$\mathcal{RoSy}$: A Rough Knowledge Base System

Robin Andersson[1], Aida Vitória[2], Jan Małuszyński[3], and Jan Komorowski[1]

[1] The Linnaeus Centre for Bioinformatics,
Uppsala University, Box 598, SE-751 24 Uppsala, Sweden
{Robin.Andersson, Jan.Komorowski}@lcb.uu.se
[2] Dept. of Science and Technology,
Linköping University, SE-601 74 Norrköping, Sweden
aidvi@itn.liu.se
[3] Dept. of Computer and Information Science,
Linköping University, SE-581 83 Linköping, Sweden
janma@ida.liu.se

Abstract. This paper presents a user-oriented view of $\mathcal{RoSy}$, a $\mathcal{Ro}$ugh Knowledge Base $\mathcal{Sy}$stem. The system tackles two problems not fully answered by previous research: the ability to define rough sets in terms of other rough sets and incorporation of domain or expert knowledge. We describe two main components of $\mathcal{RoSy}$: knowledge base creation and query answering. The former allows the user to create a knowledge base of rough concepts and checks that the definitions do not cause what we will call a model failure. The latter gives the user a possibility to query rough concepts defined in the knowledge base. The features of $\mathcal{RoSy}$ are described using examples. The system is currently available on a web site for online interactions.

1 Introduction

The rough set framework [1] is relevant from the knowledge representation and data mining perspectives. The ability to handle vague and contradictory knowledge makes rough sets an important technique that can be incorporated in knowledge base systems. In addition, rough set methods can also be used to perform data exploration, which makes them relevant from a data mining point of view.

This paper presents a $\mathcal{Ro}$ugh Knowledge Base $\mathcal{Sy}$stem, called $\mathcal{RoSy}$. The system is accessible on the web page: `http://www.ida.liu.se/rkbs` .

$\mathcal{RoSy}$ tackles two problems not fully answered by previous research in the field of rough set theory. The first problem is related to defining rough sets in terms of other rough sets. For instance, we may wish to express that a rough set is obtained as a projection of another rough set over a subset of its attributes. The second problem deals with incorporation of domain or expert knowledge. A question arises of how concept approximations can be derived by taking into account not only the examples provided explicitly by one or more tables but also domain knowledge.

In $\mathcal{RoSy}$, the user can create a knowledge base of (non-recursive) vague concepts. Vague concepts are represented as implicitly or explicitly defined rough

D. Ślęzak et al. (Eds.): RSFDGrC 2005, LNAI 3642, pp. 48–58, 2005.

sets. Implicitly defined rough sets are obtained by combining different regions of other rough sets, e.g. lower approximations, upper approximations, and boundaries. The system also allows defining rough sets in terms of explicit examples. An important feature is that $\mathcal{RoSy}$ handles some basic numerical measures and that they can be used in the implicit definition of rough sets. The system detects whether the defined vague concepts do not introduce "conflicts" leading to a so-called model failure. Finally, a knowledge base of rough concepts can be queried. Hence, the system supports the user in reasoning about the defined rough concepts and in data exploration.

As we show in this paper through some examples[1], $\mathcal{RoSy}$ allows users to describe in an declarative and concise way solutions to problems that otherwise would require constructing specific programs in some language.

To our knowledge, besides $\mathcal{RoSy}$, only the *CAKE* system [2] addresses the problem of implicitly defining rough sets. There are two major differences between the two systems. Firstly, our system distinguishes tuples for which there is no information available from the tuples for which there is contradictory evidence. The latter case corresponds to tuples in the boundary region. *CAKE* does not support this distinction: the boundary region includes tuples about which there is no information at all and tuples about which there is contradictory information. Secondly, $\mathcal{RoSy}$ supports quantitative measures. This feature seems less obvious to achieve in *CAKE*. A more detailed comparison between both systems is included in [3].

2 A Knowledge Base System for Rough Sets

In this section we present the rough knowledge base system $\mathcal{RoSy}$. We start with presenting the online user interface of $\mathcal{RoSy}$. The notion of rough sets used in our framework and a proposed language for defining rough sets are presented in Section 2.2. Finally, a rough query language is discussed in Section 2.3. For a detailed and technical description of the system and its main components consult [4].

2.1 The Online User Interface

The online user interface of $\mathcal{RoSy}$ is a simple tab-navigation system, where each tab corresponds to a different mode. The main mode is **Compile Rules or Queries**, described below. Besides this mode, $\mathcal{RoSy}$ provides a mode for overviewing the defined rough knowledge base and a tool for query construction. Fig. 1 shows the main page of $\mathcal{RoSy}$'s web interface.

The first step in using the system (Fig. 1) is to create a rough knowledge base (RKB). An RKB can be typed directly in $\mathcal{RoSy}$ in the text area or loaded from local or remote files. The second step is to compile the RKB by clicking the **Compile Rules** button. Errors detected during compilation, such as syntax

[1] All examples presented in this paper are available from: `http://www.ida.liu.se/rkbs/examples.html` .

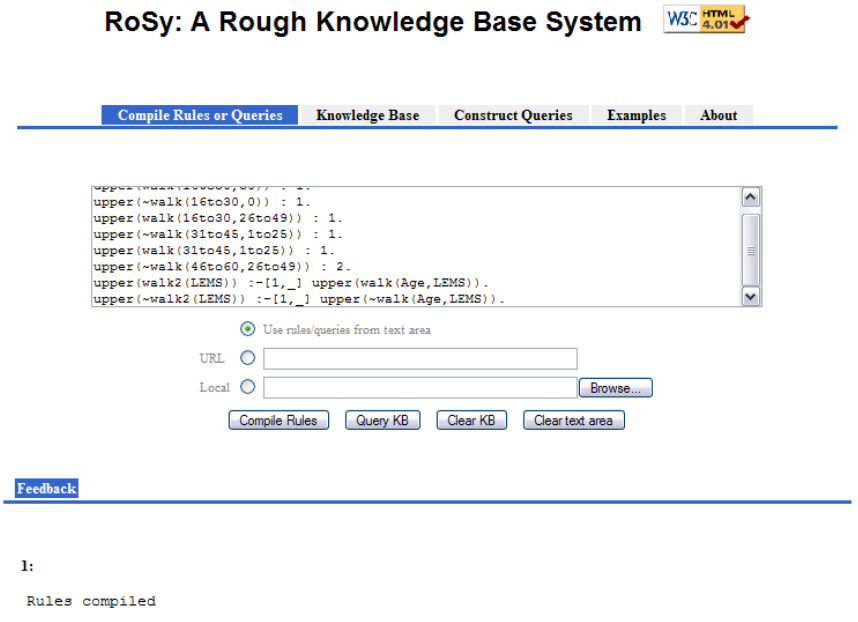

Fig. 1. $\mathcal{RoSy}$'s web interface, `http://www.ida.liu.se/rkbs`

errors, are displayed in the **Feedback** area. If the compilation procedure succeeds then the message *Rules compiled* is displayed in the feedback area. An RKB can be deleted from the system via the **Clear KB** button. The third step is to reason with the knowledge in the (compiled) RKB by querying it. Queries can be typed directly in the text area or loaded from local or remote files. Queries are evaluated via the **Query KB** button and answers to the queries are output in table format in the feedback area.

2.2 Defining Rough Relations in $\mathcal{RoSy}$

We start with presenting the notion of rough sets used in our framework [3, 5]. Consider an information system $\mathcal{I} = (U, A)$, in the sense of [6], where U is a universe of objects and A is a set of attributes. Every object in U is associated with a tuple of conditional attributes. We assume that such a tuple is the only way of referring to an object. Hence, different individuals described by the same tuple are indiscernible. For simplicity, we write t to designate a general tuple $\langle t_1, \ldots, t_n \rangle$. Let k be a positive integer. A pair $t : k$ is called a *supported tuple*, where k is called the *support*. Intuitively, $t : k$ represents k individuals which all have their conditional attribute values as indicated by the tuple t. A *rough set* (or *rough relation*[2]) S is a pair of sets $(\overline{S}, \overline{\neg S})$ of supported tuples, such that for any tuple t at most one supported tuple $t : k$ appears in each of the sets $\overline{S}$

[2] The expressions "rough set" and "rough relation" are used interchangeably in this paper.

and $\overline{\neg S}$[3]. An element $t : k \in \overline{S}$ or $t : k \in \overline{\neg S}$ indicates that the indiscernibility class described by t belongs to the *upper approximation* of a rough set S or $\neg S$, respectively, and that this class contains $k > 0$ individuals that are positive examples of the concept described by S or $\neg S$.

The *lower approximation* and the *boundary region* of a rough set S is defined as:

$$\underline{S} = \{t : k_1 \in \overline{S} \,|\, \forall\, k_2 > 0,\, t : k_2 \notin \overline{\neg S}\} \tag{1}$$

$$\overline{\underline{S}} = \{t : k_1 : k_2 \,|\, \exists\, k_1, k_2 > 0,\, t : k_1 \in \overline{S} \text{ and } t : k_2 \in \overline{\neg S}\} \tag{2}$$

A *decision table* $\mathcal{D} = (U,\, A = \{a_1, \ldots, a_n\}, \{d\})$, where $d \notin A$ is a binary decision attribute, represents a rough set $D = (\overline{D}, \overline{\neg D})$. Any decision table can be encoded by *rough facts* that describe the rows in the table. A rough fact describes a tuple in the upper approximation of a rough relation D and is in one of the following forms:

$$\texttt{upper(d(t}_1\texttt{,...,t}_n\texttt{)) : k.} \tag{3}$$

$$\texttt{upper(}\sim\texttt{d(t}_1\texttt{,...,t}_n\texttt{)) : k.} \tag{4}$$

where the predicate symbols `d` and $\sim$`d` denote the positive and negative outcome of the decision attribute respectively, each $\texttt{t}_\texttt{i}$ $(1 \leq \texttt{i} \leq \texttt{n})$ is an attribute value corresponding to attribute a_i, and `k` denotes the support of the tuple.

As an example, consider decision table $\mathcal{W}$alk [6] and its corresponding rough facts encoding rough relation *Walk* in Table 1.

Table 1. Decision table $\mathcal{W}$alk and its corresponding collection of rough facts defining rough relation *Walk*

	Age	LEMS	Walk	
o1	16to30	50	Yes	⇒ `upper(walk(16to30,50)) : 1.`
o2	16to30	0	No	⇒ `upper(`$\sim$`walk(16to30,0)) : 1.`
o3	16to30	26to49	Yes	⇒ `upper(walk(16to30,26to49)) : 1.`
o4	31to45	1to25	No	⇒ `upper(`$\sim$`walk(31to45,1to25)) : 1.`
o5	31to45	1to25	Yes	⇒ `upper(walk(31to45,1to25)) : 1.`
o6	46to60	26to49	No	⇒ `upper(`$\sim$`walk(46to60,26to49)) : 2.`
o7	46to60	26to49	No	

Note that objects o6 and o7 are indiscernible and they belong to the same decision class, which yields the support 2. From Table 1 it is easy to see that:

$$\underline{Walk} = \{\langle \texttt{16to30,50} \rangle \texttt{ : 1},\ \langle \texttt{16to30,26to49} \rangle \texttt{ : 1}\} \tag{5}$$

$$\underline{\neg Walk} = \{\langle \texttt{16to30,0} \rangle \texttt{ : 1},\ \langle \texttt{46to60,26to49} \rangle \texttt{ : 2}\} \tag{6}$$

$$\overline{\underline{Walk}} = \{\langle \texttt{31to45,1to25} \rangle \texttt{ : 1 : 1}\} \tag{7}$$

[3] Intuitively, this restriction requires that the support of each tuple t is specified only once, rather than as the sum of different supports.

Rough facts explicitly define the upper approximation of a rough relation. However, the lower approximation and the boundary region of that relation are implicitly derived from (1) and (2). $\mathcal{RoSy}$ also allows specific regions of rough relations to be defined implicitly by regions of other rough relations. This can be accomplished by *rough clauses*.

Consider again decision table $\mathcal{W}$alk in Table 1. Its corresponding rough relation *Walk* can be used to define a new rough relation *Walk2*, which corresponds to *Walk* but ignores the attribute `Age`. The following rough clauses define *Walk2*:

```
upper(walk2(LEMS)) :-[1,_] upper(walk(Age,LEMS)).        (8)
upper(∼walk2(LEMS)) :-[1,_] upper(∼walk(Age,LEMS)).      (9)
```

A rough clause is an implication on the general form *Head* ← *Body*, e.g. `upper(walk2(LEMS))` ← `upper(walk(Age,LEMS))`., meaning that the tuples of $\overline{Walk}$ are defined to be members of $\overline{Walk2}$. The body and head of clause (8) contain *rough literals* that denote upper approximations of rough relations. A head rough literal refers to the rough relation being defined and can denote either the upper or lower approximation of that relation, while the body can contain one or more rough literals that can also denote boundary regions. A general rough literal is on the form `reg(p(`$\mathtt{T}_1$`,...,`$\mathtt{T}_n$`))` , where `p` (possibly negated) denotes a rough relation P, `reg` ∈ {`upper`, `lower`, `boundary`}, and $\mathtt{T}_i$ ($1 \leq i \leq n$) denotes an attribute term representing attribute i in that relation. *Attribute terms* can either be constants, starting with lower-case letters or digits, or variables, starting with upper-case letters. A constant denotes a specific attribute value, e.g. `16to30`, while a variable can denote any attribute value in the domain of the corresponding attribute, e.g. variable `Age` above can denote any of the values `16to30`, `31to45`, or `46to60`.

Besides rough literals, certain *quantitative measures*, such as support (`supp`), accuracy (`acc`), coverage (`cov`), and strength (`strength`) can optionally be included in the body as constraints. All variables occurring as their arguments should also appear in some rough literal in the body.

The body and the head of a rough clause is separated by the implication operator `:-[`τ`,F]`. `F` is the support-combining function `sum`, `min` or `max`, that specifies how the support of the newly defined rough relation is obtained from the support of the rough relations in the body of the clause. If the body only has one rough literal then `F` is auxiliary and often set to `_` (`[`τ`,_]`). The constant $\tau \in [0, 1]$ (often set to 1) is a rational number representing the *trust* in the body of the clause. The trust is the fraction of the calculated support of the body that should be considered as support for the rough region being defined.

A rough clause defining the lower approximation of a rough relation must fulfill condition (1). If the user tries to define a rough relation P such that a tuple $t \in \underline{P}$ and $t \in \overline{\neg P}$ then, $\mathcal{RoSy}$ reports a *model failure* [4], the current compilation stops, and the definition is retracted. For more details see [4].

2.3 Querying $\mathcal{RoSy}$

$\mathcal{RoSy}$ provides a rough query language for retrieving information about defined rough relations. For instance, it may be desirable to monitor the changes in the regions of *Walk* when excluding the attribute `Age`. Such information may be retrieved by, for example, asking the following rough query.

```
boundary(walk2(LEMS)), K1 = supp(walk2(LEMS)), K2 = supp(~walk2(LEMS)).
```

The above compound query asks for all the tuples in the boundary region of *Walk2* and their support. In other words, the query requests all instantiations of the variables `LEMS`, `K1` and `K2` for the tuples in the boundary of *Walk2*. The received answer is given below.

K1	K2	LEMS
1	1	1to25
1	2	26to49

The answer states that tuples $\langle$`1to25`$\rangle$ `: 1 : 1` and $\langle$`26to49`$\rangle$ `: 1 : 2` are members of $\underline{\overline{Walk2}}$.

A rough query can be any combination of rough literals and quantitative measures (constraints or assignments). Moreover, $\mathcal{RoSy}$ provides a classification procedure written on the form `K = classify(p(`$\mathtt{T}_1$`,...,`$\mathtt{T}_n$`))`, where `K` denotes a variable to be instantiated with the classification results, `p` denotes some (defined) rough relation P, and $\mathtt{T}_i$ $(1 \leq i \leq n)$ denotes an attribute term representing attribute i of P. The classification query requests a prediction for the decision class to which a new individual described by tuple $\langle \mathtt{T}_1, \ldots, \mathtt{T}_n \rangle$ may belong. The answer to such a query is either `yes`, `no` or `unknown` together with the certainty of the prediction. For more details see [3].

A rough query can either be ground (all attribute terms are constants) or non-ground (some attribute terms are variables). If a query is ground then it requests the truth value, yes or no, of the query. If the query on the other hand is non-ground then it requests all instantiations of the attribute variables in the query that make it true.

3 A Medical Informatics Application in $\mathcal{RoSy}$

We now illustrate the use of $\mathcal{RoSy}$ with a data set that previously was analyzed with classical rough sets in [7].

Study [8] has shown that the single most important independent predictor for future hard cardiac events (cardiac death or non-fatal myocardial infarction) is an abnormal scintigraphic scan pattern. However, performing such a test is expensive, and may sometimes be redundant with respect to making a prognosis. It is therefore desirable to identify patients who are in need of a scintigraphic scan and avoid it for patients who can be prognosticated without such a test.

Table 2 describes information about patients observed in [8][4]. A group of 417 patients has been examined and each patient has an associated set of medical

[4] The number of attributes were reduced by Komorowski and Øhrn in [7] from the original data in [8].

Table 2. Attribute definitions

Attribute	Definition
Age	> 70 years old
Oldmi	Prior Infarction
Hypert	Hypertension
Dm	Diabetes
Smok	Smoking
Chol	Hypercholesterolemia
Gender	Male or female
Hfmed	History of dec. cordis
Angp	History of angina
Apstress	Angina during stress
Stt	Level of ST-T changes
Scanabn	Abnormal scintigraphic scan
Deathmi	Cardiac death or infarction

information. Attribute `Deathmi` is the decision attribute. The data is captured by rough relation *Deathmi* that is encoded by 335 rough facts.

Following the approach suggested in [7], identification of the patients who are in need of a scintigraphic scan requires monitoring changes in $\overline{\underline{Deathmi}}$ when considering only the set of attributes $A \setminus \{\texttt{Scanabn}\}$, i.e. removing the attribute `Scanabn`. Next, we show how the problem can be formulated and solved in $\mathcal{RoSy}$. A detailed comparison between our approach and the approach by [7] is outside the limits of this paper, but can be found in [3].

3.1 Avoiding the Expensive Test

The knowledge of a scintigraphic scan outcome is strictly required for the patients for whom excluding conditional attribute `Scanabn` causes migration into the boundary region from either $\underline{Deathmi}$ or $\underline{\neg Deathmi}$. In the following rough clauses in Fig. 2, we shorten the sequence of attributes for readability reasons, e.g. we write `Age,...,Scanabn` instead of `Age,Oldmi,Hypert,Dm,Smok,Chol,Gender,Hfmed,Angp,Apstress,Stt,Scanabn`[5].

First, the set of attributes is reduced to not include `Scanabn`. The new rough relations D and its explicit negation $\neg D$ are defined by *Deathmi*, ignoring the last attribute, through rough clauses (10) and (11). The set of patients migrating into the boundary region of D from either $\underline{Deathmi}$ or $\underline{\neg Deathmi}$ corresponds to the rough relation *Migrate* defined by (12) and (13). These clauses state that the set of migrating individuals are the patients who are members of $\underline{Deathmi}$ (clause (12)) or $\underline{\neg Deathmi}$ (clause (13)) and also members of $\overline{\underline{D}}$. If a patient is captured by either of these rules, one cannot make a reliable prognosis of future cardiac events without including the knowledge of the outcome of a scintigraphic scan. Rough clauses (14), (15) and (16) capture the set of non-migrating patients. Clause (14) captures those patients who were originally in the boundary region of

[5] This kind of abbreviation is, however, not allowed in the $\mathcal{RoSy}$ system.

```
upper(d(Age,...,Stt)) :-[1,_] upper(deathmi(Age,...,Stt,Scanabn)).          (10)
upper(∼d(Age,...,Stt)) :-[1,_] upper(∼deathmi(Age,...,Stt,Scanabn)).       (11)
upper(migrate(Age,...,Stt)) :-[1,min] boundary(d(Age,...,Stt)),            (12)
      lower(deathmi(Age,...,Stt,Scanabn)).
upper(migrate(Age,...,Stt)) :-[1,min] boundary(d(Age,...,Stt)),            (13)
      lower(∼deathmi(Age,...,Stt,Scanabn)).
upper(∼migrate(Age,...,Stt)) :-[1,sum]                                     (14)
      upper(deathmi(Age,...,Stt,Scanabn)),
      upper(∼deathmi(Age,...,Stt,Scanabn)).
upper(∼migrate(Age,...,Stt)) :-[1,_] lower(d(Age,...,Stt)).                (15)
upper(∼migrate(Age,...,Stt)) :-[1,_] lower(∼d(Age,...,Stt)).               (16)
```

Fig. 2. A rough program for identification of migrating patients

Deathmi. These patients obviously remain in the boundary region after removing the `Scanabn` attribute. `sum` is used for combining the support of those tuples t such that $t : k_1 \in \overline{Deathmi}$ and $t : k_2 \in \overline{\neg Deathmi}$, into $sum(k_1, k_2) = k_1 + k_2$, since $k_1 + k_2$ is the total number of individuals in the indiscernibility class t.

We want to know for which patients the scintigraphic scan test is needed for a reliable prognosis. The answer to the following query is given in Table 3.

```
upper(migrate(Age,...,Stt)),
K1 = strength(migrate(Age,...,Stt)),
K2 = strength(∼migrate(Age,...,Stt)).
```

For which patients is it useful to request the scintigraphic scan and what is the percentage of patients for whom the test is needed?

Table 3. Migrating patients

K1	K2	Age	Oldmi	Hypert	Dm	Smok	Chol	Gender	Hfmed	Angp	Apstress	Stt
0.0024	0.0073	0	0	0	0	0	0	0	0	0	0	0
0.0024	0.0049	0	0	0	0	0	0	1	1	1	2	0
0.0049	0.0000	0	1	0	0	0	0	1	0	0	0	0

For any tuple t, `strength(migrate(t))` (`strength(¬migrate(t))`) represents the proportion of patients of the universe who belong to the indiscernibility class described by t and have positive (negative) outcome for `migrate`. Hence, the percentage of patients for whom the scintigraphic test is needed can be computed by the formula

$$\Phi = 100 \cdot \sum_{t \in \overline{\texttt{Migrate}}} (\texttt{strength}(\texttt{migrate}(t)) + \texttt{strength}(\neg\texttt{migrate}(t))).$$

```
upper(deathmiApprox(Attrs)) :-[1,_] lower(deathmi(Attrs)).                    (17)
upper(∼deathmiApprox(Attrs)) :-[1,_] lower(∼deathmi(Attrs)).                  (18)
lower(deathmiApprox(Attrs)) :-[1,sum] upper(deathmi(Attrs)),                  (19)
      upper(∼deathmi(Attrs)), acc(deathmi(Attrs)) >= 0.7.
lower(∼deathmiApprox(Attrs)) :-[1,sum] upper(deathmi(Attrs)),                 (20)
      upper(∼deathmi(Attrs)), acc(deathmi(Attrs)) =< 0.3.
upper(deathmiapprox(Attrs)) :-[1,_] upper(deathmi(Attrs)),                    (21)
      acc(deathmi(Attrs)) > 0.3, acc(deathmi(Attrs)) < 0.7.
upper(∼deathmiapprox(Attrs)) :-[1,_] upper(∼deathmi(Attrs)),                  (22)
      acc(deathmi(Attrs)) > 0.3, acc(deathmi(Attrs)) < 0.7.
```

Fig. 3. A rough program approximating *Deathmi*

The answer indicates that if only the migrating patients, given by Table 3, undergo the expensive scintigraphic scan, then one may expect to avoid the test for approximately 98% of all the patients. Non-migrating patients who are members of $\underline{\overline{D}}$ cannot be reliably prognosticated. For these patients it may still be needed to perform the scintigraphic scan procedure, if that is the opinion of a medical expert.

3.2 VPRSM in $\mathcal{R}$o$\mathcal{S}$y

Quantitative measures in the body of rough clauses can be used as constraints to build more generalized rough approximations of a relation in the same spirit as with *precision control parameters* in the *variable precision rough set model* (VPRSM) [9], as discussed in [5]. This means that it is possible to define new rough relations by other ones stating that a certain constraint must be fulfilled. The new rough relation *DeathmiApprox*, see Fig. 3, can be defined by the boundary region of *Deathmi* and the constraint stating that the accuracy of *Deathmi* should be above a certain threshold, say 70% (clause (19)). ¬*DeathmiApprox* is then defined by the boundary region of *Deathmi* and the constraint stating that the accuracy of *Deathmi* should be below 30% (clause (20)). In the rough clauses of Fig. 3, we write `Attrs` to denote the sequence `Age,Oldmi,Hypert,Dm,Smok, Chol,Gender,Hfmed,Angp,Apstress,Stt,Scanabn`[6].

Rough clauses (17) and (18) state that the indiscernibility classes in $\underline{Deathmi}$ and $\underline{\neg Deathmi}$ are members of the upper approximation of respective approximate rough relation. The decision rules corresponding to these indiscernibility classes have 100% accuracies and are therefore included in the same region of the new rough relation. Rough clauses (21) and (22) state that any indiscernibility class t in the boundary such that `0.3 < acc(deathmi(t)) < 0.7` remains in the boundary.

[6] As previously, such a notation is used here only for readability reasons and is not allowed in the $\mathcal{R}$o$\mathcal{S}$y system.

To see that the previously defined *DeathmiApprox* in fact approximates the rough relation *Deathmi* one can ask the following query to $\mathcal{RoSy}$.

```
lower(∼deathmiApprox(Age,...,Scanabn)),
upper(deathmi(Age,...,Scanabn)),
K1 = acc(∼deathmi(Age,...,Scanabn)),
K2 = acc(deathmi(Age,...,Scanabn)).
```

Which indiscernibility classes are both in $\underline{\neg DeathmiApprox}$ and in $\overline{Deathmi}$, and what are the corresponding accuracies?

The answer, given in Table 4, shows that two indiscernibility classes of *Deathmi* are members of the concept $\neg DeathmiApprox$. Since $\texttt{K2} \leq 0.3$, by rough clause (20), those classes are included in $\underline{\neg DeathmiApprox}$.

Table 4. Patients who are members of $\underline{\neg DeathmiApprox}$ and $\overline{Deathmi}$

K1	K2	Age	Oldmi	Hypert	Dm	Smok	Chol	Gender	Hfmed	Angp	Apstress	Stt	Scanabn
0.8000	0.2000	0	1	0	0	0	0	1	0	0	1	0	1
0.8000	0.2000	0	1	1	0	0	0	1	0	0	0	0	1

4 Conclusions

This paper presents an overview of a system, called $\mathcal{RoSy}$, that allows users to create knowledge bases of vague concepts. The main novel aspect of this system is that concepts can be represented by intensionally defined rough relations.

The main strengths of $\mathcal{RoSy}$ can be summarized as follows.

- $\mathcal{RoSy}$ makes it possible to capture and to integrate in a uniform way vague knowledge obtained directly from experimental data and encoded as rough facts with domain or expert knowledge expressed as rough clauses. This contrasts with most of current rough set techniques that only allow definition of (vague) concepts to be obtained from experimental data.
- The expressive power of $\mathcal{RoSy}$ makes it possible to formulate in the same language several useful techniques and extensions to rough sets reported in the literature, such as [7, 9]. Section 3 illustrates this point with two concrete examples.

Another important aspect of $\mathcal{RoSy}$ is the possibility of using queries to retrieve information about the defined rough sets and patterns implicit in the data.

The functionalities of $\mathcal{RoSy}$ can be improved in several ways. Firstly, $\mathcal{RoSy}$ does not handle recursive rough programs. This extension requires compilation of rough programs to extended logic programs whose semantics is captured by

paraconsistent stable models [3]. Secondly, the query language may need extensions. For instance, assume that `danger(Roadcond, Speed)` is a rough concept indicating whether a traffic situation is dangerous in terms of road conditions (e.g. `ice`, `wet`) and vehicle speed (e.g. `high`, `medium`). The user may want to find out whether for an icy road, it is definitely dangerous to drive at any speed (based on this information a road may then be temporarily closed for safety reasons). The actual query capabilities of $\mathcal{RoSy}$ do not allow a direct formulation of this query. Thirdly, $\mathcal{RoSy}$ may be extended to find candidate hypothesis that explain observed facts, given a rough knowledge base. In this way we could combine abductive reasoning with rough relations. A possible application is reasoning about incomplete networks of chemical reactions represented in a rough knowledge base. Exploration of $\mathcal{RoSy}$ on a wide range of applications is crucial for further demonstration of its usefulness and for making relevant improvements. Among others, rough mereology [10] applications seem to be promising in that respect.

References

[1] Pawlak, Z.: Rough sets. International Journal of Computer and Information Sciences **11** (1982) 341–356
[2] Doherty, P., Łukaszewicz, W., Szałas, A.: CAKE: A computer-aided knowledge engineering technique. Proceedings of the 15th European Conference on Artificial Intelligence ECAI'2002 (2002) 220–224
[3] Vitória, A.: A framework for reasoning with rough sets. Licentiate thesis, Linköping University, Dept. of Science and Technology (2005) LiU-TEK-LIC-2004:73, Thesis No. 1144.
[4] Andersson, R.: Implementation of a rough knowledge base system supporting quantitative measures. Master's thesis, Linköping University (2004)
[5] Vitória, A., Damásio, C.V., Małuszyński, J.: Toward rough knowledge bases with quantitative measures. In Tsumoto, S., Slowinski, R., Komorowski, H.J., Grzymala-Busse, J.W., eds.: Rough Sets and Current Trends in Computing. Volume 3066 of Lecture Notes in Computer Science., Springer (2004) 153–158
[6] Komorowski, J., Pawlak, Z., Polkowski, L., Skowron, A.: Rough sets: A tutorial. In Pal, S., Skowron, A., eds.: Rough-Fuzzy Hybridization: A New Method for Decision Making. Springer Verlag, Singapore (1998) 3–98
[7] Komorowski, J., Øhrn, A.: Modelling prognostic power of cardiac tests using rough sets. Artificial Intelligence in Medicine **15** (1999)
[8] Geleijnse, M., Elhendy, A., van Domburg, R.: Prognostic value of dobutamine-atropine stress technetium-99m sestamibi perfusion scintigraphy in patients with chest pain. J Am Coll Cardiol **28** (1996) 447–454
[9] Ziarko, W.: Variable precision rough set model. Journal of Computer and Systems Science **46** (1993) 39–59
[10] Polkowski, L., Skowron, A.: Rough mereological calculi of granules: A rough set approach to computation. Journal of Computational Intelligence **17** (2001) 472–492