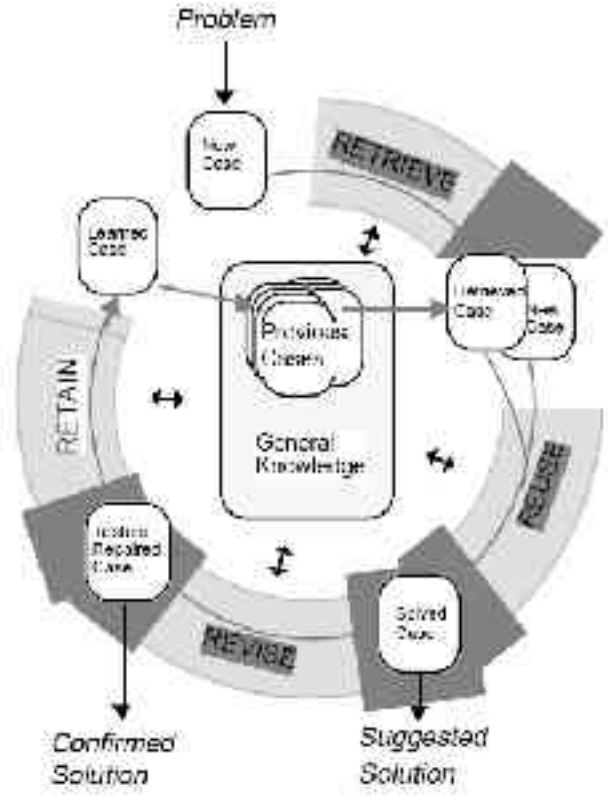


Figure 2-2.5: The CBR cycle (Aamodt & Plaza 1994)

The first system implemented as a Case-Based Reasoner was CYRUS, developed by Janet Kolodner in the early 1980s. CYRUS was based on Schank's dynamic memory model and Memory Organisation Packets (MOP) theory of problem solving and learning (Schank 1982). It was essentially a question-answering system, with knowledge of the various travels and meetings of a former US Secretary of State (Aamodt & Plaza 1994).

The Problem with Case Based Reasoning

Despite the excellent foundational principles behind CBR it still remains relatively unpopular for the development of KBSs today, mostly due to one small failing. When a CBR system is seeking to retrieve similar cases from its case-base, what exactly should it retrieve? For reasonably complex case models it becomes very difficult to summarise what a "similar" case is, because the system is given no concept of what comprises a similarity for any given classification it might make. In the same vein, when the case is being revised to provide a new classification it is not clear which of the differences between this case and the one it suggested are significant. Addressing this issue is the Ripple Down Rules technique.

2.2.2.5 Ripple Down Rules

Ripple Down Rules (RDR) is an approach to building KBSs that allows the user to incrementally build the knowledge base while the system is in use, with no outside assistance or training from a knowledge engineer (Compton et al. 1993). It generally follows a forward-chaining rule-based approach (although backward chaining style behaviour is still possible with this approach) to building a KBS. However, it differs from standard rule based systems since new rules are added in the context in which they are suggested.

To continue the previous statement which claims RDR addresses the issues of CBR, one can liken an RDR system to a CBR system, only RDR provides a utility, in the form of an algorithm, a structure and rules, with which to demonstrate which parts of the case are significant to a particular classification (Kang, B. & Compton 1994).

Early Work

RDR was first described in a paper by Paul Compton (Compton & Jansen 1989). It was then further defined in a later paper by the same author (Compton & Jansen 1992). Observations from attempts at expert system maintenance lead to the realisation that the expert often provides justification for why their conclusion is correct, rather than providing the reasoning process they undertook to reach this conclusion. That is, they say 'why' a conclusion is right, rather than 'how'. Furthermore, experts are seen to be particularly good at providing comparison between two cases and distinguishing the features which are relevant to their different classifications (Compton & Jansen 1989). With these observations in mind an attempt was made at producing a system which mimicked this approach to reasoning, with RDR being the end result.

It was first tested on the GARVAN-ES1 database of cases, and showed significant promise in both vastly reducing the time taken to add a new rule (up to 10 an hour, as opposed to a fairly standard 2 per day) and in maintaining knowledge base validity (Compton & Jansen 1989). A major demonstration of this technique was the Pathology Expert Interpretive Reporting System or PEIRS, which provided clinical interpretations for laboratory reports. It was found to grow from an initial 200 rules to 2000 rules without the use of any knowledge engineers over a relatively short time

period and was able to achieve results in the order of 95% accuracy in 20% of the entire Pathology domain (Preston, Edwards & Compton 1994).

Structure

The resultant RDR structure is that of a binary tree or a decision list (Rivest 1987), with exceptions for rules which are further decision lists. The decision list model is more intuitive since, in practice, the tree would have a fairly shallow depth of correction (Kang, B., Compton & Preston 1994). The inferencing process works by evaluating each rule in the first list in turn until a rule is satisfied, then evaluating each rule of the decision list returned by that satisfied rule similarly until no further rules are satisfied. The classification that was bound to the last rule that was satisfied is given.

Validation

With this structure it is conceivable that two or more rules might fire on the same case. Using the inferencing approach mentioned above, it would find the earliest rule which satisfied the case then fail to consider the other rule which might also have fired. In practice this is not much of a problem, since the expert is able to add refinement rules to correct this behaviour if the conclusion it produced was not the desired one. However, this problem is completely eliminated in most RDR systems by the addition of the validation phase. With this scheme a cornerstone case is stored for every rule that is created. It should be mentioned at this point that a cornerstone case is a case for which the knowledge had previously been modified and which is valid under the current context (Preston, Edwards & Compton 1994). When the expert attempts to create a new rule the system will detect that this rule would fire for a particular cornerstone case. The expert will then be required to select from a list of differences between the current case and this retrieved cornerstone case to make the new rule unique to the knowledge base.

RDR Limitations

Earlier work in RDR dealt only with single classifications (SCRDR), meaning it could only find one conclusion for any given set of facts. The PEIRS system mentioned earlier was originally built in this manner, and had to handle multiple conclusions by treating them as “compound diseases”, that is; one conclusion which mentioned two or more classifications. However, this solution had the potential to

exponentially increase the knowledge acquisition task which is clearly undesirable (Kang, B., Compton & Preston 1995). A speculative paper proposing some extensions to RDR was published in 1993 (Compton et al. 1993). The extensions proposed included Multiple Classifications which was then tested on thyroid cases from the GARVAN-ES1 KBS and described in greater detail in 1994 by Byeong Ho Kang (Kang, B., Compton & Preston 1994). PEIRS was later rebuilt to support Multiple Classifications in the form of LabWizard (PKS 2005).

Another potential problem is that the knowledge base may be poorly structured, with multiple instances of the same, or essentially the same, rule being present in different contexts (Compton et al. 1993). An example of this can be seen in Figure 2-2.6 where the "If *rain* then **umbrella**" rule appears in two separate contexts. This limitation is not considered to be a major problem in most situations because it takes so little time for the expert to add the new rules, and is a reduced problem in the event of Multiple Classifications where all pathways in the knowledge base are explored (Kang, B., Compton & Preston 1995).

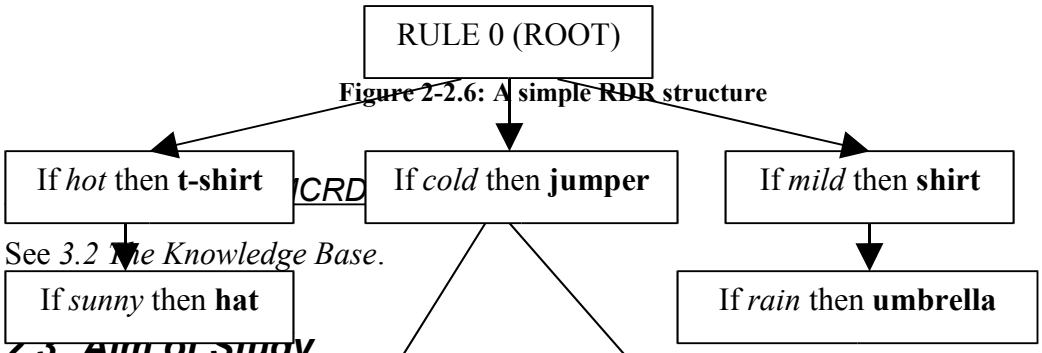


Figure 2-2.6: A simple RDR structure

See *3.2 The Knowledge Base*.

2.3 Aim of Study

It is known that knowledge representation models have typically been designed with the concept of [illegible] in the expert's brain. The idea was that this knowledge can be "mined" from the expert's brain (Buchana[illegible] this notion in principle, instead allowing the expert to justify their conclusions as required. However, RDR remains untested in domains which are not well defined. It has been tested against a host of applications - including pathology (Preston, Edwards & Compton 1994), conversation agents (Mak et al. 2004), diet (Kang, B. 2005), and document classification (Dazeley & Kang 2004; Park, S., Kim & Kang 2004) to name a few - and found to perform exceptionally well in some, but all of these domains can be said to be ones where the

knowledge being represented was well formulated and absolute, or if they weren't it was at least less important if the knowledge did not quite fit.

So, in this study it is sought to test the RDR principle against a domain which is both poorly defined and poorly structured, due to its relatively new and dynamic nature. This means that the knowledge that can be provided by the expert is highly prone to inconsistency. What effects this has on the ensuing knowledge base structure and its workability as a system shall be monitored through testing to determine whether this technique is suitable for such domains.

However, this study is not solely aimed at determining RDR's applicability to a domain with a new type of challenge. It also seeks to create an intelligent system which is capable of interpreting medication review cases and producing a comprehensive list of DRPs without falling into the trap of exhaustive checking techniques such as those Cognicare employ. This part of the study demands an easy to use system which can be comfortably incorporated into an existing work environment and be used to improve not only the quality, but also the quantity of medication reviews. On top of this it is desired that the system will be capable of reducing the amount of missed classifications (errors) that the medication reviewer tends to make in this domain, by acting as a kind of "extra expert" that can look over the reviewers' shoulders and pick up classifications they might have otherwise missed.

3 Methodology

In order to produce this health based KBS it was necessary to produce two major software elements. The first a standard implementation of a database "front-end" from which it is possible for a user to enter all the details of a given patient's case, or at least those parts which are relevant to the chosen domain. The second, an implementation of a Multiple Classification Ripple Down Rules engine (see *3.2 The Knowledge Base*) which can sufficiently encapsulate the types of conditions and knowledge required for the domain and facilitate the design of an interface from which the engine can be operated, particularly during the Knowledge Acquisition phase.

For these software elements to be successful in creating a working KBS a great deal of planning is required for the domain modelling aspect of the project. The model had to accurately, yet succinctly represent the domain, in that it both enabled the expert to specify enough of the relevant details about any given patient, and also allowed the system to process all the data in a meaningful and timely manner. In short, the expert must be able to accurately and easily specify rules that encompass their reasoning for any given conclusion.

It should also be noted that a solid interface through which to operate, even at this prototyping "proof of concept" stage, is very important. Without this the system becomes difficult for the expert to use, and may encourage a jaded viewpoint which makes the expert less likely to appreciate any possible advantages of the system. This type of system by nature requires expert cooperation to learn, meaning the experts' needs must be exhaustively considered.

3.1 Domain Modelling

When developing a KBS, the domain model that is derived is always an integral contributor to the success or failure of the system (Buchanan et al. 1983). With KBSs built using the MCRDR knowledge acquisition engine, this contribution becomes even more important, as the expert involved in incrementally building the knowledge base will invariably become disillusioned with the system if they are regularly unable to create rules that match their own reasoning processes, whereas a knowledge engineer might be a little more forgiving of their own system. To this end the domain had to be carefully researched before an adequate model could be designed.

3.1.1 Domain Research

To research the domain regular meetings were held with the experts. The first thing that was looked at was the kind of information the reviewers would normally be provided with when undertaking a review. This information would provide the basis for all their case understanding, as they are generally not expected to examine the patient personally at any time in the process. This information includes :-

1. Personal patient information
    a. Name
    b. Marital Status
    c. Gender
    d. Date of Birth
2. Medical History
    a. List of all the patient's medical conditions
    b. Includes a flag to indicate whether the condition is Past, Current or Surgical in nature
3. Medical Staff Observations
    a. Includes all symptomatic information and other treatment-related notes from the patient's medical carers
4. Notes
    a. Similar to the observations, specific however to more general doctor's or nurse's notes
5. Pathology Results
    a. List of all results from Pathology tests carried out on the patient
6. Drug Regimen

a. List of all the drugs the patient has been taking
b. Includes a flag to indicate whether they are still on the drug, or have ceased taking it
c. Includes dosage directions

It should be noted that it is relatively uncommon in KBSs to handle so many types of data, with many classical systems handling only one discrete area. However, it is partly due to the fact there are multiple fields of data that a system of this kind is needed for this domain, because it is easy for the expert to miss important links in the dense body of data that is provided to them.

3.1.2 Data Typing

Once this information was obtained it was possible to look at the specific data structures that might be required to contain it. For example, Pathology results come in numerical values with floating point precision required; however different units can be applied to the result. A result of '0.13 mmol/L' for the Creatinine levels means a very different thing to '0.13 pmol/L'. To this end a unit was tied to each Pathology reading which had to be strictly adhered to, with the unit being chosen to be the one which made most sense to the expert for each reading.

With different data types, different operations and thus rules are possible. To continue with the Pathology example, it is a simple matter to create a rule that states that a given result is greater than, or less than some value. Or even to create a range, by defining two conditions, one which is greater than a value, and one which is less than a value. However, data like the observations can only be represented in a textual string manner, making operations such as greater than or less than virtually useless. With text string data a simple keyword search function was defined, with equal to, or *not* equal to operations being the only ones eligible.

The basic structure of data for the domain was based on the Mediflags program developed by Chris Bonner (Bonner 2005), which is commonly used by medication reviewers throughout Australia (Tenni 2005). However, many elements of Mediflags data model were found to be inappropriate for the development of a KBS. Generally this occurred because Mediflags modelled a particular piece of data as a textual string, which was preferable to model as a numeric value. As stated earlier, a string

of text cannot have the same comparison operators applied as a number, or a list of nominal values might. A major problem was Mediflags' model for the drugs dosage, which is discussed in detail in the next section. There is a simple reason behind this kind of problem, in that humans are, almost by definition, intelligent. The reporting style database software developed for medication reviews in the past has been successful in achieving its goals because the humans reading those, and making those reports can apply their own intelligence to the data. They know, for example, that when a patient is on "Fenofibrate 67mg" that the 67mg means the dosage is 67 mg, and that if they're given directions to take this drug "1.5 bd" it means they're taking a total of 201 mg per day. For a piece of software to be able to comprehend this it must be explicitly given these pieces of seemingly "obvious" knowledge, and it must be given the data in a way from which it can apply that knowledge.

3.1.3 Problems with the Domain Model

When the domain model was first applied and presented to the experts in the context of creating rules, two major problems were identified (listed in *3.1.3.1 Major Concerns*), and subsequently solved. Several other minor issues were noticed (see *3.1.3.2 Minor Concerns*), but these were deemed of lower importance and noted for future iterations of the system, where the slightly higher levels of accuracy in rule creation may become important. Further still, there were identified issues which would add greatly to the range of classifications the system would be able to make, but which were deemed too complicated to add to the system at this stage and are discussed in *5.2 Further Work.*

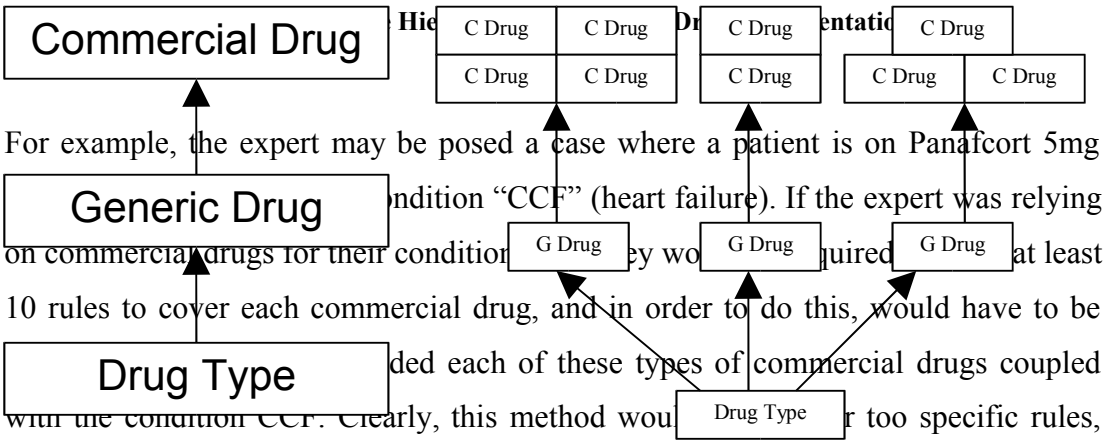
3.1.3.1 Major Concerns

There were two major concerns identified, both concerning the way the drug regimen was handled, with the drug regimen being considered one of the single-most important set of facts for the expert to base their rules upon.

Firstly it was revealed that there are three layers of hierarchy that apply to the drugs a patient can be taking. The highest level being that of the commercial drug, of which a comprehensive list was provided including 3011 different commercial drugs. The second level was that of generic drug names, being essentially the same as the commercial drug equivalent but encompassing a larger variety of these drugs. This

list is reduced massively from that of the commercial drugs, down to 837, meaning that for every generic drug there is an average of 3.6 commercial drugs which are essentially the same product with different names. The third and final level was that of the drug type (or group). This is a broad, brief description of what each generic drug does and consists of only 241 entries meaning an average of 3.5 different generic drugs and 12.5 commercial drugs which fall into each drug type category. An example of this hierarchy is shown in Figure 3-3.1.

These layers of hierarchy are important to the expert when creating conditions for the rules which lead to any given conclusion, as the conclusion might only apply to a particular brand of commercial drug, but is far more likely to apply to *all* drugs which stem from the generic drug parent of that commercial drug, and is also often likely to apply to the broader drug type grouping.



For example, the expert may be posed a case where a patient is on Panafcort 5mg [illegible] condition "CCF" (heart failure). If the expert was relying on commercial drugs for their condition [illegible] they would required at least 10 rules to cover each commercial drug, and in order to do this, would have to be provided each of these types of commercial drugs coupled with the condition CCF. Clearly, this method would require too specific rules, and would take too many cases and too many rules to provide a sufficient level of support.

However, Panafcort 5mg tablets are of the generic drug group of Prednisone, so with this level of hierarchy it becomes possible for the expert to add a rule based on the generic drug of Prednisone which would cover each and all of the 10 commercial drugs which come under this group, and if the drug was entered into the case as the generic drug Prednisone itself (which is possible) it would of course, cover this eventuality too.

Further still, in some cases (including this one) the expert may wish to base the rule on the entire group of drugs. So, for this case the expert can add a rule stating that the patient is on a drug type of "Corticosteroid – oral". This type will cover each of the 7 generic drugs which fall under this grouping, which in turn will cover all of the commercial drugs which fall under each of those generic drugs. Through this hierarchical abstraction it is possible for the expert to quickly add rules which will cover all manner of ranges of drugs, from the very specific, to the very abstract which allows them to build a knowledge base which both makes sense to them, and is quicker to build.

The second major concern was with dosages (as alluded to earlier in the chapter). When the expert attempted to start creating rules they discovered that they were unable to easily make rules based on the total daily dosage that the patient was receiving, as the list of drugs provided was simply a list of all the Commercial Drugs available in Australia, and did not include dose information at a broad level. Instead, when multiple strengths of a drug were available the drug was just listed multiple times, with different strengths applied to the name of the drug, such as "Risperdal Quicklet 0.5 mg", "Risperdal Quicklet 1 mg" and "Risperdal Quicklet 2 mg".

For example, it was possible for the expert to make a rule stating that if the patient was taking the commercial "Xalatan Eyedrops" with the directions "1 bd" (which actually means twice a day) then they were taking too much Xalatan, which should only be applied once daily. The problem with this approach is that the expert would have to define similar rules for all the other usage directions which the patient may have been given, and these rules would only be definable when a case with those directions appeared, making the knowledge acquisition task for this kind of situation unnecessarily complex and frustrating. It would be preferable if the expert were able to create a rule stating that if a patient was on any of the generic drug type of "Latanoprost Eye Drops" with a total daily dosage that was more than once per day (Latanoprost Eye Drops Daily Dosage > 1), then they were over-applying.

To facilitate this feature it was necessary to restructure the way each drug was represented to include a dosage value where possible. The only viable way to do this for 3011 different drugs was to pre-process the drugs to extract the dosage out of the

name, and then store this dosage value coupled with each drug. So the numerical dosage value in the drug name "Risperdal Quicklet 0.5 mg" would be pulled out as "0.5". To find a total daily dosage value this dosage value was then multiplied by a multiplier value which was determined for each dosage direction such as 2 for something that was taken twice daily or 0.5 for something taken every second day. Once the dosages for each drug were stored against each drug entry, and the multiplier table was built, it was possible to determine what the total daily dosage for any particular drug was for each patient. A new condition was added to the system, "Daily Dosage" that made use of this information and allowed the expert to define the rules as they desired.

One problem with this method was that some commercial drugs did not have an easily identifiable strength in the name and as such it was impossible to automatically extract a dosage value from them. To handle this, a strength of 1 (unit) was applied. There were two main reasons for a strength to be impossible to find. The first was simply no strength being present in the name, such as the "Xalatan" example above. The second was when a ratio was applied. An example of this would be "100mg/5mg/0.5mg", where the drug contained 100mg of one thing, 5mg of another, and 0.5mg of yet another. Another example might be labels like "25 mg/ml" meaning milligrams per millilitre, with no way of knowing how many millilitres the patient was taking. However, only 245 of the Commercial Drugs in the list were afflicted with these properties, only 0.08% of them, and in many of those instances the dosage value of 1 was considered correct (as was the case with the above eyedrops example), so it was deemed this method was acceptable. Although, in a future implementation it would be preferable to define a new model for representing what drugs a patient is on which allowed the data entry operator to enter the drug the patient is taking, and then a dosage value, specific to that drug for that patient, to ensure an accurate representation of the dosage for each patient as detailed in *5.2 Further Work*. This approach has an added advantage of reducing the length of the list of commercial drugs, since many of these are duplicated entries with different dosage values.

3.1.3.2 Minor Concerns

The minor concerns were mostly to do with the observations of the patient, and the way the knowledge was represented in the case. The model for representing the

patient observations was taken from a database system that was used to store cases for medical review. However, this system was not performing any processing of the data, it was merely providing a facility for the reviewer to view the case and produce a report based on it. As such, observations such as Weight, Blood Pressure and others were represented as text when they really need to be either discrete nominal value ranges (low, medium, high), or numerical values for the purposes of creating rules on them.

To amend this situation a future implementation might include a data-type field for each observation type, allowing the programmer to define what type of information is being encoded in the field so that the system can then interpret it correctly. More details on this are included in *5.2 Further Work*. This enhancement was not made for the purposes of this experiment as it was considered to be a less important enhancement, with the observations rarely being used by the expert to reach their conclusions.

Another concern was that of standardisation. This was not a minor issue, but it was considered unnecessary to address for the successful operation of the system, although it would be preferable if a standard model could be used. The problem lies in that it is difficult at best to persuade any discipline to use one standard set of terminology. The problem becomes worse when considering multiple disciplines, with each member of each field not only having their own opinions about how best to term their information, but also having opinions as to how to term the information they must represent in the other fields. In short, no standard has been agreed upon thus far, and there is no guarantee that an ideal standard exists.

This problem doesn't manifest itself so obviously in situations where humans must interpret the data, with the humans capable of understanding the numerous ways of expressing the information, but when a machine is tasked to work with this data it becomes unnecessarily complex if it must handle numerous expression types for all their sets of data. To solve this, the system would be required to either pre-process all the data with another system to standardise it for its own purposes, or it would require the expert to insert rules to handle all the different methods of expression. Obviously both these approaches are undesirable, with the task of knowledge

acquisition increasing either way. In future iterations it would be sensible to remove this dilemma by enforcing a stricter standard for all data in the system, this solution is discussed further in *5.2 Further Work.*

3.2 The Knowledge Base

Previously mentioned in this chapter was the use of the MCRDR engine to drive the knowledge representation and acquisition. This method provides a powerful and flexible representation of knowledge which can be incrementally built up by the expert who does not necessarily have any understanding of knowledge engineering. It was chosen over other methods because it is one of the few methods that are capable of sufficiently representing such a poorly structured and complicated domain, offers extremely fast system development when compared to the other methods which would be capable of accurately modelling this domain (particularly traditional rule-based methods), and is easily maintained (Compton & Jansen 1989, 1992; Kang, B. & Compton 1994; Kang, B., Compton & Preston 1995; Kang, S. J. & Chien 1998; Preston, Edwards & Compton 1994).

3.2.1 Multiple Classification Ripple Down Rules

3.2.1.1 Inferencing

As described in chapter 2, RDR is based on the concept that every time a misclassification is found an exception should be added to the rule set to deal with this misclassification, but the exception should only be added in the context of the original rule (Compton & Jansen 1992). The resultant structure can be described as a tree, with a root (always true), a first level (initial rules), and lower levels (exceptions to these rules, or exceptions to the exceptions etc.) as shown previously in Figure 2-2.6. This same example is continued throughout this following section.

However, the RDR method is limited by its inability to produce multiple conclusions for a case. To allow for this capability, as our domain must, MCRDR should be considered (Kang, B., Compton & Preston 1995) as the compound-classifications solution discussed earlier has the potential to exponentially increase the knowledge acquisition task (Preston, Edwards & Compton 1994).

MCRDR is extremely similar to RDR, preserving the advantages and essential strategy of RDR, but augmented with the power to return multiple classifications. In RDR once a rule is satisfied no other rules in the knowledge base are evaluated, in contrast MCRDR will evaluate all rules in the first level of the knowledge base then evaluate the next level for all rules that were satisfied and so on, maintaining a list of classifications that should fire, until there are no more children to evaluate or none of the rules can be satisfied by the current case (Kang, B., Compton & Preston 1994). For example, using Figure 2-2.6 as our structure, a case (*cold, rain, windy*) would actually classify **jacket** and **stay-home** as seen in Figure 3-3.2.

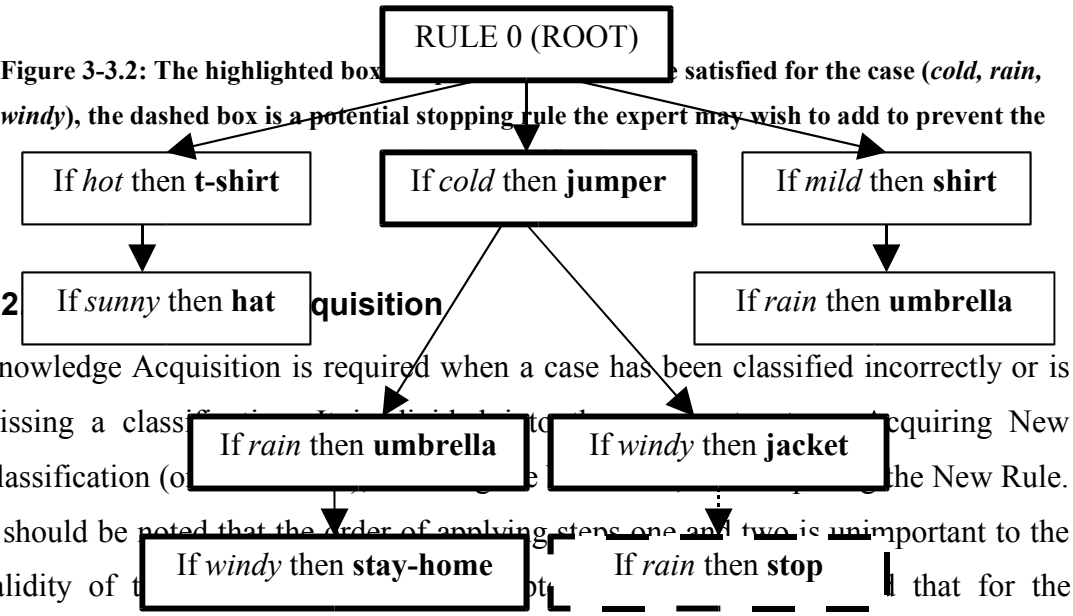


Figure 3-3.2: The highlighted box[illegible] satisfied for the case (*cold, rain, windy*), the dashed box is a potential stopping rule the expert may wish to add to prevent the

3.2 [illegible]quisition

Knowledge Acquisition is required when a case has been classified incorrectly or is missing a classi[illegible] Acquiring New Classification (o[illegible] the New Rule. It should be noted that the order of applying steps one and two is unimportant to the validity of t[illegible] that for the purposes of this experiment it made sense to locate the rule before acquiring the new classification. Hence this is what was done.

Acquiring the New Classification is a trivial problem; the system merely prompts the expert to state it (Kang, B., Compton & Preston 1994). For example, the system may produce the classifications **t-shirt**, **hat** for a given problem, the expert may decide that a **t-shirt** is ok, but a **hat** is unnecessary, and instead there should be **sunglasses** and **thongs**.

To Acquire the New Rules the expert is asked to first select valid conditions from the current case that indicate a given classification. The rule they have created thus far is then compared against the cornerstone case base. If any cornerstone cases would fire on this new rule the expert is asked to select from a difference list (see Figure 3-3.3)

between the presented case and one of the cornerstone cases in the list of cornerstone cases being considered. It should be re-iterated that a cornerstone case is a case for which the knowledge had previously been modified and which is valid under the current context (Preston, Edwards & Compton 1994). The system then re-tests all cornerstone cases in the list against the conditions selected and removes cornerstone cases from the list that do not satisfy the conditions selected so far. The expert is then prompted to choose conditions from a difference list between the current case and one of the other remaining cornerstone cases in the list. The conditions selected are added as a conjunction to the rule. The system repeats this process until there are no remaining cornerstone cases in the list to satisfy the rule (Kang, B., Compton & Preston 1994) or alternatively the expert has stated explicitly that the cornerstone cases that remain *should* fire on the new rule. So, with this system the expert will at first be considering *all* stored cornerstone cases which are valid in the context of the new rule, but gradually whittling them down as they add more conditions.

Cornerstone case	Current test case	Difference list
Rain	Rain	Not applicable
Meeting	Meeting	Not applicable
Hot		Not(Hot)
	Cold	Cold
Cold, Rain	Cold, Windy	Not(Rain), Windy

Figure 3-3.3: An example of a decision list from (Compton & Jansen 1992; Compton et al. 1993; Kang, B. & Compton 1994) note the difference list can contain negated conditions

The system must now locate where the new rule should go (see below) and add it. It then tests the remaining cornerstone cases associated with the parent rule and any cases which can satisfy the new rule are saved as a cornerstone case of the new rule. At this point the new case is added to the cornerstone case database. The lists of cornerstone cases for the other rules correctly satisfied by the case (that is, giving a correct classification for that case) are updated to include the new case (Kang, B., Compton & Preston 1994).

To determine where the new rule must go it must first be determined what type of wrong classification is being made. The three types are listed in Figure 3-3.4. When examining the previous example you may have noted that the classifications were perhaps not ideal, since wearing a jacket and then staying at home would probably be

uncomfortable, but the expert has the ability to correct this classification by adding a stopping rule to the node that states “If windy then jacket” which would prevent the classification in the event of the condition *rain* holding true as can be seen in Figure 3-3.2.

Wrong Classifications	To correct the Knowledge Base
Wrong classification to be stopped	Add a rule (stopping rule) at the end of the path to prevent the classification
Wrong classification replaced by new classification	Add a rule at the end of the path to give the new classification
A new independent classification	Add a rule at a higher level (to the root) to give the new classification

Figure 3-3.4: The three ways in which new rules correct a knowledge base (Kang, B., Compton & Preston 1994)

The entire knowledge acquisition process can be conceptualised into a flow diagram such as that in Figure 3-3.5.

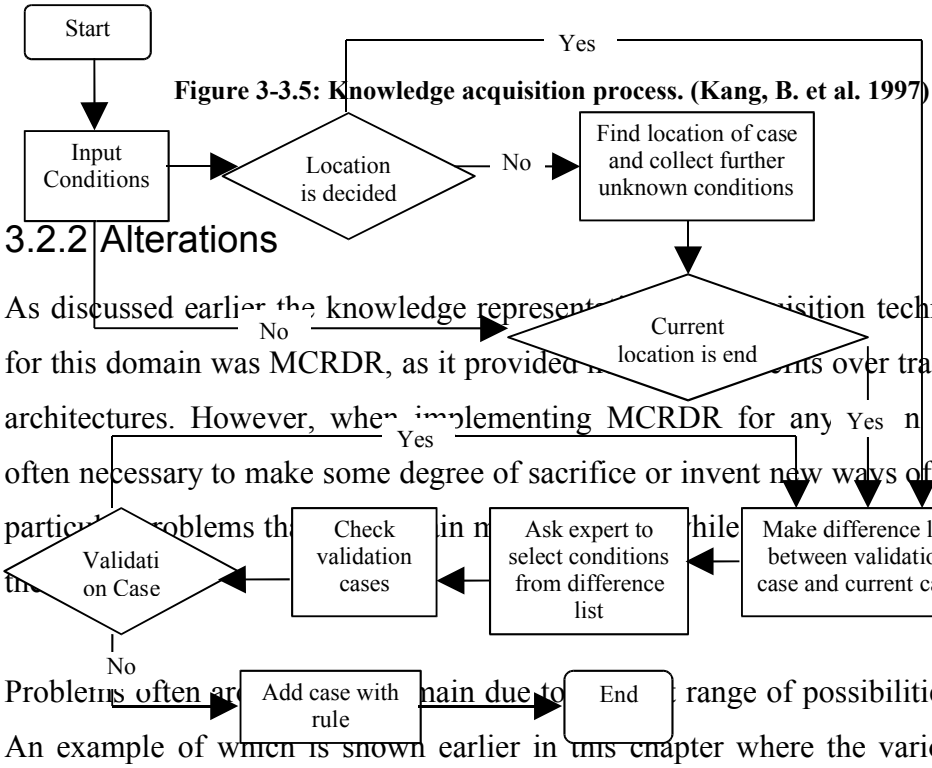


Figure 3-3.5: Knowledge acquisition process. (Kang, B. et al. 1997)

3.2.2 Alterations

As discussed earlier the knowledge represent[illegible]isition technique chosen for this domain was MCRDR, as it provided [illegible]nts over traditional KBS architectures. However, when implementing MCRDR for any n domain it is often necessary to make some degree of sacrifice or invent new ways of handling the partic[illegible]roblems th[illegible]in n[illegible]hile [illegible]validating th[illegible]

Problems often ar[illegible]main due to [illegible] range of possibilities for a case. An example of which is shown earlier in this chapter where the various levels of grouping were required to categorise the various drugs that were available to each case.

3.2.2.1 Selecting Conditions

Another one of these case-size issues was handled when trying to select the conditions to make a rule. The normal approach is to list anything which can be a valid condition for the case (and/or cornerstone case when one is being considered), but this includes things which do NOT apply to the case. In this domain, the entire list of options can, for some conditions, be impracticably large.

For example, a patient may have 12 medical conditions listed under their Medical History. That's 12 of the 984 possible medical conditions with which a patient may be afflicted. To provide a complete listing of all the medical conditions available to the current case, the remaining 972 conditions would have to be selectable also, in the context that the patient does NOT have these conditions (example: if patient does *not* have muscle cramps then conclusion-A). Further to this, if the expert was allowed to select all the possible conditions as is normally supplied in MCRDR systems it is likely they would be presented with an excess of cornerstone cases during validation. Continuing the last example, "if patient does *not* have muscle cramps then conclusion-A", this rule has the potential to return every single case in the cornerstone case base that doesn't exhibit muscle cramps, and similarly for any of the other types in the system ("If patient is not on drug-A", "If patient does not have observation-C" …).

To solve this problem it was decided that only the features that actually applied to the current case (and/or cornerstone case when one was being considered) would be selectable as conditions, as it was understood that these conditions are the ones used in the vast majority of instances. Furthermore, even when the expert may have desired to use a *not* condition they were often able to select it from one of the cornerstones which were returned, or in the worst case they were always able to add an exception rule later to repair the knowledge when an incorrect classification did occur. It is hoped that this decision will not result in a significant increase in time taken for validation.

3.2.2.2 Adding Exceptions

Another problem that was overcome when developing the system was that of adding exceptions. It was observed that the same conclusion was often reached via several

rule paths. It was considered undesirable to display the conclusion multiple times for each rule path it followed, and allow the expert to add the exception to whichever one they may desire. Another alternative, and a common approach to this problem, is to display each unique conclusion only once, and when adding an exception to it *all* the rule paths which lead to that conclusion would have the exception added to them. However, this solution has a problem in instances where the expert might wish to craft their knowledge base more precisely, saying that the conclusion shouldn't have been reached on one or more of the rule paths that reached the conclusion, but still should have been found on one or more of the others. That is, yes, it did find the right conclusion, on this path, but for the wrong reason, while on other paths it did find the same right conclusion, but for the right reasons.

To solve this problem two things were done. Firstly, when adding exceptions to rules the entire list of conditions that applied for each rule path to have fired is displayed. This lets the expert differentiate between the different rule paths used to reach the conclusion they are excepting, and has the added benefit of reminding the expert what reasoning underlies any particular conclusion. Secondly, a checkbox was placed next to each of these lists of conditions, allowing the expert to select which paths they would like the exception they are creating to apply to. In this manner the expert is given more complete control over how they craft the knowledge base.

3.3 Evaluation

Once both the database front-end and the MCRDR engine were completed, the system was handed over to the expert with absolutely no knowledge or conclusions pre-loaded. The expert was wholly responsible for populating the knowledge base; however they did have 130 real cases pre-loaded, which had been converted from existing data they provided which had previously been entered into their Mediflags-based system. Over the course of approximately 15 hours they were able to add the rules required to correctly classify 84 of these pre-loaded cases. This speed can in part be attributed to the fact that the pre-loaded cases had been pre-classified by other medication reviewers, meaning the expert was able to look for the primary indicators of the classifications already reached by the previous experts and validate the other reviewer's findings, rather than having to classify the cases completely from scratch.

During the process of populating the knowledge base the expert was asked to keep note of several key details as they went, including: -

- Current Case Number
- Number of Conclusions Identified by System
- Number of Rules Added to Correctly Classify the Case
- Number of new Conclusions Written for the Case
- Number of Classifications Removed for the Case
- Number of Classifications Replaced for the Case

This data was used for tests 1, 2, 3 and 5 as described below.

Further still, the experts actions within the system were logged as they were performed, each log entry given a timestamp, an action identifier and a parameter, with the parameter being dependent on the type of action performed. This data was used for tests 4, 6 and 7.

3.3.1 Test 1 – Growth of Knowledge Base

This test is designed to identify the trend the expert followed when adding rules to the system, by comparing how many cases had been analysed by the expert against how many rules were in the system when the expert moved to a new case. It is particularly useful for determining what stage in the knowledge acquisition the system started to slow down its learning curve.

It is expected that this test will show a linear increase in the number of rules in the system, as the expert was unable to process enough cases before the experiment was terminated to reach any kind of plateau. It was estimated by the expert that the system had reached an understanding of about 60% of the domain by the time of termination (Tenni 2005).

3.3.2 Test 2 – Correct Conclusions Found

This test seeks to identify what percentage of the correct classifications are found by the system as the expert considers progressively more cases, to demonstrate how much of the domain the system has been capable of learning in the duration of the experiment. It does this by considering how many classifications the system returned as each case was analysed, and comparing this to how many rules were added for that

case, as shown in Figure 3-3.6. Instances of divide by zero were handled by treating the result as 0.

$$\frac{C_f - C_{removed} - C_{replaced}}{C_f + R_a}$$

Figure 3-3.6: Where C_f is Conclusions Found, R_a is Rules Added, $C_{removed}$ is Conclusions Removed, $C_{replaced}$ is Conclusions Replaced

It is expected that this percentage should rise in proportion to the number of cases analysed. It has historically been demonstrated that rise would be linear until around the 80% mark before a flattening pattern was evident (Kang, B., Compton & Preston 1994). In this experiment a linear growth is expected, since the expert believes the system is up to only around a 60% classification rate (Tenni 2005) and as such is likely to still be in a heavy learning period.

3.3.3 Test 3 – Classifications Found, Expert vs. System

This test seeks to compare the number of classifications the expert initially found for each case against the number of classifications the system found once the experiment ceased. Through this comparison it is anticipated that the improvements the system can offer to the number of correct classifications found per case will be clearly evident. It is performed by comparing the number of classifications that were found on each case when the expert had finished with it to how many classifications were found on the case at the end of the experiment, after extra classifications had been added through the process of adding new rules to later cases. It is anticipated that the system should always find at least as many classifications as the expert who trained it, and will often find more than the expert, reflecting the inconsistency of the expert's classifications.

3.3.4 Test 4 – Percentage of classifications missed

This test was used to determine how regularly rules were added which actually improved the classifications made on old cases in the system. Unfortunately it is difficult to test exactly how often this occurred since not all cases are considered

when adding a new rule, but it is relatively easy to determine how often rules were added which improved old cornerstone cases by adding a previously missing classification to them. Since all testing was done during a heavy learning phase of the system, most cases examined became cornerstone cases, so this test is considered a reasonable approximation of the actual number.

For this test, log entries were processed to count how many log entries were found in the period immediately prior to actually adding the rule that indicated the expert hit "permit". This was done by counting how many "VALIDATING_HIT_PERMIT" entries were found consecutively then adding this count to the total *only* if an "ADDED_RULE" entry was found, meaning that these permitted cases were actually applied. The count was discarded if an "ADDED_NEW_CONDITION" entry was found, since this entry indicates that a new condition was chosen from the differences between the case being considered and the cornerstone case being checked against, and as such the instances of "VALIDATING_HIT_PERMIT" which had been discovered previously were never actually applied to the cases they had been permitted for. The number of valid "VALIDATING_HIT_PERMIT" entries found is compared against another counter which keeps track of how many cases had been analysed by the expert at each given point. With this data it is possible to find a percentage of the case-base which was influenced by the new rule. It requires only a small amount of further extrapolation to realise that this percentage is the percentage of errors (missed classifications) the expert made at that earlier stage in the system learning process.

It is hoped that this test will find a significant volume of instances where a rule added improves the classification of previously examined cases, with this kind of result suggesting the system can significantly improve the Quality of Service that is being provided to the patients by detecting correct classifications that the expert performing the medication review previously missed. Occurrence of this is anticipated, as it is observed that the expert assessment of a case is biased towards certain areas of examination that are determined after a cursory check, and also because of the fallible nature of human examination.

3.3.4.1 Total errors found per case

This test is an extension to Test 4, and attempts to determine a total figure for the percentage of cornerstone cases that receive new classifications in the process of analysing each case. That is, how much of the cornerstone case-base is amended in the process of considering each given case. It is performed by summing the percentage of error (as determined in Test 4 – Percentage of classifications missed) for each rule created for a given case. With this figure it is possible to get a more representative value for the level of errors the expert is making on a case by case basis.

3.3.5 Test 5 – New Rules Required

This test is aimed at showing how many new rules were required to satisfy the expert opinions of the case as progressively more cases were processed. To this end the amount of rules that are added per case are plotted on a column graph to demonstrate the emergent pattern, if any.

It is hoped that these results will indicate a clear downwards trend in the number of new rules required as progressively more cases are added, with the system detecting greater numbers of the correct classifications as it learns more and thus requiring less refinement.

3.3.6 Test 6 – Time Spent

It was considered important to know how much total time the expert spent adding rules to the system, with this information useful in determining whether the maintenance of the system is affected, and if so how much, by the nature of the domain. As such this test was carried out on the logged entries by comparing the timestamps of consecutive entries indicating a rule was entered. The total time difference was added up for each valid entry, then averaged. Time differences were considered invalid if they were greater than 10 minutes, as this time was determined by inspection of the log to be a reasonable approximation of when the difference was inspired by a significant break in activity. Similarly, the time difference was considered invalid if it was less than 10 seconds, with these entries being very likely to be related to accidental button presses rather than valid rules.

It is hoped that this time should not be excessive, with an average time spent per case not exceeding a few minutes. Historically with this kind of system figures of around 3 minutes per case can be expected (Kang, B., Compton & Preston 1995) although in this case it is suspected that the broader range of conditions available from which to produce rules from will cause the average time to exceed this slightly.

A further test was carried out, only dealing with log entries indicating a case was loaded instead of a rule being added. This test was used to determine the average time spent on each case by adding up the time differences between consecutive load-case entries. Differences were considered invalid if they were in excess of 25 minutes, this being an approximation of the point where it became reasonable to assume the difference was inspired by a significant break in activity. They were also considered invalid if they were less than 2 minutes, as this behaviour was more likely to be from the expert loading a case for inspection rather than for completion. It is estimated that the expert might spend around 15 minutes per case based on observation.

3.3.7 Test 7 – Cornerstones Seen

With this test it is sought to discover how many cornerstone cases the expert must deal with in the process of adding a new rule, specifically as the number of cases that have been analysed goes up. This is important in determining the maintainability of the system, since if the expert must deal with an excess of cornerstone cases they will be less inclined to use the system, and be able to use it less efficiently.

To do this test the logs were processed to determine how many cornerstone cases the expert was subject to for each rule they added, with results removed if they were deemed to be caused by error after inspection.

It is anticipated that this value will rise steadily early, but should plateau at some threshold as it has been observed in previous experiments with RDR that the expert is good at selecting conditions which do not need a lot of further refinement (Preston, Edwards & Compton 1994), although it is again possible that the plateau will not be observed because of the relatively low classification rate mentioned above.

3.4 Overview

As can be seen when reading the preceding sections, the results are structured such that the first two tests are aimed at determining what levels of knowledge the system has reached in the course of the experiment, the next two seek to demonstrate whether the system is capable of reducing the missed classification rates of the expert, and thus can improve the consistency of service offered. The remaining three are designed to show whether the MCRDR technique remains maintainable in light of the particulars of the domain being considered. A further analysis is also provided into the structure of the resultant MCRDR knowledge-base to determine whether any peculiarities are introduced there through the process of handling the inconsistent knowledge.

4 Results and Discussion

4.1 Test 1 – Growth of Knowledge Base

It is observed in Figure 4-4.1 that the number of rules in the system progressed linearly as more cases were analysed, at a fairly consistent rate of about 2.3 rules per case. This suggests that the system was still in a heavy learning phase when the experiment was finished, since it has previously been observed that RDR systems will show a flattening pattern in the rate of growth of the knowledge base at approximately 80% of domain coverage (Kang, B., Compton & Preston 1994). This has complications for many of the remaining tests, in that their results must be understood to reflect the knowledge base while it is still learning heavily. However, it should be observed that this is still a good result, with the expert being impressed that the system can learn so much in such a short period (Tenni 2005).

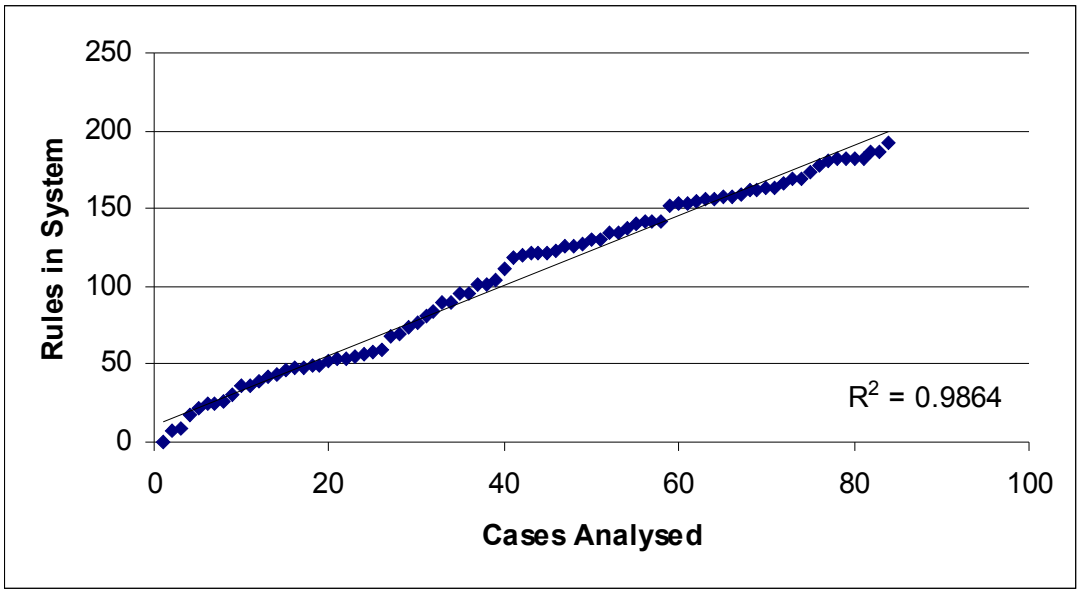


Figure 4-4.1: The number of rules in the system grows linearly as more cases are analysed

4.2 Test 2 – Correct Conclusions Found

It was estimated by the expert at the time of cessation of the experiment that the system had encapsulated around 60% of the domain (Tenni 2005), this estimation is supported by the evidence shown in Figure 4-4.2. It can be seen that the average number of correct classifications the system provided rose quite steadily into the 60th percentile, although the percentage correct from case to case did vary quite a lot, as is to be expected when the system is still learning.

The expert predicted potential classification rates in the order of 90% (Tenni 2005), so considering 84 cases had been analysed it would be estimated that in order to reach an average of 90% correct classification at least another 40 cases should be analysed, and it would be unexpected if the number of additional cases needed to be analysed exceeded about 120, based on previous figures found for systems of this kind (Kang, B., Compton & Preston 1994; Preston, Edwards & Compton 1994). These figures are justified by following the trendline in Figure 4-4.2, although it is conceded that this trendline is a rough approximation. If it is followed linearly as demonstrated thus far it reaches 90% at approximately 120 cases, if it is assumed that this trendline may begin to plateau though, as expected, it is possible that the number of extra cases required may grow considerably, to reflect the slower rate of learning.

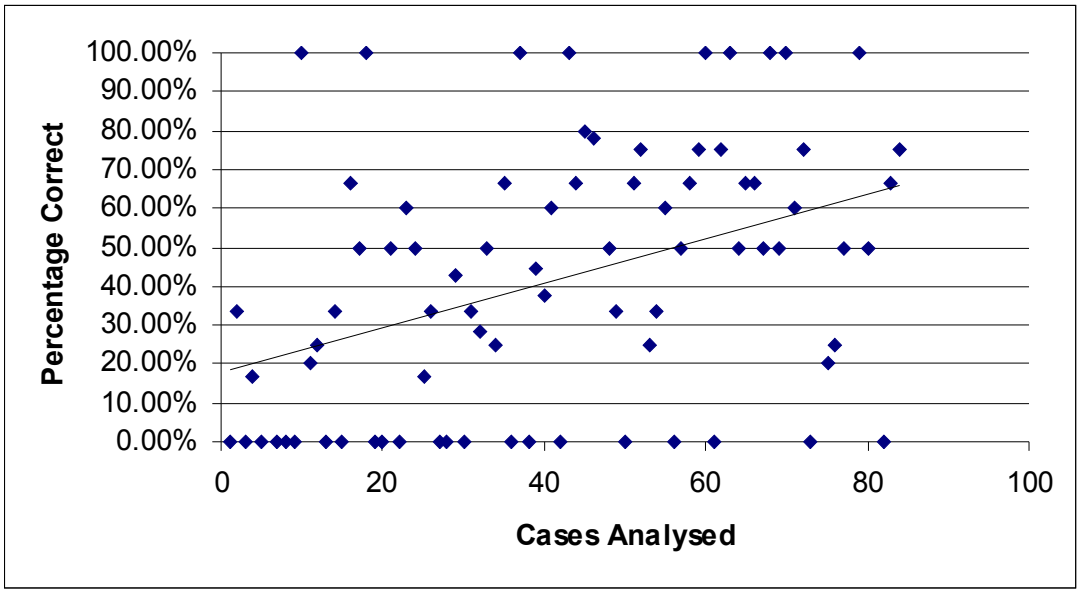


Figure 4-4.2: The percentage of conclusions provided that were correct

4.3 Test 3 – Classifications Found, Expert vs. System

The results shown in Figure 4-4.3 are very convincing, with the system sometimes finding half again as many classifications per case as the expert and quite consistently remaining at least one classification ahead. It should be re-iterated that the system found all these classifications using only a smaller set of the same knowledge the expert had. This is evidence that the expert consistently misses classifications, even though they are aware of them in principle. In other words, they just don't notice them on the particular case. The system does not suffer from this, it will notice anything that it is trained to know about without exception.

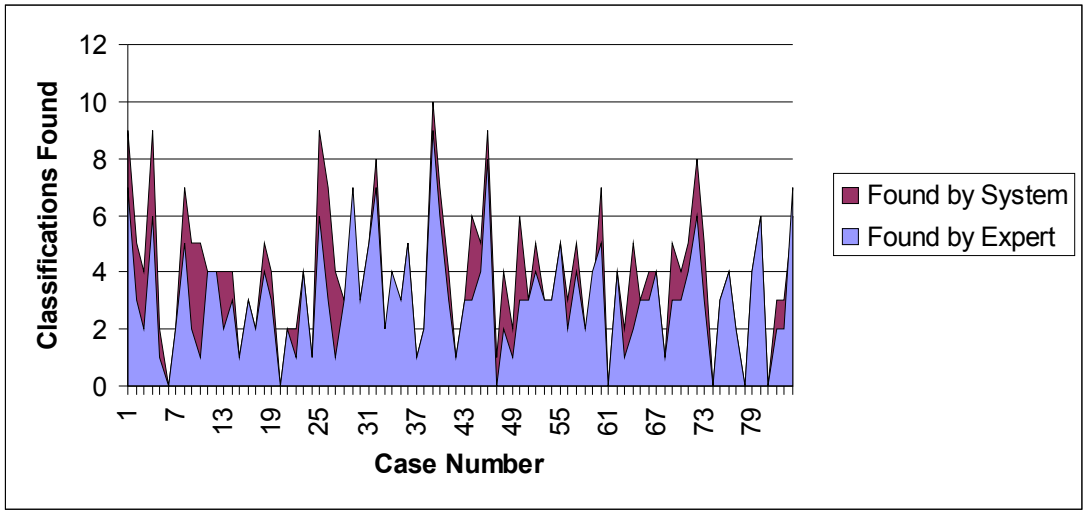


Figure 4-4.3: The system found significantly more correct classifications than the expert

4.4 Test 4 – Percentage of Classifications Missed

The results of this test were interesting, as it was found that the expert often allowed new classifications to be added to cornerstone cases. It can be seen in Figure 4-4.4 that in the early stages the expert was often missing classifications. What this means is that the expert had previously missed the classification when they first considered that case, then they had created the rule to handle the same classification for a new case and the system has realised that this new rule they created would have fired on the case previously considered. When the system detects this situation it will prompt the expert to either revise their rule or accept that the classification should have been added to the previous case (as detailed in *3.2.1.2 Knowledge Acquisition*). The percentage reduced dramatically even after only a small number of cases, suggesting

the system was rapidly helping to reduce the experts rate of missed classifications, by suggesting the classifications for them, rather than having to notice themselves. The trendline in Figure 4-4.4 is only an approximation, since relatively few cases have been analysed thus far, and noise is still significant.

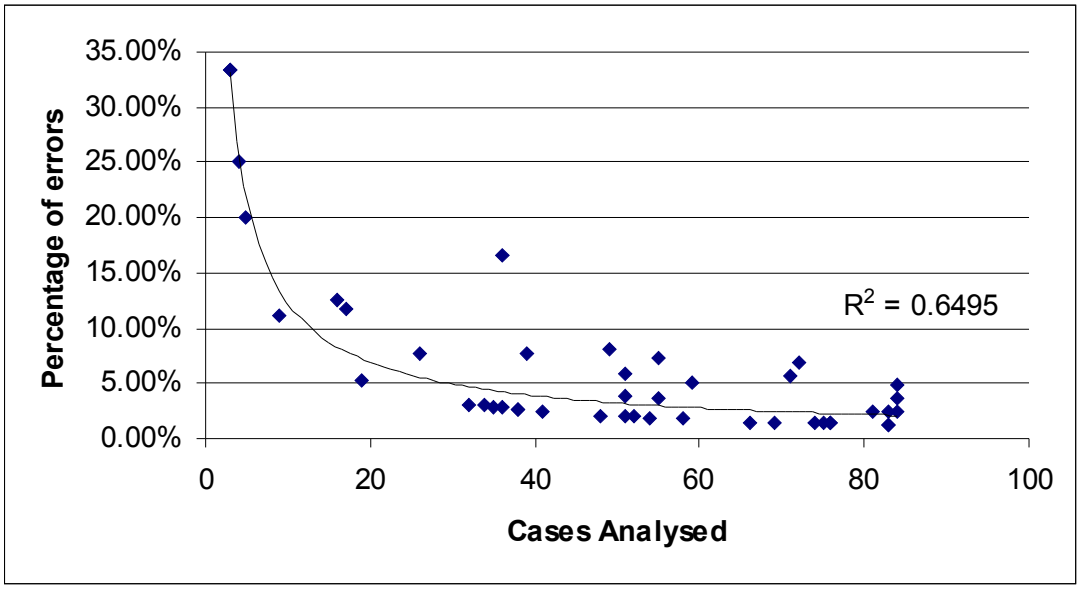


Figure 4-4.4: The percentage of cases that gained new classifications

It should be noted that these figures demonstrate only the errors that the system fixed in the course of operation, and do not show errors that have not yet been fixed. It would be reasonable to assume that since the system is still in a heavy learning phase there would still be a number of missed classifications that will be determined as more rules are added to the system, so this result is likely to actually improve further given more testing.

4.4.1 Total Errors per Case

It was found that the rate of error in each case was quite high, averaging 13.44% and with some going over 50%. Clearly the expert is making errors regularly, as was expected, and yet these numbers would be expected to be even higher were more

rules entered by the expert, since it is very unlikely that every possible error has been detected already considering the system has been shown to be covering only 60% of the domain thus far. It is important to note that the results shown in Figure 4-4.5 are representative of all the errors (missed classifications) that the system has fixed through the normal course of operation, and not the actual number of errors per case. This result is significant, since it is known that without the system providing extra checking all of these errors that were detected would have been missed.

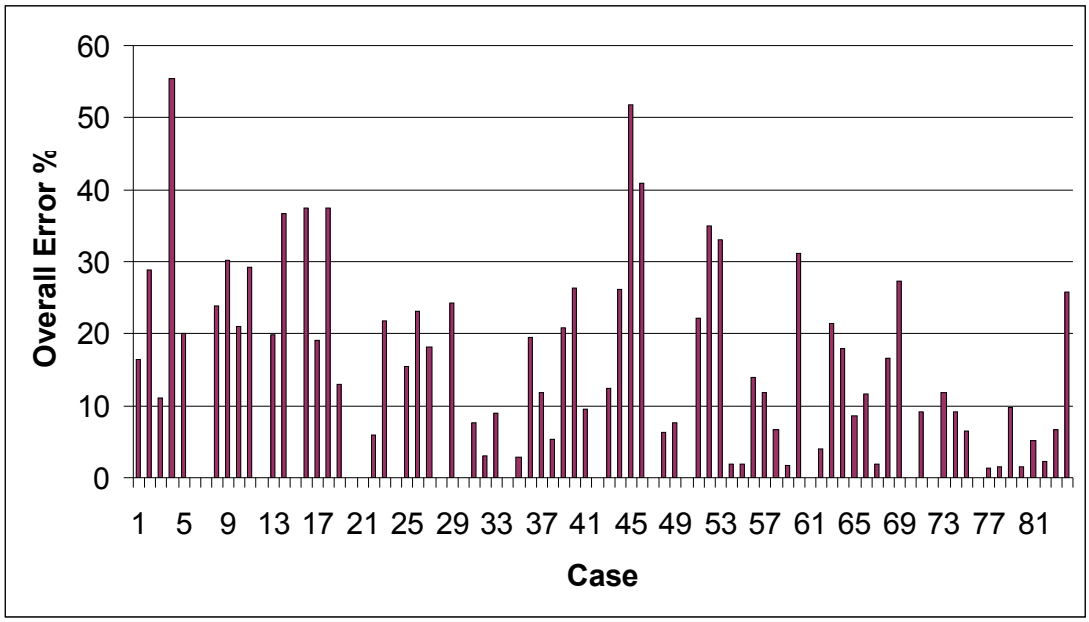


Figure 4-4.5: The final percentage of classifications missed by expert per case

4.5 Test 5 – New Rules Required

The results of this test show very little evidence of the number of new rules required tapering off towards the end of the experiment, as seen in Figure 4-4.6. This provides further testament to the learning period being unfinished. With a fully populated knowledge base figures as low as 1 new rule per 375 cases have been reported in another system of this kind (PKS 2005), while the data shown here suggests it is rare for even two cases to pass where a new rule is not added. Another feature which can be seen is that many cases require a considerable number of new rules to be represented, with some exceptional cases needing 7 or more. This alludes back to the number of classifications which can be found for a case as shown above, with up to 10 classifications found on a single case.

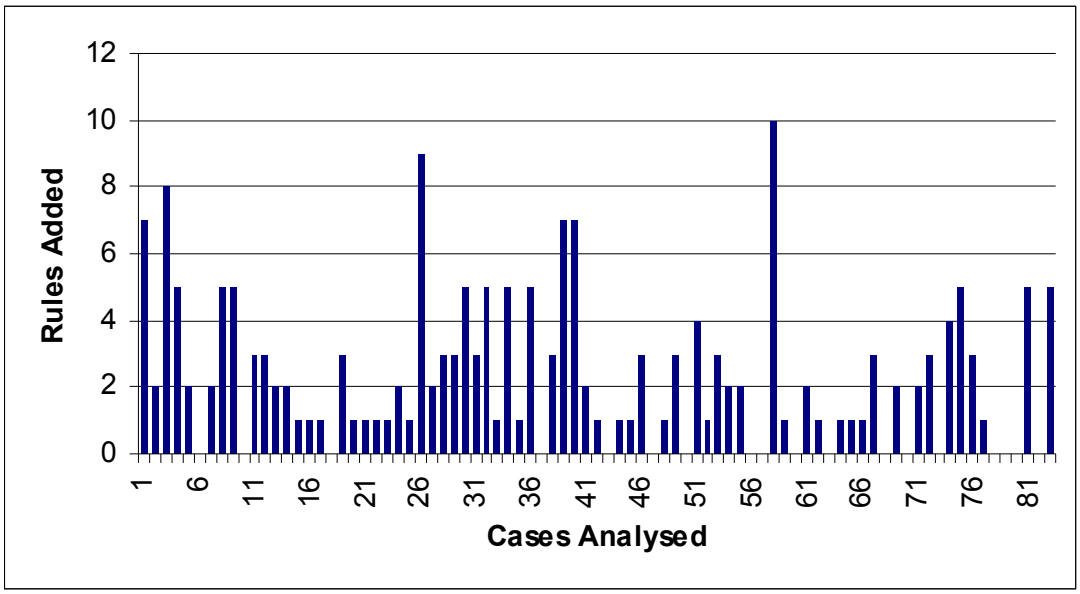


Figure 4-4.6: Number of rules added per case analysed

4.6 Test 6 – Time Spent

As discussed, the log entries were examined to determine how much time was spent performing various activities in the system, namely adding rules and analysing cases.

4.6.1 Adding Rules

Previous RDR systems have reported figures of around 3 minutes per rule (Preston, Edwards & Compton 1994), and it can be seen in Figure 4-4.7 that this system has followed that trend, with the expert averaging 183 seconds (3 minutes) per rule. However, the times were not particularly uniform, with the expert sometimes taking up to 10 minutes, although this is expected to be largely due to distraction. Remarkably though, the logs showed 10 instances where the expert managed to add a rule in 40 seconds or less, with the fastest rule taking only 20 seconds to add. The large variations in time can be partly attributed to the expert's inconsistent knowledge of the domain, with the rules added quickly reflecting well known easily identifiable conditions and rules which take longer reflecting knowledge which is less well structured and requires the expert to articulate.

These results are testament to the extremely simple nature of knowledge acquisition in an MCRDR based system such as this. Reports of a rule a day were not considered uncommon in traditional rule-based expert systems (Compton & Jansen 1989) so it is

significant that the average time taken to add a rule can be reduced to 3 minutes, particularly in this domain where an extremely vast range of conditions are possible, showing again that experts tend to be particularly good at identifying the appropriate conditions. A very slight increase in the average time taken to add a rule can be seen as more cases are analysed. This is likely due to having to consider more cornerstone cases when validating but it is not a significant increase (160 seconds to 200 seconds) and is expected early in the system's learning as the amount of cornerstone cases to be considered grows very quickly, since almost every case requires rules to be added for it. It should be noted that after the 62nd case analysed the expert was presented with a new set of cases, which were considered of a higher quality and more difficult to classify (Tenni 2005). Further to this, many of them had complications introduced by the conversion process from the Mediflags database. These complications had to be addressed by the expert before attempting to classify, which may also contribute to the increase in time taken.

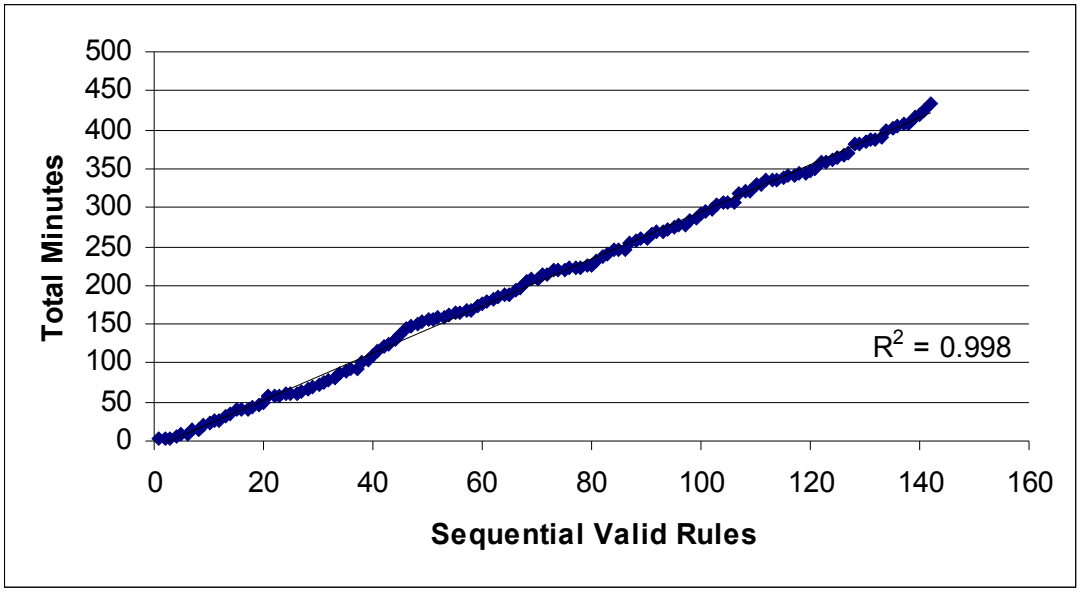


Figure 4-4.7: The time taken to add rules does not increase markedly as the knowledge base grows

4.6.2 Analysing Cases

It was found that the average time taken for the expert to complete a case analysis was about 10 minutes (621 seconds). This average extended over the entire 84 cases gives a total expert time taken as about 15 hours as reported earlier. Some cases were

done in as few as 2 minutes when no or few new classifications were required, although the process did sometimes reach over 20 minutes. Unexpectedly, it was observed that as more cases were analysed the average time taken to consider a case rose from 520 seconds to 720 seconds. It was expected this time would decrease as the system started to provide more pre-classifications for the expert. However, the source of the case data changed at the 62nd case analysed, with the expert commenting on the fact that the new case data was of a higher quality and harder to classify [had more classifications to be made] (Tenni 2005). With the invalid cases extracted, the 60th case corresponds to about the 40th valid case in Figure 4-4.8. It is expected that the time taken per case would begin to come down as the system reached greater levels of accuracy, since the system would be providing more classifications which the expert would only have to verify, rather than search for manually, although this supposition requires further testing to verify, which is documented in *5.2.1 Further Testing*. Another feature to note is that it is likely that the time taken would have been slightly higher if the expert did not have access to pre-classified copies of the data they were reviewing, since they would probably have spent more time searching the data for potential classifications.

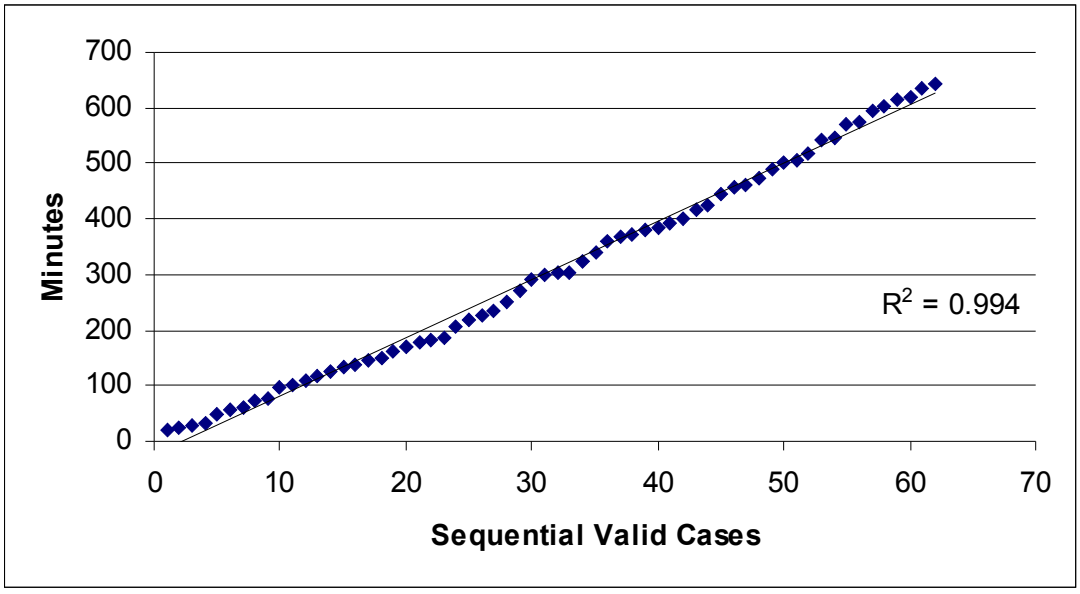


Figure 4-4.8: Cases took an average of 10 minutes to be classified

4.7 Test 7 – Cornerstones Seen

The results here are promising from a useability point of view, with the expert rarely having to consider cornerstone cases in the creation of rules, with the majority of rules having no cornerstone cases to consider. In fact the expert saw an average of only 0.42 cornerstones per rule (see Figure 4-4.9). What this means is that the expert should be able to add rules relatively quickly, with the time required to validate their rules being small. Clearly this is a good result, as it was considered possible that the number of cornerstones the expert would be presented with could be significantly increased in this inconsistent domain, especially considering the modification that was necessary for handling the difference list. This result shows that these incidents are still very low.

However, it should be noted that the average incident did increase throughout the test, from only 0.16 in the first 100 rules up to 0.68 in the next 95 as can be seen on the trendline. It is possible that with many more cases analysed this figure could continue to rise, but it is unlikely to significantly increase the time required to interface with the system. Furthermore, it has been observed in previous RDR based systems that the number of cornerstones does tend to rise quite steadily early on in the system's learning phase and slow down considerably after some level has been reached (Kang, B., Compton & Preston 1994). This suggests that the amount of time spent validating rules should plateau.

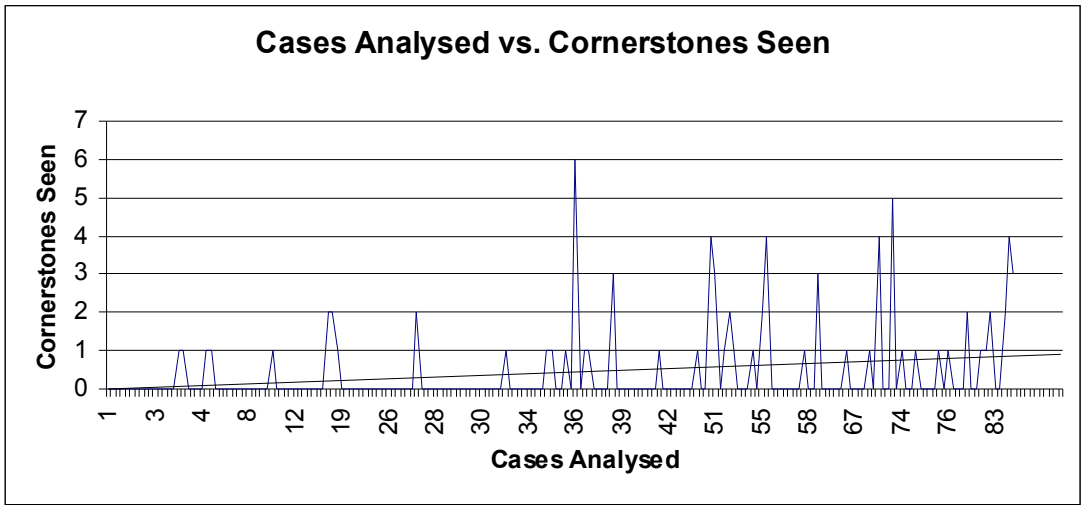


Figure 4-4.9: Number of cornerstones seen for each rule

4.8 Structure of the Knowledge Base

Always of interest when developing an MCRDR based KBS is the resultant structure of the knowledge base, with the details of this often giving some impression of the overall nature of the knowledge in the domain.

It can be determined from Table 4-4.1 that the structure of the knowledge base tree was extremely shallow and branchy, with the vast majority of rules stemming straight from the root. This is to be expected, especially given the system is still in training, because completely new rules are always added to the root and the vast majority of rules are expected to be new at this point. However, there is some indication of refinement occurring, with 53 rules being applied at the second level. One contributor to this second level of depth is incidences where the expert has made a rule and been unable to select a desired *not* condition, due to the restricted difference lists provided (see *3.2.2.1 Selecting Conditions*). This restriction often required the expert to add a refinement later to repair their incomplete rule.

Tree Property	Value
Average Depth	1.30
Depth 1	139
Depth 2	53
Depth 3	3

Table 4-4.1: Structure of the Knowledge Base Tree

The nature of the rules in the knowledge base is also of interest, with further support for the maintainability of the system shown in the fact that the average number of conditions selected in a rule was only 1.7 as can be determined based on the data seen in Figure 4-4.10. It is shown that longer rules, with 4 or 5 conditions are virtually non-existent and the most common number of conditions is 2. There are no rules in the system that have more than 5 conditions. The rules with zero conditions are explained when considering a rule being added with an incorrect conclusion (perhaps by accident, or by misunderstanding), then being given an all encompassing repair via a replacing or stopping rule with a different conclusion and no conditions. The fact there is 10 incidents of this occurring is further testament to the inconsistent nature of the expert knowledge. An alternative explanation is that a rule which fires

directly from the root of the tree but which has no conditions could be applied, then refined with exceptions later, but no such rule was found in the knowledge base.

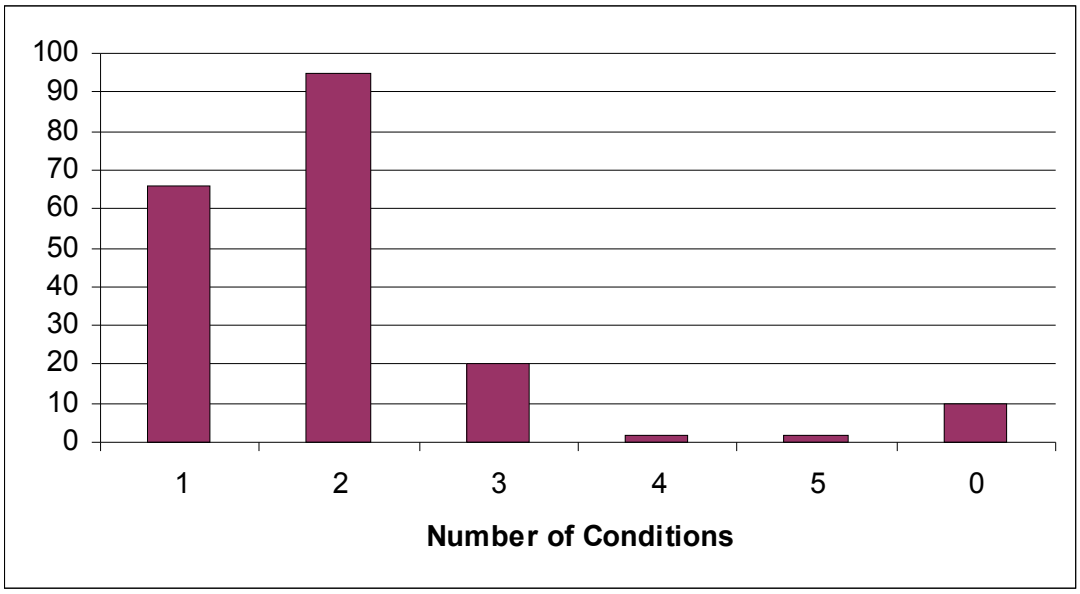


Figure 4-4.10: The nature of conditions for rules in the knowledge base

To get a more complete view of the knowledge base it is necessary to analyse what outputs the rules map to. With the knowledge base that was built in the process of this experiment only 85 individual conclusions were defined. When it is considered that every rule except stopping rules, of which there are 154, is linked to a conclusion it can be seen that there is 1 conclusion for every 1.8 rules, as can be demonstrated with the data used in Figure 4-4.11. It is evident from this figure that, although most conclusions are only used by one rule, some conclusions are used very often. In other words they have many different sets of conditions which can lead to them.

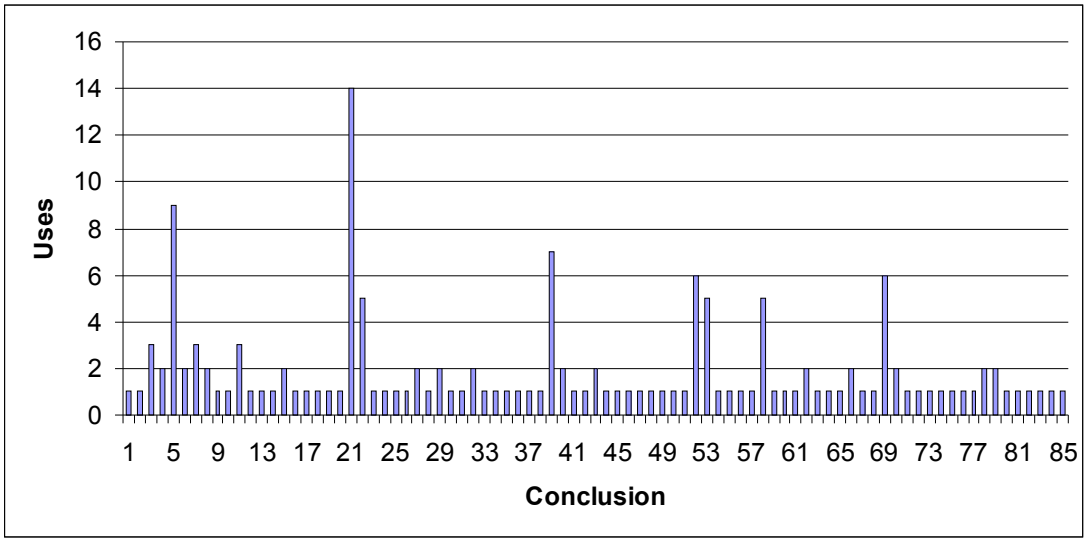


Figure 4-4.11: How many times each conclusion was used

4.9 Overall Discussion

The results shown above are very promising; suggesting that the system developed in this study using MCRDR is well suited to the task of expert classification in the medication review domain, and has not as yet suffered any adverse affects due to the poorly structured, inconsistent and multidisciplinary nature of the domain. Although it is demonstrated that the system is only at a relatively early stage in its learning, it can still be seen that it is conducive to reducing expert error by reducing the amount of classifications which are missed, consequently it is improving quality (or consistency) of service. This is expected, with the basic nature of validating the knowledge base through cornerstone case testing meaning that every time the expert accepts that the new rule they have created should apply to a previous case they are, in essence, admitting they missed this classification on that previous case. In the course of this experiment - 84 cases analysed and a total of 115 classifications - the expert was seen to do this 81 times meaning missed classifications by the expert are not short in supply for this domain, as was shown more clearly in Figure 4-4.5.

A crucial point for KBS tends to be the classification rate; how many classifications does it provide and how many should it have provided. This is where the results perhaps seem less adequate, with only about 60% coverage of the domain, and primary supporting evidence for this being anecdotal. However, there are several elements which need to be considered to put this result in context. Firstly, the amount

of test cases considered was relatively few. It is necessary in this domain to train the system on at least 200, even up to 300 cases before a rate in excess of 90% would be expected, especially when considering that the cases themselves tend to be incomplete (not all relevant data necessarily included). To quantify this issue further the time taken to produce this knowledge base should be considered. Only 15 hours of expert classification were required to build the 195 rules required to correctly classify the 84 test cases, implying that the further 120 test cases suggested as being enough to provide a classification rate in the order of 90% would be only an additional 22 expert-hours. Further, 60% is considered quite good in this domain, with the expert claiming it was the best software system he'd seen for classifying medication review cases (Tenni 2005), even at this very early stage in its learning. The next best being Cognicare (see *2.1.2 Previous Work*) which according to the field expert tends to produce false positives far in excess of its correct classifications (Tenni 2005). It should be noted that the system implemented in the course of this study produces very few false positives, and those which are produced are corrected in the form of a replacement or stopping rule so they will not occur again.

5 Conclusions and Further Work

5.1 Conclusions

Initial experimentation suggests that the proposed method using MCRDR can successfully represent knowledge where the knowledge sources (human experts) are inconsistent. The system is shown to have reached about a 60% classification rate with less than 12 expert hours and only 84 cases classified – a good outcome in the circumstances. The knowledge base structure does not show any major deviations from what would be anticipated in a normal MCRDR system at this stage. The maintainability of the system does not appear to have been adversely affected thus far, with the expert being faced with only few cornerstone cases during the knowledge base validation stage (see *4.7 Test 7 – Cornerstones Seen*), and the time taken to add rules being negligible (see *4.6 Test 6 – Time Spent*).

From a medication review perspective the system is seen to be capable of: providing classifications for a wide range of Drug Related Problems; learning a large portion of the domain of medication reviews quickly; producing classifications in a timely manner; and importantly, vastly reducing the amount of missed classifications that would otherwise be expected of the medication reviewer. It is expected that this system, or a future incarnation of this system, would be capable of achieving classification rates around 90% (Tenni 2005). If this figure is to be realised it is possible that this system would be capable of achieving all three major medication review goals stated in *2.3 Aim of Study*:-

- Reducing the amount of missed classifications
    - Thus improving the consistency (quality) of service
- Improving the speed at which medication reviews can be performed
- Improving the confidence of potential medication reviewers

It has already been noted that the amount of errors this system detected and repaired was significant, and the amount of errors was seen to reduce as the expert populated the knowledge base (see *4.4 Test 4 – Percentage of Classifications Missed*). It has also been observed that the amount of time taken to perform a medication review using this system should not be adversely affected, with the time taken to add rules being only small (see *4.6 Test 6 – Time Spent*), but it has yet to be demonstrated that this system does reduce the time taken to perform a medication review. However, it is expected that this goal is achievable if classification rates in excess of 90% are achieved, since the expert has commented that it takes less time to verify a classification the system provides than it takes to determine that classification manually (Tenni 2005). As for the final point it is anticipated that having a system such as this available to potential medication reviewers should increase their confidence in a similar manner as having another expert in the field there to check your answers might, since this system is designed to act as an expert in the field did.

Further experimentation with a larger set of cases is required in order to prove beyond doubt that a system of this nature is capable of operating at a high level in an inconsistent and poorly defined domain and that it is capable of achieving all three of the goals outlined above. However, there are no indicators in the results thus far which suggest problems may arise with further testing. Before undertaking further experimentation it is suggested that some or all of the enhancements outlined in *5.2 Further Work* be implemented, such that the system is capable of representing the kind of knowledge the expert wishes to model more closely. With these enhancements it is expected that the potential classification rate will rise, thus improving the usefulness and outcomes of the system.

In short, this system has demonstrated that the technique employed is well suited to a domain of this nature thus far and has furthermore demonstrated thoroughly that it is capable of improving the quality of service that the medication reviewer can provide. Further testing is strongly encouraged, since this system has, even at this infant stage, demonstrated that it is superior in many ways to any previous attempts at producing software based decision support to medication reviewers, and evidence suggests it will continue to improve with further testing and enhancement. This further work is discussed in the proceeding section.

5.2 Further Work

5.2.1 Further Testing

Results from initial experimentation indicate a longer and more thorough experiment should be considered. The system was clearly unfinished with its learning, and it is considered important to demonstrate that the system is capable of encapsulating more of the domain (up to or above 90%) and does in fact start to need new rules added only irregularly, before a solid tender for industry application of this type of system could be justified. However, in saying that, the evidence collected so far suggests these benchmarks are achievable and further testing is warranted, and it is unfortunate that the time constraints enforced in the course of this research and on the expert themself did not allow a longer test to be carried out.

An experiment of further interest would be to compare the levels of consistency against several types of reviewer, both with and without the system. This system would be targeted towards all medication reviewers, but has as yet only been tested with one expert in the field, who can be expected to miss fewer classifications than an entry level, or a standard level reviewer. He acknowledged that using the system had improved his consistency of classification but it would be expected that the improvement in consistency would be even greater with a medication reviewer of less skill and experience.

In a similar vein, the expert observed that he thought he had improved his consistency of classification when performing medication reviews without the system after his experience with using the system (Tenni 2005). It would be worthwhile to test a medication reviewer against a set of test cases, record how many classifications they found and how many they missed, benchmarking against the expert's performance. Then apply the same reviewer to another set of test cases, but using the system, again recording their performance. Then it would be possible to test them against another set of test cases without the system, to see if their experience with the system had actually enhanced their performance without the system.

To extend this experiment and to determine whether this system would be capable of improving the time taken to process a case for medication review it would be

desirable to have the experts' record the time taken to perform each step in the review process. When reviewing the cases without the system the reviewer would simply have to record how long they spent trying to find the classifications. Then, when using the system they would be recorded on the time taken verifying the results provided by the system, the time taken analysing the case for new conclusions, and the time spent adding the rules. It seems plausible that a system with a reasonable level of accuracy (~90%) would reduce the time taken per case, with the time taken to check a system-provided classification being less than the time required to find those classifications manually. However, as the expert is still required to check the case for classifications which the system may have missed there may be no significant increase in speed.

5.2.2 System Enhancements

The system could be more powerful and better encompass the domain by including some or all of the additional features mentioned below; however some of these features would require considerably more implementation efforts to apply successfully, while others are not so influential to have been worth considering for this prototype and as such they were not included in this experiment to date.

Time Series Data

The first of these features would be adding date field processing and functionality. Such that the expert would be able to define rules such as "increasing" or "decreasing" for things like Weight, Blood Pressure, or a Pathology result. Further still, they might define things like "recent" or "old", which check whether a result is older or younger than defined thresholds, newest, oldest, average and others. As the system stands it will fire on a rule that states "Creatinine > 0.12" even if the result which says their Creatinine level was 0.13 was taken 15 years prior. This is undesirable, with the meaning of the results varying across periods of time such that the expert may wish to define rules based on different types of results.

Better Text Searching

The second of these features would be the addition of a more powerful keyword searching facility. The current implementation of a keyword search is extremely basic, doing a primitive pattern match for the given words at some point inside the specified field. It would be preferable to allow more powerful searching constructs, such as wildcards, Boolean operators and string matching. Or, taking the concept further still, an RDR based pre-processing system might be derived which could allow for the rules on which search string matches which target string to be defined incrementally by a human expert, or even by a machine learning approach.

Defined Observations

It was identified earlier that the system suffered from poor handling of the patient observations, using the same model for these as Mediflags, which defined each observation as being a tuple of (name, string). The problem here lies in the fact that many of the observations should not be represented as a string. To solve this problem the system would have to have a data-type defined for each different observation type, either "String" for pure text observations, "List" for the nominal value ranges or "Number" for numerical inputs. Through these data-types the system would know what kind of operations could be applied to any given field. These data types can not be inferred through pre-processing as the dosages were, since the strings are written in natural language and are considered too complicated to parse reliably, so each observation's data-type would have to be defined manually.

Previous Conclusions

A feature that would be less influential perhaps, but still worthwhile, would be the ability to include previous conclusions as conditions in a rule. For example, a rule might be defined that states that if the patient previously had a critical DRP flagged on their last annual review, and they have this same conclusion again, then a more emphatic conclusion might be provided instead. An "even more critical" DRP, if you will, letting the reviewer know that this issue was not addressed properly at the last annual review. A feature such as this would also give researchers the ability to get a reasonable determination of how often issues go unaddressed after a medication review, and perhaps even demonstrate which GPs do not treat medication reviews seriously.

In the same vein it would be good to include the ability to create rules based on the conclusions the system has reached. For example, if the patient exhibited two or more DRPs that had the potential to cause a more serious combined problem, then the expert might define a rule based on exhibiting both those conclusions. The expert would be able to do this in the current system simply by making a new rule which fired in the event of all the contributing problems conditions being met at once, but this is less intuitive to the expert who sees the combined problem as being caused by having the DRPs, not as being caused by having all the symptoms which cause those individual DRPs. Further to this it is less user-friendly, with a lot of conditions needing to be selected to create the rule. An elegant way to implement this would involve having a separate structure for the conclusion-based rules which was processed after the standard inference had already been run, using the results of that inference as its "case" data. This solution would be restricted to one level of depth though, that is, the conclusions it produced would not then be further tested for conclusion-based matches again. However, it is considered unlikely that the expert would require this kind of facility to be multi-layered, and in any case they are still able to define the rules based on the conditions as mentioned earlier.

Explanation

To improve the level of trust that and engagement that the medication reviewers and the Pharmaceutical industry as a whole attribute to the system, it would be desirable to have some kind of explanatory facilities in the system. As it stands the user is restricted to seeing what conditions were satisfied in order for a given conclusion to fire, which is in most cases suitable because the expert knows what the conclusion is, and what those conditions mean. However, when the person using the system does not have the fundamental understanding of those conditions they will desire a more formal explanation (Richards 2003). This becomes more important when multiple users are using the system, with each user having different levels of understanding of the domain. To date this has not occurred and is as such less necessary.

Standardisation

Another issue that was touched on briefly earlier in this work was that of standardisation and consistency. It was observed that the knowledge acquisition

workload is increased when inconsistent nomenclature is allowed. To prevent this increased workload for the expert, it would be prudent to derive and enforce a strict scheme for the data input. A possible complication is that users may find it difficult to locate options which are not named as expected. To handle this it would be possible to implement another interpretive layer of hierarchy, essentially allowing the user to use their own preferred nomenclature, then defining within the system that their chosen nomenclature is synonymous to whichever standardised equivalent is selected by the system designers.

5.2.2.1 Overview

With some or all of the enhancements described above it is considered likely that the ease of use of the system and effectiveness of the system will be enhanced. With every new logical type of condition the expert is able to define, new rules become possible, which previously the system may not have been able to successfully or efficiently represent. If the system can be granted enough functionality and can be made to sufficiently represent the way the experts reach their conclusions for this domain it is possible that classification rates in excess of the current expert estimate of 90% may be achievable. If this were to be the case, the applicability of this type of system for medication review may be far reaching indeed.

6 References