J Intell Inf Syst (2013) 41:187–210
DOI 10.1007/s10844-013-0241-z

Granular computing for relational data classification

Piotr Hońko

Received: 29 November 2012 / Revised: 15 February 2013 / Accepted: 18 February 2013 /
Published online: 8 March 2013

Abstract We propose a novel framework for generating classification rules from relational data. This is a specialized version of the general framework intended for mining relational data and is defined in granular computing theory. In the framework proposed in this paper we define a method for deriving information granules from relational data. Such granules are the basis for generating relational classification rules. In our approach we follow the granular computing idea of switching between different levels of granularity of the universe. Thanks to this a granule-based relational data representation can easily be replaced by another one and thereby adjusted to a given data mining task, e.g. classification. A generalized relational data representation, as defined in the framework, can be treated as the search space for generating rules. On account of this the size of the search space may significantly be limited. Furthermore, our framework, unlike others, unifies not only the way the data and rules to be derived are expressed and specified, but also partially the process of generating rules from the data. Namely, the rules can be directly obtained from the information granules or constructed based on them.

Keywords Multi-relational data mining · Database models · Granular computing · Classification

1 Introduction

The task of classification has extensively been studied in the field of data mining (see, e.g., Han et al. 2011; Tan et al. 2005; Banks et al. 2004). This issue has also been widely investigated for relational data (see, e.g., Džeroski and Lavrač 2001b; Zhen et al. 2009; Thangaraj and Vijayalakshmi 2011). One can indicate many different

P. Hońko (✉)
Department of Computer Science, Bialystok University of Technology,
Wiejska 45A, 15-351 Bialystok, Poland
e-mail: p.honko@pb.edu.pl

techniques and algorithms for classifying relational data; however, a unified framework for this task does not seem to have been introduced so far. Such a framework is needed for unifying operations that are independent of the technique or algorithm applied for classifying relational data. One can indicate the following essential operations that need to be unified: relational object representation, search space limitation and generation of relational patterns. These issues will briefly be discussed.

1. An object of a single-table database is represented by a tuple of table attribute values. An object of a database with a relational structure can be represented not only by a tuple that belongs to a table to be analyzed, but also by a certain part of the tuples of other tables that are directly or indirectly joined to the table under consideration. Therefore, relational object representation can vary depending on a given data mining task.
2. The search space for discovering relational patterns (e.g., relational classification rules) may be very huge. This problem is typically overcome by applying language bias which imposes some constraints on the patterns to be discovered, thereby the search space is limited. However, the search space, after such a limitation has been imposed on it, may still be large.
3. Rule-based classification is one of the most common classifying methods in data mining. The way of deriving rules from data is usually provided not by a given framework for mining relational data, but by a concrete algorithm that can be defined in the framework; therefore, the whole process of the generation of rules may be conducted from scratch each time any of the algorithm's parameters change.

The contribution of this paper is a framework for rule generation from relational data (Sections 5 and 6). It addresses the three above-mentioned issues and is a specialization of the general framework for mining relational data, which was proposed in Hońko (2013) and is described in Section 4. The general framework is defined in granular computing theory.

Granular computing (Pedrycz et al. 2008) can be viewed as a label of theories, methodologies, techniques, and tools that make use of granules in the process of problem solving (Yao 2000).

The general framework is based on an information system defined for relational data. Information granules derived from the information system are defined based on the notion of related sets, that is, sets of objects related (i.e., joined) to the objects to be analyzed. Such granules are the basis for discovering relational knowledge.

In the framework proposed in this paper, relational data is represented as a class of granules. Due to applying the granular computing idea of switching between different levels of granularity of the universe, the accuracy level of the data representation can be adjusted to a given data mining task, e.g., classification.

The generalized information granules obtained in our proposal can be viewed as an abstract representation of relational data. Such a representation is treated as the search space for discovering relational classification rules. Thanks to this, the size of the search space may be significantly limited.

Since the search space is constructed based on information granules, the process of discovering rules from the data is partially unified. Namely, rules can thus directly be received from such granules or constructed based on them.

In the following sections restate the existing frameworks for mining relational data (Section 2), briefly describe granular computing (Section 3), introduce the general framework for mining multi-relational data based on granular computing (Section 4), and propose and analyze a specific version of the framework intended for generating classification rules from relational data (Sections 5–7). The paper ends with concluding remarks (Section 8).

2 Related works

In this section we describe the general frameworks for mining relational data in the context of the generation of classification rules.

The research field within which the frameworks described below can be considered is multi-relational data mining (MRDM) (Džeroski and Lavrač 2001b; De Raedt 2008). MRDM concerns knowledge discovery from relational databases consisting of multiple relations (tables). MRDM aims to integrate methods from existing fields applied to an analysis of data represented by multiple relations, i.e. to produce new techniques for mining multi-relational data.

One can indicate two commonly used frameworks for mining relational data: inductive logic programming and relational database theory frameworks.

Early approaches for pattern discovery in relational data were defined in an inductive logic programming (ILP) framework (Džeroski and Lavrač 2001a).

ILP is a research field at the intersection of machine learning and logic programming. It provides a formal framework as well as practical algorithms for learning, in an inductive way, relational descriptions from data represented by target examples and background knowledge.

In ILP, data and induced patterns are represented as formulas in a first-order language. Data is stored in deductive databases, where relations can be determined extensionally as sets of ground facts and intensionally as sets of database clauses. Patterns are typically expressed as logic programs, i.e., sets of Horn clauses.

In ILP, the pattern structure is determined by the so-called declarative bias (Nédellec et al. 1996). It imposes some constraints on the patterns to be discovered. Thanks to this bias one can determine, among other things, such as which relations and how many times they may be used in patterns or how to replace a relation attribute with a variable or a constant.

The task of generating classification rules is carried out in ILP in the following way. A set E^+ of positive examples and a set E^- of negative examples of a target relation are given along with a background knowledge B which usually consists of Horn clauses. The task is to find a hypothesis H (a set of Horn clauses) that covers all of the positive examples and does not cover any negative example. H is equivalent to a set of classification rules.

An alternative framework (Knobbe 2006; Knobbe et al. 2000) for discovering patterns in relational data is defined in relational database theory (RDB). In a relational database, relations are usually determined extensionally as sets of tuples of constants; however, they can also be determined intensionally as sets of views. The relational patterns discovered in the relational database can be expressed as SQL queries.

Unlike in the ILP framework, a specification of the pattern structure is not required. Instead, the patterns are specified by the relationships that occur between the database entities, and are shown by the entity-relationship diagram. Alternatively, a class diagram, which is a part of the Unified Modeling Language (UML), is used to express bias (Knobbe et al. 2000). The UML class diagram shows how associations (i.e., structural relationships) between given classes (which correspond to the database tables) determine how objects in each class relate to objects in another class. Moreover, the multiplicities of associations are also examined. Such an association multiplicity provides information on how many objects in one class are related to a single object in another, and vice versa.

The task of generating a classification is performed in an analogous way as in the ILP framework. The set of tuples of the target relation is divided into two sets: a set of objects of the class for which rules are to be generated and a set of objects of the remaining classes. The task is to find a set of rules that are satisfied by all objects from the former set and are not satisfied by any object from the latter set.

The main difference between the two frameworks with respect to classification rule generation is in the way the data and rules are expressed. Therefore, one can easily adapt a rule generation mechanism from one framework to another.

In order to summarize the two frameworks in the context of the issues described in Section 1, i.e., relational object representation, search space limitation, and generation of relational patterns, we can mention the following features.

1. In each of the frameworks relational data is represented by a database with a relational structure, i.e., a deductive or relational database. Relational object representation is alternatively adjusted to a given data mining task not by the framework, but by a concrete algorithm.
2. The search space is limited in the frameworks only by bias or by the relationships that occur between the database entities. Furthermore, the search space is not explicitly given, but only determined by the constraints.
3. The frameworks unify the way the data and rules are described and specified. However, this unification does not concern the process of discovering rules from the data.

3 Granular computing

When analyzing data to discover knowledge, regardless of the tool used, we usually aggregate the objects with common features into the same clusters (i.e., groups). Such clusters can be treated as information derived from the database which is, in turn, the basis for the discovery of knowledge. The clusters can be obtained in a variety of ways depending, among others, on the task to be performed. Moreover, one can receive many different partitions of the universe, i.e., families of clusters, for the same task. The choice of the most proper partition can depend on which solution accuracy of the problem under consideration is sufficient enough. The challenge is thus to develop a framework for constructing and processing such clusters of data.

A field within which frameworks are developed for problem solving by the use of granules (e.g., clusters of data) is granular computing (GC) (Bargiela and Pedrycz 2003; Pedrycz et al. 2008). This is a relatively new, rapidly growing field of research (see, e.g., Zadeh 2010, 2011; Pedrycz and Chen 2011; Skowron et al. 2012). It can be viewed as a label of theories, methodologies, techniques, and tools that make use of granules in the process of problem solving (Yao 2000).

A granule is usually understood as a collection of entities drawn together by indistinguishability, similarity, proximity or functionality (Zadeh 1997). Therefore, a granule can be defined as any object, subset, class, or cluster of a given universe. The process of the formation of granules is called granulation. To clearly differentiate granulation from clustering, the semantic aspect of granular computing is taken into account. Therefore, we treat information granulation as a semantically meaningful grouping of elements based on their indistinguishability, similarity, proximity or functionality (Bargiela and Pedrycz 2008). To distinguish, in this work, granules obtained as a result of plain clustering from those obtained as a result of semantically meaningful grouping we will call the former *1-type information granules* and the latter *2-type information granules*.

Granulation can be performed by applying a top-down or bottom-up method. The former concerns the process of dividing a larger granule into smaller and lower-level granules, and the latter the process of forming a larger and higher-level granule with smaller and lower-level sub-granules (Yao 2005).

One can receive many granularities of the same universe which differ in their levels. A granule of high-level granularity, i.e., a high-level granule, represents a more abstract concept, and a low-level granule a more specific one. A basic task of GC is to switch between different levels of granularity. A more specific level granularity may reveal more detailed information. On the other hand, a more abstract level granularity may improve a problem's solution thanks to omitting irrelevant details.

Much research has been devoted to granular computing in data mining (see, e.g., Pedrycz et al. 2008; Bargiela and Pedrycz 2003; Lin 2005; Lin and Zadeh 2004; Skowron et al. 2012). However, a relatively small part of this research concerns the multi-relational aspect of data mining (see, e.g., Stepaniuk 2008; Hońko 2010, 2013). Moreover, a framework for generating classification rules from relational data by employing granular computing does not seem to have been introduced into scientific literature.

4 Granular computing framework for mining relational data

In this section we introduce the framework for mining relational data (Hońko 2013). The framework is constructed based on granular computing theory. We introduce a method for deriving information granules from relational data. Such granules are the basis for the discovery of knowledge of a different type.

4.1 Relational data

We assume that we are given relational data that resides in a relational database; however, our framework can also be defined for data stored in a deductive database. We also assume that data is alternatively transformed, i.e., operations such as attribute discretization, attribute aggregation, attribute reduction, or others, are applied.

Definition 1 (Relational database) A relational database can be defined in the context of MRDM by the following notions:

– A relation schema is an expression of the form $R(a_1, a_2, \ldots, a_n)$, where R is a relation name and a_i $(1 \leq i \leq n)$ are the attributes.

– A relation is a subset of the Cartesian product $V_{a_1} \times V_{a_2} \times \cdots \times V_{a_n}$, where V_{a_i} $(1 \le i \le n)$ are the value sets of attributes a_i.
– A relational database $D = T \cup B$ is a collection of logically connected relations, where $T = \{R_1^T, R_2^T, \ldots, R_{n_T}^T\}$ and $B = \{R_1^B, R_2^B, \ldots, R_{n_B}^B\}$ consist of target and background relations, respectively.

The target table (i.e., relation)[1] includes objects to be analyzed, e.g., objects for which classification rules are mined. Such objects may reside in more than one table; for example, each target table includes the objects of one class. Background tables include additional objects which are directly or indirectly joined to the objects of the target table. The same terms are used for the objects of the target and background tables, i.e., the target and background objects.

Example 1 We are given a database$D = \{customer\} \cup \{product, purchase\}$ for the customers of a grocery store.

Customer

c_id	Name	Class
1	Tom Jackson	1
2	Susan Clark	1
3	John Osborne	1
4	Adam Smith	0
5	Eve Lee	0
6	Ann Thompson	0

Product

p_id	Name	Price
1	bread	2.00
2	butter	3.50
3	milk	2.50
4	tea	5.00
5	coffee	6.00
6	cigarettes	6.50

Purchase

c_id	p_id	Amount	Date
1	1	1	21.09.2010
1	3	2	21.09.2010
2	1	1	25.09.2010
2	3	1	26.09.2010
4	6	1	26.09.2010
4	2	3	26.09.2010
5	5	2	27.09.2010
6	4	2	27.09.2010

The target table *customer* includes basic data about the customers. The data is divided into two groups according to the values of the attribute *class*. The background tables *product* and *purchase* include data about products purchased by the customers.

To examine objects apart from the tables they belong to, we use the notion of relational object.

[1]The notions of relation and table are used in this paper interchangeably.

Definition 2 (Relational object) We are given a database relation with the schema $R(a_1, a_2, \ldots, a_n)$. An expression of the form $R(v_1, v_2, \ldots, v_n)$ is an object of R if and only if $(v_1, v_2, \ldots, v_n)$ is a tuple of R.

For example, the first tuple of table *customer* from Example 1 is represented by the object *customer*(1, Tom Jackson, 1).

In our approach we represent a relational database by an information system that is constructed based on a standard information system (Pawlak 1991).

Definition 3 (Information system) An information system is a pair $IS = (U, A)$, where U is a non-empty finite set of objects, called the universe, and A is a non-empty finite set of attributes.

Now we are ready to construct the information system for storing relational data. We use D_T and D_B to denote, respectively, the sets of target and background relations of database $D = T \cup B$.

Let $U_{D_T} = \bigcup\limits_{R \in D_T} R$ and $U_{D_B} = \bigcup\limits_{R \in D_B} R$ be, respectively, the set of all target and background objects of database D. Subsequently, let $A_{D_T} = \bigcup\limits_{R \in D_T} A_R$[2] and $A_{D_B} = \bigcup\limits_{R \in D_B} A_R$ be, respectively, the set of all attributes of the target and background relations of database D.

We use the following representation of a relational database.

Definition 4 (Information system for a relational database) A relational database $D = T \cup B$ is represented by an information system $IS_D = (U_D, A_D)$[3], where

- $U_D = U_{D_T} \cup U_{D_B}$ is a non-empty finite set of objects, called the universe,
- $A_D = A_{D_T} \cup A_{D_B}$ is a non-empty finite set of attributes.

Example 2 Database D of Example 1 can be represented by information system $IS_D = (U_D, A_D)$, where $U_D = U_{D_T} \cup U_{D_B}$, $A_D = A_{D_T} \cup A_{D_B}$ are defined as follows:

$U_{D_T} = \{customer(1, Tom\ Jackson, 1), customer(2, Susan\ Clark, 1), \ldots,$
$customer(6, Ann\ Thompson, 0)\}$,
$U_{D_B} = \{product(1, bread, 2.00), product(2, butter, 3.50), \ldots,$
$product(6, cigarettes, 6.50),$
$purchase(1, 1, 1, 21.09.2010), purchase(1, 3, 2, 21.09.2010), \ldots,$
$purchase(6, 4, 2, 27.09.2010)\}$,
$A_{D_T} = \{customer.id, customer.name, customer.class\}$,
$A_{D_B} = \{product.p\_id, product.name, product.price, purchase.c\_id, purchase.p\_id,$
$purchase.quantity, purchase.date\}$.

[2] A_R denotes here the set of all attributes of relation R.

[3] The information system is a logical representation rather than a physical one.

4.2 Relational information

In our approach essential information acquired from the relational data are descriptions of target objects. These descriptions are used, in a sense, to identify the objects, i.e., the objects are compared to one another or to the patterns (e.g., classification rules) based on their descriptions. We construct a description for each target object based on the background relations. To construct such descriptions we introduce the notion of a related set (Hońko 2010).

Definition 5 (Related objects) Object o is related to object o' if and only if there exists a key attribute joining o with o'.[4]

In our approach the key attribute is, in general, understood as an important attribute for joining tables. It is usually a primary or foreign key. However, in some cases it can also be another attribute by which one table can be joined with another table or with itself.

A target object's description is expressed by a set of background objects joined with the target object. More precisely:

Definition 6 (Related set) A related set of a target object o, denoted by $rlt(o)$, is a set of background objects directly or indirectly related to the target object.

Each target object in our approach is processed along with its related set.

Example 3 Consider the target objects $o_1 = customer(1, Tom\ Jackson, 1), o_2 = customer(5, Eve\ Lee, 0)$ from the information system of Example 2. The related sets of o_1 and o_2 are $rlt(o_1) = \{purchase(1, 1, 1, 21.09.2010), purchase(1, 3, 2, 21.09.2010), product(1, bread, 2.00), product(3, milk, 2.50)\}$ and $rlt(o_2) = \{purchase(5, 5, 2, 27.09.2010), product(5, coffee, 6.00)\}$, respectively. The objects of relation *purchase* (*product*) are directly (indirectly) related to the target objects by attribute *c_id* (by relation *purchase* and attribute *p_id*).

For a given target object one can usually obtain more than one description, each of which describes the object with different precision. Our objective is to choose an appropriate description of the target object with respect to a given data mining task. The precision of the target object's description (i.e., the related set) can be tuned by its depth level. To define a related set of a given depth level, we generalize Definition 5.

Definition 7 (n-related objects) Object o_0 is n-related to object o_n if and only if there exist $n - 1$ objects such that o_i is related to o_{i+1}, where $n > 0$ and $0 \le i \le n - 1$.

[4]The tables the objects belong to are not assumed to be different.

One can note that for $n = 1$, Definitions 5 and 7 are equivalent.

A related set of a given depth level is defined as follows.

Definition 8 (n-related set) The n-th depth level related set of a target object o, denoted by $rlt^n(o)$, is a set of background objects, each of which are m-related to object o and $m \leq n$.

We assume that $rlt^n(o) = \emptyset$ for $n = 0$. It is reasonable to study a target object without its related set (i.e., the related set is empty) when the object itself includes information, i.e., descriptive attributes occur in the target relation (e.g., attribute *class* of relation *customer*).

Example 4 Consider the target object $o = customer(1, Tom\ Jackson, 1)$ from the information system of Example 2.

We can receive two different non-empty descriptions of o, namely $rlt^1(o) = \{purchase(1, 1, 1, 21.09.2010),\ purchase(1, 3, 2, 21.09.2010)\}$ and $rlt^2(o) = \{purchase(1, 1, 1, 21.09.2010),\ purchase(1, 3, 2, 21.09.2010),\ product(1, bread, 2.00),\ product(3, milk, 2.50)\}$.

As presented above, each target object is represented by the set of background objects related to the target object. It is natural to treat such a set as *a granule of objects drawn together by their relationships with the target object*. Therefore, we examine a granule defined by the pair $(o, rlt(o))$, where o is a target object from a given information system. Granules of this form are treated as 1-type information granules.

A related set of a given target object can be viewed as its specific description. In order to derive relational patterns (e.g., classification rules), the target object's description is generalized. To obtain a general description of a target object itself and its related set, they are both generalized.

Definition 9 (Generalized target object) A generalized target object o, denoted by o_{gen}, is the target object with certain components replaced according to a given substitution.[5]

Definition 10 (Generalized related set) A generalized related set of a target object o, denoted by $rlt_{\text{gen}}(o)$, is the related set with certain components replaced according to the substitution (partially) constructed during the generalization of the target object.

A generalized n-related set is defined in an analogous way.

Related sets can be generalized in a variety of ways (for more details, see Hońko (2010)). A method for generalization can be developed taking into consideration language bias.

Example 5 Consider again the target object $o = customer(1, Tom\ Jackson, 1)$ from the information system of Example 2 and its related set $rlt^2(o) = \{purchase(1,$

[5] A component of an object can be replaced with either a variable, a list of constants, or symbol "_" if the component is not important for the consideration.

1, 1, 21.09.2010), *purchase*(1, 3, 2, 21.09.2010), *product*(1, *bread*, 2.00), *product*(3, *milk*, 2.50)}. The generalized target object and its related set can be of the following forms $o_{\text{gen}} = customer(A, \_, 1)$ and $rlt^2_{\text{gen}}(o) = \{purchase(A, B, \_, \_), product(B, [bread, milk], \_)\}$,[6] respectively.

An object of the relation *customer* can be generalized according to the following language bias constraint $mode(customer(+type(c\_id), \_, \#[0, 1]))$,[7] which means that the first argument of the relation *customer* has to be replaced with an input variable of a type that is the same as that of attribute *c_id*, the second one has to be omitted, and the third one can be replaced with 0 or 1 (i.e., the class label). In turn, the substitution according to which object o is generalized is of the form $\{1_{c\_id}/A, Tom\ Jackson/\_\}$.

Based on the generalized target objects and their related sets, we define information granules by their syntax and semantics. For this purpose we extend to a relational case the method for constructing information granules (Skowron and Stepaniuk 2001).

In our approach an elementary granule is defined by a conjunction of relational descriptors, i.e., expressions of the form $R(t_1, t_2, \dots, t_n)$, where R is a relation name and t_i $(1 \leq i \leq n)$ are the terms.[8]

We are given information system $IS_D = (U_D, A_D)$.

- A generalized target object o_{gen} of object o from IS_D is a trivial elementary granule, i.e., a single relational descriptor.
  The meaning (i.e., semantics) of the granule, denoted by $SEM_{IS_D}(o_{\text{gen}})$, is the set of target objects that satisfy the descriptor.
- A generalized related set $rlt_{\text{gen}}(o)$ of target object o from IS_D is an elementary granule where each descriptor is constructed based on a background relation.
  The meaning of the granule, denoted by $SEM_{IS_D}(rlt_{\text{gen}}(o))$, is the set of target objects for each of which there exists a substitution such that each descriptor under the substitution is satisfied.
- A generalized target object o_{gen} with its generalized related set $rlt_{\text{gen}}(o)$ is represented by the granule $(o_{\text{gen}}, rlt_{\text{gen}}(o))$, i.e., the 2-type information granule.[9] The meaning of the granule is $SEM_{IS_D}\big((o_{\text{gen}}, rlt_{\text{gen}}(o))\big) = (SEM_{IS_D}(o_{\text{gen}}), SEM_{IS_D}(rlt_{\text{gen}}(o)))$.

Example 6 Consider information system IS_D of Example 2 and the generalized target object $o_{\text{gen}} = customer(A, \_, 1)$ and its related set $rlt^2_{\text{gen}}(o) = \{purchase(A, B, \_, \_), product(B, [bread, milk], \_)\}$.

[6]The denotation $[v_1, v_2, \dots, v_n]$ occurring in an object's argument list means that the corresponding attribute may take any of the values $v_1, v_2, \dots, v_n$. We assume that lists are formed for attributes that take on a relatively small number of values. Otherwise, the attributes are previously discretized.

[7]The mode declaration method used in this work is similar to that introduced in Muggleton (1995).

[8]A term is either a constant, a variable or a compound term. Here we use compound terms limited to lists of constants.

[9]If the type of information granules is not mentioned, the granules are assumed to be of the 2-type.

The meaning of granule $(o_{\text{gen}}, rlt^2_{\text{gen}}(o))$ is $SEM_{IS_D}\left((o_{\text{gen}}, rlt^2_{\text{gen}}(o))\right) = (\{o_1, o_2, o_3\}, \{o_1, o_2\})$ (o_i stands for the i-th customer of database D).

The information granules as defined above can be viewed as an abstract representation of relational data. The accuracy level of the representation can easily be changed by taking another depth level of the related sets. Moreover, a representation constructed based on the information granules of all of the target objects is treated in our approach as the search space for discovering patterns. Thanks to this the size of the search space may be significantly limited.

Granularity of the universe is defined by the set $\{SEM_{IS_D}(rlt^n_{\text{gen}}(o)) : o \in U_{D_T}\}$. Thus different depth levels of related sets correspond to different levels of information granulation. As the depth level increases, a lower-level granularity is obtained.

4.3 Relational knowledge

The information granules defined in the previous section are the basis for the discovery of relational knowledge. Thanks to constructing such granules we are able to obtain knowledge of a different type. Therefore, we can consider as granules, e.g., frequent patterns and relational association rules, relational classification rules, and relational clusters and their descriptions.

We will show how relational classification rules can be represented by granules. Firstly, we restate the definition of a relational classification rule.

Definition 11 (Relational classification rule) A relational classification rule is an expression of the form[10]

$$R(t_1, t_2, \ldots, t_n) \leftarrow R_1\left(t^1_1, t^1_2, \ldots, t^1_{n_1}\right) \wedge R_2\left(t^2_1, t^2_2, \ldots, t^2_{n_2}\right) \wedge \cdots \wedge R_m\left(t^m_1, t^m_2, \ldots, t^m_{n_m}\right),$$

where R is a target relation, R_i $(1 \leq i \leq m)$ are background relations, and indexed t are the terms.

For simplicity's sake we denote a relational rule as $\alpha \leftarrow \beta$.
The accuracy (coverage) of the rule $\alpha \leftarrow \beta$ is the ratio between the number of objects that satisfy $\alpha \wedge \beta$ and the number of objects that satisfy β (α).

Example 7 We are given database D of Example 1 and the classification rule $\alpha \leftarrow \beta$, where $\alpha = customer(A, \_, 1)$[11] and $\beta = purchase(A, B, \_, \_) \wedge product(B, [bread, milk], \_)$. Rule premise β is satisfied by objects o_1, o_2, each of which is from class 1. Rule conclusion α is satisfied by objects o_1, o_2, o_3. Hence, the rule's accuracy and coverage are 1 and 2/3, respectively.

[10]One can also take into account rules including negated descriptors or conditions formed based on arguments of descriptors previously added.

[11]In our approach we assume that the last term of the rule conclusion is a constant that defines the class the rule describes.

Now we define granules through the relation classification rules.
We are given information system $IS_D = (U_D, A_D)$.

- A relational classification rule $\alpha \leftarrow \beta$ in IS_D is represented by granule (α, β), where α and β correspond to o_{gen} and $rlt_{\text{gen}}(o)$, respectively.
  The meaning of the granule is $SIM_{IS_D}((\alpha, \beta)) = (SIM_{IS_D}(\alpha), SIM_{IS_D}(\beta))$. The rule's accuracy and coverage can be calculated by $acc_{IS_D}(\alpha \leftarrow \beta) = \frac{|SEM_{IS_D}(o_{\text{gen}}) \cap SEM_{IS_D}(rlt_{\text{gen}}(o))|}{|SEM_{IS_D}(rlt_{\text{gen}}(o))|}$ and $cov_{IS_D}(\alpha \leftarrow \beta) = \frac{|SEM_{IS_D}(o_{\text{gen}}) \cap SEM_{IS_D}(rlt_{\text{gen}}(o))|}{|SEM_{IS_D}(o_{\text{gen}})|}$, respectively.
- A set of relational classification rules is represented by the set of granules $\{(\alpha_i, \beta_i) : 1 \le i \le k\}$, where k is the cardinality of the set of rules. The meaning of the granule is $\{SEM_{IS_D}((\alpha_i, \beta_i)) : 1 \le i \le k\}$.

Example 8 We are given information system IS_D of Example 2 and the classification rule of Example 7, i.e., $\alpha \leftarrow \beta$, where $\alpha = customer(A, \_, 1)$ and $\beta = purchase(A, B, \_, \_) \wedge product(B, [bread, milk], \_)$.

Consider the generalizations of object $o = customer(1, Tom\ Jackson, 1)$: $o_{\text{gen}} = customer(A, \_, 1)$, $rlt^2_{\text{gen}}(o) = \{purchase(A, B, \_, \_), product(B, [bread, milk], \_)\}$. The rule $\alpha \leftarrow \beta$ can be represented by the granule $(o_{\text{gen}}, rlt^2_{\text{gen}}(o))$ with the meaning $SEM_{IS_D}(\alpha \leftarrow \beta) = (\{o_1, o_2, o_3\}, \{o_1, o_2\})$. The rule's accuracy and coverage are $acc_{IS_D}(\alpha \leftarrow \beta) = \frac{|SEM_{IS_D}(o_{\text{gen}}) \cap SEM_{IS_D}(rlt^2_{\text{gen}}(o))|}{|SEM_{IS_D}(rlt^2_{\text{gen}}(o))|} = 1$ and $cov_{IS_D}(\alpha \leftarrow \beta) = \frac{|SEM_{IS_D}(o_{\text{gen}}) \cap SEM_{IS_D}(rlt^2_{\text{gen}}(o))|}{|SEM_{IS_D}(o_{\text{gen}})|} = 2/3$.

5 Granular computing framework for generating relational classification rules

We propose the following framework for generating relational classification rules. In this approach the rules can be generated by applying a top-down or bottom-up method.

Given:

- $IS_D = (U_D, A_D)$ – the information system of database D;
- n – the depth level of related sets;

Find:

- RS – a set of classification rules;

Steps:

1. $RS := \emptyset$;
2. For each target object o of IS_D compute $rlt^n(o)$;
3. Choose one object from the target objects;
4. Based on the chosen object, generate an initial rule, i.e., a granule of the form $r = (o_{\text{gen}}, rlt^m_{\text{gen}}(o))$, where
   4.1 *The top-down case*: $m := 0$;
   4.2 *The bottom-up case*: $m := n$;

```
5. Refine the initial rule $r$:
    5.1 Compute a set of candidate rules:
        5.1.1 The top-down case: $RS' := \{r\} \cup special(r, m)$; Next $m := m + 1$;
        5.1.2 The bottom-up case: $RS' := \{r\} \cup general(r, m)$; Next $m := m - 1$;
    5.2 $r := best\_candidate(RS')$;
    5.3 Repeat steps 5.1 and 5.2 until $stop\_criterion(r)$;
6. $RS := RS \cup \{r\}$;
7. Repeat steps 3–6 until $stop\_criterion(RS)$;
```

We will study selected steps of the above proposal.

Re 3. The way of choosing target objects is defined by the algorithm to be used. One can observe that in the top-down case the choice of a target object is not important because the generalization of each target object of a given class is the same.

Re 4. The way target objects and their related sets are generalized in the top-down case may vary from that in the bottom-up case.

Re 4.1. and 4.2.

The top-down case:	An initial rule is to be the most general one, hence $m = 0$.
The bottom-up case:	An initial rule is to be the most specific one, hence $m = n$.

Re 5.1. The function $special(r, m)$ ($general(r, m)$) returns a set of allowed specializations (generalizations) of a rule r at a level m.[12]

Re 5.1.1. A specialization of a rule r is done in one of the following ways:

1. A variable that occurs in r is replaced with a list of values the variable may take, e.g., a specialization of the rule $customer(A, \_, 1) \leftarrow purchase(A, B, C, \_)$ is $customer(A, \_, 1) \leftarrow purchase(A, B, [1, 2], \_)$;[13]
2. A list of values that occurs in r as a component is replaced with its non-empty sublist, e.g., a specialization of the rule $customer(A, \_, 1) \leftarrow purchase(A, B, [1, 2], \_)$ is $customer(A, \_, 1) \leftarrow purchase(A, B, 2, \_)$;
3. r is extended by an additional condition, e.g., a specialization of the rule $customer(A, \_, 1) \leftarrow purchase(A, C, [1, 2], \_)$ is $customer(A, \_, 1) \leftarrow purchase(A, B, [1, 2], \_) \wedge product(B, milk, \_)$.

Re 5.1.2. A generalization of a rule r is done in one of the following ways:

1. A list of values that occurs in r as a component is replaced with a new variable, e.g., a generalization of the rule $customer(A, \_, 1) \leftarrow purchase(A, B, [1, 2], \_)$ is $customer(A, \_, 1) \leftarrow purchase(A, B, C, \_)$;

[12] Allowed specializations or generalizations of a given rule are understood as those rules that can be formed according to given constraints.

[13] If a list of values consists of one element, then the list is replaced with that element.

2. A list of values that occurs in r as a component is replaced with its superlist, e.g., a generalization of the rule $customer(A, \_, 1) \leftarrow purchase(A, B, 2, \_)$ is $customer(A, \_, 1) \leftarrow purchase(A, B, [1, 2], \_)$;
3. r is reduced by removing one of its conditions, e.g., a generalization of the rule $customer(A, \_, 1) \leftarrow purchase(A, B, [1, 2], \_) \wedge product(B, milk, \_)$ is $customer(A, \_, 1) \leftarrow purchase(A, C, [1, 2], \_)$.

Re 5.2. The function $best\_candidate(S)$ returns a rule from S that has the highest quality based on a given quality measure.

Re 5.3 The stop criterion is defined by a given technique or algorithm for generating classification rules. For step 7 it is done analogously.

We give an illustrative example of the generation of a rule.

Example 9 We are given information system IS_D from Example 2. We examine the top-down case and $n = 1$. We evaluate a rule based on its accuracy and use the following constraints during the construction of the rule:

1. $mode(customer(+type(c\_id), \_, \#[0, 1]))$,
2. $mode(1, purchase(+type(c\_id), -type(p\_id), -type(amount), \_))$, [14]
3. $mode(1, purchase(+type(c\_id), -type(p\_id), \#[1, 2, 3], \_))$,

We have $RS := \emptyset$; Suppose that $o_2 = customer(2, \text{Susan Clark}, 1)$ is a chosen object. We have $rlt^1(o_2) = \{purchase(2, 1, 1, 25.09.2010), \quad purchase(3, 1, 1, 26.09.2010)\}$. According to the above constraints, we get the following generalizations $o_{2_{\text{gen}}} = customer(A, \_, 1)$, $rlt^1_{\text{gen}}(o_2) = \{purchase(A, B, C, \_)\}$. The case $m = 0$. We have the initial rule $r = (o_{2_{\text{gen}}}, rlt^0_{\text{gen}}(o_2)) = (customer(A, \_, 1), \emptyset)$[15] with accuracy 1/2. Since the set of specifications of r is empty under the given constraints, then r is the best candidate. The case $m = 1$. Based on r we obtain the following rule of level 1: $r := (customer(A, \_, 1), purchase(A, B, C, \_))$. We have the following specialization of r at level 1 (accuracy given after the colon): $(customer(A, \_, 1), \{purchase(A, C, 1, \_)\}) : 2/3$, $(customer(A, \_, 1), \{purchase(A, C, 2, \_)\}) : 1/3$, $(customer(A, \_, 1), \{purchase(A, C, 3, \_)\}) : 0$, $(customer(A, \_, 1), \{purchase(A, C, [1, 2], \_)\}) : 1/2$, $(customer(A, \_, 1), \{purchase(A, C, [1, 3], \_)\}) : 2/3$, $(customer(A, \_, 1), \{purchase(A, C, [2, 3], \_)\}) : 1/3$. We take the first rule with the highest accuracy, i.e., $r := (customer(A, \_, 1), \{purchase(A, C, 1, \_)\})$. Since $m = n$, then $RS := RS \cup \{r\}$.

6 Analysis of the framework

In this section we analyze the framework in terms of its properties, information lost during granulation, and the set theoretical interpretation of granulation.

[14]An argument preceded by symbol "+" ("−") has to be replaced with an input (output) variable. The first argument of function *mode* (i.e., value 1) means that relation *purchase* can be used in the construction of a pattern at most once.

[15]The granule $(customer(A, \_, 1), \emptyset)$ can be transformed into the rule $customer(A, \_, 1) \leftarrow 1$, where the rule premise is satisfied by any object.

6.1 The framework's properties

We discuss the properties of our framework in the context of the essential problems of relational data mining.

1. *Relational object representation* This issue is more complicated for the relational case than when data resides in a single table. Namely, an object can be represented not only by the tuple that belongs to one relation (i.e., the target relation), but also by tuples that belong to other relations that are directly or indirectly joined to the target relation. The crucial task is thus to find a representation such that, on the one hand, it is specific enough to identify objects and, on the other hand, it is general enough to avoid too detailed information. In our framework this problem is resolved thanks to applying the granular computing idea of switching between different levels of granularity of the universe. Namely, we can replace a given object representation with another one by changing the depth level of related sets. Thanks to this the relational object representation can be adjusted to a given data mining task (e.g., classification).
2. *Search space limitation* Relational data is distributed over multiple tables, and the search space for discovering relational patterns may be very huge. In the current frameworks (i.e., ILP and RDB), this problem is overcome by applying language bias. It imposes some constraints on the patterns to be discovered, thereby the search space is limited. Our framework is independent of the way language bias is specified, thus bias from the ILP or RDB framework can be adapted. Moreover, the search space in our framework is additionally limited. Namely, this is given explicitly as a class of information granules derived from the data.
3. *Generation of relational patterns* The current frameworks unify the way to generalize objects; however, the method for deriving patterns (e.g., classification rules) from relational data is provided not by the frameworks, but by the concrete algorithms that can be defined in the frameworks. Our framework partially unifies the process of discovering rules from data. Namely, the rules are not directly derived from the data, but from the information granules. A rule can directly be obtained from an information granule (i.e., the rule is equivalent to the granule) or constructed based on the granule (i.e., the rule is a specialization or generalization of another rule that is equivalent to the granule).

6.2 Granular representation and information loss

The general idea is to construct a granular representation of the data that is convenient for rule generation. The transformation of data into information granules may cause information loss. However, we can control this during both the computation of granules and rule generation. In our framework there are two cases when some information may be lost.

1. *Depth level change* The idea of the depth level is to adjust the granular representation to a given data task (to preserve essential information and to remove the detailed information). In order to avoid or minimize information loss that may be caused by a limitation to a given depth level we follow the principle below:
   If target objects that are distinguishable in the original database are distinguished based on a given granular representation, then alternative information loss is not essential.

In order to follow this principle we check distinguishability of the objects at a given depth level. If some objects are indistinguishable, we extend their representation by applying a deeper level.

2. *Transforming 1-type information granules into 2-type information granules*. The idea of the transformation is to achieve granules that are semantically different from clustered data. Despite the fact that such a representation is richer (it can reveal additional properties hidden in the data), some information can be lost during the generalization of 1-type information granules. To overcome this problem, granules can be modified (see the *special* function) in the process of rule generation to retrieve information lost during the generalization.

Summing up, both a properly selected depth level and generalization method should guarantee no essential information loss. However, if the former fails, we examine a deeper level for the conflict target objects. If the latter fails, we decrease the level of generalization by specializing the granules of the conflict target objects.

6.3 Set theoretical interpretation of relational data granulation

The commonly used approach for information granulation is based on the assumption that all important features can be captured from data by forming granules as subsets of the original data set. A consequence of this is that the information granules are not semantically distinct from clustered entities. To overcome this inconvenience, information granulation is viewed as a semantically meaningful grouping of elements based on their indistinguishability, similarity, proximity or functionality (Bargiela and Pedrycz 2008). Therefore, the problem to be solved can be cast in the formalism of non-classical set theory, e.g., NBG set theory as proposed in Bargiela and Pedrycz (2008), where apart from sets, classes that are semantically distinct from the sets are also examined.

In terms of set theory we investigate in our framework a class of granules constructed based on generalized related sets which is semantically distinct from a set of granules formed based on non-generalized related sets. The class of granules is a richer representation of the data. The granules reveal additional features hidden in the data, namely, based on the semantics of a granule one can define its incidence (the number of objects that shares the features encoded by the granule) and its significance (decision class purity among objects that share the features encoded by the granule). Both the incidence and significance of particular granules provide properties essential for constructing classification rules. Therefore, one can say that this representation is task-oriented since it shows additional properties of the original data that are helpful for generating classification rules.

The way classification rules are derived from 1-type information granules differs from that for 2-type information granules. In the former case, we search all of the features possible under a given language bias and check through the granules that represent objects which of them satisfy the features. In the latter case, we only search and alternatively modify features encoded by the granules and check which objects share the features. Despite these differences, 2-type information granulation is also consistent with the original data. This is, in fact, an intermediate representation between the original data and the classification rules to be derived.

7 Evaluation of the framework

In this section we provide a theoretical evaluation of the framework. We show its correctness and completeness as well as analyze its time complexity.

7.1 The Framework's correctness and completeness

Firstly, we evaluate the way the information granules are constructed (see Definitions 9 and 10. For the purposes of the evaluation form a simple algorithm.

Algorithm 1: *Generate_Granular_Rep*$(IS_D = (U_D, A_D))$

```
begin
    $GR := \emptyset$;
    foreach $o' \in U_{D_T} \subset U_D$ do
        $g_o := (o_{gen}, rlt_{gen}(o)); GR := GR \cup \{g_o\}$;
    return $GR$;
```

We show that the set of granules returned by *Generate_Granular_Rep* is a proper representation of the whole search space.

Definition 12 (Correctness of an algorithm for generating a representation of the search space) An algorithm that generates a representation of the search space is correct if every relationship produced by the algorithm occurs in the search space.

Definition 13 (Valid relationship) A relationship *rs* is valid in database *D* if there exists an object *o* in *D* such that *o* satisfies *rs*.

Proposition 1 *The Generate_Granular_Rep algorithm is correct.*

Proof Since every relationship *rs* is produced by the algorithm based on a target or background object *o*, then *o* satisfies *rs*. Hence, *rs* is valid by Definition 13.

Definition 14 (Completeness of an algorithm for generating a representation of the search space)

An algorithm that generates a representation of the search space is complete if it produces all valid relationships that occur in the search space.

Proposition 2 *The Generate_Granular_Rep algorithm is complete.*

Proof (by contradiction) Suppose that there exists a valid relationship *rs* in the search space *SS* such that it is not produced by the algorithm (1). Let *o* be an object that satisfies *rs*, and let rs' be a relationship produced by the algorithm based on *o*. We have *o* satisfies *rs* and rs' (2). By (1), we have $rs \neq rs'$ (3).

1. The case: *rs* is more general than rs'.
   We obtain that *rs* is constructed based on more than one object. Let *O* be the set of all objects that satisfy *rs*. We have $o \in O$. Let *RS* be the set of relationships produced by the algorithm such that each $rs'' \in RS$ is constructed based on one

$o' \in O$. We have $rs' \in RS$. We receive $rs \equiv \bigvee_{rs'' \in RS} rs''$. Therefore, rs can be produced by the algorithm. Hence, contradiction with (1).

2. The case: rs' is more general than rs.
   We obtain that rs is constructed based on more than one object. Let O be the set of all objects that satisfy rs'. There exists a subset $O' \in O$ such that $rs \equiv \bigvee_{rs'' \in RS} rs''$ and each $rs''_i \in RS$ is constructed based on one $o \in O'$. Therefore, rs can be produced by the algorithm. Hence, a contradiction with (1).
3. Case: rs is not more general than rs' and vice versa.
   (a) The case: $rs \equiv rs'$.
   It leads to a contradiction with (3).
   (b) The case: $rs \not\equiv rs'$.
   We obtain that o satisfies neither rs nor rs'. Hence, a contradiction with (2).

□

Definition 15 (Correctness of the rule generation algorithm) An algorithm that generates a rule set is correct if every rule produced by the algorithm has a quality not less than a given threshold.

Proposition 3 *An algorithm constructed based on the framework is correct.*

Proof The correctness is guaranteed by step 5.3 of the framework. Namely, a rule is added to the rule set if the rule's quality is not less than a given threshold. □

Definition 16 (Completeness of the rule generation algorithm)
An algorithm that generates a rule set is complete if it produces all (required) rules that have a quality not less than a given threshold.

Proposition 4 *An algorithm constructed based on the framework is complete.*

Proof Suppose there exists $o \in U_{D_T}$ such that o does not satisfy any $r \in RS$, where RS is the rule set produced by the algorithm. Let r' be the rule constructed based on o. We have $r' \notin RS$ (1). Regardless of the quality measure, we can assume that the quality of r' is lower than a given threshold due to at least one of the following:

1. The number of objects $o \in U_{D_T}$ satisfying r' such that $class(o) = class(r)$[16] is lower than a given threshold.
   Let c be the condition of r' such that c decreases the quality of r'. Let c' be a condition found over the whole search space such that r' after replacing c with c' is of the desired quality. We obtain that c' is more general than c. Therefore, $c' \equiv c \vee C$, where C is a disjunction of conditions. By Proposition 2, we have that each condition of C can be constructed by the *Generate_Granular_Rep* algorithm. Hence, c' can be constructed based on results produced by *Generate_ Granular_Rep* or by the *general* function. Hence,a contradiction with (1).

[16]The *class* function returns the class label for an object and, for a rule, the label of the class the rule describes.

2. The number of objects $o \in U_{D_T}$ satisfying r' such that $class(o) \neq class(r)$ is lower than a given threshold.
   Let c and c' be defined as in the previous case. We have c is more general than c'. Therefore, $c \equiv c' \vee C$, where C is a disjunction of conditions. Hence, c' can be constructed by omitting the condition of C or by using the *special* function. It leads to a contradiction with (1). □

7.2 The framework's complexity

We provide an analysis of the framework's time complexity. We study the operations of granule formation and rule generation.
Let $n = |U_{D_T}|$ and $m = |U_{D_B}|$.

1. The cost of the formation of granules $(o, rlt(o))$ for all $o \in D_T$ is

$$T(n, m) = nm' \leq nm = O(nm),$$

   where m' is the number of all objects 1. from the database's tables to be scanned. In a pessimistic case we have $m' = m$.
2. The cost of the generalization of all granules $(o, rlt(o)) \in U$ is

$$T(n, m) = |U| \sum_{o' \in \{o\} \cup rlt(o)} \sum_{a \in attr(o')} 1 = n(|rlt(o)| + 1)C \leq n(m + 1)C = O(nm),$$

   where $C = \sum_{a \in attr(o')} 1$ is the cost of the generalization of an object o'.[17] C does not depend on the data size.

Relational data is represented by a class of granules of the form $(o_{\text{gen}}, rlt_{\text{gen}}(o))$. One can observe that the size of this representation only depends on the size of U_{D_T}. Namely, we assume that a given database is representative, i.e., (almost) all relationships occur in the database. Therefore, adding new background objects does not affect (or hardly affects) the form of the generalized related sets. Hence, we can ignore the size of $rlt_{\text{gen}}(o)$ when analyzing the complexity of the approach for rule generation.

1. Construct a rule (without checking the rule's satisfaction).
   Let o be a target object based on which a rule is constructed. To generate a rule we need to scan objects from $rlt_{\text{gen}}(o)$ to check which of them should be taken as the rule's conditions. We assume that the cost of scanning a set is equal to its cardinality.
   The cost of rule generation is

$$T(n) = \sum_{o' \in rlt_{\text{gen}}(o)} 1 = |rlt_{\text{gen}}(o)| \leq C = O(1),$$

   where $C = \max\{|rlt_{\text{gen}}(o)| : o \in U_{D_T}\}$.

[17] $attr(o)$ is the collection of all components of an object o.

2. Compute *special*(r, i) or *general*(r, i) for all conditions of a rule.
   (a) Step 5.1.2.1
   To generalize a rule we need to replace a list of values that occur in the rule's condition with a variable. We assume that the cost of the replacement of a list of values is 1.
   Let RS be the set of all rules, r the rule to be generated, $cond(r)$ the set of all conditions of a rule r, and $comp_{\mathrm{mod}}(c)$ the set of components (constants, a list of constants or variables) of a rule condition c to be modified, i.e., generalized or specialized. The cost of the generalization of a rule is

$$T(n) = \sum_{c \in cond(r)} \sum_{l \in comp_{\mathrm{mod}}(c)} 1 = |cond(r)||comp_{\mathrm{mod}}(c)| \leq C_1 C_2 = O(1),$$

   where $C_1 = \max\{|cond(r)| : r \in RS\}$, $C_2 = max\{|comp_{\mathrm{mod}}(c)| : c \in cond(r), r \in RS\}$. Values C_1 and C_2 are small and do not depend on the data size.
   (b) Step 5.1.1.1
   To specialize a rule we need to replace its condition variable with a list of values. We assume that the cost of the replacement of a variable is 1.
   Let $val(V)$ be the values set of a variable V , and L_V the set of lists of values taken into account during the replacement of a variable V. The cost of the specialization of a rule is

$$T(n) = \sum_{c \in cond(r)} \sum_{V \in comp_{\mathrm{mod}}(c)} \sum_{l \in L_V} 1 \leq \sum_{c \in cond(r)} \sum_{V \in comp_{\mathrm{mod}}(c)} (2^{|val(V)|} - 2)$$
$$= |cond(r)||comp_{\mathrm{mod}}(c)|(2^{|val(V)|} - 2) \leq C_1 C_2 (2^{C_3} - 2) = O(1),$$

   where $C_1 = \max\{|cond(r)| : r \in RS\}$, $C_2 = \max\{|comp_{\mathrm{mod}}(c)| : c \in cond(r), r \in RS\}$, $C_3 = \max\{|val(V)| : V \in var(c), c \in cond(r), r \in RS\}$. In the pessimistic case we have $L_V = P(val(V)) \setminus \{\emptyset, val(V)\}$.[18] C_1 and C_2 are small and do not depend on the data size and neither does C_3, since we assume that the data is discretized.
   (c) Step 5.1.1.2 and 5.1.2.2
   Analogously to the above point.
   (d) Step 5.1.1.3
   To specialize a rule we need to choose a new condition. We assume that the cost of the choice of a condition is 1.
   Let $cond_i(r)$ be the set of all conditions to be generated for a rule r at a given level i. The cost of the specialization of a rule is

$$T(n) = \sum_{c \in cond_i(r)} 1 = |cond_i(r)| \leq C = O(1),$$

   where $C = \max\{|cond_i(r)| : r \in RS\}$ is small and does not depend on the data size.

[18] $P(X)$ is the power set of X.

(e) Step 5.1.2.3
To generalize a rule we need to scan all of the rule's conditions in order to remove one of them. We assume that the cost of the removal of a condition is 1.
Let r be a rule to be specialized. The cost of the generalization of a rule is

$$T(n) = \sum_{c \in cond(r)} 1 = |cond(r)| \leq C = O(1),$$

where $C = \max\{|cond(r)| : r \in RS\}$ is small and does not depend on the data size.

3. Check if the target objects satisfy a rule.
If a rule is only constructed by adding or removing conditions (step 5.1.1.3 or 5.1.2.3), it is enough to scan $rlt_{\text{gen}}(o)$ to check if o satisfies the rule. If any condition of a rule is generalized or specialized (steps 5.1.1.1-2 or 5.1.2.1-2), we need to additionally scan $rlt(o)$ to check if o satisfies the condition. However, we assume that the background objects from $rlt(o)$ are associated with the corresponding objects from $rlt_{\text{gen}}(o)$. Thanks to this there is no need to scan the whole $rlt(o)$. Hence, we can ignore the cost of finding a background object to satisfy a given condition and we assume that the cost of the verification of a condition is 1.
Let $O \subseteq U_{D_T}$ be a set of objects for which a rule is to be checked, and $mod(c)$ the set of all conditions derived from a condition c by using the *special* or *general* function. The cost of checking the rule satisfaction is

$$\begin{aligned} T(n) &= \sum_{o \in O} \sum_{o' \in rlt_{\text{gen}}(o)} \sum_{c \in cond(r)} \sum_{c' \in \text{mod}(c)} 1 \\ &= |o \in O||rlt_{\text{gen}}(o)||\{c \in cond(r)\}||mod(c)| \leq nC_1C_2C_3 = O(n), \end{aligned}$$

where $C_1 = \max\{|rlt_{\text{gen}}(o)| : o \in U_{D_T}\}$, $C_2 = |cond(r)|$, $C_3 = \max\{|mod(c)| : c \in cond(r)\}$. When the whole rule is only constructed by adding or removing conditions (step 5.1.1.3 or 5.1.2.3), then $C_3 = 1$.

The operations from points 1–3 are independent, thus the complexity of the generation of a rule is $O(1) + O(1) + O(n) = O(n)$. We assumed that the database is representative, hence we obtain that the number of rules does not depend on the data size. Thus, the complexity of the generation of a rule set RS is

$$|RS|O(n) = O(n).$$

Based on the above analysis we can immediately show the rule generation approach's scalability with respect to the data size.

Definition 17 (Algorithm's scalability) An algorithm is scalable with respect to data size n if it has a linear time complexity.

Proposition 5 *An algorithm for rule generation based on the framework is scalable.*

Proof The algorithm's time complexity for data size n is $O(n)$, hence, and by Definition 17, the algorithm is scalable. □

8 Conclusions and future directions

We have proposed in this work a granular computing framework for generating classification rules from relational data. The framework is based on the general framework for mining relational data. The structure for storing relational data in this framework is an information system that is constructed by adapting the notion of a standard information system. Information granules derived from the information system are used to construct relational classification rules.

Conclusions that involve the approach proposed in this paper are the following.

1. The framework can be helpful when a given database consists of many tables and some background objects are joined with the target ones through a number of tables. In this case there arises the problem of how deeply one should search the database for background objects that are joined with the target ones. In our framework the search level can easily be changed so as to adjust the target object representation to a given data mining task.
2. Our framework can also be useful when the search space limitation affected by language bias is not sufficient. The search space can additionally be limited, since it is given as a set of information granules derived from the data.
3. Our framework has an advantage over the ILP and RDB frameworks in terms of generation of patterns. Namely, the framework, unlike others, partially unifies the process of discovering patterns from data. This is done by constructing the search space based on information granules. The patterns can thus directly be derived from such granules or constructed based on them.

Furthermore, in this framework one can define new algorithms as well as redefine existing ones for generating relational classification rules.

Since our research as presented in this paper is theoretical, a direction for future work is to develop an algorithm in the framework. More detailed directions that involve some important practical issues are the following.

1. *Granulation of semantically distinct attributes* In the approach proposed in this paper one can indicate two types of attributes: descriptive attributes (which are replaced with values) and key attributes (which are replaced with variables). Such an attribute distribution usually naturally results from the database structure. However, in some cases an additional type of attributes is desirable. Namely, except for descriptive and key attributes one can consider functional ones. They can be used to characterize the functional nature of the data. As the results reported in Pedrycz and Bargiela (2010) show the conceptual distinction between descriptive and functional attributes can have an essential influence on information granulation.
2. *Optimization of the levels of granularity* In our approach data is transformed into information granules. Thanks to that, we obtain a higher-level representation of data. Namely, an information granule can represent a group of target objects, revealing their important properties. The information granules can be adjusted by applying a depth level. However, it is possible that at a given level the number of possible descriptors can be relatively high. Therefore, an additional adjustment that relies on the selection of the most important descriptions would be helpful. Such an adjustment can be based on a criterion maximizing the

number of target objects from the same class that are represented by the same granule, cf. Pedrycz and Bargiela (2012).

3. *Interpretability of information granules* An important issue of granular computing is interpretability of information granules. This task is complex and requires a comprehensive analysis of all aspects of the environment on which granules are developed and used (Mencar 2009). Interpretability of granules proposed in this paper can be estimated by examining relational classification rules obtained by the transformation of the granules. Any information granule in our approach can in an easy way be transformed into a classification rule. Therefore, information granules can be evaluated by measuring interpretability of the corresponding rules from the point of view of understandability. For example, it can be done by adapting an approach proposed in Mencar et al. (2011) that evaluates how much the semantics of fuzzy rules is coherent with their logical view.

Acknowledgement The research was supported by the grant W/WI/2/2011 from the Bialystok University of Technology.

References

Banks, D., House, L., McMorris, F.R., Arabie, P., Gaul, W. (2004). *Classification, clustering, and data mining applications*. Secaucus, NJ, USA: Springer-Verlag New York, Inc.

Bargiela, A., & Pedrycz, W. (2003). *Granular Computing: An Introduction*. Boston: Kluwer Academic Publishers.

Bargiela, A., & Pedrycz, W. (2008). Toward a theory of granular computing for human-centered information processing. *IEEE Transactions on Fuzzy Systems 16*(2), 320–330.

De Raedt, L. (2008). *Logical and relational learning*. Berlin Heidelberg: Springer-Verlag.

Džeroski, S., & Lavrač, N. (2001a). An introduction to inductive logic programming. In Džeroski and Lavrač (2001b) (pp. 48–71). Berlin: Springer.

Džeroski, S., & Lavrač, N. (Eds.) (2001b). *Relational data mining*. Berlin: Springer.

Han, J., Kamber, M., Pei, J. (2011). *Data mining: Concepts and techniques* (3rd edn.). San Francisco, CA, USA: Morgan Kaufmann Publishers Inc.

Hońko, P. (2010). Similarity-based classification in relational databases. *Fundamenta Informaticae, 101*(3), 187–213.

Hońko, P. (2013). Association discovery from relational data via granular computing. *Information Sciences*. Elsevier. doi:http://dx.doi.org/10.1016/j.ins.2013.01.004.

Knobbe, A.J. (2006). *Multi-relational data mining. Frontiers in artificial intelligence and applications* (Vol. 145). Amsterdam, Netherlands: IOS Press.

Knobbe, A.J., Siebes, A., Blockeel, H., Van Der Wallen, D. (2000). Multi-relational data mining, using UML for ILP. In *Principles of data mining and knowledge discovery* (pp. 1–12).

Lin, T.Y. (2005). Introduction to special issues on data mining and granular computing. *International Journal of Approximate Reasoning 40*(1–2), 1–2.

Lin, T.Y., & Zadeh, L.A. (2004). Special issue on granular computing and data mining. *International Journal of Intelligent Systems 19*(7), 565–566.

Mencar, C. (2009). Interpretability of fuzzy information granules. In Bargiela, A., & Pedrycz, W. (Eds.), *Human-centric information processing through granular modelling, studies in computational intelligence* (Vol. 182, pp. 95–118). Springer-Verlag.

Mencar, C., Castiello, C., Cannone, R., Fanelli, A.M. (2011). Interpretability assessment of fuzzy knowledge bases: a cointension based approach. *International Journal of Approximate Reasoning 52*(4), 501–518.

Muggleton, S. (1995). Inverse entailment and Progol. *New Generation Computing 13*(3–4), 245–286.

Nédellec, C., Rouveirol, C., Adé, H., Bergadano, F., Tausend, B. (1996). Declarative bias in ILP. In De Raedt, L. (Ed.), *Advances in inductive logic programming* (pp. 82–103). Amsterdam, Netherlands: IOS Press.

Pawlak, Z. (1991). *Rough sets. Theoretical aspects of reasoning about data*. Dordrecht: Kluwer Academic.

Pedrycz, W., & Bargiela, A. (2010). Fuzzy clustering with semantically distinct families of variables: descriptive and predictive aspects. *Pattern Recognition Letters 31*(13), 1952–1958.

Pedrycz, W., & Bargiela, A. (2012). An optimization of allocation of information granularity in the interpretation of data structures: toward granular fuzzy clustering. *IEEE Transactions on Systems, Man, and Cybernetics, Part B 42*(3), 582–590.

Pedrycz, W., & Chen, S.M. (Eds.) (2011). Granular computing and intelligent systems: Design with information granules of higher order and higher type. *Intelligent systems reference library* (Vol. 13). Berlin Heidelberg: Springer-Verlag.

Pedrycz, W., Skowron, A., Kreinovich, V. (Eds.) (2008). *Handbook of granular computing*. New York: Wiley.

Skowron, A., & Stepaniuk, J. (2001). Information granules: towards foundations of granular computing. *International Journal of Intelligent Systems 16*(1), 57–85.

Skowron, A., Stepaniuk, J., Swiniarski, R. (2012). Modeling rough granular computing based on approximation spaces. *Information Sciences 184*(1), 20–43.

Stepaniuk, J. (2008). Rough-granular computing in knowledge discovery and data mining. *Studies in computational intelligence* (Vol. 152). Berlin-Heidelberg: Springer-Verlag.

Tan, P.N., Steinbach, M., Kumar, V. (2005). *Introduction to data mining* (1st edn.). Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc.

Thangaraj, D.M., & Vijayalakshmi, C. (2011). A study on classification approaches across multiple database relations. *International Journal of Computer Applications 12*(12), 1–6.

Yao, J.T. (2005). Information granulation and granular relationships. In Hu, X., Liu, Q., Skowron, A., Lin, T.Y., Yager, R.R., Zhang, B. (Eds.), *Proc. the IEEE Conference on Granular Computing* (pp. 326–329). IEEE.

Yao, Y.Y. (2000). Granular computing: Basic issues and possible solutions. In *Proc. the 5th joint conference on information sciences* (pp. 186–189).

Zadeh, L.A. (1997). Towards a theory of fuzzy information granulation and its centrality in human reasoning and fuzzy logic. *Fuzzy Sets and Systems 90*(2), 111–127.

Zadeh, L.A. (2010). A summary and update of "fuzzy logic". In Hu, X., Lin, T.Y., Raghavan, V.V., Grzymala-Busse, J.W., Liu, Q., Broder, A.Z. (Eds.), *Proc. the IEEE international conference on granular computing, IEEE Computer Society* (pp. 42–44).

Zadeh, L.A. (2011). A note on z-numbers. *Information Sciences 181*(14), 2923–2932.

Zhen, P., Wu, L., Wang, X. (2009). Research on multi-relational classification approaches. In *Proceedings of the 2009 international conference on computational intelligence and natural computing* (Vol. 1, pp. 51–54). Washington, DC, USA, CINC '09: IEEE Computer Society.