Complex Intell. Syst. (2017) 3:83–92
DOI 10.1007/s40747-017-0034-z

ORIGINAL ARTICLE

Constructing enhanced default theories incrementally

Ghassan Beydoun[1] · **Achim Hoffmann**[2] · **Asif Gill**[1]

Received: 8 September 2016 / Accepted: 25 January 2017 / Published online: 8 February 2017

Abstract The main difference between various formalisms of non-monotonic reasoning is the representation of non-monotonic rules. In default logic, they are represented by special expressions called defaults. In default logic, commonsense knowledge about the world is represented as a set of named defaults. The use of defaults is popular because they reduce the complexity of the representation, and they are sufficient for knowledge representation in many naturally occurring contexts. This paper offers an incremental process to acquire defaults from human experts directly and at the same time it provides added semantics to defaults by adding priorities to defaults and creating additional relations between them. The paper uses an existing incremental framework, NRDR, to generate these defaults. This framework is chosen as it not only enables incremental context driven formulation of defaults, but also allows experts to introduce their own domain terms. In choosing this framework, the paper broadens its utility.

Keywords Knowledge-based system · Ripple Down Rules · Default thoeries

✉ Ghassan Beydoun
ghassan.beydoun@uts.edu.au

Achim Hoffmann
achim@cse.unsw.edu.au

Asif Gill
asif.gill@uts.edu.au

[1] Faculty of Engineering and Information Technology, University of Technology Sydney, Ultimo, Australia

[2] School of Computer Science and Engineering, University of New South Wales, Sydney, Australia

Introduction

The development of knowledge-based information systems is typically iterative. It spirals towards the final system with commonly observed development phases of: requirement definition, analysis, design and implementation [2,28]. The process often gets bogged down during the design stage as concrete models are not readily available and modeling from scratch may be too expensive. This problem is stultified using incremental approaches based on interactions between an expert and a data stream input [9]. The performance of an evolving prototype is improved with iterative development. This merges the design, implementation and maintenance phases. In this paper, we expand the scope of incremental approach to enable the generation of default theories that proved a powerful representation that enables non-monotonic reasoning [17,18,20,27]. In default logic, commonsense knowledge about the world is represented as a default theory (D, W) where D is a set of named defaults, and W is a set of axioms of the theory. D and W are expressed as first order sentences. D provides extension for the theory not derivable from W. *Defaults* can be used to reduce the complexity of the knowledge representation. They have proved popular and sufficient for knowledge representation in many naturally occurring contexts [18,20].

To construct default theories incrementally, we assume availability of streams of data as a source of knowledge to assist in development of knowledge base. This is not an overly strong assumption and it is commonly made in various applications e.g. [1,3]. Furthermore, using this stream incrementally, we aim to develop default theories that correspond to taxonomic structures. As a basis towards this, we use the knowledge base development framework that generates Nested Ripple Down Rules knowledge bases [6,21]. This knowledge base is then systematically converted into

a set of default theories. The choice of this framework is made, not only to accommodate the incremental processing, but also to enable experts to introduce their own terms when they conceptualise their domain knowledge. Indeed, human expertise is believed to be holistic in nature, in that the meaning of their domain concepts cannot be absolutely taken in separation from the rest of their domain knowledge. These explanations constitute the NRDR knowledge base which is then systematically converted, with the expert introduced terms, to a collection of inter-related default theories. The systematic conversion is developed in this paper and is a key contribution. The rest of the paper is organised as follows: Sect. "Related work and background" describes related work. Section "RDR and NRDR incremental processes" highlights the technical details of NRDR that will provide the basis of the incremental construction of default theories. Section "Systematic conversion of NRDR to default logic" provides the systematic framework to convert NRDR to default theories. This consists of both, the static representation and the dynamic knowledge acquisition process. Section "Discussion and conclusion" concludes the paper.

Related work and background

The appeal of incremental knowledge bases development is that it simplifies the knowledge engineering (i.e., in the analysis phase). Moreover, knowledge maintenance is also a simplified process. This combines an evolving prototypical performance, with iterative development. This inspires the incremental construction of default theories in this paper. It is essentially a form of case-based reasoning where past classification scenarios are used as basis for generating new solutions. A particularly popular approach of interest in this paper is Ripple Down Rules (RDR) [16]. Knowledge maintenance in RDR is simplified with an interface that guides a domain expert in developing the knowledge bases and incrementally refining them as required. Much empirical research has been done with RDR involving development of medium to large size knowledge bases. It is generally popular in medical knowledge-based information systems [13]. It also proved successful in building many useful applications over the past two decades, e.g. call management [32], mechanical engineering [7], medical systems [15,22], Chess [5].

RDR theoretical research has been more limited. Earlier such research was focused on automatic construction of RDR knowledge bases, e.g. [23,29]. Later and more relevant research to our work here, is concerned with analyzing the knowledge acquisition process itself or formalizing the resultant knowledge in order to assess its representational capacity, e.g. [14,24]. Our work extends this second category of research to map not only the RDR knowledge base to default logic rules, but also the incremental process of generating RDR to one of generating default logic statements.

The main difference between various formalisms of non-monotonic reasoning is the representation of non-monotonic rules. In default logic, they are represented by special expressions called defaults. A default $d$looks as follows: $d = (\alpha : \beta)/\gamma$ where α is the prerequisite for the default d to be considered. γ is the consequent which is believed if believing the justification β is consistent. Normal defaults with $\beta = \gamma$ are popular because they reduce the complexity of the representation, and they are sufficient for knowledge representation in many naturally occurring contexts. To develop defaults incremental, we assume the availability of streams of data as a source of knowledge to assist in development of knowledge base. However, this is not an overly strong assumption, various applications make such assumptions, e.g. [3]. To control the complexity of reasoning with defaults, sequencing the reasoning process by placing priorities on defaults has been proposed in [12]. In using an RDR-based approach to incrementally construct a set of interacting defaults, we follow a similar approach. We resort to prioritising the defaults to stop them from interacting beyond the semantics of an RDR tree.

Another key contribution of this paper is that it provides incremental construction of defaults to accommodate the expert's natural tendencies in introducing intermediate terms. Left to their own nature, human experts introduce intermediate concepts when articulating their knowledge. These terms are context driven and enable experts to express themselves as they explain (justify) their expertise. Experts articulation of such terms may depend on previous articulation of other terms, which may not yet be made explicit or completely defined. Hence, interactions between intermediate concepts makes the knowledge base evolution more complex. It can create inconsistencies. This has been a focus of many modelling tools and much knowledge representation research [19]. The knowledge representation framework Nested Ripple Down Rules (NRDR) [6,21] allows an expert to give his/her explanations using his/her own terms and provides a process to control the inconsistencies as the knowledge base evolves. The terms are operational while they are still incompletely defined. The NRDR framework relies on representing each term as a separate RDR knowledge base to allow experts to deal with exceptions, and to refine definitions of their terms. In this sense, NRDR allows the incremental construction of multiple hierarchical ontologies. These ontologies simplify the relationships between the concepts, but however they remain quite an effective tool for knowledge reuse in software development. They have been advocated in information systems development generally [10,25,26] and complex systems in particular [11,30,33].

In the next section, we detail both of the frameworks, RDR and NRDR. To construct default theories with expert inter-

mediate terms, we create a process that mirrors the NRDR process. Hence, NRDR incremental process is described first.

RDR and NRDR incremental processes

Developing an RDR knowledge base relies on direct interactions with a domain expert. The resultant knowledge base is a collection of interconnected simple rules organised in a binary tree structure where every rule can have two branches to two other rules: A false and a true branch (an exception branch) (Fig. 1). When a rule applies a true branch is taken, otherwise a false branch is taken. The root node condition is always satisfied and, therefore, has only a true-branch. It has the form "If true then default conclusion". The classification begins at the root. If a 'true-branch' leads to a terminal node t, and the condition of t is not fulfilled then the conclusion of the rule in the parent node of t is taken. In other words, if the conclusion of an exception rule ('true-branch' child rule) is satisfied, it overrides the conclusion of its parent rule. If a 'false-branch' leads to a terminal node t, and the condition of t is not fulfilled, then the conclusion of the last rule satisfied 'rippling down' to t is returned by the knowledge base. The described conditional branching is repeated until a leaf node is reached. The knowledge base is guaranteed to return a conclusion as at least the default rule is satisfied 'rippling down' to t.

RDR incremental construction is based on the idea that when a knowledge-based system makes an incorrect conclusion, a new rule r that is added to correct that conclusion should only be used in the same context in which the mistake was made [16]. This context is represented by the sequence of rules that were evaluated leading to a wrong conclusion which caused the addition of r. Rules are actually never corrected or changed because corrections are contained in rules added on to the end. Rules are attached to sequences of rules describing the context of their application. A new rule is always added as a leaf node. An added rule r satisfies the case for which the original sequence failed, and it excludes

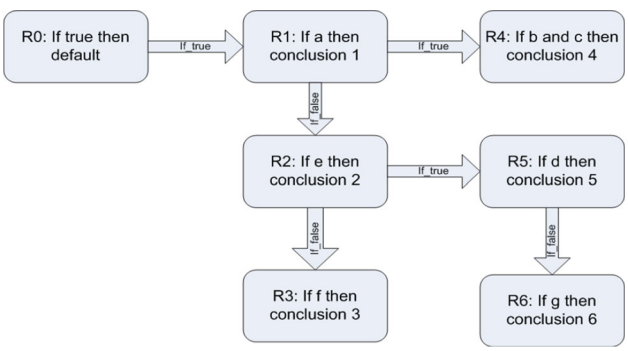

Fig. 1 A classification RDR tree. A case to be classified starts at the root default node and ripple down to a leaf node. The conclusion returned is the conclusion of the last satisfied rule to the leaf node

all cases covered by its predecessor rule. Corrections entered by the expert are always guaranteed to be valid, because of the way conditions of new rules are chosen. RDR modifications are simple enough for the expert to directly do it.

From RDR to NRDR In explaining their knowledge, experts tend to introduce new terms. The RDR framework however does not accommodate such intermediate explanations. When asked to explain such intermediate concepts, experts may oversee a definition of the concept in some contexts outside the current situation which they are explaining. They fail to provide a complete explanation that always covers their use, instead they provide an operational definition sufficient for the purpose of explaining the context on hand. The NRDR framework extends the RDR framework to enable experts to introduce intermediate terms. Every new term is represented as a new concept that is defined as a separate simple RDR tree to discriminate input objects into two sets of mutually exclusive classes: Positive objects belonging to the class indicated by the expert concept, and negative objects falling outside it. Expert introduced concepts (or classes) can in turn be used as higher order attributes (conditions) by the expert to define other concepts. The elementary level is the level of explanatory domain primitives. The entry point in NRDR is typically the highest level RDR tree.

As the expert developing an NRDR knowledge base invents new concepts, s/he makes implicit assumptions about any concepts that s/he introduces, in that s/he has a certain disposition about what these concepts mean and when they do actually apply. Such assumptions are extended in terms of new rules to the RDR trees representing these concepts. Alternatively, old assumptions are modified by adding exceptions to rules in the old representation. The expert expects his/her assumptions to hold to future unseen cases as much as they do for the already seen case. This expectation translates into a predictive capability of the rules that s/he enters. Such expectation of concepts to hold to unseen cases—simply because they have features that apply to seen cases—implies a subtle inductive hypothesis used by the expert. This raises the following question: while concepts are introduced in a particular context how are they used outside the context of their introduction, i.e., how do we project these new predicates (NRDR concepts) beyond their initial context of use. This is answered in two parts: Firstly the projection itself of concepts is assumed incomplete and open for modifications to capture any new contexts. Secondly, an NRDR knowledge base is holistically maintained, i.e., other concept definitions are checked and may be modified when a single concept is being updated.

Similar to the RDR framework, the incremental construction of an NRDR knowledge base driven by disagreements with a domain expert. However, the hierarchical structure of NRDRs makes it harder to keep the entire knowledge

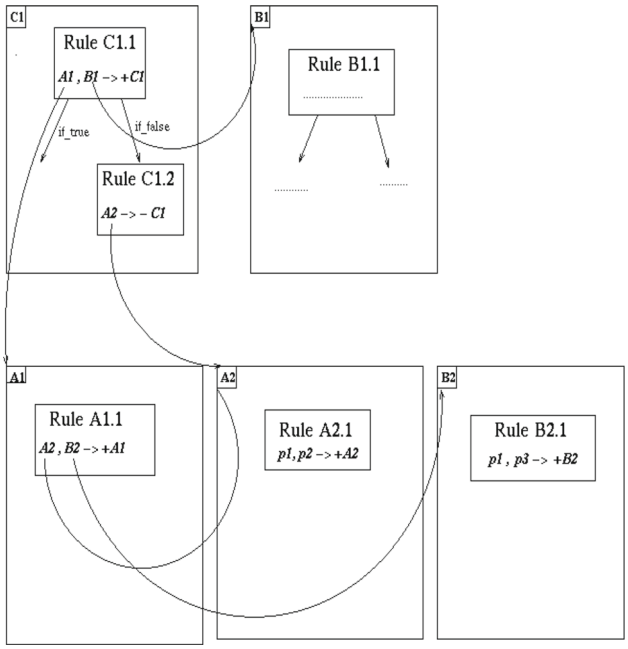

Fig. 2 A simple example of Nested Ripple Down Rules. In the above, an update in concept *A2* can cause changes in the meaning of rules *C1.1*, *C1.2*, and *A1.1* of the knowledge base

base consistent when a single concept definition needs to be altered (Fig. 2). During knowledge acquisition, given a case x that requires an NRDR knowledge base to be modified, the modification can occur in a number of places. For example—referring to Fig. 2 say case x satisfies conditions $A1$ and $B1$ in rule $C1.1$ but the expert thinks that case x is not $C1$. Hence, the knowledge base needs to be modified to reflect this. A rule can be added as an exception for the RDR tree describing $C1$, or alternatively, the meaning of attribute $A1$ can be changed by updating the definition $A1$, or $A2$ in rule $A1.1$; and so forth. The number of possibilities depends on the depth of the concept hierarchy in the knowledge base. A more serious maintenance issue is dealing with inconsistencies due to localised updates in the hierarchical knowledge base. For instance, if the expert updates the meaning of $A1$ by changing the meaning of attribute $A2$ in rule $A1.1$, s/he may inadvertently cause a change in the meaning of rule $C1.2$ that contains $A2$. Generally, when a condition X is defined in terms of lower order RDR, and X is repeatedly used in different rules, the update of X—by adding an extra rule or an exception to an existing rule—has an effect everywhere it is used. This may cause inadvertent inconsistencies with respect to past classifications. In simple RDRs, a corner stone case is associated with every rule. The rule is the justification for the classification of this case [16]. In NRDR, every rule has a set of corner stone cases. This set contains all cases that a rule classified correctly under verification of an expert. Their classifications must always hold as the knowledge base evolves. Cases can travel between sets of corner stone cases because of interactions within an NRDR knowledge base. Checking for inconsistencies when a concept C is modified requires access to all cases previously classified by C. Following every knowledge base update, these cases are classified again. A case x is inconsistent if the new classification differs from the old classification. During the discovery of x, because of the nested structure of the RDR's some lower order concept descriptions are found about x. To repair the inconsistency of the knowledge base with respect to the case, some of those concepts describing it, may also need to be updated. This may in turn cause more inconsistencies to occur. Hence, the process of checking inconsistencies is also recursive. In [9], we presented why the number of inconsistencies is too small to have a major impact to the cost of the total knowledge acquisition process.

Systematic conversion of NRDR to default logic

We first discuss the strong relationship between RDR and default theories. We also characterised default theories that can be mapped to RDR trees. We then discuss the relationship between default theories and NRDR. We discuss how the policies presented in the previous section translate to constraints on default theories that can be mapped to NRDR. We will show how default theories, that can be mapped to NRDR, are less constrained than theories which can be mapped to RDR. That is, the set of default theories that can be mapped to NRDR is larger than the set of those that can be mapped to RDR.

RDR formalism and default logic

A widely used example of a normal default is: $\frac{Bird(x):Can\_Fly(x)}{Can\_Fly(x)}$ This is interpreted as "For every x, if x is a bird and it is consistent to believe that x can fly, then it is believed that x can fly". So, if all we know about *Tweety* is that it is a bird then we are permitted to believe that it flies. However, if we learn that *Tweety* is a penguin, and we know penguins don't fly, it is inconsistent to believe that *Tweety* flies, and the application of the default is blocked. Ripple Down Rules do not map straightforwardly to a set of defaults. Normal defaults have conflicts when their prerequisites are not mutually exclusive. Their consequents can be contradictory. Exception rules are never mutually exclusive and have always—by definition—contradictory conclusions (consequents). To overcome this, we attach priorities to defaults representing Ripple Down Rules trees.

Ripple Down Rule methodology overlaps maintenance and use of the knowledge base. A Ripple Down Rule knowledge base is useable, and gets used while incomplete. A Ripple Down Rule knowledge base k grows non-monotonically. That is some of the premises derivable from the knowledge

base may get overridden by future rules: After an expert enters a new rule r we denote the knowledge base by k'. The addition of r is one of possible two scenarios:

- r is added as an exception rule. Hence, some conclusions of its parent are no longer possible.
- r is added as a new rule—i.e. r is attached to the outer false link. Hence, some conclusions of the default rule no longer apply.

In both cases, conclusions which are no longer possible, apply to cases in the domain of r. So, adding a new r retracts some premises in k. That is premises of $k' \not\subset$ premises of k *(i.e. growth of k is non-monotonic).*

Every Ripple Down Rule knowledge base has a default rule, which has "True" as a condition. Being the root node, the default conclusion of the default rule is taken when the knowledge base fails to give a conclusion. Therefore, the reasoning in an RDR knowledge base shows default reasoning in two respects:

1. The default rule is taken when all rules on the outer false link chain fail.
2. When a rule fires, its conclusion is taken only if it has no exceptions. Otherwise its conclusion is taken by default if none of its exceptions fire.

The second aspect of the default behaviour in an RDR knowledge base is equivalent to saying that exception rules have a higher priority than their parents. When they fire, their conclusions supersede that of their parents. This is enforced by the tree-like structure of an RDR knowledge base. When a case is being classified by an RDR knowledge base, the case filters down (ripples down) to a leaf node l, if the condition of l is satisfied the conclusion of l is taken; otherwise, the conclusion of the last satisfied rule s on the path to l. So, the structure of the RDR tree gives l a higher priority than s. A rule of depth n has a lower priority than a rule of depth $n+1$. When two rules have the same depth, the rule with the lower rank has the higher priority (i.e. the rule higher up in the false link chain has a higher priority). Further, the default rule (i.e., the root node of an RDR tree) has the lowest priority. Note, priorities of rules are implicit within the structure of a Ripple Down Rule tree. They are captured during the knowledge acquisition process.

In what follows, we sketch an algorithm which maps an RDR knowledge base to a *default theory.* The mapping preserves the default behaviour of RDR rules.

Mapping RDR to a default theory

In a Ripple Down Rule tree T, a rule r can be rewritten as $l \rightarrow u$. If r is the lowest priority rule in T (i.e. the default rule), then it can be rewritten as the following default d:

$$d = \frac{l(x) : u(x)}{u(x)}$$

where x is a case being classified. Rules in RDR are applied in context. To convert a rule r at depth n to a default, we must consider the path from the root node to r. The default d corresponding to r would then become:

$$d = \frac{\alpha : \beta}{\beta}$$

where α is a conjunct of all the conditions of rules which fired on the way to r (the true links) and the conjunct of every condition negation of every rule which did not fire (the false links). β is the conclusion of r. The priority of this default is n, where n is the depth of r.

Every rule in the RDR tree is converted into a default according to the above. The default rule will have the lowest priority (its depth is 0). The set of axioms with which the defaults must remain consistent is given by the database of all past seen cases. In implementations of RDR [15], this was only the set of corner stone cases. RDR trees as used within NRDR are maintained consistent with respect to all past seen cases (see "Discussion and conclusion" in the previous section).

For every rule r in an RDR tree, the corresponding default is constructed by considering the path from the root node to r. Its priority is given by the depth of the added rule. The added default is unique because the path to every leaf node in an RDR tree is unique. Moreover, when the priority of a number of defaults is the same, these defaults will have mutually exclusive prerequisites. This is consistent with the behaviour of an RDR tree, where the conclusion of a single rule is taken.

We outlined how an RDR tree can be mapped to a set of prioritised defaults which apply in mutual exclusion. That is for a given instance, exactly one default applies in the reasoning process. The mutual exclusion of defaults during reasoning is guaranteed by both: the way prerequisites of defaults are derived, and their priorities.

In the next section, we characterise default theories that can be converted to RDR trees. We also outline how those default theories are mapped to RDR trees.

Mapping a default theory to an RDR tree

In an RDR tree, paths in the tree apply in mutual exclusion. In the previous section, we mapped every path to a default. Hence, to apply the corresponding reverse process to a default theory (D, W), whereby every default corresponds to a path in the corresponding RDR tree, defaults in D must apply in separation. Further, because we defined rules in an RDR

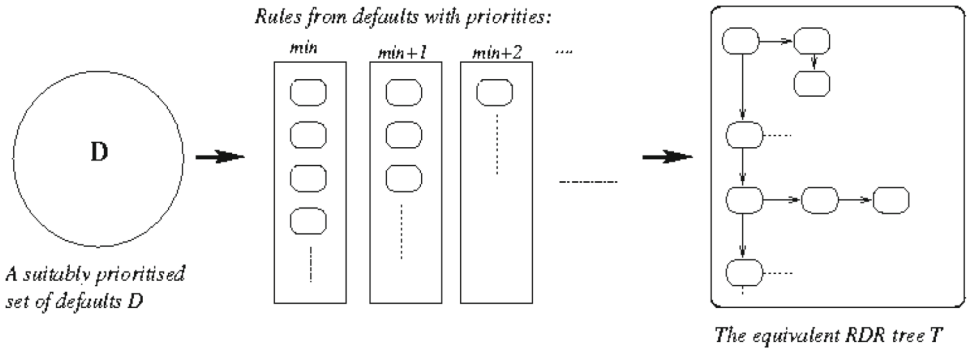

Fig. 3 Conversion steps from a prioritised set of defaults D to an RDR tree T. *Step 1* Rules from equal priorities defaults are grouped together, rule conditions in a group are mutual exclusive due to the third constraint on the source default theory. *Step 2* Subsumption relations between rules from these groups are determined to form the RDR tree T

tree to have propositional value conclusions, we also restrict the default theories that can be converted to RDR trees to have propositional value consequences. This translates to the following set of constraints which must apply to the default theory (D, W) for it to be convertible to an RDR tree:

1. Every default must be a normal default (see before for definition of normal defaults).
2. For every default $d = \frac{\alpha:\beta}{\beta}$, $\beta \in V$ where V is the set of class values (conclusions) which are propositional values.
3. Every default $d \in D$ must have a priority n. During inference, defaults with higher priorities are chosen before defaults with lower priorities, which are chosen only if none of the higher priority defaults are applicable.
4. Any two defaults in D with equal priorities must have mutually exclusive prerequisites. That is, given two defaults: $d_1 = \frac{\alpha_1:\beta_1}{\beta_1} \in D$ and $d_2 = \frac{\alpha_2:\beta_2}{\beta_2} \in D$ with equal priorities, then we have mutual exclusion between α_1 and α_2. That is, $\alpha_1 \wedge \alpha_2 = \text{False}$.
5. For any two defaults: $d_1 = \frac{\alpha_1:\beta_1}{\beta_1}$ and $d_2 = \frac{\alpha_2:\beta_2}{\beta_2}$ we have: β_2 is not required to compute α_1. That is, if $W|-\alpha_2$ then $W\backslash\{\beta_1\}|-\alpha_2$.
6. Exactly one default, d_{default}, has the lowest priority. d_{default} applies only when all other defaults in D do not apply, and it has an empty prerequisite. That is:

$$d_{\text{default}} = \frac{:\beta_{\text{default}}}{\beta_{\text{default}}}$$

For a given default theory (D, W) which satisfies the above six constraints, we now outline guidelines to convert (D, W) to an RDR tree T. These are:

1. The default d with the lowest priority min is converted into the following default rule which forms the root node of T:

   If True then $\boldsymbol{\beta}_{\text{default}}$

2. Defaults with priority min + 1 form a first false link chain on the first exception level in T. Order of rules in this chain is irrelevant because defaults have mutually exclusive prerequisites. Defaults with same priorities >min+1 are also converted to rules in T, and they form separate false link chains of rules within subsequent exception levels of T (see step 1 in Fig. 4.3).
3. To get the RDR tree T equivalent to D, subsumption relations between rules from defaults of different priorities are determined (see step 2 in Fig. 3). Rules converted from defaults with priorities min + n are represented as exceptions for equivalent rules of defaults with priorities min + n − 1. The resultant tree T reflects relationships between defaults with differing priorities (Fig. 3). In determining these subsumption relationships, the following must be observed: For two defaults $d_1 = \frac{\alpha_1:\beta_1}{\beta_1}$ and $d_2 = \frac{\alpha_2:\beta_2}{\beta_2}$, where priority$(d_1)$ < priority(d_2): The rule converted from d_2 is an exception of the rule converted from d_1 if α_1 subsumes α_2, that is, if $w \cup \alpha_2 \mid -\alpha_1$. Because defaults with same priority are guaranteed to be mutually exclusive (see constraint 4 earlier), α_2 can only be subsumed by exactly one prerequisite of exactly one default in a collection of defaults of equal priorities. However, the subsumption restriction may lead to more than one exception rule of the rule corresponding to a default d_1. In other words, a rule in T can be an exception for at most one other rule. But a rule can have more than one exception rule, when this occurs, all generated exceptions are linked in a false chain in which the order is irrelevant because of the mutual exclusion constraint (constraint 4 earlier).

Finally, the incremental knowledge acquisition process with RDR corresponds to an incremental development of the corresponding default theory D. In D, if an expert disagrees with the returned consequence of the chosen default with priority n, s/he will need to add to the theory a new default with a priority larger than n. This corresponds to adding an exception rule to a corresponding RDR tree.

This section completes a two-way mapping between RDR trees and default theories. The significance of this mapping is in that, it shows that the RDR framework can be seen to solve the problem of controlling the interactions and conflicts between normal defaults. This is done by implicitly assigning priorities to defaults during the knowledge acquisition process. This implicit priority is captured through the context of rules in the knowledge base. The mapping from an RDR tree to a default theory makes these priorities explicit. In characterising constraints on default theories that can be mapped to RDR trees, we highlight the expressiveness limits of RDR trees. NRDR expressive power is beyond those limits, and the range of default theories that can be mapped to NRDR is wider.

Mapping NRDR to default theories

An NRDR knowledge base is a collection of interacting RDR trees. To convert an NRDR knowledge base K to a default theory (D, W), every single RDR tree in K is converted to a default theory according to the mapping which was presented in Sect. "Systematic Conversion of NRDR to default logic". In doing this, every NRDR concept c is mapped to a set of prioritised defaults that have the same consequence $\beta \in \{c, \sim c\}$. Note that the conclusions of all RDR trees within an NRDR knowledge base are propositional values, therefore any consequence β has a propositional value (see Definition 1 in Sect. 4.1.1 for semantics of rules in an RDR tree). The complete set of resulting defaults from K can be partitioned into a collection of subsets of defaults, where defaults in a given subset have the same consequence. The number of different consequences (or subsets) corresponds to the number of different concepts in K.

Priorities of defaults, generated during the mapping of every NRDR concept (see Sect. "Mapping NRDR to default theories"), ensure mutual exclusion when applying defaults which have the same consequence. However during inference, defaults with different consequences can apply simultaneously. Given a subset of defaults in D which have the same consequence, the default with the highest priority in this subset is applied first. Analysis of what is permitted in the inference process over D will be expanded in the next section, where we analyse how the update policies and the hierarchical structure of an NRDR knowledge base translate into a number of constraints on the resultant default theory (D, W). This analysis will also allow a characterisation of a default theory which can actually be mapped to NRDR. We also sketch how such a default theory, which follows those constraints, can be mapped to an NRDR knowledge base.

Mapping default theories to NRDR

To make our analysis of which default theories can be mapped to NRDR succinct, we introduce the following definition:

Definition Given a set of defaults D, a *stratum* of D is a proper subset of defaults $S \subset D$ such that any two defaults in S have the same absolute consequence $|\beta|$, but different pre-requisites. Further, given S is a *stratum* of D where defaults in S have absolute consequence $|\beta|$, $\forall d \in D$ with absolute consequence $|\beta|$, we have d ∈ S. Where two defaults $d_1 = \frac{\alpha_1:\beta_1}{\beta_1}$ and $d_2 = \frac{\alpha_2:\beta_2}{\beta_2}$ are said to have the same absolute consequence, denoted by $|\beta_1|$ *or* $|\beta_2|$,if $\beta_1 = \beta_2$ *or* $\beta_1 = \sim\beta_2$.

A default theory (D, W) mapped from an NRDR K, is a collection of *strata* where every stratum corresponds to a concept definition. Every stratum is a prioritised set of defaults which follow the constraints listed in Sect. 4.4.3. Briefly, these are:

7. Every default must be a normal default (see Sect. 4.4.1).
8. Consequences for all defaults are propositional values.
9. Every default $d \in D$ must have a priority n.
10. Any two defaults in $D$with equal priorities must have mutually exclusive pre-requisites.
11. For any two defaults: $d_1 = \frac{\alpha_1:\beta_1}{\beta_1}$ and $d_2 = \frac{\alpha_2:\beta_2}{\beta_2}$: β_2 is not required to compute α_1. That is, if $W| - \alpha_2$ then $W\backslash\{\beta_1\}| - \alpha_2$.
12. Exactly one default, d_{default}, has the lowest priority.

For defaults in D from different strata, constraint 4 does not apply, that is, defaults with same priority may apply. This is completely consistent with semantics of NRDR knowledge base. An NRDR knowledge base can give more than one conclusion at any one time. Moreover, constraint 5 does not apply for defaults from different strata of D. In other words, given two defaults with different consequences $d_1 = \frac{\alpha_1:\beta_1}{\beta_1}$ and $d_2 = \frac{\alpha_2:\beta_2}{\beta_2}$ then: β_2 can be used to compute α_1. However, a constraint, which corresponds to the prohibition of recursion and circularity of concepts in an NRDR knowledge base, applies to dependency between defaults from different strata. To express this constraint, we introduce the notion of a *dependency graph* for a set of defaults:

Definition A dependency graph G of a set of defaults D is a directed graph, where every default in D corresponds to a node in G, and if the application of $d_1 = \frac{\alpha_1:\beta_1}{\beta_1}$ is essential to apply $d_2 = \frac{\alpha_2:\beta_2}{\beta_2}$, that is only $W \cup \{\beta_1\}| - \alpha_2$, then there is a direct link from the node corresponding to d_2 to the node corresponding to d_1 in G.

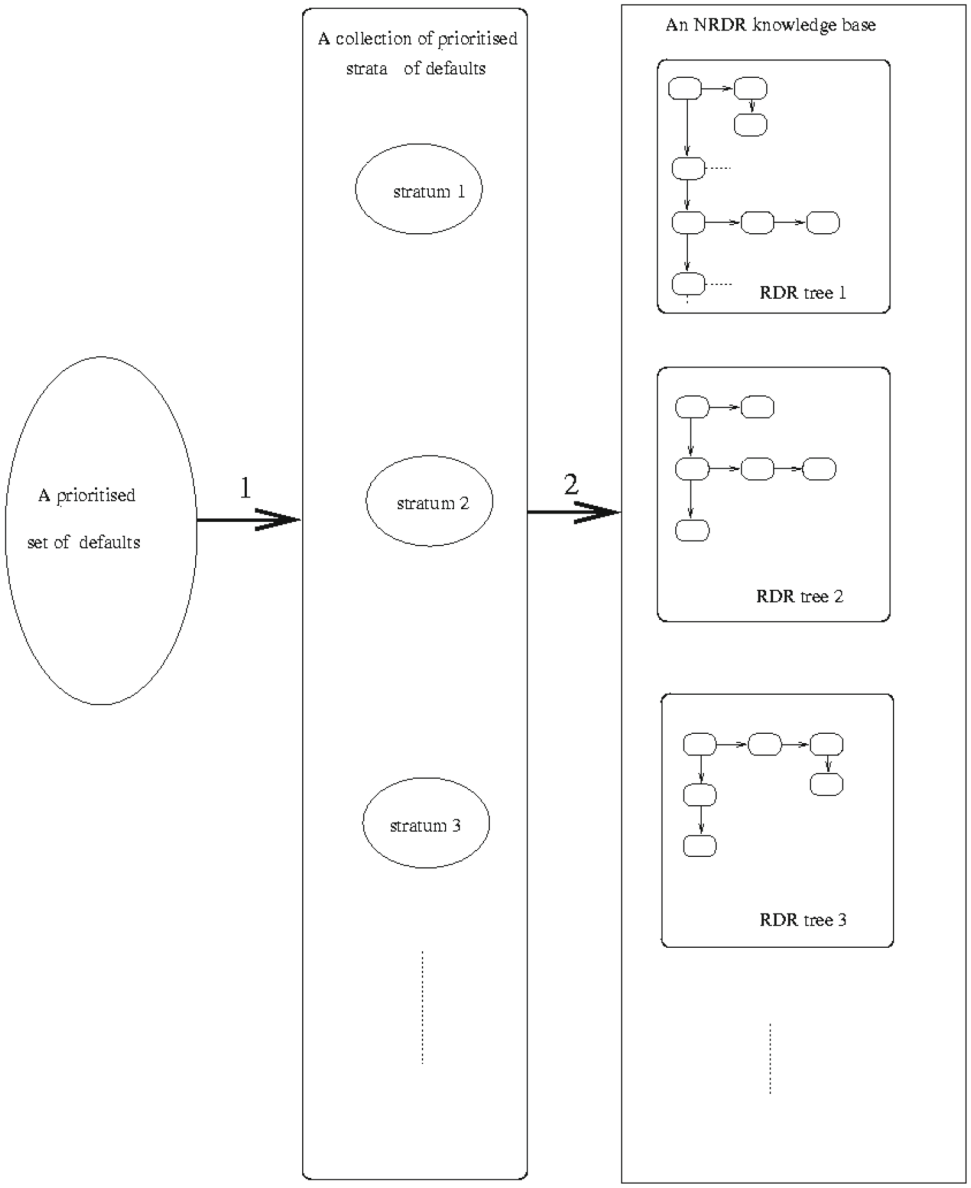

Fig. 4 Conversion steps from a prioritised default theory D to an NRDR Knowledge Base K. Step 1 is grouping the defaults into a set of prioritised strata of defaults. Step 2 is converting each of the stratum to an RDR tree (as shown in Fig. 3). The result is a collection of RDR trees forming the NRDR knowledge base K

The constraint on defaults, which corresponds to prohibition of recursion and circularity of concepts in an NRDR knowledge base, can then be expressed as follows:

For a default theory (D, W) to be convertible to an NRDR representation, then the dependency graph G of D must be acyclic.

In summary, given a default theory (D, W)which follows conditions 1 to 6 for all its strata, and the above constraint on defaults from different strata, then (D, W)can be converted to an NRDR knowledge base. The conversion is as follows:

1. Group defaults into strata.
2. Apply conversion of defaults to RDR trees—outlined in chapter 4—to each stratum.

The above translation is illustrated in Fig. 4.

Discussion and conclusion

The translation of RDR to a default theory discussed does not allow using more than one default during inference, which in contrast is generally possible in a knowledge-based system implementing a default theory. RDR imposes the maximum possible restriction on interactions between defaults: No interactions between defaults are actually possible. While

this eases maintenance, it decreases the inference power of the corresponding default theory. In this paper, we discussed how a default theory can be mapped to an NRDR knowledge base. We highlighted restrictions on the corresponding defaults. Inference restrictions still applied, but these were much weaker than those imposed by the RDR formalism. For example, the total mutual exclusion imposed on defaults in RDR equivalent default theories no longer applied. That is, much more powerful default theories can be mapped from (and to) an NRDR knowledge base. In other words, NRDR is a more powerful knowledge representation scheme than RDR. The knowledge acquisition process with Nested Ripple Down Rules as the underlying representation has three distinct features: Firstly, it allows the expert to introduce new domain terms during the knowledge acquisition process. Secondly, these terms are operational while still incomplete, e.g. they are always open for amendments. Thirdly, the knowledge base is viewed as an interconnected whole and multiple points of change are available during maintenance.

References

1. Aggarwal C, Yu P (2010) On clustering massive text and categorical data streams. Knowl Inf Syst 24:171–196
2. Akerkar R, Sajja P (2010) Knowledge-based systems. Jones and Bartlett, Ontario
3. Barouni-Ebrahimi M, Ghorbani A (2008) An interactive search assistant architecture based on intrinsic query stream characteristics. Comput Intell 24(2):158–190
4. Beydoun G, García-Sánchez F, Lopez-Lorca A, Vincent-Torres C et al (2013) Providing metrics and automatic enhancement for hierarchical taxonomies. Inf Process Manag 49(1):67–82
5. Beydoun G, Hoffmann A (1998) Simultaneous Modelling and Knowledge Acquisition using NRDR. In: 5th Pacific Rim Conference on Artificial Intelligence (PRICAI98). Springer, Singapore
6. Beydoun G, Hoffmann A (2001) Theoretical basis for hierarchical incremental acquisition. Int J Human Comput Stud 54(3):407–452
7. Beydoun G, Hoffmann A, Hamade R (2010) Automating dimensional tolerancing using Ripple Down Rules (RDR). Expert Syst Appl 37(7):5101–5109
8. Beydoun G, Lopez-Lorca A, García-Sánchez F et al (2011) How do we measure and improve the quality of a hierarchical ontology? J Syst Softw 84(12):2363–2373
9. Beydoun G, Hoffmann A (2013) Dynamic evaluation of the development process of knowledge based information systems. Knowl Inf Syst 35(1):233–247
10. Beydoun G, Low G, García-Sánchez F, Valencia-García R, Martínez-Béjar R (2014) Identification of ontologies to support information systems development, Information Systems, vol 46. Elsevier, Amsterdam, pp 45–60
11. Beydoun G, Tran N, Low G, Henderson-Sellers B (2006) Foundations of ontology-based methodologies for multi-agent systems. In: Kolp M, Bresciani P, Henderson-Sellers B, Winikoff M (eds) Procs. AOIS2005 LNAI 3529. Springer, Berlin, pp 111–123
12. Brewka G (1994) Reasoning about priorities in default logic. In: The Twelfth National Conference on Artificial Intelligence (AAAI-94). AAAI, Washington
13. Bichindaritz I, Montani S (2009) Introduction to the special issue on case-based reasoning in the Health Sciences. Comput Intell 25(3):161–164
14. Cao T, Compton P (2006) Knowledge acquisition evaluation using simulated experts managing knowledge in a World of Networks. Springer, New York, pp 35–42
15. Compton P, Kang B, Preston P et al (1993) Knowledge Acquisition Without Knowledge Analysis. European Knowledge Acquisition Workshop (EKAW93). Springer, New York
16. Compton P, Peters L, Edwards G et al (2006) Experience with Ripple Down Rules. Knowl Based Syst 19(5):356–362
17. Courtney A, Antoniou G, Foo N (1996) Exten: A System for Computing Default Logic Extensions. Fourth Pacific Rim International Conference on Artificial Intelligence (PRICAI96), vol 1. Springer, Australia, pp 411–423
18. Dunin-Kęplicz B, Strachocka A (2013) Perceiving rules under incomplete and inconsistent information, computational logic in multi-agent systems. Lect Notes Comput Sci 8143(2013):256–272
19. Egyed A (2011) Automatically detecting and tracking inconsistencies in software design models. IEEE Trans Softw Eng 37(2):188–204
20. Gañán C, Mata-Díaz J, Muñoz JL, Esparza O, Alins J (2014) A model for revocation forecasting in public-key infrastructures. Knowl Inf Syst
21. Hamade RF, Moulianitis VC, D'Addonna D et al (2010) A dimensional tolerancing knowledge management system using Nested Ripple Down Rules (NRDR). Eng Appl Artif Intell 23(7):1140–1148
22. Hyeon J, Oh K, Kim Y et al (2016) Constructing an initial knowledge base for medical domain expert system using induct RDR. In: International Conference on Big Data and Smart Computing (BigComp). Korea
23. Kivinen J, Mannila H, Ukkonen E (1993) Learning rules with local exceptions. ACM Conference on Computational Theory, Santa Cruz
24. Kwok RBH (2000) Translations of Ripple Down Rules into logic formalisms. In: The 12th European Knowledge Acquisition Conference (EKAW2000). Springer, France
25. Lopez-Lorca A, Beydoun G, Valencia-Garcia R, Martinez-Bejar R (2016) Supporting agent oriented requirement analysis with ontologies. Int J Human Comput Stud 87:20–37
26. Othman SH, Beydoun G, Sugumaran V (2014) Development and validation of a disaster management metamodel. Inf Process Manag 50(2):235–271
27. Reiter R (1980) A logic for default reasoning. Artif Intell 13:81–132
28. Sadraei E, Aurum A et al (2007) A field study of the requirements engineering practice in Australian software industry. Requir Eng 12(3):145–162
29. Scheffer T (1995) Learning rules with nested exceptions. International Workshop on Artificial Intelligence
30. Tran N, Beydoun G, Low G (2007) Design of a Peer-to-Peer Information Sharing MAS Using MOBMAS (Ontology-Centric Agent Oriented Methodology). In: Wojtkowski W, Wojtkowski WG, Zupancic J, Magyar G, Knapp G (eds) Advances in Information Systems Development. Springer, New York, pp 63–76
31. Wada T, Horiuchi T, Motoda H et al (1998) A New Look at Default Knowledge in Ripple Down Rules Method. Pacific Rim Knowl-

edge Acquisition Workshop (PKAW98). National Univeristy of Singapore, Singapore
32. Wobcke W, Chan R, Limaru A (2006) A Call Handling Assistant for Mobile Devices. In: International Conference on Intelligent Agent Technology (IAT06). IEEE/WIC/ACM, Hong Kong
33. Xu D, Wijesooriya C, Wang X-G et al (2011) Outbound logistics exception monitoring: a multi-perspective ontologies' approach with intelligent agents. Expert Syst Appl 38(11):13604–13611