Reasoning about Knowledge using Rough Sets

Weiru Liu

School of ISE, University of Ulster at Jordanstown
Newtownabbey, Co. Antrim BT37 0QB, UK
w.liu@ulst.ac.uk

Abstract. In this paper, we first investigate set semantics of propositional logic in terms of rough sets and discuss how truth values of propositions (sentences) can be interpreted by means of equivalence classes. This investigation will be used to answer queries that involve general values of an attribute when the actual values of the attribute are more specific. We then explore how binary relations on singletons can be extended as set-based relations, in order to deal with non-deterministic problems in an information system. An example on test-case selection in telecommunications is employed to demonstrate the relevance of these investigations, where queries either contain values (concepts) at higher granularity levels or involve values of an attribute with non-deterministic nature or both.

1 Introduction

In rough sets, information and knowledge is usually represented using data tables or decision tables [9]. Each column in such a table is identified by an attribute, which describes one aspect of the objects being processed in an information system. Attribute values can be defined at different granularity levels, according to a specific requirement. In the past, most of the research work using rough sets has focused on how to use or manipulate or discover knowledge from the information carried by a table, under the assumption that attribute values in such a table have been chosen at the right granularity level (e.g., [10], and [11]).

Nevertheless, problems of granules of attribute values in query processing have been addressed by some researchers. In [5], a high level data table is derived from a lower level table when the concept required by the query is not matched by the values of the relevant attribute in the lower level table. In this case, the values of the attribute in the higher level table are replaced by more general values (concepts). In [12], approaches to answering non-standard queries in distributed information systems were explored. Values of an attribute at different granule levels are arranged as nodes in a tree structure, with the attribute name as the root and the most specific values of the attribute as the leafs. In contrast to the two approaches above, in [1], rough predicates are defined. These predicates associate user-defined lower and upper approximations with attribute values, or with logical combinations of values, to define a rough set of tuples for the result of the predicates. Each predicate, similar to the definition of a function on an

entity in a functional data model, does not define an attribute as a function on a relation rather it chooses a possible value of an attribute as a function of the relation. For example, if relation *Horse* has an attribute *Age*, then a predicate *young(Horse)* is defined with the lower approximation containing those horses with age below 1, and the upper approximation consisting of those horses with ages in {2,3,4}.

Another common problem associated with attribute values in a data table is that an object has a set of possible values instead of just one value for a particular attribute. Although only one of the values is surely true but we cannot say (determine) precisely which value it is yet. When a data table involves this kind of attributes (non-determinictic), it is necessary to extend usual binary equivalence relations to be set-based relations (e.g., [13]).

In this paper, we aim at solving these two problems when a query either contains values (concepts) at higher granularity levels or involves values of an attribute with non-deterministic nature or both. We discuss how to reason about knowledge at different levels from a data table using the combination of logic and rough sets, when values of an attribute are given at the most specific level, in order to answer queries. To achieve this objective, we investigate set semantics of propositional logic first in rough sets and then explore the relationships between equivalence relations and propositions in terms of partitioning a data table. We then discuss how extended set-based binary relations can be applied to compute tighter bounds when the values of an attribute are non-deterministic (set-based) [6]. An example on test-case selection in telecommunications is employed to demonstrate the relevance of our research result. The paper is organized as follows. Section 2 introduces the basic notions of rough sets and set based computations in non-deterministic information systems. Section 3 explores the set semantic of propositional logic. Section 4 discusses how to apply the results to solve complex queries. Finally Section 5 summarizes the paper.

2 Deterministic and Non-deterministic Information Systems

Basics of rough sets: Let U be a set, also called a universe, which is non-empty and contains a finite number of objects (this assumption will not lose general properties of rough sets), and R be an equivalence relation on U. An equivalence relation is reflexive, symmetric and transitive. An equivalence relation R on U divides the objects in U into a collection of disjoint sets with the elements in the same subset indiscernible. We denote each partition set, known as an *equivalence class*, as W_l^R and an element in W_l^R as w_{lj}^R. The family of all equivalence classes $\{W_1^R, ..., W_n^R\}$ is denoted as U/R. W_l^R and w_{lj}^R are simplified as W_l and w_{lj} respectively when there is no ambiguity about which equivalence relation R we are referring to.

Given a universe, there can be several ways of classifying objects. Let R and R' be two equivalence relations over U, $R \cap R'$ is a refined equivalence relation.

$\cap$ can be understood as *and*. The collection of equivalence classes of $R \cap R'$ is

$$U/(R \cap R') = \{W_l^R \cap W_j^{R'} \mid W_l^R \in U/R, W_j^{R'} \in U/R', W_l^R \cap W_j^{R'} \neq \emptyset\}. \quad (1)$$

Equivalence relation $R_1 \cap R_2 \cap ... \cap R_n$, from $\mathcal{R} = \{R_1, ..., R_n\}$ on a universe U, is usually denoted as $IND(\mathcal{R})$ [9].

Definition 1. *Structure $(U, \Omega, V_a)_{a \in \Omega}$ is called an information system where:*

1. U is a finite set of objects,

2. Ω is a finite set of primitive attributes describing objects in U,

3. For each $a \in \Omega$, V_a is the collection of all possible values of a. Attribute a also defines a function, $a : U \to V_a$, such that $\forall u \in U$, $\exists x \in V_a$, $a(u) = x$.

Such an information system is also called a deterministic information system. For a subset $Q \in \Omega$, function R_Q defined by $u_1 R_Q u_2 \Leftrightarrow \forall a \in Q, a(u_1) = a(u_2)$ is an equivalence relation. When Q consists of only one attribute a, i.e., $Q = \{a\}$, R_Q is called an *elementary equivalence relation*. In [9], each equivalence class in an elementary equivalence relation is referred to as an *elementary concept*. Any other non-elementary equivalence relation R_Q can be represented by a set of elementary equivalence relations using the following expression, $R_Q = \cap_{a \in Q} R_{\{a\}}$.

Definition 2. *Let U be a universe and R be an equivalence relation on U. For a subset X of U, if X is the union of some W_l^R, then X is called R-definable; otherwise X is R-undefinable.*

Let $\mathcal{R} = \{R_1, ..., R_n\}$ be a collection of n equivalence relations. Then any subset $X \subseteq U$ obtained by applying $\cap$ and $\cup$ to some equivalence classes in any $U/\mathbf{R}$ (where $\mathbf{R} \subseteq \mathcal{R}$) is $IND(\mathcal{R})$-definable. This statement identifies all the concepts that are definable under equivalence relation $IND(\mathcal{R})$. For any subset X of U with a given R, we can also use two subsets of U to describe it as follows:

$$\underline{R}X = \cup\{W_l^R \mid W_l^R \subseteq X\}, \qquad \overline{R}X = \cup\{W_l^R \mid W_l^R \cap X \neq \emptyset\}.$$

When X is R-definable, $\underline{R}X = \overline{R}X = X$. Subsets $\underline{R}X$ and $\overline{R}X$ are called *R-lower* and *R-upper approximations* of X.

Set-based computation: In real world applications, not all attributes in a data table will be assigned with single values against individual objects (e.g., [6],[8]). The definition below defines those information systems where an attribute can have a set of values for a particular object.

Definition 3. *Structure $(U, \Omega, V_a)_{a \in \Omega}$ is called a non-deterministic information system where:*

1. U is a finite set of objects,

2. Ω is a finite set of primitive attributes describing objects in U,

3. For each $a \in \Omega$, V_a is the collection of all possible values of a. Attribute a also defines a function, $a : U \to 2^{2^{V_a}}$, such that $\forall u \in U$, $\exists S \subseteq 2^{V_a}$, $a(u) = S$.

Table 1. A sample data table

U	Manufactures	Color	Weight (g)	Age-Group
u_1	{{UK, France}, {UK, Japan}}	{grey, black}	{1,2}	{infant, Toddler}
u_2	{{Japan, Korea }, {UK, Japan}}	{grey}	{2,3,4}	{Toddler, Pre-school}
u_3	{{France, Germany}}	{grey}	{2,3,4}	{Pre-School}
u_4	{{Japan, Germany}}	{grey, brown}	{1}	{All}
u_5	{{UK, France}}	{brown}	{4,5}	{Teenager}

Table 1 shows a non-deterministic information system (also called an attribute system in [3]) with all attributes non-deterministic. Attribute *Manufactures* is even more complicated: each object is assumed to be manufactured jointly by two countries. When we don't know for sure which two countries manufactured a specific object, we assign several pairs of possible countries, such as for u_1. In [3], four possible explanations of the values in $a(u)$, a set assigned to an object against a particular attribute, are provided. We supplement 5th explanation on top of that to cover the situation as shown by attribute *Manufactures*. These five explanations are:

(1) $a(u)$ is interpreted disjunctively and exclusively: one and only one value is correct, such as the weight of an object (assume we use a closest integer to measure the weight of each object),

(2) $a(u)$ is interpreted disjunctively and non-exclusively: more than one value may be correct, such as the (suitability of) age groups of a toy,

(3) $a(u)$ is interpreted conjunctively and exclusively: all the correct values are included, such as the color of a toy (when we list all the colors involved),

(4) $a(u)$ is interpreted conjunctively and non-exclusively: all the values (but not limited to the values) in $a(u)$ are correct, such as the color of a toy (when we list main colors only),

(5) the combination of (1) and (3): one and only one value (subset) is correct and this value is the combination of individual values, such as the manufactures of a toy.

For the first 4 categories, set-based operations are enough to deal with attribute values. However, for category 5, we will need to use interval-based operations, since each value itself is again a set.

Definition 4. *(from [13]) Let r be a binary relation on V_a, a set of possible values of attribute a. A pair of extended binary relations (r_*, r^*) on $2^{V_a} \setminus \emptyset$ is defined as:*

$$Ar_*B \iff (\forall a \in A, \forall b \in B)\ arb, \qquad Ar^*B \iff (\exists a \in A, \exists b \in B)\ arb. \quad (2)$$

Let Q be a query that involves values in subset B of V_a, then retrieval sets

$$Ret_*(Q) = \{u_i \mid a(u_i) = A, Ar_*B\}; \qquad Ret^*(Q) = \{u_i \mid a(u_i) = A, Ar^*B\},$$

give the lower and upper approximations of a set of objects that support query Q under condition B. For example, if Q = 'select grey objects', and we set $B = \{grey\}$ and r be '$=$', then $Ret_*(Q) = \{u_2, u_3\}$ and $Ret^*(Q) = \{u_1, u_2, u_3, u_4\}$.

However, Eqs. in (2) cannot be used to deal with values for attribute *Manufactures* because there may be several subsets of values assigned to an object. Therefore, we need to further extend the equations.

Given two sets $A_1, A_2 \in 2^{V_a}$ with $A_1 \subseteq A_2$, set $\mathcal{A}$ defined by $\mathcal{A} = [A_1, A_2] = \{X \in 2^{V_a}, A_1 \subseteq X \subseteq A_2\}$ is called a closed *interval set*.

Definition 5. *Let $\mathcal{A}$ and $\mathcal{B}$ be two interval sets from 2^{V_a}. A pair of extended binary relations $(\supseteq_*, \supseteq^*)$ on $2^{2^{V_a}} \setminus \emptyset$ is defined as:*
$\mathcal{A} \supseteq_* \mathcal{B} \Longleftrightarrow \forall X \in \mathcal{A}, \forall Y \in \mathcal{B}\ X \supseteq Y$,
$\mathcal{A} \supseteq^* \mathcal{B} \Longleftrightarrow \exists X \in \mathcal{A}, \exists Y \in \mathcal{B}\ X \supseteq Y$.

Let Q be a query that involves conditions described in interval set $\mathcal{B} = [B_1, B_2] \subseteq 2^{V_a} \setminus \emptyset$, then two retrieval sets $Ret_*(Q)$ and $Ret^*(Q)$ defined by

$$Ret_*(Q) = \{u_i, \mid a(u_i) = \mathcal{A}, \mathcal{A} \supseteq_* \mathcal{B}\};\ \ Ret^*(Q) = \{u_i, \mid a(u_i) = \mathcal{A}, \mathcal{A} \supseteq^* \mathcal{B}\}, \tag{3}$$

give the lower and upper approximations of a set of objects that support query Q. For instance, if query Q says 'select UK manufactures related objects' and we set $\mathcal{B} = [\{UK\}, \{UK\}] = \{\{UK\}\} \subseteq 2^{V_{manu}} \setminus \emptyset$, then $Ret_*(Q) = \{u_1, u_5\}$ and $Ret^*(Q) = \{u_1, u_2, u_5\}$.

3 Set Semantics of Propositional Logic

A deterministic information system can be best demonstrated using a data table in rough sets. Each data table contains a number of rows labelled by objects (or states, processes etc.) and columns by primitive attributes. Each primitive attribute is associated with a set of mutually exclusive values that the attribute can be assigned to. Each attribute also defines an elementary equivalence relation and each equivalence class of the relation is uniquely identifiable by an attribute value. When an attribute can choose values from different value sets, only one of the possible value sets will be used in a particular data table. Each equivalence class in a partition is also naturally corresponding to a concept which can be characterized by a proper proposition. In the following, if we take P, $P = \{q_1, q_2, ..., q_n\}$, as a finite set of atomic propositions, then as usual $\mathcal{L}(P)$ is used to denote the propositional language formed from P. $\mathcal{L}(P)$ consists of P, logical constants *true* and $false$, and all the sentences constructed from P using logical connectives $\{\neg, \wedge, \vee, \rightarrow, \leftrightarrow\}$ as well as parentheses (,).

Definition 6. *Let U be a non-empty universe with a finite number of objects, P be a finite set of atomic propositions. Function $val : U \times P \rightarrow \{true, false\}$ is called a valuation function, which assigns either true or $false$ to every ordered pair (u, q) where $u \in U$ and $q \in P$.*

$val(u, q) = true$, denoted as $u \models_S q$, can be understood as *q is true with respect to object u in S*, where $S = (U, \Omega, V_a)_{a \in \Omega}$ is an information system. Based on *val*, another mapping function $v : P \rightarrow 2^U$ can be derived as:

$$v(q) = \{u \mid u \in U, u \models_S q\}, \tag{4}$$

where $u \in v(q)$ is interpreted as q holds at state u (or is proved by object u). Function v can be extended to a mapping $v : \mathcal{L}(P) \rightarrow 2^U$ as follows. For any $\phi, \psi \in \mathcal{L}(P)$,

$$v(\phi \wedge \psi) = \{u \mid u \in U, (u \models_S \phi) \ and \ (u \models_S \psi)\}, \tag{5}$$

$$v(\phi \vee \psi) = \{u \mid u \in U, (u \models_S \phi) \ or \ (u \models_S \psi)\}, \tag{6}$$

$$v(\neg\phi) = \{u \mid u \in U, u \not\models_S \phi\}. \tag{7}$$

Therefore, the subset of U containing those objects supporting formula ϕ (non-atomic proposition) can be derived through the initial truth assignment *val*. An atomic proposition can be formally defined as: there exists one and only one attribute $a \in \Omega$ in an information system $(U, \Omega, V_a)_{a \in \Omega}$, such that there exists only one x, $x \in V_a$, $v(q) = \{u \mid a(u) = x\}$.

Definition 7. *Let (U, R, P, val) be a structure where R is an elementary equivalence relation on U, P is a finite set of atomic propositions, and val is an valuation function on $U \times P$. If there is a subset $P' = \{q_1, ..., q_n\}$ of P such that $U/R = \{v(q_1), v(q_2), ..., v(q_n)\}$ holds, then subset P' is said to be equivalent to R, denoted as $U/R = v(P')$.*

$v(P')$ is defined as a collection of subsets of U, i.e., $v(P') = \{v(q_1), v(q_2), ..., v(q_n)\}$ for all $q_l \in P'$. This definition suggests that there can be a subset of a set of atomic propositions P which is functionally equivalent to an elementary equivalence relation in terms of partitioning a universe, regarding to a particular aspect (attribute) of the objects in the universe.

Table 2. A sample test case data table

U	ID	Engineer	Feature	Purpose
c_1	408	N Ross	STM-4o	
c_2	356	N Ross	STM-1o	Undefined
c_3	228	T Smith	Connections	Undefined
c_4	175	T Smith	Protection Switching	{{Forced Path Protection Switch is successful when Standby Path is faulty}, {Pass criteria: Path Protection to the Standby Path occurs}, {Fail criteria: Path Protection to the Standby Path} not occur}}
c_5	226	T Smith	Synchronisation	{{STM-N/ESI ports added to the SETG priority list, Ensure ports not logically equipped not added}}
c_6	214	none	2Mbit/s	Undefined
c_7	48	N Ross	STM-4o	
c_8	50	N Ross	STM-1o	{{Can configure Alarm Severity of Card Out, Default value of Severity is Minor}, {When Severity is changed Alarm should raise}}
c_9	72	N Ross	STM-1o	{{Can display card type, Card variant, and Unique serial No}, {Otherwise, Alarm should raise}}
c_{10}	175	P Hay	STM-1o	{{HP-UNEQ Alarm raised when C2=00 5 times}, {Alarm not raised when C2 is set 00}}

Example 1. Assume U is a universe containing 10 simplified snap-shot of test cases in telecommunications (Table 2). Let R be an equivalence relation on U which divides U into three disjoint sets, one with those cases for which the value of *Purpose* is empty, one with *Purpose Undefined*, and one with *Purpose Defined* (if the details of *Purpose* of an object are given, we say it is defined). Similarly, relation R', which divides U into six disjoint sets based on the names of Feature, is also an equivalence relation. The equivalence classes generated by R and R' are: $U/R = \{\{c_1, c_7\}, \{c_2, c_3, c_6\}, \{c_4, c_5, c_8, c_9, c_{10}\}\}$ and $U/R' = \{\{c_1, c_7\}, \{c_2, c_8, c_9, c_{10}\}, \{c_3\}, \{c_4\}, \{c_5\}, \{c_6\}\}$. $R = R_{\{Purpose\}}$ and $R' = R_{\{Feature\}}$ are elementary equivalence relations, but $R \cap R'$ is not. Let q_1, ..., q_6 be six atomic propositions, *'A test case has feature STM-4o'*, ..., *'A test case has feature 2Mbit/s'* respectively, these six atomic propositions divide U into six disjoint subsets: $v(q_1) = \{c_1, c_7\}$, $v(q_2) = \{c_2, c_8, c_9, c_{10}\}$, $v(q_3) = \{c_3\}$, $v(q_4) = \{c_4\}$, $v(q_5) = \{c_5\}$, and $v(q_6) = \{c_6\}$, where $v(q_i) = W_i^{R'}$ for $i = 1, ..., 6$. Therefore $v(P') = U/R$.

Definition 8. *Let (U, R, P, val) be a structure defined in Definition 7. For a formula ϕ in $\mathcal{L}(P)$, if $v(\phi)$ defined in Eq. (4) is R-definable then ϕ is said to be an R-definable formula. Otherwise, ϕ is R-underfinable. Formulae true and $false$ are always R-definable with $v(true) = U$ and $v(false) = \emptyset$.*

Theorem 1. *Let $(U, IND(\mathcal{R}), P, val)$ be a structure defined in Definition 7 with $\mathcal{R} = \{R_1, R_2, ..., R_n\}$ containing n elementary equivalence relations on U. When $U/R_i = v(P_i)$ holds for $i = 1, ..., n$ and $P_i \subseteq P$, every formula in $\mathcal{L}(P')$ ($P' = \cup_i P_i$) is an $IND(\mathcal{R})$-definable formula.*

Example 2. Let U be a set of objects containing a group of 10 test cases as given in Table 2. Let P_1 and P_2 be two subsets of a set of atomic propositions P as $P_1 = \{q_{11}, q_{12}, q_{13}\}$={Purpose is empty, Undefined, Defined} and $P_2 = \{q_{21}, q_{22}, q_{23}, q_{24}, q_{25}, q_{26}\}$={feature with STM-4o, STM-1o, Connections, protection-Switching, Synchronisation, 2Mbit/s}. These two subsets of atomic propositions are equivalent to the two elementary equivalence relations, R and R', in Example 1.

The following formulae:

ϕ =test cases with feature *STM-1o* and purpose given,

ψ =either test cases with feature *Connections* or with purpose undefined,

φ =test cases with feature is neither *STM-4o* nor *STM-1o* and purpose known,

which can be re-written into disjunctive normal forms:

$\phi = (q_{13} \wedge q_{22})$,

$\psi = (q_{12}) \vee (q_{23})$,

$\varphi = (q_{13} \wedge \neg(q_{12} \vee q_{22})) = (q_{13} \wedge (\neg q_{21} \wedge \neg q_{22}))$

are all $R_1 \cap R_2$-definable. The subsets of objects supporting these formulae, i.e., $v(\phi)$, $v(\psi)$, and $v(\varphi)$ are $\{c_8, c_9, c_{10}\}$, $\{c_2, c_3, c_6\}$, and $\{c_4, c_5\}$ respectively.

Valuation function *val* requires full information about every ordered pair (u, q) in the space $U \times P$. This is an ideal situation where for every formula ϕ in $\mathcal{L}(P)$ it is possible to identify all the objects that support ϕ, and this

set is $v(\phi)$ through Eq. (4). When a universe U is very large, it may not be practical to require function *val* being fully specified, but be quite reasonable to have information about a particular elementary equivalence relation (R) and its corresponding equivalent subset of a set of atomic propositions (P'). In this case, $v(\phi)$ can be determined only when $\phi \in \mathcal{L}(P')$, as $v(\phi)$ can be represented using elements in U/R.

Still, this is an unavoidable question that one may ask: is it realistic to assume that the relevant equivalence relations (hence equivalence classes) are given as prior knowledge? The answer may be 'No' for many applications, however, the answer is 'Yes' for the test-case selection scenario in telecommunications, because the feature or sub-feature of all already designed test cases must be given.

Definition 9. *Structure* (U, R, P, P', val) *is called a partial rough logic theory*

1. *U is a universe consisting of a finite number of objects,*
2. *P is a finite set of atomic propositions,*
3. *R is an elementary equivalence relation on U,*
4. *Valuation function val is only partially specified on space $U \times P$*
5. *$P' \subset P$. For each $q_l \in P'$, $v(q_l) = W_l$ and W_l is in U/R.*

Based on a partial rough logic theory (U, R, P, P', val), the following equations hold only for formulae ϕ, ψ in $\mathcal{L}(P')$.

$$v(\neg\phi) = U \setminus v(\phi), \qquad v(\phi \wedge \psi) = v(\phi) \cap v(\psi),$$
$$v(\phi \vee \psi) = v(\phi) \cup v(\psi), \qquad v(\phi \rightarrow \psi) = (U \setminus v(\phi)) \cup v(\psi).$$

Each partial rough logic theory defines the set of objects supporting a formula in $\mathcal{L}(P')$ precisely with the knowledge of relevant elementary equivalence relation R. That is, all formulae in $\mathcal{L}(P')$ are $R-$definable. For $\psi \in \mathcal{L}(P) \setminus \mathcal{L}(P')$ which is not R-definable, it is only possible to define the upper and lower approximations of $v(\phi)$.

$$\underline{v}(\phi) = \cup\{v(\psi) \mid \psi \models \phi, \psi \in \mathcal{L}(P')\} = \cup\{W_i \subseteq v(\psi) \mid \psi \models \phi, \psi \in \mathcal{L}(P')\}, \quad (8)$$

$$\overline{v}(\phi) = U \setminus \underline{v}(\neg\phi). \quad (9)$$

Eq. (8) defines the lower bound of the set of objects that make formula ϕ true and Eq. (9) gives the upper bound of that set. The algebraic properties of $(\underline{v}, \overline{v})$ can be found in [2]. All objects in the lower bound will definitely satisfy formula ϕ while an object in the upper bound is known not to satisfy $\neg\phi$, therefore, it may support ϕ. In terms of Dempster-Shafer theory of evidence, if a frame of discernment is defined as elements being the equivalence classes of R, then Eq.(8) will yield a belief function and Eq.(9) will produce a plausibility function ([7]).

4 Reasoning about knowledge

From general concepts (values) to specific concepts (values) or vice versa: An information system, exemplified by a data table, provides the basic

information to answer relevant queries. Since each attribute in a data table is confined to an exclusive set of values, some intermediate values cannot always be explicitly shown in this table. When a query involves in an intermediate value, a system has to have an approach to matching it with the more specific/general values available in the table. This process requires additional knowledge about the application domain that is being dealt with. We call the tables holding the domain knowledge as meta-level tables, such as Table 3.

Table 3. A meta-data table

→	Feature-details	Feature
	T25 Alarm Reporting - Unterminated Through Connections	STM-4o
	T24 STM-4 Alarm Correlation - HP-REI masked by HP-RDI	STM-4o
	T20 Eqpt Alarms - ALS-Dis (STM-4)	STM-4o
	T19 Eqpt Alarms - Write Protect Jumper Fitted	STM-4o
	T12 Plug-in Unit Alarms - Unexpected Card	STM-4o
	T11 Plug-in Unit Alarms - Card Out	STM-4o
	T10 Loopback - Operation	STM-4o
	...	

Now we visit Example 1 again. In Table 2, one of the values of attribute *Feature* is *STM-4o*. In fact, *STM-4o* covers wide range of test activities, such as, *T20 Eqpt Alarms - ALS-Dis (STM-4)* or *T10 Loopback - Operation* (see Table 3 for more). Therefore, it is more useful to provide these details in a data table than just giving *STM-4o*. We now replace attribute *Feature* with *Feature-details* and update the values of *Feature-details* as appropriate as shown in Table 4. For instance, if feature *STM-4o* is not replaced by a set of detailed features, it is then difficult to answer the following query **Q1**: select test cases with features relevant to *Plug-in unit alarms*. With Table 4, it is easy to answer Q1. However, it raises problems when queries like Q2 below are issued, **Q2**: select test cases with *STM-4o* related *plug-in* tests.

To deal with the connections/relationships between general and specific concepts in a given domain, meta-level knowledge needs to be available. Meta-level tables can be used as supplements to data tables when answering queries. In this way, knowledge *"T25 Alarm Reporting → STM-4o"* is stored as a record in a meta-level table as shown in Table 3. There are in total 14 most general features, hence 14 meta-level tables are required. Now, let us assume that P is a set of atomic propositions with q_1 standing for *'A test case has feature T25 Alarm Reporting'*, q_2 for *'A test case has feature STM-1o'*, ..., , q_7 for *'A test case has feature Plug-in Unit Alarms'* respectively. Let us also assume that R is an elementary equivalence relation which partitions test cases according to their features. Based on Definition 7, subset $P' = \{q_1, q_2, .., q_7\}$ is equivalent to R and $U/R = v(P')$. Given the knowledge about P' and R, according to Theorem 1, every formula in $\mathcal{L}(P')$ is $R-$definable. Query Q1 above which can be re-written as a proposition, φ_1 = a test case has feature *Plug-in Unit Alarms* $= q_7$, can be answered based on knowledge R. Similarly, query Q2 which means 'a test case has feature *STM-4o* and feature *Plug-in*' can be expressed as $\varphi_2 = (q_1 \vee q_7 \vee q_8 \vee q_9 \vee q_{10} \vee q_{11} \vee q_{12}) \wedge q_7 = q_7$, is also R-definable, where q_8, ..., q_{12} stand for 5

atomic propositions that a test has feature with *T24, T20, T19, T11,* or *T10* respectively (see the details given above). Therefore test cases (objects) supporting it are obtained straightforwardly. However, query **Q3**: select test cases relevant to alarms (or alarm raise), is not R-definable, since test cases $c8$ and $c10$ are also relevant to alarms problems as shown in the column *Purpose* and they cannot be summarized into an equivalence class of R. If we use φ_3 to denote query Q3, we have $q_1 \models \varphi_3$ and $q_7 \models \varphi_3$, where $p_1 \models p_2$ means whenever an interpretation makes p_1 true, it must make p_2 true as well. According to Eqs. (8) and (9),

$$\underline{v}(\varphi_3) = \cup\{v(q) \mid q \models \varphi_3\} = v(q_1) \cup v(q_7) = W_1^R \cup W_7^R = \{c1, c7\},$$
$$\overline{v}(\varphi_3) = U \setminus \underline{v}(\neg\varphi_3) = U \setminus \emptyset = U.$$

$\underline{v}(\varphi_3)$ gives us those test cases which should be definitely selected while $\overline{v}(\varphi_3)$ covers those test cases that might be selected. For this query, $\overline{v}(\varphi_3)$ does not provide much useful information, since it contains all test cases. To further eliminate worthless test cases, we need to make use of other information in the database. Because values of attribute *Purpose* cannot be used to partition the universe due to its non-deterministic nature. When the details of Purpose of a test case are given, it usually contains several possible outcomes of a test, each of which may in turn consist of several symptoms simultaneously. To model this phenomena, we apply set-based computations discussed in Section 2.2.

Table 4. A set based sample test case data table

U	Purpose-key-word
c1	
c2	Undefined
c3	Undefined
c4	{{Forced Path Protection Switch, success, Standby Path faulty}, {Path Protection, Standby Path, occur}, {Path Protection, Standby Path, not occur}}
c5	{{Stm-N/ESI ports, Setg priority list, Ports not logically equipped, not added}}
c6	Undefined
c7	
c8	{{Alarm Severity, Card Out, Default value, Severity, Minor}, {Severity change, Alarm raise}}
c9	{{Card type, Card variant, Unique Serial No}, {Alarm raise}}
c10	{{HP-UNEQ, Alarm raised, C2=00 5 times}, {Alarm not raised, C2 set 00}}

Refining upper bounds using set-based computations: Equipped with Definition 5 and Eqs. in (3), we revise Table 2 *Purpose* to obtain Table 4 *Purpose-key-word* (we only include this attribute in Table 4). It is worth pointing out that when a test case has multiple values for attribute *Purpose*, each value is a possible outcome of that test case and the value cannot be decided until the test case is used in a specific test. In addition for each possible outcome, a set of joint descriptions is possible. In this situation, those descriptions should be read conjunctively. For example, value {*Can display card type; Card variant; Unique serial No*} means a user 'can read card type and card variant, and unique serial number'. In order to process the sentence descriptions in column *Purpose* more efficiently, we have identified a set of key-words used in all possibly purpose specifications. The sentence descriptions of *Purpose* of a test case are thus replaced by the combinations of these key-words, as shown in Table 4. Therefore, each

possible outcome identified in column *Purpose-key-word* can be treated as a set of values[1]. This enables set-based computations applicable.

Let V_{purp} be the set of all key-word collections used for describing *Purpose*, and let p be a key-word appeared in a given query Q, logically expressed as formula q, then interval set $\mathcal{B} = [\{p\}, \{p\}] = \{\{p\}\} \subseteq 2^{V_{purp}} \setminus \emptyset$ is called a *base interval set*. Further more, let u_i be a test case in $\overline{v}(q)$, and let $\mathcal{A}_i = Purpose{-}key{-}word(u_i)$ be the set of subsets of purpose key-word collections of u_i. Then sets $\overline{v}(q)_*$ and $\overline{v}(q)^*$ defined by the following two equations are referred to as the *tighter upper bound* and the *looser upper bound* of $\overline{v}(q)$ respectively,

$$\overline{v}(q)_* = Ret_*(Q_{\mathcal{B}}) \cup \underline{v}(q) = \{u_i \mid purpose{-}k{-}w(u_i) = \mathcal{A}_i, \mathcal{A}_i \supseteq_* \mathcal{B}\} \cup \underline{v}(q), \quad (10)$$

$$\overline{v}(q)^* = Ret^*(Q_{\mathcal{B}}) \cup \underline{v}(q) = \{u_i \mid purpose{-}k{-}w(u_i) = \mathcal{A}_i, \mathcal{A}_i \supseteq^* \mathcal{B}\} \cup \underline{v}(q). \quad (11)$$

It is observed that the purpose of a test case may not be defined and it is also possible that the purpose of a test case in $\underline{v}(q)$ can either not be defined or not contain key-word p. Therefore, we will have to union $\underline{v}(q)$ to the selected set. Also, subscript $\mathcal{B}$ of Q can be omitted is there is no confusion about which base interval set we refer to.

When a query Q involves several key-words and generates multiple base interval sets, $\mathcal{B}_1, ..., \mathcal{B}_j$, Eqs. (10) and (11) will be repeatedly applied to all base interval sets. As for any two base interval sets $\mathcal{B}_i$ and $\mathcal{B}_j$ defined from two distinct key-words, the effect of conjunction or disjunction of the key-words in a query will be reflected by the computation of joint tighter/looser bounds using the following equations:

$$\overline{v}(q_{\mathcal{B}_i and \mathcal{B}_j})_* = \overline{v}(q_{\mathcal{B}_i})_* \cap \overline{v}(q_{\mathcal{B}_j})_*, \qquad \overline{v}(q_{\mathcal{B}_i and \mathcal{B}_j})^* = \overline{v}(q_{\mathcal{B}_i})^* \cap \overline{v}(q_{\mathcal{B}_j})^*;$$
$$\overline{v}(q_{\mathcal{B}_i or \mathcal{B}_j})_* = \overline{v}(q_{\mathcal{B}_i})_* \cup \overline{v}(q_{\mathcal{B}_j})_*, \qquad \overline{v}(q_{\mathcal{B}_i or \mathcal{B}_j})^* = \overline{v}(q_{\mathcal{B}_i})^* \cup \overline{v}(q_{\mathcal{B}_j})^*.$$

Now looking back at query Q_3, if we assume $\mathcal{B}$ =[{Alarm}, {Alarm}]={{Alarm}}, and apply Eqs. (10) and (11) to $\overline{v}(\varphi_3)$, we get $\overline{v}(\varphi_3)_* = \{c_8, c_{10}\} \cup \{c_1, c_7\}$ and $\overline{v}(\varphi_3)^* = \{c_8, c_9, c_{10}\} \cup \{c_1, c_7\}$.

5 Conclusion

In this paper, we have presented novel approaches to coping with two common problems usually involved in a query: general concepts that are not explicitly defined in a data table and non-determinictic values among a set of possible choices. A logical based method is used to deal with the former while set based computations are applied to the latter.

The method in [5] is not applicable in test case selection problem, since there is a large number of attributes (24) involved in test case data table with thousands of records (test cases). It is not practical to re-generate the whole test case table every time a query is issued. The approach in [1] is also inadequate for this specific application because there are no user defined bounds available

[1] In fact, we rename the existing attribute *Purpose* as *Purpose-description* and add an additional attribute *Purpose-key-word*. In this way, we will be able to look at the detailed descriptions of test case purposes for those selected test cases.

to generate possible predicates. However, our mechanism is very similar to the knowledge representation schema in [12], where a tree is used to represent all the possible values of an attribute at different levels. Instead of using trees, we use meta-level tables to do the same job. Each meta-level table, equivalent to a tree in [12], can have more than two columns, with the most specific values in the far-left column and the most general values at the far-right. The manipulation and maintenance of these tables are almost identical to any data table in an information system, so there is very little extra work involved in building these meta-level tables.

Acknowledgments: I would like to thank Andrzej Skowron and Ivo Düntsch for their valuable comments on an earlier version of the paper, and Alfons Schuster for providing the telecommunication database. This project (Jigsaw) is jointly supported by the Nortel Networks and the IRTU, Northern Ireland.

References

1. Beaubouef, T. and Petry, F. E., (1995) Rough querying of crisp data in relational databases. In [4], 85-88.
2. Düntsch, I., (1997) A logic for rough sets. *Theoretical Computer Science* **179**(1-2), 427-236.
3. Düntsch, I. et al, (2001) Relational attribute systems, *International Journal of Human Computer Studies*, To appear.
4. Lin, T.Y. and Wildberger, A.M. (eds), (1995) *Soft Computing. Proceedings of Third International Workshop on Rough Sets and Soft Computing.* San Jose State University, USA, November 10-12.
5. Lin, T.Y., (1998) Granular computing on binary relations I: Data mining and neighborhood systems. In [10], 107-121.
6. Lipski, W.J., (1981) On databases with imcomplete information. *J. of the ACM*, Vol 28, 41-70
7. Liu, W., (2001) *Propositional, Probabilistic and Evidential Reasoning: integrating numerical and symbolic approaches.* to appear in *Studies in Fuzziness and Soft Computing* series. Springer-Verlag (Physica-Verlag).
8. Pagliani, P., (1998) A practical introduction to the model-relational approach to approximation spaces. In [10], 207-232.
9. Pawlak, Z., (1991) *Rough Sets. Theoretical Aspects of Reasoning about Data.* Kluwer Academic Publishers,
10. Polkowski, L. and Skowron, A., (ed.) (1998a) *Rough Sets in Knowledge Discovery 1: methodology and applications.* In *Studies in Fuzziness and Soft Computing* Vol 18. Springer-Verlag (Physica-Verlag).
11. Polkowski, L. and Skowron, A., (ed.) (1998b) *Rough Sets in Knowledge Discovery 2: applications, case studies and software systems.* In *Studies in Fuzziness and Soft Computing* Vol 19. Springer-Verlag (Physica-Verlag).
12. Ras, Z.W., (1998) Answering non-standard queries in distributed knowledge based systems. In [11], 98-108.
13. Yao, Y.Y. and Noroozi, N., (1995) A unified model for set-based computations. In [4], 252-255.