# Chapter 7

# 7 Activity Reuses

The major goal of this thesis has been to see if it was possible to capture knowledge for one purpose and reuse that knowledge for different activities without the need to capture multiple copies of the knowledge tailored to each usage. It was found that the key to supporting different uses of the knowledge was a way of capturing or deriving abstracted knowledge, the relationships between the various levels of concepts and structuring these. Through the use of FCA it has been possible to take the simpler approach to derive, rather than capture, this sort of information from the behavioural knowledge that is relatively easily acquired using the MCRDR technique. This chapter is concerned with demonstrating how MCRDR/FCA functions as a complete system that supports a wide range of activities. It is not claimed that the range of activities included is exhaustive but that the range indicates that most common purposes to which knowledge can be put could be supported.

Firstly, the issues that affect which activity is likely to be selected by the user are briefly given followed by a review of each of the activities and how they are supported in RDR systems[1], particularly MCRDR/FCA.

## *7.1 The Issues Affecting the Activity Selected.*

The choice of activity will depend greatly on the situation and the users involved. Factors to be considered include the intentions, expectations and pre-existing knowledge of users. Pollack, Hirschberg and Webber (1982) see interaction as a "negotiation process". Salle and Hunter (1990) make the following observations. Users want to discuss, beliefs and justifications. Novices and experts have preconceptions which affect how they perceive the system. ES may need to detect the novices motivation, goals and strategies as is done by experts. Users are not always clear in defining goals and may actively try to hide them. MCRDR/FCA does not try to detect such things as beliefs, intentions, motivation, goals and strategies but caters for these differences between individuals and groups by providing a range of tools and ways of accessing the knowledge from which the user can select. This is seen to be more in keeping with a

---

[1] The discussion in this chapter on RDR systems in general is an expanded version of the paper "Richards, D and Compton, C (1996) Building Knowledge Based Systems that Match the Decision Situation Using Ripple Down Rules, *Intelligent Decision Support IDS'96* 9th September, 1996, Monash University, 114-126.

situated view of knowledge and the need for ES to provide feedback and user control, than a system that controls the interaction based on detected features of the user.

The following is a list from Salle and Hunter (1990, p.10-11) of the types of questions a system should be able to answer in addition to the traditional questions "What is the fault ?" and "what is the remedy ?".

1. What would happen if ....?
2. Why would X happen.....?
3. How could X be prevented.....?
4. What are the critical factors in X.....?
5. What other solutions would work....?
6. Isn't solution X as effective.....?
7. Which is the best remedy....?
8. Is X the right remedy.....?
9. Why did remedy X work.....?
10. Why didn't remedy X work....?

An inferencing or consultation system will answer question 7. A good explanation system should handle all the "why" questions (numbers 2, 9 and 10). A causal modeling system is concerned with questions 1,2, 3 and 4. Causal modeling could also be adapted to answer questions 9 and 10 if the remedies were made part of the system or a fault and a remedy model were developed. Questions 1, 5 and 6 are addressed by a ‘what-if’ exploration system. Critiquing systems answer question 8. A teaching system should let the user seek out the knowledge needed to answer the other questions which relate to alternatives and choices and improved understanding of the domain.

In this thesis, the activities have been broadly classified into one or a combination of two main types of modes of use: reflexive and reflective which can be seen as similar to the distinctions made in Chapter Six between a performance and explanation system. Using the RDR paradigm, consultation or inferencing, knowledge acquisition, maintenance and validation form the reflexive modes of activity. Critiquing, causal modelling, explanation, tutoring, ‘what-if’ analysis, hypothesis testing and student modelling form the reflective modes of activity. In first generation ES there appears to have been greater success with the reflexive modes. First generation ES were weak in supporting the reflective activities and revealed the inability of ES to adapt well to changed usage. Since the existing RDR approach to the reflexive modes has been demonstrated in other studies, such as (Edwards et al 1993), these activities were not changed in this study although the KA and validation activities have been enhanced by being provided in reflective modes as well. Also, since progress has been slowest in the reflective modes, the focus of this thesis is on this area.

The distinction between some activities is not clear and will depend on the situation. For example an expert and student may perform the same actions but be performing the explanation or tutoring activity, respectively. Consultation, 'what-if' analysis and critiquing can be used as inferencing activities, but the latter two can also be used for explanation. Causal modeling is treated as an activity in its own right but may also be used for explanation or tutoring. Despite the possible overlap of activities the main feature of each reuse will be given below together with a synopsis of how RDR handles each activity.

## *7.2 Reflexive Modes*

It was noted in Section 1.2 that most human, and particularly expert, action is reflexive. Rather than building or using abstractions and reflectively solving a given problem, experts tend to respond immediately with a solution. When reflection or in-depth analysis of the problem is performed it is often because the problem is stretching the expert to apply their knowledge beyond their current confidence or competence levels. In response to the reflexive nature of most human expertise, RDR were designed to provide simple and intuitive HCI which were introduced in Section 3.2.1 and are described further in this chapter in Sections 7.2.1 to 7.2.3. We look now at the reflective modes of consultation, KA, maintenance and validation.

### 7.2.1 Consultation/Inferencing

The most common usage of ES is to support the consultation activity, where the system poses questions to the user and produces a recommendation at the end. As Kang (1995) points out, many systems of this style failed to reach routine use because reliance on the user for input was difficult due to restrictions on the user's time and/or the user's inability or unwillingness to provide the answers. In contrast, initial RDR systems received input from information systems (IS) and output a report, which was then checked by the expert. In the main current implementations, the set of input cases may be historical cases, computer-generated cases or the user may enter hypothetical cases. However, the main role of the user is not to provide input but to evaluate the classification assigned by the system, see Figure 7.1.

An experimental technique more like the traditional consultation mode and designed to get the system to ask for more information, known as Interactive (I)RDR, was developed. IRDR used special rules with conclusions of the form "What is the value for rainfall ?" and rule conditions which had the value "unavailable". A more sophisticated system known as Error Recovery System (ERS) has been developed to support a help desk application (Kang et al 1997). This involves multiple inference cycles using a propose and revise strategy (Zdrahal and Motta 1995) and explicit rules that prompt the user for data. Kang (1995) proposed that the question rules be placed at the top level in an MCRDR tree and the questions be ranked and ordered according to whether they were satisfied by data in a previous inference cycle.

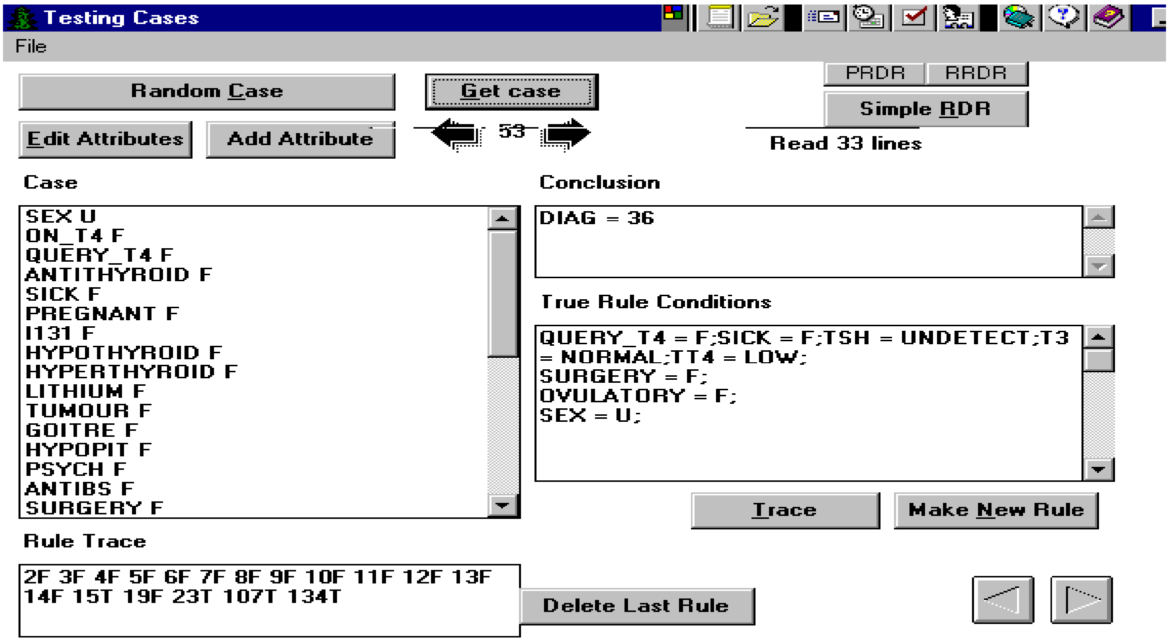


*Figure 7.1. An inferencing episode using XRDR for Windows. A case is retrieved and assigned a conclusion. The user decides whether to change the conclusion by adding a new rule.*

More recently a system has been developed which allows the user to specify which attributes should be prompted for when a particular attribute-value pair has been found. This system was briefly mentioned in 3.1.3, where the proposal to use a ML technique to determine in real-time which attributes were relevant was also put forward. The KADS solution to the SISYPHUS III problem (Jansen, Schreiber and Weilinga 1998) offered three possible ways of deciding which attribute to select. These methods include: random selection which may result in the user being asked the value for a large number of irrelevant attributes; ordering the attributes which is similar to the method implemented above and the use of an information-gain algorithm to determine the most informative attribute. The KADS submission employed all three option and used an information-theoretic algorithm similar to ID3 (Quinlan 1988) for the third option.

Previously the traditional consultation system-style has not been appropriate to the domains in which RDR has been used. An exception is the Lotus agricultural domain where a questionnaire screen as shown in Figure 7.2 was developed to provide a consultation style interface. However, to expand the suitability of RDR to a wider range of domains the typical “prompt user for input, output a recommendation” activity will be necessary but further work is postponed until the need becomes application driven.

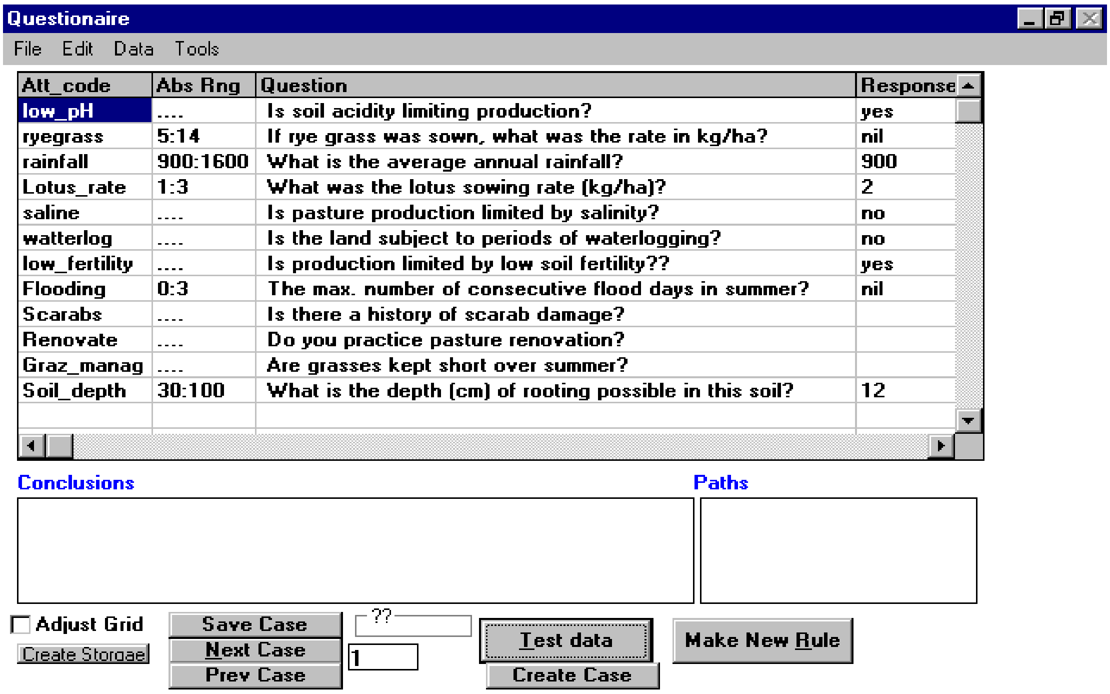


*Figure 7.2: The Questionnaire Screen in MCRDR which was developed for the Lotus Agricultural Domain is an example of a consultation style system interface provided by RDR.*

### 7.2.2 Knowledge Acquisition

In many ES, knowledge acquisition is a separate task from use of the knowledge base for inferencing. Puerta et al (1992) point out that this aspect is often forgotten and the concentration on reuse of problem solving methods has overshadowed the need for reuse of knowledge acquisition tools for the same knowledge domain. KA using RDR was described in Section 3.2.1 and is depicted in Figure 7.3. In the case of KA, there is no clear distinction between the building of the system and initial knowledge acquisition, the maintenance phase and even use of the system for reasoning. All three activities are performed as part of the KA process, although it is possible to perform inferencing as an activity on its own for the purposes of consultation as described in the previous section. As was shown in PEIRS, an RDR system can be set up quickly with minimal assistance from a knowledge engineer and then maintained completely by the expert. Since the same system is used for KA and inferencing it allows validation at KA time by performing an inference on the case which prompted the new rule.

Typically, one thinks of KA as a means to an end rather than an end in itself. It has been found that due to the ease of KA using RDR that KA itself has become the goal and has helped to capture and develop evolving knowledge in a number of applications covering agricultural (Hochman et al 1996) and medical domains (Edwards 1996, Feldman, Compton and Smythe 1989). The ability to uncover higher-level models using FCA from operational knowledge in an RDR KBS is particularly useful in domains where knowledge is emerging or in the common situation where it is difficult for experts to describe how they arrive at a conclusion. Also by viewing the knowledge in a number of ways and exploring the relationships that exist between the various concepts, the benefits of KA can be a significant source of new understanding of a field. In traditional systems where the domain and/or PSM models need to be described before KA can commence such an innovative use is not as easily possible. One of the main benefits of RDR is the ability to quickly build systems based on cases and learn about your domain without high resource requirements or commitment to get it right the first time. As noted in 3.3.1 this feature is shared with systems built on PCP such as Knowledge Support System Zero (KSS0) (Gaines and Shaw 1989) and Expertise Transfer System (ETS) (Boose et al 1989).

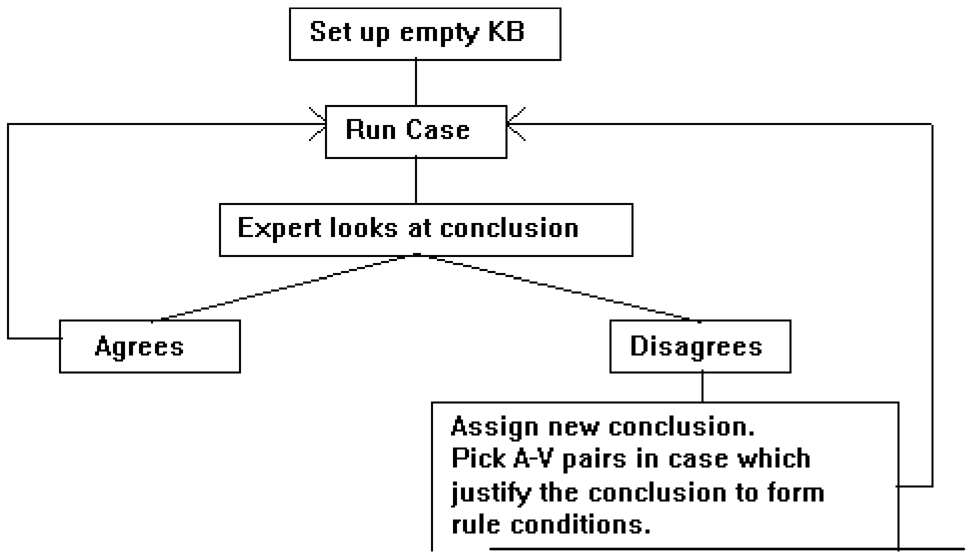


*Figure 7.3: The KA process in MCRDR/FCA.*

Some work has been done on providing KA as a critiquing activity that both critiques the proposed conclusion and the proposed rule. This is described in Section 7.3.1 on critiquing. The ability to perform KA in a reflective mode may result in a conflict in the goals of the system. KA using RDR is designed to be simple. Allowing multiple ways in which KA can be performed adds complexity to the KA process. This is true but in defense of offering the user KA in an alternative mode, the user can still perform KA in a reflexive manner as before but they now have the option of getting a second opinion if they wish to reflect on the conclusion and conditions for the rule they are proposing to add.

### 7.2.3 Maintenance and Validation

As described in Section 3.1, RDR were developed to handle the typical maintenance problems which occur with ES. RDR offer incremental maintenance and on-line validation. Maintenance and validation are not separate tasks in RDR but are part of inferencing and KA so the discussion of these activities has already been covered. Support for validation has been strengthened by allowing the user to perform KA as a critiquing activity and is described more in Section 7.3.1.

### 7.2.4 The Reflexive Modes in Action

To give the reader a feel for the reflexive modes of inferencing, KA and maintenance the screens that are used in MCRDR/FCA and how they are used are now given. The lack of distinction between these activities will be apparent as they are all part of the same process. These screens are virtually the same as in the current MCRDR for Windows implementation developed by Phil Preston and the MCRDR/FCA system is based on that code. I have added some extra options to some screens for the reflexive modes discussed in the following section.

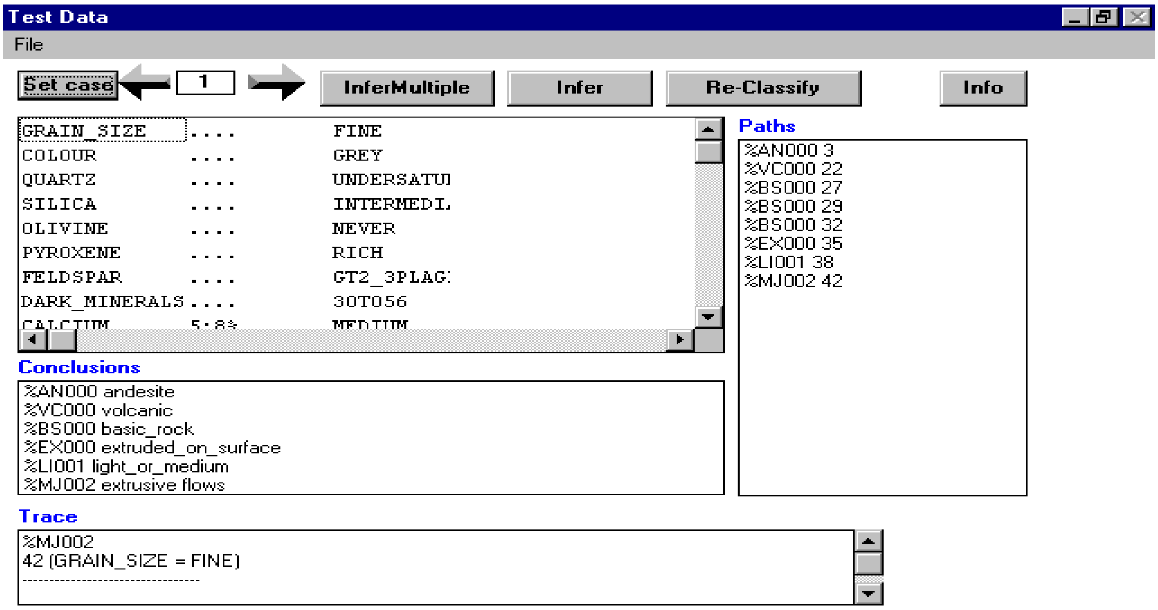


*Figure 7.4: The Test Screen in MCRDR/FCA which is used for inferencing.*

After the user has specified the KB and case file they wish to work with they are taken to the screen in Figure 7.4 which performs an inference on the last case accessed for this KB. The user may select another case to review if this is not the case in which they are interested. The user is shown the conclusion codes and a brief description of those codes in the conclusions list box. The codes and the numbers of the rules that fired are shown in the Paths list box. A rule trace that includes the conclusion, rule number and the rule conditions is shown in the Trace list box. The user considers the conclusions assigned to the case by the system and if they agree they select another case to review. If they do not agree they select the RECLASSIFY button, which takes them to the screen shown in Figure 7.5.

On the Reclassify Screen in Figure 7.5 the user is again shown the current case and the list of conclusions. The user can double-click on an existing conclusion code to move it into the rules to be stopped, click add to put a new conclusion into the "want" list box or specify which new conclusion should be a refinement (an exception) for a stopped rule. If the ADD command button is chosen, the user is taken to the CONCLUSION screen in Figure 7.6.

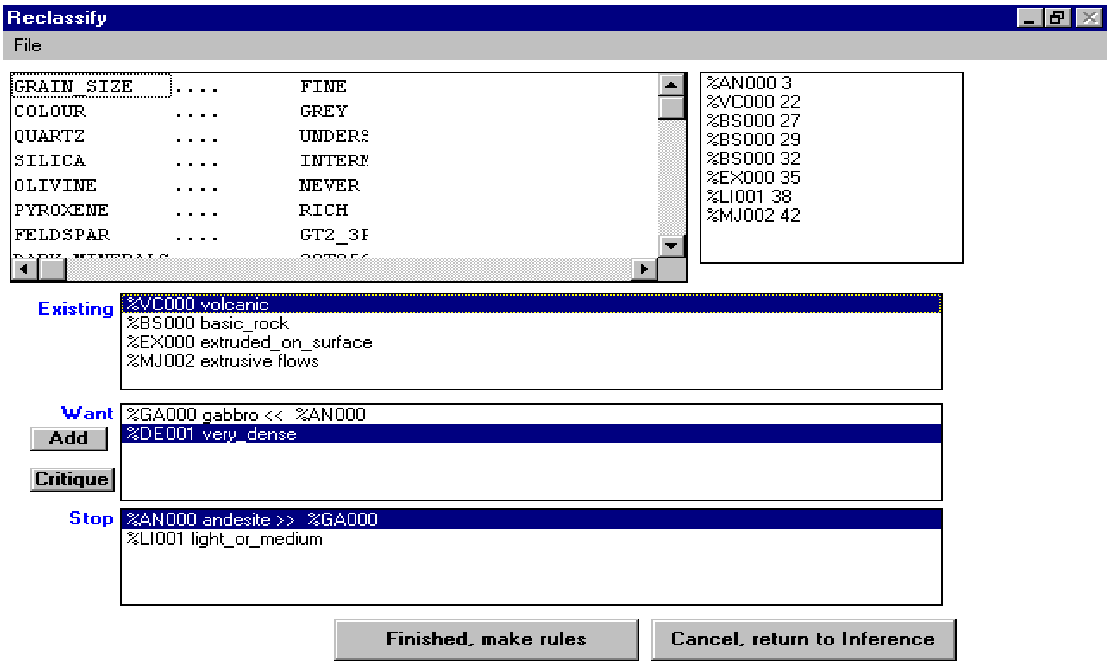


*Figure 7.5: The MCRDR/FCA Reclassify screen that allows the user to specify which conclusions should be added, changed or stopped.*

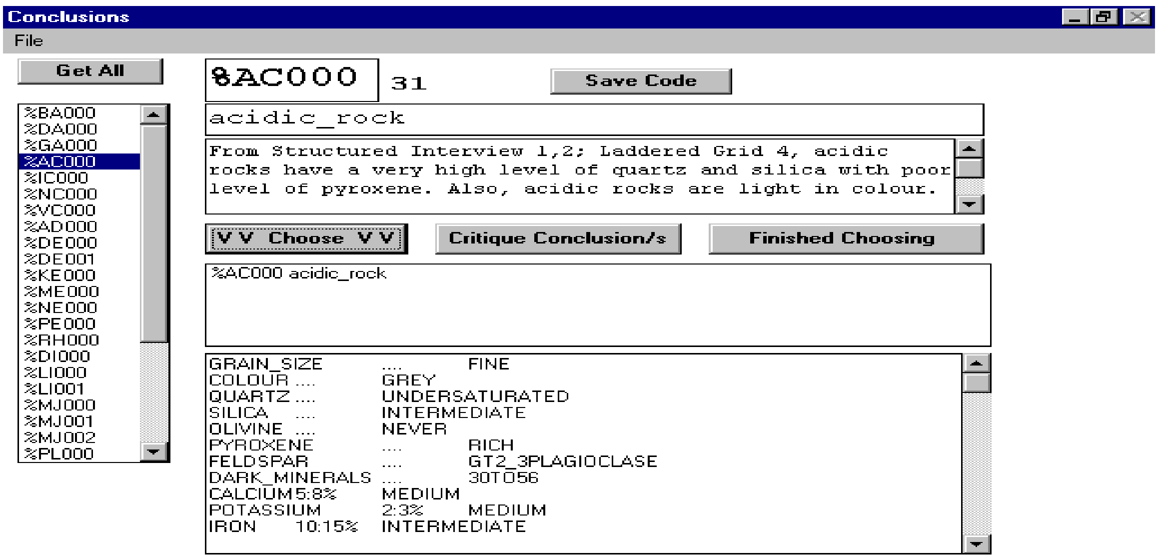


*Figure 7.6: The MCRDR/FCA Conclusion screen for selecting a new conclusion.*

The Conclusion Screen in Figure 7.6 is used to select which new conclusions to add for the current case. The user can pop-up a list of existing conclusion codes. When a code is selected from the list the short and long description are shown. The user can also add a new code by entering the new details into the code, short and long description sections on the screen. If a user wants to include a new or selected conclusion they click the CHOOSE button. When they have selected all the appropriate codes they take the FINISHED CHOOSING button which returns them to the Reclassify Screen in Figure 7.5.

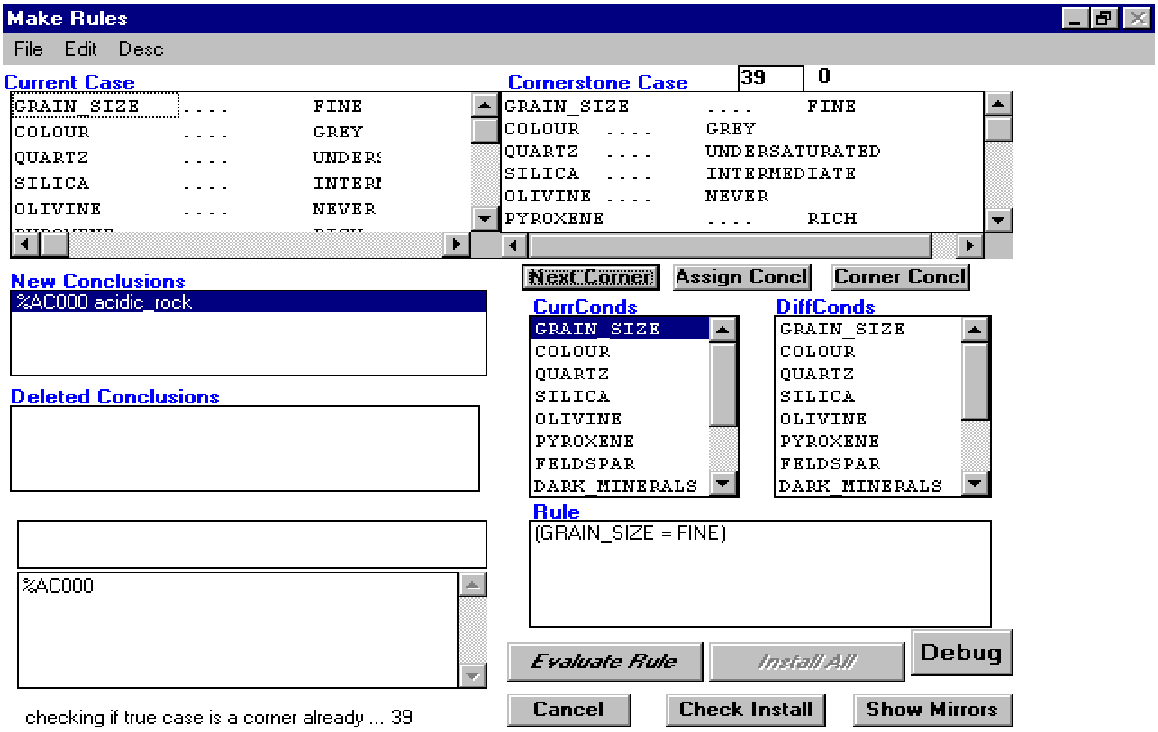


*Figure 7.7: The MCRDR Make screen for entering new rules*

In the Make Screen in Figure 7.7 the user first selects the conclusion for which they wish to make a new rule (remember multiple conclusions are possible). They then click on an attribute to be used in a condition for this rule. A pop-up window appears with a list of possible values for that attribute based on the differences between the current case and cornerstone cases. The cornerstone case being processed is shown in the window on the top right hand side. If the case is differentiated by the condition chosen the cornerstone list is traversed until a case is found that is not differentiated and it is put in the window. An additional condition must be added to differentiate this cornerstone case from the current case. When all cornerstone cases have been differentiated the INSTALL ALL button becomes active and the user can choose this option to add the rule. This may require patches in a number of locations. This process is repeated for each conclusion that was selected. When all conclusions have rules associated with them the user is returned to the Test screen in Figure 7.4 where they are shown the new set of conclusions for the current case. This provides immediate confirmation that the new rules have been added correctly. After checking the new list they may perform another inference on a different case and repeat the process outlined.

### *7.3 Reflective Modes*

Just as humans need to reflect on their knowledge, there are times when the user will need to reflect on the knowledge in the KBS. In the following sections each reflective mode of use is described together with a summary of how RDR handles each activity. The reflective activities considered are critiquing, causal modelling, explanation, tutoring, 'what-if' analysis, hypothesis testing and student modelling.

#### 7.3.1 Critiquing

Critiquing systems are an alternative approach to consultation style that may better suit certain domains and users. Salle and Hunter (1990) describe the critiquing system style as a system where the user can receive advice from the computer, have their own decision critiqued and be able to override the recommendation of the computer. The diagram in Figure 7.8 from Miller (1986) shows that a critiquing system differs from traditional ES that attempt to simulate an expert, in that it takes the physician's plan as input in addition to the patient description (the current case) and outputs a critique of the plan rather than a recommendation. Miller claims that such an approach is often more suitable, especially in medical domains, where it is inappropriate for the system to tell the user what to do. Critiquing does not offer a "do this" generic solution but provides a way of supporting the expert and providing that support in a way that is based on the user's thinking and style of practice.

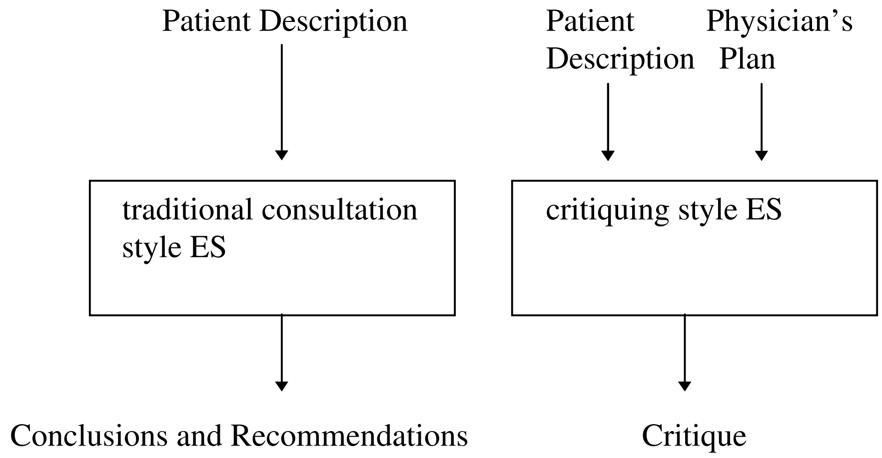


*Figure 7.8: A comparison of consultation and critiquing ES*
Adapted from Miller (1986)

Miller (1986) finds critiquing to be best suited to applications where there are a number of alternatives with different risks and benefits and where the knowledge is dynamic.

Woods (1987) sees the ability to provide such a facility as a means of assisting users to construct and test hypotheses by exploring the search space. This allows experimentation and adds "conceptualisation power" (Salle and Hunter 1990, p.13). Lehner and Kralj (1988) put forward the idea that the inconsistency that will result from a two expert problem solver, one human and one machine, will lead to better decision making. Miller (1986) sees critiquing as a way of helping the expert consider alternatives they may have missed and to keep up with current thinking in the field.

The benefits of critiquing style systems are not just limited to expert users. Studies by Kidd (1985) and Pollack, Hirschberg and Webber (1982) and Salle and Hunter (1990) all revealed that human to human interaction between experts and novices involved a negotiation process where the novice often proposes the solution, wants to know why something was recommended and may choose to reject the solution. Where this was not possible, interaction was found to be unsatisfactory (Coombs and Alty 1984).

Miller (1986) has implemented numerous prototype systems in a range of medical domains. Each prototype explores issues unique to its domain. The original system is ATTENDING which critiques anesthetic management plans. This system uses an Augmented Transition Network (ATN) but other prototypes use different architectures. A critiquing ES shell, E-ATTENDING, has been developed which has a data gathering component, production rule interpreter, expressive frames and a prose generator. A number of systems using this shell have been built. VA-ATTENDING critiques ventilator management plans and considers the use of strategic and tactical knowledge and how to resolve goal conflict. HV-ATTENDING looks at the pharmacological management of essential hypertension and is implemented as a *interactive paper* which allows an expert to disseminate their findings in a way that other physicians can access. PHEO-ATTENDING does not consider management but the medical area of *workup* which is more structured and constrained and looks at sequencing of tests and procedures to determine which diagnoses are correct. PHEO-ATTENDING addresses the issue of how to combine differing expert opinion. Conflict is resolved by breaking the problem into sub-problems and merging the related frame.

Another early critiquing system was ONCOCIN (Langlotz and Shortliffe 1983) which assists in the treatment and care of cancer patients. This system is of particular interest to this study because ONCOCIN is an example of how knowledge captured for one purpose may be reused for another purpose. The knowledge in ONCOCIN was not changed but a different system-style was offered to make the system more acceptable to users. The users, in this case oncologists, found it time consuming and annoying to have to perform a consultation with the system for each case. If the oncologist did not agree with the recommendation given by the system they had to override the conclusion and provide lengthy justifications for the difference, even if only minor. The adaptation to a critiquing system allowed the user to enter their own recommendation which the system would then analyse against its own recommendation using hierarchical plan analysis. Only variations of a prespecified size were reported by the system. The system became like another expert with which the expert could confer. This system differs from the system by Miller in that rather than actually critiquing the user's plans it explains it's own plan when it detects a mismatch. As emphasized by Langlotz and Shortliffe (1983) this approach demands a good explanation system to work and thus the findings are also relevant to explanation research. ONCOCIN also differs from the ATTENDING approach in the nature of the domain that was used. Oncology patient management can be handled by formalised oncology protocols. Miller classifies this domain as one that is based on objective criteria and therefore also an atypical medical domain.

More recent work that has followed on from ONCOCIN is the Therapy-Helper Project (T-Helper) (Lauderdale 1995) which has become part of the EON Project (Tu 1997). The Eon Project is concerned with protocol-based care. Online patient data and knowledge concerning protocols from a wide range of clinical information systems are used to determine when a protocol applies, to recommend a course of treatment and "to critique patient care that deviates from recommended practice patterns" (Tu 1997, p.1). These projects point out that critiquing requires reasoning and they are looking at developing new data models which contain the necessary temporal knowledge and new problem solving methods to assist in this reasoning process. T-Helper uses the Protégé-II methodology to configure the episodic skeletal-plan refinement (ESPR) method. The ESPR method is made up of three main subtasks which includes: using a *skeletal plan* to propose a *plan action,* problem identification and plan modification. Although a number of gaps in the methodolgy and tool kit are noted, T-Helper is an example of how Protégé-II can provide reusable ontologies.

Silverman (1992b) offers a survey of existing critiquing systems. Work in this area should shed light on the way decisions are made, the validity of decisions made by so-called experts and how decision making can be improved. Silverman (1992a) describes a system developed by the US army to critique decisions. A model of the human error process was created using social judgement theory and subjective decision theory. The two main causes of errors identified were missing knowledge and erroneous reasoning. He stressed the importance of judgement and substantive knowledge in the reasoning process and advocates the inclusion of deep knowledge.

To summarise the possible benefits and requirements of a critiquing system as offered in the various systems mentioned, a critiquing system:

- allows the user to enter the plan/recommendation,
- allows the user to test hypotheses and consider alternatives,
- the system critiques the user's plan or the system explains its own plan,
- the system provides a second expert opinion,
- allows the novice to negotiate a solution and
- includes deep and structured knowledge.

These points will be used to evaluate the critiquing activity offered in MCRDR/FCA which is what we look at next.

#### 7.3.1.1 Critiquing in RDR

To initially test out the usefulness of RDR for critiquing, the screen in Figure 7.9 was added to the MCRDR for Windows interface that presented the user with a case and let the user select the appropriate conclusion/s for that case. The system then determined its own conclusions and compared the two recommendations. The user was notified whether the user's conclusions were in agreement or disagreement. If a conclusion entered by the user was not one of the system's conclusions the user was asked to enter a rule, which should represent the reasoning behind the conclusion. If there were conclusions offered by the system that the user did not agree with, the user could look at the explanation, an extended trace as discussed under explanation in 7.3.4.1, and add a new rule to modify the rule which gave the incorrect conclusion or add a stopping rule so that the conclusion would no longer be given in those circumstances. In a traditional ES such an approach would not be as easy because activities, such as KA, critiquing, maintenance or inferencing, may be in different systems or performed by different users.

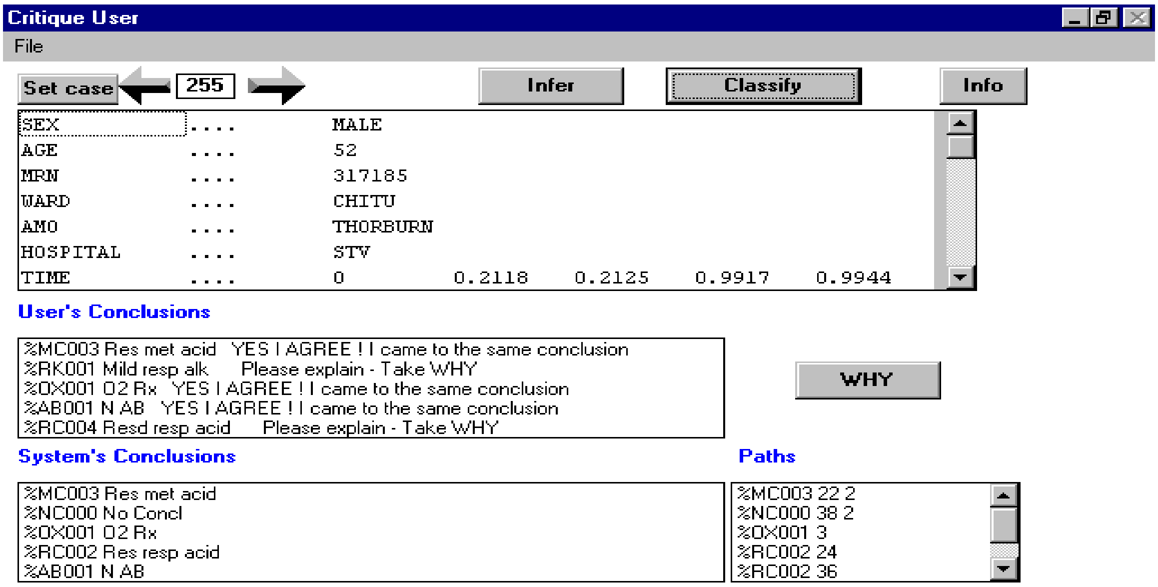


*Figure 7.9: Critiquing Screen 1 in MCRDR/FCA.*

Critiquing is offered at the conclusion and rule levels, in keeping with the KA approach offered by RDR. When a user decides to assign a new conclusion (the first step in developing a new rule), the user may select a conclusion on the RECLASSIFY screen and click the critique button to have the new conclusion evaluated against all other paths (rules) that gave the same conclusion. If those rules are inconsistent with the current case the user is notified of the anomaly. In Figure 7.10, using the data from the SISYPHUS III (Shadbolt 1996) experiments, I have attempted to add a rule that states that a particular rock, already classified as volcanic, should also be classified as plutonic. These conclusions are mutually exclusive, therefore it is inconsistent to assign both conclusions to the one case. In Figure 7.10, this inconsistency has been detected by the system. By showing the user the rules in conflict they can either change their conclusion, select an existing rule to add a refinement to, add a new rule higher in the tree or add a stopping rule to stop the conclusion being reported for this case (and any others with the same features).

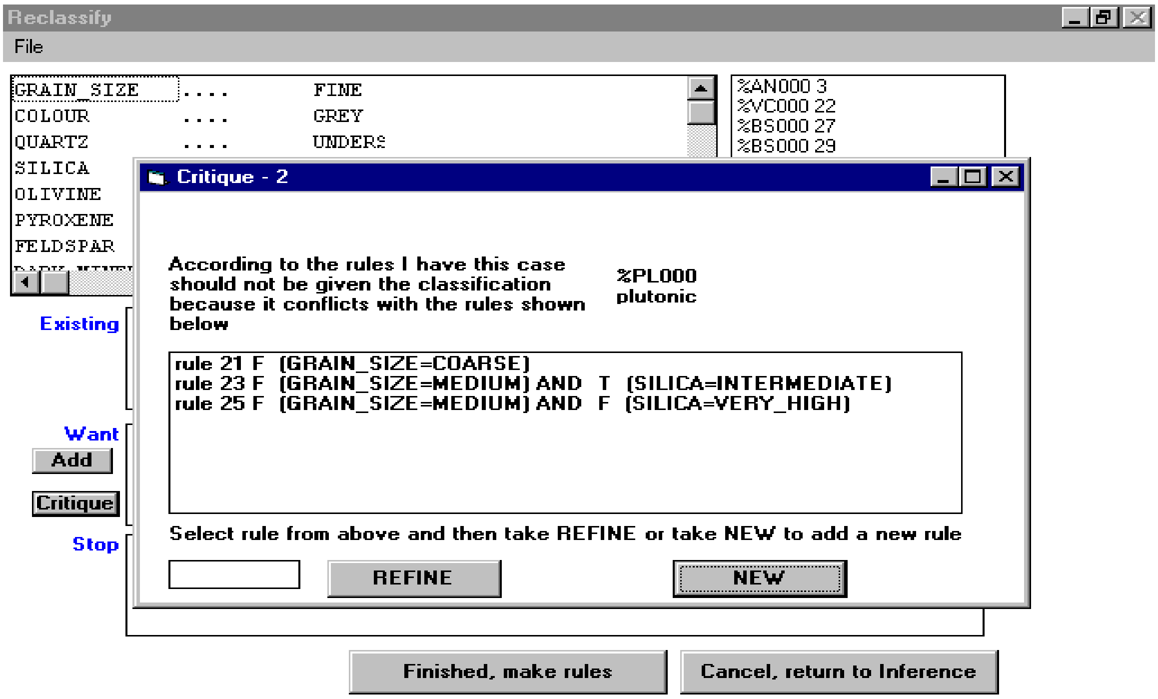


*Figure 7.10: The critiquing screen 2 in MCRDR/FCA for Windows.*
*The user is shown any pathways that are inconsistent for the specified conclusion and current case.*

Once a conclusion has been decided on, there is also assistance in forming the rule. When the user enters a new rule on the Make Screen they may click the EVALUATE RULE command button as shown in Figure 7.7 to take them to the Test Pathways screen shown in Figure 7.11. The proposed rule is placed in the Path 1 text box. The user is able to compare the proposed rule against existing rule pathways using the nearest neighbour algorithm described in 3.3.2.1 to provide a list of proximity scores with other rule pathways. Alternatively, they may chose to be shown where the new concept fits into the hierarchy of concepts derived using formal concept analysis. With this option the user is presented with a listbox of all the other pathways in the knowledge base that are matches, sub or superconcepts. If the new rule is identified with concepts that seem inappropriate this is a warning to the user that the knowledge in the new rule or an existing rule is incorrect. For this purpose only the intensional definition of the concepts derived using Wille's technique are used because as discussed in Section 3.3.1.3 the extensional definition is too restrictive when we are dealing with new concepts. It is feasible to compare two rules (objects) to see if they have similar properties but it is unlikely that the set of objects would be the same or similar since many objects will not yet have been defined.

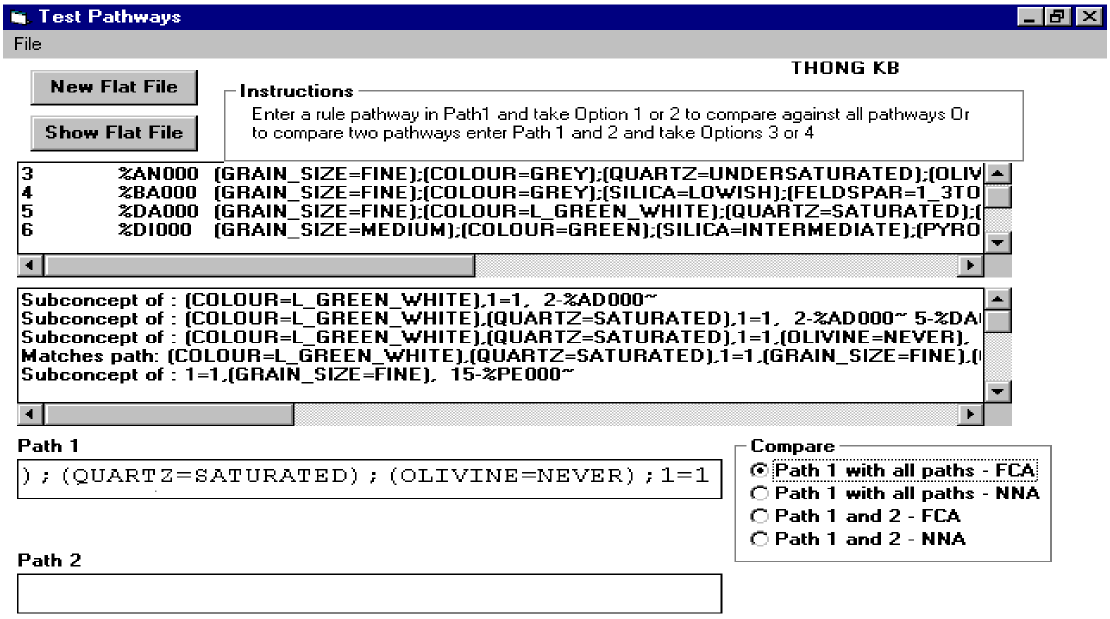


*Figure 7.11: The Test Pathways Screen for comparing rule pathways in MCRDR/FCA.*

If the user is satisfied with the new rule they return to the Make Screen in Figure 7.7 and take the INSTALL ALL button. Alternatively, the user may decide to modify the proposed rule or to modify a different existing rule that was identified as needing further refinement. In this way critiquing assists the user with KA and validation of the KBS.

The two main critiquing features in MCRDR/FCA are the conclusion critiquing screen (figure 7.10) and the test pathways screen (figure 7.11). Using the description of critiquing systems given earlier, let us see how MCRDR/FCA measures up.

- allows the user to enter the plan/recommendation. MCRDR/FCA allows the user to enter the conclusion (the recommendation) and see whether this conclusion is inconsistent for the current case based on other rules that reach the same conclusion. This allows the user to reconsider their recommendation. If they decide to keep that recommendation they can see what rule/s are in conflict and may need a refinement added to cater for the current case. Alternatively, the user may detect that the situation which the current case highlights has not already been identified and in this way missing knowledge may be identified and added.

The main limitation of this approach is that there may be other conclusions that are similar, but not identical, which would also reveal inconsistent or missing knowledge in the KBS. It is hoped that the next step where the proposed rule is critiqued will catch such situations. Another option is to allow the user to identify which conclusions are related, which could be stored in a table, and all related conclusions would be used in the conclusion critiquing screen. The use of conclusion families are a natural way of implementing such an approach.

- allows the user to test hypotheses and consider alternatives. The test pathways screen is a way of seeing what other rule pathways are similar to the proposed one. Each pathway represents an alternative. By considering the alternatives the user can propose new rules and in this way test out new hypotheses. Moving beyond the critiquing activity, the user can alter attribute values in the case to perform 'what-if' analysis or view parts of the knowledge base by selecting various foci of attention as alternative uses of the knowledge in which hypothesis testing and scenario analysis can be performed.
- the system critiques the user's plan or the system explains its own plan. The type of critiquing offered by MCRDR/FCA does not quite fall into either of these categories but offers aspects of both. The user's plan, in the form of a conclusion or rule conditions is used to determine which existing paths in the knowledge base may be of interest to the user. Figures 7.9 and 7.10 both provide feedback to the user on whether the conclusion proposed appears appropriate. The rule pathways provided in figure 7.9 and 7.10 form the explanation of why the system does not agree with the user's conclusion, thus the explanation is in terms of the system's plan, not the users.

  The main limitation of the type of critiquing offered by MCRDR/FCA concerns the nature of the explanations given. As described later in the explanation section, rule traces are often not terribly easily understood and a natural language interpretation would be more user friendly.
- system provides a second expert opinion. It is envisaged that critiquing in MCRDR/FCA would be used when the expert is unsure of the conclusion or rule conditions they are entering and wants to compare their own solution with those already found in the system. In this way the system becomes the second expert. If more than one expert is using and maintaining the system the ability to critique new rules will be particularly beneficial because the system will provide a repository of shared knowledge that each expert can consult. Without the ability to critique a

solution, inconsistencies are more likely to occur as it would be more difficult for one expert to determine what rules another expert had entered. The RE work described in the previous chapter could be employed to further strengthen this feature by annotating rules with the source of the rule and by using the FCA modelling tools to compare expert models.

- allows the novice to negotiate a solution. It was noted earlier that novices often prefer to be involved in the recommendation process. The test pathways screen was originally developed as a means of understanding the relationships between concepts (using the FCA option) and the closeness of concepts (using the NNA option). When the novice is given a recommendation they could use the rules that fired as the pathways which they want to compare to see alternative pathways that are similar. Based on the alternatives the novice can decide which recommendation they prefer.

  The novice is also able to use screen 7.9 as a way to test out their own solution. Critiquing would be particularly useful for teaching as the student could propose the solutions, without having been influenced by the system's conclusions, and then have the two compared. To improve support for the novice/student the full range of critiquing tools which allow a conclusion to be added or a rule to be proposed needs to be included in a combined critiquing-what-if activity that allows any changes made to be undone afterwards.

- includes deep and structured knowledge. The extra information critiquing offers adds value to the recommendation and adds a depth of knowledge not available in conventional MCRDR inferencing. The FCA option on the test pathways screen provides greater depth of knowledge by showing abstracted concepts not obvious directly from the rule pathways. The FCA option also provides the structure of the knowledge by showing the sub, super and matching concepts.

In conclusion, MCRDR/FCA allows the user to input their own recommendations and have the system either state that it agrees or show how the system would reason about the same case. The user is able to override the system's conclusion but is offered an explanation of the system's plan so that the user may contemplate whether they missed an important consideration. The MCRDR/FCA abstraction hierarchy provides some understanding of the relationships and structure of the knowledge while the nearest neighbour algorithm provides a measure of closeness between concepts. Without the addition of these two features to MCRDR it was not possible to offer critiquing beyond stating whether the system agreed or didn't agree. While critiquing is now satisfactorily supported further work on providing a more user-friendly presentation of the knowledge in the system should be provided to assist the user in evaluating the system's plan against its own.

### 7.3.2 Causal Modelling

In this section we look at the importance of causal modelling as an activity supported by KBS. Scott and Weckert (1995) consider causal explanation to be a means of offering justification by providing the link between the input from the user to the output of the system. This output may be the final conclusion - the what and how, or the reasoning behind the system asking a particular question - the why. White and Frederiksen (1989) found that causal models could be used as intelligent tutoring systems in the fields of science and engineering. This discovery was in response to evidence that many physics students could not solve qualitative problems and didn't really know how to apply the formulae they had been taught. Traditional approaches to problem solving had concentrated on quantitative methods where the variables in the problem needed to be matched to equations, similar to the use of heuristic pattern matching in ES. White and Frederikson (1989) hypothesized that if students were able to learn the domain in terms of qualitative causal models they would gain a more robust and thorough grasp of the underlying theory and its application. They saw the qualitative causal model as a better starting point for learning, which could later be improved by the introduction of more complex qualitative models and progress onto quantitative models. They argued that other research has shown that experts progress in their problem solving in a similar sequence. Their work is discussed further in Section 7.3.4.

A lot of interest has been generated in causal modelling in the Stanford "How Things Work (HTW): Knowledge-Based Modeling of Physical Devices" Project. This project aims to develop:

- *"representation techniques for encoding knowledge about engineered devices in a form that enables the knowledge to be used in multiple systems for multiple reasoning tasks;*
- *reasoning methods that enable the encoded knowledge to be effectively applied to the performance of the core engineering task of simulating and analyzing device behaviour" (Torsten_Heycke 1994).*

The first aim is clearly interested in the reuse of causal knowledge across tasks, domains and applications which is similar to the goal of this thesis except that this thesis is concerned with the reuse of more than causal knowledge alone. The project offers tools for simulating and analysing the behaviour of electromechanical devices to support design and redesign of such devices. The system being developed is known as the Device Modeling Environment (DME) which they refer to as a "designer's associate". As part of this project a common Compositional Modeling Language (CML) and the Causal Functional Representation Language (CFRL) (Vescovi et al 1993) have been developed. CFRL is used to define the intended device functionality. As was found in the HTW Project, causal modelling may require special knowledge representations and processing. Causal modelling may also require the knowledge to be presentated in a different way to conventional text explanations. Lehner and Kralj (1988) began their research with the theory that people evaluate the worth of a decision based on experiences they take with them. People expect to see what they are used to. This involves a complex model of causations. They performed an experiment to test cognitive consistency and modelling. They found that users see the interface as causal evidence of how an ES is internally working. They argued that the user interface gives cues to the user - such as covariance, continuity, temporal order and similarity. Following their line of reasoning, it is important to present knowledge to users in a way that fits their thinking about the subject. Explanation is typically thought of as words describing why a question has been asked or why a conclusion has been made. Users' expectations and backgrounds together with the user interface will affect the way they are able to use a system (Lehner and Kralj 1988). In many professions such as medicine and electrical engineering a diagram is a more natural way to consider causality. The explanations given by Lee and Compton (1995) are in such a form and are known as a CMOD model. Referring to a graph which showed how the final conclusion fitted in with the known findings Clancey comments: "for the purpose of teaching, this graph could perhaps be the best way to reify the process of diagnosis" (Clancey 1986, p.56).

| |
|---|
| Get a rule from rule base<br>Extract all items and conclusion from the rule<br>For each object, test causal links from the rules<br>If causal link is not found, ask expert to add<br>Causal effect (+/-signs) computed for each link<br>Draw a CMOD model |

*Figure 7.12 Developing a causal model from heuristic rules (Adapted from Lee & Compton 1995).*

Lee and Compton (1995) have reused heuristic knowledge so that it provides causal explanations. They argue that causal models provide more useful explanations, particularly for learning, than those missing causal links but such explanations are difficult to construct. While it is easy for the user to assign a conclusion and enter rules that differentiate one case from another it is exceedingly more difficult for users to explain the underlying theories required to enter causal models. The work done by Lee and Compton takes the heuristic model and uses an algorithm, summarised in figure 7.12, which seeks to make explicit the causal links implicit in the heuristic rules. It has been found that it is not possible to determine all links and some collaboration with the expert to determine causation has been used. The model is also useful for validation as it is checked for consistency each time a heuristic rule is added. If links are missing or wrong the model is modified automatically or with user assistance.

Webster (1995) also looked at the relationship between models and heuristics and whether the causal links that Lee (1996) was deriving, often using expert opinion, could be derived by some form of machine learning. He investigated several ML techniques including AQ, C4.5, FOIL and MIS. The propositional learners, AQ and C4.5 were found to be unsuitable because they are aimed at finding minimal solutions to solve a classification problem. A rule such as IF a, b and c THEN d will provide a heuristic but does not give us any real idea about causality. The Inductive Logic Programming (ILP) techniques such as FOIL(Quinlan 1990) or MIS (Shapiro 1981) were also found to be inappropriate because they required positive and negative examples and background knowledge. The method that Webster used to derive causal models from heuristic rules hinged upon using the conclusion as the cause and condition attributes as the effect. The direction and sign were determined for each rule. Attributes that had a normal value or were in a state of equilibrium were ignored. This simplistic approach worked were there was only one pathway from cause to effect. Where there was more than one pathway the ambiguity needed to be clarified to determine the actual causes. This was done using the concepts of transitive and euclidean links. The assumption was that if a transitive or euclidean link exists then that is evidence of ambiguity. Where ambiguity was detected the expert was asked about the links and was able to forbid or confirm a link. This approach was tested with rules from the Garvan KBS and another KBS developed for the health fitness domain. It was found that just querying the transitive links worked reasonably well. This system was greatly limited by its formatting requirements so that it could be input into prolog. Webster’s work has concluded that machine learning alone is not sufficient and that a collaborative approach using ML and expert input is required. This is in keeping with the findings by (Shiraz and Sammut 1998, Webb and Wells 1995) where an expert interacts with a machine learning system in the KA process.

Mahadidia (1994) has developed an ILP technique that he calls 'unfolding learning' to derive causal models from rules not in Horn Clause form such as data in A-V pairs. He discusses a system called JUSTIN which derives causal explanations in neuroendocrinology. JUSTINand the solutions proposed above by Lee and Compton (1995) and Webster(1995) all rely on expert involvement to resolve conflicts in the causal models they derive. Unfolding learning has been developed to overcome this limitation and to deal with non-Horn clause logic. B $\wedge$ Bar Effects are used to generate negative clauses ($U_{ni}$), B $\wedge$ Causes are used to generate positive unit clauses ($l_i$), where B is the background knowledge of an existing causal qualitative model in the form of a horn-clause logic program. Each negative clause is combined with one or more positive clauses to form an hypothesis. Different suitability criteria, such as adding the minimal number of clauses, can be applied to restrict the number of hypotheses. In this way a too specific theory is generalised in keeping with Shapiro's MIS framework (Shapiro 1981).

Causal modelling has not been investigated in great depth in this thesis. This is partly because the PhD work of Lee (1996) has addressed this knowledge usage and partly because this activity is not applicable to all domains. However, the line diagrams produced using FCA are similar to the connected graphs to which Clancey (1986) refers. As is shown in the dialogue presented with the line diagram in Figure 8.21 for the Blood Gases Domain (where the domain can be described as causal) it is possible to infer that certain conditions are positively or negatively causally related to one another. For example at one node we may have a particular condition occurs when BLOOD-PC02 is HIGH and BLOOD_BIC is LOW. At another node we may have BLOOD_PC02 is LOW and BLOOD_BIC is HIGH which results in the opposite conclusion. It would appear that the two attributes are inversely related and that when one is elevated the other goes down. Line diagrams and typical causal diagrams can both represent the ideas of sources, sinks and flows with directional lines but the line diagram does not include the "+" and "-" annotations which indicate the nature of the relationship. However, since each attribute is actually an A-V pair indicating the value of the attribute it can be determined if the flow is increasing or decreasing. To make the research in this thesis more complete it would have been interesting to implement Lee's algorithm and to look at possible modifications to the FCA line diagram to support causal relationships such as specifying the nature of a link. However, this is not a small task. As Buchanan and Shortliffe (1984) point out, providing an explanation of causal knowledge requires more than stating the rule, which can suffice for other types of knowledge. Thus, on the one hand it can be argued that causal modelling may be performed to some extent in MCRDR/FCA using the line diagrams. On the other hand, there has not been sufficient investigation of this activity and its inclusion into MCRDR/FCA to claim that causal modelling is adequately supported and achieving this is left to further research.

### 7.3.3 Explanation[2]

Unlike causal modelling which is not applicable to all domains, explanation is an activity that must be supported in all KBS. ES need to offer explanations because of imprecise domains and the use of heuristics (Lee and Compton 1995). Verification is not enough. ES need to justify and be accountable (Swartout and Moore 1993). Through explanation the user can see if the recommendation is valid. Explanation is a driving force in second generation systems. First generation ES tended to paraphrase rules. This

[2] Much of sections 7.3.3 and 7.3.4 are included in a paper submitted to PKAW'98 titled "A New Look at Explanation for Knowledge-Based Systems".

meant some of the important parts of the knowledge were left out and users found the rule trace format unnatural. There was generally no distinction between knowledge roles (Clancey 1984 was an exception). Some rules may have been used to handle the user interface or control inferencing but they were still used in the trace given to the user. The problem of unnatural dialogues required better technologies for explanation generation. Second generation ES explanation focuses on context, goals and actions, methods and justification (Swartout and Moore 1993).

There has been much criticism of the usefulness of the rule-trace style explanation. Chandrasekaran, Tanner and Josephson (1988) expound the need for control strategies and deep models in support of explanation. Expert systems aim to handle knowledge rather than simply data. They argue that most KR are at too a low level of abstraction to provide adequate explanation. Swartout and Moore (1993) agree that the problem of lost knowledge can be addressed by the knowledge representation.

Chandrasekaran, Tanner and Josephson (1988) describe 3 categories that cover what they consider to be the best ideas that have emerged on explanation in first generation ES. The first approach was embodied in the MYCIN ES where the user could ask WHY and receive a trace of rules. The second approach was developed through NEOMYCIN (Clancey 1986) which is able to give strategic explanations because the knowledge has been abstracted into different levels. These first two approaches do not provide justification. The third approach is found in Xplain (Swartout and Moore 1993) where the system details are input to an automatic programmer which generates the explanation. Through the use of the automatic programmer the relevant and related knowledge is used to build the rule base. A trace is kept which stores the initial knowledge which may be later retrieved for explanation purposes. Chandrasekaran, Tanner and Josephson (1988) define 3 possible types of explanation. Firstly, the system can explain how the raw data matches the goals. Secondly, justification may be given through the description of the KB contents itself. The third type is the control strategy explanation. They see the issue of problem solving types as necessary for explanation and suggest the use of generic tasks (GTs) to provide explanation types one and three. They summarise problems into four generic tasks: classification, state abstraction, knowledge directed information passing or design by plan selection and refinement. They claim that generic tasks act as higher level building blocks. Chandrasekaran, Tanner and Johnson state that there is much support in the explanation literature for providing an explanation and justification of the reasoning processes and PSM.

This claim has interesting implications for the current work in MCRDR where the representation and inferencing strategy that is used for classification has been used for a configuration task with the key difference being that the latter requires more knowledge to be supplied and additional inference cycles. Since the systems reasoning processes are the same for classification and configuration an explanation of this process does not appear to be terribly important. This finding is potentially good news for explanation research because it allows the effort to focus on explanation of domain knowledge. An argument explored further in Section 7.3.4.1 is whether it is necessary to understand the system's reasoning since we often don't understand or use each other's reasoning and since humans tend to offer justifications directed to the audience rather than explanations.

Swartout and Moore (1993) discuss five aspects of a good explanation:

1. Fidelity - how accurate is the representation ? Fidelity is aided by a simple inference engine.
2. Understandability - is the content and context understandable? The components to be considered are terminology, user sensitivity, abstraction into different levels, summarisation (the amount of detail offered in an explanation), the possible system perspectives, linguistic competence and feedback.
3. Sufficiency - is there enough knowledge to provide explanation in different contexts? This will require knowledge to be stored that allows explanation about the system's behaviour, justification, preferences, domain explanations and terminology definitions.
4. Low construction overhead - how time consuming and difficult is it to build explanations ?
5. Efficiency - what will the system's response time be like, will it take a long time to generate an explanation ?

These five aspects of a good explanation will be used as the basis of an evaluation of the RDR approach to explanation and is covered in 7.3.4.1.

Similar to the first generation ES approaches to explanation described by Chandrasekaran, Tanner and Johnson (1988), Swartout and Moore (1993) describe the four main architectures found in second generation ES explanation.

1. The approach offered by Clancey in NEOMYCIN used metarules for problem solving strategy and demonstrated how control knowledge could be abstracted.
2. Chandrasekaran offers another perspective through the use of GTs. The approach is founded on the premise that problem solving methods have similarities that can be grouped and abstracted. Each GT has its own language and its own explanation routines. Explanation nodes may be suspended, rejected or accepted. Backtracking is possible by returning to suspended nodes. The grouping of the knowledge is

beneficial and the use of a generic method improves coherence. The drawback of the approach is that GTs are highly tailored to a particular task which means they have high construction overheads and pose a fidelity risk because if the problem solving method is changed the explanation method must also be modified.

3. Swartout calls his technique Explainable Expert Systems (EES). The first step is to develop an abstract high-level specification to pass to the automatic programmer. The Design History keeps the decision steps and is used for explanation. The Interpreter executes the ES code. EES is able to supply the reasoning behind actions. The automatic programmer means links are automatically derived, unlike most approaches. In an EES KB three types of knowledge are held: terminological (using a KL-ONE type structure), domain model and problem solving knowledge. Reformulations are used to adjust goals if no plan is found. There are three types of reformulations: covering, redescription and individualistic. The argument behind EES is that in order to provide adequate justifications some of the knowledge available at acquisition time needs to be programmed into the system. Swartout claims greater depth of knowledge (because the “connective tissue” is kept), improved fidelity, sufficiency and efficiency (executable code is produced which doesn't have to be generated while running). The main drawback of EES is the increased overhead in building the inputs to the system.
4. The fourth method used is reconstructive explanation. This technique uses one KB for reasoning and another for explanation. The philosophy behind this is that to have understandable explanations you need to use the users terms. In the research by Wick and Thompson (1992) a trace is kept by the problem solving KB which is passed through a filter to the explanation KB. Such a system offers high understandability but also high overheads and poses a fidelity risk.

Feedback is a key role of explanation. If feedback is provided in the explanation strategy the problem can be negotiated between the system and the user. Using feedback it is possible to start with imprecise domain and user models because the user can follow the reasoning and ask further questions. It may also be useful to keep track of goals and what was understood by the user. Several communication strategies may be necessary in case one way is not understandable. This also necessitates recovery strategies.

Cawsey (1993) describes two main approaches to explanations. The first tailors the text to the user using a model of the user's knowledge and gives explanations as deemed necessary. The second approach lets the user ask follow-up questions, this feature is often in the form of asking ”why” when prompted for some input or “how” when a conclusion is given. A good model of the system and the user is needed for the first approach and is much more difficult and costly to implement. The locus of control is different with both approaches. The system takes responsibility for providing explanations in the first method. The downside of letting the user control when and what explanation is given is that the user may not know when and what to ask. This may be particularly a problem where the purpose of the explanation is teaching and the user does not know what to ask next. The current HCI view is to always give the user control (Kaplan and Rock 1995). This study has adopted this view and even though the user may not always know what they need to learn about or look at next they are probably in no worse position to make this judgement than the programmer who can not possibly predict the situation and the needs of each user.

Explanation is much more complex than choosing between the two approaches described above. Explanation can vary in the level of detail, technical terms and expressions, the way things are explained or presented and what material is included. Some have sought to simplify the possible combinations by creating one level of explanation for novices and another for experts. However, this is not as simple as it sounds. Cawsey (1993) has found that “there is no simple relation between... [the] level of user expertise and level of detail presented” (Cawsey 1993, p4). Cawsey has developed a system called EDGE which uses content planning rules to determine what to add into the explanation. The domain knowledge is kept separate from the user model. A noteworthy point made by Cawsey is that “tailoring based on a bad user model is probably worse than no attempts at user tailoring at all” (Cawsey 1993, p.7). She sees the ability to update the user model easily and naturally as part of the explanatory dialogue as essential. The system must be able to ask the right questions at the right time so that it can correctly assess the appropriate explanation level.

Swartout and Moore (1993) comment that the development of user models for customised explanations is very time consuming. They suggest that the use of stereotypical models may be a better approach. Using stereotypes the level of detail is customised to each stereotype. The levels of experience mentioned above are common stereotypes used. Rules have different levels associated with them and are fired if the user belongs to that level. Such a system is easy to implement but the explanation will be unsatisfactory if the user does not fit the stereotype.

Another form of customisation is tailoring to the users goals. A good explanation should show how the user’s goals were used and point out alternatives. Such a system has been implemented by Cohen et al (1989).

Scott and Weckert (1995) point out that the knowledge needed for decision making may not be the most appropriate method for explanation. They consider two main issues: what type of knowledge is needed for explanation and how should it be represented.

Using a subdomain of viticulture, the diagnosis of phosphorus in grapevines, they have explored the combined use of schema, rules and models to provide explanations that are generated for each consultation. The system, Vitiex, includes knowledge of the domain, knowledge of explanation and knowledge of the user. As described under causal modelling, Scott and Weckert provide causal explanations.

#### 7.3.3.1 A Summary of the Explanation Activity

Three ideas were repeated by the explanation researchers above, better explanation requires *more*, *deeper* and *abstracted* knowledge. These ideas were identified in Chapter One as important to the reuse of knowledge in general and have been discussed in Chapter Three in relation to RDR. The development of conceptual hierarchies based on RDR rules using FCA is the approach taken in this thesis to provide more knowledge which is structured into various levels of abstraction that is not so obvious from browsing the rules in their primitive form.

Despite its limitations, the primitive form of explanation using rule traces seen in first generation ES was seen as a major benefit of ES over other technologies. In RDR KBS rule traces also offer a history of changes and cases provide contextual explanations. The need for explanation comes back to the need for user control. The ability to explain the reasoning process allows experts to exercise their judgement whether to accept a conclusion or override it. Just as experts tend to justify their decisions based on the situation rather than provide a detailed explanation of the decision processes involved, an expert system must be able to point out why it has asked a question or recommended a course of action based on the context.

From the discussion above of explanation it can be seen that there are numerous approaches to offering explanation. To properly address explanation for KBS, whether using RDR or another knowledge representation, would require this study to focus completely on explanation. This thesis is interested in reusing knowledge for multiple activities with explanation being only one of those usages, albeit a significant one. Thus the approach to explanation offered is a general package of ways that the knowledge in an RDR KBS can be understood. As much of the toolkit is seen as beneficial for teaching we first consider the activity of tutoring and then the RDR approach to explanation and tutoring is described.

### 7.3.4 Tutoring

Kaplan and Rock (1995) claim that Intelligent Tutoring Systems (ITS) are the lost cousin of AI. Typically in an ITS there is a model of the student and pedagogue which allow the system to behave differently depending on the situation. This ability to adapt adds to the amount of knowledge to be kept and the complexity of how it is utilised. System adaptation must also be achieved while bearing in mind that user control is essential in ITS (Kaplan and Rock 1995).

Buchanan and Shortliffe (1984) describe ITS research as the attempt to answer:

- the nature of expertise (what do we want to teach ?)
- modelling (how to determine what the student knows ?)
- tutoring (how to improve the student's performance ?)

They clarify that teaching is different to explaining. In teaching, the student needs to solve problems independently. Therefore mnemonics are good in justifying rules because they assist memory. The ability to explain a conclusion is even more essential in a tutoring system because it is assumed that the user is a novice and will be unable to fill in the gaps that may be possible for an experienced user. Systems that only offer How or Why (in the form of a rule trace) are inadequate. Buchanan and Shortliffe (1984, p.540) mention four different types of justification which include: identification, causal, world fact and domain fact rules. Identification concerns determining the class to which an object belongs based on its properties. The nature of the knowledge in a causal rule will affect the type of explanation it can offer. For example, causal rules may provide one or more of the following: a causal link which is known to exist but is not understood (an empirical association); the direction of the causal link is known but the process is not well understood (complication); or a causal link whose process is well understood and can be modelled (mechanism). World fact rules include commonsense knowledge about the world. Domain fact rules concern domain specific knowledge and provide links between various hypotheses. The explanation given in MYCIN depended on the kind of justification upon which it was based. The need to determine the type of justification to be given is consistent with the observations of Jansen and Compton (1990) that the context affects the justification which is offered as an explanation.

While not the same, there is still a strong correspondence between the needs of explanation and tutoring systems. Both need structural knowledge which acts as the glue that holds it all together. GUIDON, which was built by Clancey as part of the MYCIN family, is perhaps the best known tutoring system. In GUIDON (Clancey 1984) the knowledge needed for teaching was kept separate from the domain knowledge. Knowledge was distinguished by its roles. These roles include:

- The heuristic rule, which links the data and the diagnosis or therapy,
- Structure, how rules are indexed and controlled,
- Strategy, the when, why and how of rule application
- Support or justification for the rule and includes world knowledge/commonsense.

GUIDON has a modelling component representing the user. It tries to work out such things as how much the student knows and if they picked an answer for the right reason. Confidence factors are used to determine the system's confidence that the student really understands the domain. Various types of knowledge are kept in the system, including knowledge of dialogue patterns for explanations and discussion, domain knowledge and knowledge of the communication. The communication model has an overlay student model, case lesson plan and focus record.

Clancey (1984) explains that in tutoring there is not necessarily one correct path or what appears to be alternatively arbitrary choices (according to the rule base) are not necessarily arbitrary in real life. The order of rule conditions may affect the way the system behaves. It is therefore necessary to provide look ahead and planning to guide the user but also to provide flexibility.

White and Frederiksen (1989) have found causal models to be useful as intelligent tutoring systems in the fields of science and engineering. They use a progression of models based on qualitative causal models. Each model builds on the previous one, starting with the simplest and moving to the most complex. The models are introduced in the form of problems to be solved which are used to build models that can be used for causal explanation and to show problem-solving strategies. Rather than trying to manage the complexity of all possible combinations of how the student could use the system, the causal model derived by the system is compared to that derived by the student. In this manner the student can pin-point where their model is deficient. The system generates problems and explanations that help the student discover the differences between the two models and how to amend their own.

A major benefit that White and Frederiksen (1989) claim in using causal modelling for tutoring is that it provides the freedom and flexibility necessary in ITS without the complexity that many other approaches need. The user can solve problems, watch demonstrations and ask for explanations progressing from novice to expert. The strategies offered include: exploration and discovery learning, learning via induction and feedback, learning from examples and explanations or a combination of mixed strategies. A map is kept for each student so that they can see the progress they have made and the direction in which they are heading. The freedom of this ITS supports multiple learning styles and encourages initiative and motivation which they will need to eventually qualify as experts. Their application was evaluated with seven students.

The socratic method of teaching is where a student is lead through various questions to formulate general principles from individual cases. It is a means of proposing and testing hypotheses. Domains that support causal modelling are best at offering such a teaching strategy (Wenger 1987).

Wenger (1987) sees "communicability" as a key differentiating feature of ITS. There is a spectrum ranging from *black box* to *glass box*. Wenger provides the following history of ITS in which the similarity of ITS to causal modelling is evident. One of the early systems was SCHOLAR which used semantic nets to support an *overlay model*. The distance between nodes in the semantic net was an indicator of relevance. Numerical tags were attached to nodes to guide the selection of topics. The use of the semantic net and numerical tags is similar to the use of a symbolic hierarchy and weights in the CBR approach used by Kriegsman (1993). Wenger argues that the semantic net approach could not represent procedural knowledge adequately and as found by Brachman and Levesque "inheritance of properties in conceptual hierarchies is a complex problem in general" (Wenger 1987, p.33).

The use of scripts followed the use of semantic nets. Scripts use stereotypical sequences of events and are better at representing processes. The scripts are organised in a hierarchy allowing the user to progress from general understanding of the concepts through to more specific details. Although seen as an improvement, the scripts in WHY were insufficient to explain the mechanisms or correct student misconceptions (Wenger 1987, p. 42). Scripts provided a view that was still too local. The need for multiple viewpoints was seen and developed into the search for multiple mental models. The componential view of mental models is similar to Lee's (1996) use of compartmental models in her work on causal modelling. Wenger (1987, p. 53) found that causal models need to distinguish between time bound processes and functional relationships. There appears to be a move in ITS towards simulation models such as those supported by Augumented Transition Networks (ATN). The HTW work (Vescovi et al 1994) described under causal modelling includes the use of simulation models.

The following discussion reviews how RDR, including previous implementations and the MCRDR/FCA tool developed in this study, support the activities of explanation and tutoring. First we will revisit how previous RDR implementations support explanation and then look at how the addition of FCA and other work reported in this thesis has added to the explanatory power of RDR. There is no intention to build user models or customised explanations as this would conflict with the goal of allowing the user to choose what activity they wish to perform and in what way.

### 7.3.4.1 The RDR approach to Explanation and Tutoring

RDR offers a paradigm shift from existing explanation research. In section 7.3.3 there was major criticism of the first generation ES style of explanation that simply involved rule traces. One of the problems with conventional rule traces was that they did not give a global, or deeper, view of the knowledge.

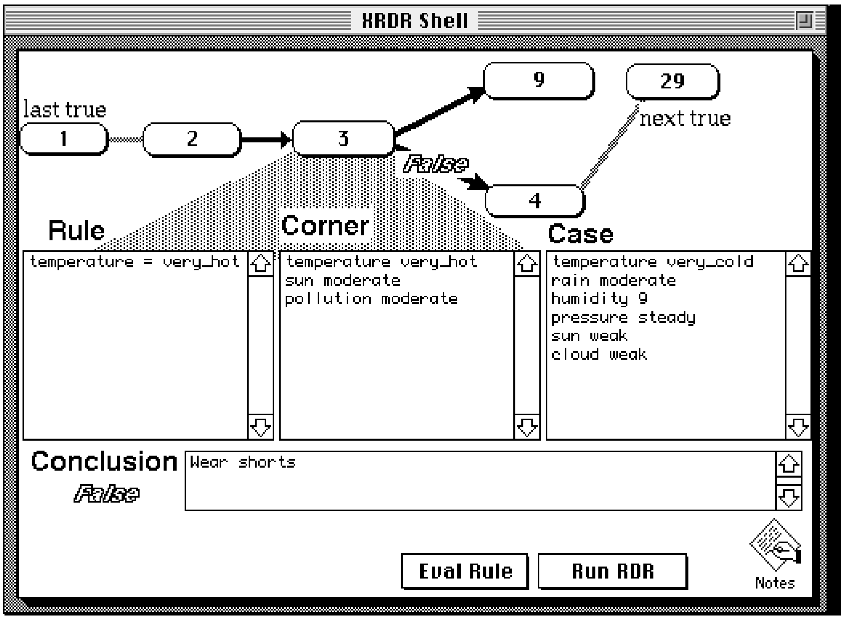


*Figure 7.13: A rule trace in single classification RDR*

Explanations that rely on the use of sub-goaling to reach conclusions often do not provide explanations that match the way in which the expert and/or user reach the same conclusion (Clancey 1984). Clancey found that for explanation or learning purposes ES "need to articulate how rules fit together, how they are constructed" (Clancey 1984, p.59). Similarly, Swartout (1993) argues that in most systems the *deeper knowledge* needed for explanation is *compiled out* and lost at development time. This loss of information is not so severe in an RDR KBS since a history of the evolution of the rule base is stored in the tree structure and associated cornerstone cases. It has been claimed that this history can help to reduce ES brittleness (Kang 1995). Clancey (1984) makes the point that students should not just be able to confirm a diagnosis but should be able to learn under what circumstances that conclusion should be considered and what other possible diagnoses explain the data. This is supported through the RDR exception structure by offering not only an explanation of why a particular conclusion was given but also why other conclusions had not been given and what values could be altered to reach another conclusion. This can be seen in the rule browser in figure 7.13 which is part of the XRDR system on the Macintosh. Alternatives are harder to determine in conventional ES due to complex interaction of rules and the numerous possible pathways to arrive at a conclusion. The rule conditions identify the salient features in a case that prompted the recommendation. The rule conditions on the true and false branches in the rule trace provide a deeper view of the knowledge because they provide the history and context in which a rule exists. The cases associated with each rule support another contextual dimension of explanation.

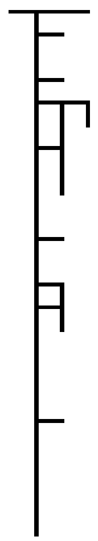

Figure 7.14: Typical RDR Tree

In figure 7.14 we can see the structure of a typical single classification RDR KBS. The false branch is shown moving down and the true branch moves to the right. We can see that most rules are attached to the false branch. Each time we run a case a new path is traversed and the trace is kept. Figure 7.1 showed the rule trace generated when case 53 is evaluated. For the purposes of explanation, it seems pointless to keep the false branches before the first branching from the main stem occurs, since there may be many different combinations of attribute-value (A-V) pairs. It is debatable whether the first true rule is significant and representative of some more general features that distinguish this case from others and that subsequent rules are refinements of this knowledge. The important information that a rule trace provides is not found by consideration of true or false branches only but is provided in the differences between the cases along the way (Compton, personal communication). In keeping with this, it did seem useful to provide some of the false branch information as part of the explanation given to the user but only after branching from the main trunk. To test this idea a modification was made to the Test Screen as shown in Figure 7.15. The false rules prior to the first true branch have been stripped away. The conclusion and the conditions of *all* rules after the first true rule were displayed to the user showing the true conditions in one box and the false conditions in another. The conditions of true and false rules were shown in separate boxes for clarity. Previously, only the conclusion from the final true rule and the conditions of true rules were given to the user. It can be seen in Figure 7:15 that a key reason why conclusion 36 ( the conclusion for rule 134 which was the last true rule) has been given is because the sex of the patient is unknown. This feature in the case has resulted in rule 19 (with the condition SEX=FE) failing. We can see that if the sex of the patient was female conclusion 61 would have been recommended. We could test out this by changing the value in the case and seeing if any other exceptions would override conclusion 61. According to Clancey's criteria for ITS previously mentioned, the expanded explanation provided the means to determine the final diagnosis as well as which other diagnoses should be considered and what alternative conclusions are likely. It was also possible to alter the value of attributes in the case and see how the path and conclusions changed. It is possible to use this trace to answer all of the questions a user may want to ask as outlined in Section 1, except for critiquing (question 8).

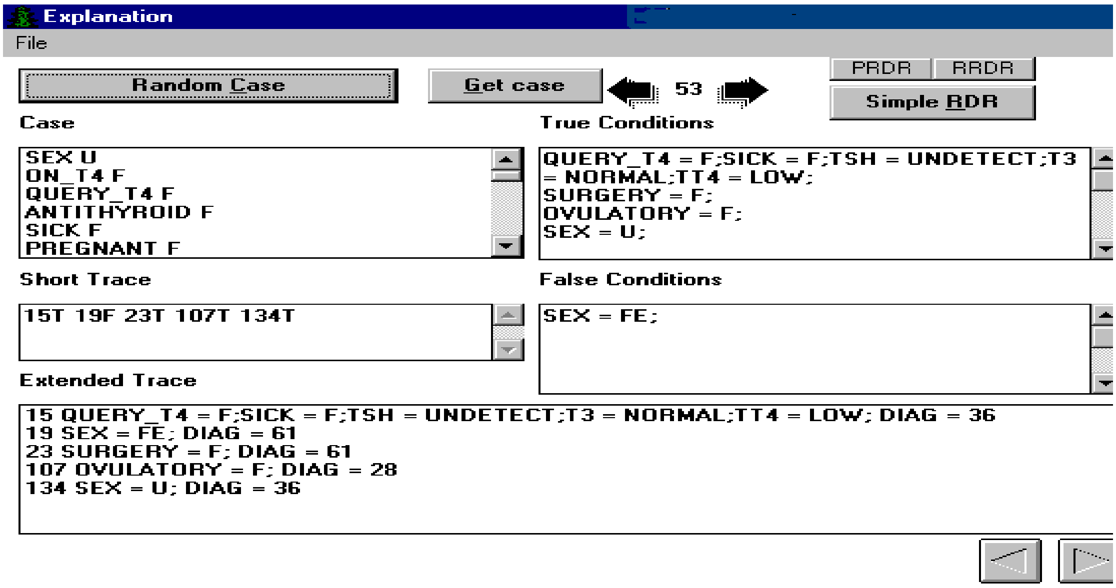


*Figure 7.15: An extended explanation screen.*
*This screen shows the conditions on the true and false branches after the first branching from the main stem. False rules prior to branching are discarded from the trace.*

The impetus for providing better explanation stems from the attitude that if the user understands why the ES came to a certain conclusion or asked a particular question then the user is empowered to use the knowledge in a better way. An alternative and possibly contentious suggestion (Edwards 1996) is that explanation is not really so important in deciding whether to accept advice. Rather, the lack of trust stems from a lack of system credibility. If statistics were kept regarding the number cases seen and correctly classified, it would be possible to determine the accuracy of the system and let the user evaluate the system's credentials before deciding to accept the current conclusion. As mentioned in Section 3.1.3, Edwards (1996) has implemented such a strategy for RDR KBS which collects various statistics and determines when a system is acting prudently.

The focus of much explanation research is on providing a detailed line of reasoning to the user. It is argued here that a different approach to explanation is worth considering since we do not really understand what expertise is and how an expert arrives at a conclusion. It may be that if we provide the user with enough information on the circumstances under which the rule was meant to apply and the relationships of that rule to other rules, which is offered to a large extent by the RDR structure, the user can develop a line of reasoning that works for them.

This is the approach adopted in this thesis. The RDR toolkit which supports the user in performing explanation and tutoring includes: the rule trace which can be visually represented as a tree (Figure 7.14), rule and tree browsers (Figure 7.13), traces and extended traces (Figures 7.1 and 7.15), the test pathways screen for comparing two rules or one rule against all other rules (Figure 7.11), the knowledge structured into an abstraction hierarchy as shown in the concept matrix (Figure 5.3) or lattice (Figure 5.6), the ability to select numerous views of the KBS on which to focus (Figure 5.11) and the ability to compare the models in different KBS (Figure 6.9). The user is free to use and reuse knowledge in any of these ways according to their current needs and preferences.

Although ITS has additional requirements, much of the above discussion on explanation applies to ITS. The use of RDR for tutoring has not been explored in great depth. From the ITS literature, to adequately support ITS it would be necessary to incorporate teaching knowledge together with a model of the student, to guide explanation. The closest usage of knowledge in an RDR KBS for teaching is the work done by Lee and Compton (1995) on causal modeling. The approach taken in this thesis to ITS is the same as the approach to explanation and the same toolkit is offered for learning purposes. Taking a situated view of knowledge and the variability in student's needs, the approach in this thesis it to let the user decide what tool is most appropriate to their current educational requirements. To test this idea some evaluation has been performed using the concept lattice as a teaching tool. The method and results are described in Section 8.2.

Explanation in RDR can also be evaluated using Swartout and Moore's (1993) criteria for a good explanation. Let us consider each of the aspects they mention in relation to RDR explanation.

1. **Fidelity** asks how accurate is the representation ?
   The explanations offered in MCRDR/FCA cover a range of representations based directly on the rules and generated automatically by the system. There is no discrepancy between what is shown in the explanation and what is held in the KBS which is more likely to occur when a separate system is used for explanation or when an automatic programmer is used that selectively chooses what information should go into the performance system and what should go into the explanation system. The use of GTs with specialised explanations are also prone to errors in fidelity because as pointed out before if the problem solving method is changed the explanation system must be changed accordingly.

   Swartout and Moore also commented that fidelity is aided by a simple inference engine. The inference engine in RDR is simple. To obtain a conclusion using single classification RDR the system simply tests the case against the rules and follows the true or false branch depending on the result. The last true rule on the pathway provides the conclusion. Inferencing in MCRDR is also simple and requires the children of all true nodes to be evaluated. The conclusion for rules at the end of pathways are the conclusions given for the case.

2. **Understandability** asks whether the content and context are understandable? Cases provide the context in RDR and provide an easily understood approach to context. Testing the comprehensibility of the content of the RDR explanations is not so easy. An evaluation of the line diagram format is given in Section 8.2.1 where 12 computer engineering students and academics took part in an evaluation of the line diagrams compared to other representations of the same data. In that survey it was found that 10 out of the 12 participants were able to learn to read a line diagram within a few minutes. However, trying to use the line diagrams to learn from was not found to be such an easy task. Within the half hour experiment some people were able to draw some conclusions regarding knowledge in the domain but others found the representation difficult to follow. For more details refer to that chapter.

   The components offered by Swartout and Moore can be used to focus on more specific aspects of understandability. They are:

   *terminology:* an RDR system is developed by the expert and the terminology used for rule conditions is based on the cases and the terminology used for conclusions is determined by the expert. Therefore the terminology should be acceptable to the expert user.

   *user sensitivity* - MCRDR/FCA does not use any type of user modelling and thus is not sensitive to individual users. However, it is envisaged that any application system based on MCRDR/FCA would be customised to the user group so that the range of available *activity-uses* would support the range of different users within that group.

   *abstraction into different levels* - the FCA line diagrams uncover and order the concepts in the MCRDR rules into an abstraction hierarchy.

   *summarisation* (the amount of detail offered in an explanation) - Rule traces allow the user to focus on the part of the KBS of interest to them. The line diagrams may be generated from a number of different views that provide a summary of just that aspect of the KBS and the amount of labelling in a line diagram may be adjusted to show a combination of attributes, rule conclusions, concept names or full labelling which includes attributes and objects that can be reached by ascending or descending pathways, respectively.

   *the possible system perspectives* - as noted in the previous point, and described in Section 5.2.1, there are a number of views of the knowledge from which the user can select to derive a line diagram.

   *linguistic competence* - as noted above the terminology used is essentially what is appropriate for the domain and what is used by the expert. However this terminology may not always be the easiest to follow particularly for a new user. A major problem with explanations based on rules is that when an automatic explanation is generated they are limited to the terminology used in the rules and transposition into natural language often does not result in sentences which, from a human perspective, appear all that natural.

*Feedback* - was discussed in Section 3.3.1.4 and is seen as an integral part of any system that relies on the user for system development and maintenance. Feedback in MCRDR/FCA is immediate in that changes made to the value of an attribute in a case or the addition of a new rule is immediately reflected in the KBS and in any subsequent line diagrams using that knowledge.

3. **Sufficiency** asks whether there is enough knowledge to provide explanations in different contexts. Sufficient knowledge to support an adequate explanation may not be present in an MCRDR KBS. Lee and Compton (1995) found that for causal explanations more knowledge needed to be acquired from the expert where the link was not provided in the heuristic rules. Clancey (1984) found that rules which gave guidance on the teaching strategy were necessary in GUIDON and others have found that the problem solving method should be part of the explanation. All rules in an RDR KBS are operational domain knowledge and do not include rules with different roles. This study questions whether PSM knowledge is really necessary. Since we don't really know how the expert solves the problem, the usefulness of explaining how the computer solves the problem is debatable.

   Swartout and Moore (1993) describe the types of knowledge that may be needed to support sufficient explanation and include knowledge about the system's behaviour, justification, preferences, domain explanations and terminology definitions. The explanations provided by MCRDR/FCA are domain explanations with the associated cornerstone cases providing the justifications. Since there are no control rules or user models, system or user preferences are not applicable. The system's behaviour is standard for all MCRDR systems and is not related to the domain or the task. The lack of terminological definitions, such as comprehensive descriptions of conclusions and the attributes and values found in rule conditions, is an area that could be improved. Any deployed system should offer assistance in understanding both the terminology related to the MCRDR interface and the domain in such forms as help screens and user manuals.

4. **Low construction overhead** - how time consuming and difficult is it to build the explanations ? The explanations provided by MCRDR/FCA have very low construction overheads as they are all automatically generated from the rules.

5. **Efficiency** - what will the system response time be like, will it take a long time to generate an explanation ? Although the construction overhead is low the efficiency of building those explanations, in particular line diagram explanations, is computationally expensive as the calculation of concepts is NP-Hard and can become a memory and response time problem as the KBS size grows. This is a common problem in AI and is related to the search strategy used by FCA in deriving concepts. The solution to the problem is to restrict what is included in a formal context as described in Section 5.2.1.

A summary of how MCRDR/FCA measures up to Swartout and Moore's criteria for good explanations is given in Table 7.1. The four main explanation architectures they described are included for comparison. A question mark indicates that insufficient information on the architecture did not allow an evaluation to be made. The ratings used are poor, good or excellent and are guesstimates based on discussions in the literature.

| Approach / Criteria | Neomycin | Generic Tasks | EES | Reconstructive | MCRDR/FCA |
|---|---|---|---|---|---|
| Fidelity | Good | Poor | Good | Poor | Excellent |
| Understandability | Good | Excellent | Excellent | Excellent | Good |
| Sufficiency | Good | Good | Good | Good | Good |
| Low Const. O/H | Good | Poor | Poor | Poor | Excellent |
| Efficiency | Good | ? | Good | ? | Good |

*Table 7.1: A Comparison Table of how MCRDR/FCA compares to other well-known KBS explanation tools.*

It can be seen from Table 7.1 that MCRDR/FCA compares favourably with other explanation approaches and is particularly strong when fidelity and construction overheads are considered. The approach to explanation offered by MCRDR/FCA is novel when compared to these other approaches. The other approaches place most of the responsibility for explanation on the system with the user playing a less active role in deciding the type of explanation they should be given. In contrast, MCRDR/FCA offers a toolkit that the user employs to provide the type of explanation they are seeking.

### 7.3.5 What-if analysis[3]

The ability to test out different scenarios is an important human activity as evidenced by the popularity of spreadsheet software which contributed to the widespread acceptance

[3] Much of this section has been taken from Richards, D.C. (1994) "The Adaptation of Traditional Expert Systems to Provide 'What-if' Analysis in the Decision Support System Style" Master's Thesis, Charles Sturt University, Riverina.

of the personal computer and Decision Support Systems (DSS) in the 1980s. Due to the high end-user support for such systems, this section considers not only ‘what-if’ analysis but also the various features of a typical DSS that may be desirable to include in a user-centred system.

Philippakis (1988, p.366) sees choice and exploration as the two main purposes of ‘what-if’ analysis. Philippakis also explains that traditional ‘what-if’ functions found in many Decision Support System (DSS) are unsatisfactory due to the lack of optimal decision processes for large volumes of data and 'avoidance of disconfirming evidence'. In ‘what-if’ analysis subsequent steps depend on the outcome of earlier steps (Leigh and Doherty 1986), not a feature easily incorporated in traditional ES.

Schultheis and Sumner (1992) offer the following description of ‘what-if’ analysis in the context of DSS:

- User has the ability to enter a number of different values and test the result of each situation (p.138, p.360);
- Important decision tool, that assists but doesn’t make the actual decision (p. 360);
- Allows the development of models (p.139);
- Project management software allows questions to be asked regarding changes in project event sequences and resource allocation, providing optimisation of resources (p. 144);
- ‘What-if’ modelling capability is found in some financial analysis software (p. 377);
- Built in financial, statistical and mathematical functions are often found in software providing 'what-if' processing (p.377);
- Usually associated with numerical or financial data;
- Some programs let the user specify a goal. The system then works backwards, providing the values that satisfy the goal (p. 377). Also known as goal-seeking (Turban 1993).

Flaaten et al (1989) include the following additional features. DSS which offer ‘what-if’ analysis often have graphic capabilities, usually support strategic and tactical decisions and are mostly PC-based systems. The data used in DSS are generally copies taken from the transaction processing system for individual use. The latter leads to the problems of keeping data current and avoiding data redundancy. Flaaten et al (1989, p. 132) point out that a major concern with DSS is that the conclusions they give may not be current. Many of the users of DSS are not aware of the proper use of statistical formulas, mathematical functions and even graphs.

The presentation and appearance of the output of DSS is of great importance to users, generally management. Even this can be misused to make bad or wrong data look presentable and be accepted as correct. Flaaten et al (p. 266) claim that the processing involved in the analysis of this data can take hours, a point disputed by Leigh and Doherty (1986) and Hopple (1988).

Turban (1993, p.65) refers to ‘what-if’ analysis as a type of sensitivity analysis and concisely describes it as a means of testing what will happen to outputs if there is a change in inputs. Turban discusses goal programming in the ‘what-if’ context. Goal programming handles optimisation problems, allowing the user to allocate resources where there are multiple and conflicting goals. ‘What-if’ is facilitated in three ways: by letting the user alter the priority of goals, modify the goal target values or step back through the consultation changing values.

Turban (1993, pp.851-872) discusses some of the techniques used in the financial analysis software, Interactive Financial Planning System (IFPS). Features of interest to this study are the ability to temporarily change data, reset data back to its original status (base case) and the use of command files to store different scenarios. Turban & Watkins (1985, p.150) explain that DSS may answer ‘what-if’ but in an ES the user can also ask 'why'.

While not specific to ‘what-if’ analysis the following list provided by Klein and Methlie (1990, p.150) is useful in describing the desirable features of a DSS style system.

1. End-user usage and interactivity;
2. End-user definition;
3. Easy access to pertinent information;
4. High interaction between users, the system and the learning situation;
5. Capacity to adapt to fast evolution of user needs;
6. Portability and peripheral support;
7. Reliability;
8. Performance.

Bennett (1983, p.20) provides a different, but equally useful list of DSS features, summarised below:

1. Multiple processes and decision types exist.
2. Inclusion of tools to help conceptualise problems, support Intelligence, Design and Choice activities and memory aids.
3. Customised decision making and user control is provided.

In a discussion on modelling, Young (1989, p.51) stresses the importance of user control. The user needs to be able to initiate the process, vary inputs, modify the model and select outputs and formats for display. A non-procedural language should be used so that the developer can control the model in a natural way without concern for syntax (p.52). He defines three types of variables to be considered: outcome variables, controllable impact variables and uncontrollable impact variables (p.84). He finds two classifications of models useful: the treatment of time and the treatment of uncertainty (p.81).

The features of 'what-if' analysis offered by the above authors becomes the criteria against which the altered system can be evaluated. These points are outlined in Table 7.2 below.

1. Users can test different scenarios through the manipulation of variable values
2. Assists decision making, particularly tactical and strategic decisions
3. Allows models to be set up and tested
4. Projections and forecasting ability is offered
5. Allow optimisation to be performed
6. Built in functions are usually available to assist model building
7. Ease-of-use and presentation are important, often includes graphics
8. Usually PC-based systems
9. In some packages users can specify a goal and be given the values that satisfy that goal
10. Usually associated with numerical data
11. Data is often derived from external sources, eg a transaction processing system
12. Problems of data repetition and lack of expertise in using the functions provided
13. The end-user usually defines, creates and uses the system
14. Need to be flexible and adaptable to the user's needs.

*Table 7.2: Features of a DSS providing 'what-if' processing.*

Much of the work by Richards and McDonald (1995) on adding decision support system (DSS) features to an existing ES was based on traditional ES that are not easily modified and offer the user very limited control. RDR ES are primarily built and controlled by the user and offer many of the features of 'what-if' and DSS software.

#### 7.3.5.1 The RDR approach to What-if Analysis

The ability to let the user test out different scenarios by changing attribute values is a feature of existing RDR implementations. In current versions of XRDR and MCRDR for Windows it is possible to add or delete attributes or alter one or more values in a case and then look at the result. With the addition of the extended rule trace, mentioned above under explanation, it is possible to determine how the change in one or more attribute values affected the conclusion given. The ability to enter a conclusion and determine what value/s would satisfy that conclusion, which is a version of backward chaining known as goal programming, has not been explored. As with the study by Richards and McDonald (1995) it was found that the problem was one of combinatorial explosion on the number of possibilities that a user could enter or alternatively leave for the system to determine.

In many consultation-style systems that ask the user for input it is of course possible for the user to alter the values they provide to the system and then to see the effect of the change on the recommendation. However, a key aspect of 'what-if' analysis is the ability to choose whether or not the values should be updated and the ability to restore values to their original value. In many systems some of the values are received from external sources, such as was the case with AusVit, and it is not desirable to overwrite these historical values. RDR does allow manipulation of all attribute values in cases without affecting the actual case.

To conclude, the features of a 'what-if' system as outlined in Figure 7.2 are used as evaluation criteria for the MCRDR/FCA system. Some of the comments relate specifically to the MCRDR/FCA system and others apply generally to RDR systems.

1. In MCRDR/FCA users are able to change the value in a case to test different scenarios. An individual case represents a scenario that can be replayed and a set of cases is similar to the use of command files in IFPS which are used to store different scenarios. This tool has great value for 'what-if' activities as it is possible to see the impact of changes and to determine what changes are worth investigating.
2. The purpose of RDR systems, as with most KBS, is to assist decision making. However, DSS tend to be particularly employed to assist in tactical and strategic decisions. To date RDR systems have been more concerned with operational decisions perhaps with the exception of using RDR for acquiring search knowledge (Beydoun and Hoffman 1997) or flight control knowledge (Shiraz and Sammut 1998).
3. RDR systems allow performance knowledge to be captured into a simple model but it is difficult to determine what higher level models may exist. MCRDR/FCA overcomes this limitation. By varying the selection criteria, higher level models can be set up and explored.
4. The ability to use the knowledge in an MCRDR/FCA system for projections and forecasting has not been explored. This would require at least some model of time to be built into the system that would alter the behaviour of the system over time. However, the ability of many RDR implementations to handle time course data could be exploited together with the work on causal modelling to provide forecasts.
5. No particular optimisation strategies are offered with MCRDR/FCA. At this point the user would be responsible for trying out a number of variable values to determine which values produce the optimal result.
6. RDR systems are designed to be easy-to-use so that they may be managed by the end-user. Presentation of the knowledge is seen to be important and the ability to offer a number of representations of the same data is seen as beneficial to the user.

However, the focus of research into RDR has not been on the presentation of the user interface and there is no claim that the screens in this thesis are well-designed.

7. The RDR systems have been PC-based, this includes Macintosh microcomputers.
8. The ability to enter a goal and see the values that satisfy that goal has not been particularly explored in MCRDR/FCA. However, there are two *activity-uses* that could be used for this purpose. The first one is critiquing which does allow the user to input their own conclusions. When the user clicks the critique button they are shown all other pathways that lead to that conclusion and the user could derive the values from inspection of the rules. Alternatively, the user can select a particular conclusion from which to generate a formal context and then be shown all rules that have that conclusion in the form of a line diagram.
9. The data used by RDR KBS is typically qualitative although the data in the cases may be numerical which have been preprocessed to fit into a qualitative category. So while qualitative data are more common, numerical data may also be used.
10. The data in initial RDR systems, such as PEIRS, came from external sources.
11. RDR does suffer somewhat from repetition of the knowledge but this is due to the exception structure and not due to the prolific development of PC-based systems using copies of the mainframe data that is often associated with DSS.
12. The lack of expertise found in the users of many DSS is due to lack of understanding of the appropriate use of various statistical functions and graphs. The functions offered in RDR systems are typically user-defined so that their appropriate use should be understood. When it comes to development of the knowledge base it is assumed that the developer is an expert in that domain so there should not be a lack of expertise regarding domain knowledge either.
13. RDR systems are designed to be defined, created and used by the expert end-user.
14. The simple knowledge structure and inferencing method used by RDR has resulted in systems that appear to be more flexible that many other KBS approaches that are highly tailored to a particular task or domain. The MCRDR/FCA system has been particularly built based on the belief that systems need to be flexible and adaptable to the user's needs.

These observations are summarised in Figure 7.3. Points 4, 5, 6 and 12 are not included in Table 7.3 as features 4, 5 and 6 have not been considered necessary for the applications currently considered by MCRDR/FCA. Point 12 relates to problems associated with DSS and are therefore not desirable features to be included in an evaluation.

| Feature | Supports |
|---|---|
| 1. Manipulate variable values | YES |
| 2. Assists decision making | YES |
| 3. Build and test models | YES |
| 7. Ease-of-use and presentation | YES but can be improved |
| 8. Usually PC-based systems | YES |
| 9. Users can specify the goal | YES to some extent |
| 10. Uses numeric data | YES but tends to use symbolic |
| 11. Data derived from external sources | YES for initial systems |
| 13. The end-user defines, creates and uses the system | YES |
| 14. Flexible and user-adaptable | YES |

*Table 7.3: An Evaluation of MCRDR/FCA against the DSS features in Table 7.2*

The addition of FCA to MCRDR supports 'what-if' analysis not just for inferencing but also for KA and maintenance. By using the Test Pathways screen the user is able to consider how the addition of a new concept will impact on the existing KBS. While not currently implemented it would be a small enhancement to temporarily save the proposed pathway and to draw line diagrams that included the new concept/s and which were based on a number of different views according to the selection criteria used in generating a formal context.

It is also envisaged that improvements to the line diagram interface which allowed the user to drop or expand selected nodes would also facilitate 'what-if' reasoning about the knowledge. The ability to perform 'what-if' analysis at both a conceptual and behavioural level using MCRDR/FCA is a significant extension to what is typically supported by 'what-if' systems. We now go on to look at a refinement of the 'what-if' analysis activity which is referred to as hypothesis testing.

### 7.3.6 Hypothesis Testing

Another form of 'what-if' analysis is hypothesis testing. Hypothesis testing should be an activity supported by KBS as Clancey (1984) argues that people use hypotheses to reason and learn. Hypothesis testing is a more specialised application of 'what-if' analysis because the user starts with an expected outcome and explores various scenarios in pursuit of confirming or disconfirming their hypothesis. As noted in the previous section, a major problem with 'what-if' systems is that people tend to avoid "disconfirming evidence" (Philippakis, 1988, p.366). The use of ES for hypothesis testing is seen as superior because there is expert knowledge which can be used to show inconsistencies in the reasoning process. For example, the work by Feldman, Compton and Smythe. (1989) found previous assumptions (hypotheses) could not account for all the results they had and prompted further exploration and refinement of the hypotheses.

Similar work on a smaller scale but using the same data was performed by Menzies and Compton (1997). This work did not involve RDR. No work in this thesis has been conducted specifically for the purpose of hypothesis testing but the activities already covered in the critiquing, causal modelling, explanation, tutoring and 'what-if' activities all support hypothesis testing when the user starts with an expected outcome in mind and changes values in cases or looks at different views of the knowledge and refines the hypothesis based on the models presented by the system. In the evaluations described in Chapter Eight, it is shown that the FCA line diagrams can be used to propose a hypothesis which can be tested and refined by using some aspect of the original line diagram to generate a new line diagram.

### 7.3.6 Student Modelling

Buchanan and Shortliffe (1984) classify student modelling as a different usage of knowledge which involves just experimenting with knowledge structures and control strategies. They see that student modelling can assist cognitive research.

Clancey makes the point that representations for generation (for example automatic programming) and recognition are different and will affect different applications of the knowledge:

> *"This distinction between representations required for generation and recognition is crucial for any knowledge acquisition or student modelling system that does not receive a well-formed specification, but must abstract it from a sequence of observed behaviours" (Clancey 1992, p.71).*

The ramification of this is that a person's frame of reference will determine how a strategic theory is perceived. When strategy is part of what we are trying to teach then we are interested not only in giving the correct conclusion but showing the appropriate means of reaching that conclusion. This is a very difficult task as people will use different strategies from one another and particularly from a machine in inferring a solution.

A nonmonotonic inductive student modeling system known as THEMIS has been developed by Kono, Ikeda and Mizoguchi (1994). THEMIS uses DeKleer's ATMS to develop single world and multiple world contradictions from the observed knowledge of the student. These contradictions are seen as key parts of the student's learning process and identification of them is valuable in determining where a students knowledge is inconsistent and in tracking changes in the student's understanding.

As with hypothesis testing, the student modelling *activity-use* has not been explored in any detail in this thesis. However, given the difficulty of trying to build systems that cater for a wide range of learning strategies a group of students may use, the solution offered in this thesis is to provide sufficient information in various forms so that the student may interact with the system in a way that fits their learning style. The ability to critique and test out hypotheses using MCRDR/FCA does provide feedback to the student that their thinking may be incorrect and provides an explanation of the expert's knowledge contained in the KBS. However, since no model of the student is kept it is not possible to track changes in the students thinking or to query the system as to what areas the student's knowledge is most deficient. Perhaps statistics regarding use of the modelling tool could be kept for each user in a manner similar to the storage of statistics on the behaviour of the system as is done in Edwards (1996) *credentials.* There is, however, value in having a student develop an RDR KBS as a means of testing out their knowledge about a domain. In a study performed by the Health Insurance Commission (HIC) of NSW (Wang et al 1996) RDR was used to develop a KBS regarding detection of fraudulent behaviour. As part of the process the HIC had an expert classify the practice profiles of 1,500 medical practitioners. A member of the computer department then used these cases to develop a KBS. Through the use of RDR it was soon realised that there were many inconsistencies in the classifications assigned by the so-called expert. If the RDR KA technique had been followed the expert would have been presented with the cases and been forced to provide consistent conclusions as each rule was being built. RDR would have forced the *expert* to review his knowledge of the domain particularly where he held conflicting opinions. This would have also saved a duplication of effort and would have resulted in a set of well-classified cases and a validated knowledge base.

### *7.4 Chapter Summary*

This chapter considers whether and how the original goal of this work has been achieved. That is, is it possible to reuse the same knowledge for a wide range of activities. The activities described in this chapter include inferencing, KA, maintenance, validation, critiquing, causal modelling, explanation, tutoring, 'what-if' analysis, hypothesis testing and student modelling. The first four activities were already well supported in RDR systems. The approach to KA/maintenance and validation have been strengthened through the inclusion of FCA into MCRDR which allows these activities to be performed reflectively.

The reflective activities such critiquing, explanation, tutoring and 'what-if' analysis had different requirements to the reflexive activities that MCRDR was best at supporting. The addition of FCA modelling tools has provided the links between concepts and the uncovering of higher level abstractions that were previously not obvious in the RDR assertions. The activities of causal modelling, hypothesis testing and student modelling have not been considered and treated in any great depth in this thesis but it has been argued that MCRDR/FCA still offers some support of these activities which could be extended further, if required.

The main benefits that the MCRDR/FCA system offers over the original MCRDR system are enhanced validation techniques, the ability to critique proposed conclusions and rules, the ability to perform 'what-if' reasoning at higher levels and a wider range of explanation and tutoring tools that allow the user to view and explore the knowledge in more ways by making explicit the conceptual relationships, abstractions and their structure that were implicitly contained in the RDR rules. Many of the claimed benefits of MCRDR/FCA hinge on the usefulness of the line diagram. It is therefore important to evaluate such things as how easily the line diagram can be learnt and comprehended. Such an evaluation is the concern of the next chapter.