On the Convergence of Incremental Knowledge Base Construction

Tri M. Cao[1], Eric Martin[1,2], and Paul Compton[1]

[1] School of Computer Science and Engineering
University of New South Wales
Sydney 2052, Australia
{tmc,emartin,compton}@cse.unsw.edu.au
[2] National ICT Australia Limited*

Abstract. Ripple Down Rules is a practical methodology to build knowledge-based systems, which has proved successful in a wide range of commercial applications. However, little work has been done on its theoretical foundations. In this paper, we formalise the key features of the method. We present the process of building a correct knowledge base as a discovery scenario involving a user, an expert, and a system. The user provides data for classification. The expert helps the system to build its knowledge base incrementally, using the output of the latter in response to the last datum provided by the user. In case the system's output is not satisfactory, the expert guides the system to improve its future performance while not affecting its ability to properly classify past data. We examine under which conditions the sequence of knowledge bases constructed by the system eventually converges to a knowledge base that faithfully represents the target classification function. Our results are in accordance with the observed behaviour of real-life systems.

1 Introduction

An expert can be defined as someone who is able to make excellent judgements in some specific domain; that is, someone who is able to draw appropriate conclusions from the data available. A central problem for building knowledge-based systems is that although an ability to draw conclusions from data generally implies some ability to indicate features relevant to the conclusion, it does not imply an ability to provide a general model of the whole domain. That is, the expert will only indicate some distinguishing features and never all the features that distinguish a particular conclusion from all other conclusions in the domain. Compton and Jansen [5] and Richards and Compton [14] argue that people cannot give a comprehensive explanation for their decision making, but at most justify why a decision is preferable to the other alternate decisions under consideration in the context.

Ripple-Down Rules (RDR) is a knowledge acquisition methodology developed to deal with this contextual nature of knowledge. It requires experts only to deal with

* National ICT Australia is funded by the Australian Government's Department of Communications, Information Technology and the Arts and the Australian Research Council through Backing Australia's Ability and the ICT Centre of Excellence Program.

E. Suzuki and S. Arikawa (Eds.): DS 2004, LNAI 3245, pp. 207–218, 2004.

specific cases and to justify why a specific conclusion applies to a case rather than some other conclusion. The expert has to identify features in the case that distinguish it from other specific cases that have the alternate conclusion. The approach is further grounded in an expert's concrete experience, by building the knowledge base while it is in use processing real cases. The expert supervises the output of the system and corrects it when it makes an inappropriate decision, by identifying the distinguishing features of the case. The expert only deals with cases and makes no attempt to generalise or structure the knowledge; the RDR system is responsible for organising the knowledge. This approach contrasts significantly with conventional knowledge acquisition where generally the expert and knowledge engineer collaborate to design a knowledge model of the domain, and it is hoped that this knowledge is effectively complete before being put into use.

RDR systems have been developed for a range of application areas and tasks. The first industrial demonstration of this approach was the PEIRS system, which provided clinical interpretations for reports of pathology testing and had almost 2000 rules built by pathologist [5, 13]. The approach has also been adapted to a range of tasks: control [16], heuristic search [2], document management using multiple classification [9], and configuration [6]. The level of evaluation in these studies varies, but overall they clearly demonstrate very simple and highly efficient knowledge acquisition. There is now significant commercial experience of RDR confirming the efficiency of the approach. Following the PEIRS example, one company, Pacific Knowledge Systems supplies tools for pathologist to build systems to provide interpretative comments for medical Chemical Pathology reports. One of their customers now processes up to 14,000 patient reports per day through their 23 RDR knowledge bases with a total of about 10,000 rules, giving very highly patient-specific comments. They have a high level of satisfaction from their general practitioner clients and from the pathologists who keep on building more rules – or rather who keep on identifying distinguishing features to provide subtle and clinically valuable comments. A pathologist generally requires less than one day's training and rule addition is a minor addition to their normal duties of checking reports; it takes at most a few minutes per rule (Pacific Knowledge Systems, personal communication[1]).

Given the success of the knowledge representation scheme and the knowledge revision procedure, it is of interest to investigate the properties of RDR to account for its success and shape its future developments. We examine the relevance of the approach in the perspective of formal learning theory (Inductive Inference). A paradigm of formal learning theory investigates under which conditions the sequence of hypotheses output by an agent, in response to longer and longer finite initial segments of a potentially infinite stream of data, converges to an accurate and finite description of a target function. A simple learning scenario together with the corresponding concepts of "convergence" and "accuracy" were first defined by Gold in his model of learning in the limit [8]. This model is well adapted to address the kind of issues we are interested in. However, further investigation in other learning frameworks, such as PAC learning or statistical query models [10], would be necessary to provide complementary insight into the issues.

[1] Dr. L. Peters http://www.pks.com.au/main.htm

In this paper, we formalise the process of building a correct knowledge base as a learning scenario where success is expressed in terms of convergence in the limit. More precisely, we examine important conditions that do or do not guarantee whether the sequence of knowledge bases built by the system eventually converges to a knowledge base that faithfully represents the target classification function.

In Section 2, we give an informal representation of Ripple Down Rules. In Section 3, we formalise the process of incremental knowledge base construction. The convergence of the process is investigated in Section 4. We conclude in Section 5.

2 Ripple Down Rules

In this section, we present a knowledge representation and reasoning scheme of RDR systems. There seems to be two reasonable approaches for translating an RDR knowledge base into a logical theory. The first approach would use prioritised logic, which would involve a second-order predicate to capture the refinement semantics. The second approach is presented in more detail since it is our preferred choice.

A binary Ripple Down Rules knowledge base is a finite binary tree with two distinct types of edges, labelled with either *except* or *if-not*, whose nodes are labelled with rules of the form *"if α then C"* or *"if α then not C"*, where α is a formula in a propositional language and C is a fixed atom outside this language. Figure 1 depicts an example of a binary RDR knowledge base. We call α the condition and C the conclusion of the rule.

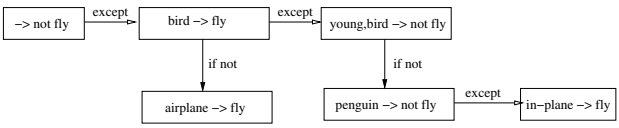

Fig. 1. An RDR knowledge base.

Note that we only consider binary RDR knowledge bases. (In full generality, knowledge bases can have n branches and conclusions can be built from different atoms. It is not difficult to reduce such knowledge bases to the kind of knowledge base we consider here; details are omitted.)

In the above example, we can identify the following elements of the knowledge base:

- attributes, namely: bird, young, airplane, penguin, in-plane,...;
- conclusions, namely: fly and not fly.

The binary RDR knowledge base is used as follows. A data case d is passed to the tree starting from the root. If the data case entails the condition of the current node, the data case is said to fire the rule and the conclusion is temporarily recorded (and overrides the previous conclusion). The data case is then passed to the next node in the *except* branch. If the case does not entail the condition of the rule, it is passed to the next node

in the *if-not* branch. The process continues until there is no suitable node to pass the case to. This algorithm defines a path from the root of the tree to a node for each case.

Procedures to translate a binary RDR tree into a set of logical rules have been proposed in [15, 11, 4]. One of the results is a representation such that, for any given data case d, there is a unique rule from the translated set which can be applied to the data case. For example, the above RDR rule tree can be translated into the following set:

$$\text{bird} \wedge \text{not young} \wedge \text{not penguin} \rightarrow \text{fly}$$
$$\text{not bird} \wedge \text{airplane} \rightarrow \text{fly}$$
$$\text{bird} \wedge \text{young} \rightarrow \text{not fly}$$
$$\text{bird} \wedge \text{not young} \wedge \text{penguin} \wedge \text{not in-plane} \rightarrow \text{not fly}$$
$$\text{bird} \wedge \text{not young} \wedge \text{penguin} \wedge \text{in-plane} \rightarrow \text{fly}$$
$$\text{not bird} \wedge \text{not airplane} \rightarrow \text{not fly}$$

One of the strengths of the RDR framework is that rule bases are easy to revise. When an expert spots a rule which gives a wrong conclusion, she only needs to create a new exception to that rule. A new condition will be required from the expert to distinguish between the current data case and the past data cases which have been correctly classified by the rule. In the rule set, this action will break the rule into two rules. The conditions of the new rules will be the condition of the old rule in conjunction with the differentiating condition (from the expert) or its negation. For example, if we feed the RDR knowledge base in Figure 1 with the data case bird $\wedge$ kiwi, the answer from the system will then be fly, which is undesired, and the rule which gives the wrong conclusion is

$$\text{bird} \wedge \text{not young} \wedge \text{not penguin} \rightarrow \text{fly}$$

Therefore, if the differentiating condition given by the expert is kiwi, then the RDR knowledge base becomes as represented in Figure 2.

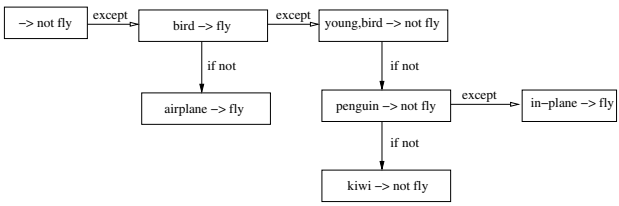

Fig. 2. The revised RDR knowledge base.

The new translated rule set is

$$\text{not bird} \wedge \text{airplane} \rightarrow \text{fly}$$
$$\text{bird} \wedge \text{young} \rightarrow \text{not fly}$$
$$\text{bird} \wedge \text{not young} \wedge \text{penguin} \wedge \text{not in-plane} \rightarrow \text{not fly}$$
$$\text{bird} \wedge \text{not young} \wedge \text{penguin} \wedge \text{in-plane} \rightarrow \text{fly}$$

bird ∧ not young ∧ not penguin ∧ not kiwi → fly
bird ∧ not young ∧ not penguin ∧ kiwi → not fly
not bird ∧ not airplane → not fly

In the next section, we will formalise this kind of translated set of rules and use the formalisation as the knowledge representation scheme of the system.

3 The Formal Discovery Framework

Our framework can be intuitively described as a scenario which involves: an agent or *user*, a *system*, an *expert*, and a *classification function*. The expert knows whether a given datum is negatively or positively classified by the classification function ϕ, though he does not know ϕ. The user presents the system with a stream of data. The aim of the system is to correctly classify all data in the stream with the help of the expert. We will formalise this scenario in Section 3.2 after we have introduced the necessary definitions.

3.1 Basic Concepts

We denote by $\mathcal{L}$ the set of propositional formulas built from a fixed countable set of propositional atoms $\{a_0, a_1, \ldots\}$. We choose an arbitrary tautology and refer to it by $\top$. Depending on the context, we will refer to members of $\mathcal{L}$ either as *formulas* or as *classification functions*.

Definition 1. *A* data case *is a finite set of natural numbers.*

A data case d can be seen as a model in which finitely many atoms, namely $\{a_i : i \in d\}$, have the value true. Hence given $\alpha \in \mathcal{L}$, we will write $d \models \alpha$ to mean that d is a model of α. We denote by Truth the binary function that maps every pair of the form (α, d), where α is a formula and d is a data case, to 1 if $d \models \alpha$, and to 0 otherwise.

Definition 2. *Let a finite set $F = \{\alpha_1, \alpha_2, \ldots, \alpha_n\}$ of formulas be given.*

F is said to be independent *iff for all data cases d and for all distinct members j, k of $\{1 \ldots n\}$, if $d \models \alpha_j$ then $d \models \neg\alpha_k$.*

F is said to be a coverage of the domain *iff for all data cases d, there exists a member j of $\{1 \ldots n\}$ such that $d \models \alpha_j$.*

F is said to be a partition of the domain *iff F is both independent and a coverage of the domain.*

We abstract an RDR knowledge base as a finite set of rules whose antecedents are formulas that represent conditions to be tested against data cases and whose conclusions are either 1 or 0 depending on whether the data case is positively or negatively classified by the system. A key condition is that the set of premises of the rules forms a partition of the domain.

Definition 3. *A* knowledge base *is a finite set of the form:*

$$\{(\alpha_1, c_1), (\alpha_2, c_2), \ldots, (\alpha_n, c_n)\}$$

where $\alpha_1, \ldots, \alpha_n$ *are formulas such that* $\{\alpha_1, \alpha_2, \ldots, \alpha_n\}$ *is a partition of the domain and* $c_1, \ldots, c_n$ *are members of* $\{0, 1\}$.

Intuitively, if a pair of the form $(\alpha, 1)$ (respect., $(\alpha, 0)$) belongs to a knowledge base K, then for every data case d that is a model of α, K positively (respect., negatively) classifies d.

Our scenario involves infinite sequences of knowledge bases. Two issues have to be considered: convergence of the sequence and, in case of convergence, correctness of the limit knowledge base. The next two definitions capture these concepts.

Definition 4. *Let a sequence* $(K_i)_{i\in\mathbb{N}}$ *of knowledge bases and a knowledge base* K *be given. We say that* $(K_i)_{i\in\mathbb{N}}$ converges to K *iff there exists* $n \in \mathbb{N}$ *such that for all* $i \geq n$, $K_i = K$.

Definition 5. *Let* $K = \{(\alpha_1, c_1), (\alpha_2, c_2), \ldots, (\alpha_k, c_k)\}$ *be a knowledge base.* K *is said to be* correct with respect to *a classification function* ϕ *iff:*

$$\models \bigvee_{(\alpha,1)\in K} \alpha \rightarrow \phi \quad \textit{and} \quad \models \bigvee_{(\alpha,0)\in K} \alpha \rightarrow \neg\phi.$$

In the previous definition, implications can be replaced by equivalences.

Proposition 1. *Let* $K = \{(\alpha_1, c_1), (\alpha_2, c_2), \ldots, (\alpha_k, c_k)\}$ *be a knowledge base.* K *is correct with respect to a classification function* ϕ *iff*

$$\models \bigvee_{(\alpha,1)\in K} \alpha \leftrightarrow \phi \quad \textit{and} \quad \models \bigvee_{(\alpha,0)\in K} \alpha \leftrightarrow \neg\phi.$$

Proof. Let d be a data case and assume that $d \models \phi$. By the definition of K, there exists a unique $(\gamma, c) \in K$ with $d \models \gamma$. It suffices to show that $c = 1$. By way of contradiction, assume that $c = 0$. Since $\models \bigvee_{(\alpha,0)\in K} \rightarrow \neg\phi$, $\gamma \in \{\alpha : (\alpha, 0) \in K\}$ and $d \models \gamma$, it follows that $d \models \neg\phi$. So $c = 1$.

The previous proposition expresses that we can get the normal form of the target classification function by combining the antecedents of the rules in the correct knowledge base. The next definition abstracts the behaviour of the system which, for a given data case d, will automatically determine the unique rule in its knowledge base that applies to d.

Definition 6. *The* system function *is the* (*unique*) *function, denoted by* S*, that maps any pair of the form* (K, d)*, where* K *is a knowledge base and* d *is a data case, to the unique member* (α, c) *of* K *such that* $d \models \alpha$*; we denote* α *by* $S_1(K, d)$ *and* c *by* $S_2(K, d)$.

Two knowledge bases may represent distinct classification functions but from the user's point of view, both knowledge bases are equivalent if they agree on that user's (finite) set of data. The next definition formalises this notion.

Definition 7. *Let a set* D *of data cases and two knowledge bases* K, K' *be given. We say that* K *and* K' *are* D-equivalent *iff for all members* d *of* D, $S_2(K, d) = S_2(K', d)$.

3.2 The Discovery Scenario

Before we formalise the scenario that has been outlined at the beginning of the previous section, we describe it in more detail, using the concepts that have been introduced.

1. The user presents the system with some datum d.
2. The system returns $S(K, d) = (\alpha, c)$ where K is the current knowledge base in the system.
3. The expert analyses (α, c). More precisely, if $c = 1$ (respect., $c = 0$) and the expert considers that d should be positively (respect., negatively) classified, then go back to step 1; otherwise, go to step 4.
4. The expert E determines a formula δ such that if (α, c) is removed from K and replaced by the more specific rules $(\alpha \wedge \delta, c)$ and $(\alpha \wedge \neg\delta, 1 - c)$, then the resulting knowledge base K' has the following properties:
   - if D is the set of data received by the system before d, then K and K' are D-equivalent;
   - If the expert positively (respect., negatively) classifies d, then $S_2(K', d) = 1$ (respect., $S_2(K', d) = 0$).

   Go back to step 1.

To fix the knowledge base, the expert essentially breaks the rule that gives wrong classification on the last datum into two rules by adding a differentiating condition. The first rule will take care of all the previous data cases that apply to the old rule. The second rule will apply to the new data case only.

The next lemma is important because it guarantees that the expert can always find a differentiating condition and thus refine the knowledge base in an incremental manner. This is one of the strengths of RDR in building knowledge-based systems. Note that the lemma is just an existence statement. It is a theoretical result that does not impose any restriction on how an expert might come up with a differentiating condition. In practice, such a condition is likely to be more complex than the one given here.

Lemma 1. *Let a knowledge base K, a classification function ϕ, and a finite set D of data cases be such that for all members d of D, $S_2(K, d) = \mathrm{Truth}(\phi, d)$. Let a data case d that does not belong to D be given. Let $(\alpha, c) = S(K, d)$. Then there exists a formula δ with the following properties. Set:*

$$K' = K \backslash \{(\alpha, c)\} \cup \{(\alpha \wedge \delta, c), (\alpha \wedge \neg\delta, 1 - c)\}.$$

Then:

- *K' is a knowledge base;*
- *K and K' are D-equivalent;*
- *$\mathrm{Truth}(\delta, d) = c$.*

Proof. Define $D_\alpha = \{d' \in D : d' \models \alpha\}$ to be the support data of the rule (α, c). We will construct a formula δ such that $d \models \alpha \wedge \delta$ and for all $d' \in D_\alpha$, $d' \models \alpha \wedge \neg\delta$. Given a data case d', define:

$$\mathrm{Diff}(d, d') = \{a \in \mathcal{L} : d \models a \wedge d' \models \neg a\}.$$

In case d' is a data case that is distinct from d then the set $\mathrm{Diff}(d, d') \cup \mathrm{Diff}(d', d)$ is not empty. For all $d' \in D_\alpha$, let

$$\delta_{d'} = \bigwedge_{a \in \mathrm{Diff}(d,d')} a \wedge \bigwedge_{a \in \mathrm{Diff}(d',d)} \neg a.$$

Let $\delta = \bigwedge_{d' \in D_\alpha} \delta_{d'}$. Then δ satisfies the claims of the lemma.

We are now ready to formalise the learning scenario outlined at the beginning of this section. For a given sequence of data cases and a given target classification function, it defines:

- a sequence of formulas that represent the expert's strategy in response to the data received so far, and the way these data are classified by the system;
- a sequence of knowledge bases that are built incrementally by the system following the expert's guidance.

Definition 8. *Let a sequence $(d_n)_{n \in \mathbb{N} \setminus \{0\}}$ of data cases and a classification function ϕ be given. An* admissible scenario for $(d_n)_{n \in \mathbb{N} \setminus \{0\}}$ and ϕ *is a pair*

$$((\delta_n)_{n \in \mathbb{N} \setminus \{0\}}, (K_n)_{n \in \mathbb{N}})$$

where $(\delta_n)_{n \in \mathbb{N} \setminus \{0\}}$ is a sequence of formulas and $(K_n)_{n \in \mathbb{N}}$ is a sequence of knowledge bases such that $K_0 = \{(\top, 1)\}$ and for all $n \in \mathbb{N}$, the following holds. Let $(\alpha, c) = S(d_{n+1}, K_n)$ and $D = \{d_1, \ldots, d_n\}$. Then:

- *if* $\mathrm{Truth}(\phi, d_{n+1}) = c$ *then* $\delta_{n+1} = \top$*;*
- $K_{n+1} = K_n \setminus \{(\alpha, c)\} \cup \{(\alpha \wedge \delta_{n+1}, c), (\alpha \wedge \neg\delta_{n+1}, 1 - c)\}$*;*
- K_n *and* K_{n+1} *are* D*-equivalent;*
- $\mathrm{Truth}(\delta_{n+1}, d_{n+1}) = c$*.*

Let a sequence $(d_n)_{n \in \mathbb{N} \setminus \{0\}}$ of data cases and a classification function ϕ be given. In Definition 8, the existence of at least one admissible scenario for $(d_n)_{n \in \mathbb{N} \setminus \{0\}}$ and ϕ is justified by Lemma 1. Intuitively, δ_{n+1} is the expert's opinion in response to d_{n+1}, α and c. In the above definition, if the expert agrees with the system on the classification of data case d_n then δ is assigned $\top$ (the tautology) and as a result, K_{n+1} is the same as K_n.

Technically, it is convenient to be able to refer to the sequence of knowledge bases, with no mention of the sequence of formulas put forward by the expert:

Definition 9. *Let a sequence $(d_n)_{n \in \mathbb{N} \setminus \{0\}}$ of data cases and a classification function ϕ be given. A* knowledge base refinement *for $(d_n)_{n \in \mathbb{N} \setminus \{0\}}$ and ϕ is a sequence $(K_n)_{n \in \mathbb{N}}$ of knowledge bases for which there exists a sequence $(\delta_n)_{n \in \mathbb{N} \setminus \{0\}}$ of formulas such that $((\delta_n)_{n \in \mathbb{N} \setminus \{0\}}, (K_n)_{n \in \mathbb{N}})$ is an admissible scenario for $(d_n)_{n \in \mathbb{N} \setminus \{0\}}$ and ϕ.*

4 Convergence Properties

To explore the scenario we described in the previous section, we need to introduce an additional notion. As in standard learning paradigms, we assume that data are noise-free and complete in the following sense.

Definition 10. *We say an enumeration of data cases is* complete *iff it contains at least one occurrence of every data case.*

Though enumerations of data might not be complete in practice, completeness is one of the main assumptions in all basic paradigms of formal learning theory. A good understanding of scenarios based on this assumption paves the way to more realistic descriptions where data can be noisy or incomplete. So we apply the scenario to the case where $(d_n)_{n\in\mathbb{N}\setminus\{0\}}$ is a complete enumeration of data cases. Since the knowledge base correctly classifies all the observed data cases, it performs no worse than the expert on those cases. Hence even if the stream of data is not complete, the sequence of generated knowledge bases is practically valuable; the completeness assumption is not necessary to guarantee the consistency of the knowledge bases with past data.

We now investigate the conditions under which the sequence of knowledge bases built by the system converges. The reason we are interested in the convergence is that the limit knowledge base correctly classifies all data cases with respect to the target classification function, as expressed in the next proposition.

Proposition 2. *Let a complete sequence $(d_n)_{n\in\mathbb{N}\setminus\{0\}}$ of data cases and a classification function ϕ be given. Let $(K_n)_{n\in\mathbb{N}}$ be a knowledge base refinement for $(d_n)_{n\in\mathbb{N}\setminus\{0\}}$ and ϕ. If $(K_n)_{n\in\mathbb{N}}$ converges to some knowledge base K then K is correct w.r.t. ϕ.*

Proof. For all $n \in \mathbb{N}$, if $K_{n+1} = K_n$ then K_n correctly classifies d_{n+1}. Moreover, for all $n \in \mathbb{N}$, K_n correctly classifies $d_0, \ldots, d_n$ by construction. The proposition follows. The next proposition shows that some real expertise is needed in order to help the system to converge to some knowledge base.

Proposition 3. *For every classification function ϕ, there exists uncountably many complete enumerations of data $(d_n)_{n\in\mathbb{N}\setminus\{0\}}$ for which some knowledge base refinement for $(d_n)_{n\in\mathbb{N}\setminus\{0\}}$ and ϕ does not converge to any knowledge base.*

Proof. We show that there exists an enumeration of data on which the knowledge base is not guaranteed to converge. Let's assume without loss of generality that the classification function ϕ is a_0. Let $\{n_i\}_{i\in\mathbb{N}}$ be an increasing sequence of numbers with $n_0 > 0$. Let

$$p_{n_i} = \mathrm{Odd}(i)\,\overbrace{0\ldots0}^{n_0-1}1\;\overbrace{0\ldots0}^{n_1-n_0-1}\;1\ldots$$

denote the data case where the 0^{th} attribute is classified as positive (resp. classified as negative) if i is odd (resp. even) and only the $n_0^{\text{th}}, \ldots, n_i^{\text{th}}$ attributes are positive. We construct a complete enumeration $(d_j)_{j\in\mathbb{N}\setminus\{0\}}$ of data cases from the sequence $(p_{n_i})_{i\in\mathbb{N}}$ as follows:

$$\begin{array}{rcl} d_1 = p_{n_0} & = & 00\ldots010\ldots \\ & & \quad A_0 \\ d_{j_1} = p_{n_1} & = & 10\ldots010\ldots10\ldots \\ & & \quad A_1 \\ & \ldots & \end{array}$$

where for all $i \in \mathbb{N}$, A_i is an enumeration of all data cases d such that for all $k \geq n_i$, $d \models \neg a_k$. Remember from Definition 8 that $K_0 = \{(\top, 1)\}$. When the user presents

d_1 to the system, the system misclassifies d_1 (positively instead of negatively). In order to correctly classify d_1, the expert can choose the formula $\neg a_{n_0}$ as the differentiating condition, in which case the system updates its knowledge base to $K_1 = \{(\neg a_{n_0}, 1), (a_{n_0}, 0)\}$. Some of the member of A_0 will be misclassified by K_1. In order to remedy these misclassifications, the expert will need to break the first rule of K_1 (namely, $(\neg a_{n_0}, 1)$) and subsequently break some of the derived rules; indeed, the expert cannot break the rule $(a_{n_0}, 0)$ since none of the members of A_0 is a model of a_{n_0}. When d_{j_1} is presented to the system by the user, the faulty rule $(a_{n_0}, 0)$ has to be broken into two rules so that the system correctly classifies d_{j_1}. Suppose that the expert chooses the formula $\neg a_{n_1}$. The two new rules are $(a_{n_0} \wedge \neg a_{n_1}, 0)$ and $(a_{n_0} \wedge a_{n_1}, 1)$. Then the pattern described before repeats itself: the rule $(a_{n_0} \wedge a_{n_1}, 1)$ is left untouched until d_{j_2} (which is p_{n_2}) shows up, etc. For all $i \in \mathbb{N} \setminus \{0\}$, the knowledge base will be changed in view of d_{j_i} and the sequence of knowledge bases does not converge. Since there are uncountably many increasing sequences $\{n_i\}_{i \in \mathbb{N}}$, it follows that there are uncountably many data enumerations that can fail the expert.

The proof of Proposition 3 shows that experts who can correctly classify a single data case, but who are unable to select the relevant attributes, will induce the system to produce infinitely many knowledge bases. At any given time, the number of attributes occurring in the data cases observed so far is finite. When the expert has to revise the current knowledge base, he could take into consideration only the finite set of observed attributes, and converge to a correct knowledge base. The problem with the expert defined in the proof of the previous proposition is not that the number of attributes is potentially infinite (with more and more of them being observed as more and more data come in), but the fact that he chooses the wrong attribute from the finite set of available ones. It should also be noted that the behaviour of experts described in the proof is in accordance with practical RDR based systems.

Theorem 1. *Let a complete enumeration of data $(d_n)_{n \in \mathbb{N} \setminus \{0\}}$, a classification function ϕ, a sequence $(\delta_n)_{n \in \mathbb{N} \setminus \{0\}}$ of formulas, and a sequence $(K_n)_{n \in \mathbb{N}}$ of knowledge bases be such that:*

- *$((\delta_n)_{n \in \mathbb{N} \setminus \{0\}}, (K_n)_{n \in \mathbb{N}})$ is an admissible scenario for $(d_n)_{n \in \mathbb{N} \setminus \{0\}}$ and ϕ;*
- *there exists a finite set A of propositional atoms such that for all $n \in \mathbb{N}$, δ_n is built from members of A only.*

Then $(K_n)_{n \in \mathbb{N}}$ converges to some knowledge base.

Proof. Let $n \in \mathbb{N}$ be given. By Definition 3, for all distinct members (α_1, c_1), (α_2, c_2) of K_n, α_1 and α_2 are not logically equivalent. Moreover, it follows from Definition 8 that for all $(\alpha, c) \in K_n$, α is built from members of A only. Hence there exists $N \in \mathbb{N}$ such that for all $n \in \mathbb{N}$, the number of rules in K_n is bounded by N. Also from Definition 8, the number of rules in K_{n+1} is greater than or equal to the number of rules in K_n, for all $n \in \mathbb{N}$. This immediately implies the sequence $(K_n)_{n \in \mathbb{N}}$ converges.

The theorem states that if the expert bases his judgement of the classification on a finite set of attributes, then the knowledge base will eventually converge to some limiting knowledge base. Proposition 2 guarantees that the limit knowledge base is correct with respect to the target classification function.

One of the possible implications of the previous theorem can be applied to the domain of web page classification. If the keywords are taken as attributes, then it is possible to assume that the number of distinct attributes in the domain is large (infinite). On the other hand, the number of attributes that occur in a given web page is small (finite). Hence a web page can be identified as a data case in the sense of our framework. Therefore, if we assume that the classification function can be defined in terms of a finite number of attributes (keywords), then a good expert, namely, an expert who considers new attributes only when she is forced to, will eventually construct a good classifier.

There are alternative approaches for building knowledge bases from a stream of cases, some of these approaches might be conceptually simpler, and enjoy a straightforward convergence result. It is beyond the scope of this paper to compare RDR with competing approaches. RDR has turned out to be very valuable in practice, in particular because it not only classifies cases but also encodes knowledge that supports classification. Hence RDR is a useful tool for explanation and teaching. For this reason, it is important that the convergence property of the RDR approach is guaranteed.

Feature selection is the problem of removing irrelevant or redundant features from a given set [3] and is often formalised as a search problem. It is considered as a hard problem (even if the number of attributes is small) that has attracted lots of attention recently [7, 12]. Our work is related to this problem in the sense that it analyses how expertise can be used in the selection process. RDR has proved useful in acquiring search knowledge [2] and could provide some insight to the feature selection problem.

5 Conclusion and Further Work

Ripple Down Rules has proved successful in practice as a methodology to build knowledge based systems. However, there has been so far no systematic analysis of the fundamental features that make it successful. In this paper, we have presented a crude but accurate formalisation of RDR methodology. We have shown that concepts from learning theory can be fruitfully applied and capture a desirable property of practical systems, namely that they eventually stabilise to an accurate representation of a target classification function. In practice, the number of potential attributes is very large and neither the expert nor the system knows the small number of those that are relevant. The number of attributes that appear in a data case as well as the number of attributes on the basis of which the target classification function can be expressed is small. We have examined the consequences of this asymmetry and shown that it might prevent the system from converging to a correct knowledge base, but that experts can apply some strategy to ensure convergence. We also showed that convergence guarantees correctness.

Our aim is to come up with formalisations of general classes of strategies, and evaluate their chance of success. This paper represents a first step in this direction, and illustrates the potential of the approach. It paves the way to more fine-grained models where the behaviour of the agents involved (user, expert, system) can be described more realistically. In particular, further models will incorporate possibly noisy or incomplete data, and consider less stringent classification criteria. We will also consider under which conditions the convergence of the sequence of knowledge bases is subject to an ordinal mind change bound [1]. Finally, alternative models should be investigated to shed light on other issues, e.g., speed of convergence.

References

1. A. Ambainis, R. Freivalds, and C. Smith. Inductive inference with procrastination: back to definitions. *Fundam. Inf.*, 40(1):1–16, 1999.
2. G. Beydoun and A. Hoffmann. Incremental acquisition of search knowledge. *Journal of Human-Computer Studies*, 52:493–530, 2000.
3. A. Blum and P. Langley. Selection of relevant features and examples in machine learning. *Artificial Intelligence*, 97(1-2):245–271, 1997.
4. R. Colomb. Representation of propositional expert systems as partial functions. *Artificial Intelligence*, 109(1-2):187–209, 1999.
5. P. Compton and G. Edwards. A philosophical basis for knowledge acquisition. *Knowledge Acquisition*, 2:241–257, 1990.
6. P. Compton, Z. Ramadan, P. Preston, T. Le-Gia, V. Chellen, and M. Mullholland. A trade-off between domain knowledge and problem solving method power. In B. Gaines and M. Musen, editors, *11th Banff KAW Proceeding*, pages 1–19, 1998.
7. M. Dash and H. Liu. Consistency-based search in feature selection. *Artificial Intelligence*, 151(1-2):155–176, 2003.
8. M. E. Gold. Language identification in the limit. *Information and Control*, 10:447–474, 1967.
9. B. Kang, K. Yoshida, H. Motoda, and P. Compton. A help desk system with intelligence interface. *Applied Artificial Intelligence*, 11:611–631, 1997.
10. M. J. Kearns. Efficient noise-tolerant learning from statistical queries. In *Proceedings of the 25th ACM Symposium on the Theory of Computing*, pages 392–401. ACM Press, 1993.
11. R. Kwok. Translation of ripple down rules into logic formalisms. In Rose Dieng and Olivier Corby, editors, *EKAW 2000*, volume 1937 of *Lecture Notes in Computer Science*. Springer, 2000.
12. H. Motoda and H. Liu. Data reduction: feature aggregation. In *Handbook of data mining and knowledge discovery*, pages 214–218. Oxford University Press, Inc., 2002.
13. P. Preston, G. Edwards, and P. Compton. A 2000 rule expert system without a knowledge engineer. In B. Gaines and M. Musen, editors, *8th Banff KAW Proceeding*, 1994.
14. D. Richards and P. Compton. Taking up the situated cognition challenge with ripple down rules. *Journal of Human-Computer Studies*, 49:895–926, 1998.
15. T. Scheffer. Algebraic foundation and improved methods of induction of ripple down rules. In *Pacific Rim Workshop on Knowledge Acquisition Proceeding*, 1996.
16. G. Shiraz and C. Sammut. Combining knowledge acquisition and machine learning to control dynamic systems. In *Proceedings of the 15th International Joint Conference in Artificial Intelligence (IJCAI'97)*, pages 908–913. Morgan Kaufmann, 1997.