The Knowledge Engineering Review, Vol. 24:2, 159–184. © 2009, Cambridge University Press
doi:10.1017/S0269888909000241 Printed in the United Kingdom

Two decades of Ripple Down Rules research

DEBBIE RICHARDS

Computing Department, Division of Information and Communication Sciences, Macquarie University, Sydney, NSW 2109, Australia;
e-mail: richards@ics.mq.edu.au

Abstract

Ripple Down Rules (RDR) were developed in answer to the problem of maintaining medium to large rule-based knowledge systems. Traditional approaches to knowledge-based systems gave little thought to maintenance as it was expected that extensive upfront domain analysis involving a highly trained specialist, the knowledge engineer, and the time-poor domain expert would produce a complete model capturing what was in the expert's head. The ever-changing, contextual and embrained nature of knowledge were not a part of the philosophy upon which they were based. RDR was a paradigm shift, which made knowledge acquisition and maintenance one and the same thing by incrementally acquiring knowledge as domain experts directly interacted with naturally occurring cases in their domain. Cases played an integral part of the acquisition process by motivating the capture of new knowledge, framing the context in which new knowledge would apply and ensuring that previously correctly classified cases remained so by requiring that the classification of the new case distinguish it from the system's classification and be justified by features of the new case. RDR has moved beyond its first representation which handled single classification tasks within the domain of pathology to support multiple conclusions across a wide range of domains such as help-desk support, email classification and RoboCup and problem types including configuration, simulation, planning and natural language processing. This paper reviews the history of RDR research over the past two decades with a view to its future.

1 Introduction, background and philosophy

Traditional knowledge-based systems (KBS), in past times better known as expert systems (ES), promised to be the next big thing following the progress made into database and information systems. With expertise always in short supply, the ability to capture what experts 'knew' and have other people or computers reuse that knowledge was clearly appealing. In addition to offering a system that could support or even do decision making, ES offered the ability to ask 'why' the system had given a particular conclusion by reviewing the rule and 'how' by providing a trace of the rule pathway leading to that conclusion (Ignizio, 1991). However, ES failed to take into account two very important features of knowledge that have clearly emerged from ES research and practice: the contextual and evolving nature of knowledge (Agnew *et al.*, 1997; Clancey, 1997). If knowledge was in fact something that could be unequivocally stated and remains true over time and across individuals, then the declaration of rules and facts would result in a body of knowledge that could be acquired and utilized without major concern for maintenance of that knowledge. But as they say, 'rules are made to be broken' or at least in the case of Ripple Down Rules (RDR), rules are made to be corrected.

The philosophy underlying RDR distinguishes it from much other work, particularly the views prevailing at the time of its inception. Rather than view knowledge as an artifact that could be

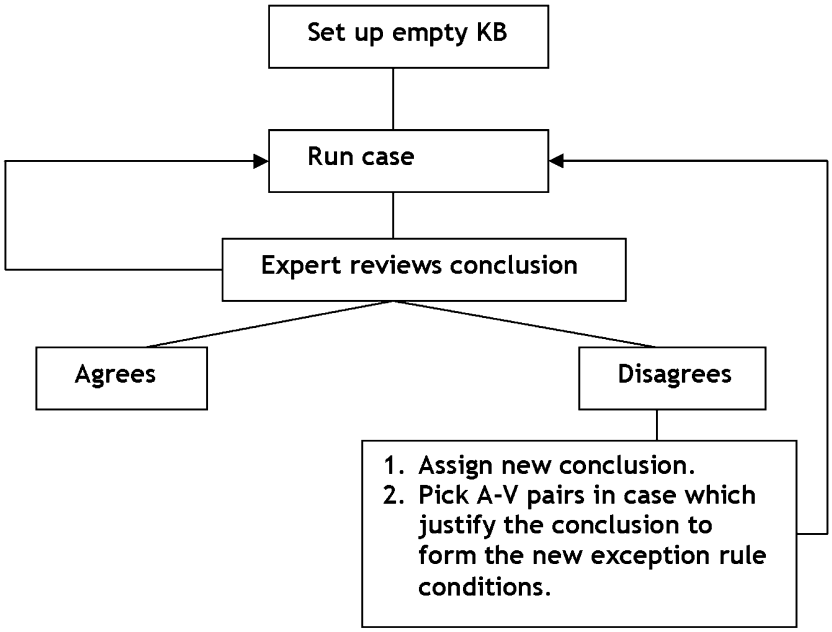

Figure 1 The KA process in RDR

collected, stored and used by others, it argues from a situated cognition perspective that experts never *explain* how they reach a conclusion, rather they *justify* that a conclusion is correct, and provide this justification in a particular context (Compton & Jansen, 1988, 1990). The ramifications of the view that knowledge is made up to fit each context is that all knowledge acquisition (KA) must be incremental and captured within the context in which it holds. The result is a combined rule-based and case-based reasoning approach where the cases provide the context for when the rules are valid and the rules provide the index by which the cases can be retrieved. Thus, the approach seeks to combine the strengths of case- and rule-based reasoning (Kang & Compton, 1994). As will be described in greater detail later, regardless of whether the cases are historical, occurring naturally in the domain, or hypothetical, the cases provide: grounding in the real world, automated validation and guidance for the direct acquisition of knowledge from domain experts.

This latter point is another key distinguishing feature of RDR. The simplicity of the KA approach is the result of observing the natural behaviour of experts at the Garvan Institute which typically involved looking at a case/problem, making a recommendation, and then, if prompted, providing some features of the situation to account for their conclusion. As shown in Figure 1, KA using RDR follows this same cycle and is performed by the domain expert, not the knowledge engineer (KE) (Compton *et al.*, 1991). In contrast, most ES approaches required a KE to conduct various KA activities and analyses such as structured interviews, protocol analysis, repertory grids, laddered grids and card sorts involving the domain expert from which knowledge would be elicited and encoded by the KE.

A number of knowledge engineering approaches based on Newell's (1982) Knowledge Level emerged to address the recognized KA bottleneck, that is, the difficulty in acquiring knowledge from experts due to their unavailability, unwillingness or inability to articulate all that is relevant of what they know or how they reason. Note that RDR does not see this situation as a shortcoming of the expert, but due to the nature of expertise. RDR resolves this problem by shifting the focus to validation of cases (including the identification of the correct conclusion and salient features), not the KE process. These knowledge level approaches included knowledge analysis and design support (Schreiber *et al.*, 1993, 1999) and generic tasks (Chandrasekaran, 1986), which provided a more structured methodology for acquiring knowledge and sought to reduce the KA effort by restricting the KA focus to a specific problem type and/or problem-solving method (PSM).

In the 1990s, the use of ontologies (e.g. Grosso *et al.*, 1999) were promoted as a way of providing structure and a shared set of terms and relationships for the concepts within the domain of interest. Reuse was seen as a key way to maximize the usage of knowledge, which had been effort-intensively acquired. Despite the benefits that these approaches bring, the RDR approach provides a counter-example to the view that complex descriptive models are necessary prerequisites for building KBS. Many of the problems identified with knowledge-level PSMs, ontologies and generic models are related to the inherent limits of descriptive models and the effort needed to interpret those models when they are applied. This situation is summed up by Chapman when he states:

> *Representation is generally thought to make problems easy; if a problem seems hard, you probably need more representation. In concrete activity, however, representation mostly just gets in the way.*
> (Chapman, 1989: 48)

A useful description of the types of models that are involved in capturing conceptual models is provided by Shaw and Woodward (1989). It is possible that experts have their own internalized model known as the mental model. When the expert articulates that model it becomes the conceptual model and the model that is developed from that is a model of the conceptual model, not the internalized model. To add to the difficulty in achieving a model that is truly representative of the expertise, the mental model may not correspond to how the problem is really being solved, the expert may be unable to articulate their mental model into a conceptual one and the KE may not interpret or represent accurately the conceptual model into the final developed model. Norman (1986) explains that when the developers' conceptual model of the users' mental models do not match, the design model will conflict with how the user expects to use the system. The key is to make the mental model explicit so that it is clear what the design model should be. This is an argument for building KBS using the RDR approach, that is, capture knowledge in the way the expert exercises their knowledge and then look at the models that underlie the knowledge. In this way RDR KBS becomes a mediating representation taking the user from what they do to how they think (Catlett, 1992).

By allowing the expert to incrementally acquire knowledge bounded by the case at hand as a simple extension to their normal work (Edwards & Compton, 1993), we address the KA bottleneck, together with maintenance and validation issues. It is for these reasons that RDR, unlike many ES approaches of the 1980s and 1990s, achieved commercial success.

It is not surprising that RDR has achieved user acceptance since it has been driven by its potential users. RDR were born out of the frustration experienced by KEs and pathologists seeking to maintain a traditional rule-based system known as Garvan ES1 (Compton & Jansen, 1988). Garvan ES1 was identified by Buchanan (1986) as one of the first four medical ES to reach routine use and was continuously maintained from 1984 until at least 1989. The goal of the system was to add clinical comments to patient biochemistry reports to assist general practitioners (GPs) in patient management. Experience with Garvan ES1 led to the development of the first RDR system in routine use known as Pathology Expert Interpretative Reporting System (PEIRS) (Edwards & Compton, 1993). RDR has evolved significantly since PEIRS, particularly with respect to the introduction of multiple classification RDR (MCRDR) and improvements to the human–computer interface.

In the next section we describe the two major RDR-based representations. In Section 3 we look at numerous implementations, applications and variations of the RDR theme and conclude with final remarks in Section 4.

2 The RDR approach

RDR uses a failure-driven approach to KA, where knowledge found to be incorrect by the domain expert, is patched locally within the context of the case on hand at the point where the knowledge was in error.

> *The most important resource for knowledge maintenance is a data base of cornerstone cases. These are cases which at some stage have required a change in the system's knowledge. Every time the*

> *system's knowledge is changed the interpretations for all the cornerstone cases are checked to see that the additions to the system's knowledge have been incremental and have not corrupted the knowledge.*
>
> (Compton & Jansen, 1990: 244)

Almost paradoxically, the RDR engine reasons non-monotonically over the knowledge by allowing exception rules to be added to override, and thus update, previous knowledge. In RDR, rules are never deleted or changed, at least in its purer implementations. Although RDR has undergone a number of transformations driven by the application and/or problem domain the variations are based on either single classification RDR (SCRDR) or MCRDR. Both representations allow a knowledge base (KB) containing exception rules to be built by a domain expert by interacting with cases with little or no involvement of a specialist KE. While at the outset a KE will probably assist with: determining the initial attribute–value (A–V) space, the structure of cases, preprocessing of the data if necessary and provide some training, the domain expert will become their own KE as they interact with the system specifying conclusions, cases and/or heuristics similar to the activities they perform in their role as a domain expert.

When we compare the manual acquisition of knowledge using RDR with the automated capture of knowledge via machine learning (ML), we see that RDR has a number of strengths. When we use the RDR KA cycle to classify cases and create rules, we are essentially identifying the salient features in the domain, the associated classes and the relationships between these, and getting classified cases whose classification has been validated by a human for free as part of the process. Supervised ML algorithms require cases to be classified upfront. In many domains this must be done manually and is prone to error and inconsistency simply due to human fatigue, memory limitations, emotions and so on (Wang *et al.*, 1996). By classifying cases whilst incrementally acquiring rules, we are forcing experts to check each classification against previous decisions they have made and justify why they were either wrong before or why this new case deserves a different classification. We discuss RDR and machine learning further in Section 3.6.

The following subsections introduce basic SCRDR and MCRDR and conclude with a summary of the key commercializations of RDR.

2.1 Single classification RDR

Single classification RDR (Compton & Jansen, 1988) has been likened to the animal game. Think of an animal (e.g. frog). Think of one or more features that identify that animal (e.g. makes croaking sound). Then think of another animal (e.g. a leopard with black spots). Along comes another animal that croaks and has black spots. What is it a leopard or a frog? It's neither, it's a toad. The current KA task is to determine what is different about a toad when compared to a frog or a leopard. This failure-driven process of acquiring knowledge within the context of the current case and past cases can be structured into a SCRDR KB as shown in Figure 2.

We begin with the situation where nothing is known, the empty KB, and define a default rule (rule 0), which is always true as indicated by the dummy rule condition $1 = 1$ in Figure 2. To speed up KA this can be the default or most common conclusion. For example, in the pathology domain the most frequent conclusion given is 'no conclusion' meaning that the pathology results indicate nothing abnormal and no further action. As an alternative to building a conceptual model or trying to articulate 'what we know' in the form of facts and rules about the domain, we instead acquire knowledge by interacting with examples or situations. Also in contrast to many other case-based techniques such as Case-Based Reasoning (CBR), the use of formal contexts in Formal Concept Analysis (FCA) (Wille, 1992) or Repertory Grids (Gaines & Shaw, 1993), RDR are designed to acquire knowledge incrementally as cases are seen or situations arise rather than identifying hypothetical or stereotypical cases through which knowledge can be uncovered or structured. The cases used in the SCRDR and first MCRDR examples are given in Table 1.

Figure 2 reveals a representation containing true (except) branches and false (if-not) branches. The inference engine starts at the top node (rule 0) and tests whether the next rule node is true

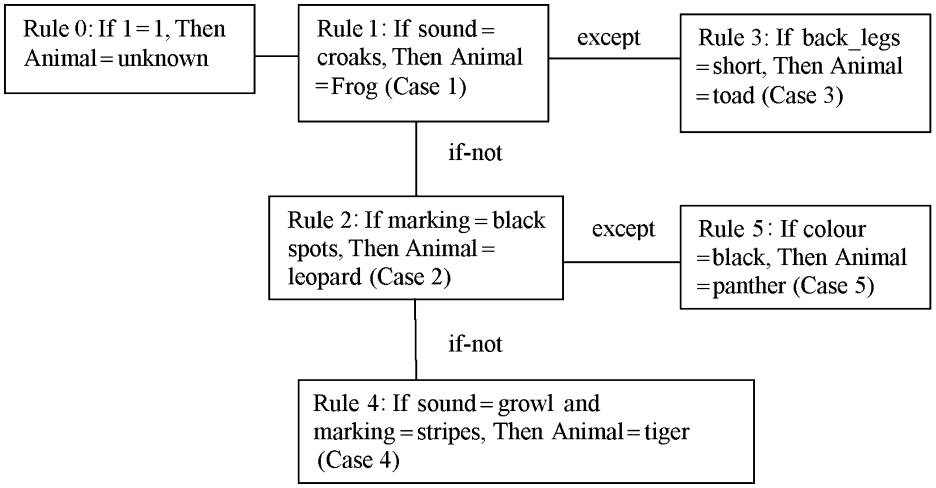

Figure 2 An SCRDR for our animal domain

Table 1 Cases in our animal domain

Case 1	Sound = croaks	Reproduces = in_water		
Case 2	Marking = black spots	Sound = growl	Suckles_young	
Case 3	Sound = croaks	Back_legs = short	Marking = black spots	Reproduces = in_water
Case 4	Sound = growl	Marking = stripes	Suckles_young	
Case 5	Marking = black spots	Colour = black	Suckles_young	

or false. If it is true it tests whether any child nodes are true or false and so on. The last true rule is the conclusion given. As indicated in the name, SCRDR only allows one conclusion. Referring to Figure 2, if we imagine that only cases 1 and 2 have been seen (and thus only rules 0, 1 and 2 exist), when we encounter a toad (case 3) that both croaks and has black spots, rule 0 is tested and fires, then rule 1 is tested and fires; rule 2 and any lower rules are not tested. Since the conclusion `animal = frog` is not considered by our domain expert to be true, an exception rule (rule 3) is added to rule 1 which overrides rule 1. To achieve this, the expert indicates that the correct conclusion is `animal = toad` and selects some differentiating feature between the current case (the toad) and the case associated with rule 1 (the frog case). Comparison is possible because each time a new rule is added the case that prompted the rule to be added is stored in association with that rule and becomes known as the rule's *cornerstone case*. The conditions of all rules on the pathway are to be included in the explanation of the conclusion. For example, rule 5 can be read as `IF NOT sound = croaks AND marking = black spots AND colour = black THEN animal = panther`. Ensuring that the two cases are differentiated ensures that the previous knowledge remains valid within the context it was acquired. In this way the whole KB does not need to be reevaluated and potentially modified as the new knowledge only patches the rule it extends. In early systems, such as Garvan ES1 discussed below, to assist with the validation process the user was provided with a list of differences between the current and cornerstone cases.

2.2 Multiple classification Ripple Down Rules

For some domains, if not most, one case may lead to a number of conclusions. To allow multiple conclusions for a given case, MCRDR has been developed (Kang *et al*., 1995). Despite the name, a conclusion can represent a recommendation, classification, state, intermediate conclusions/classification and/or action. For example, we might conclude that our animal is a vertebrate, a tiger, an endangered species and in need of further treatment or testing. The conclusion can even be to ask for further information or to rerun a case.

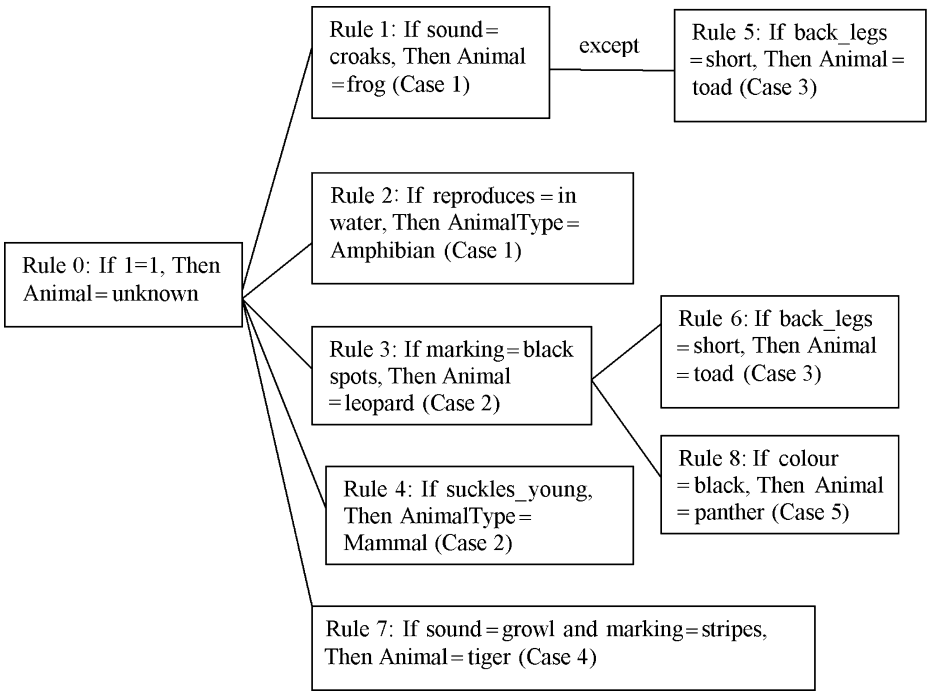

Figure 3 An MCRDR for our animal domain

In MCRDR there are only true (exception) branches. If a node evaluates to true, all children of that node are tested and so on. The last true node at the end of each pathway provides the set of conclusions given to the user. The justification for a conclusion is comprised of the conjunctions of all conditions on the pathway leading to the final true rule. Scheffer (1996) offers the following distinction between SCRDR and MCRDR, which he refers to as RDR and RDR-sets, respectively, in RDR-sets:

> *rules in a list of successors may fire synchronously, while in a RDR, each rule suppresses its successors. Decision Lists contain rules that suppress their successor – hence negation and exceptions can be expressed – but there is no distinction between rules, the instances of which are proper subsets, and rules that partially overlap (p. 279).*

Even for single classification domains, MCRDR has been shown to allow coverage of the domain quicker (Kang, 1996). We can see even in this small animal example that more knowledge has been acquired using MCRDR as compared with SCRDR. In Figure 3 the domain expert has seen the same number and set of cases (Table 1) as in Figure 2 but the end result is eight rules in MCRDR compared with five rules, one per case, in SCRDR. However the two KBs do not produce the same conclusions. The sample SCRDR only answers which animal has been found (frog, toad, leopard, panther or tiger). As only one conclusion can be given for each case in SCRDR, it is not possible to add one rule which classifies a particular case as a toad and a separate rule which classifies the same case as an amphibian as is possible in the MCRDR KB. In MCRDR it is possible to focus on different parts of the case to provide different rules and conclusions and create new rules for each area of focus or subdomain. For example, rules 1 and 2 use the same case, as do rules 3 and 4. Comparing the KBs in Figures 2 and 3, we see that rules 2 and 4 in the MCRDR KB use conditions (`reproduces = in_water, suckles_young`) not used in the SCRDR KB. These conditions could have been chosen to provide the same conclusions given in the MCRDR KB (`AnimalType = Amphibian, AnimalType = Mammal`) but it would not have been possible to also enter separate rules for those cases concluded (`Animal = Frog, Animal = Leopard`). To provide the same conclusions the SCRDR would have to provide compound conclusions based on a mix of conditions (for e.g., `If sound = croaks and reproduces = in_water`,

`Then Animal = Frog and AnimalType = Amphibian`). However, this would mean that it is not possible to add one rule to cover all cases of amphibians as can be done in the MCRDR example. Instead every rule conclusion would need to contain the animal and animal type classification. Even less satisfactory with the compound conclusion solution is the explanation provided by the system as it does not clearly link which attributes result in which conclusion. Further, while the use of a compound conclusion to provide multiple conclusions is a viable alternative for this set of cases as every animal also has an animal type (i.e., Animal IS_A subclass of the AnimalType parent class), in another domain, set of cases (for e.g., where information regarding Animal Type is missing) or class type (such as what food the animal should eat or whether the animal is a herbivore, carnivore or omnivore which is not a sub or superclass of mammal but potentially a subclass of many animal types), it may not be possible to provide a conclusion which covers the classification for all the classes being considered. One alternative is to have a different set of cases to those given in Table 1 which include A–V pairs relating to either the animal or animal type, but not both. However, as RDR is an incremental technique which expects to handle cases as they arise in the real world, structuring cases so that some of the case information goes into one type of case and other information goes into another type of case would probably be unsatisfactory and artificial.

For the SCRDR example given in Figure 2, if the domain expert later decided to update the KB to provide a classification for both animal and animal type they would need to override each of the five rules, with a compound conclusion resulting in a KB with 10 rules. The MCRDR KB only required eight rules to provide the same classifications. If more new classifications are required in these examples, the size of redundancy in SCRDR increases exponentially. MCRDR thus produces more compact KB with less redundancy and requires fewer cases to be seen to cover multiple subdomains.

However, MCRDR is not always better than SCRDR, particularly for single classification domains. In the examples given if we begin KA with the goal to provide a classification that includes both animal and animal type, we only need five rules in the SCRDR KB and six rules in the MCRDR KB. Note that if the order of cases were changed and the toad case (case 3) was tested before the leopard cases (case 2) or if the expert selected two conditions in rule 3 (e.g. `marking = black_spots and sound = growl`) to distinguish this animal, rule 6 would not have been needed and MCRDR would also require five rules to cover the same set of cases and produce the same set of compound conclusions.

We can see that the parent rule (rule 3) relating to the leopard case (case 2) is an over-generalization. The exception rule (rule 6) in Figure 3 attempts to correct this in light of a case for a toad (case 3). A number of other alternatives were available to the expert to correct the overgeneralization. One likely method is for the expert to decide they made a mistake in the first place and had not been specific enough. Rather than add the exception rule 6 in Figure 3, which results in the definition of a leopard as having black spots and not having short back legs, they may choose to stop the conclusion given by rule 3 by adding a stopping rule (rule 9) as shown in Figure 4. In this example, when a case for toads appears (now case 5 as shown in Table 2), the toad will be correctly classified because of exception rules 7, 8 and 9 overriding rules 1, 3 and 5 in Figure 4.

Not illustrated, alternatively, the domain expert may decide that they want to undo their overgeneralization error in rule 3 and start again by making rule 9 an unconditional stopping rule, that is a rule that has no conditions or a dummy condition (such as 1 = 1 in the default rule root node 0) and a STOP conclusion which is equivalent to `conclusion = no_conclusion`. This would mean that the leopard case (case 2) would receive no conclusion by the system and thus the case needs to be reclassified. To rectify this, a new rule can be added at the top level (i.e., as an exception to rule 0) which states `marking = black spots and sound = growl Then Animal = leopard`. When the panther case is encountered it is likely that an exception rule is added to the leopard parent rule which states `colour = black Then Animal = panther`.

Note that in order to demonstrate another feature of MCRDR in Figure 4 a new case containing the feature `eats = flies` (but not the A–V pair `sound = croak`) has replaced the

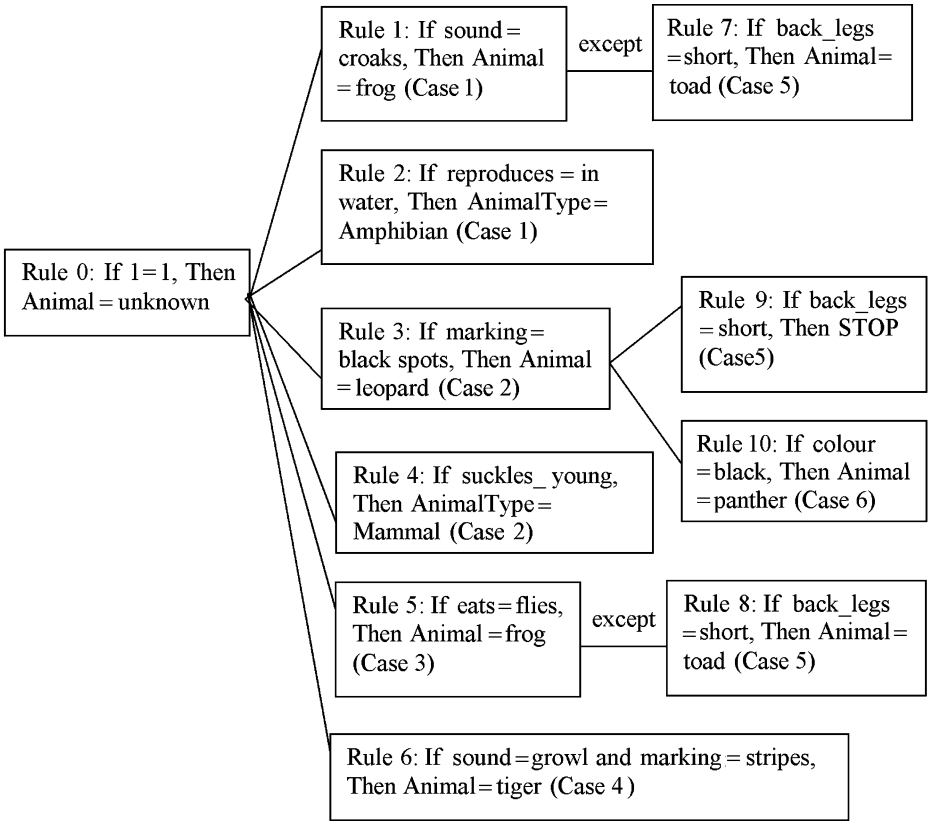

Figure 4 An MCRDR for our animal domain using a conditional STOPPING rule

Table 2 Alternative set of cases in our animal domain

Case 1	Sound = croaks	Reproduces = in_water			
Case 2	Marking = black spots	Sound = growl	Suckles_young		
Case 3	Eats = flies	Reproduces = in_water			
Case 4	Sound = growl	Marking = stripes	Suckles_young		
Case 5	Sound = croaks	Back_legs = short	Marking = black spots	Eats = flies	Reproduces = in_water
Case 6	Marking = black spots	Colour = black	Suckles_young		

previous case 3, the toad case has moved to case 5 and the panther case has become case 6. As shown in Table 2, we now have two frog cases (cases 1 and 3) that contain different features thus requiring the two different rules we see in rules 1 and 5. When the toad case comes along, rules 1 and 5 both need patching (i.e. an exception rule needs to be added). We can see that rules 7 and 8 are added to fix the two incorrect frog conclusions, which the system provided for the toad case. Patching in more than one place to correct the same conclusion can occur in MCRDR because multiple true pathways are possible. To reduce the repeated KA effort to correct the same error, in commercial implementations automated patching is provided that allows the user to define the first patch and the system to automatically add exception rules to any other rules that would give the same misclassification (Compton *et al.*, 2006). Despite this seeming inefficiency, empirical and simulation studies have shown that the depth of correction is usually two or three rules, or conversely a sufficiently precise rule is provided by the expert after seeing two to three cases (Kang *et al.*, 1995), so repetition is in practice not a major problem.

The use of a difference list in early SCRDR was adapted for use in the first MCRDR system as the Garvan ES1 data was used for development and testing and because the difference list was useful to automatically generate new rules in the simulation studies used to evaluate MCRDR. Many RDR systems do not use the difference list as part of KA as the many combinations of differences resulted in a list that was very long and therefore too difficult for the expert to review. Fundamentally, SCRDR/MCRDR focuses on how to validate systems with cases by comparing the current case with associated cornerstone cases and thus difference lists are not critical.

2.3 Commercial implementations of Ripple Down Rules

RDR in some form has been in routine use in an industrial setting since the development of PEIRS in 1989. Today the most noteworthy fruit of this work can be found in the Labwizard commercial RDR system by Pacific Knowledge Systems[1]. Recent communications estimate that to date about 30 million patient reports have been processed by Labwizard, and about 100 KBs have been built by 14 chemical pathology laboratories[2]. In contrast to traditional rule-based systems, where there is a noteworthy increase in KA time as the KB grows due to the need to ensure that inconsistencies are not introduced in other parts of the rule tree, in Labwizard a rule takes on average about 1 minute of expert time and is independent of KB size, even with KBs of thousands of rules (Compton *et al.*, 2006). This is considerably less time than when the interpretations had to be written on each report, and also ensures greater consistency in the interpretations. Effectively, the GP receives an interpreted report that has been reviewed by two experts, one human and one machine, the former high in commonsense and deep smarts, the latter not prone to boredom, fatigue or mood changes with more reliable memory and computational ability. The value of RDR to the GP was validated in an initial independent survey report by Illawarra Regional Information Service (IRIS) Ltd research and is further confirmed in the fact that the KBs continue to grow despite the change to a per-usage pricing model.

However, RDR has moved beyond both the research and pathology laboratory and can be found in a number of other commercial products. The RDR research led by Byeong Ho Kang, currently at the University of Tasmania, resulted in the spinoff company KMAgent[3] which uses RDR to add intelligence to Web browsers. Kang has also introduced a number of Korean firms to RDR for applications such as Help Desk Systems. Further systems using RDR technology and concepts are underway with various large commercial partners in a range of fields including medical applications besides pathology and email management (see the EMail Management Assistant (EMMA) system in Section 3.5).

RDR techniques are provided by a number of consulting and professional services companies such as Ucube[4] which offers 'a very rich rule engine component, developed based on the RDR algorithm' and SriCom offers the Likewise[5] Proforma Tool which assists decision making by using a series of items to prompt the user for relevant information following the RDR approach to KA and exception representation.

On a larger and more international scale, the Sonetto system[6] has been developed by the Ivis Group[7] for Tesco.com, the world's largest on-line grocer with a turnover exceeding $2B. RDR has been used to create hundreds of thousands of business rules, a more popular term in industry, to manage product knowledge which is being used by affiliate sites (e.g. Kelkoo) and as a product

[1] http://ww.pks.com.au/

[2] http://www.cse.unsw.edu.au/~compton/commercial_RDR.html

[3] http://www.kmagent.com/

[4] http://www.ucube.net.au/Products/tabid/119/Default.aspx

[5] http://likewize.com/about.html

[6] http://www.eurobizrules.org/Uploads/Files/Sarraf_2c_20Q_2.pdf, accessed February 20, 2008.

[7] http://www.ivisgroup.com/

recommendation system to their customers. The Ivis Sonnetto product has won two Microsoft Awards, including one for innovation[8] and a 2006 award for Retail Affiliate Management[9].

The Yawl Group, which stand for Yet Another Workflow Language, have incorporated RDR into their workflow management system which they use in their consultancy and training services for companies developing business process management. The product is the outcome of a collaborative research effort between QUT's Business Process Management research group and Eindhoven University of Technology. In addition to a number of other techniques such as petri-nets, RDR are used to provide a 'unique solution to dynamic workflow'[10].

The impact of RDR in the commercial world is not always easy to identify as it tends to be a component within a larger system and the ideas have been available in the public domain since the 1980s. One example is a system known as Erudine[11] which sounds like RDR in the description contained in one of its whitepapers[12] and was previously known as Rippledown Solutions[13]. Among the uses of Erudine, is the re-engineering of legacy systems where the legacy system acts as a case base from which a new system is developed.

3 Variations, extensions and implementations

Over the past two decades many variations and extensions to SCRDR and MCRDR have been developed, particularly to the inferencing cycle, according to the problem domain and depending on the system and other techniques and technologies being used. While there have been numerous implementations of RDR, each one has essentially acquired knowledge in the same manner, by presenting cases to an expert who accepts a conclusion or creates a new rule with the correct conclusion attached as an exception to the incorrect rule. Standard tools included in most implementations of RDR were the ability to browse and visualize the tree, browse individual traces and examine rules and their status based on the current case. Various statistics were computed and stored such as a count of the number of times a conclusion is used, the complexity of individual rules (the number of conditions in the antecedent), the frequency that whole conditions occur, the full complexity of a path, the number of rules in spine subtrees and a list of conclusions and which rules use them. Numerous other statistics, debugging, browsing and evaluation features and functions were added to prototype systems according to the issues under consideration. The research in this section is presented in rough chronological order but also grouped thematically resulting in some loss in historical sequence.

3.1 Early systems

As introduced earlier, the first RDR system PEIRS was created for the pathology domain around 1990. This was the first of a number of systems based on SCRDR, such as the XRDR general purpose classification system. Noteworthy SCRDR extensions are Time Course RDR (TCRDR) which was used by PEIRS and Recursive RDR (RRDR) (Mulholland *et al.*, 1993). TCRDR used various functions and features to handle time course data. For example, if there were data covering five time periods the user could define a function CURR to determine which A–V pair would be used in evaluations of the current value of that attribute. This type of preprocessing of the input data is typical of many of the implementations of RDR and does not affect the way

[8] http://www.microsoft.com/emea/presscentre/pressreleases/rad2006winnerspr_1522006.mspx

[9] http://www.nabble.com/CG%3A-Sonetto-CG-based-rule-and-search-engine-wins-awards-for-Retail-Affiliate-Management-to3086701.html

[10] http://www.yawl-system.com/

[11] http://www.erudine.com/

[12] http://www.erudine.com/downloads/whitepaper_tacit.pdf

[13] http://web.archive.org/web/20051222121140/www.rippledownsolutions.com/page.aspx?id=27

KA or inferencing is performed. On the other hand, RRDR did require a modification to the inferencing process involving repeated inference cycles and using multiple machine-learnt SCRDR, one for each component of the ion chromatography method (Mulholland *et al.*, 1996). On each cycle the last true conclusion at the end of each SCRDR path were returned. A heuristic was then used to choose which conclusions to accept and these were used as inputs and inferencing commenced from the beginning again. Another implementation combined RRDR with Possible RDR (PRDR) (Mulholland *et al.*, 1993) and reported all last true conditions after taking any branches that might possibly be true. PRDR considers a subset of the branches used in RRDR. Both of these techniques can be difficult to manage as often too many alternatives are provided and it is hard to determine the useful ones.

Interactive RDR (IRDR) is a technique that allows the RDR system to prompt the user for more information when required (personal communication, Quoc Thong Le Gia). The user may enter a value or indicate that the value is 'unavailable'. The response is added to the case, which is then reevaluated. When changes are made to the case, no matter how small, it is possible that another path will be found through the tree. Some, unpublished, simulation studies have been done using IRDR that yielded poor results. It is conjectured that it is very difficult for the user to anticipate how the path will change. Therefore determining the appropriate change to an A–V pair to test out predictions is very complex. Later work involved the use of rules that are added by the expert to state what A–V pair should be requested next when the value of a certain attribute is known. A further suggested refinement was to use information gain to interactively compute the key attributes that should be queried. Another alternative to using information gain alone would be to use the rules as the input cases and the statistics collected by *credentials* (described later in this section) to weight each rule according to the number of times the rule has correctly fired. Use of these statistics is important because, unlike the use of algorithms like C4.5 with examples, there is only one of each rule and the importance of one rule over another cannot be determined without some measure of usage. Rules whose conclusion is to ask a question, like the approach used in Collaborative RDR (CRDR) in Section 3.7, can also be incorporated to override the suggestions made by ML.

Another implementation (Edwards, 1996; Kang, 1996), known as WISE, first determined the conclusion of a case using standard RDR and then found all other branches that gave the same conclusion. The goal of the study was to determine the usefulness of storing the history of corrections, provided in an RDR KBS structure. The pathways below each satisfied rule node were compared (normally only one pathway is traversed and kept in SCRDR). These multiple pathways postulated four possible patterns of knowledge correction and could be used to suggest that a conclusion may be wrong. This would add even further to the explanatory power of the knowledge and allow comparison between alternative classifications of the data.

As a follow-on from WISE, Edwards looked at building RDR KBS that use reflective learning through what is termed *prudence* and *credentials* (Edwards, 1996). The idea behind *prudence* was to reduce some of the brittleness problem that ES suffer from when they reach the limits of their knowledge. This was achieved by altering the behaviour of the ES rather than the knowledge contained therein. WISE developed into Feature Recognition Prudence, which was implemented to see whether a different conclusion could have been reached by an alternative path by checking the children of other nodes with the same conclusion. Another approach was termed Feature Exception Prudence, which allowed the system to know when it was reaching its boundaries. This was achieved by keeping a context profile containing the permissible features for each interpretation. When an inference was performed all A–V combinations (features) in the case, not just the rule conditions, could be evaluated against the permissible range of values for each attribute in the context profile. If a feature was not found in the context profile the pathologist was warned that the conclusion may be incorrect. In the *exceptions* study it was found that knowledge in the form of *filters* was needed so that in certain circumstances data would not be checked, such as checking the value of the attribute *ovulatory* when the patient is a male. One alternative to the use of filters is to provide a secondary KBS that contains rules with valid combinations and other knowledge required to support reflection to be used in conjunction with the primary KBS containing the main RDR rules.

The other method of reflection was the use of *credentials* much in the same way as we may seek the qualifications of an expert before deciding to accept their advice. *Credentials* kept track of various statistics such as how many times a rule had fired and how many of these had been accepted by the pathologist. With these statistics the user can determine how the accuracy of knowledge in this rule can be compared to the accuracy of the total system and decide whether to accept or reject the system's recommendation.

3.2 Acquiring control knowledge

RDR have been successfully extended to acquire complex control knowledge needed for a flight simulator (Shiraz & Sammut, 1997). Much of the knowledge required for this task is subconscious or tacit and cannot be acquired through techniques such as interviews. By reasoning about cases the operator is able to evaluate and modify the KBS. Dynamic RDR (DRDR) was developed based on SCRDR. However, unlike SCRDR, DRDR handles four KBS concurrently and produces multiple conclusions one for each of the four conclusions regarding Elevator, Flaps, Roll and Throttle. To further assist the dynamic system with learning, ML was used to modify and complement the rules created by the flight operator using the flight log. The algorithm was known as Learning Dynamic RDR (LDRDR) and unlike most ML algorithms it learns incrementally by determining when an action was performed and using the action and attributes that caused the action to form a new rule. This work had begun while MCRDR was still under development and it would be interesting to compare the benefits of DRDR compared to MCRDR for the same task.

More recent work also involving the incremental acquisition of control and parametric knowledge is in the field of lung boundary extraction from High Resolution Computed Tomography scans (Misra *et al*., 2004). Scan imaging involves multiple possible procedures that vary according to the quality of the image, type of image, particular domain or other circumstance. To address the need to acquire procedural knowledge ProcessRDR was proposed. In contrast to the work of Park *et al*. (2001) using MCRDR for feature extraction in this domain, ProcessRDR was designed to manage the knowledge related to the process of feature extraction where separate and multiple ProcessRDR were used to carry out some specific image processing. The ProcessRDR were connected in a sequence and thus impacted on one another. The technique addresses the problem of *ad hoc* procedures and would be applicable to other processing system domains. The key open issues being explored were the dependence between ProcessRDRs and how to handle the large number of cornerstone cases which occurred in some worst-case scenarios.

3.3 Hierarchies of RDR: Nested Ripple Down Rules

Nested RDR (NRDR) (Beydoun & Hoffmann, 1997, 2000) allows multiple SCRDR trees to be combined into a hierarchical conceptual structure. The initial work was concerned with capturing search knowledge and allows concepts and rules at various levels of abstraction to be defined and reused. NRDR makes use of cases and cornerstone cases but does not create a difference list to assist with feature/condition selection. The chess domain was used as the testbed for this work but the work was concerned with human search knowledge in general. For every concept to be defined, a simple SCRDR tree is used. Rule conclusions within a concept definition are Boolean and can be used as conditions in other concept definitions to form a hierarchy of concepts. Similar to the earlier work by Mulholland, multiple reasoning cycles are used to traverse the multiple SCRDR trees to provide conclusions based on the set of trees. Realizing the importance of providing the user with a visual representation of the rules the system developed, SMS1.2 allowed the user to view the rules as a tree structure similar to those provided in other implementations of RDR. Figure 5 shows three SCRDR trees that form part of a concept hierarchy relevant to the Animal domain, which given a case {hasPlacenta, noFeathers, noScales, warmBlooded} will first conclude {Mammal} then {hasHair) and finally {Mammal}.

The ability to share and reuse simple SCRDR KB to provide a structured knowledge representation at multiple levels of abstraction is its key strength. It is noted that 'this hierarchical

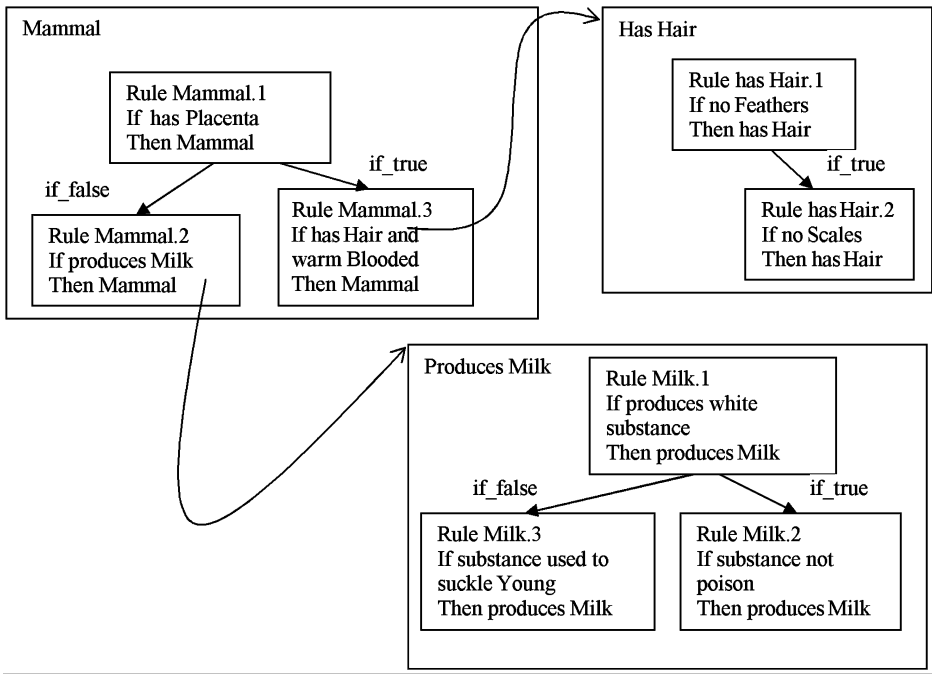

Figure 5 A subset of animal-related SCRDR showing how knowledge can be structured and reused in NRDR

structure of the knowledge base causes problems for keeping the entire knowledge base consistent when a single concept needs to be altered' and 'a more profound update problem is dealing with inconsistencies due to localized updates in the hierarchical knowledge base' (Beydoun & Hoffmann, 1997: 7). While they provide some solutions to these problems it appears that the imposition of more structure and levels have impacted on the ease of maintenance in standard RDR. Also, as can be seen from Figure 5, a change in one concept can change another concept and thus navigating around concepts and also keeping in mind the overall interrelationships is quite different to the simply looking at a sequential pathway of rules. To take some of the cognitive load from the user, the user interface plays an important role in assisting the user with navigation and the management of the hierarchies and interrelationships.

3.4 The role of ontologies and the reuse of RDR knowledge

Nested RDR, as described earlier, provided a means of reusing RDR concept hierarchies. An alternative ontological approach to KA using RDR can be found in the system known as Ripple down rule-Oriented Conceptual Hierarchies (ROCH) (Martinez-Bejar *et al.*, 1997, 1998a). Using ROCH the expert defines a conceptual hierarchy of the domain from which cases are first developed followed by rules to cover those cases. The hierarchy includes IS-A and PART-OF relationships and is verified by the system and validated by the user as it is developed. ROCH was extended to cover fuzzy domains with the development of fuzzy ROCH (Martinez-Bejar *et al.*, 1998b). As with NRDR, but unlike mainstream approaches, ROCH is an incremental technique allowing rules and/or the conceptual hierarchy to be developed on a case-by-case basis by the domain expert.

The work by Lee and Compton (1995) considers whether knowledge in an existing RDR KBS could be reused to find causal links to support causal modelling. In this approach it was found that additional knowledge from the expert, in the form of causal links, was often required. The work of Richards (1998, 2000) explored a wider range of potential *activity-reuses* of RDR knowledge to support critiquing, user modelling, explanation, tutoring, what-if analysis and hypothesis testing. This was achieved in a number of different ways involving the use of a nearest-neighbour

algorithm to identify similar concepts and the use of FCA (Wille, 1992) to automatically generate conceptual hierarchies of the RDR KB. The work was further applied to compare and reconcile multiple sources of expertise contained in multiple KB (Richards & Menzies, 1998) and to multiple stakeholder viewpoints in requirements engineering (Richards, 2003). The technique offered an alternative to labour-intensive specification of ontologies that combined rapid, incremental and manual KA using RDR with automatic generation of ontologies using FCA to enable reverse engineering of ontologies (Richards, 2004). Suryanto and Compton (2000) took this work further to allow learning of classification taxonomies and the relations of mutual-exclusivity and similarity between classes in addition to the subsumption relation provided by FCA.

3.5 RDR, the Internet and text classification

The Internet has brought with it some new challenges and opportunities for extending RDR. The challenges often relate to the effective and efficient utilization of the vast amount of resources now accessible. A number of approaches have been developed to support retrieval of Web-based documents. Kang *et al*. (1997) integrated key word searching with CBR indexing to provide a filter and guide searching. Kim *et al*. (1999) developed a prototype to assist the expert to create a KB of keywords (rule conditions) linked to documents (conclusions). To further improve searching, Kim and Compton (2004) used FCA to generate a browsable hierarchical conceptual clustering of documents. To improve the user's query, Cho and Richards (2006) developed an algorithm known as Admixture MCRDR and FCA which incorporated the creation of rules using MCRDR and FCA and achieved better precision results compared to some common information retrieval techniques such as support vector machines (SVM), term frequency-inverse document frequency (TFIDF) and naïve Bayesian with Threshold. The call centre prototype CRDR system (Vazey & Richards, 2006a), described in Section 3.7, offered a Web-based solution to bring together knowledge in the form of URLs, documents, databases, technical specification, etc., from many vendors and sources.

Personalized retrieval of Web information has been explored in the work of Park *et al*. (2004). This work seeks to overcome the cognitive, linguistic and knowledge representation barriers, which occur when there is: a lack of structure and context, and communication difficulties and mismatches between domain experts, KEs and between the ways that different people represent and use knowledge. Similar to Catlett's (1992) claim over a decade ago that RDR provide a mediating representation to bridge some of this gap, Park *et al*. (2004) use a 'folder structure user interface … as a mediating representation method and difference lists and cornerstone cases as [an] intermediating representation' (p. 64). An MCRDR based algorithm is used to incrementally classify documents using the typical RDR failure-driven KA process, which is triggered when the classifier suggests a folder deemed inappropriate by the user. Usability experiments have shown the technique to be simple, yet robust, and to work well in domains where machining learning training sets are hard to create and knowledge is constantly increasing.

Email management has been the focus of a number of RDR projects. Cho and Richards (2004) have used a combination of the naïve Bayesian with Threshold algorithm with MCRDR (BayesTh–MCRDR) achieving a 3% improvement over the Bayesian with Threshold algorithm alone. This is achieved by constructing a related word KB to guide classification. A similar but more comprehensive approach to email management which includes message sorting into virtual folders, prioritizing, reading, replying, archiving and deleting emails has been developed in the EMMA by Ho *et al*. (2003). As in the BayesTh–MCRDR approach, EMMA uses MCRDR for classification but offers three alternative uses of naïve Bayesian classification to enhance coverage and usability. This work has been evaluated with users and achieved accuracy between 95% and 99% during the trial period. As mentioned in Section 2.3, the system is being commercialized and a patent for the work has been lodged.

Another email system called Intelligent Electronic Mail (IEMS) (McCreath *et al*., 2006) combined handcrafted rules in the form of decision lists which was a modification of basic MCRDR (i.e. inferencing continued instead of stopping after a stopping rule was found) with machine learnt

rules using an approach which generates an explicit hypothesis based on an instance-based learning technique. The handcrafted rules were found to be more precise though make fewer predictions, whereas, the learnt rules were less precise but were able to make predictions on a larger percentage of incoming messages. Hence by considering the handcrafted rules first and the learnt rules second, they were able to improve the overall performance.

Many of the Web-technology approaches described earlier have moved RDR beyond case classification into text classification (e.g. EMMA), information extraction and natural language processing. An example of the latter is the use of language technology techniques such as tokenizers, part-of-speech tagging and semantic tagging, together with RDR to extract positive attributions from scientific papers (Pham & Hoffmann, 2004a), thus reducing some of the information overload facing scientists. The work builds on RDR concepts such as incremental acquisition of rules in response to misclassified text segments (which take the place of cases) but extends the representation significantly with a rule annotation language used to specify the rule conditions of the exception rule covering that segment. Alternatively a new rule may be added for the segment if no inconsistencies are introduced. When compared against gold standard text mining algorithms (that is, C4.5, Naïve Bayes and SVM), the KB created achieved far superior results for sentence classification of the same sentences. For information retrieval, the initial experiments yielded precision rates of at least 74% and recall rates of more than 88% on unseen corpora. To support the rapid prototyping of language technology systems, KAFTIE (Knowledge Acquisition Framework for Text classification and Information Extraction) has been developed, which in addition to the annotation-based rule language includes a customizable Shallow Parser (Pham & Hoffmann, 2004b).

Further language technology research using RDR is in the area of text summarization. The Knowledge Acquisition Framework for DIScourse processing (KAFDIS) assists in the acquisition of the knowledge needed for the reliable construction of the discourse structure graph and the level-of-detail tree based on cue phrases (Hoffmann & Pham, 2003). The expert uses KAFDIS to identify and justify, thus indirectly developing rules, the relationships between two sentences. The reasonably simple process outputs an annotated training corpus including a complex discourse structure graph built up through elaboration of various subgraphs in the form of level-of-detail trees. The goal is to develop an interactive document viewer incorporating more extensive KBs for the automatic processing of scientific papers.

3.6 RDR and machine learning

As we saw earlier, RDR has frequently been used in conjunction with machine learning (ML) approaches. Earlier work connecting RDR and ML involved the use of ML to produce RDRs, mainly due to the compactness of the knowledge representation (Catlett, 1992; Gaines & Compton, 1992; Kivinen *et al.*, 1993; Siromoney & Siromoney, 1993; Scheffer, 1996). Scheffer's (1996) work is of particular importance as it does not simply employ exceptions or RDR as an alternative representation, but the work provides an algebraic foundation for RDR and includes a comparison with a number of similar algorithms (Vere, 1980; Wrobel, 1988; Helmbold *et al.*, 1989; Bain & Muggleton, 1991; Kivinen *et al.*, 1993) used to induce exceptions or which use counter-examples. Scheffer describes SCRDR as a list of rules in which each rule may be associated with another list of (exception) rules with the following advantages.

> *The implied concept of locality (an exception is applicable only if its next-general rule is applicable) makes RDRs comfortable for human needs: rough rules can be expressed first, the exceptions of which can be modelled as a refinement of the hypothesis later.*
>
> (Scheffer, 1996: 279)

To achieve even smaller KB size and handle continuous attribute domains, Scheffer offers the Cut95 algorithm that rewrites a given RDR and a new example, where rules are conjunctions of interval constraints to yield a classifier through bottom-up generalization of examples and top-down specialization of exceptions, according to the RDR manner.

As pointed out by Scheffer (1996), the exception structure which inherits the conditions of parent rules, provides a shorter description length than an equivalent decision list or traditional production rules which could be termed 'flat' rules. A number of ML researchers have recognized this strength of RDR. Early work includes Gaines and Compton (1995) which used the Induct algorithm to automatically generate compact RDR rules from cases. Yao *et al.* (2005) have also used RDR together with the Prism and Induct algorithms to produce a binary tree for the purpose of security information analysis. Pham and Sammut (2005) have used the Cut95 (Scheffer, 1996) algorithm to acquire domain specific object recognition rules based on inputs from the vision systems in the Aibo soccer robots with results comparable to handcrafted rules but with the advantages of KA being incremental, automatic and up to date.

The combination of RDR with ML makes a lot of sense. In many cases, manual KA using RDR is replaced by a machine learner, which outputs knowledge in the form of RDR exceptions. These supervised ML techniques tend to be batch processors such as the RIDOR RDR learner available in WEKA. The manual acquisition of rules by humans in most RDR approaches is sometimes seen as a weakness that could be overcome by using automated ML techniques. However, in supervised ML, humans are typically required to classify the cases or annotate the text. As was found by Pham and Hoffmann (2004a), the KB they produced using manual RDR was far superior to the ML approaches they evaluated on the same text. They found that time taken to add rules was a small fraction of the time when compared to the time needed to classify the sentences as rules may cover many sentences. They conjecture,

> *This suggests that our approach uses the expert's time much more economically than supervised machine learning approaches which only utilize the class label as the indication of what to do with a sentence.*
>
> (Pham & Hoffmann, 2004a: 181)

Pham and Hoffmann see real benefit in a mixed initiative approach, where the system may suggest rules that can be modified by the experts. As many RDR researchers have found, the inclusion of the human in the loop at optimal times can greatly reduce the search space and overcome some of the brittleness problem as humans bring their commonsense and experience to bear on the problem at hand. Also by working with the human, the amount of training data can often be greatly reduced thus reducing the number of cases which need to be classified (Pham & Hoffmann, 2004a).

A body of work has considered the combination of manual KA using standard RDR and a human with automated KA using ML. The work sees a synergy between RDR and ML as both approaches seek to express a minimum description to cover the knowledge in the domain. The initial work by Wada *et al.* (2001) was motivated to incorporate ML with RDR to address the limitations of human expertise, capture up to date knowledge in changing environments and take advantage of large datasets available in many domains. They suggest that the proper balance is to use humans in the early development of the system when fewer cases can be found and to use ML to refine the system when sufficient cases exist. The Minimum Description Length Principle (MDLP) algorithm is used in conjunction with RDR to provide a more structured and common strategy for integrating machine and human KA. By using the description length, data can be used to supplement human expertise, particularly important where expertise may be inadequate. In general the technique developed a KB equivalent to the rules that can be produced using C4.5 from the same amount of data. The approach produced SCRDR binary trees but the authors suggested that the technique could also be used with MCRDR and might be useful to provide guidance in making the decision of where to place new rules in an RDR, such as at the end of a rule pathway or placed higher up the tree as a new rule pathway (Yoshida *et al.*, 2004). The study by Yoshida and Motoda (2005) conducted a performance evaluation of the method involving (1) switching the knowledge source from expert to data and vice versa at any time and (2) additionally changing the environment, that is, the class distribution. The results were better for all methods except the use of data only when compared with the results for C4.5. MDLP was found to work

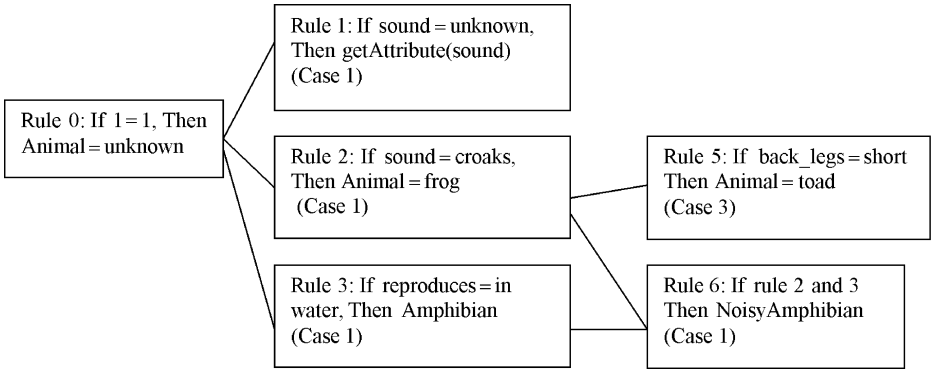

Figure 6 A partial CRDR for our animal domain

very well in general as a unifying principle, and timely use of a human expert was found to help reduce the search space. Further, the studies showed that knowledge deletion and pruning are shown to be very effective when an environment changes over time.

3.7 Collaborative RDR (CRDR)

CRDR[14] has been proposed to address the volatility and complexity of the call centre domain. In the call centre, cases are worked up by multiple people around the world and around the clock. Cases continue to change until the recommendations or answers offered to the client result in a satisfactory outcome. Given the complex interactions between products, platforms and vendors and the large range of solutions including URLS, databases, documents, hints, patches, etc., in the call centre of the multinational research partner, weeks or even months may pass before a particular course of action may be confirmed. Thus CRDR allows rules and cases to change and multiple people to work on the same parts of the KB. While standard MCRDR allowed change to rules by adding exception rules and change to cases by entering a modified case, which was then treated as a new case, CRDR actually allows the original rule or case to be modified. To support collaborative KA and maintenance, multiple views of the knowledge are allowed. Inconsistency is managed via extensive tracking of the states ('live', 'registered', 'pending', 'accepted' or 'rejected' (Vazey & Richards, 2004)) and relationships between the various elements in the CRDR KB. Any detected conflicts will be brought to the attention of one or more domain experts and reconciled collaboratively. To support reuse and incrementally acquire cases over time various new functions can be used in conclusions and rule conditions (such as the getAttribute(attrName) conclusion type[15] to request information from the user) (Figure 6).

To minimize redundancy and encourage reuse, CRDR offers a new representation, which resembles a network rather than a tree or set of decision lists. In SCRDR a node can have one parent and up to two children (a true and false branch). In MCRDR a node can only have one parent but multiple children who in turn can have multiple children. In the CRDR representation, a node can additionally have multiple parents (see Figure 6). Further, SCRDR or MCRDR do not explicitly distinguish the type of conclusion being provided even though implicitly the type of

[14] To reflect how rules and cases are 'worked-up', the approach was initially called Interactive Recursive RDR (Vazey & Richards, 2004). That name reflected that it shared the goals of earlier separate work into interactivity and recursion, which were described under early systems. Most recently the technique has become known as 7Cs as it supports Collaborative Configuration and Classification of a stream of incoming problem Cases via a set of ConditionNodes, that is, rulenodes, linked to their Classes and associated Conclusions. Here for simplicity the technique is referred to as CRDR.

[15] The implementation of CRDR includes several conclusion types (e.g. stop, getAttribute(attrName'), showfile(fileName), advise(errorCodeNNN), refer(RuleNodeID)) and attribute types (e.g. Text, URI, combo, imageURI, case(), file(), image(), refer()).

conclusion could differ (e.g. an action or classification). CRDR allows conclusions that are classifications to be identified as such and reused or 'referred to' in other rules. Data structures have been created and changes have been made to allow: multiple parents; the labelling of nodes as classes; reuse of nodes; the use of intermediate conclusions; the order of rules to be changed and thus the order of conclusions presented to the user to be changed. Also standard MCRDR inferencing has been enhanced to also allow recursive evaluation and backtracking as in the case of intermediate conclusion rule nodes, which allow the engine to jump to a rule in another part of the KB and backtrack to the root node. This is similar to the inferencing mechanism used with multiple SCRDR KB for configuring an ion chromatographer (Mulholland *et al*., 1993).

The ability to change rules and cases is a major departure from the underlying RDR philosophy to provide intuitive and low-maintenance incremental KA. While, in practice, commercial systems may relax the RDR philosophy to allow variations such as occasional editing of rules, automatic patching of all affected rules or non-differentiation of cases, in general, such practices are avoided as they add to the user's cognitive load and introduce potential maintenance problems. To handle this, CRDR checks for conflicts or inconsistencies that may arise.

We can report that after 1 hour of training, domain users were able to use the prototype system to collaboratively acquire knowledge to cover about 15% of the 5000 daily cases after a total of 7 hours of KA (Vazey & Richards, 2006b). This conservatively represents a savings of at least 4.5 hours trouble shooting per day, and thus in less than two days the effort invested was returned. The cases being solved represent 27% of a subdomain with estimated direct labour costs globally of approximately $3.3 million per annum.

It is still early days for this work and the practicality of the approach on a large scale is yet untested. From the studies to date, the approach appears to have a greater need for KE support with rule forming due to the increased number of choices and functions. The lesser emphasis on case-driven bottom-up KA in favour of rule-driven top-down KA raises questions regarding validation of the knowledge. While many mechanisms have been put in place to identify and resolve inconsistencies that might arise, scalability may be an issue as it is still unknown whether CRDR systems will get increasingly harder to maintain as the KB grows, which was the key problem in conventional KBS that SCRDR and MCRDR sought to address.

3.8 Other extensions to and applications of RDR

The early work begun by Mulholland in the area of Ion Chromatography using multiple machine learnt SCRDR and reasoning cycles has been extended further to support manual KA using MCRDR (Ramadan *et al*., 1998). Unlike SCRDR, MCRDR provides an 'n-ary tree structure with no false branches [which] more transparently supported the possible requirement to hypothesize conclusions' (Compton *et al*., 1998: 10). Following standard RDR, KA is required if any component or solution is wrong or missing. As in the early work, to allow the ion chromatographer to be configured, repeated inference and KA cycles are used. In the approach, once all errors from the first inference pass are corrected, that is rule addition is complete, the case is rerun, together with the conclusions from the first pass. If a single value is proposed for any unassigned attribute, the A–V pair is added to working memory. Inference is performed repeatedly with the revised set of values until no more single values for remaining attributes can be found. Then the user is asked to select a single value for each attribute that has received multiple possible values and the inference cycle is repeated. The approach differs from typical propose and revise methods in that rather than moving to a different part of the search space by modifying the values to an unsatisfactory solution, the domain knowledge is revised through additional manual KA to improve the local search (Compton *et al*., 1998). This work was an example of how the MCRDR classification PSM can be extended to handle a configuration or parametric design task. We consider this further in the section regarding the generality of RDR.

Drake and Beydoun (2000) have considered the incorporation of predicate logic into NRDR using situation theory to model domain instances as situations. This step takes RDR beyond

propositional logic and offers a means to handle the combinatorial explosion inherent in many problem domains as identified in the use of RDR for the SISYPHUS I room allocation problem (Richards & Compton, 1999). Again, more difficult maintenance and KA are the key issues that arise in supporting this additional expressivity and the need to provide a good interface and assistance to the user were seen as critical features in any system allowing predicates.

To tackle some of the NRDR maintenance issues, Beydoun *et al*. (2000) have modified the standard SCRDR update and maintenance algorithms so that incremental KA is more order-independent. Classical SCRDR typically produces KB with most rules attached to the false branch. Two variations of the standard RDR algorithm are provided. The first seeks to make the most of each case by allowing more than one rule to be acquired due to the same misclassification, thereby allowing for faster convergence and less dependency on the order of cases. The second algorithm allows restructuring of the tree with a single list of exceptions (only else-if branches), which produces more compact SCRDR KBs. These extensions offer potential improvements over standard SCRDR, but as the authors point out, the algorithms allow SCRDR to achieve some of the benefits offered by MCRDR but with greater effort.

Nested RDR have been the basis of work in the area of channel routing and switchbox routing in VLSI design (Bekmann & Hoffmann, 2004). An approach known as HeurEAKA has been designed, to handle complex combinatorial problems by incrementally introducing domain knowledge in order to make the search tractable. Unlike standard deterministic RDR, the NRDR in HeurEAKA are probabilistic, incorporate genetic algorithms (GA) and an action can be a reference to an NRDR leading to further actions. The goal is to support 'probably approximately correct' problem-solving rather than optimal or classification-focused problem-solving. This is achieved by first identifying parameters such as the appropriate fitness functions to constrain the GAs search space. The HeurEAKA framework (Bekmann & Hoffmann, 2005) is a knowledge management tool which includes advanced techniques such as the use of trigger functions, NRDR usage profiles and weightings, mutation look-aheads, automatic case identification and the mutation history of a genome to bias the search into more recent areas of exploration. When tested in the domain of VLSI design, HeurEAKA was able to solve accepted benchmarks competitively and more efficiently than conventional algorithms.

Typical of many ES approaches, the RDR engine has been embedded within larger systems in a range of application domains. One example is the use of an RDR KA module within a conversational agent (Mak *et al*., 2004). The component was introduced to allow the domain knowledge of the agent to be customized. The Probot agent unusually does not perform any language processing. Turn-taking is used to alternate conversation between the system and the user and is achieved via pattern matching of the user's input with scripts divided into topics. The approach is particularly interesting because it combines the use of frames, which share many features and benefits in common with objects, to structure knowledge together with the rule-based approach of RDR.

Finlayson and Compton (2004) have explored the use of RDR for soccer simulation, which can be seen as a cooperation and planning task, in a complex and interactive multiagent environment. Seeking to build on the strengths of RDR in capturing knowledge directly from a domain expert, the research provides a user interface to monitor and gradually refine strategies of soccer. The framework seeks to build RDR soccer Agents (RDRAs). The knowledge contained in an RDRA may be the conditions under which a particular action is taken, for example, the location and role of the agent and the status of the ball may indicate that the ball should be passed to a particular type of player. While the soccer expert watches the game simulation, the expert is able to stop play to enter new knowledge. A similar approach has been proposed in Richards and Szilas (2006) for the acquisition of knowledge for training simulations for risk situations such as customs control. The preliminary results for soccer have been promising showing the approach to be worth exploring further. The issues being grappled with include how to provide an appropriate set of features and behaviours that are structured at the right level of abstraction and comprehensive modelling of the domain.

A number of RDR researchers, such as Kang and Compton (1994), have espoused the benefits of RDR as a type of CBR system which addresses the key problems faced by CBR systems: how to retrieve the appropriate case/s for a particular situation and then how to adapt the case to fit. Hoffmann and Khan (2006) have taken an approach in line with the techniques used in the field of CBR, namely the use of a distance metric such as a *k*-nearest neighbour algorithm to determine the distance between two cases. Hoffmann and Khan (2006) use RDR to incrementally capture the knowledge representing a complex distance function. In the approach, they begin with a 'default' distance function, which is refined when found to be inadequate. Exception rules are added to the default rule to build up a KBS defining a set of distance functions for the set of cases in the system.

Fujiwara *et al.* (2002) also incorporate ideas from CBR, this time to assist with KA in RDR. This work is based on the premise that KA in RDR involves two key tasks: (1) identify the correct class label of a case and (2) select features in the case, which distinguish the two class labels. Task 2 is seen as the more difficult one. In this work they induce an approximate case base (Satoh & Nakagawa, 2000) using a negative case (the current case incorrectly classified) and its nearest neighbour positive case (the cornerstone case) to construct a lattice based on their differences. The expert is asked to classify the cases in the lattice. Through gradual refinement, the boundary distinguishing the positive and negative cases is defined. The work suggests and compares six different algorithms for generating cases. Experiments showed that the case generation method produces an RDR KB as good as one constructed which requires the expert to define the distinguishing conditions. The claim is by reducing all KA to class labelling (task 1), the cognitive load of the expert is reduced (Fujiwara *et al.*, 2002).

3.9 The generality of the RDR paradigm

As we can see, over its history RDR has been applied to a wide range of problem-types. RDR has most widely been used for classification tasks but its ability to be used to control a flight simulator, capture search knowledge for chess or to configure an ion chromatographer using basically the same structure indicates that RDR constitute a family of PSMs that can be used to handle multiple types of problems. In the 1990s there was a major focus by KA researchers on PSMs or generic tasks. Domain knowledge was seen to be another issue that received less attention. In contrast, the major focus of RDR has been on facilitating acquisition of domain knowledge, so that a domain expert is able to add the bulk of the knowledge without formal knowledge engineering support or skills. The key difference between RDR and other approaches seems to be that RDR research is working towards an approach which supports the addition of domain knowledge specifically to overcome the limitations of the problem solver (Compton *et al.*, 1998). This raises the question (or rather returns to one of the earliest questions for KBS) of whether PSM reuse may be better achieved by facilitating KA with a few coarse-grained PSMs rather than developing libraries of highly specific PSMs. It also leads to the further suggestion that modelling should be secondary to KBS development rather than being a necessary precursor.

The use of a simple, 'one size fits all', generic KA approach is quite unusual and contrary to much mainstream KBS research, where knowledge level modelling has been the driver for KA. This notion of generalized RDR, while never given a name, has been specifically explored in Compton and Richards (2000) and described in a provisional patent lodged in 1999. Generalized RDR seeks to handle construction type problems, and thus classification problems which can be seen to be a simple construction problem that can be solved in one inference cycle, where the solution has a finite number of components and relationships between components without the need for searching, backtracking or changing parts of the solution already developed. The motivation behind this is that while experts do propose and revise, essentially backtracking indicates an error and shortcoming of the expert, which we do not want to emulate. Through repeated inferencing cycles the solution case is added to working memory until the correct solution is established. These cycles allow the order of rule evaluation and activation to follow what would have been the expert's 'ideal' sequence of problem resolution. However, the benefit is that the

expert was not required to come up with this specific order, but their role was to perform the more manageable and concrete task of identifying the presence or absence of relevant features.

Compton *et al*. (2004) take the generalization of RDR further to consider how all KA could be reduced to application of the relations of sequence and correction. Using these relations, knowledge components, including other KBS or non-rule-based programs, can be added and linked together. KA becomes a generic process using the basic concepts of inputs, outputs, primary rules, an RDR agent and a control mechanism. The ideas have been applied to develop an image processing system where decision trees created via ML have been linked in a refinement structure (Kerr & Compton, 2003).

A body of work has considered more general issues such as simulating expertise and modelling KA (e.g. Compton *et al*., 1995; Hoffmann *et al*., 2001; Compton *et al*., 2004; Cao & Compton, 2005; Vazey, 2006). Compton (2000) reports development of a simulator for SCRDR KBS, which tests, when KBS are at their limits, using a number of parameters and measures including levels of expertise, correction of errors and types of errors. The simulation technique developed by Cao and Compton (2005) offers a way of evaluating KA which was tested with three variants of RDR so that levels of expertise could be quantified and the effects of overgeneralization and overspecialization errors modelled. Dazeley and Kang (2004) have sought to build KBS, which predict the error that occurs with incomplete knowledge and results in the brittleness problem. An augmented hybrid system known as Rated MCRDR (RM) passes as input the MCRDR conclusions and justification (i.e., the rule conditions) into a resource-allocating radial basis function (RBF) neural network that has been trained using a single-step-Δ-update-rule. Like Edward's (1996) earlier mentioned work involving exceptions, prudence and credentials, these explorations seek to provide a framework for understanding and modelling the process of KA itself which could be extended to other KA techniques beyond RDR. Most recently, the prudence work has been extended (Prayote & Compton, 2006) to detect when a case is beyond the limits of the current KB by using RDR to incrementally create homogenous partitions within the domain in conjunction with simple statistical methods. Using minimal data, the statistical methods detect anomalies within a partition to suggest the creation of a new RDR partition. The approach has been applied to the detection of network intrusion.

4 Conclusion and final remarks

The various representations, implementations and extensions described in this paper indicate the versatility of the RDR structure. The features common in general to each approach include:

- cases to provide context, assist in forming rules and for validation;
- an exception structure for knowledge representation;
- the use of some form of preprocessor of the raw data;
- incremental KA and maintenance performed primarily by the domain expert; and
- the development of a simple model in terms of A–V pairs and conclusions.

The differences tended to concern: how the knowledge was being presented and manipulated; the systems and/or other techniques being combined with RDR; or the inferencing strategy. Deviations varied according to the domain or problem type. Following best practice, most of the RDR research reported outlines future work and directions. Again, according to the domain or problem type these directions will differ. While some of the work seeks greater structure, for example NRDR, some work seeks greater flexibility in changing previous knowledge, for example CRDR. A common concern was greater expressivity. For example, Pham and Sammut (2005) would like to expand the expressiveness of the RDR rules so that a library of background knowledge can be called upon to select appropriate features together with a more expressive language for representing conditions. However, it is well accepted within knowledge representation circles that increased expressivity leads to greater complexity. As we have seen when the underlying philosophy of RDR, which mandates simple and incremental KA, is violated there is a

cost to be borne usually in the areas of cognitive and time effort, verification, validation and maintenance.

An experience report on the commercial use of RDR in one pathology laboratory conducted over a 29-month period, involving over 16 000 rules, six million cases and 323 hours of KA confirmed that RDR does offer a simple, practical and yet scalable approach (Compton *et al*., 2006). Despite the large number and diverse range of RDR-related research projects being conducted and the commercial success of RDR, the uptake of RDR technology is mostly limited to research groups and companies within Australia. It is hoped that this article will allow greater dissemination of knowledge regarding the breadth of RDR research since its conception nearly two decades ago with a view to the possibilities ahead in decades to come.

Acknowledgements

Many thanks to Paul Compton, the father of RDR, for reviewing an early draft of this paper and for the information, mostly gained from his Website, regarding commercial RDR systems. Also thanks to the anonymous reviewers for their extensive comments and guidance over a couple of revisions. Since the list of acknowledgements relevant to this paper would add even more pages, I extend my thanks to everyone named in the references, most of whom I have had the pleasure of discussing their work with them personally. I hope I have represented your research in a way that is acceptable. Thank you for your work in moving RDR forward. RDR research has been generously funded by the Australian Research Council through multiple grants over the past two decades.

References

Agnew, N. M., Ford, K. M. & Hayes, P. J. 1997. Expertise in context: personally constructed, socially situated, and reality-relevant? In *Expertise in Context: Human and Machine*, Feltovich, P. J., Ford, K. M. & Hoffman, R. R. (eds). AAAI Press/MIT Press, 219–244.

Bain, M. & Muggleton, S. H. 1991. Non-monotonic learning. In *Machine Intelligence Vol. 12 Towards an Automated Logic of Human Thought*, Hayes, J. E., Michie, D. & Tyugu, E. (eds). Clarendon Press, 105–119.

Bekmann, J. P. & Hoffmann, A. G. 2004. HeurEAKA—a new approach for adapting GAs to the problem domain. In *Proceedings of the Pacific Rim Conference on Artificial Intelligence (PRICAI)*, Springer-Verlag, 361–372.

Bekmann, J. P. & Hoffmann, A. G. 2005. Improved knowledge acquisition for high-performance heuristic search. In *Proceedings of the International Joint Conference on Artificial Intelligence, IJCAI-05*, 41–46.

Beydoun, G. & Hoffmann, A. 1997. Acquisition of search knowledge. In *Knowledge Acquisition, Modeling and Management*, Plaza, E. & Benjamins, R. (eds), 10th European Workshop, EKAW'97, Lecture Notes in Artificial Intelligence **1319**, 1–16. Springer-Verlag.

Beydoun, G. & Hoffmann, A. 2000. Incremental acquisition of search knowledge. *Journal of Human-Computer Studies* **52**, 493–530.

Beydoun, G., Kwok, R. & Hoffmann, A. 2000. Towards order independent incremental KA. In *Proceedings of the 6th Pacific Knowledge Acquisition Workshop*, The University of New South Wales, Sydney, Australia, 1–16.

Buchanan, B. 1986. Expert systems: working systems and the research literature. *Expert Systems* **3**(1), 32–51.

Cao, T. M. & Compton, P. 2005. A simulation framework for knowledge acquisition evaluation. In *Twenty-Eighth Australasian Computer Science Conference (ACSC2005)*. Newcastle, 353–360.

Catlett, J. 1992. Ripple-Down-Rules as a mediating representation in interactive induction. In *Proceedings of the Second Japanese Knowledge Acquisition for Knowledge-Based Systems Workshop*, Nov 9–13, Kobe, Japan, 155–170.

Chandrasekaran, B. 1986. Generic tasks in knowledge-based reasoning: high level building blocks for expert system design. *IEEE Expert* **1**(3), 23–30.

Chapman, D. 1989. Penguins can make cake. *AI Magazine* **10**, 45–50.

Cho, W. & Richards, D. 2004. E-Mail document categorization using BayesTH-MCRDR algorithm. In *Proceedings of the Pacific Knowledge Acquisition Workshop 2004*, Kang, B. H., Hoffmann, A., Yamaguchi, T. & Yeap, W. K. (eds). University of Tasmania Eprints Repository, Auckland, 226–240.

Cho, W. C. & Richards, D. 2006. Automatic construction of a concept hierarchy to assist Web document classification. In *Proceedings of 2nd International Conference on Information Management and Business (IMB.2006)*, February 13–16, Sydney, Australia.

Clancey, W. 1997. The conceptual nature of knowledge, situations and activity. In *Expertise in Context: Human and Machine*, Feltovich, P. J., Ford, K. M. & Hoffman, R. R. (eds). AAAI Press/MIT Press, 247–291.

Compton, P. 2000. Simulating expertise. In *Proceedings of the 6th Pacific International Knowledge Acquisition Workshop*. School of Computer Science and Engineering, UNSW, Sydney, 51–70.

Compton, P., Cao, T. & Kerr, J. 2004. Generalising incremental knowledge acquisition. In *Proceedings of the Pacific Knowledge Acquisition Workshop 2004*, Kang, B. H., Hoffmann, A., Yamaguchi, T. & Yeap, W. K. (eds). University of Tasmania Eprints Repository, Auckland, 44–53.

Compton, P. & Jansen, R. 1988. Knowledge in context: a strategy for expert system maintenance. In *Proceedings of the 2nd Australian Joint Artificial Intelligence Conference*, Lecture Notes in Computer Science **406**, 292–306. Springer.

Compton, P. J. & Jansen, R. 1990. A philosophical basis for knowledge acquisition. *Knowledge Acquisition* **2**, 241–257.

Compton, P., Peters, L., Edwards, G. & Lavers, T. G. 2006. Experience with ripple-down rules. *Knowledge-Based System Journal* **19**(5), 356–362.

Compton, P., Preston, P. & Kang, B. 1995. The use of simulated experts in evaluating knowledge acquisition. In *Proceedings of the 9th Banff KA workshop on Knowledge Acquisition for Knowledge-Based Systems*, Gaines, B. & Musen, M. (eds). Banff, Canada, 1–12.

Compton, P., Ramadan, Z., Preston, P., Le-Gia, T., Chellen, V. & Mullholland, M. 1998. A trade-off between domain knowledge and problem solving method power. In *11th Banff Knowledge Acquisition Workshop (KAW) Proceedings*, Gaines, B. & Musen, M. (eds). Banff, Canada, 1–19.

Compton, P. & Richards, D. 2000. Generalising ripple-down rules (short paper). In *Proceedings of the 12th European Workshop on Knowledge Acquisition, Modeling and Management*. Springer-Verlag, 380–386.

Compton, P., Srinivasan, A., Edwards, G., Malor, R. & Lazarus, L. 1991. Knowledge base maintenance without a knowledge engineer. In *Proceedings World Congress on Expert Systems*, **1**, 668–675. Pergmon Press, Orlando.

Dazeley, R. & Kang, B. H. 2004. Detecting the knowledge frontier: an error predicting knowledge based system. In *Proceedings of the Pacific Knowledge Acquisition Workshop 2004*, Kang, B. H., Hoffmann, A., Yamaguchi, T. & Yeap, W. K. (eds). University of Tasmania Eprints Repository, Auckland, 241–253.

Drake, B. & Beydoun, G. 2000. Predicate logic-based incremental knowledge acquisition. In *Proceedings of the Sixth Pacific International Knowledge Acquisition Workshop*, Compton, P., Hoffmann, A., Motoda, H. & Yamaguchi, T. (eds). Sydney, 71–88.

Edwards, G. 1996. *Reflective Expert Systems in Clinical Pathology*. MD Thesis, University of New South Wales.

Edwards, G. & Compton, P. 1993. PEIRS: a pathologist maintained expert system for the interpretation of chemical pathology reports. *Pathology* **25**, 27–34.

Finlayson, A. & Compton, P. 2004. Incremental knowledge acquisition using RDR for soccer simulation. In *Proceedings of the Pacific Knowledge Acquisition Workshop 2004*, Kang, B. H., Hoffmann, A., Yamaguchi, T. & Yeap, W. K. (eds). University of Tasmania Eprints Repository, Auckland, 102–116.

Fujiwara, K., Yoshida, T., Motoda, H. & Washio, T. 2002. Case generation method for constructing an RDR knowledge base. *PRICAI 2002*, 228–237.

Gaines, B. R. & Compton, P. 1992. Induction of ripple-down rules. *5th Australian Conference on Artificial Intelligence*, Hobart, Tasmania, 349–355.

Gaines, B. R. & Compton, P. 1995. Induction of ripple-down rules applied to modeling large databases. *Journal for Intelligent Information Systems* **5**(3), 211–228.

Gaines, B. R. & Shaw, M. L. G. 1993. Knowledge acquisition tools based on personal construct psychology. *Knowledge Engineering Review* **8**(1), 49–85.

Grosso, W. E., Eriksson, H., Fergerson, R. W., Gennari, H., Tu, S. W. & Musen, M. 1999. Knowledge modelling at the millenium (The design and evolution of Protégé-2000). In *Proceedings of the 12th Workshop on Knowledge Acquisition, Modeling and Management* (KAW'99), 16–21 October, Banff, Alberta.

Helmbold, D., Sloan, R. & Warmuth, M. K. 1989. Learning nested differences of intersection-closed concept classes. In *Proceedings of the Second Annual Workshop on Computational Learning Theory*, Rivest, R., Haussler, D. & Warmuth, M. K. (eds). Santa Cruz, California, United States, Morgan Kaufmann Publishers, 41–56.

Ho, V., Wobcke, W. & Compton, P. 2003. EMMA: an E-mail Management Assistant. In *IEEE/WIC International Conference on Intelligent Agent Technology*, Liu, J., Faltings, B., Zhong, N., Lu, R. & Nishida, T. (eds). IEEE, Los Alamitos, CA, 67–74.

Hoffmann, A. G. & Khan, A. S 2006. A new approach for the incremental development of retrieval functions for CBR. *Applied Artificial Intelligence* **20**(6), 507–542.

Hoffmann, A. G., Kwok, R. & Compton, P. 2001. Simulations for comparing knowledge acquisition and machine learning. In *Proceedings of the Australian Joint Conference on Artificial Intelligence 2001*. Springer-Verlag, 273–284.

Hoffmann, A. & Pham, S. B. 2003. Towards topic-based summarization for interactive document viewing. In *Proceedings of the 2nd International Conference on Knowledge Capture* (Sanibel Island, FL, USA, October 23–25, 2003). K-CAP '03, ACM, New York, NY, 28–35.

Ignizio, J. P. 1991. *Introduction to Expert Systems: The Development and Implementation of Rule-Based Expert Systems*. McGraw-Hill Inc.

Kang, B. 1996. *Validating Knowledge Acquisition: Multiple Classification Ripple Down Rules*. PhD Thesis, School of Computer Science and Engineering, University of NSW, Australia.

Kang, B. & Compton, P. 1994. A maintenance approach to case based reasoning. In *Advances in Case-Based Reasoning, Second European Workshop, EWCBR-94, Chantilly, France, November 7–10, 1994*. Lecture Notes in Computer Science **984**, 226–239. Springer.

Kang, B., Compton, P. & Preston, P. 1995. Multiple classification ripple down rules: evaluation and possibilities. In *Proceedings of the 9th Banff Knowledge Acquisition for Knowledge Based Systems Workshop*. Banff, Feb 26–March 3, **1**, 17.1–17.20.

Kang, B., Yoshida, K., Motoda, H. & Compton, P. 1997. A help desk system with intelligence interface. *Applied Artificial Intelligence* **11**, 611–631.

Kerr, J. & Compton, P. 2003. Toward generic model-based object recognition by knowledge acquisition and machine learning. In *Proceedings of the Workshop on Mixed-Initiative Intelligent Systems, IJCAI 2003*. Acapulco, Mexico.

Kim, M. & Compton, P. 2004. Evolutionary document management and retrieval for specialized domains on the web. *International Journal of Human Computer Studies* **60**(2), 201–241.

Kim, M., Compton, P. & Kang, B. 1999. Incremental development of a web-based help desk system. *The Fourth Australian Knowledge Acquisition Workshop*. University of NSW, Sydney, 157–171.

Kivinen, J., Mannila, H. & Ukkonen, E. 1993. Learning rules with local exceptions. In *Proceedings of the European Conference on Computational Learning Theory (COLT)*. `http://citeseer.ifi.unizh.ch/kivinen93learning.htm`

Lee, M. & Compton, P. 1995. From heuristic knowledge to causal. In *Explanations Proceedings of Eighth Australian Joint Conference on Artificial Intelligence AI'95*, Yao, X. (ed.). 13–17 November, Canberra, World Scientific, 83–90.

Mak, P., Byeong, H. K., Sammut, C. & Kadous, W. 2004. Knowledge acquisition module for conversational agents. In *Proceedings of the Pacific Knowledge Acquisition Workshop 2004*, Kang, B. H., Hoffmann, A., Yamaguchi, T. & Yeap, W. K. (eds). University of Tasmania Eprints Repository, Auckland, 54–62.

Martinez-Bejar, R., Benjamins, R., Compton, P., Preston, P. & Martin-Rubio, F. 1998a. A formal framework to build domain knowledge ontologies for Ripple-Down Rules-based systems. In *Eleventh Workshop on Knowledge Acquisition, Modeling and Management (KAW'98)*, Gaines, B. & Musen, M. (eds). Banff, Alberta, Canada, 18–23 April, **2**, SHARE.13.

Martinez-Bejar, R., Benjamins, R. & Martin-Rubio, F. 1997. Designing operators for constructing domain knowledge ontologies. In *Knowledge Acquisition, Modeling and Management*, Plaza, E. & Benjamins, R. (eds). Lecture Notes in Artificial Intelligence **1319**, 159–173. Springer-Verlag.

Martinez-Bejar, R., Shiraz, G. M. & Compton, P. 1998b. Using Ripple Down Rules-based systems for acquiring fuzzy domain knowledge. In *Eleventh Workshop on Knowledge Acquisition, Modeling and Management (KAW'98)*, Gaines, B. & Musen, M. (eds). Banff, Alberta, Canada, 18–23 April, **1**, KAT.2.

McCreath, E., Kay, J. & Crawford, E. 2006. IEMS—an approach that combines hand-crafted rules with learnt instance-based rules. *Australian Journal of Intelligent Information Processing Systems* **9**(1), 49–63.

Misra, A., Sowmya, A. & Compton, P. 2004. Incremental learning of control knowledge for lung boundary extraction. In *Proceedings of the Pacific Knowledge Acquisition Workshop 2004*, Kang, B. H., Hoffmann, A., Yamaguchi, T. & Yeap, W. K. (eds). University of Tasmania Eprints repository, Auckland, 211.

Mulholland, M., Preston, P., Haddad, P., Hibbert, B. & Compton, P. 1996. Teaching a computer ion chromatography from a database of published methods. *Journal of Chromatography* **739**, 15–24.

Mulholland, M., Preston, P., Sammut, C., Hibbert, B. & Compton, P. 1993. An expert system for ion chromatography developed using machine learning and knowledge in context. In *Proceedings of the 6th International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems*. Edinburgh.

Newell, A. 1982. The knowledge level. *Artificial Intelligence* **18**, 87–127.

Norman, D. A. 1986. Cognitive engineering. In *User Centered System Design: New Perspectives on Human-Computer Interaction*, Norman, D. A. & Draper, S. W. (eds). Lawrence Erlbaum Associates.

Park, S., Kim, Y. & Kang, B.-H. 2004. Personalized web document classification using MCRDR. In *Proceedings of the Pacific Knowledge Acquisition Workshop 2004*, Kang, B. H., Hoffmann, A., Yamaguchi, T & Yeap, W. K. (eds). University of Tasmania Eprints Repository, Auckland, 63–73.

Park, M., Wilson, L. S. & Jin, J. S. 2001. Automatic extraction of lung boundaries by a knowledge-based method. In *Selected Papers From the Pan-Sydney Workshop on Visualisation—Volume 2* (Sydney, Australia), Eades, P. & Weckert, J. (eds). ACM International Conference Proceeding Series, Australian Computer Society, **9**, 11–16.

Pham, S. B. & Hoffmann, A. G. 2004a. Extracting positive attributions from scientific papers'. In *Proceedings of the Discovery Science Conference 2004*. Springer-Verlag, 169–182.

Pham, S. B. & Hoffmann, A. G. 2004b. KAFTIE: a new KA framework for building sophisticated information extraction systems. In *Engineering Knowledge in the Age of the Semantic Web: 14th International Conference, EKAW'04*. Springer-Verlag.

Pham, K. C. & Sammut, C. 2005. RDRVision—Learning vision recognition with Ripple Down Rules. In *Australasian Conference on Robotics and Automation*, Sammut, C. (ed.). Australian Robotics and Automation Association, 7.

Prayote, A. & Compton, P. 2006. Detecting anomalies and intruders. In *AI 2006: Advances In Artificial Intelligence, 19th Australia Joint Conference on Artificial Intelligence*. Hobart, Australia, Springer, 1084–1088.

Ramadan, Z., Mulholland, M., Hibbert, D. B., Preston, P., Compton, P. & Haddad, P. R. 1998. Towards an expert system in ion-exclusion chromatography by means of multiple classification ripple-down rules. *Journal of Chromatography A* **804**(1), 29–35.

Richards, D. 1998. *The Reuse of Knowledge in Ripple Down Rules Knowledge Bases Systems*. PhD Thesis, University of New South Wales.

Richards, D. 2000. The reuse of knowledge: a user-centred approach. *International Journal of Human Computer Studies* **52**(3), 553–579.

Richards, D. 2003. Merging individual conceptual models of requirements. *Special Issue on Model-Based Requirements Engineering for the International Journal of Requirements Engineering* **8**, 195–205.

Richards, D. 2004. Addressing the ontology acquisition bottleneck through reverse ontological engineering. *Journal of Knowledge and Information Systems* **6**, 402–427.

Richards, D. & Compton, P. 1999. Revisiting Sisyphus I—an incremental approach to resource allocations using ripple-down rules. In *12th Banff Knowledge Acquisition for Knowledge-Based Systems Workshop*, Gaines, B., Kremer, R. & Musen, M. (eds). SRDG Publications, University of Calgary, 7-7.1–7-7.20.

Richards, D. & Menzies, T. 1998. Extending the SISYPHUS III experiment from a knowledge engineering to a requirements engineering task. In *11th Workshop on Knowledge Acquisition, Modeling and Management (KAW'98)*, Gaines, B. & Musen, M. (eds). Banff, Canada, SRDG Publications, Departments of Computer Science, University of Calgary, **1**, SIS-6.

Richards, D. & Szilas, N. 2006. Training the training system. In *Proceedings of 2006 International Conference on Intelligent User*, January 29–February 1, Sydney, Australia.

Satoh, K. & Nakagawa, R. 2000. Discovering critical cases in case-based reasoning. In *Online Proceedings of the Sixth International Symposium on Artificial Intelligence and Mathematics*, Florida.

Scheffer, T. 1996. Algebraic foundations and improved methods of induction or rippledown rules. In *Proceedings of the 2nd Pacific Rim Knowledge Acquisition Workshop*, PKAW'1996. `http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.1.7100`, accessed August 13, 2008.

Schreiber, G., Akkermans, H., Anjewierden, A., de Hoog, R., Shadbolt, N., Van De Velde, W. & Weilinga, B. 1999. *Knowledge Engineering and Management: The Common KADS Methodology*. MIT Press.

Schreiber, G., Weilinga, B. & Breuker, J. (eds). 1993. KADS: a principled approach to knowledge-based system development. In *Knowledge-Based Systems*. Academic Press.

Shaw, M. L. G. & Woodward, J. B. 1989. Mental models in the knowledge acquisition process. In *Proceedings of Fourth Knowledge Acquisition for Knowledge-Based Systems Workshop*, Banff.

Shiraz, G. & Sammut, C. 1997. Combining knowledge acquisition and machine learning to control dynamic systems. In *Proceedings of the 15th International Joint Conference in Artificial Intelligence (IJCAI'97)*. Morgan Kaufmann, 908–913.

Siromoney, A. & Siromoney, R. 1993. Variations and local exceptions in inductive logic programming. *Machine Intelligence* **14**, 213–234.

Suryanto, H. & Compton, P. 2000. Discovery of class relations in exception structured knowledge bases. In *Proceedings of the Linguistic on Conceptual Structures: Logical Linguistic, and Computational Issues*, August 14–18, Ganter, B. & Mineau, G. W. (eds). Lecture Notes in Computer Science **1867**, 113–126. Springer-Verlag.

Vazey, M. 2006. Stochastic foundations for the case-driven acquisition of classification rules. In *15th International Conference on Knowledge Engineering and Knowledge Management Managing Knowledge in a World of Networks (EKAW 2006)*, 2–6 October, Podebrady, Czech Republic.

Vazey, M. & Richards, D. 2004. Achieving rapid knowledge acquisition in a high-volume call center. In *Proceedings of the Pacific Knowledge Acquisition Workshop 2004*, Kang, B. H., Hoffmann, A., Yamaguchi, T. & Yeap, W. K. (eds). University of Tasmania Eprints Repository, Auckland, 74–86.

Vazey, M. & Richards, D. 2006a. A case-classification-conclusion IR-RDR approach to knowledge acquisition: applying a classification logic Wiki to the problem solving process. *International Journal of Knowledge Management* **2**(1), 72–88.

Vazey, M. & Richards, D. 2006b. Evaluation of the FastFIX Prototypes 5Cs CARD System. In *Proceedings of the Pacific Knowledge Acquisition Workshop (PKAW 2006)*, in conjunction with The Eighth Pacific Rim International Conference on Artificial Intelligence, August 7–11, 2006, Guilin, China, 106–117.

Vere, S. A. 1980. Multilevel counterfactuals for generalizations of relational concepts and productions. *Artificial Intelligence* **14**, 139–164.

Wada, T., Motoda, H. & Washio, T. 2001. Knowledge acquisition from both human expert and data. In *Proceedings of the 5th Pacific-Asia Conference on Knowledge Discovery and Data Mining (April 16–18, 2001)*, Cheung, D. W., Williams, G. J. & Li, Q. (eds). Lecture Notes In Computer Science, **2035**, 550–561. Springer-Verlag.

Wang, J.C., Boland, M., Graco, W. & He, H. 1996. Use of ripple-down rules for classifying medical general practitioner practice profiles repetition. In *Proceedings of Pacific Knowledge Acquisition Workshop PKAW'96*, Compton, P., Mizoguchi, R., Motoda, H. & Menzies, T. (eds). October 23–25, Coogee, Australia, 333–345.

Wille, R. 1992. Concept lattices and conceptual knowledge systems. *Computers and Mathematics Applications* **23**(6–9), 493–515.

Wrobel, S. 1988. Automatic representation adjustment in an observational discovery system. In *Proceedings of the 3rd European Working Session on Learning*, Sleeman, D. (ed.). Pitman, 253–262.

Yao, Y., Wang, F.-Y., Wang, J. & Zeng, D. 2005. Rule + Exception strategies for security information analysis. *IEEE Intelligent Systems* **20**(3), 52–57.

Yoshida, T. & Motoda, H. 2005. Performance evaluation of fusing two different knowledge sources in Ripple Down Rules method. In *Proceedings of the 2005 International Conference on Active Media Technology* (AMT 2005), May 19–21, Brisbane, Australia, 69–74.

Yoshida, T., Wada, T., Motoda, H. & Washio, T. 2004. Adaptive Ripple Down Rules method based on minimum description length principle. *Intelligent Data Analysis* **8**(3), 239–265.