Information Sciences 593 (2022) 546–576

Contents lists available at ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

A landscape and implementation framework for probabilistic rough sets using PROBLOG

Patrick Doherty[a,b,*,1], Andrzej Szałas[c,b,2]

[a] School of Intelligent Systems and Engineering, Jinan University (Zhuhai Campus), Zhuhai, China
[b] Department of Computer and Information Science, Linköping University, SE-581 83 Linköping, Sweden
[c] Institute of Informatics, University of Warsaw, Banacha 2, 02-097 Warsaw, Poland

A R T I C L E I N F O

Article history:
Received 18 April 2021
Received in revised form 13 December 2021
Accepted 16 December 2021
Available online 29 December 2021

Keywords:
Probabilistic rough sets
ProbLog
Probabilistic programming
Knowledge representation
Rule-based languages

A B S T R A C T

Reasoning about uncertainty is one of the main cornerstones of Knowledge Representation. More recently, combining logic with probability has been of major interest. Rough set methods have been proposed for modeling incompleteness and imprecision based on indiscernibility and its generalizations and there is a large body of work in this direction. More recently, the classical theory has been generalized to include probabilistic rough set methods of which there are also a great variety of proposals. Pragmatic, easily accessible, and easy to use tools for specification and reasoning with this wide variety of methods is lacking. It is the purpose of this paper to fill in that gap where the focus will be on probabilistic rough set methods. A landscape of (probabilistic) rough set reasoning methods and the variety of choices involved in specifying them is surveyed first. While doing this, an abstract generalization of all the considered approaches is derived which subsumes each of the methods. One then shows how, via this generalization, one can specify and reason about any of these methods using PROBLOG, a popular and widely used probabilistic logic programming language based on PROLOG. The paper also considers new techniques in this context such as the use of probabilistic target sets when defining rough sets and the use of partially specified base relations that are also probabilistic. Additionally, probabilistic approaches using tolerance spaces are proposed. The paper includes a rich set of examples and provides a framework based on a library of generic PROBLOG relations that make specification of any of these methods, straightforward, efficient and compact. Complete, ready to run PROBLOG code is included in the Appendix for all examples considered.

1. Introduction

1.1. Probabilistic rough sets

Formal theories for approximate reasoning are numerous and highly varied due to different types of approximate concepts intended to be modeled such as vagueness, imprecision and incompleteness. Rough set theory [2,7,19,25,38,47,48]

* Corresponding author at: Department of Computer and Information Science Linköping University, SE-581 83 Linköping, Sweden.
E-mail addresses: patrick.doherty@liu.se (P. Doherty), andrzej.szalas@liu.se (A. Szałas).

[1] This work has been supported by the ELLIIT Network Organization for Information and Communication Technology, Sweden; the Swedish Foundation for Strategic Research SSF (Smart Systems Project RIT15-0097); a (guest) professor grant from Jinan University and an RExperts Program Grant 2020A1313030098 from the Guangdong Department of Science and Technology, China.

[2] This work has been supported by grant 2017/27/B/ST6/02018 of the National Science Centre Poland.

https://doi.org/10.1016/j.ins.2021.12.062
0020-0255/© 2021 The Author(s). Published by Elsevier Inc.

is one such formal representation and it has been used to model incompleteness and imprecision using indiscernibility relations and approximations based on such relations. Classical rough set theory has been generalized to use *base* relations weaker than equivalence relations [7,8,38]. For instance, tolerance spaces [23,39] are defined in terms of a base relation that is only reflexive and symmetric. Rough sets are additionally characterized by three regions, a positive region containing all individuals that are a member of a set, a negative region containing all individuals that are not in the set, and a boundary region containing elements that may or may not be in the set. For each rough set, these regions are determined by the base relation used in defining that particular set.

In [9], the authors show how Answer Set Programming (ASP) [10] can be used as a basis for representing and reasoning about classical rough relations and their generalizations together with standard relations in a nonmonotonic context. ASP is a knowledge representation framework based on the logic programming and nonmonotonic reasoning paradigms that uses an answer set/stable model semantics for logic programs. There are many other implementations of libraries and tools for rough sets in general (see [17,30,32,33] and references there). However, to the best of our knowledge, no implementation of probabilistic approaches have been reported in the literature.

Early in the development of rough set theory and its applications, it was noticed that probabilistic information implicit in rough set theory was not explicitly formalized and exploited [21]. A natural extension of the rough set method that uses a probabilistic model was proposed in [43]. Since then there have been many additional proposals and applications of probabilistic rough set methods [12,13,18,37,44–50].

In the standard rough set model, given a crisp set S, the lower and upper approximations of S are defined based on an underlying equivalence relation. If an equivalence class is a subset of S, then all individuals in that class are in the lower approximation of S. If an equivalence class intersects with the set S, then that equivalence class is part of the upper approximation for S.

The basic intuition underlying probabilistic rough set methods is based on the observation that for those equivalence classes in the upper approximation of a set, the proportion of individuals in that class that are in the set in comparison to those that are not, offers a quantitative measure for any randomly chosen individual in the equivalence class being a member of the set S. One can then define a membership function $\mu(x)$ which returns this ratio for any individual. $\mu(x)$, together with the use of threshold values, can be used for more fine-grain assessments of which individuals are in the boundary region of a rough set or not.

1.2. Motivations and contributions

Given the widespread use of probabilistic rough set methods, the purpose of this paper is to show how PROBLOG [4–6], a well-known probabilistic logic programming language, can be used as a basis for representing and reasoning about probabilistic rough relations and their generalizations in a succinct and elegant manner. In fact, the paper will show how a majority of the more well-known probabilistic rough set methods can be generalized and defined using one parameterized definition (Definition 8.4) that provides the basis for representing and reasoning with any of these methods using PROBLOG in a straightforward manner. By using higher-order PROBLOG programming techniques, a generic set of PROBLOG relations can be defined and instantiated for specifying any of the probabilistic rough set methods considered in the paper. This set of relations can be defined in roughly half a page of PROBLOG (see Program 6).

The paper also proposes a number of new generalizations of existing probabilistic rough set methods not seen in the literature. Normally, rough set methods assume a target set S that is a crisp set not definable with the elementary equivalence classes associated with an approximation space. In the first generalization, the paper proposes the use of both crisp sets and probabilistic sets as input. In the second generalization, the paper proposes the use of probabilistic base relations as a basis for approximation spaces. These additional generalizations are also subsumed by the parameterized Definition 8.4 for generalized approximation operators. The paper uses a rich set of PROBLOG examples and reproducible case studies to show the efficacy of the approaches that are proposed.

Fig. 1 provides an incremental roadmap of the variety of rough set approximation operators considered in the paper. The lower lane begins with classical approximation spaces and operators and provides definitions of the various generalizations in the literature. These incremental generalizations lead to Definition 8.4, a definition for generalized probabilistic approximation spaces and operators that can be instantiated to generate the preceding incremental generalizations. The upper lane in the roadmap begins with classical tolerance-based approximation spaces and operators that use tolerance spaces where the base relation is only reflexive and symmetric. This is generalized to the probabilistic case which then again leads to Definition 8.4 which subsumes this lane also.

The following contributions are considered in this paper:

1. The paper begins with a thorough summary of rough set methods in the literature and their generalization to probabilistic rough set methods. In doing this, the variety of choices used to define different rough set theories, both non-probabilistic and probabilistic, are brought to light. A number of definitions for existing and new probabilistic rough set methods are considered that incrementally generalize each other as reflected in the definition roadmap in Fig. 1.

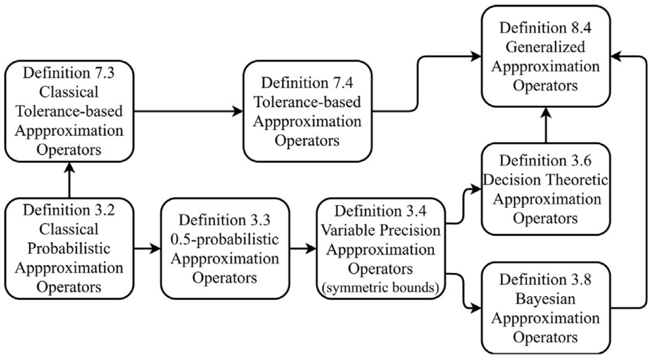
Definition 8.4 Generalized Appproximation Operators
Definition 7.3 Classical Tolerance-based Appproximation Operators
Definition 7.4 Tolerance-based Appproximation Operators
Definition 3.6 Decision Theoretic Appproximation Operators
Definition 3.2 Classical Probabilistic Appproximation Operators
Definition 3.3 0.5-probabilistic Appproximation Operators
Definition 3.4 Variable Precision Appproximation Operators (symmetric bounds)
Definition 3.8 Bayesian Appproximation Operators

Fig. 1. Roadmap of definitions for probabilistic approximation operators.

2. Definitions of *generalized probabilistic approximation spaces and operators* are provided. These parameterized definitions essentially subsume a majority of probabilistic rough set methods considered in the literature. This in itself is not surprising since the methods are closely related, but it does enable compact, efficient and highly expressive specification mechanisms in PROBLOG.
3. As alluded to in the previous point, using the definitions for generalized probabilistic approximation spaces and operators as a basis, an efficient and compact encoding for a majority of probabilistic rough set methods encountered in the literature is proposed. It is based on parameterizing both the formal definitions and then using a set of higher-order relational definitions in PROBLOG that correlate with different parameterizations.
4. A number of new concepts are introduced that enhance the expressibility and use of existing probabilistic rough set methods.
   - The input to a majority of rough set methods is a crisp target set in a domain, generally not definable using the elementary classes in the quotient set generated by a base relation in the approximation space in question. One then proceeds to define its lower and upper approximations. The first generalization allows for partially specified, probabilistic target sets as input to probabilistic rough set methods.
   - The base relations used in a majority of rough set methods are non-probabilistic. The second generalization is to allow base relations to be defined as probabilistic relations.

For both these generalizations to be useful, one has to define a probabilistic neighborhood relation based on the use of thresholds in order to generate useful equivalence classes or coverings for a generalized approximation space. This is an additional contribution.

5. Tolerance spaces provide a particularly interesting challenge in terms of generalizing such spaces to their probabilistic counterparts. This is because individuals in a domain can be in more than one neighborhood at the same time. The paper identifies why this is a problem for the probabilistic case and then provides a solution in terms of a modified definition for lower and upper probabilistic approximations.
6. A PROBLOG programming methodology is provided for specifying and reasoning about almost any type of probabilistic rough set method based on the instantiation of a collection of generic higher-order PROBLOG relations described previously. The resulting probabilistic rough relations can be combined with non-probabilistic and probabilistic relations defined in the conventional manner using PROBLOG's features.
7. A number of examples and case studies are provided throughout, showing the efficacy, expressivity and power of these concepts. The intent is that the PROBLOG based tool proposed for specifying and reasoning about the variety of probabilistic rough set methods can be easily usable and reproducible in a straightforward manner. To show this, out-of-the-box executable PROBLOG code is provided for a majority of the examples and case studies to guide the potential user of these techniques.

All the above points contain original results either by synthesizing concepts previously introduced and investigated by other authors (points 1, 2) or by reporting original developments (points 3–7).

1.3. Feasibility of the approach

The worst-case complexity of computing PROBLOG queries is exponential wrt the number of variables/ground literals. The same applies to other probabilistic logic programming languages using the Sato distribution semantics [34]. Though still exponential in the worst case, the techniques used to compute probabilistic queries in PROBLOG exhibit a much better performance pragmatically (see also a discussion in Remark 4.1). The PROBLOG solver essentially combines SLD-resolution with approximation algorithms and methods for representing Boolean formulas and computing their probabilities. As indicated in [6], PROBLOG's solver is able to deal with belief bases sometimes containing up to 100000 formulas.

In fact, PROBLOG has been used in a variety of real-world applications. For example, the PROBLOG web page [28], indicates over 20 papers concerning larger scale applications in biology, actions theories, robotics and tracking, games, natural languages, as well as in other areas.

The specification framework developed in the current paper is built on top of the PROBLOG engine. It only adds polynomial time complexity, so it can be applied in the areas mentioned in addition to other areas. Our framework inherits the methodology and applications of probabilistic rough sets, so it can be particularly useful when objects can be clustered and gathered in equivalence classes or tolerance neighborhoods reflecting indiscernibilities/similarities among objects. This typically occurs and is described in previously cited sources on probabilistic rough sets. It may also occur in the application areas indicated in [28]. In [6], similarity relations are used in link mining for large networks of biological concepts. A typical query considers whether a given gene is connected to a given disease. As indicated in [6], "probabilities of edges can be obtained from methods that predict their existence based on, e.g., co-occurrence frequencies or sequence similarities". The study shows that "the connection query could be solved [...] for graphs with up to 1 400 to 4 600 edges".

We also demonstrate the use of indiscernibility and similarity relations in the case study discussed in Section 9. There the case study is simplified for the purpose of presentation and pedagogic clarity. It is intended to serve as a showcase illustrating the underlying methodology and features described in the paper.

1.4. Paper structure

Section 1 provides an introduction and overview of the topic area and the approach taken in the construction of a generic PROBLOG based tool for probabilistic rough set methods. Section 2 provides an overview of classical rough set methods and a landscape of choices that can be made when defining the underlying base relations for rough sets. Section 3 provides an overview of probabilistic rough set methods in the literature. Additionally, it considers an example of the application of the most basic probabilistic rough set method where information necessary for defining rough sets is generated from a table of data. Section 4 provides a short summary of PROBLOG and its semantic theory which is based on distributional semantics. Section 5 provides a generic PROBLOG program structure and template for specifying probabilistic rough sets using informal, but concise syntax. Section 6 provides some examples that use the generic program structure and also provides executable PROBLOG code in the appendices based on these examples. Section 7 considers the use of tolerance spaces and their generalization to the probabilistic case. Section 8 considers the generalizations of probabilistic rough set methods. It also shows how all previous probabilistic methods considered so far are subsumed by a new definition of generalized probabilistic approximation spaces and operators. Additionally, the section specifies a small, compact set of higher-order PROBLOG relations used to specify any of these methods through instantiation. Section 9 then considers a relatively complex case study on a toy recommendation system that uses many of the concepts considered previously in the paper. The case study is encoded using the generic PROBLOG relations defined in the previous section and it also shows how different types of relations interact naturally in a PROBLOG program. Section 10 then summarizes results and concludes the paper.

2. Rough set reasoning landscape

2.1. Approximations and approximate/rough sets

In the paper we will consider both classical rough set methods based on the use of indiscernibility relations (reflexive, symmetric and transitive), as well as their generalizations, where the requirements as to the underlying base relation are relaxed. For example, rather than using indiscernibility as a basis for the base relation, one may require similarity (proximity, tolerance) among individuals instead. In this case, transitivity would be removed as a requirement and the focus would be on reflexive and symmetric base relations instead. These are just two of many choices.

In the rest of the paper we shall assume that:

- '*dom*' is a fixed finite domain, where x, y, and z (possibly subscripted) are used to denote objects in *dom* and c and d are used to denote subsets of *dom*.
- $\sigma \subseteq dom \times dom$, possibly with an index, is a base relation intended to represent indiscernibility. It is assumed that $\sigma(x, y)$ is true if and only if $\langle x, y\rangle \in \sigma$ holds. $\sigma(x) \stackrel{\text{def}}{=} \{y | \sigma(x, y)\}$ will signify the equivalence class that an individual x belongs to, i.e., the set of individuals equivalent to x. $|\sigma(x)|$ will be used to denote its cardinality. Later on in the paper, the symbols τ and υ will be used for generalizations of the base relation σ.

In the following definition we set no requirements on the base relation used to define approximations. The requirements used in subsequent parts of the paper are listed in Table 1.

Definition 2.1 (*Approximations, base relations, boundary regions, approximate sets*). Let σ be a binary relation on *dom* and $c \subseteq dom$. Then the *lower approximation* c^{+}_{σ} and the *upper approximation* $c^{\oplus}_{\sigma}$ *of c wrt* σ are:

Table 1
Properties of base relations in terms of approximations and first-order correspondences.

Notation	Property of $c^+_\sigma, c^\oplus_\sigma$	First-order correspondence	Property of σ
D	$c^+_\sigma \rightarrow c^\oplus_\sigma$	$\forall x \exists y(\sigma(x,y))$	Seriality
T	$c^+_\sigma \rightarrow c$	$\forall x(\sigma(x,x))$	Reflexivity
B	$c \rightarrow (c^\oplus_\sigma)^+_\sigma$	$\forall x \forall y(\sigma(x,y) \rightarrow \sigma(y,x))$	Symmetry
4	$c^+_\sigma \rightarrow (c^+_\sigma)^+_\sigma$	$\forall x \forall y \forall z((\sigma(x,y) \land \sigma(y,z)) \rightarrow \sigma(x,z))$	Transitivity
5	$c^\oplus_\sigma \rightarrow (c^\oplus_\sigma)^+_\sigma$	$\forall x \forall y \forall z((\sigma(x,y) \land \sigma(x,z)) \rightarrow \sigma(y,z))$	Euclidicity

$$c^+_\sigma \overset{\text{def}}{=} \{x | \forall y(\sigma(x,y) \rightarrow y \in c)\}; \tag{1}$$

$$c^\oplus_\sigma \overset{\text{def}}{=} \{x | \exists y(\sigma(x,y) \land y \in c)\}. \tag{2}$$

The difference $c^\oplus_\sigma \setminus c^+_\sigma$ is called the *boundary region* of c wrt σ. The relation σ is called the *base relation* for approximations $c^+_\sigma, c^\oplus_\sigma$. The pair $\langle c^+_\sigma, c^\oplus_\sigma \rangle$ is called an *approximate set*.

Definition 2.2. [Approximation space, rough approximations, rough sets] If the base relation σ of Definition 2.1 is an equivalence relation (reflexive, symmetric and transitive) then $AS = \langle dom, \sigma \rangle$ is called an *approximation space* and the approximations defined by (1)–(2) are called *rough approximations*. A *rough set* is an approximate set with the base relation being an equivalence relation.□

Remark 2.3. Note that arbitrary (n-argument with $n > 1$) approximate relations can be modelled by assuming that the domain *dom* consists of n-tuples of elements of "more elementary" domains. In the rest of the paper we will deal with a language with arbitrary relations. Definition 2.1 applies to such relations as well. □

Given an approximation space $AS = \langle dom, \sigma \rangle$, any set $c \subseteq dom$ can be partitioned into three disjoint regions:

- $POS_{AS}(c) = c^+_\sigma = \bigcup\{\sigma(x) | \sigma(x) \subseteq c^+_\sigma\}$, the *positive region* of c in AS;
- $BND_{AS}(c) = c^\oplus_\sigma \setminus c^+_\sigma = \bigcup\{\sigma(x) | \sigma(x) \subseteq (c^\oplus_\sigma \setminus c^+_\sigma)\}$, the *boundary region* of c in AS;
- $NEG_{AS}(c) = dom \setminus c^\oplus_\sigma = \bigcup\{\sigma(x) | \sigma(x) \subseteq (dom \setminus c^\oplus_\sigma)\}$, the *negative region* of c in AS.

2.2. Correspondences between approximations and properties of base relations

The properties listed in Table 1 have been extensively investigated in the area of modal logics and correspondence theory that studies the relation between modalities and Kripke accessibility relations [40]. Insights from these areas have been used to relate properties of approximations to properties of the underlying base relation σ (see, e.g., [8,47]). In particular:

- **D** ensures that the lower approximation of a set is included in its upper approximation;
- **T** ensures that the lower approximation is included in the approximated set;
- **B** ensures that the approximated set is included in the lower approximation of the upper approximation of the set;
- **4** ensures that the lower approximation of a set is included in the lower approximation of its lower approximation (so that iterating lower approximations does not change the result of its first application);
- **5** ensures that the upper approximation of a set is included in the lower approximation of its upper approximation.

Remark 2.4. Note that requirement **D** is equivalent to the seriality of the base relation σ. The inclusion of the lower approximation in the upper approximation is fundamental in approximate reasoning. However **D** alone does not guarantee that an

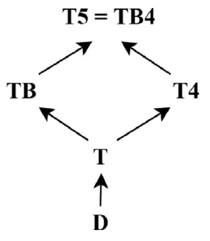

Fig. 2. Relationships among properties of base relations considered in the paper. An arrow $P \rightarrow Q$ indicates that the requirements P are weaker than the requirements Q.

object is indistinguishable/similar to itself. Therefore, in the methods considered in this paper we will assume (at least) **T**, the reflexivity of σ. Of course, **D** is entailed by **T**.□

Fig. 2 shows dependencies among properties of binary relations that are used as a basis for approximations. These dependencies are well-known from modal logic, and later, rough set methods (see, e.g., [40,47]). Note that the bottom of the figure, **D**, defines serial relations while the top, **T5**, defines equivalence relations. **TB** corresponds to tolerance relations. Note also that **T5** = **TB4** (in modal correspondence theory they both correspond to **S5**).

In the specification of rough set methods in PROBLOG, users will have the ability to represent selections of these properties in PROBLOG programs when specifying different approximation relations. This is shown in several sections of the paper.

3. Probabilistic rough set reasoning landscape

In this section, the focus will be on classical rough sets, where the base relation σ is an equivalence relation. A brief description of the different ways to generalize classical rough set methods to probabilistic rough set methods is provided. There have been many investigations of probabilistic generalizations of classical rough set theory in the literature [12,13,18,37,44–50].

In the classical theory, for a rough set c and for any object $x \in c^{+}_{\sigma}$, its associated equivalence class $\sigma(x)$ is a subset of the target set c, $\sigma(x) \subseteq c$. Similarly, for any object $x \in c^{\oplus}_{\sigma}$, its equivalence class $\sigma(x)$ intersects with c and the overlap must be non-empty ($\sigma(x) \cap c \neq \emptyset$). The starting point for probabilistic generalizations of rough sets is to take into account the degree of overlap that equivalence classes have with the target set c. These proportions can then be used as a basis for determining such quantities as the probabilities of individual objects relative to rough target sets, the degree of dependencies between rough sets, and construction of probabilistic rough set approximations, among others. The most basic generalization begins with the concept of *rough membership function* introduced in [43] (see also [20]).

Let *dom* be a finite set of individuals, $P : 2^{dom} \rightarrow [0,1]$, be a probability function on the powerset of *dom*, and σ be a binary relation that is also an equivalence relation on *dom*. The triplet $AS_P = \langle dom, \sigma, P\rangle$ is called a *probabilistic approximation space* [21].

Given $AS_P = \langle dom, \sigma, P\rangle$, by definition of P, it can be assumed that

- $P(\sigma(x))$ is known for all $\sigma(x)$ in the quotient set of *dom*, denoted dom/σ and defined as $\{\sigma(x) | x \in dom\}$;
- for each set $c \subseteq dom$ and equivalence class $\sigma(x)$, the probabilities $P(c)$ and $P(c \,|\, \sigma(x))$ are known.

Definition 3.1 (*Rough membership*). Given $AS_P = \langle dom, \sigma, P\rangle$, for a subset $c \subseteq dom$, the *rough membership function* with regard to c is:

$$\mu_{c,\sigma}(x) \overset{\text{def}}{=} P(c|\sigma(x)).$$

The function $\mu_c(x)$ provides a quantitative degree to which the object x belongs to the rough set c. It is defined as the conditional probability $P(c|\sigma(x))$ which characterizes the probability that a randomly chosen individual in $\sigma(x)$ is a member of the rough set c. Assuming a finite domain, it is often computed as the proportion,

$$\mu_{c,\sigma}(x) = \frac{|c \cap \sigma(x)|}{|\sigma(x)|}. \tag{3}$$

In a similar manner, the membership function μ can be generalized to *rough set inclusion* [26], the degree to which a rough set d is a subset of another rough set c. This inclusion, denoted by $\upsilon(d|c)$, is defined as,

$$\upsilon(d|c) \overset{\text{def}}{=} \frac{|c \cap d|}{|c|}. \tag{4}$$

If d is an equivalence class $\sigma(x) \in dom/\sigma$, then

$$\upsilon(\sigma(x)|c) = \frac{|c \cap \sigma(x)|}{|c|} \tag{5}$$

where it can easily be seen that $\upsilon(\sigma(x)|c) = \mu_{c,\sigma}(x)$.

Given a membership function $\mu_{c,\sigma}(x)$ for a rough set c, one can now define the classical rough set approximation operators for c probabilistically in terms of $\mu_{c,\sigma}(x)$.

Definition 3.2 (*Probabilistic approximations*). Given $AS_P = \langle dom, \sigma, P\rangle$, and $c \subseteq dom$, the *probabilistic lower* and *upper approximations* of c wrt σ are defined by:

$$c_\sigma^{P_+} \overset{\text{def}}{=} \left\{x | \mu_{c,\sigma}(x) = 1.0\right\} = \{x | P(c \mid \sigma(x)) = 1.0\}; \tag{6}$$

$$c_\sigma^{P_\oplus} \overset{\text{def}}{=} \left\{x | \mu_{c,\sigma}(x) > 0.0\right\} = \{x | P(c \mid \sigma(x)) > 0.0\}. \tag{7}$$

The difference $c_\sigma^{P_\oplus} \setminus c_\sigma^{P_+}$ is called the *probabilistic boundary region* of c wrt σ.

Given a probabilistic approximation space $AS_P = \langle dom, \sigma, P\rangle$, any set $c \subseteq dom$ can be partitioned into three disjoint regions:

- $POS_{AS_P}(c) \overset{\text{def}}{=} c_\sigma^{+} = \bigcup\{\sigma(x) | P(c|\sigma(x)) = 1.0\}$;
- $BND_{AS_P}(c) \overset{\text{def}}{=} c_\sigma^{\oplus} \setminus c_\sigma^{+} = \bigcup\{\sigma(x) | 0 < P(c|\sigma(x)) < 1.0\}$;
- $NEG_{AS_P}(c) \overset{\text{def}}{=} dom \setminus c_\sigma^{\oplus} = \bigcup\{\sigma(x) | P(c|\sigma(x)) = 0\}$.

Note that the probabilistic approximation operators in Definition 3.2 are *equivalent* to the approximation operators in Definition 2.1 in the following sense. Given $AS = \langle dom, \sigma\rangle$ and $AS_P = \langle dom, \sigma, P\rangle$, a non-probabilistic approximation space and a probabilistic approximation space, respectively, where the *dom*'s are the same and the base relations σ are the same, for any set $c \subseteq dom$,

$$POS_{AS_P}(c) = POS_{AS}(c), \;\; BND_{AS_P}(c) = BND_{AS}(c), \;\; \text{and} \;\; NEG_{AS_P}(c) = NEG_{AS_P}(c).$$

Consequently, probabilistic rough set models are a natural extension and will be shown to be generalizations of their classical counterparts.

In Pawlak et al. [21], a rough probabilistic model is introduced to characterize statistical dependencies between sets of attributes in an information system, in particular dependencies between input (condition) attributes and target (action) attributes. This is useful in constructing decision rules that take as input a subset of instantiated input attributes and return an instantiation of the target attributes. Many such tables of data lack sufficient information to induce deterministic mappings and the authors call these tables non-deterministic. Deterministic tables can be handled using the classical rough set model while non-deterministic tables require a generalization of the classical model that takes account of available probabilistic information in the rough set model. The probabilistic rough set model that is proposed in [21] is defined below in terms of 0.5-probabilistic approximations.

Definition 3.3 (*0.5-Probabilistic approximations*). Given $AS_{P(0.5)} = \langle dom, \sigma, P\rangle$, and $c \subseteq dom$, the 0.5-*probabilistic lower* and *upper approximation* of c wrt σ are defined by:

$$c_\sigma^{P_+} \overset{\text{def}}{=} \left\{x | \mu_{c,\sigma}(x) > 0.5\right\} = \{x | P(c \mid \sigma(x)) > 0.5\}; \tag{8}$$

$$c_\sigma^{P_\oplus} \overset{\text{def}}{=} \left\{x | \mu_{c,\sigma}(x) \geq 0.5\right\} = \{x | P(c \mid \sigma(x)) \geq 0.5\}. \tag{9}$$

Given a 0.5-probabilistic approximation space $AS_{P(0.5)} = \langle dom, \sigma, P\rangle$, any set $c \subseteq dom$, can be partitioned into three disjoint regions:

- $POS_{AS_{P(0.5)}}(c) \overset{\text{def}}{=} c_\sigma^{P_+} = \bigcup\{\sigma(x) | P(c|\sigma(x)) > 0.5\}$;
- $BND_{AS_{P(0.5)}}(c) \overset{\text{def}}{=} c_\sigma^{P_\oplus} \setminus c_\sigma^{P_+} = \bigcup\{\sigma(x) | P(c|\sigma(x)) = 0.5\}$;
- $NEG_{AS_{P(0.5)}}(c) \overset{\text{def}}{=} dom \setminus c_\sigma^{P_\oplus} = \bigcup\{\sigma(x) | P(c|\sigma(x)) < 0.5\}$.

This model has the flavor of a majority rule model. If more than 50% of the objects in an equivalence class $\sigma(x)$, overlap with c then $\sigma(x) \subseteq c_\sigma^{+}$. If exactly 50% of the objects in an equivalence class $\sigma(x)$, overlap with c then $\sigma(x) \subseteq (c_\sigma^{\oplus} \setminus c_\sigma^{+})$. If less than 50% of the objects in an equivalence class $\sigma(x)$, overlap with c then $\sigma(x) \subseteq (dom \setminus c_\sigma^{\oplus})$. If an object x is in $POS_{AS_{P(0.5)}}(c)$ or $NEG_{AS_{P(0.5)}}(c)$, the idea is that one can be statistically confident that the object does or does not satisfy the properties of the concept c.

The remaining rough probabilistic models that will be considered in this paper can be viewed as generalizations of Definition 3.2 and Definition 3.3 (although see Remark 3.5) through the use of parameterization. This perspective is elegantly described in Yao [44].

Ziarko [49] proposed the *variable precision rough set model* where one allows for different variable levels of set inclusion in the definitions of approximation operators. Recall from Definition 3.3 that for $c_\sigma^{P_+}$,

$$\sigma(x) \subseteq c \equiv v(\sigma(x)|c) > 0.5 \equiv P(c|\sigma(x)) > 0.5. \tag{10}$$

Generalizing this, for $\alpha \in (0, 1]$,

$$\sigma(x) \subseteq c \equiv v(\sigma(x)|c) > \alpha \equiv P(c|\sigma(x)) > \alpha. \tag{11}$$

For the lower approximation, the majority rule constraint would constrain α to be greater than 0.5. Consequently, $\alpha \in (0.5, 1]$. Additionally, for the upper approximation, in order for it to retain the property of being a dual operator to the lower approximation, its parameter should be $1 - \alpha$. This pair of parameters is $(\alpha, 1 - \alpha)$ is referred to as the symmetric bounds. The use of symmetric bounds ensures that the lower approximation is always a subset of the upper approximation.

Definition 3.4 (*Variable precision approximations, symmetric bounds*). Let $\alpha \in (0.5, 1.0]$. Given $AS_{VP(\alpha)} = \langle dom, \sigma, P, \alpha \rangle$, and $c \subseteq dom$, the *variable precision lower and upper approximation(s) of c wrt α and σ* are defined by:

$$c_{\sigma}^{V_+} \overset{\text{def}}{=} \left\{ x | \mu_{c,\sigma}(x) \geq \alpha \right\} = \{ x | P(c|\sigma(x)) \geq \alpha \}; \tag{12}$$

$$c_{\sigma}^{V_\oplus} \overset{\text{def}}{=} \left\{ x | \mu_{c,\sigma}(x) > 1 - \alpha \right\} = \{ x | P(c|\sigma(x)) > 1 - \alpha \}. \tag{13}$$

The difference $c_{\sigma}^{V_\oplus} \setminus c_{\sigma}^{V_+}$ is called the *variable precision boundary region* of c wrt σ.

Remark 3.5. Observe that the variable precision lower approximation differs from the 0.5-lower approximation by using '$\geqslant \alpha$' rather than '$> \alpha$' (with $\alpha = 0.5$). However, in order to define the classical (non-probabilistic) lower approximation, one needs $\alpha = 1.0$. In order to be able to subsume the 0.5-model, with $\geqslant$, as in Eq. (12), one has to use $\alpha = 0.5 + \epsilon$ with a sufficiently small ϵ $(0 < \epsilon < 1/|dom|)$. In a similar manner, the 0.5-upper approximation differs from the variable precision upper approximation (here $>$ is used rather than $\geqslant$). $\square$

Given a variable precision approximation space $AS_{VP(\alpha)} = \langle dom, \sigma, \alpha \rangle$, any set $c \subseteq dom$ can be partitioned using parameter α into three disjoint regions:

- $POS_{AS_{VP(\alpha)}}(c) \overset{\text{def}}{=} c_{\sigma}^{V_+} = \bigcup \{ \sigma(x) | P(c|\sigma(x)) \geqslant \alpha \}$;
- $BND_{AS_{VP(\alpha)}}(c) \overset{\text{def}}{=} c_{\sigma}^{V_\oplus} \setminus c_{\sigma}^{V_+} = \bigcup \{ \sigma(x) | 1 - \alpha < P(c|\sigma(x)) < \alpha \}$;
- $NEG_{AS_{VP(\alpha)}}(c) \overset{\text{def}}{=} dom \setminus c_{\sigma}^{V_\oplus} = \bigcup \{ \sigma(x) | P(c|\sigma(x)) \leqslant 1 - \alpha \}$.

Asymmetric bounds for variable precision rough set models were considered in Katzberg and Ziarko [14] (for the more recent versions see [49,50]). Here one introduces two parameters, α and β, for the lower and upper approximations, respectively, where $0.0 \leqslant \beta < \alpha \leqslant 1.0$. For the cases of rough set inclusion where $v(\sigma(x)|c) = P(c|\sigma(x))$, this approach is equivalent to the most general case we consider which derives probabilistic approximations using a decision-theoretic model [46].

Definition 3.6 (*Decision-theoretic approximations*). Let $0.0 \leqslant \beta < \alpha \leqslant 1.0$. Given $AS_{DT(\alpha,\beta)} = \langle dom, \sigma, P, \alpha, \beta \rangle$, and $c \subseteq dom$, the *decision-theoretic lower and upper approximation(s) of c wrt α, β and σ* are defined by:

$$c_{\sigma}^{D_+} \overset{\text{def}}{=} \left\{ x | \mu_{c,\sigma}(x) \geq \alpha \right\} = \{ x | P(c|\sigma(x)) \geq \alpha \}; \tag{14}$$

$$c_{\sigma}^{D_\oplus} \overset{\text{def}}{=} \left\{ x | \mu_{c,\sigma}(x) > \beta \right\} = \{ x | P(c|\sigma(x)) > \beta \}. \tag{15}$$

The difference $c_{\sigma}^{D_\oplus} \setminus c_{\sigma}^{D_+}$ is called the *decision-theoretic boundary region* of c wrt σ.

Given a decision theoretic approximation space $AS_{DT(\alpha,\beta)} = \langle dom, \sigma, \alpha, \beta \rangle$, any set $c \subseteq dom$ can be partitioned using the parameters α and β into three disjoint regions (see Fig. 3):

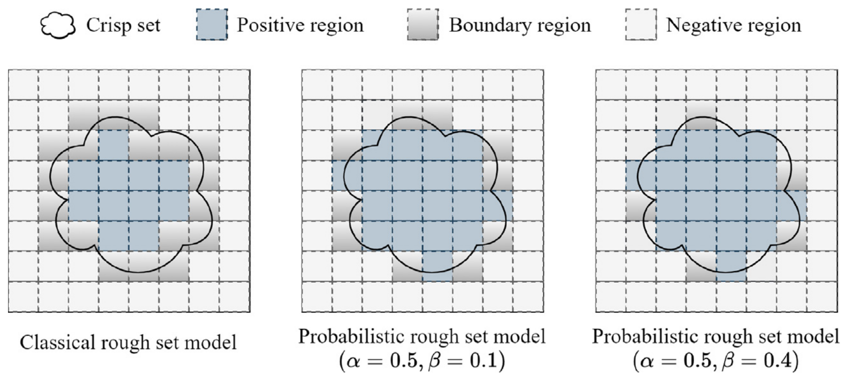
Crisp set
Positive region
Boundary region
Negative region
Classical rough set model
Probabilistic rough set model
$(\alpha = 0.5, \beta = 0.1)$
Probabilistic rough set model
$(\alpha = 0.5, \beta = 0.4)$

Fig. 3. Classical vs probabilistic rough set model.

- $POS_{AS_{DT(\alpha,\beta)}}(c) \stackrel{\text{def}}{=} c_\sigma^{D_+} = \bigcup\{\sigma(x) | P(c|\sigma(x)) \geqslant \alpha\}$;
- $BND_{AS_{DT(\alpha,\beta)}}(c) \stackrel{\text{def}}{=} c_\sigma^{D_\oplus} \setminus c_\sigma^{D_+} = \bigcup\{\sigma(x) | \beta < P(c|\sigma(x)) < \alpha\}$;
- $NEG_{AS_{DT(\alpha,\beta)}}(c) \stackrel{\text{def}}{=} dom \setminus c_\sigma^{D_\oplus} = \bigcup\{\sigma(x) \,|\, P(c|\sigma(x)) \leqslant \beta\}$.

The distinction between the asymmetric bounds variable precision approximation model and the decision-theoretic approximation model is that the latter proposes methods for specifying the parameters α and β in a theoretically principled manner using Bayesian decision theory. In this case, the risk in making a wrong choice for particular levels of set inclusion is quantified and reflected in the choice of α and β used in the definition of the approximation operators. For the interested reader, techniques for determining α and β are considered in [47,46]. In the remainder of the paper, we will assume that these parameters are set appropriately by the users of our PROBLOG program specifications.

The following proposition is well-known. It shows that we can focus on the decision theoretic approximation model without loss of generality.

Proposition 3.7. *The decision theoretic approximation model subsumes:*

- *the probabilistic approximation model (Definition 3.2), where $\alpha = 1.0$ and $\beta = 0.0$;*
- *the 0.5-probabilistic approximation model (Definition 3.3), where $\alpha = 0.5 + \epsilon$ and $\beta = 0.5 - \epsilon$, where ϵ is explained in Remark 3.5;*
- *the variable precision approximation model (Definition 3.4), where $\beta = 1.0 - \alpha$.*

An extension of the variable precision model, called *Bayesian Rough Sets*, has been introduced and discussed in [37,50]. The idea behind this approach, as expressed in [37], is that "in some applications, for example in stock market, medical diagnosis etc., the objective is to achieve some certainty prediction improvement rather than trying to produce rules satisfying preset certainty requirements."

Definition 3.8 (*Bayesian approximations*). Given $AS_B = \langle dom, \sigma, P\rangle$, called the *Bayesian approximation space*, and $c \subseteq dom$, the *Bayesian lower* and *upper approximation* of c wrt σ are defined by:

$$c_\sigma^{P_+} \stackrel{\text{def}}{=} \left\{x | \mu_{c,\sigma}(x) > P(c)\right\} = \{x | P(c \mid \sigma(x)) > P(c)\}; \tag{16}$$

$$c_\sigma^{P_\oplus} \stackrel{\text{def}}{=} \left\{x | \mu_{c,\sigma}(x) \geq P(c)\right\} = \{x | P(c \mid \sigma(x)) \geq P(c)\}. \tag{17}$$

Given a Bayesian approximation space $AS_B = \langle dom, \sigma, P\rangle$, any set $c \subseteq dom$, can be partitioned into three disjoint regions:

- $POS_{AS_B}(c) \stackrel{\text{def}}{=} c_\sigma^{P_+} = \bigcup\{\sigma(x) | P(c|\sigma(x)) > P(c)\}$;
- $BND_{AS_B}(c) \stackrel{\text{def}}{=} c_\sigma^{P_\oplus} \setminus c_\sigma^{P_+} = \bigcup\{\sigma(x) | P(c|\sigma(x)) = P(c)\}$;
- $NEG_{AS_B}(c) \stackrel{\text{def}}{=} dom \setminus c_\sigma^{P_\oplus} = \bigcup\{\sigma(x) | P(c|\sigma(x)) < P(c)\}$.

Before proceeding to a consideration of PROBLOG and how one would use PROBLOG for reasoning with probabilistic approximations of rough sets, an example as to how probabilistic rough sets can be derived from table data is provided.

A common use of rough set reasoning is based on the use of table data. Generally, one assumes a universe of individuals U, possibly infinite, but one only has access to a finite subset of these individuals $dom \subset U$. One then associates attribute/value pairs with these individuals representing *input attributes* and one is interested in understanding relationships between sets of individuals defined by these input attributes and sets of individuals defined by *target attributes*. It is often the case that these sets of individuals can not be fully characterized in terms of the input attributes and so can be interpreted as rough sets relative to an indiscernibility relationship implicit in the table of data. Given a table of data, each row can be viewed as a sample pertaining to an individual. In the following, reference will be made to Table 2 which is also found in Russell and Norvig [31].

Example 3.9. Suppose one is interested in the relationship between the input attributes *Hun*, *Type* and the target attribute *WillWait*. Table 2 has 12 samples. Let us create a corresponding variable precision approximation space with symmetric bounds, $AS_{VP(\alpha)} = \langle dom, \sigma, \alpha\rangle$, where $\alpha = 0.5, dom = \{x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8, x_9, x_{10}, x_{11}, x_{12}\}$ and σ is the indiscernibiltiy relation constructed from the combination of attributes *Hun* and *Type*. Since the domain of *Hun* (Yes, No), has cardinality 2 and the domain of *Type* (French, Thai, Burger, Italien), has cardinality 4, there are 8 equivalence classes uniquely distinguished by the values assigned to the attributes. The eight equivalence classes compose the quotient set dom/σ: see Table 3.

Table 2
Restaurant Example: Russell, Norvig [31, p. 657]with the attributes: Alt: is there a suitable alternative restaurant nearby? Bar: has a bar area? Fri: is today Friday or Saturday? Hun: hungry right now? Pat: people in the restaurant, Price: the price range, Rain: raining outside? Res: is a reservation made? Type: the kind of restaurant, Est: host's waiting time estimate (in minutes), WillWait: will wait for a table?

Example	Input Attributes										Output
	Alt	Bar	Fri	Hun	Pat	Price	Rain	Res	Type	Est	WillWait
x_1	Yes	No	No	Yes	Some	$ $ $	No	Yes	French	0 – 10	Yes
x_2	Yes	No	No	Yes	Full	$	No	No	Thai	30 – 60	No
x_3	No	Yes	No	No	Some	$	No	No	Burger	0 – 10	Yes
x_4	Yes	No	Yes	Yes	Full	$	Yes	No	Thai	10 – 30	Yes
x_5	Yes	No	Yes	No	Full	$ $ $	No	Yes	French	> 60	No
x_6	No	Yes	No	Yes	Some	$ $	Yes	Yes	Italien	0 – 10	Yes
x_7	No	Yes	No	No	None	$	Yes	No	Burger	0 – 10	No
x_8	No	No	No	Yes	Some	$ $	Yes	Yes	Thai	0 – 10	Yes
x_9	No	Yes	Yes	No	Full	$	Yes	No	Burger	> 60	No
x_{10}	Yes	Yes	Yes	Yes	Full	$ $ $	No	Yes	Italien	10 – 30	No
x_{11}	No	No	No	No	None	$	No	No	Thai	0 – 10	No
x_{12}	Yes	Yes	Yes	Yes	Full	$	No	No	Burger	30 – 60	Yes

Table 3
Conditional probabilities for equivalence classes.

$P(WW\|\sigma_{Hun=yes,Type=French})$	1.0	$P(WW\|\sigma_{Hun=no,Type=French})$	0.0
$P(WW\|\sigma_{Hun=yes,Type=Thai})$	2/3	$P(WW\|\sigma_{Hun=no,Type=Thai})$	0.0
$P(WWW\|\sigma_{Hun=yes,Type=Burger})$	1.0	$P(WW\|\sigma_{Hun=no,Type=Burger})$	1/3
$P(WW\|\sigma_{Hun=yes,Type=Italien})$	1/2	$P(WW\|\sigma_{Hun=no,Type=Italien})$	0.0

$$\begin{aligned}
&\sigma_{Hun=yes,Type=French} = \{x_1\}, \quad \sigma_{Hun=yes,Type=Thai} = \{x_2, x_4, x_8\},\\
&\sigma_{Hun=yes,Type=Burger} = \{x_{12}\}, \quad \sigma_{Hun=yes,Type=Italien} = \{x_6, x_{10}\},\\
&\sigma_{Hun=no,Type=French} = \{x_5\}, \quad \sigma_{Hun=no,Type=Thai} = \{x_{11}\},\\
&\sigma_{Hun=no,Type=Burger} = \{x_3, x_7, x_9\}, \quad \sigma_{Hun=no,Type=Italien} = \{\}
\end{aligned}$$

For instance, the equivalence class $\sigma_{Hun=yes,Type=Thai}$ can be characterized as a logical rule as follows:

$$\sigma_{YT}(X) \leftarrow hun(X, yes), type(X, thai), \tag{18}$$

where '$\leftarrow$' is to be understood as an implication from right to left.

The base (target) set of interest will be the set defined by all individuals with attribute $WillWait = Yes$. This set is denoted as WW, $WW = \{x_1, x_3, x_4, x_6, x_8, x_{12}\}$. The following logical rule can be defined for WW:

$$ww(X) \leftarrow willWait(X, yes). \tag{19}$$

The interest then is defining the rough set correlate of WW consisting of the lower and upper approximation of WW that is denoted as WW_r and defined as:

$$WW_r = \langle WW_\sigma^+, WW_\sigma^\oplus \rangle. \tag{20}$$

Recall that rough membership of a base set c is defined as $\mu_c(x) = P(c|\sigma(x))$, where the conditional probability provides the probability that an individual, randomly chosen from its equivalence class $\sigma(x)$, belongs to the rough set derived from c. In the case of the example,

$$\mu_{ww}(x) = P(ww(x)|\sigma(x)) = \frac{|ww(x) \cap \sigma(x)|}{|\sigma(x)|}. \tag{21}$$

The following logical rule will be useful when defining the approximation operators for WW,

$$ww_r(x) \leftarrow \mu_{ww}(x). \tag{22}$$

Table 3.9 provides the conditional probabilities required which are derived from rule (22) and the sample data in Table 2. According to the rough probability model being used,

$$\begin{aligned}
WW_\sigma^{P_+} &= \sigma_{Hun=yes,Type=French} \cup \sigma_{Hun=yes,Type=Thai} \cup \sigma_{Hun=yes,Type=Burger} = \{x_1, x_2, x_4, x_8, x_{12}\}\\
WW_\sigma^{P_\oplus} &= \sigma_{Hun=no,Type=French} \cup \sigma_{Hun=yes,Type=Thai} \cup \sigma_{Hun=yes,Type=Burger} \cup \sigma_{Hun=yes,Type=Italien}\\
&= \{x_1, x_2, x_4, x_6, x_8, x_{10}, x_{12}\},
\end{aligned}$$

The following logical rules can be used to determine membership in WW^{+}_{σ} and $WW^{\oplus}_{\sigma}$, respectively,

$$\begin{aligned}
&lowerWW(x) &&\leftarrow ww_r(x) \geqslant \alpha;\\
&upperWW(x) &&\leftarrow ww_r(x) > 1 - \alpha.
\end{aligned}$$

Additionally,

$$\begin{aligned}
POS_{AS_{VP(0.5)}}(WW) &= \sigma_{Hun=yes,Type=French} \cup \sigma_{Hun=yes,Type=Burger} \cup \sigma_{Hun=yes,Type=Thai})\\
&= \{x_1, x_2, x_4, x_8, x_{12}\}\\
BND_{AS_{VP(0.5)}}(WW) &= \sigma_{Hun=yes,Type=Italien} = \{x_6, x_{10}\}\\
NEG_{AS_{VP(0.5)}}(WW) &= \sigma_{Hun=no,Type=French} \cup \sigma_{Hun=no,Type=Thai} \cup \sigma_{Hun=no,Type=Burger}\\
&\quad \cup \sigma_{Hun=no,Type=Italien} = \{x_3, x_5, x_7, x_9, x_{11}\}.
\end{aligned}$$

The following logical rules can be used to determine membership in $POS(WW), NEG(WW)$ and $BND(WW)$, respectively,

$$\begin{aligned}
&POS_{ww}(x) &&\leftarrow ww_r(x) \geqslant \alpha;\\
&BND_{ww}(x) &&\leftarrow 1 - \alpha < ww_r(x) < \alpha;\\
&NEG_{ww}(x) &&\leftarrow ww_r(x) \leqslant 1 - \alpha.
\end{aligned}$$

For the purpose of basic reasoning with probabilistic rough sets, Example 3.9 exhibits most of the required functionality. $ww_r(x)$, defined in (22), is a probabilistic predicate where the distribution is derived from the equivalence class σ. Associated with $ww_r(x)$ are a number of other properties defined using the probability model: $lowerWW(x), upperWW(x), POS(WW), NEG(WW), BND(WW)$. The idea will be to embed probabilistic rough sets and their properties into PROBLOG programs. One can then combine probabilistic rough relations/properties such as $ww_r(x)$ with standard probabilistic and non-probabilistic relations in defining probabilistic knowledge bases consisting of tables and relational rules. Before proceeding to more complex examples that do this, a short summary of PROBLOG is provided in the next section.

4. Short summary of PROBLOG and its semantics

In the past decades there has been a fundamental interest in combining logic with probabilities, in particular in the form of probabilistic programming languages [4,11,24]. One area of major interest has been the development of probabilistic logic programming languages, including CLP(BN) [3], CP [41], ICL [27], LP$^{\mathrm{MLn}}$ [16], P-LOG [1], PRISM [35], PROBLOG [5], PROPPR [42].

PROBLOG is a probabilistic extension of PROLOG that is one of the more popular languages due to its ease of use in addition to having a large user community. PROBLOG [3] will be used in this paper as a basis for specifying the diverse collection of probabilistic rough set models considered in this paper.

Program 1: PROBLOG program from the on-line PROBLOG Tutorial [29].

```
person(john).       person(mary).
0.7::burglary.      0.2::earthquake.
0.9::alarm  :-  burglary, earthquake.
0.8::alarm  :-  burglary, \+ earthquake.
0.1::alarm  :-  \+ burglary, earthquake.
0.8::calls(X)  :-  alarm, person(X).
0.1::calls(X)  :-  \+ alarm, person(X).
evidence(calls(john),true).
evidence(calls(mary),true).
query(burglary).
query(earthquake).
```

Consider the basic PROBLOG program shown as Program 1 and taken from the on-line PROBLOG Tutorial [29]. It specifies the well-known Bayesian Network example in [22].A PROBLOG program consists of probabilistic facts, probabilistic clauses, rules and queries. In Program 1):

[3] For an online PROBLOG implementation that can be used for validating examples considered in the paper see https://dtai.cs.kuleuven.be/problog/editor.html. Note that the online version is somewhat unstable for larger examples. In this case it is better to download the PROBLOG distribution (accessible at the same site) that is compatible with Python.

- the probabilistic fact “0.7::burglary” (Line 2) represents a random variable “burglary” with probability distribution 0.7:: true, 0.3::false;
- an unannotated fact such as “person(john)” (Line 1) has the probability distribution: 1.0::true, 0.0::false.

A probabilistic clause is syntactic sugar for a set of conditionalized probabilistic facts. For example, the rule in Line 6 encodes a set of probabilistic rules, one for each person,

$$0.8 :: \text{calls}(\text{john}) :\text{-}\ \text{alarm},\ \text{person}(\text{john}). \tag{23}$$
$$0.8 :: \text{calls}(\text{mary}) :\text{-}\ \text{alarm},\ \text{person}(\text{mary}). \tag{24}$$

Clause (23) implicitly represents a probabilistic fact and a clause:

$$0.8 :: \text{calls\_aux}(\text{john}).$$
$$\text{calls}(\text{john}) : -\text{calls\_aux}(\text{john}), \text{alarm}, \text{person}(\text{john}).$$

Given a probabilistic knowledge base, one can query it in various ways. In this example, one specifies two conditional probabilistic queries,

$$P(burglary|calls(john), calls(mary)); \quad P(earthquake|calls(john), calls(mary)).$$

Here the relation ‘evidence(·)’ is used to specify evidence facts, while ‘query(·)’ is used to specify the question to be asked. Note that ‘query(·)’ can be used without any ‘evidence(·)’. In the example, the result of these queries would show that

$$P(burglary|calls(john), calls(mary)) = 0.981939,$$
$$P(earthquake|calls(john), calls(mary)) = 0.226851.$$

PROBLOG is based on *distribution semantics* [5,27,34]. In its very spirit, PROBLOG is like restricted Prolog except that one may additionally specify particular facts (or consequences of rules) as being probabilistic. This gives rise to many possible worlds. In each world, probabilistic literals are chosen to be true or false, and the probability of this choice contributes to the probability of the world. That is, in each world one obtains a pure Prolog program and the worlds differ in the set of facts. Distribution semantics calculates probabilities using the obtained probability distribution on worlds.

The following summary of the semantics follows [15]. A PROBLOG program

$$\Pi = \{p_1 :: c_1, \ldots, p_n :: c_n\} \cup KB,$$

consists of a set of labeled facts (probabilistic facts), and KB a set of definite clauses, the Knowledge Base. Given a finite set of substitutions $\left\{\theta_{j1}, \ldots, \theta_{ji_j}\right\}$ for each probabilistic fact $p_j :: c_j$, let

$$F = \left\{c_1\theta_{11}, \ldots, c_1\theta_{1i_1}, \ldots, c_n\theta_{n1}, \ldots, c_n\theta_{ni_n}\right\}$$

be the maximal set of grounded logical facts that can be added to KB. The random variables corresponding to facts in F are mutually independent, so the program Π defines a probability distribution over ground logic programs $F' \subseteq F$:

$$P(F'|\Pi) = \prod_{c_i\theta_j \in F'} p_i * \prod_{c_i\theta_j \in F \setminus F'} (1 - p_i). \tag{25}$$

The probability that a ground query q is true given a program Π is

$$P(q|\Pi) = \sum_{F' \subseteq F} P(q|\Pi) \cdot P(F'|\Pi). \tag{26}$$

$P(q|\Pi)$ is 1.0 if there is a substitution θ such that $F' \cup \Pi \models q\theta$ otherwise it is 0.0 and $P(F'|\Pi)$ is as defined in Eq. 25. $P(q|\Pi)$ is called the success probability of q.

Remark 4.1. The complexity of computing probabilities of queries directly using (26) is exponential wrt the number of distinct probabilistic predicates occurring in the program. However, as indicated in [5], “as fixing the set of facts yields an ordinary logic program, the entailment check can use any reasoning technique for such programs.” Such techniques are typically much more efficient, however, still exponential in the worst case. A better performance may be achieved by statistical sampling or using bounds, where the set of possible worlds where the given query is true is approximated by their subset and superset [5].□

PROBLOG extends classical PROLOG with stratified negation and uses the symbol \+ for negation. As mentioned before, the main difference between PROBLOG and PROLOG is that PROBLOG supports probabilistic facts. The following provides an incomplete list of those parts of PROBLOG (and PROLOG) used in the paper, in addition to a number of other useful features. For additional features the reader is referred to the PROBLOG on-line tutorial [29] and related literature, including [4,5], in addition to other references there.

- To specify *probabilistic predicates*, an operator '::' is used. If $\overline{a}$ is a tuple of constants, p is a relation symbol and $r \in [0.0, 1.0]$ is a real number, then the specification:

$$r :: p(\overline{a}) \tag{27}$$

states that $p(\overline{a})$ is true with probability r and false with probability $(1 - r)$.
Probabilistic predicates can be used to specify probabilistic facts and heads of probabilistic clauses.
- *Queries* are used to compute probabilities. The probability of $p(\overline{a})$ is returned as the result of the query:

$$query(p(\overline{a})). \tag{28}$$

  - *Evidence* is used to specify any observations on which one wants to condition a probability. Each piece of evidence is specified using a binary relation $evidence$ where the first argument indicates a fact and the second one indicates a classical truth value:

$$evidence(p(a), true). \qquad evidence(q(b), false). \tag{29}$$

They indicate that $p(a)$ is true (respectively, $q(b)$ is false). Evidence facts are useful in computing (conditional) probabilities by restricting worlds to those where (in this case) $p(a)$ is *true* (respectively, $q(b)$ is *false*).
Queries can be also be used inside the bodies of rules by using the *subquery* predicate:

$$subquery(p(a), P, ListOfEvidence) \tag{30}$$

which evaluates p(a) and returns P as the conditional probability of p(a) given the evidence listed in *ListOfEvidence*. *ListOfEvidence* may also be empty.
- *Annotated disjunctions* support choices. If $r_1, \ldots, r_k \in [0.0, 1.0]$ such that $0.0 \leqslant r_1 + \ldots + r_k \leqslant 1.0, \overline{a_1}, \ldots, \overline{a_k}$ are tuples of constants and p is a relation symbol then an *annotated disjunction* is an expression of the form:

$$r_1 :: p(\overline{a_1}); \ldots; r_k :: p(\overline{a_k}). \tag{31}$$

It states that at most one of $r_1 :: p(\overline{a_1}), \ldots, r_k :: p(\overline{a_k})$ is chosen as a probabilistic fact. If $r_1 + \ldots + r_k = 1.0$ then exactly one of the listed literals is selected, otherwise there is an implicit *null choice* indicating that none of the options is taken with the probability $1 - (r_1 + \ldots + r_k)$. An annotated disjunction may occur as a fact or as a head of a rule.
- *Negation* is expressed by \+, where '\' stands for 'not' and '+' stands for 'provable'. That is, \+ represents *negation as failure*. For backward compatibility the connective 'not' can be used instead of \+.[4]

A number of standard PROLOG commands, available in PROBLOG will also be used:

- $call(p, \bar{a})$, where p is a relation symbol and $\bar{a}$ abbreviates 1 to 8 parameters. When $call(p, \bar{a})$ is encountered as a sub-goal, it evaluates $p(\bar{a})$;
- $findall(X, Q, L)$: creates a list L of all values of X making query $Q(X)$ true;
- predicates from a standard library *lists*:
  - *list_to_set*(L, S): computes S from L by removing duplicates;
  - $length(L, N)$: computes N as the length of the list L.
  - $intersection(L1, L2)$: computes the intersection of the two lists $L1$ and $L2$.

5. Implementing probabilistic rough sets in PROBLOG

5.1. The structure of programs used in the paper

In [9] the authors developed a generic program structure for Answer Set programs which could be used as a tool for implementing (non-probabilistic) rough sets and their generalizations. Program 2, adapted from [9], shows a corresponding structure for PROBLOG programs that takes into account probabilistic rough sets and their generalizations. For the sake of readability, a number of program structuring keywords, not belonging to the syntax of PROBLOG, will be used. These are written in boldface font and should be treated as comments:

- **constants** – specify global constants using one-argument relations, like $alpha(0.9)$;
- **crisp/probabilistic set(s)** – specify the domain '*dom*' used, in addition to explicitly specifying or implicitly generating crisp or probabilistic sets. Specifications may use crisp and/or probabilistic predicates. For simplicity we assume that all constants occurring in a program have to belong to its domain '*dom*';

[4] The connective 'not' tends to be avoided to emphasize that the negation operator used in PROLOG and PROBLOG differs from classical negation.

- **crisp/probabilistic base relation(s)** – specify explicitly or generate base relations. The relations may be crisp and/or probabilistic. The properties that may be considered (**T**, **B**, **4**, **5**) are listed in Lines 11–13 and should be selected or commented out in order to reflect the desired properties of relations. The property **T** is required in all rough set generalizations as it corresponds to the requirement that any object is indistinguishable from/similar to itself.
- **approximations** – specify or generate lower and/or upper approximations for concepts and relations. Additionally define other approximation operators;
- **knowledge base** – specify a background knowledge base using PROBLOG rules and facts.

Note that we assume that for each crisp set c_i there is a base relation $\sigma_i(\ldots)$ used for approximating c_i. For example, a base relation for similarity among company clients is different from similarity among items being offered by the company. For the same reasons we allow for many domains.

Program 2: The structure of PROBLOG programs used in the paper.

```
1  :- use_module(library(lists)).
constants:
2  | $alpha_i(\ldots)$.                  % specification of global constants $\alpha_i$ of Definition 3.6 ($1 \le i \le k$)
3  | $beta_i(\ldots)$.                   % specification of global constants $\beta_i$ of Definition 3.6 ($1 \le i \le k$)
4  | $gamma_i(\ldots)$                   % specification of global constants $\gamma_i$ of Definition 8.1 ($1 \le i \le k$)
5  | ...                                 % specification of other global constants
crisp/probabilistic set(s):
6  | $dom(\ldots)$.                      % specification of domain(s)
7  | ...
8  | ... $c_i(\ldots)$. ....             % facts about crisp/probabilistic sets/relations $1 \le i \le m$
crisp/probabilistic base relation(s):
9  | ... $s_i(\ldots)$. ....                        % facts about base relations $1 \le i \le k$
10 | $\sigma_i(X,Y)$ :- $s_i(X,Y)$.                  % $\sigma_i$ includes $s_i$ ($1 \le i \le k$)
11 | $\sigma_i(X,X)$ :- $dom(X)$.                    % property T for the $i$-th base relation ($1 \le i \le k$)
12 | $\sigma_i(X,Y)$ :- $dom(X), dom(Y), \sigma_i(Y,X)$.   % property B for the $i$-th base relation ($1 \le i \le k$)
13 | $\sigma_i(X,Y)$ :- $dom(Y), s_i(X,Z), \sigma_i(Z,Y)$. % property 4 for the $i$-th base relation ($1 \le i \le k$)
14 | $\sigma_i(Y,Z)$ :- $dom(X), s_i(X,Y), \sigma_i(X,Z)$. % property 5 for the $i$-th base relation ($1 \le i \le k$)
approximations:
15 | ... $c^{+}_{j_{\sigma_i}}(\ldots)$. ...      % a specification of lower approximations $1 \le i \le k, 1 \le j \le m$
16 | ... $c^{\oplus}_{j_{\sigma_i}}(\ldots)$. ... % a specification of upper approximations $1 \le i \le k, 1 \le j \le m$
knowledge base:
17 | ...                                 % ProbLog rules
```

Program 2 provides the basic structure of a program using informal syntax. In Line 1, the standard PROLOG library 'lists' is loaded. It provides built-in operations on lists.

As in [9], properties **T**, **B**, **4**, **5** formulated in Lines 11–14, reflect first-order conditions shown in Table 1.

6. Some examples

6.1. Probabilistic rough set reasoning

Example 6.1. An instantiation of the program template (Program 2) considered in Section 5.1 is shown in Program 3 where, in addition to Table 2, we use data for table rating contained in Table 4. An executable PROBLOG code for this example is shown in A.1. A brief description of the program is now provided and a number of queries that use this example are then considered that illuminate the use of probabilistic rough sets.

Consider the annotated program schema shown as Program 3.

Table 4
Restaurant rating table.

Restaurant	TypeOf	Rating	Restaurant	TypeOf	Rating	Restaurant	TypeOf	Rating
r_1	French	4	r_6	Italien	3	r_{11}	Korean	3
r_2	Thai	3	r_7	Burger	1	r_{12}	Chinese	4
r_3	Burger	2	r_8	Thai	5	r_{13}	Japanese	5
r_4	Thai	4	r_9	Japanese	4	r_{14}	Chinese	3
r_5	French	5	r_{10}	Steak	5	r_{15}	Italien	2

Program 3: PROBLOG program schema for Example 6.1. For executable PROBLOG code see A.1.

```
1  :- use_module(library(lists)).
constants:
2      alpha(0.5).                % specification of global constant $\alpha$ of Definition 3.6
3      beta(0.5).                 % specification of global constant $\beta$ of Definition 3.6
crisp/probabilistic set(s):
4      $dom_1(x_1) \dots dom_1(x_{12})$.     % specification of domain(s)
5      $dom_2(r_1) \dots dom_2(r_{15})$.     % specification of domain(s)
6 7    $wwait(x_1,-) \dots wwait(x_{12},-), \dots hun(x_1,-) \dots hun(x_{12},-), \dots$   % data based on Table 2
8      $type(x_1,-) \dots type(x_{12},-), typeOf(r_1,-) \dots typeOf(r_{15},-), \dots$
9      $rating(r_1,-) \dots rating(r_{15},-)$                                  % data based on Table 4
10 11  ww(X) :- wwait(X, yes).    % a rule defining WW
crisp/probabilistic base relation(s):
12     $\sigma\_(X,Y)$ :- hun(X,-), type(X,-), hun(Y,-), type(Y,-).   % equivalence classes
13 14  $\sigma(X,Y)$ :- s(X,Y).                  % $\sigma$ includes $\sigma\_$
15     $\sigma(X,X)$ :- dom(X).                  % property T
16     $\sigma(X,Y)$ :- dom(X), dom(Y), $\sigma(Y,X)$.   % property B
17     $\sigma(X,Y)$ :- dom(Y), s(X,Z), $\sigma(Z,Y)$.   % property 4
18     $\sigma_1(X,L)$ :- findall(Y, $\sigma(X,Y)$, L1), list_to_set(L1, L).   % returns list of individuals in $\sigma(X)$
approximations:
19     lower(X,C) :- rmb(X,C,P), alpha(A), P >= A.    % a specification of lower approximation
20     upper(X,C) :- rmb(X,C,P), beta(B), P > B.      % a specification of upper approximation
21     rmb(X,C,P) :- $\sigma_1(X,L2)$, length(L2,P), 0 =:= P.   % rough membership: $\mu(x) = \frac{|C \cap \sigma(X)|}{|\sigma(X)|}$
22     rmb(X,C,P) :- dom(X), findall(Z, call(C,Z), L1), list_to_set(L1, S1),
23                   $\sigma_1(X,L2)$, list_to_set(L2, S2), intersection(S1, S2, S3),
24                   length(S2, N1), length(S3, N2), P is N2/N1.
25     pos(X,C) :- rmb(X,C,P), alpha(A), P >= A.     % Region operators
26     bnd(X,C) :- rmb(X,C,P), alpha(A), beta(B), B < P, P < A.
27     neg(X,C) :- rmb(X,C,P), beta(B), P =< B.
knowledge base:
28     P :: wwR(X) :- rmb(X, ww, P).                  % Probability that X is in rough set wwR
29     lowerWW(X) :- $dom_1(X)$, lower(X, ww).        % Approximation specialization for ww
30     upperWW(X) :- $dom_1(X)$, upper(X, ww).
31     posWW(X) :- $dom_1(X)$, pos(X, ww).
32     bndWW(X) :- $dom_1(X)$, bnd(X, ww).
33     negWW(X) :- $dom_1(X)$, neg(X, ww).
34     0.60 :: willLike(P,R) :- rating(R,X), X >= 4, hun(P, yes).   % Probabilistic clauses for willLike
35     0.25 :: willLike(P,R) :- rating(R,X), X = 3, hun(P, yes).
36     0.15 :: willLike(P,R) :- rating(R,X), X < 3, hun(P, yes).
```

- In Lines 2–3, constants α and β are bound to 0.5, respectively. Symmetric bounds are used for this example so $\beta = 1 - \alpha$.
- Lines 4–5 specify two domains: for Table 2 and Table 4.
- Lines 7–9 specify schemata for the actual table data.
- Line 11 defines the crisp set $ww(X)$ of interest. Since it is not definable relative to the indiscernibility relation σ based on the two attributes *hun* and *type*, a rough set correlate $wwR(X)$ will be defined in Line 28 with its lower and upper approximations and regions (Lines 29–33).
- In Line 12, base facts about the base relation $\sigma(X,Y)$ are defined using the relation $s(X,Y)$. Pairs of individuals having the same values for attributes *hun* and *type* are in $s(X,Y)$.
- Lines 14–17 specify the properties of $\sigma(X,Y)$ and generate the transitive closure of $s(X,Y)$, since we are interested in an equivalence relation among individuals.
- Line 18 specifies the useful relation $\sigma_1(X,L)$, which given an individual X, returns a list L containing all individuals in X's equivalence class $\sigma(X) = [X]$.
- Lines 19–20 specify the generic relations for the lower and upper approximations of the rough set $wwR(X)$ ($lower(X,C)$ and $upper(X,C)$, respectively). The parameter C is a relation for a crisp set. In the example, it is *ww*.

- Lines 21–24 define the rough membership relation $rmb(X, C, P)$ which computes the rough membership function $\mu(x) = \frac{|C \cup \sigma(X)|}{|\sigma(X)|}$. This provides the probability that a randomly chosen individual in X's equivalence class $\sigma(X)$, belongs to the rough set correlate $wwR(X)$ of $C = ww(X)$. For the sake of clarity, we denote this relation by *rmb* since *mu* is reserved for a generic definition of μ, which will be provided in Program 6.
- Lines 25–27 specify the generic region relations for a rough set, taking its crisp set correlate C as an input parameter.
- Line 28 in the knowledge base, specifies the probabilistic clause that defines the probability that an individual is in the rough set $wwR(X)$, in terms of $rmb(X, ww, P)$.
- Lines 29–30 define the instantiations of the generic relations $lower(X, C)$ and $upper(X, C)$ for the crisp relation *ww*.
- Lines 31–33 define instantiations of the generic relations $pos(X, C), bnd(X, C)$ and $neg(X, C)$, representing the positive, boundary and negative regions for *ww*, respectively.
- Lines 24–36 specify probabilistic clauses for the relation $willLike(P, R)$ which provides the probability a person P will like a restaurant R depending on whether P is hungry and the rating of R.

Remark 6.2.

It is important to emphasize the use of PROLOG meta-predicates in the definition of the rough membership relation $rmb(X, C, P)$ which takes a relation C as input parameter, and uses the meta-predicates *findall*/3 and *call*/2. Since the main approximation relations, $lower(X, C), upper(X, C), pos(X, C), neg(X, C)$, and $bnd(X, C)$, are all defined in terms of $rmb(X, C, P)$, this provides an elegant, compact, higher-order relational technique for dealing with multiple numbers of rough set specifications through simple instantiation. In our case, the crisp set of interest, $C = ww$ is used throughout. Later in the paper, even more use will be made of such meta-predicates and higher-order programming techniques. □

Given this specification, one can now combine crisp, probabilistic and rough probabilistic relations in additional rules to reason about individuals and their restaurant preferences. Here are some examples. The relation $willmeet(X, Y, R)$ is defined in terms of a crisp relation $type(Y, R)$ and a rough probabilistic relation $wwR(Y)$:

$$willmeet(X, Y, R) :- type(Y, R), wwR(Y). \tag{32}$$

The query 'query(willmeet(jim,Y,thai))', will return the information that *jim* will meet individuals x_2, x_4 and x_8 with probability 0.667 and will meet individual x_{11} with probability 0.0. This makes sense since the probability of *jim* meeting someone at a *thai* restaurant is restricted to those individuals willing to wait to go to a *thai* restaurant, where waiting is determined by the probabilistic rough relation $wwR(Y)$. The next rule uses the *subquery* relation and conditionalizes the conclusion on the probability that *Y*′*s* rough membership in *wwR* will be greater than 0.7.

$$willmeet(X, Y, R) :- type(Y, R), subquery(wwR(Y), P), P > 0.7. \tag{33}$$

In this case, 'query(willmeet(jim,X,R)' will return information that $willmeet(jim, x1, french)$ and $willmeet(jim, x12, burger)$ with probability 1.0. If one replaces $P > 0.7$ with $P > 0.6$, one can additionally conclude $willmeet(jim, x2, thai), willmeet(jim, x4, thai)$ and $willmeet(jim, x8, thai)$, also with probability 1.0.The final query in this example uses a rule that combines standard relations, $typeOf(Y, R)$ and $willmeet(X, Y, R, T)$, a probabilistic relation $willLike(P, R)$, and a rough probabilistic relation $wwR(P)$:

$$willmeet(X, P, R, T) :- typeOf(R, T), willLike(P, R), wwR(P). \tag{34}$$

In this case, suppose *jim* wants to meet a client x_{12} at a Chinese restaurant. What choice of Chinese restaurant would result in the highest probability of them meeting given the behavior of x_{12}? The query 'query(willmeet(jim, x12, R, chinese)' would result in the following choices $willmeet(jim, x12, r12, chinese), willmeet(jim, x12, r14, chinese)$, with probabilities 0.6 and 0.25. Consequently, *jim* should choose restaurant *r*12. Note that these probabilities do not sum up to 1.0 since probabilistic relation $willLike(P, R)$ is involved.

6.2. Decision rules

A common application of rough set theory is rule induction via the generation of reducts. Rather than delve into rule induction in the general case, this section shows how one can define decision rules for a concept *c*, based on the three regions $POS(c), NEG(c)$, and $BND(c)$. Yao [45] considers a semantically sound means of constructing three-way decision rules for a concept *c* based on the regions.Recall in Example 6.1, that one constructed a variable precision probabilistic approximation space $A_{VP(\alpha)} = \langle dom, \sigma, \alpha \rangle$, where σ was derived from Table 2. The table can be described as an information system which is a formal counterpart of a table:

$$I = \langle dom, At, \{V_a | a \in At\}, \{I_a | a \in At\} \rangle, \tag{35}$$

where *dom* is a finite empty set of objects, *At* is a set of attributes, V_a are the value domains for each attribute in *At*, and $I_a : U \rightarrow V_a$ is an information function which maps an object to its attribute value, for each attribute *a*. The information table associated with the part of Table 2 used in Example 3.9 is,

$$I_{ww} = \langle dom, \{Hun, Type, WillWait\}, \{yes, no\}, \{French, Thai, Burger, Italien\}\},$$
$$\{I_{Hun}, I_{Type}, I_{WillWait}\}\rangle.$$

The indiscernibility relation, σ in $A_{VP(\alpha)}$ was defined using the subset of attributes $A = \{Hun, Type\} \subseteq At$ as follows

$$\sigma(x, y) \equiv \forall a \in A(I_a(x) = I_a(y)),$$

where two objects are in the same equivalence class if they share the same values for attributes A.

For each equivalence class in the quotient set dom/σ associated with $A_{VP(\alpha)}$, one can provide a logical definition in terms of conjunctions of attribute/value pairs. For instance,

$$\begin{aligned} Des(\sigma_{Hun=yes,Type=French}) &= Hun = yes \wedge Type = French \\ &\vdots \\ Des(\sigma_{Hun=no,Type=Burger}) &= Hun = no \wedge Type = Burger \\ Des(WW) &= WillWait = Yes \end{aligned}$$

$Des(WW)$ describes the target set of all individuals that are willing to wait. Let $[x] \in dom/\sigma$. In the classical case, if $x \in POS(WW)$, it belongs to WW with certainty. If $x \in NEG(WW)$, it does not belong to WW with certainty. If $x \in BND(WW)$, if can not be decided with certainty whether or not it belongs to WW. Consequently, one has three types of decision rules:

- $Des([x]) \rightarrow_P Des(c)$ for $[x] \subseteq POS(c)$;
- $Des([x]) \rightarrow_N Des(c)$ for $[x] \subseteq NEG(c)$;
- $Des([x]) \rightarrow_B Des(c)$ for $[x] \subseteq BND(c)$.

A more fine-grained approach introduces the notion of relative certainty. There are many ways to do this, but for this discussion, the following will be used, where P, N, and B stand for positive, negative, and boundary region, respectively, $\Theta \in \{P, N, B\}$ and cf is a function that returns the certainty factor for a decision rule,

$$cf(Des([x]) \rightarrow_\Theta Des(c)) = \mu_{c,\sigma}(x) = \frac{|c \cap \sigma(x)|}{|\sigma(x)|} = P(c|\sigma(x)). \quad (36)$$

cf can be understood as a confidence value for concluding the right-hand side of the rule. This results in three types of decision rules,

- $Des([x]) \xrightarrow{cf}_P Des(c)$;
- $Des([x]) \xrightarrow{cf}_N Des(c)$;
- $Des([x]) \xrightarrow{cf}_B Des(c)$.

In the first two cases, $cf = 1.0$, and $cf = 0.0$, respectively, so with absolute certainty, one can be confident that for $y \in [x], y \in c$ and not $y \in c$, respectively. For the third rule, $0.0 < cf < 1.0$, so for any $y \in [x]$, one can not say anything with certainty.

In the probabilistic case, the certainty factor for all three rule types would be in the interval $0.0 \leqslant cf \leqslant 1.0$, so the question then becomes what rule type should be applied in what situation. The following approach (one of many) would take additional probabilistic information into account in determining which rule to use:

- $Des([x]) \xrightarrow{cf}_P Des(c)$ if $cf \geqslant \alpha$;
- $Des([x]) \xrightarrow{cf}_B Des(c)$ if $1 - \alpha < cf < \alpha$;
- $Des([x]) \xrightarrow{cf}_N Des(c)$ if $cf \leqslant 1 - \alpha$.

The following are some instantiations of these decision rules for Example 3.9:[5]

[5] For clarity, we write $cf > \alpha$ rather than $cf \geqslant \alpha + \epsilon$, etc. (see Remark 3.5).

$$\begin{aligned}
&Des(\sigma_{Hun=yes,Type=French}) \overset{cf}{\rightarrow}_P Des(WW) \text{ if } cf > 0.5 \Rightarrow \\
&\qquad H(x) = yes \wedge Type(x) = French \overset{1.0}{\rightarrow}_P WillWait(x) = Yes; \\
&Des(\sigma_{Hun=yes,Type=Italien}) \overset{cf}{\rightarrow}_B Des(WW) \text{ if } cf = 0.5 \Rightarrow \\
&\qquad H(x) = yes \wedge Type(x) = Italien \overset{2/3}{\rightarrow}_B unknown; \\
&Des(\sigma_{Hun=yes,Type=Thai}) \overset{cf}{\rightarrow}_N Des(WW) \text{ if } cf < 0.5 \Rightarrow \\
&\qquad H(x) = yes \wedge Type(x) = Thai \overset{1/3}{\rightarrow}_N (WillWait(x) = Yes).
\end{aligned}$$

The first rule is always safe to use since $cf = 1.0$ and it will always be correct when used. The second rule is relatively safe since $1.0 - 1/3 = 2/3$, so statistically it will be correct two times out of three. Recall that an N rule asserts the conclusion is not true. The last rule should not be used due to lack of information. It's precondition is in the boundary region of the concept *WillWait*.

These decision rules can then be encoded in PROBLOG and then used to reason about the probabilistic rough set *WW*.

Example 6.3. In this example, the PROBLOG code required is shown as Program 4. Extending the PROBLOG code in Program 3 for the decision rule encoding is straightforward and compact. In the full code (see A.1), one does not require a rule for each equivalence class. There is one rule each for positive, negative, and boundary regions, respectively, where the α check is built into the definitions for $pos(X,C), neg(X,C)$ and $bnd(X,C)$. $rmb(X,C,P)$ implements $\mu_{c,\sigma}(x)$. $dom1(X)$ is a domain check to restrict individuals to the table data being processed.

Program 4: PROBLOG program encoding decision rules from Example 6.3.

```
1  drP(X, C, Cf) :- dom1(X), pos(X, C), rmb(X, C, P), Cf is P.
2  drN(X, C, Cf) :- dom1(X), neg(X, C), rmb(X, C, P), Cf is P.
3  drB(X, C, Cf) :- dom1(X), bnd(X, C), rmb(X, C, P), Cf is P.
4  query(drP(_, ww, Cf)).
5  query(drN(_, ww, Cf)).
6  query(drB(_, ww, Cf)).
```

Each of the queries in Lines 4–6 results in the facts about the rules' conclusions that are true, and each fact also returns the certainty factor associated with application of the decision rule:

```
drP(x1,ww,1.0), drP(x2,ww,0.67), drP(x4,ww,0.67), drP(x8,ww,0.67), drP(x12,ww,1.0);
drN(x3,ww,0.33), drN(x5,ww,0.0), drN(x7,ww,0.33), drN(x9,ww,0.33), drN(x11,ww,0.0);
drB(x6,ww,0.5), drB(x10,ww,0.5).
```

7. Generalization toward tolerance spaces

There are many commonsense scenarios where viewing similarity or indiscernibility between individuals as an equivalence relation does not apply, in particular in regard to transitivity. For example [23], in Fechner's weight-lifting sensitivity example, it is given that one has a set of small weights (in grams), x, y and z, each weighing 10, 11 and 12 grams, respectively. Let τ be a tolerance relation. $\tau(x,y)$ and $\tau(y,z)$, but $\neg\tau(y,z)$. In other words, weights differing by 1 gram are indistinguishable from one another when held in the palms of both hands, while weights differing by 2 or more grams are distinguishable. Another example is *resemblance* [39], where just because Jim resembles Fred and Fred resembles Frank does not imply that Jim resembles Frank.So far, the approaches considered have focused on approximation spaces, $AS_{DT(\alpha,\beta)} = \langle dom, \sigma, P, \alpha, \beta \rangle$, where the base relation σ is an equivalence relation. The generalization to tolerance spaces removes the constraint of transitivity placed on the base relation σ as considered in Section 2.2. The paper [36] provides a detailed discussion on theory and choices when using tolerance spaces in the rough set context.

Definition 7.1 (*Tolerance relation, tolerance space, tolerance neighborhood*). A *tolerance relation* τ on a set *dom*, is a relation $\tau \subseteq dom \times dom$, that is reflexive and symmetric. A *tolerance space* $TS = \langle dom, \tau \rangle$, consists of a domain *dom* and a tolerance relation τ. The *tolerance neighborhood of x wrt* τ, denoted n_τ, is $n_\tau(x) \overset{\text{def}}{=} \{y | \tau(x,y)\}$.

Note that given a tolerance space *TS*, the family of equivalence classes generated from σ and their partition of *dom* used in approximation spaces is generalized to a family of neighborhoods generated from τ. The set $\{n_\tau(x) : x \in dom\}$ is a covering of *dom*. Generally, to derive τ, one assumes an application dependent allowed tolerance between individuals x, y, in order to determine neighborhoods. For example, one may consider:

$$y \in n_\tau(x) \overset{\text{def}}{\equiv} |\phi(x) - \phi(y)| < \epsilon_x, \tag{37}$$

where ϕ is a measurement function associated with the domain of interest. One can assume a tolerance for each individual x_i, or a common tolerance ϵ for the whole domain *dom*. Another way to state this is,

$$\forall x, y \in dom\,(\tau(x,y) \equiv |\phi(x) - \phi(y)| < \epsilon).$$

When appropriate, the notation $TS_\tau = \langle dom, \tau, \phi, \epsilon\rangle$ can be used to make the measurement function ϕ and the tolerance constant ϵ explicit.

Given a neighborhood operator n_τ, one can also derive its tolerance relation τ:

$$\forall x, y \in dom\left(\tau(x,y) \overset{\text{def}}{\equiv} y \in n_\tau(x)\right).$$

Assuming a measurement function ϕ and a tolerance constant ϵ,

$$\forall x, y \in dom\,(\tau(x,y) \equiv y \in n_\tau(x) \equiv |\phi(x) - \phi(y)| < \epsilon).$$

Definition 7.2 (*Tolerance-based rough membership*). Given $TS = \langle dom, \tau\rangle$, and $c \subseteq dom$, the *tolerance-based rough membership function* with regard to c is

$$\mu_{c,\tau}(x) \overset{\text{def}}{=} P(c|n_\tau(x)) = \frac{|c \cap n_\tau(x)|}{|n_\tau(x)|}.$$

Tolerance-based approximations are straightforward generalizations of rough approximations where equivalence classes and partitions are replaced by neighborhoods and coverings. It is important to note that an individual x, can now be a member of more than on neighborhood. This contrasts with rough approximation spaces where an individual has one unique equivalence class it belongs to.

Definition 7.3 (*Classical tolerance-based approximations*). Given $AS_P = \langle dom, \tau, P\rangle$, and $c \subseteq dom$, the *lower* and *approximation* of c wrt τ are defined by:

$$c^+_\tau \overset{\text{def}}{=} \left\{x \mid \mu_{c,\tau}(x) = 1.0\right\} = \{x \mid P(c \mid n_\tau(x)) = 1.0\}; \tag{38}$$

$$c^\oplus_\tau \overset{\text{def}}{=} \left\{x \mid \mu_{c,\tau}(x) > 0.0\right\} = \{x \mid P(c \mid n_\tau(x)) > 0.0\}. \tag{39}$$

The difference $c^\oplus_\tau \setminus c^+_\tau$ is called the *tolerance-based boundary region* of c wrt τ.

It is then relatively straightforward to generalize tolerance approximations to tolerance-based probabilistic approximations with a qualification. As pointed out, since it can be the case that an individual can belong to more than one neighborhood, for a particular tolerance space, this has to be taken into account when defining what tolerance-based probabilistic approximations mean semantically.

Recall that the intuition behind the probabilistic rough set methods that have been encountered so far is that the degree of overlap of an individual's neighborhood with a target set c determines whether that individual is in the lower or upper approximation of c. Conceptually, the focus is on neighborhoods in the boundary region of c. If a neighbor N overlaps with a degree greater than α then it is defined as being in the lower approximation. In a similar manner, if a neighbor N overlaps with a degree greater than or equal to β then it is defined as being in the upper approximation. Suppose an individual belongs to two neighborhoods, N_1 and N_2. For N_1, the overlap is greater than α, but for N_2, the overlap is less than β. In this case, the individual would be defined as being in both the positive and negative region of the target set c, which does not make sense and is undesirable.

In order to remedy this situation, the definitions of tolerance-based probabilistic approximation operators have to be constrained to rule out such situations. Let's focus on the definition of lower approximation since similar arguments apply to the upper approximation. Rather than define membership in a lower approximation c^+_τ as $\left\{x|\mu_{c,\tau}(x) \geqslant \alpha\right\}$, an additional constraint is added as follows:

$$c^{P+}_\tau \overset{\text{def}}{=} \left\{x|\forall y\left(x \in n_\tau(y) \rightarrow \mu_{c,\tau}(y) \geq \alpha\right)\right\}. \tag{40}$$

By the symmetry of tolerance relations, $x \in n_\tau(y)$ is equivalent to $y \in n_\tau(x)$. Therefore, the additional constraint (40) states that for an individual x to be in the lower approximation of c, the degree of overlap of c with any neighborhood for which x is a member, must be greater than α. A similar argument applies for the definition of the upper approximation. This additional constraint ensures that an individual can only belong to one region of a rough set.

Definition 7.4 (*Tolerance-based probabilistic approximations*). Let $0.0 \leqslant \beta < \alpha \leqslant 1.0$. Given $AS_{DT(\alpha,\beta)} = (dom, \tau, P, \alpha, \beta)$, where τ is a tolerance relation, the *decision-theoretic tolerance-based lower and upper approximation(s) wrt* α, β *and* τ are defined by:

$$c_{\tau}^{D_+} \overset{\text{def}}{=} \left\{x | \forall y \left(x \in n_{\tau}(y) \rightarrow \mu_{c,\tau}(y) \geq \alpha\right)\right\} = \{x | \forall y \,(x \in n_{\tau}(y) \rightarrow P(c|n_{\tau}(y)) \geq \alpha)\}; \tag{41}$$

$$c_{\tau}^{D_\oplus} \overset{\text{def}}{=} \left\{x | \forall y \left(x \in n_{\tau}(y) \rightarrow \mu_{c,\tau}(y) > \beta\right)\right\} = \{x | \forall y \,(x \in n_{\tau}(y) \rightarrow P(c|n_{\tau}(y)) > \beta)\}. \tag{42}$$

The difference $c_{\tau}^{D_\oplus} \setminus c_{\tau}^{D_+}$ is called the *tolerance-based probabilistic boundary region* of c wrt τ.

8. Some generalizations of probabilistic rough sets

In Section 7, the generalization of the equivalence relation σ to a tolerance relation τ was considered and it was shown that defining tolerance-based probabilistic approximation operators was relatively straightforward. In essence, this approach works for any binary relation $\upsilon : U \times U$ that generates a covering for U. Since in our paper υ is defined to satisfy at least reflexivity (**T**), it is serial, which guarantees that for any set $c, c_{\sigma}^{D_+} \subseteq c_{\sigma}^{D_\oplus}$. **T** also guarantees that the relation υ generates a covering for $U, \forall y \exists x \,\upsilon(x, y)$.

So far, definitions of similarity or tolerance relations υ, have been defined qualitatively. In the following generalization, this property is relaxed. Assume a binary tolerance relation τ based on measuring the quantitative resemblance between two individual images in terms of probabilities. For instance, Table 5 below shows the probabilistic resemblance relation between a set of visual images given the following probabilistic facts about the resemblance relation:

$0.4 :: \tau_c(vi_1, vi_2)$. $0.8 :: \tau_c(vi_1, vi_3)$. $0.7 :: \tau_c(vi_1, vi_4)$.
$0.5 :: \tau_c(vi_2, Vi_3)$. $0.9 :: \tau_c(vi_2, vi_4)$. $0.6 :: \tau_c(vi_3, vi_4)$.

The PROBLOG Program 5, shows how such tables can be generated in a straightforward manner.

Program 5: The PROBLOG program which can be used to compute probabilities in Table 5.

```
crisp/probabilistic set(s):
    dom(vi_1). dom(vi_2). dom(vi_3). dom(vi_4).
crisp/probabilistic base relation(s):
    0.4 :: τ_c(vi_1, vi_2).   0.8 :: τ_c(vi_1, vi_3).   0.7 :: τ_c(vi_1, vi_4).
    0.5 :: τ_c(vi_2, vi_3).   0.9 :: τ_c(vi_2, vi_4).   0.6 :: τ_c(vi_3, vi_4).
    τ_vi(X, Y) :- τ_c(X, Y).
    τ_vi(X, X) :- dom(X).                            % property T for τ_vi
    τ_vi(X, Y) :- dom(X), dom(Y), τ_vi(Y, X).        % property B for τ_vi
    query(τ_vi(X, Y)).
```

Given this technique, one can reason about probabilistic similarity or tolerance relations υ, using PROBLOG. Given that neighborhoods are defined in terms of υ, Definition 7.1 would also need to be generalized in the following manner.

Definition 8.1 (*Probabilistic neighborhood(s)*). Let $0.0 \leqslant \gamma \leqslant 1.0$. Given $AS_{DT(\alpha,\beta,\gamma)} = \langle dom, \upsilon, P, \alpha, \beta, \gamma \rangle$ and $x \in dom$, by the *probabilistic neighborhood of x wrt* γ, we mean the set:

$$n_{\upsilon,\gamma}(x) \overset{\text{def}}{=} \{y | P(\langle x, y \rangle \in \upsilon) \geqslant \gamma\}.$$

Definition 8.2 (*Probabilistic neighbor-based rough membership*). Let $0.0 \leqslant \gamma \leqslant 1.0$ and $n_{\upsilon,\gamma}(x)$ be a probabilistic neighborhood parameterized using γ. Given $AS_{DT(\alpha,\beta,\gamma)} = \langle dom, \upsilon, P, \alpha, \beta, \gamma \rangle$, for $c \subseteq dom$, the *probabilistic neighbor-based rough membership function* with regard to c is:

$$\mu_{c,\upsilon,\gamma}(x) = P(c | n_{\upsilon,\gamma}(x)) = \frac{|c \cap n_{\upsilon,\gamma}(x)|}{|n_{\upsilon,\gamma}(x)|}. \tag{43}$$

Let us now show that $\mu_{c,\upsilon,\gamma}(x)$ can be understood as a probability distribution.

Table 5
Image resemblance.

τ_{vi}	Vi_1	Vi_2	Vi_3	Vi_4
Vi_1	1.0	0.4	0.8	0.7
Vi_2	0.4	1.0	0.5	0.9
Vi_3	0.8	0.5	1.0	0.6
Vi_4	0.7	0.9	0.6	1.0

Proposition 8.3. *Let $AS_{DT(\alpha,\beta,\gamma)} = \langle dom, \upsilon, P, \alpha, \beta, \gamma\rangle$, where $\upsilon \subseteq dom \times dom$ is a (probabilistic) base relation constrained by* **T** *together with a subset of the properties* $\{\mathbf{B}, \mathbf{4}, \mathbf{5}\}$, *and $n_{\upsilon,\gamma}(x)$ are (probabilistic) neighborhoods derived from γ. Let $P'_{\upsilon,\gamma}(x \in c) \overset{\text{def}}{=} \mu_{c,\upsilon,\gamma}(x)$. Then $P'_{\upsilon,\gamma}$ satisfies the Kolmogorov probability axioms.*

Proof. Note that by axiom **T**, for every $x \in dom, n_{\upsilon,\gamma}(x) \neq \emptyset$, so $|n_{\upsilon,\gamma}(x)| \neq 0.0$.

Axiom 1: for all $x \in dom$, and $c \subseteq dom, P'(x \in c) \geqslant 0.0$:

$$P'(x \in c) \overset{\text{def}}{=} P(c | n_{\upsilon,\gamma}(x)) \geq 0.0.$$

Axiom 2: for every $x \in dom, P'(x \in dom) = 1.0$:

$$P'(x \in dom) \overset{\text{def}}{=} \mu_{dom,\upsilon,\gamma}(x) = \frac{|dom \cap n_{\upsilon,\gamma}(x)|}{|n_{\upsilon,\gamma}(x)|} = \frac{|n_{\upsilon,\gamma}(x)|}{|n_{\upsilon,\gamma}(x)|} = 1.0.$$

Axiom 3: for $c_1, c_2 \subseteq dom$ such that $c_1 \cap c_2 = \emptyset$, and for every $x \in dom, P'(x \in c_1 \cup c_2) = P'(x \in c_1) + P'(x \in c_2)$:

$$P'(x \in c_1 \cup c_2) \overset{\text{def}}{=} \mu_{c_1 \cup c_2,\upsilon,\gamma}(x) = \frac{|(c_1 \cup c_2) \cap n_{\upsilon,\gamma}(x)|}{|n_{\upsilon,\gamma}(x)|} = \frac{|\big(c_1 \cap n_{\upsilon,\gamma}(x)\big) \cup \big(c_2 \cap n_{\upsilon,\gamma}(x)\big)|}{|n_{\upsilon,\gamma}(x)|}.$$

Since $c_1 \cap c_2 = \emptyset$, also $\big(c_1 \cap n_{\upsilon,\gamma}(x)\big) \cap \big(c_2 \cap n_{\upsilon,\gamma}(x)\big) = \emptyset$. Therefore:

$$|\big(c_1 \cap n_{\upsilon,\gamma}(x)\big) \cup \big(c_2 \cap n_{\upsilon,\gamma}(x)\big)| = |c_1 \cap n_{\upsilon,\gamma}(x)| + |c_2 \cap n_{\upsilon,\gamma}(x)|.$$

Hence,

$$P'(x \in c_1 \cup c_2) = \frac{|\big(c_1 \cap n_{\upsilon,\gamma}(x)\big) \cup \big(c_2 \cap n_{\upsilon,\gamma}(x)\big)|}{|n_{\upsilon,\gamma}(x)|} = \frac{|c_1 \cap n_{\upsilon,\gamma}(x)| + |c_2 \cap n_{\upsilon,\gamma}(x)|}{|n_{\upsilon,\gamma}(x)|} =$$
$$\frac{|c_1 \cap n_{\upsilon,\gamma}(x)|}{|n_{\upsilon,\gamma}(x)|} + \frac{|c_2 \cap n_{\upsilon,\gamma}(x)|}{|n_{\upsilon,\gamma}(x)|} = P'(x \in c_1) + P'(x \in c_2).$$

The following definition provides a general and abstract means of considering many probabilistic rough set methods and will serve as the basis for specifying such methods in PROBLOG.

Definition 8.4 (*Generalized probabilistic approximation space and operators*). Let $0.0 \leqslant \beta < \alpha \leqslant 1.0$ and $0.0 \leqslant \gamma \leqslant 1.0$. Then $GAS = \langle dom, \upsilon, \alpha, \beta, \gamma \rangle$ is a *generalized probabilistic approximation space*, where $\upsilon \subseteq dom \times dom$ is a (probabilistic) base relation constrained by **T** together with a subset of the properties $\{\mathbf{B}, \mathbf{4}, \mathbf{5}\}$, and $n_{\upsilon,\gamma}(x)$ are (probabilistic) neighborhoods derived from γ. Additionally, given a set $c \subseteq dom$, the *generalized lower and upper approximation(s) wrt α, β, γ and υ* are defined by:

$$c^{G_+}_{\upsilon,\gamma} \overset{\text{def}}{=} \Big\{ x | \forall y \Big(x \in n_{\upsilon,\gamma}(y) \to \mu_{c,\upsilon,\gamma}(x) \geq \alpha \Big) \Big\} = \{ x | \forall y (x \in n_{\upsilon,\gamma}(y) \to P(c | n_{\upsilon,\gamma}(x)) \geq \alpha) \}; \tag{44}$$

$$c^{G_\oplus}_{\upsilon,\gamma} \overset{\text{def}}{=} \Big\{ x | \forall y \Big(x \in n_{\upsilon,\gamma}(y) \to \mu_{c,\upsilon,\gamma}(x) > \beta \Big) \Big\} = \{ x | \forall y (x \in n_{\upsilon,\gamma}(y) \to P(c | n_{\upsilon,\gamma}(x)) > \beta) \}. \tag{45}$$

The difference $c^{G_\oplus}_{\upsilon,\gamma} \setminus c^{G_+}_{\upsilon,\gamma}$ is called the *generalized boundary region* of c wrt υ.

Lemma 8.5. Definition 8.4 subsumes Definitions 3.2, 3.3, 3.4, 3.6, 3.8, 7.3 and 7.4, where each definition can be instantiated by the relevant values for υ, α, β, and γ, in addition to choosing the elementary granule types used, as shown in Table 6. □

Table 6
Approximation operators (ϵ is explained in Remark 3.5, which also applies to Bayesian approximations).

Type	Definition	υ	c	α	β	γ	Granule
Classical Approx. Operators	Def. 3.2	**TB4**, crisp	crisp	1.0	0.0	0.0	$n_{\upsilon,\gamma}(x) = \sigma(x)$
0.5-Approx. Operators	Def. 3.3	**TB4**, crisp	crisp	$0.5 + \epsilon$	$0.5 - \epsilon$	0.0	$n_{\upsilon,\gamma}(x) = \sigma(x)$
Variable Precision Approx. Operators (symmetric bounds)	Def. 3.4	**TB4**, crisp	crisp	$0.5 \leqslant \alpha \leqslant 1.0$	$1 - \alpha$	0.0	$n_{\upsilon,\gamma}(x) = \sigma(x)$
Decision-theoretic Approx Operators	Def. 3.6	**TB4**, crisp	crisp	$0.0 \leqslant \alpha \leqslant 1.0$	$0.0 \leqslant \beta \leqslant 1.0$	0.0	$n_{\upsilon,\gamma}(x) = \sigma(x)$
Bayesian Rough Sets Approx. Operators	Def. 3.8	**TB4**, crisp	crisp	$P(c) + \epsilon$	$P(c) - \epsilon$	0.0	$n_{\upsilon,\gamma}(x) = \sigma(x)$
Classical Tolerance-based Approx. Operators	Def. 7.3	**TB**, crisp	crisp	1.0	0.0	0.0	$n_{\upsilon,\gamma}(x) = n_\tau(x)$
Tolerance-based Probabilistic Approx. Operators	Def. 7.4	**TB**, crisp	crisp	$0.0 \leqslant \alpha \leqslant 1.0$	$0.0 \leqslant \beta \leqslant 1.0$	0.0	$n_{\upsilon,\gamma}(x) = n_\tau(x)$
Generalized Probabilistic Approx. Operators	Def. 8.4	**T**, plus any subset of {**B**, **4**, **5**}; crisp or probabilistic	crisp or probabilistic	$0.0 \leqslant \alpha \leqslant 1.0$	$0.0 \leqslant \beta \leqslant 1.0$	$0.0 \leqslant \gamma \leqslant 1.0$	$n_{\upsilon,\gamma}(x)$

8.1. Implementing generalized approximation operators in PROBLOG

PROBLOG offers the opportunity to define second-order relations through the use of meta-predicates such as 'call' and 'apply', in addition to *set-predicates* such as 'findall', all which can take relations as arguments. This expressivity provides a compact and elegant means for specifying and reasoning about generalized probabilistic approximation spaces and operators as defined in Definition 8.4. This section provides an overview of the library of meta-predicates constructed for such specifications.

Program 6: Generic definitions of rough membership, approximations and regions.

```
 1  :- use_module(library(lists)).
 approximations:
 2  neighborhood(X, Upsilon, Gamma, S) :-             % generic neighborhood function
 3          findall(Y, (subquery(call(Upsilon, X, Y), P), call(Gamma, G), P>=Gamma), L),
 4          list_to_set(L, S).
 5  mu(X, C, Upsilon, Gamma, Dom, P)  :-              % generic rough membership mu/6
 6          call(Dom, X), neighborhood(X, Upsilon, Gamma, S),
 7          length(S, P), P =:= 0.
 8  mu(X, C, Upsilon, Gamma, Dom, P)  :-
 9          call(Dom, X), findall(Z, call(C, Z), L1), list_to_set(L1, S1),
10          neighborhood(X, Upsilon, Gamma, S2), intersection(S1, S2, S3),
11          length(S2, N1), length(S3, N2), P is N2/N1.
12  P::mu(X, C, Upsilon, Gamma, Dom)  :-              % generic rough membership mu/5
13          mu(X, C, Upsilon, Gamma, Dom, P).
14  lapprox(X, C, Upsilon, Alpha, Gamma, Dom) :-      % generic lower approximation
15          call(Dom, X), mu(X, C, Upsilon, Gamma, Dom, P),
16          call(Alpha, A), P>=A.
17  uapprox(X, C, Upsilon, Beta, Gamma, Dom) :-       % generic upper approximation
18          call(Dom, X), mu(X, C, Upsilon, Gamma, P),
19          call(Beta, B), P>B.
20  pos(X, C, Upsilon, Alpha, Gamma, Dom) :-          % generic positive region for C
21          call(Dom, X), lapprox(X, C, Upsilon, Alpha, Gamma, Dom).
22  bnd(X, C, Upsilon, Alpha, Beta, Gamma, Dom) :-    % generic boundary region for C
23          call(Dom, X), uapprox(X, C, Upsilon, Beta, Gamma, Dom),
24          \+ lapprox(X, C, Upsilon, Alpha, Gamma, Dom).
25  neg(X, C, Upsilon, Beta, Gamma, Dom) :-           % generic negative region for C
26          call(Dom, X), \+ uapprox(X, C, Upsilon, Beta, Gamma, Dom).
```

Program 6 provides the generic relations used specifying generalized approximation spaces and operators. The PROBLOG program in A.2 provides executable code based on this specification. The conjunction 'call(Upsilon, X, Y), P), call(Gamma, G), P>=Gamma' specified in subquery in Line 3 of Program 6 first determines 'P' as the probability of $\upsilon(X, Y)$ and then makes sure that the probability is greater than or equal to γ, as required in Definition 8.1.

We have the following lemma.

Lemma 8.6. [Correctness] *Assuming that the relation υ is at least reflexive, Program 6 correctly implements generalized probabilistic approximation spaces as formalized in Definition 8.4.*

Assume for each relation c_i of interest, the following:

- A relation $dom_i(X)$, a base binary relation, $upsilonBase_i(X, Y)$ and a binary relation$upsilon_i(X, Y)$ defined from $upsilonBase_i(X, Y)$;
- A crisp relation $c_i \subseteq dom_i(X)$;
- Relations $alpha_i(X), beta_i(X)$, and $gamma_i(X)$, for values α_i, β_i and γ_i, for each c_i.

Given these relations for each c_i, one can instantiate each of the generic relations provided above to generate the proper general approximation spaces for each c_i. These instantiations can then be used for constructing PROBLOG theories that combine the use of many different types of relations in heterogeneous rules, as discussed in Section 9.

Remark 8.7. It is also worth emphasizing that Bayesian Rough Sets are covered by our implementation in the sense that Alpha and Beta used in Program 6 should be set to the probability 'P(C)' where 'C' is the target set specified as an argument (with the adjustment by ϵ as discussed in Remark 3.5 and shown in Table 6).□

9. Case study: recommendation

The following case study will use a number of the new generalized features that have been proposed in Section 8. In particular, for the concept *Destinations* used in the case study, a tolerance space will be used and its base relation will be defined as a probabilistic relation. Additionally, for the concept *Customers*, a classical approximation space will be used and its base relation will also be defined as a probabilistic relation. In the case study, a toy recommendation system will be specified that provides advice to travel customers as to where a good place to travel might be based on that customers similarity with other customers and their likes and dislikes.

Let's begin with the customer domain:

$$dom_c(eve).\quad dom_c(jack).\quad dom_c(kate).\quad dom_c(mark). \tag{46}$$

Let's assume that partial data exists on pairwise similarity between individuals in the customer domain based on previous surveys and customer history, where each of the customers fits into a customer segment based on such features as tourists traveling with families, single tourists, health tourists, etc. For instance,

$$\begin{array}{lll} 0.4 :: s_c(eve, jack). & 0.8 :: s_c(eve, kate). & 0.7 :: s_c(eve, mark). \\ 0.5 :: s_c(jack, kate). & 0.9 :: s_c(jack, mark). & 0.6 :: s_c(kate, mark). \end{array} \tag{47}$$

The model will also assume that customer segmentation creates a partition. Consequently, the base relation for the *Customer* concept will be an equivalence relation. This creates a probabilistic approximation space $AS = \langle dom_c, \sigma_c, P \rangle$.

In order to take advantage of the generic relations defined in Program 6, the approximation space AS will be recast as a generalized probabilistic approximation space. The generalized probabilistic approximation space for customers is

$$GAS_c = \langle dom_c, \sigma_c, \alpha_c, \beta_c, \gamma_c \rangle \tag{48}$$

where $\alpha_c = 0.9, \beta_c = 0.2$ and $\gamma_c = 0.5$, and σ_c has the properties **T** (reflexivity), **B** (symmetry) and **4** (transitivity).

Using the customer data in Eq. (47) and Program 7 below, the following closure of σ_c would result in, see Table 7.

Program 7: The PROBLOG program which can be used to compute probabilities in Table 7.For a runnable PROBLOG code see respective parts of A.2.

crisp/probabilistic set(s):

```
dom(eve).  dom(jack).  dom(kate).  dom(mark).
```

crisp/probabilistic base relation(s):

```
0.4 :: s_c(eve, jack).    0.8 :: s_c(eve, kate).     0.7 :: s_c(eve, mark).
0.5 :: s_c(jack, kate).   0.9 :: s_c(jack, mark).    0.6 :: s_c(kate, mark).
σ_c(X, Y) :- s_c(X, Y).
σ_c(X, X) :- dom(X).                              % property T for σ_c
σ_c(X, Y) :- dom(X), dom(Y), σ_c(Y, X)            % property B for σ_c
σ_c(X, Y) :- dom(Y), s_c(X, Z), σ_c(Z, Y).        % property 4 for σ_c
query(σ_c(X, Y)).
```

In a similar manner, a resemblance relation between destinations can be defined. The destination domain consists of:

$$dom_d(kerala).\quad dom_d(marseille).\quad dom_d(quebec).\quad dom_d(suzhou).\quad dom_d(venice). \tag{49}$$

The base relation τ_d, will be defined as a tolerance relation, where the following partial data exists about τ_d, concerning the pairwise resemblance of destinations to each other:

$$\begin{array}{lll} 0.0 :: s_d(kerala, marseille). & 0.0 :: s_d(kerala, quebec). & 0.8 :: s_d(kerala, suzhou). \\ 0.6 :: s_d(kerala, venice). & 0.9 :: s_d(marseille, quebec). & 0.0 :: s_d(marseille, suzhou). \\ 0.05 :: s_d(marseille, venice). & 0.0 :: s_d(quebec, suzhou). & 0.07 :: s_d(quebec, venice). \\ 0.8 :: s_d(suzhou, venice). & & \end{array} \tag{50}$$

The generalized probabilistic approximation space for destinations is:

$$GAS_d = \langle dom_d, \tau_d, \alpha_d, \beta_d, \gamma_d \rangle, \tag{51}$$

where $\alpha_d = 0.8, \beta_d = 0.1$, and $\gamma_d = 0.7$, and τ_d has the properties **T** (reflexivity), and **B** (symmetry).

Using the destination data in Eq. (50), the following closure of σ_d would result in probabilities shown in Table 8.

Given any subset of customers in dom_c, one can define lower and upper approximations for that subset using GAS_c and Definition 8.4. In a similar manner, given any subset of destinations in dom_d, one can define lower and upper approximations for that subset using GAS_d and Definition 8.4.

Given the probabilities in Table 7 and Table 8, one can express rules for calculating success probabilities of recommendations which would be part of the knowledge base for a PROBLOG program, e.g.,

Table 7
Indiscernibility (σ_c) among customers.

σ_c	Eve	Jack	Kate	Mark
Eve	1.0	0.86	0.9	0.9
Jack	0.86	1.0	0.54	0.9
Kate	0.9	0.54	1.0	0.6
Mark	0.9	0.9	0.6	1.0

Table 8
Resemblance among destinations using s_d and properties of σ_d.

σ_d	Kerala	Marseille	Quebec	Suzhou	Venice
Kerala	1.0	0.0	0.0	0.8	0.6
Marseille	0.0	1.0	0.9	0.0	0.05
Quebec	0.0	0.9	1.0	0.0	0.07
Suzhou	0.8	0.0	0.0	1.0	0.8
Venice	0.6	0.05	0.07	0.8	1.0

$$rec(X,Y) : -customer(X), on\_offer(Y), \tag{52}$$
$$visited(X,Y_1), \sigma_d(Y,Y_1), Y \neq Y_1, \tag{53}$$
$$rec(X,Y) : -customer(X), \tag{54}$$
$$visited(X_1,Y_1), \tag{55}$$
$$\sigma_c(X,X_1), X \neq X_1, \tag{56}$$
$$\sigma_d(Y,Y_1), Y \neq Y_1. \tag{57}$$

That is, one recommends destination Y to customer X when:

- Lines (52)–(53): Y is on–offer as a destination and X visited a place Y_1, which has a resemblance to Y; or
- Lines (54)–(57): a customer X_1 visited a destination Y_1 such that customers X and X_1 are in the same equivalence class associated with σ_c (representing the same tourist market segment), where $X_1 \neq X$, and the destinations Y, Y_1, resemble each other wrt σ_d where $Y \neq Y_1$.

PROBLOG definitions of *rec* are provided in the section **knowledge base** of the PROBLOG program included in A.2.
Let's assume that:

- there are two current customers of interest, Kate and Mark;
- Quebec, Suzhou and Venice are destinations currently on–offer;
- similarity and tolerance relations are specified in Tables 7 and 8 for customers and definitions, respectively;
- and, in a given situation it is uncertain which destinations should be selected to offer as recommendations for each client. So, for every destination Y on–offer, $on\_offer(Y)$ is selected with probability 0.5. Of course, given additional statistical knowledge about customers' preferences, the probabilities could be suitably adjusted to reflect that knowledge.

Table 9 shows the results of the PROBLOG program when executed on $query(rec(\_,\_))$ (which would return probabilities for all customer/destination pairs:

The relation *rec* is probabilistic due to probabilistic relations in the body of rules defining *rec*. In order to obtain crisp recommendations, one could define relations such (c_rec) in terms of rough properties of the customer class such as the lower or upper approximation:

$$c\_rec(X,Y) : -customer(X), to\_offer^{D+}_{\sigma_d}(Y). \tag{58}$$

The rule in Line (58) results in recommending to each customer X a destination Y when Y is in the lower approximation of *on_offer* (interpreted as *certainly* on offer). Using data from the case study, $query(c\_rec(kate,Y))$ as well as $query(c\_rec(mark,Y))$ is true for $Y = venice$.

A complete, executable PROBLOG program is provided for the case study described in this section in A.2. In that program, relations specific to destinations are indexed with d and those specific to customers are indexed with c.

10. Conclusions

This paper has proposed a general framework and programming methodology for specifying and reasoning about probabilistic rough sets using PROBLOG. It provides a definition of generalized probabilistic approximation spaces and operators that subsumes many of the proposed approaches in the literature. This definition is used as a basis for specification of probabilis-

Table 9
Probabilities of *rec* computed by program enclosed in A.2.

rec	Quebec	Shuzhou	Venice
Kate	0.9	0.94	0.83
Mark	0.56	0.92	0.84

tic rough set methods in PROBLOG. Although the focus is on probabilistic rough set methods, the definitions subsume non-probabilistic rough set methods too. The general framework can be leveraged not only for use with existing approaches to probabilistic rough set methods but also with new approaches, due to its generality. The use of meta-predicates in PROBLOG facilitates efficient and compact specification of these methods in a principled manner with a principled programming methodology. The framework proposed offers a powerful tool for not only application oriented activity, but also as a research tool. Examples of the latter shown in the paper, are the generalization of rough target sets to partially specified, probabilistic target sets, the generalization of base relations as probabilistic base relations and the study of probabilistic tolerance spaces.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Appendix A. PROBLOG codes for examples

A.1. Executable PROBLOG Program for Example 6.1

```
% A program for Basic Probabilistic Rough Set Reasoning (Example 6.1)

:- use_module(library(lists)).

%---------------------------------------
% constants:
%---------------------------------------

alpha(0.5).          % threshold alpha (see Definition 3.6)
beta(0.5).           % threshold beta  (see Definition 3.6)

%---------------------------------------
% crisp/probabilistic set(s):
%---------------------------------------

% specification of domain(s):

dom1(x1). dom1(x2). dom1(x3). dom1(x4). dom1(x5). dom1(x6).
dom1(x7). dom1(x8). dom1(x9). dom1(x10). dom1(x11). dom1(x12).

dom2(r1). dom2(r2). dom2(r3). dom2(r4). dom2(r5). dom2(r6).
dom2(r7). dom2(r8). dom2(r9). dom2(r10). dom2(r11). dom2(r12).
dom2(r13). dom2(r14). dom2(r15).

dom(X) :- dom1(X).
dom(X) :- dom2(X).
person(X) :- dom1(X).
restaurant(X) :- dom2(X).

% Table 2 and Table 4 Data

wwait(x1, yes). wwait(x3, yes). wwait(x4, yes).
wwait(x6, yes). wwait(x8, yes). wwait(x12, yes).

wwait(x2, no). wwait(x5, no). wwait(x7, no).
wwait(x9, no). wwait(x10, no). wwait(x11, no).
```

```
hun(x1, yes). type(x1, french). hun(x2, yes).  type(x2, thai).
hun(x3, no).  type(x3, burger). hun(x4, yes).  type(x4, thai).
hun(x5, no).  type(x5, french). hun(x6, yes).  type(x6, italien).
hun(x7, no).  type(x7, burger). hun(x8, yes).  type(x8, thai).
hun(x9, no).  type(x9, burger). hun(x10, yes). type(x10,italien).
hun(x11, no). type(x11,thai).   hun(x12, yes). type(x12, burger).

typeOf(r1, french).    rating(r1, 4).  typeOf(r2, thai).     rating(r2, 3).
typeOf(r3, burger).    rating(r3, 2).  typeOf(r4, thai).     rating(r4, 4).
typeOf(r5, franch).    rating(r5, 5).  typeOf(r6, italien).  rating(r6, 3).
typeOf(r7, burger).    rating(r7, 1).  typeOf(r8, thai).     rating(r8, 5).
typeOf(r9, japanese).  rating(r9, 4).  typeOf(r10, steak).   rating(r10,5).
typeOf(r11, korean).   rating(r11,3).  typeOf(r12, chinese). rating(r12, 4).
typeOf(r13, japanese). rating(r13, 5). typeOf(r14, chinese). rating(r14, 3).
typeOf(r15, italien).  rating(r15, 2).

%input a target set ww of individuals.
% we are interested in its rough approximation wwR

ww(X) :- wwait(X,yes).

%-------------------------------------
% crisp/probabilistic base relation(s):
%-------------------------------------

% Equivalence classes

sigma_(X,Y) :-  hun(X, yes), type(X, french), hun(Y, yes), type(Y, french).
sigma_(X,Y) :-  hun(X, yes), type(X, thai), hun(Y, yes), type(Y, thai).
sigma_(X,Y) :-  hun(X, yes), type(X, burger), hun(Y, yes), type(Y, burger).
sigma_(X,Y) :-  hun(X, yes), type(X, italien), hun(Y, yes), type(Y, italien).

sigma_(X,Y) :-  hun(X, no), type(X, french), hun(Y, no), type(Y, french).
sigma_(X,Y) :-  hun(X, no), type(X, thai), hun(Y, no), type(Y, thai).
sigma_(X,Y) :-  hun(X, no), type(X, burger), hun(Y, no), type(Y, burger).
sigma_(X,Y) :-  hun(X, no), type(X, italien), hun(Y, no), type(Y, italien).

% Equivalence relation properties (for dom1)

sigma(X,Y) :- sigma_(X,Y).                      % sigma includes sigma_

sigma(X,X) :- dom1(X).                          % reflexivity (T)
sigma(X,Y) :- dom1(X),dom1(Y),sigma_(Y,X).      % symmetry (B)
sigma(X,Y) :- sigma_(X,Z), dom1(Z), sigma(Z,Y). % transitivity (4)

% returns a list containing individuals in sigma(X)

sigma1(X,L) :- findall(Y,subquery(call(sigma,X,Y),_),L1), list_to_set(L1,L).

%-------------------------------------
```

```
% approximations:
%--------------------------------------

% generic lower and upper approximation operators
% take a crisp target set C as input.

lower(X,C) :- rmb(X,C,P), alpha(A), P>A.
upper(X,C) :- rmb(X,C,P), beta(B), P>=B.

% rough Membership
% Computes the probability of X's membership in the rough set associated with C.
% mu(x) = |C cup sigma(X)| / |sigma(X)|

rmb(X,C,P) :- sigma1(X,L2), length(L2,P), 0 =:= P.  %check for 0 in denominator
rmb(X,C,P) :- dom(X), findall(Z, call(C,Z),L1), list_to_set(L1,S1),
              sigma1(X,L2), list_to_set(L2,S2), intersection(S1,S2,S3),
              length(S2,N1), length(S3,N2), P is N2/N1.

pos(X,C) :- rmb(X,C,P), alpha(A), P>=A.
bnd(X,C) :- rmb(X,C,P), alpha(A), beta(B), B<P, P<A.
neg(X,C) :- rmb(X,C,P), beta(B), P=<B.

%--------------------------------------
% knowledge base:
%--------------------------------------

% Rough Probabilistic clause
% The probability individual X is in the rough set wwR()
% derived from the target set ww.

P::wwR(X) :- dom1(X), rmb(X,ww,P).

% Specialized (unary) predicates for crisp set ww.

lowerWW(X) :- dom1(X), lower(X,ww).
upperWW(X) :- dom1(X), upper(X,ww).
posWW(X) :- dom1(X), pos(X,ww).
negWW(X) :- dom1(X), neg(X,ww).
bndWW(X) :- dom1(X), bnd(X,ww).
rmbWW(X,P) :- dom1(X), rmb(X,ww,P).

% Standard Probabilistic clauses

0.60::willLike(P,R) :- rating(R,X), X>=4, hun(P,yes).
0.25::willLike(P,R) :- rating(R,X), X=3,  hun(P,yes).
0.15::willLike(P,R) :- rating(R,X), X<3,  hun(P,yes).

% Example clauses and example queries from the paper

willmeet(X,P,R,T) :- typeOf(R,T), willLike(P,R), wwR(P).
query(willmeet(jim,x12,R,chinese)).
```

```
% willmeet(X,Y,R) :- type(Y,R), wwR(Y).
% query(willmeet(jim,Y,thai)).

% willmeet(X,Y,R) :- type(Y,R), subquery(wwR(Y), P), P>0.7.
% query(willmeet(jim,Y,R)).

% query(ww(X)).      query(wwR(X)).
% query(rmbWW(X,P)).
% query(lowerWW(X)). query(upperWW(X)).
% query(posWW(X)).   query(negWW(X)). query(bndWW(X)).
```

A.2. PROBLOG Program for the Case Study

```
% A program for generic definitions (Program 6) and case study (Section 9)

%% Begin Generic Part (Program 6)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

:- use_module(library(lists)).

neighborhood(X, Upsilon, Gamma, S) :-
        findall(Y, (subquery(call(Upsilon, X, Y), P), call(Gamma,G), P>=G), L),
        list_to_set(L, S).

mu(X, C, Upsilon, Gamma, Dom, P) :-
        call(Dom,X), neighborhood(X, Upsilon, Gamma, S),
        length(S, P), P=:=0.

mu(X, C, Upsilon, Gamma, Dom, P) :-
        call(Dom, X), findall(Z, call(C, Z), L1), list_to_set(L1, S1),
        neighborhood(X, Upsilon, Gamma, S2),
        intersection(S1, S2, S3),
        length(S2, N1), length(S3, N2), P is N2/N1.

P::mu(X, C, Upsilon, Gamma, Dom) :-mu(X, C, Upsilon, Gamma, Dom, P).

lapprox(X, C, Upsilon, Alpha, Gamma, Dom) :-
        call(Dom,X), mu(X, C, Upsilon, Gamma, Dom,P),
        call(Alpha,A), P>=A.

uapprox(X, C, Upsilon, Beta, Gamma, Dom) :-
        call(Dom, X), mu(X, C, Upsilon, Gamma, Dom, P),
        call(Beta, B), P>B.

pos(X, C, Upsilon, Alpha, Gamma, Dom) :-
        call(Dom,X), lapprox(X, C, Upsilon, Alpha, Gamma, Dom).

bnd(X, C, Upsilon, Alpha, Beta, Gamma, Dom) :-
        call(Dom,X),  uapprox(X, C, Upsilon, Beta, Gamma, Dom),
        \+lapprox(X, C, Upsilon, Alpha, Gamma, Dom).
```

```
neg(X, C, Upsilon, Beta, Gamma, Dom) :-
        call(Dom,X), \+uapprox(X, C, Upsilon, Beta, Gamma, Dom).

%% End Generic Part
%%%%%%%%%%%%%%%%%%%%

%% customers

dom_c(eve).  dom_c(jack).  dom_c(kate). dom_c(mark).

0.4::s_c(eve, jack).  0.8::s_c(eve, kate).   0.7::s_c(eve, mark).
0.5::s_(jack, kate).  0.9::s_c(jack, mark).  0.6::s_c(kate, mark).

sigma_c(X,Y):- s_c(X,Y).                          % indiscernibility among customers

sigma_c(X,X):- dom_c(X).   % reflexivity (T)
sigma_c(X,Y):- dom_c(X), dom_c(Y), sigma_c(Y, X).  % symmetry (B)
sigma_c(X,Y):- s_c(X,Z), dom_c(Y), sigma_c(Z, Y).  % transitivity (4)

alphaC(0.9). betaC(0.2). gammaC(0.5).

%% Input set

cR(eve). cR(mark).

lowerCust(X) :- lapprox(X, cR, sigmaC, alphaC, gammaC, domC).
upperCust(X) :- uapprox(X, cR, sigmaC, betaC, gammaC, domC).

% define a set C and use as argument.

lowerCust(X,C) :- lapprox(X, C, sigmaC, alphaC, gammaC, domC).
upperCust(X,C) :- uapprox(X, C, sigmaC, betaC, gammaC, domC).

%% Destinations

alphaD(0.8). betaD(0.1). gammaD(0.7).

dom_d(kerala). dom_d(marseille).  % domain of destinations
dom_d(quebec). dom_d(suzhou).  dom_d(venice).

0.0::s_d(kerala,marseille).  0.0::s_d(kerala,quebec).
0.8::s_d(kerala,suzhou).     0.6::s_d(kerala,venice).
0.9::s_d(marseille,quebec).  0.0::s_d(marseille,suzhou).
0.05::s_d(marseille,venice). 0.0::s_d(quebec,suzhou).
0.07::s_d(quebec,venice).    0.8::s_d(suzhou,venice).

sigma_d(X,Y):- s_d(X,Y).           % similarity among destinations
sigma_d(X,X):- dom_d(X).           % reflexivity
sigma_d(X,Y):- dom_d(X), dom_d(Y), % symmetry
               sigma_d(Y,X).
```

```
%% Input set

dR(venice). dR(suzhou).

lowerDest(X) :- lapprox(X, dR, sigma_d, alphaD, gammaD,dom_d).
upperDest(X) :- uapprox(X, dR, sigma_d, betaD, gammaD,dom_d).
posDest(X) :- pos(X,dR, sigma_d, alphaD, gammaD,dom_d).
negDest(X) :- neg(X,dR, sigma_d, betaD, gammaD,dom_d).
bndDest(X) :- bnd(X,dR, sigma_d, alphaD, betaD, gammaD,dom_d).

% define a set C and use as argument.

lowerDest(X,C) :- lapprox(X, C, sigma_d, alphaD, gammaD,dom_d).
upperDest(X,C) :- uapprox(X, C, sigma_d, betaD, gammaD,dom_d).
posDest(X,C) :- pos(X, C, sigma_d, alphaD, gammaD,dom_d).
negDest(X,C) :- neg(X, C, sigma_d, betaD, gammaD,dom_d).
bndDest(X,C) :- bnd(X, C, sigma_d, alphaD, betaD, gammaD,dom_d).

%% Knowledge Base

visited(eve,kerala).     visited(eve,suzhou). % places visited by customers
visited(jack,quebec).    visited(jack,venice).
visited(kate,marseille). visited(kate,venice).
visited(mark,quebec).

on_offer(suzhou).             % destinations currently in offer
on_offer(quebec).
on_offer(venice).

customer(kate).  customer(eve).   % current customers

eq_c(X,X):- dom_c(X).  % equality on dom_c
eq_d(X,X):- dom_d(X).  % equality on dom_d

rec(X,Y):- customer(X),
           on_offer(Y),
           visited(X,Y1),
           sigma_d(Y,Y1), not eq_d(Y,Y1).

rec(X,Y):- customer(X),
           on_offer(Y),
           visited(X1,Y1),
           sigma_c(X,X1), not eq_c(X,X1),
           sigma_d(Y,Y1), not eq_d(Y,Y1).

c_rec(X,Y) :- customer(X),          % lower approximation
           lowerDest(Y,on_offer).   % of 'on_offer' wrt sigma_d


%% Queries for testing
%%%%%%%%%%%%%%%%%%%%%




    query(sigma_c(X,Y)).   % computes values in Table 7
    query(sigma_d(X,Y)).   % computes values in Table 8

    query(rec(X,Y)).       % computes values in Table 9
    query(c_rec(X,Y)).     % computes results related to Equation (57)
```

References

[1] C. Baral, M. Gelfond, J.N. Rushton, Probabilistic reasoning with answer sets, TPLP 9 (2009) 57–144.
[2] D. Ciucci, D. Dubois, Three-valued logics, uncertainty management and rough sets, in: J.F. Peters, A. Skowron (Eds.), Transactions on Rough Sets XVII, Springer, 2014, pp. 1–32.
[3] S. Costa, D. Page, M. Quazi, J. Cussens, CLP(BN): Constraint logic programming for probabilistic knowledge, in: L. De Raedt, P. Frasconi, K. Kersting, S. Muggleton (Eds.), Probabilistic Inductive Logic Programming - Theory and Applications, Springer, 2008, pp. 156–188.
[4] L. De Raedt, K. Kersting, S. Natarajan, D. Poole, in: Statistical Relational Artificial Intelligence: Logic, Probability, and Computation Synthesis Lectures on Artificial Intelligence and Machine Learning, Morgan & Claypool Pub, 2016.
[5] L. De Raedt, A. Kimmig, Probabilistic (logic) programming concepts, Mach. Learn. 100 (2015) 5–47.
[6] L. De Raedt, A. Kimmig, H. Toivonen, ProbLog: A probabilistic Prolog and its application in link discovery, in: Proc. of the 20th IJCAI, Morgan Kaufmann Pub. Inc., 2007, pp. 2468–2473. .
[7] P. Doherty, W. Łukaszewicz, A. Skowron, A. Szałas, Knowledge Representation Techniques. A Rough Set Approach, in: volume 202 of Studies in Fuziness and Soft Computing. Springer, 2006.
[8] P. Doherty, A. Szałas, On the correspondence between approximations and similarity, in: Tsumoto, S., Słowiński, R., Komorowski, H.J., Grzymala-Busse, J. (Eds.), Proc. 4th Conf. RSCTC Rough Sets and Current Trends in Computing, Springer, 2004, pp. 143–152. .
[9] P. Doherty, A. Szałas, Rough set reasoning using answer set programs, Int. J. Approximate Reasoning 130 (2021) 126–149.
[10] M. Gebser, R. Kaminski, B. Kaufmann, T. Schaub, Answer Set Solving in Practice. Synthesis Lectures on Artificial Intelligence and Machine Learning, Morgan and Claypool Pub, 2012.
[11] A.D. Gordon, T.A. Henzinger, A.V. Nori, S.K. Rajamani, Probabilistic programming, in: Proc. Future of Software Engineering, ACM, 2014, pp. 167–181. .
[12] S. Greco, B. Matarazzo, R. Słowiński, Parameterized rough set model using rough membership and Bayesian confirmation measures, Int. J. Approximate Reasoning 49 (2008) 285–300.
[13] J.W. Grzymala-Busse, P.G. Clark, M. Kuehnhausen, Generalized probabilistic approximations of incomplete data, Int. J. Approximate Reasoning 55 (2014) 180–196.
[14] J.D. Katzberg, W. Ziarko, Variable precision rough sets with asymmetric bounds, in: W.P. Ziarko (Ed.), Rough Sets, Fuzzy Sets and Knowledge Discovery, Springer, 1994, pp. 167–177.
[15] A. Kimmig, B. Demoen, L. De Raedt, V.S. Costa, R. Rocha, On the implementation of the probabilistic logic programming language ProbLog, Theory and Practice of Logic Programming 11 (2011) 235–262.
[16] J. Lee, Z. Yang, Lp$^{\text{mln}}$, weak constraints, and P-log, in: Proc. 31st AAAI Conf., 2017, pp. 1170–1177. .
[17] O. Lenz, D. Peralta, C. Cornelis, fuzzy-rough-learn 0.1: A Python library for machine learning with fuzzy rough sets, in: Bello, R., Miao, D., Falcon, R., Nakata, M., Rosete, A., Ciucci, D. (Eds.), Proc. IJCRS 2020, Springer, 2020, pp. 491–499. .
[18] C. Luo, T. Tianrui Li, Y. Yao, Dynamic probabilistic rough sets with incomplete data, Inf. Sci. 417 (2017) 39–54.
[19] Z. Pawlak, Rough Sets. Theoretical Aspects of Reasoning about Data, Kluwer Academic Pub, 1991.
[20] Z. Pawlak, A. Skowron, Rough membership functions: a tool for reasoning with uncertainty, Banach Center Pub. 28 (1993) 135–150.
[21] Z. Pawlak, S.K.M. Wong, W. Ziarko, Rough sets: Probabilistic versus deterministic approach, Int. J. Man-Mach. Stud. 29 (1988) 81–95.
[22] J. Pearl, Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference, Morgan Kaufmann Pub. Inc., 1988.
[23] J. Peters, P. Wasilewski, Tolerance spaces: Origins, theoretical aspects and applications, Inf. Sci. 195 (2012) 211–225.
[24] A. Pfeffer, Practical Probabilistic Programming, Manning Pub. Co., 2016.
[25] L. Polkowski, Rough Sets. Mathematical Foundations. volume 15 of Advances in Intelligent and Soft Computing, Physica-Verlag, 2002. .
[26] L. Polkowski, A. Skowron, Rough mereology: A new paradigm for approximate reasoning, Int. J. Approximate Reasoning 15 (1996) 333–365.
[27] D. Poole, The independent choice logic and beyond, in: L. De Raedt, P. Frasconi, K. Kersting, S. Muggleton (Eds.), Probabilistic Inductive Logic Programming – Theory and Applications, Springer, 2008, pp. 222–243.
[28] ProbLog-Team, 2021. ProbLog: Applications and datasets. URL: https://dtai.cs.kuleuven.be/problog/applications.html. [Online; accessed 29-Sept-2021]. .
[29] ProbLog-Team, 2021. ProbLog tutorial example: Bayesian networks: First-order. URL: https://dtai.cs.kuleuven.be/problog/tutorial.html. [Online; accessed 19-Mar-2021]. .
[30] L. Riza, A. Janusz, C. Bergmeir, C. Cornelis, F. Herrera, D. Ślęzak, J. Benítez, Implementing algorithms of rough set theory and fuzzy rough set theory in the R package "RoughSets", Inf. Sci. 287 (2014) 68–89.
[31] S. Russell, P. Norvig, Artificial Intelligence: A Modern Approach, fourth ed., Pearson, 2020.
[32] H. Sakai, Software tools for RNIA (rough sets non-deterministic information analysis), 2021. URL: http://www.mns.kyutech.ac.jp/∼sakai/RNIA/. [Online; accessed 24-Sept-2021]. .
[33] H. Sakai, M. Nakata, W.Z. Wu, D. Miao, G. Wang, Rough sets and data mining, CAAI Transactions on Intelligence Technology (Special Issue) 4 (2019) 201–260.
[34] T. Sato, A statistical learning method for logic programs with distribution semantics, in: Proc 12th Int. Conf. on Logic Programming ICLP, MIT Press, 1995, pp. 715–729.
[35] T. Sato, Y. Kameya, PRISM: A language for symbolic-statistical modeling, in: Proc. of the 15th IJCAI, Morgan Kaufmann, 1997, pp. 1330–1339. .
[36] D. Ślęzak, P. Wasilewski, Granular sets – foundations and case study of tolerance spaces, in: An, A., Stefanowski, J., Ramanna, S., Butz, C.J., Pedrycz, W., Wang, G. (Eds.), Rough Sets, Fuzzy Sets, Data Mining and Granular Computing, Springer, 2007, pp. 435–442..
[37] D. Ślęzak, W. Ziarko, The investigation of the Bayesian rough set model, Int. J. Approximate Reasoning 40 (2005) 81–91.
[38] R. Słowiński, D. Vanderpooten, A generalized definition of rough approximations based on similarity, IEEE Trans. Knowl. Data Eng. 12 (2000) 331–336.
[39] A. Sossinsky, Tolerance space theory and some applications, Acta Applicandae Mathematicae 5 (1986) 137–167.
[40] J. Van Benthem, Correspondence theory, in: D. Gabbay, F. Guenthner (Eds.), Handbook of Philosophical Logic, D. Reidel Pub, Co, 1984, pp. 167–247.
[41] J. Vennekens, M. Denecker, M. Bruynooghe, Representing causal information about a probabilistic process, in: M. Fisher, W. van der Hoek, B. Konev, A. Lisitsa (Eds.), Proc. Logics in AI, 10th JELIA., Springer, 2006, pp. 452–464.
[42] W.Y. Wang, K. Mazaitis, W.W. Cohen, Programming with personalized pagerank: a locally groundable first-order probabilistic logic, in: Q. He, A. Iyengar, W. Nejdl, J. Pei, R. Rastogi (Eds.), 22nd ACM Int. Conf. on Information and Knowledge Management, 2013, pp. 2129–2138.
[43] S.K.M. Wong, W. Ziarko, Comparison of the probabilistic approximate classification and the fuzzy set model, Fuzzy Sets Syst. 21 (1987) 357–362.
[44] Y.Y. Yao, Probabilistic rough set approximations, Int. J. Approximate Reasoning 49 (2008) 255–271.
[45] Y.Y. Yao, Three-way decisions with probabilistic rough sets, Inf. Sci. 180 (2010) 341–353.
[46] Y.Y. Yao, S.K.M. Wong, A decision theoretic framework for approximating concepts, Int. J. Man-Mach. Stud. 37 (1992) 793–809.
[47] Y.Y. Yao, S.K.M. Wong, T.Y. Lin, A review of rough set models, in: T.Y. Lin, N. Cercone (Eds.), Rough Sets and Data Mining, Springer, 1997, pp. 47–75.
[48] Q. Zhang, Q. Xie, G. Wang, A survey on rough set theory and its applications, CAAI Trans. Intell. Technol. 1 (2016) 323–333.
[49] W. Ziarko, Variable precision rough set model, J. Comput. Syst. Sci. 46 (1993) 39–59.
[50] W. Ziarko, Probabilistic approach to rough sets, Int. J. Approximate Reasoning 49 (2008) 272–284.