Generalized Indiscernibility Relations: Applications for Missing Values and Analysis of Structural Objects

Wojciech Jaworski

Faculty of Mathematics, Computer Science and Mechanics
Warsaw University, Banacha 2, 02-07 Warsaw, Poland
wjaworski@mimuw.edu.pl

Abstract. In this paper, we discuss an approach to structural objects based on a generalisation of indiscernibility relation used in rough set theory. The existing results in rough set theory are based on the assumption that objects are perceived by attribute value vectors.

We propose the new point of view on rough set theory. We replace information systems with the knowledge representation models that incorporate information relative to the structure of objects. We redefine the indiscernibility relation as a relation on objects characterised by some axioms. Such a definition can be naturally applied for information systems with missing values and multivalued attributes. We extend the approach on structural objects. We introduce the meaning representation language for expressing properties of structural objects and we show how to select relevant formulae from this language for sequential data.

Keywords: Indiscernibility relation, rough sets, knowledge representation, sequential data, missing values, multivalued attributes, structural objects, model of reality, data analysis, information system.

1 Introduction

Rough set theory [28,29] is based on the analysis of information systems consisting of sets of objects characterised by attribute value vectors. This data representation found successful application in many fields [30,31,32]. However, such information systems have some drawbacks when we deal with multivalued attributes or attributes with missing values. Moreover, the representation by means of vector of feature values should be treated as a consequence of a more primitive representation of structural objects. This representation is next used for computing the values of relevant attributes.

For the above reasons, we postulate a new approach to knowledge representation. We model the reality and our domain knowledge using an approach based on logic, in particular on model theory [22]. We describe our knowledge by a set of axioms written in a formal language. Such an approach solves semantic ambiguities which are present, for example, in information systems with missing values: the missing value of an attribute may be interpreted as originally

J.F. Peters and A. Skowron (Eds.): Transactions on Rough Sets VIII, LNCS 5084, pp. 116–145, 2008.

specified for a given object, yet unknown under the present knowledge or the attribute may be not applicable for a certain case or the attribute may have any value from its domain for this object. In order to solve these ambiguities, an additional domain knowledge, not represented in a given information system, is needed.

We define rough set concepts such as indiscernibility, definability and set approximations in terms of axioms. We prove that, in thecase of complete information systems, the proposed approach is equivalent to the approach used so far in rough set theory [28,29]. Then we compare our idea with extensions of rough set theory for information systems with missing values and multivalued attributes presented in [8,9,14,15,16,17,18,19,20,4,27].

Finally, we extend the rough set approach on structural objects. We introduce the meaning representation language for expressing properties of structural objects represented by data. Our language provides description of features of data irreducible relative to the attribute value vectors. The idea of knowledge representation by means of formal language was thoroughly studied in Artificial Intelligence [36].

Structural objects are often represented by means of the sequential data generated as the result of purposeful actions, for example, textual documents, voice signals or recorded parameters of cars on road. Unfortunately, the sequential representation of structural objects is computationally opaque — the structure is hidden in the sequence and the fair deal of knowledge is needed to extract it [3,34].

We show how to extract object descriptions from the sequential data using a relevant rule system (grammar of regular language) equipped with semantic attachments. Our approach is motivated by Structural Pattern Recognition [5,24,26], Natural Language Processing [12] and Information Extraction [21] as well as by Knuth's Attributed Grammars [13]. We already studied this feature extraction task in [11].

Structural objects are composed out of subobjects connected by relational constraints. These relations together with properties of objects create an ontology [2,7]. The ontology defining the structure of objects, is acquired from an expert. It helps to select the object properties relevant for the further applications. The ontology is representing hints showing how to translate data into the meaning representation language formula and how to identify objects.

The novelty of our approach is that it makes possible the generalisation of the definition of indiscernibility, making it independent from the existence of missing values, multivalued attributes and other characteristics of data. This definition is driven by semantics and it depends on the meaning of features. This allows us to merge indiscernibility with domain knowledge, introduce structure on the attribute values, express the fact that one value may be a specific case of another and deal with the problem that one object may have many different descriptions.

The indiscernibility classes are defined on the basis of the structure of objects. They provide us patterns that approximate high level concepts relevant

for classification and knowledge discovery. These patterns are expressed by logical formulae.

Missing values and multivalued attributes are examples of ambiguity in data. These are simple cases illustrating ambiguity arising during the sequential data processing. Due to the ambiguity we obtain various contradicting interpretations of data. Each of them represents a possible model of reality. We may not determine which of these interpretations is correct without acquiring further information such as a domain knowledge or more data.

This paper is structured as follows. In Section 2, we define our knowledge representation model. In Section 3, we present the application of our knowledge representation model to the information systems. In Section 4, we show how to represent structural objects as the meaning representation language formulae. In Section 5, we describe the transformation of sequential data into the meaning representation language formulae. In Section 6, we define the set approximations. In Section 7, we conclude our paper presenting applications and extensions of our ideas.

2 Knowledge Representation

The goal of data analysis is to discover information about reality from a given data. However, the data itself can also be treated as a kind of information. In this section we present the knowledge representation model and we consider the problem of a relationship between this representation and the reality.

The latter question is known in the philosophy of language as the reference problem. Some ideas presented below corresponds to picture theory of language developed by Wittgenstein in [37].

Overview of various theories of reference in the computer science context is presented in [10] and an example of the basic data model for rough set theory is reported in [6].

2.1 Reality Perceived by Sensors

We do not possess the direct insight into the nature of reality. We perceive it by means of sensors. Sensors generate structural data on the basis of reality, by extracting objects and relations among them. Sensors recognise properties of objects as well as relations between them. As examples of sensors, one can consider a thermometer, a camera as well as an expert. Generally, every analysis of the reality that results in structural data may be understood as obtained by means of sensor measurements.

We represent sensor measurements by means of relational structures developed using model theory [23]. In model theory structures are composed out of a set of individuals, and some relations among them. The set of all individuals is called as the universe of the structure. We also refer to the individuals as objects or entities.

Individuals and relations from a relational structure are represented by symbols. The set of such symbols is called a signature. The interpretation is a function that maps symbols from a given signature to individuals and relations. The

interpretation assigns objects or relations to names. It assures that the object or relation represent the aspect of sensor activity stated by its name. For example, the symbol `red` is interpreted as an object that represents the red colour. The signature is a lexicon for every syntactic construct (i.e. language formula, information system, image) by means of which we describe sensor measurements.

Let the structure $\mathcal{P}$ be a model of the reality observed by our sensors [1]. We assume that $\mathcal{P}$ is a relational structure of signature $\Sigma_{\mathcal{P}}$ and with the universe P. $\Sigma_{\mathcal{P}}$ consists of symbols of constants $u_1, u_2, \ldots$ and relational symbols $a_1, a_2, \ldots$. In symbols, $\mathcal{P}$ can be described by:

$$\mathcal{P} = \langle P, u_1^{\mathcal{P}}, u_2^{\mathcal{P}}, \ldots, a_1^{\mathcal{P}}, a_2^{\mathcal{P}}, \ldots \rangle,$$

where P includes objects as well as attribute values for objects $u_1^{\mathcal{P}}, u_2^{\mathcal{P}}, \ldots$, and $a_1^{\mathcal{P}}, a_2^{\mathcal{P}}, \ldots$ are interpretations of the symbols from $\Sigma_{\mathcal{P}}$. Constants are mapped to individuals and relational symbols to relations.

Let us consider the following example: Our goal is to describe persons. Our sensors recognise person properties such as `name`, `age`, `colour of hair`, `number of hairs`, `known languages`. A person may be a `parent` for another person and the `colour of hair` of some person may be `similar` to the `colour of hair` of another person.

In our example of a world, we are given 10 persons: $p_1, p_2, \ldots, p_{10}$. We know the `names`: `Alice`, `Bob`, `Charlie`, `David`. The `age` may be `young` or `old`. The possible `colours of hair` are `blond`, `brown`, `black` or non specified directly h_1, h_2, h_3. The `number of hairs` varies from `none`, `little`, `many`. The `known languages` are `Pascal`, `Ocaml` and `Cobol`.

We denote our example world as $\mathcal{P}_1$. The signature $\Sigma_{\mathcal{P}_1}$ consists of

- constants $p_1, p_2, \ldots, p_{10}$, `Alice`, `Bob`, `Charlie`, `David`, `young`, `old`, `blond`, `brown`, `black`, h_1, h_2, h_3, `none`, `little`, `many`, `Pascal`, `Ocaml`, `Cobol`,
- unary relations `is a person`, `is a hair colour` that define categories of objects
- binary relations `name`, `age`, `colour of hair`, `number of hairs`, `known languages`, `parent` that define person attributes .
- binary relation `similar` that define the similarity between `colours of hair`.

The structure $\mathcal{P}_1$ is defined as follows:

$$\begin{gathered}\mathcal{P}_1 = \langle P_1, p_1^{\mathcal{P}_1}, p_2^{\mathcal{P}_1}, \ldots, p_{10}^{\mathcal{P}_1}, \texttt{Alice}^{\mathcal{P}_1}, \texttt{Bob}^{\mathcal{P}_1}, \texttt{Charlie}^{\mathcal{P}_1}, \texttt{David}^{\mathcal{P}_1}, \\ \texttt{young}^{\mathcal{P}_1}, \texttt{old}^{\mathcal{P}_1}, \texttt{blond}^{\mathcal{P}_1}, \texttt{brown}^{\mathcal{P}_1}, \texttt{black}^{\mathcal{P}_1}, h_1^{\mathcal{P}_1}, h_2^{\mathcal{P}_1}, h_3^{\mathcal{P}_1} \\ \texttt{none}^{\mathcal{P}_1}, \texttt{little}^{\mathcal{P}_1}, \texttt{many}^{\mathcal{P}_1}, \texttt{Pascal}^{\mathcal{P}_1}, \texttt{Ocaml}^{\mathcal{P}_1}, \texttt{Cobol}^{\mathcal{P}_1}, \\ \texttt{is a person}^{\mathcal{P}_1}, \texttt{is a hair colour}^{\mathcal{P}_1}, \\ \texttt{name}^{\mathcal{P}_1}, \texttt{age}^{\mathcal{P}_1}, \texttt{colour of hair}^{\mathcal{P}_1}, \texttt{number of hairs}^{\mathcal{P}_1}, \\ \texttt{known languages}^{\mathcal{P}_1}, \texttt{parent}^{\mathcal{P}_1}, \texttt{similar}^{\mathcal{P}_1} \rangle.\end{gathered}$$

[1] We mean the *model* in a sense of mathematical modelling, and the *structure* in a sense of model theory.

There is no clear distinction between individuals and values of individual attributes. Therefore, we consider interpretation of every constant as an object. For example, property values like names of colours are objects. In our example, the hair colours are values of the attribute `colour of hair` and individuals on whose the relation `similar` is defined.

Objects may be abstract entities. The constant `Alice` refers to the abstract entity which is a *name*, while each p_i points to a certain person.

Let us consider the digital camera as an another example. We may define photos, data produced by this sensor, as a structure whose universe is composed out of pixels, colours and objects that represent the picture itself. We introduce binary relations `horizontal neighbours` and `vertical neighbours` that defines the topology of pixels and the 3-argument relation `colour of pixel` whose arguments are a picture, pixel and colour. The latter relation provide us an access to the picture contents.

Assume that we would like to recognise people on photos. We add new sensor which provides data structuralised by concepts such as `man`, `face`, `eye`, `hand`. The problem of people recognition is equivalent to the problem of describing one sensor concept in terms of some other concepts.

We consider objects in the universe as atomic entities. We describe their structure by means of relations yet we do not asume that objects are explicitely represented by vectors of feature values. There are two reasons for such a decision. First, in order to define structure of a given object one must possess its unambiguous decomposition into essential components. And the problem is whether such essential components exist. The second reason is that that such a structure is accessible only on the metalanguage level and therefore is useless. Instead of this, we represent the internal structure of objects by means of relations between them, for examle the *meronymy relation*.

Objects perceived by sensors need not necessarily be the real things. Objects may be epiphenomena, created by sensors (for example by human perception), nevertheless we need them in order to operate on reality. While modelling the reality we do not recognise existing objects. We define them.

2.2 Semantics of Knowledge

Knowledge, or information, provides us an insight into $\mathcal{P}$, the model of the reality observed by sensors. Information about $\mathcal{P}$ is represented by means of symbols connected by syntactic rules. Languages, information systems, images, tables, time series etc. are examples of such representations.

Besides the symbolic representation we do not possess any insight into $\mathcal{P}$ and this representation describes the results of sensor measurements in an incomplete and ambiguous way. It includes information only for a part of interesting us sensors and objects. For a given information there exists lots of different models consistent with this information.

We introduce semantics for all symbolic representations. This semantics allows us to define the symbolic representations in the formal way and translate information between such representations.

Definition 1. *We describe semantics of symbolic representation by the class of structures, which we name as the class* possible worlds *and denote as* $\mathbb{P}$. *Every* $\mathcal{Q} \in \mathbb{P}$ *is a model of possible reality, i.e., the world that does not contradict our knowledge. Each* $\mathcal{Q} \in \mathbb{P}$ *posses signature* $\Sigma_{\mathcal{Q}}$ *and interpretation* $I_{\mathcal{Q}}$.

The class of structures $\mathbb{P}$ defines knowledge independently from the syntactic medium. Every $\mathcal{Q} \in \mathbb{P}$ describes sensor measurements in a precise way ($\mathcal{Q}$ describes also the possible reality with the precision relative to the sensor precision). Imprecise description provided by the symbolic representation is interpreted as a collection of precise descriptions. Possible worlds consists of all the possible extensions of the set of given sensor measurements without any estimation or inductive reasoning over unknown measurements. Even as the multicipity of possible worlds refer to the incompleteness of knowledge, the $\mathcal{P} \in \mathbb{P}$ statement defines its correctness: the knowledge is correct if the real world belongs to the class of possible ones.

For each $\mathcal{Q} \in \mathbb{P}$ its signature $\Sigma_{\mathcal{Q}}$ is a set of atomic symbols available for symbolic representation. Signature differs across the possible worlds since some symbols may be unknown to us or we may assume existence of something that in fact does not really exist.

The interpretation is a link between sensor measurements and symbols. A part of symbols refers to the sensor construction. Interpretations of these symbols should be correlated among the possible worlds. In the digital camera example constants that represent pixels and colours as well as relations `horizontal neighbours` and `vertical neighbours` should be identical in all possible worlds.

In order to formalise the above we introduce *primitives*.

Definition 2. Primitives *are symbols such that*

- *For each primitive symbol* σ
$$\sigma \in \Sigma_{\mathcal{P}}.$$
- *For each constant primitive symbol* σ *and for each* $\mathcal{Q} \in \mathbb{P}$ *if* $\sigma \in \Sigma_{\mathcal{Q}}$ *then*
$$I_{\mathcal{P}}(\sigma) = I_{\mathcal{Q}}(\sigma).$$
- *For each* n*-ary relational primitive symbol* σ *and for each* $\mathcal{Q} \in \mathbb{P}$ *if* $\sigma \in \Sigma_{\mathcal{Q}}$ *then*
$$\forall_{u_1,\ldots,u_n \in P \cap Q}\ I_{\mathcal{P}}(\sigma)(u_1, \ldots, u_n) = I_{\mathcal{Q}}(\sigma)(u_1, \ldots, u_n).$$

In other words, each primitive has the same interpretation for all real objects in every possible world. The primitives for unreal objects may be defined in an arbitrary way. We require from the interpretations of all possible worlds to preserve the constraints defined by primitives.

Primitives connect names with specific objects. They allow us state that objects have diverse properties, belong to distinct sorts. We provide the *meaningful names* for primitives in order to represent their metalanguage definitions.

In our example the constants `Alice`, `Bob`, `Charlie`, `David`, `young`, `old`, `blond`, `brown`, `black`, `none`, `little`, `many`, `Pascal`, `Ocaml`, `Cobol` and the categories of objects `is a person`, `is a hair colour` are primitives

2.3 Language

Signature symbols denote basic concepts given by sensors. Complex concepts are defined out of basic ones by means of language. The language is a set of syntactic rules for connecting signature symbols. Each syntactic rule generate a language formula. For each rule there is provided a method of calculation the interpretation of the formula generated by the rule .

Definition 3. *Let* $\mathbb{A}$ *be a set of language formulae without free variables. We say that* $\mathbb{A}$ *is* valid *in the structure* $\mathcal{Q}$ *iff for each formula* $a \in \mathbb{A}$*, truth is an interpretation of* a *in the structure* $\mathcal{Q}$*. We denote the above as*

$$\mathcal{Q} \models \mathbb{A}.$$

We use language for the knowledge representation. In terms of language we define sensor properties.

We create constraints for symbols whose interpretation depend on the nature of the sensor. The requirement of category for the relation argument is defined by the formula that states that if a relation is true for a certain objects as its arguments, then these objects must belong to a certain category. The category itself is determined by a primitive relation. For example

$$\forall_{x,y}\ \texttt{similar}(x,y) \Longrightarrow \texttt{is a hair colour}(x) \wedge \texttt{is a hair colour}(y)$$

and

$$\forall_{x,y}\ \texttt{name}(x,y) \Longrightarrow \texttt{is a person}(x) \wedge$$
$$\wedge\ (y = \texttt{Alice} \vee y = \texttt{Bob} \vee y = \texttt{Charlie} \vee y = \texttt{David}).$$

In the same way we may state that for each object the attribute has exactly one value. For example

$$\forall_x \exists!_y\ \texttt{name}(x,y).$$

We provide the meaningful names for symbols whose interpretation is restricted by the properties of sensors

Such an approach assures us that the given symbol represents corresponding relations in all possible worlds. For example, the relation denoted by symbol `colour of hair` may vary in different possible worlds, yet it is desired for it to point in each world to the colour of hair.

The second use of language is to define values of sensor measurements.

We define the class $\mathbb{P}(\mathbb{A})$ of possible realities by a set of language formulae $\mathbb{A}$, which we call axioms:

$$\mathbb{P}(\mathbb{A}) = \{\mathcal{Q} : \mathcal{Q} \models \mathbb{A}\}.$$

Definition 4. *We say that the set of language formula* φ *is a* semantic consequence *of axioms* $\mathbb{A}$ *iff for each structure* $\mathcal{Q}$*, such that* $\mathcal{Q} \models \mathbb{A}$*,*

$$\mathcal{Q} \models \varphi.$$

We denote the above as

$$\mathbb{A} \models \varphi.$$

Table 1. Notation

Symbol	Interpretation
$\mathcal{P}$	a structure that is a real world
$\Sigma_{\mathcal{P}}$	the signature of $\mathcal{P}$
P	the universe of $\mathcal{P}$
$I_{\mathcal{P}}$	the interpretation of $\mathcal{P}$
$\mathcal{Q}$	an arbitrary structure
$\Sigma_{\mathcal{Q}}$	the signature of $\mathcal{Q}$
Q	the universe of $\mathcal{Q}$
$I_{\mathcal{Q}}$	the interpretation of $\mathcal{Q}$
$\mathbb{P}$	the class of possible worlds
$\mathcal{Q} \in \mathbb{P}$	a possible world
$\mathbb{A}$	the set of axioms
$\models$	the relation of semantic consequence
$\mathcal{Q} \models \mathbb{A}$	the axioms $\mathbb{A}$ are satisfied in the structure $\mathcal{Q}$
$\mathbb{A} \models \varphi$	the formula φ is the semantic consequence of axioms $\mathbb{A}$
U	the universe of information system
A	the set of attributes of information system
$m(u, a)$	the set of values of the attribute a for the object u in the information system.
$\underline{\mathbb{A}}X$	lower approximation of X
$\overline{\mathbb{A}}X$	upper approximation of X

The knowledge is provided in three ways: Primitives defined in metalanguage assure the connection between symbols and elements of sensor measurements. Axioms expressed as language formulae describe the properties of sensors. Language is used also to formulate axioms that define values of sensor measurements.

The Table 1 contains the summary of notation introduced in this and the following sections.

3 Information Systems

In this section, we consider data sets presented in a form of *information system* [27]. We propose axiomatic representation of information systems, define indiscernibility and set approximations for complete information systems, then we extend the definitions on the case of missing values and multivariate attributes. We compare our approach with the literature.

3.1 Complete Data

An example of information system is presented in Table 2, we denote this system by I_1. The system I_1 contains information about structure $\mathcal{P}_1$ from Section 2, yet not the whole information. Rows of the information system represent known objects, elements of $\mathcal{P}$ universe, while columns are labelled by known attributes. Attributes are relations in $\mathcal{P}$. The set of labels of objects described in the information system will be denoted by U. In Table 2, $U = \{p_1, p_2, p_3, p_4, p_5\}$. The set of all attributes included in the information system will be denoted by

Table 2. A complete information system

	name	age
p_1	Alice	young
p_2	Alice	old
p_3	Bob	young
p_4	Bob	old
p_5	Bob	young

A. In Table 2, $A = \{\texttt{name}, \texttt{age}\}$. We assume that both U and A are finite. Each attribute a defines a relation between the set of objects and the set of attribute values V_a. In Table 2, $V_{\texttt{name}} = \{\texttt{Alice}, \texttt{Bob}\}$ and $V_{\texttt{age}} = \{\texttt{young}, \texttt{old}\}$. Any information system defines attribute values for given objects. Let

$$m(u, a)$$

denote the set of values of the attribute a for the object u in the information system. Usually each attribute has exactly one value for each object, i.e $m(u, a)$ contains one element for every u and a. In such a case information system is called *complete*.

We consider an information system as a sensor. Constants that represent attribute values are primitives. We introduce the primitive relation `is an object` which recognise objects measured by sensor described in the information system. For each $u \in U$ we the statement $\texttt{is an object}(u)$ is true, yet the relation `is an object` is broader: it contains all objects that the sensor perceived, perceive and will perceive. The information system provides also the structural information about the domains of the attributes. We encode this information in the following way: for each attribute a we state

$$\mathcal{P} \models \forall_{x,y}\, a(x, y) \Longrightarrow \texttt{is an object}(x) \wedge y \in V_a.$$

The complete information system also states that every attribute has exactly one value for each object: for each attribute a we state

$$\mathcal{P} \models \forall_x \left(\texttt{is an object}(x) \Longrightarrow \exists!_y\, a(x, y)\right).$$

We encode the information system as a set of axioms $\mathbb{A}$ in the following way: For each $u \in U$, for each $a \in A$ we state

$$\mathcal{P} \models a(u, v),$$

where $v \in m(u, a)$ in the information system.

The above transformation treats both an object etiquette and an attribute value as constants. The attributes are considered as binary relations.

For Table 2, our knowledge about $\mathcal{P}_1$ provided by an information system I_1 is restricted to the following axiom:

$$\mathcal{P}_1 \models \texttt{name}(p_1, \texttt{Alice}) \wedge \texttt{age}(p_1, \texttt{young}) \wedge \texttt{name}(p_2, \texttt{Alice}) \wedge \texttt{age}(p_2, \texttt{old}) \wedge$$
$$\wedge \texttt{name}(p_3, \texttt{Bob}) \wedge \texttt{age}(p_3, \texttt{young}) \wedge \texttt{name}(p_4, \texttt{Bob}) \wedge \texttt{age}(p_4, \texttt{old}) \wedge$$
$$\wedge \texttt{name}(p_5, \texttt{Bob}) \wedge \texttt{age}(p_5, \texttt{young}).$$

Axioms derived for Table 2 allow us to define many different structures. The set of possible worlds $\mathbb{P}(\mathbb{A})$ consists of all the possible extensions of information system presented in Table 2. $\mathbb{P}(\mathbb{A})$ will include

$$\mathcal{Q}_1 = \langle Q_1, p_1^{\mathcal{Q}_1}, p_2^{\mathcal{Q}_1}, \ldots, p_5^{\mathcal{Q}_1}, \texttt{Alice}^{\mathcal{Q}_1}, \texttt{Bob}^{\mathcal{Q}_1}, \texttt{young}^{\mathcal{Q}_1}, \texttt{old}^{\mathcal{Q}_1}, \texttt{name}^{\mathcal{Q}_1}, \texttt{age}^{\mathcal{Q}_1} \rangle$$

as well as

$$\mathcal{Q}_2 = \langle Q_2, p_1^{\mathcal{Q}_2}, p_2^{\mathcal{Q}_2}, \ldots, p_{10}^{\mathcal{Q}_2}, \texttt{Alice}^{\mathcal{Q}_2}, \texttt{Bob}^{\mathcal{Q}_2}, \texttt{young}^{\mathcal{Q}_2}, \texttt{old}^{\mathcal{Q}_2}, \texttt{name}^{\mathcal{Q}_2}, \texttt{age}^{\mathcal{Q}_2} \rangle$$

or

$$\mathcal{Q}_3 = \langle Q_3, p_1^{\mathcal{Q}_3}, p_2^{\mathcal{Q}_3}, \ldots, p_5^{\mathcal{Q}_3},$$
$$\texttt{Alice}^{\mathcal{Q}_3}, \texttt{Bob}^{\mathcal{Q}_3}, \texttt{young}^{\mathcal{Q}_3}, \texttt{old}^{\mathcal{Q}_3}, \texttt{name}^{\mathcal{Q}_3}, \texttt{age}^{\mathcal{Q}_3}, \texttt{parent}^{\mathcal{Q}_3} \rangle.$$

Axioms do not provide any information about objects $p_6, \ldots, p_{10}$. They may have arbitrary properties. Relations `name`, `age` and `parent` are specified in any way that satisfy $\mathbb{A}$. They do not need to be consistent with their definition in $\mathcal{P}$.

Rough set theory [28,29] is based on the idea of an indiscernibility relation. Let B be a nonempty subset of the set A of all attributes. The indiscernibility relation $IND(B)$ is a relation on objects in a complete information system defined for $x, y \in U$ as follows

$$(x, y) \in IND(B) \text{ iff } \forall a \in B \big(m(x, a) = m(x, a)\big)$$

or equivalently

$$(x, y) \in IND(B) \text{ iff } \forall a \in B\ \forall v \in V_a\ \big(a(x, v) \Longleftrightarrow a(y, v)\big).$$

For example, for Table 2, p_3 and p_5 are indiscernible with respect to the attributes `name` and `age`.

The indiscernibility is an equivalence relation. We will denote its equivalence class generated by object u as

$$[u]_{IND(B)}.$$

Definition 5. *By a* query *over the set of attributes A we denote any formula*

$$\bigwedge_{i=1}^{n} a_i(x, v_i),$$

where $a_i \in A$, $v_i \in V_{a_i}$ and $n \leq |A|$. x is a free variable, which is valuated to an object.

Consider the query:

$$\varphi(x) = \texttt{name}(x, \texttt{Bob}) \wedge \texttt{age}(x, \texttt{young}).$$

This formula is satisfied either if p_3 is the value for x or its value is p_5. p_3 and p_5 cannot be distinguished by formula $\varphi(x)$.

We postulate the following definition of indiscernibility:

Definition 6. *Let $\mathbb{A}$ be a set of axioms. Let $\varphi(x)$ be a query with free variable x. Let u_1 and u_2 be constants. We say that u_1 and u_2 are* indiscernible by the query $\varphi(x)$ *if*

$$\big(\mathbb{A} \models \varphi(u_1)\big) \Longleftrightarrow \big(\mathbb{A} \models \varphi(u_2)\big).$$

Theorem 1. *Let $I = (U, A)$ be a complete information system with the set of objects' labels U and the set of attributes A. Let B be a subset of A. Objects $u_1 \in U$ and $u_2 \in U$ are indiscernible with respect to attribute set B iff they are indiscernible with respect to every query over the set of attributes B.*

Proof. Let $\mathbb{A}$ be axioms derived from I.

If $(u_1, u_2) \in IND(B)$, for every $a \in B$ there exists v_a such that

$$m(u_1, a) = m(u_2, a) = \{v_a\}.$$

Then the set of formulae $\{a(u_1, v_a) : a \in B\}$ is a subset of $\mathbb{A}$ and the set of formulae $\{a(u_2, v_a) : a \in B\}$ is a subset of $\mathbb{A}$. So for all $a \in B$ and $v \in V_a$ we have

$$\mathbb{A} \models a(u_1, v) \Leftrightarrow \mathbb{A} \models a(u_2, v).$$

Thus for every query $\varphi(x)$ we obtain $\mathbb{A} \models \varphi(u_1) \Leftrightarrow \mathbb{A} \models \varphi(u_2)$.

If $(u_1, u_2) \notin IND(B)$ we have $a \in B$ and v_1, v_2 such that $v_1 \neq v_2$, $m(u_1, a) = \{v_1\}$ and $m(u_2, a) = \{v_2\}$. Thus

$$\mathbb{A} \models a(u_1, v_1) \wedge \mathbb{A} \not\models a(u_2, v_1)$$

and the query $\varphi(x) = a(x, v_1)$ distinguishes u_1 and u_2.

Assume that we have two sensors. Measurements of both of them are represented by information systems and these information systems shares their sets of objects, i.e. the relation `is an object` is identical for both sensors. We wish to describe the measurements of one sensor by means of measurements of the second one. Alternatively we say the that we are describing the value of attribute in an information system (which we call the decision attribute) by the values of the rest of attributes. We may reduce this problem to the problem of description of the set objects in the information system for which the decision attribute has a certain value. Such a set is either *definable* or *indefinable* by other attributes.

Definition 7. *Let X be a subset of U. We say that X is* definable *by $\mathbb{A}$ iff there exist queries $\varphi_1(x), \ldots, \varphi_n(x)$ such that*

$$\forall u \in U \ \big(u \in X \Longleftrightarrow \mathbb{A} \models \varphi_1(u) \vee \cdots \vee \varphi_n(u)\big).$$

Each definable set is a sum of objects that satisfy at least one of a given queries.

Proposition 1. *Let $I = (U, A)$ be a complete information system with set of objects' labels U and set of attributes A. Let $\mathbb{A}$ be axioms derived from I. X is* definable *by $\mathbb{A}$ iff*

$$X = [u_1]_{IND(A)} \cup \cdots \cup [u_n]_{IND(A)}$$

for some $u_1, \ldots, u_n \in U$.

Proof. X is definable by $\mathbb{A}$ if and only if

$$X = \bigcup_{i=1}^{n} \{u \in U : \mathbb{A} \models \varphi_i(u)\}.$$

Theorem 1 states that $u_1, u_2 \in \{u \in U : \mathbb{A} \models \varphi_i(u)\}$ iff $(u_1, u_2) \in IND(A)$.

An indefinable set $X \subset U$ may be approximated by two definable sets. The first one is called the *lower approximation* of X, denoted by $\underline{\mathbb{A}}X$, and is defined by

$$\bigcup \{Y \mid Y \subset X \wedge Y \text{ is definable by } \mathbb{A}\}.$$

The second set is called the *upper approximation* of X, denoted by $\overline{\mathbb{A}}X$, and is defined by

$$\bigcap \{Y \mid X \subset Y \wedge Y \text{ is definable by } \mathbb{A}\}.$$

$\overline{\mathbb{A}}X \subset U$ because every definable set is a subset of U.

Proposition 2. *The lower and the upper approximations of any set $X \subset U$ are definable.*

Proof. For a given information system, there is a finite number of definable sets. Thus

$$\bigcup \{Y \mid Y \subset X \wedge Y \text{ is definable}\} = Y_1 \cup \cdots \cup Y_n,$$

where Y_i is defined by a formula $\varphi_i(x)$. Hence, $Y_1 \cup \cdots \cup Y_n$ is defined by $\varphi_1(x) \vee \cdots \vee \varphi_n(x)$. Similarly

$$\bigcap \{Y \mid X \subset Y \wedge Y \text{ is definable}\} = Y_1 \cap \cdots \cap Y_n,$$

where every Y_i is defined by a formula $\varphi_i(x)$. Hence, $Y_1 \cap \cdots \cap Y_n$ is defined by $\varphi_1(x) \wedge \cdots \wedge \varphi_n(x)$. The last formula may be transformed into a form of an alternative of queries.

Theorem 2. *Let $I = (U, A)$ be a complete information system with set of objects' labels U and set of attributes A, such that U and A are finite. Let $\mathbb{A}$ be the set of axioms derived from I. Then*

$$\underline{A}X = \underline{\mathbb{A}}X \text{ and } \overline{A}X = \overline{\mathbb{A}}X.$$

Proof. According to Prop. 1

$$\underline{\mathbb{A}}X = \bigcup \{[u_1]_{IND(A)} \cup \cdots \cup [u_n]_{IND(A)} \subset X\} =$$

$$= \bigcup \{[u]_{IND(A)} \mid [u]_{IND(A)} \subset X\} = \underline{A}X,$$

$$\overline{\mathbb{A}}X = \bigcap \{X \subset [u_1]_{IND(A)} \cup \cdots \cup [u_n]_{IND(A)}\}.$$

$IND(A)$ is an equivalence relation, so

$$\overline{\mathbb{A}}X = \bigcup \{[u]_{IND(A)} \mid [u]_{IND(A)} \cap X \neq \emptyset\} = \overline{\mathbb{A}}X.$$

Let the set X be the subset of universe of information system for which the decision attribute d takes the value v. Let $\underline{\varphi}(x)$ and $\overline{\varphi}(x)$ be the formulae that define the lower and upper approximation of X. $\underline{\varphi}(x)$ is equivalent to $\overline{\varphi}(x)$ when X is definable. Symbolically:

$$\{x \mid x \in U \wedge \underline{\varphi}(x)\} \subseteq \{x \mid x \in U \wedge d(x,v)\} \subseteq \{x \mid x \in U \wedge \overline{\varphi}(x)\}.$$

The claim that

$$\{x \mid \texttt{is an object}(x) \wedge \underline{\varphi}(x)\} \subseteq \{x \mid \texttt{is an object}(x) \wedge d(x,v)\} \subseteq$$
$$\subseteq \{x \mid \texttt{is an object}(x) \wedge \overline{\varphi}(x)\}.$$

we denote as *inductive reasoning*.

Inductive reasoning bases on the assumption that the definition generated for the specific data is still valid in the general case. $\underline{\varphi}(x)$ and $\overline{\varphi}(x)$ constitutes a classifier that assigns values of decision attribute to new samples.

Since a sample of objects included in the information system is not representative enough to define the bounds correctly, statistical methods are used in order to obtain bounds that are correct with the high probability. Such as method are for example: the limit of the number of queries in the bound definition, the minimal support for each query in the bound and so on.

3.2 Incomplete Data

Real-life data are frequently incomplete, i.e. values for some attributes are missing. We will assume three different interpretations of missing values:

- missing attribute values that are *lost*, i.e they are specified, yet their value are unknown
- attributes *not applicable* for a certain case, e.g. the colour of hair for a completely bald person
- *do not care values*: the attribute may have any value from its domain.

We will extend the definition of $m(u,a)$. $m(u,a) =?$ will mean that the value of attribute a for object u is lost, $m(u,a) = \star$ that it is 'do not care' and $m(u,a) = -$ that it is not applicable.

The problem of missing values was thoroughly studied (see e.g. [8,9,14,15]). The presented ideas were based on various modifications of indiscernibility relation so it could handle missing values and remain definable in terms of attributes.

The definitions of indiscernibility, definability, lower and upper approximation we stated in the above section may do not need to be modified for information systems with missing values. They are equivalent to the definitions proposed in the cited papers.

We express the various types of missing value semantics using axioms:

- for each $u \in U$, for each $a \in A$ we state

$$\mathcal{P} \models a(u,v),$$

  where $v \in m(u,a)$ in the information system.

Table 3. An information system with missing values

	number of hairs	colour of hair
p_1	none	-
p_2	little	brown
p_3	?	blond
p_4	$\star$	brown

– 'lost' values we define as follows: for each $u \in U$, for each $a \in A$ we state

$$\mathcal{P} \models a(u, v_1) \vee \cdots \vee a(u, v_n),$$

where $v_1, \ldots, v_n$ are all possible values of attribute a.

– for each $u \in U$, for each $a \in A$ whose value is not applicable we state

$$\mathcal{P} \models \forall_x \neg a(u, x),$$

– for each $u \in U$, for each $a \in A$, for each v from the domain of a we state

$$\mathcal{P} \models a(u, v),$$

when the value of a is 'do not care' for object u.

We may describe contents of Table 3 using the following formula [2]:

$$\mathcal{P}_1 \models \texttt{number of hairs}(p_1, \texttt{none}) \wedge \forall_x \neg \texttt{colour of hair}(p_1, x) \wedge$$
$$\wedge\ \texttt{number of hairs}(p_2, \texttt{little}) \wedge \texttt{colour of hair}(p_2, \texttt{brown}) \wedge$$
$$\wedge \big(\texttt{number of hairs}(p_3, \texttt{none}) \vee \texttt{number of hairs}(p_3, \texttt{little}) \vee$$
$$\vee\ \texttt{number of hairs}(p_3, \texttt{many})\big) \wedge \texttt{colour of hair}(p_3, \texttt{blond})$$
$$\wedge\ \texttt{number of hairs}(p_4, \texttt{none}) \wedge \texttt{number of hairs}(p_4, \texttt{little}) \wedge$$
$$\wedge\ \texttt{number of hairs}(p_4, \texttt{many}) \wedge \texttt{colour of hair}(p_4, \texttt{brown}).$$

Since indiscernibility with respect to the set of attributes does not work for incomplete information systems authors extended it or replaced it by another concepts.

The extension proposed in [14] for the information systems with 'do not care' missing values is the relation

$$(x, y) \in SIM(B) \text{ iff } \forall a \in B\big(m(x, a) = \star \vee m(y, a) = \star \vee m(x, a) = m(y, a)\big).$$

[2] Since it is impossible to have none, little and many hairs at the same time, the formula $\texttt{number of hairs}(p_4, \texttt{none}) \wedge \texttt{number of hairs}(p_4, \texttt{little}) \wedge \texttt{number of hairs}(p_4, \texttt{many})$ is contradictory. Yet, for the purpose of example, we do not take this fact into account.

Theorem 3. *Let $I = (U, A)$ be an information system with 'do not care' missing vales. Let U be the set of objects and A be the set of attributes. Let B be a subset of A. If objects $u_1 \in U$ and $u_2 \in U$ are indiscernible with respect to every query over the set of attributes B then*

$$(u_1, u_2) \in SIM(B).$$

The reverse implication is not valid for information systems with nontrivial missing values.

Proof. Let $\mathbb{A}$ be axioms derived from I. If $(u_1, u_2) \notin SIM(B)$ we have $a \in B$ and $v_1, v_2 \in V_a$ such that $v_1 \neq v_2$, $m(u_1, a) = \{v_1\}$ and $m(u_2, a) = \{v_2\}$. Thus

$$\mathbb{A} \models a(u_1, v_1) \text{ and } \mathbb{A} \not\models a(u_2, v_1)$$

and the query $\varphi(x) = a(x, v)$ distinguishes u_1 and u_2.

For the case of reverse implication let us consider Table 3. We have

$$(p_2, p_4) \in SIM(\{\texttt{number of hairs}\}),$$

yet the query

$$\texttt{number of hairs}(x, \texttt{none})$$

distinguish them.

In [8] an another approach for 'do not care' and 'lost' missing values is presented. The indiscernibility with respect to the set of attributes is replaced by the concept of characteristic set:

Definition 8. *For an object $u \in U$ the* characteristic set *$K_A(u)$ is defined as*

$$K_A(u) = \bigcap_{a \in A} K(u, a),$$

where $K(u, a)$ is defined in the following way

- *if $m(u, a) = \{v\}$ then*

$$K(u, a) = \{u' \in U \mid m(u', a) = \{v\} \vee m(u', a) = \star\}.$$

- *if $m(u, a) =?$ or $m(u, a) = \star$ then $K(u, a) = U$.*

Lemma 1. *Let $I = (U, A)$ be an information system with 'lost' and 'do not care' missing vales. Let U be the set of objects and A be the set of attributes. Let $\mathbb{A}$ be axioms derived from I. For every $a \in A$ and for each $u \in U$ such that $m(u, a) = \{v\}$*

$$x \in K(u, a) \Longleftrightarrow \mathbb{A} \models a(x, v).$$

Proof. Let x be an element of $K(u, a)$. Then $m(u, a) = \{v\}$ or $m(u, a) = \star$. In both cases $a(x, v)$ is satisfied by $\mathbb{A}$.

If $\mathbb{A} \models a(x, v)$, then either the value of a on x was specified as v either it was 'do not care' missing value. In both cases $x \in K(u, a)$.

Theorem 4. *Let $I = (U, A)$ be an information system with 'lost' and 'do not care' missing vales. Let U be the set of objects and A be the set of attributes. Let $\mathbb{A}$ be axioms derived from I. The set $X \subset U$ is definable iff X is the union of characteristic sets.*

Proof. Let $u_1, \ldots, u_n$ be such that

$$X = \bigcup_{i=0}^{n} K_A(u_i) = \bigcup_{i=0}^{n} \bigcap_{a \in A} K(u_i, a) = \bigcup_{i=0}^{n} \bigcap_{a \in A_i} K(u_i, a),$$

where A_i is the set of all attributes specified for u_i. Let $m(u_i, a) = \{v_{a,i}\}$. According to Lemma 1

$$K(u_i, a) = \{x \in U \mid \mathbb{A} \models a(x, v_{a,i})\}.$$

Thus $x \in X$ iff

$$\mathbb{A} \models \bigvee_{i=0}^{n} \bigwedge_{a \in A_i} a(x, v_{a,i}).$$

Theorem 5. *Let $I = (U, A)$ be an information system with 'lost' and 'do not care' missing vales. Let U be the set of objects and A be the set of attributes. Let $\mathbb{A}$ be axioms derived from I. For each $X \subset U$*

$$\underline{\mathbb{A}}X = \bigcup \{K_A(x) \mid K_A(x) \subset X\},$$

$$\overline{\mathbb{A}}X = \bigcup \{K_A(x) \mid x \in U, K_A(x) \cap X \neq \emptyset\}.$$

Lower and upper approximations are equivalent to subset *lower and upper approximations (defined in [8]).*

Proof. $\underline{\mathbb{A}}X$ is definable, so according to Thm. 4

$$\underline{\mathbb{A}}X = \bigcup_{i=1}^{n} K_A(u_i)$$

for some $u_1, \ldots, u_n \in U$. Since $\underline{\mathbb{A}}X \subset X$, we obtain $K_A(u_i) \subset X$. If $K_A(x) \subset X$ then $K_A(x) \subset \bigcup_{i=1}^{n} K_A(u_i)$, because $K_A(x)$ is definable and $\underline{\mathbb{A}}X$ is the largest definable subset of X.

$\overline{\mathbb{A}}X$ is definable, so according to Thm. 4

$$\overline{\mathbb{A}}X = \bigcup_{i=1}^{n} K_A(u_i)$$

for some $u_1, \ldots, u_n \in U$. Since $\overline{\mathbb{A}}X$ is the smallest definable set such that $X \subset \overline{\mathbb{A}}X$, we obtain $K_A(u_i) \cap X \neq \emptyset$.

3.3 Multivalued Attributes

Multiple valued attributed (introduced in [27] and studied in [20]) may reflect our incomplete knowledge about their values, what makes them similar to 'lost' missing values. The may also represent attributes that have a few values simultaneously, in which case the are like 'do not care' missing values.

- 'lost' multiple values we define as follows: for each $u \in U$, for each $a \in A$ we state
$$\mathcal{P} \models a(u, v_1) \vee \cdots \vee a(u, v_n),$$
where $v_1, \ldots, v_n$ are all possible values of attribute a for object u mentioned in information system.
- for each $u \in U$, for each $a \in A$, for each value v of attribute a for object u in information system
$$\mathcal{P} \models a(u, v),$$
when the value of a is 'do not care' multiple value for object u.

Table 4. A multiple valued information system

	name	known languages
p_5	Bob	Pascal, Ocaml, Cobol
p_6	David, Alice	Ocaml

For example objects in Table 4 will be described by the following formula:

$$\mathcal{P}_1 \models \texttt{name}(p_5, \texttt{Bob}) \wedge \texttt{known languages}(p_5, \texttt{Pascal}) \wedge$$

$$\wedge\, \texttt{known languages}(p_5, \texttt{Ocaml}) \wedge \texttt{known languages}(p_5, \texttt{Cobol}) \wedge$$

$$\wedge\, (\texttt{name}(p_6, \texttt{David}) \vee \texttt{name}(p_6, \texttt{Alice})) \wedge \texttt{known languages}(p_6, \texttt{Ocaml}).$$

Multivalued attributes is a simple extension of the 'missing values' case and the whole theory derived for the information systems with missing attributes is applicable here.

4 Structural Objects

Information systems are devoted to representation of simple objects described by a vector of attributes. What makes compound objects different from the simple ones is the internal structure. Structural objects are composed of subobjects connected by relations.

We shown in Section 3 that the idea of representing knowledge in terms of axioms provides us a flexible and extendable framework for coherent theory of data analysis. Now, we formally define the language for knowledge representation, which allow us to describe properties of structured objects. We call it a *meaning representation language*.

Table 5. A set of data sequences

s_1	`Alice has brown hair.`
s_2	`Charlie and David know Ocaml and Cobol.`
s_3	`Bob's hair colour is black, similar to David's hair colour.`
s_4	`Parents of Alice and Bob are old.`
s_5	`Alice, Bob, Charlie and David are old.`

The meaning representation language represents concepts included in data and dependencies between them. It is an extension of the formulae used to describe axioms derived from information systems in Section 3.

Syntax of the language is defined as follows: We have the set of constants and the set of predicate names.

Constants play the role of labels for entities described in data. Constant names may be meaningful (see Section 2) or may not carry any information about pointed entity. The anonymity of constants reflects the fact that we do not posses direct access to the entities. We know only the relations between entities and these relation does not define entities in an exact way. It reflects the incompleteness of our knowledge.

Predicates posses lists of one or more arguments. Number of arguments for a given predicate is not fixed. Predicates represent relations on finite sequences of entities. The predicates have meaningful names.

Atomic formula is a predicate. Formula is composed of one or more atomic formulae connected by means of conjunction or alternative. We do not use quantifiers, functions or negation.

The semantics of the meaning representation language is based on the concepts presented in Section 2. The structure $\mathcal{P}$ plays the role of the reality model. Information about $\mathcal{P}$ is represented by axioms $\mathbb{A}$. The axioms are written using the meaning representation language. $\mathbb{P}(\mathbb{A})$ is the set of possible world defined by $\mathbb{A}$. We assume that data are consistent; in other words:

$$\mathcal{P} \models \mathbb{A}.$$

For example, Table 5 provides us knowledge representation of s_1 by the following axioms:

$$\mathcal{P}_1 \models \texttt{name}(u_1, \texttt{Alice}) \wedge \texttt{colour of hair}(u_1, \texttt{brown}).$$

Note that object identifiers are not sequence identifiers s_i. One sequence may describe many objects. The same object may be mentioned in several sequences, yet we must use the domain knowledge in order to assure that different constants denote the same object. Let us consider now s_2:

$$\mathcal{P}_1 \models \texttt{name}(u_2, \texttt{Charlie}) \wedge \texttt{name}(u_3, \texttt{David}) \wedge \mathit{and}(u_4, u_2, u_3) \wedge$$

$$\wedge \texttt{known languages}(u_4, \texttt{Ocaml}) \wedge \texttt{known languages}(u_4, \texttt{Cobol}).$$

The conjunction `and` in the sequence s_2 has two meanings. The second time it is used as logical '$\wedge$', while in the first case of use it forks the sequence. We represent this operation using symbol *and* defined as

$$and(a, a_1, \ldots, a_n) \wedge \varphi(a, a_1, \ldots, a_n) \Longleftrightarrow$$

$$\Longleftrightarrow \varphi(a_1, a_1, \ldots, a_n) \wedge \cdots \wedge \varphi(a_n, a_1, \ldots, a_n).$$

In sequence s_3, the `colour of hair` is an object and the property of person in the same time:

$$\mathcal{P}_1 \models \texttt{name}(u_5, \texttt{Bob}) \wedge \texttt{colour of hair}(u_5, \texttt{black}) \wedge$$

$$\wedge \texttt{name}(u_6, \texttt{David}) \wedge \texttt{colour of hair}(u_6, u_7) \wedge \texttt{similar}(\texttt{black}, u_7).$$

The sequence s_4 is ambiguous: in first interpretation `Parents of Bob are old` and in the second `Bob is old`. We use '$\vee$' in order to represent both possibilities.

$$\mathcal{P}_1 \models \texttt{name}(u_8, \texttt{Alice}) \wedge \texttt{parent}(u_9, u_8) \wedge \texttt{parent}(u_{10}, u_8) \wedge \texttt{name}(u_{11}, \texttt{Bob}) \wedge$$

$$\wedge \big(and(u_{12}, u_9, u_{10}, u_{11}) \vee \big(\texttt{parent}(u_9, u_{11}) \wedge \texttt{parent}(u_{10}, u_{11}) \wedge$$

$$\wedge and(u_{12}, u_9, u_{10}) \big) \big) \wedge \texttt{age}(u_{12}, \texttt{old}).$$

In sequence s_5 we have the list of objects that could be arbitrary long:

$$\mathcal{P}_1 \models \texttt{name}(u_{13}, \texttt{Alice}) \wedge \texttt{name}(u_{14}, \texttt{Bob}) \wedge \texttt{name}(u_{15}, \texttt{Charlie}) \wedge$$

$$\wedge \texttt{name}(u_{16}, \texttt{David}) \wedge and(u_{17}, u_{13}, u_{14}, u_{15}, u_{16}) \wedge \texttt{age}(u_{17}, \texttt{old}).$$

5 Sequential Data Processing

Now, we show how to obtain structural object description written in form of axioms. We assume that the source information is given as the sequence of symbols, for example textual data, recorded sound, sequence of some measurements etc. We will carefully study the process of translation of sequential data into our meaning representation language in the following sections. This process is similar to the segmentation of images [35]. As we will see it is tightly connected with the syntax of the meaning representation language.

Sequential data are a description of some world $\mathcal{P}$. This description is not a precise definition, rather the theory of the set of possible worlds $\mathbb{P}(\mathbb{A})$. Our goal is to transform this description into the set of axioms $\mathbb{A}$ that would define the same theory of $\mathbb{P}(\mathbb{A})$. Thanks to this processing the data obtain the description that has a formal semantics and identifies objects and their properties.

The sequential data processing is an example of complex translation from one sensor into another one. We show in Section 6 that this process may be considered as rough set approximation.

The classical approach to sequential data analysis consists in splitting the data into subsequences of constant length denoted as windows. Then each window

is treated as a vector of attribute values. The advantage of this approach is a simple translation into an information system. The disadvantage is that it does not reflect the semantics of the data. If the windows are small, they cut object descriptions. If they are large, they do not distinguish objects. When the sequence length used for describing objects varies, the proper window size does not exist. In addition windows does not allow to express the properties of structured objects described by the sequential data.

In our approach, we divide sequences into the windows that vary in size and merge windows into larger structures. We transform data sequences into axioms using the methodology of the attributed grammars [13]. The basic idea is to perform the syntactic decomposition of the sequence using generative grammar and add the semantic value for each grammar symbol. In our case, these semantic values are formulae of meaning representation language. The semantic values are calculated by means of semantic attachments assigned to grammar rules. We extract concepts explicitly stated in sequence, not the ones that can be deduced from it.

5.1 Syntactic Rules

First, we define grammar which we will use for describing syntactic structure of data sequences.

We decided that our grammar would recognise regular languages. Yet we may replace our grammar with Context-Free Grammar without deep modification in the system.

We represent syntactic rules using a modification of context-free grammars by adding some special rule, called, a *term accumulation rule*. Formally let

$$G = (\Sigma, N, X_I, R, +, \prec)$$

be such that

- Σ is a finite set of terminal symbols,
- N is a finite set of non-terminal symbols.
- $X_I \in N$ is the start-symbol of grammar.
- R is a finite set of production rules. Each production has the form $A \to \alpha$ or $A \to \beta+$, where A is a non-terminal and α is a sequence of terminals and non-terminals and $\beta \in \Sigma \cup N$. $A \to \beta+$ is a shortcut for set of rules: $A \to \beta, A \to \beta\beta, A \to \beta\beta\beta, \ldots$.
- $\prec$ is binary relation of $\Sigma \cup N$ such that $A \prec B$ if and only if there is a rule $A \to \alpha$ in R such that B belongs to α or there is a rule $A \to B+$.
- $\prec$ is a irreflexive and transitive partial order.

We will denote every subsequence parsed to a grammar symbol as a phrase.

Proposition 3. *Language L can be recognised by grammar defined above if and only if L is regular language.*

For example, for sequences from Table 5 the following grammar may be generated:

```
[name] ::= Alice | Bob | Charlie | David
[age] ::= young | old
[colour of hair] ::= brown | black
[number of hairs] ::= none | little | many
[known language] ::= Pascal | Ocaml | Cobol
[known languages] ::= [known language] |
     [known language] and [known language]
[name,] ::= [name] ,
[name list] ::= [name,] +
[names] ::= [name] | [name] and [name] | [name list] and [name]
[parents] ::= [names] | parents of [names]
[person] ::= [parents] | [parents] and [parents]
X_I ::= [person] are [age] |
     [person] has [colour of hair] hair |
     [person] know [known languages]
```

Names of the symbols in the grammar reflect the concept names. The grammar is ambiguous. The sequence s_4 may be parsed in two different ways which reflect two possible interpretations of sequence.

5.2 Data Sequence Representation

We are looking for all the possible derivation trees for a given data sequence and grammar.

We need representation that can describe ambiguous, partially parsed data. We represent it as directed acyclic graph whose edges are labelled by grammar symbols. We call it *the graph of syntactic decomposition.*

We represent data sequence as a graph that is a list. Formally, let $\{\sigma_i\}_1^n$, $\sigma_i \in \Sigma$ be the sequence. We create graph with vertexes $V = \{v_0, \ldots, v_n\}$ and set of edges $E = \{v_0 \xrightarrow{\sigma_1} v_1, \ldots, v_{n-1} \xrightarrow{\sigma_n} v_n\}$.

While applying the rule we find path in the graph with edge labels that match to the rule. Then we add to graph a new edge from beginning to end of the path labelled with rule production.

In order to apply the rule $A \to \alpha_1, \ldots, \alpha_k$ we find all paths

$$v_{a_0} \xrightarrow{\alpha_1} v_{a_1} \xrightarrow{\alpha_2} v_{a_2} \ldots v_{a_{k-1}} \xrightarrow{\alpha_k} v_{a_k}$$

and we add for each of them the edge

$$v_{a_0} \xrightarrow{A} v_{a_k}$$

to the graph.

While applying the $A \to \beta+$ rule, we find all paths

$$v_{a_0} \xrightarrow{\beta} v_{a_1} \xrightarrow{\beta} v_{a_2} \ldots v_{a_{k-1}} \xrightarrow{\beta} v_{a_k}$$

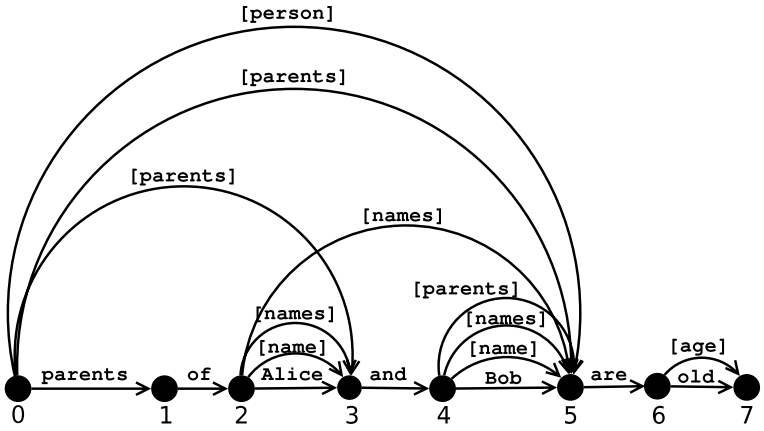

Fig. 1. Part of the graph of syntactic decomposition for sequence s_4

and we add for each of them the edge

$$v_{a_0} \xrightarrow{A} v_{a_k}$$

to the graph.

We will denote the edge labelled α such that $v_i \xrightarrow{\alpha} v_j$ by $\alpha_{i,j}$.

As a result of parsing process we obtain the edge from the beginning to the end of graph labelled by the start symbol of grammar.

5.3 Parser Algorithm

Having defined the data representation, we describe the parser algorithm.

We divide the set of symbols into layers: Let $N_0 = \Sigma$ and let

$$N_{n+1} = \{A : \exists A \to \alpha_1 \dots \alpha_k \forall i(\alpha_i \in N_n) \cup \exists A \to \beta+(\beta \in N_n)\}.$$

Now we divide the rules set R into layers. Let $R_{-1} = \emptyset$ and

$$R_n = \{A \to \alpha_1 \dots \alpha_k : \forall i \alpha_i \in N_n\} \cup \{A \to \beta+ : \beta \in N_n\} \setminus R_{n-1}.$$

Since we do not allow recurrent symbol to occur there is finite number of layers.

For example, for grammar created for sequences from Table 5 we obtain:

$$\begin{aligned}
N_0 = \{&\texttt{Alice}, \texttt{Bob}, \texttt{Charlie}, \texttt{David}, \texttt{young}, \texttt{old}, \texttt{brown}, \texttt{black},\\
&\texttt{none}, \texttt{little}, \texttt{many}, \texttt{Pascal}, \texttt{Ocaml}, \texttt{Cobol},\\
&\texttt{and}, \texttt{,}, \texttt{parents}, \texttt{of}, \texttt{are}, \texttt{has}, \texttt{know}, \texttt{hair}\}\\
N_1 = \{&\texttt{[name]}, \texttt{[age]}, \texttt{[colour of hair]},\\
&\texttt{[number of hairs]}, \texttt{[known language]}\}\\
N_2 = \{&\texttt{[known languages]}, \texttt{[name,]}\}\\
N_3 = \{&\texttt{[name list]}\}\\
N_4 = \{&\texttt{[names]}\}\\
N_5 = \{&\texttt{[parents]}\}\\
N_6 = \{&\texttt{[person]}\}\\
N_7 = \{&X_I\}
\end{aligned}$$

and

$R_0 =$
```
[name] ::= Alice | Bob | Charlie | David
[age] ::= young | old
[colour of hair] ::= brown | black
[number of hairs] ::= none | little | many
[known language] ::= Pascal | Ocaml | Cobol
```

$R_1 =$
```
[known languages] ::= [known language] |
      [known language] and [known language]
[name,] ::= [name] ,
```

etc.

Rules belonging to each layer are independent. Hence we may go through the sequence once for each layer and apply all matching rules simultaneously.

For each path in graph, the algorithm finds all rules that match to the path and add their production to graph. We begin with graph (V, E_0), where $E_0 = E$. We obtain graph (V, E_{n+1}) by applying to (V, E_n) rules from R_n. For each text's subsequence we find all its possible syntactic consequences.

In order to do it efficiently we create prefix tree out of every layer: For each rule $A \to \alpha_1 \dots \alpha_k$ in R_n we create path in the tree from the root labelled by symbols α_1 till α_k and we label the leaf tree node by A. For each node we merge paths that have identical labels.

Using this data structure we can apply all $A \to \alpha$ rules in layer in $\mathcal{O}(|E_n| l \log |\Sigma \cup E| + |E_n||R_n^+|)$ time, where

$$l = \max_{R_n}\{k : A \to \alpha_1 \dots \alpha_k \in R_n\}.$$

Since l, $\log|\Sigma \cup E|$ and number of layers is relatively small $|E_n|$ is crucial for parser performance.

For the different kinds of grammar parser may be replaced with other known in literature parsers [12].

5.4 Semantic Values of Grammar Symbols

In case of ambiguous grammar, the number of possible syntax derivation trees may be exponential to the sequence length. The concept of the graph of syntactic decomposition is their compact representation. The number of possible semantic values of the sequence is equal to the number of syntax derivation trees. That is why we cannot represent them directly. Instead, we distribute the semantic values across the graph of syntactic decomposition.

The meaning representation language formulae must have syntax coherent with the graph of syntactic decomposition. This requirement creates the dependence between the syntax of the meaning representation language and the process of translating data into axioms.

The formulae are spread across the graph in a way presented below.

Consider the edge $\alpha_{i,j}$ of the graph. This edge was created as the result of parsing a phrase. The phrase described an entity. We represent this entity by means of constant $a_{\alpha,i,j}$. We describe its properties derived from the phrase by

the formula of meaning representation language. We name this formula *semantic value of grammar symbol* and denote it as $[\![\alpha]\!]_{i,j}$. We assign the formula $[\![\alpha]\!]_{i,j}$ to the edge $\alpha_{i,j}$ on the implementation level.

The formula $[\![\alpha]\!]_{i,j}$ has the following structure:

$$[\![\alpha]\!]_{i,j} := \bigvee_{k=1}^{n} p_k(a_{\alpha,i,j}, a_{\alpha_1^k,i_1^k,j_1^k}, \ldots, a_{\alpha_{m_k}^k,i_{m_k}^k,j_{m_k}^k}) \wedge \bigwedge_{l=1}^{m_k} [\![\alpha_l^k]\!]_{i_l^k,j_l^k}.$$

Each $[\![\alpha_l^k]\!]_{i_l^k,j_l^k}$ is assigned to the edge $\alpha_{l\ i_l^k,j_l^k}^k$, so only the set of atomic formulae

$$\{p_1(a_{\alpha,i,j}, a_{\alpha_1^1,i_1^1,j_1^1}, \ldots, a_{\alpha_{m_k}^1,i_{m_k}^1,j_{m_k}^1}), \ldots$$
$$\ldots, p_n(a_{\alpha,i,j}, a_{\alpha_1^n,i_1^n,j_1^n}, \ldots, a_{\alpha_{m_k}^n,i_{m_k}^n,j_{m_k}^n})\}$$

must by associated with graph edge on the implementation level.

Semantics for a terminal symbol $\alpha_{i,j}$ is an one-argument predicate whose name is α and whose argument is the variable $a_{\alpha,i,j}$.

For example, for graph of syntactic decomposition presented on Fig. 1 we will obtain the following semantic values:

$$\begin{aligned}
[\![\texttt{[name]}]\!]_{2,3} &= \texttt{[name]}(a_{\texttt{[name]},2,3}, \texttt{Alice}) \\
[\![\texttt{[names]}]\!]_{2,5} &= \mathit{and}(a_{\texttt{[names]},2,5}, a_{\texttt{[name]},2,3}, a_{\texttt{[name]},4,5}) \wedge [\![\texttt{[name]}]\!]_{2,3} \wedge \\
&\quad \wedge [\![\texttt{[name]}]\!]_{4,5} \\
[\![\texttt{[parents]}]\!]_{0,3} &= \texttt{[parents]}(a_{\texttt{[parents]},0,3}, a_{\texttt{[names]},2,3}) \wedge [\![\texttt{[names]}]\!]_{2,3} \\
[\![\texttt{[person]}]\!]_{0,5} &= \big(\mathit{equal}(a_{\texttt{[person]},0,5}, a_{\texttt{[parents]},0,5}) \wedge [\![\texttt{[parents]}]\!]_{0,5}\big) \vee \\
&\quad \vee \big(\mathit{and}(a_{\texttt{[person]},0,5}, a_{\texttt{[parents]},0,3}, a_{\texttt{[parents]},4,5}) \wedge \\
&\quad \wedge [\![\texttt{[parents]}]\!]_{0,3} \wedge [\![\texttt{[parents]}]\!]_{4,5}\big).
\end{aligned}$$

5.5 Semantic Attachments

Semantic values of grammar symbols are constructed using semantic attachments of grammar rules. Semantic attachment are functions that compose semantics of greater objects out of semantics of smaller ones.

Let $A \to \alpha_1 \ldots \alpha_k$ be a syntactic rule and $f_{A\to\alpha_1\ldots\alpha_k}$ be a semantic attachment assigned to it. Assume that the rule was matched to the path $\alpha_{1,i_0,i_1}, \ldots, \alpha_{k,i_{k-1},i_k}$. As the rule was applied the symbol A_{i_0,i_k} was created.

The semantic value for A_{i_0,i_k} is constructed as follows: first we calculate the value of

$$f_{A\to\alpha_1\ldots\alpha_k}([\![\alpha_1]\!]_{i_0,i_1}, \ldots, [\![\alpha_k]\!]_{i_{k-1},i_k}).$$

We demand from the values of the semantic attachments to be predicate. Let

$$p(a_{A,i_0,i_k}, a_{\beta_1,j_1,k_1}, \ldots, a_{\beta_n,j_n,k_n}) := f_{A\to\alpha_1\ldots\alpha_k}([\![\alpha_1]\!]_{i_0,i_1}, \ldots, [\![\alpha_k]\!]_{i_{k-1},i_k}),$$

where every β_{i,j_i,k_i} belongs to $\{\alpha_{1,i_0,i_1}, \ldots, \alpha_{k,i_{k-1},i_k}\}$. Now we define semantics of A_{i_0,i_k} as

$$[\![A]\!]_{i_0,i_k} := p(a_{A,i_0,i_k}, a_{\beta_1,j_1,k_1}, \ldots, a_{\beta_n,j_n,k_n}) \wedge \bigwedge_{1\le i\le n} [\![\beta_i]\!]_{j_i,k_i}) \vee [\![A]\!]_{i_0,i_k}.$$

$[\![A]\!]_{i_0,i_k}$ on the right side of assignment is the semantic value of the edge A_{i_0,i_k} before the rule application. The semantic value for nonexistent edge is falsity.

The first argument of the predicate p is the constant a_{A,i_0,i_k} whose value is the entity described by phrase parsed to A_{i_0,i_k}.

The semantic attachment manipulates on formulae considering them as terms. It extracts parts of formulae and constructs the predicate p using them. In most cases it uses the constant pointing to the entity described by the predicate. The semantic attachment often concatenates the names of predicates that are the semantic values of terminal symbols

For example, the grammar created for sequences from Table 5 may have the following semantic attachments:

`[name]` ::= `Alice`
  `name`$(u_{\texttt{[name]}}, \texttt{Alice})$
`[age]` ::= `old`
  `age`$(u_{\texttt{[age]}}, \texttt{old})$
`[known language]` ::= `Pascal`
  `[known language]`$(u_{\texttt{[name]}}, \texttt{Pascal})$
`[known languages]` ::= `[known language]`
  $equal(u_{\texttt{[known languages]}}, u_{\texttt{[known language]}})$
`[known languages]` ::= `[known language]`$_1$ `and` `[known language]`$_2$
  $equal(u_{\texttt{[known languages]}}, u_{\texttt{[known language]}_1}, u_{\texttt{[known language]}_2})$
`[names]` ::= `[name]`
  $equal(u_{\texttt{[names]}}, u_{\texttt{[name]}})$
`[names]` ::= `[name]`$_1$ `and` `[name]`$_2$
  $and(u_{\texttt{[names]}}, u_{\texttt{[name]}_1}, u_{\texttt{[name]}_2})$
`[parents]` ::= `[names]`
  $equal(u_{\texttt{[parents]}}, u_{\texttt{[names]}})$
`[parents]` ::= `parents of [names]`
  `parents`$(u_{\texttt{[parents]}}, u_{\texttt{[names]}})$
`[person]` ::= `[parents]`
  $equal(u_{\texttt{[person]}}, u_{\texttt{[names]}})$
`[person]` ::= `[parents]`$_1$ `and` `[parents]`$_2$
  $and(u_{\texttt{[person]}}, u_{\texttt{[parents]}_1}, u_{\texttt{[parents]}_2})$
X_I ::= `[person] are [age]`
  $equal(u_{X_I}, u_{\texttt{[person]}}, u_{\texttt{[age]}})$
X_I ::= `[person] know [known languages]`
  $equal(u_{X_I}, u_{\texttt{[person]}}, u_{\texttt{[known languages]}})$.

The space complexity determines the representation of semantics. Each rule application add one predicate, so space complexity of semantics is proportional to the number of applied rules.

Thank to such representation of formulae we omit combinatorial explosion during the analysis of data with high ambiguity.

Various predicates generated for a given subsequence are possible descriptions of an entity. That is why we point that entity by the same constant in each predicate.

We connect different possible subsequence interpretations by means of alternative. Only one of them may be correct because only one of contradicting formulae may be consistent, ensuring that only one clause of alternative will be true. In case when text interpretations do not contradict, it could happen that a few clauses will be true (consistent with the facts described in the document) at the same time, despite the fact that only one of them could be meant do be written.

For for each $A \rightarrow \beta+$ rule we assign semantic action $f_{A\rightarrow\beta+}$ such that

$$[\![A]\!] = f_{A\rightarrow\beta+}([\![\beta_1]\!], [\![\beta_2]\!], \dots).$$

We represent semantic value of the symbol generated by accumulation rule as a graph, whose vertexes are constants that are arguments of predicate. Each path from beginning to ending vertex in such a graph represents a list of predicate arguments. The predicates are connected by alternative. We denote such a graph as *graph of logic structure* of accumulation symbol.

Size of semantics for accumulation rule is smaller than $\frac{n(n+1)}{2}$, where n is number of vertexes in the graph of syntax decomposition.

We may add a few different semantic attachments to a syntactic rule. Obtaining rules that are grammatically identical but differ on semantic level.

In the end of parsing process we obtain an edge labelled by start symbol of grammar. Its semantic value is a formula that contains every possible translation for the entire text into meaning representation language.

6 Set Approximations

Now, when we studied the process of generating axioms for a given data, we define rough sets for objects described by axioms written as the meaning representation language formulae.

The most important difference with the case of information systems is the extension of the definition of query that makes it suitable for data represented in a form of meaning representation language formulae.

Definition 9. *By a* query *we denote any formula* $\varphi(x)$ *of the form*

$$\exists_{x_1,\dots,x_n} \bigwedge_{i=1}^{k} p_i(a_1^i, \dots, a_{k_i}^i),$$

where each a_j^i *either is a variable belonging to* $\{x, x_1, \dots, x_n\}$ *or a constant.* x *is a free variable and* p_i *are predicates.*

For example, the query

$$\varphi_1(x) = \exists_{x_1} \texttt{colour of hair}(x, x_1) \wedge \texttt{similar}(x_1, \texttt{brown})$$

refers to the people whose hair has colour similar to brown. In Table 6, the formula $\varphi_1(x)$ is satisfied either if p_1 is the value of x or its value is p_2. p_1 and p_2 cannot be distinguished by formula $\varphi_1(x)$.

Table 6. A set of axioms

$$\mathcal{P}_1 \models \texttt{colour of hair}(p_1, h_1) \wedge \texttt{similar}(h_1, \texttt{brown})$$
$$\mathcal{P}_1 \models \texttt{colour of hair}(p_2, h_2) \wedge \texttt{similar}(h_2, \texttt{brown})$$

The indiscernibility is defined in the same way as for information systems (compare with Definition 6 in Section 3):

Definition 10. *Let $\mathbb{A}$ be a set of axioms. Let $\varphi(x)$ be a query with free variable x. Let u_1 and u_2 be constants. We say that u_1 and u_2 are* indiscernible by the query $\varphi(x)$ *if*

$$\big(\mathbb{A} \models \varphi(u_1)\big) \Longleftrightarrow \big(\mathbb{A} \models \varphi(u_2)\big).$$

In case of information systems, the set of objects' labels U were given. When data are represented as a set of axioms, we define the set of objects' labels U as the set of all constant symbols included in axioms.

Definition 11. *Let X be a subset of U. We say that X is* definable *by $\mathbb{A}$ iff there exist queries $\varphi_1(x), \ldots, \varphi_n(x)$ such that*

$$\forall u \in U \ \big(u \in X \Longleftrightarrow \mathbb{A} \models \varphi_1(u) \vee \cdots \vee \varphi_n(u)\big).$$

Nondefinable X may be approximated by two definable sets. The first one is called *lower approximation* of X, denoted by $\underline{\mathbb{A}}X$ and defined as

$$\bigcup\{Y \mid Y \subset X \wedge Y \text{ is definable by } \mathbb{A}\}.$$

The second set is called *upper approximation* of X, denoted by $\overline{\mathbb{A}}X$ and defined as

$$\bigcap\{Y \mid X \subset Y \wedge Y \text{ is definable by } \mathbb{A}\}.$$

$\overline{\mathbb{A}}X \subset U$ because every definable $Y \subset U$.

Lower and upper approximations are definable, so

$$\underline{\mathbb{A}}X = \{u \in U \mid \underline{\varphi}(u)\},$$

$$\overline{\mathbb{A}}X = \{u \in U \mid \overline{\varphi}(u)\}.$$

We consider rough set approximation as a pair of formula

$$\forall_{u \in U} \ \big(\underline{\varphi}(u) \Longrightarrow u \in X\big),$$

$$\forall_{u \in U} \ \big(\neg\overline{\varphi}(u) \Longrightarrow \neg u \in X\big),$$

providing that $\underline{\varphi}$ and $\overline{\varphi}$ are the strongest formulae for which the above implications holds.

The sequential data processing methodology presented in the Section above may be interpreted as rough set approximation. Syntactic rules are queries and semantic attachments define the measurements of the approximated sensor. As the effect of sequential data processing we obtain the upper approximation of structural sensor by means of means of sequential sensor measurements.

7 Conclusions

When we process data into meaning representation language formulae we obtain knowledge base, which we may use for various data mining applications.

We can look for information using concepts from the documents. We describe properties of desired objects by means of queries and then we find the set of objects that satisfy the query.

The meaning representation language formulae provide us features for clustering and classification of the structural objects. If we define a decision attribute, we may construct rule based classifiers that use queries as selectors in decision rules. We plan to adapt the classical rule generation algorithms, so they could analyse the information contained in the properties of structurally described objects.

One of the classification tasks is an object identification. The goal of object identification is to determine the object's ontological category. We use properties of objects extracted from the sequential data for that purpose.

Object descriptions extracted from the sequential data are often incomplete. Part of attributes is missing. When we identify the object's ontological category, we add the values of missing attributes to the set of axioms as 'lost' missing values.

We plan also to study the process of high level concept extraction: the algorithms for defining the concepts that are not included in data but only approximated.

We will search for automatic methods for generating ontologies and determining the structure of objects in a way that would provide features useful for further application.

The ontology is most important during the process of feature selection. In our case the development of grammar and semantics for processing the sequential data into the meaning representation language formulae. This suggest that both problems are tightly connected and should be studied together.

On the other hand we plan to derive methods of extracting features from visual data and to extend our system on numerical data sequences.

Acknowledgments

I would like to thank Professor Andrzej Skowron for the inspiration and comprehensive support during the writing of this paper.

The research has been supported by the grant N-N206-400234 from Ministry of Science and Higher Education of the Republic of Poland.

References

1. Ballesteros, S., Shepp, B.E.: Object Perception: Structure and Process. Lawrence Erlbaum Associates, Hillsdale (1989)
2. Bazan, J., Nguyen, S.H., Nguyen, H.S., Skowron, A.: Rough set methods in approximation of hierarchical concepts. In: Tsumoto, S., Słowiński, R., Komorowski, J., Grzymała-Busse, J.W. (eds.) RSCTC 2004. LNCS (LNAI), vol. 3066, pp. 346–355. Springer, Heidelberg (2004)

3. Bazan, J., Skowron, A., Peters, J.F., Synak, P.: Spatio-temporal approximate reasoning over complex objects. Fundamenta Informaticae 67(1-3), 249–269 (2005)
4. Demri, S., Orłowska, E.: Incomplete Information: Structure, Inference, Complexity. Springer, Heidelberg (2002)
5. Duda, O., Hart, P.E., Stark, D.G.: Pattern Classification. A Wiley-Interscience Publication John Wiley & Sons, Chichester (2001)
6. Düntsch, I., Gediga, G.: Rough set data analysis: A road to non-invasive knowledge discovery. Methodos, Bangor (2000)
7. Gruber, T.R.: A translation approach to portable ontologies. Knowledge Acquisition 5(2), 199–220 (1993)
8. Grzymała-Busse, J.W., Grzymała-Busse, W.J.: An Experimental Comparison of Three Rough Set Approaches to Missing Attribute Values. In: T. Rough Sets, vol. 6, pp. 31–50 (2007)
9. Grzymała-Busse, J.W.: A Rough Set Approach to Data with Missing Attribute Values. In: Wang, G.-Y., Peters, J.F., Skowron, A., Yao, Y. (eds.) RSKT 2006. LNCS (LNAI), vol. 4062, pp. 58–67. Springer, Heidelberg (2006)
10. Hanseth, O., Monteiro, E.: Modelling and the representation of reality: some implications of philosophy on practical systems development. Scandinavian Journal of Information Systems 6(1), 25–46 (1994)
11. Jaworski, W.: Learning Compound Decision Functions for Sequential Data in Dialog with Experts. In: Greco, S., Hata, Y., Hirano, S., Inuiguchi, M., Miyamoto, S., Nguyen, H.S., Słowiński, R. (eds.) RSCTC 2006. LNCS (LNAI), vol. 4259, Springer, Heidelberg (2006)
12. Jurafsky, D., Martin, J.H.: Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition. Prentice-Hall, Englewood Cliffs (2000)
13. Knuth, D.E.: The Genesis of Attribute Grammars. In: WAGA 1990, pp. 1–12 (1990)
14. Kryszkiewicz, M.: Rough Set Approach to Incomplete Information Systems. Inf. Sci. 112(1-4), 39–49 (1998)
15. Kryszkiewicz, M.: Properties of incomplete information systems in the framework of rough sets. In: Polkowski, L., Skowron, A. (eds.) Studies in Fuzziness and Soft Computing. Rough Sets in Knowledge Discovery 1, vol. 18, pp. 422–450. Physica-Verlag, Heidelberg (1998)
16. Latkowski, R.: On decomposition for incomplete data. Fundamenta Informaticae 54(1), 1–16 (2003)
17. Latkowski, R.: On indiscernibility relations for missing attribute values. In: Proceedings of the Workshop on Concurrency, Specification and Programming (CSP 2004), Caputh, Germany, September 24-26, 2004, Informatik-Bericht 170, strony pp. 330–335. Humboldt Universitt, Berlin (2004)
18. Latkowski, R., Mikołajczyk, M.: Data decomposition and decision rule joining for classification of data with missing values. LNCS Transactions on Rough Sets 3100(1), 299–320 (2004)
19. Latkowski, R.: Flexible indiscernibility relations for missing attribute values. Fundamenta Informaticae 67(1-3), 131–147 (2005)
20. Lipski Jr., W.: On Databases with Incomplete Information. J. ACM 28(1), 41–70 (1981)
21. Moens, M.-F.: Information Extraction: Algorithms and Prospects in a Retrieval Context. The Information Retrieval Series, vol. 21. Springer, Heidelberg (2006)
22. Marker, D.: Model Theory: An Introduction. Springer, Heidelberg (2002)

23. Mendelson, E.: Introduction to Mathematical Logic, 4th edn. International Thomson Publishing (1997)
24. Mohr, R., Pavlidis, T., Sanfeliu, A. (eds.): Structural Pattern Analysis. World Scientific, Teaneck (1990)
25. Nguyen, S.H., Bazan, J., Skowron, A., Nguyen, H.S.: Layered learning for concept synthesis. LNCS Transactions on Rough Sets 3100(1), 187–208 (2004)
26. Pavlidis, T.: Structural Pattern Recognition. Springer, Heidelberg (1977)
27. Pawlak, Z.: Information systems — theoretical foundations. Inf. Syst. 6(3), 205–218 (1981)
28. Pawlak, Z.: Rough Sets. International Journal of Computer and Information Sciences 11, 341–356 (1982)
29. Pawlak, Z.: Rough Sets. Theoretical Aspects of Reasoning about Data. Kluwer Academic Publishers, Dordrecht (1991)
30. Pawlak, Z., Skowron, A.: Rudiments of rough sets. Information Sciences 177(1), 3–27 (2007)
31. Pawlak, Z., Skowron, A.: Rough sets: Some extensions. Information Sciences 177(1), 28–40 (2007)
32. Pawlak, Z., Skowron, A.: Rough sets and Boolean reasoning. Information Sciences 177(1), 41–73 (2007)
33. Peters, J.F., Skowron, A. (eds.): Transactions on Rough Sets III. LNCS, vol. 3400. Springer, Heidelberg (2005)
34. Roddick, J.F., Hornsby, K., Spiliopoulou, M.: YABTSSTDMR - Yet Another Bibliography of Temporal, Spatial and Spatio-Temporal Mining Research. In: Uthurusamy, R., Unnikrishnan, K.P. (eds.) SIGKDD Temporal Data Mining Workshop, San Francisco, CA, pp. 167–175. ACM Press, Springer (2001)
35. Russ, J.C.: The Image Processing Handbook, Fourth Edition, 4th edn. CRC Press, Inc., Boca Raton (2002)
36. Russell, S.J., Norvig, P.: Artificial Intelligence. A Modern Approach. Prentice-Hall, Englewood Cliffs (1995)
37. Wittgenstein, L.: Tractatus Logico-Philosophicus. Routledge, London (1974)