Layered Learning for Concept Synthesis

Sinh Hoa Nguyen[1], Jan Bazan[2], Andrzej Skowron[3], and Hung Son Nguyen[3]

[1] Polish-Japanese Institute of Information Technology
Koszykowa 86, 02-008, Warsaw, Poland
[2] Institute of Mathematics,University of Rzeszów
Rejtana 16A, 35-959 Rzeszów, Poland
[3] Institute of Mathematics, Warsaw University
Banacha 2, 02-097 Warsaw, Poland
{hoa,bazan,skowron,son}@mimuw.edu.pl

Abstract. We present a hierarchical scheme for synthesis of concept approximations based on given data and domain knowledge. We also propose a solution, founded on rough set theory, to the problem of constructing the approximation of higher level concepts by composing the approximation of lower level concepts. We examine the effectiveness of the layered learning approach by comparing it with the standard learning approach. Experiments are carried out on artificial data sets generated by a road traffic simulator.
Keywords: Concept synthesis, hierarchical schema, layered learning, rough sets.

1 Introduction

Concept approximation is an important problem in data mining [10]. In a typical process of concept approximation we assume that there is given information consisting of values of conditional and decision attributes on objects from a finite subset (training set, sample) of the object universe and on the basis of this information one should induce approximations of the concept over the whole universe. In many practical applications, this standard approach may show some limitations. Learning algorithms may go wrong if the following issues are not taken into account:

Hardness of approximation: A target concept, being a compositions of some simpler one, is too complex, and cannot be approximated directly from feature value vectors. The simpler concepts may be either approximated directly from data (by attribute values) or given as domain knowledge acquired from experts. For example, in the hand-written digit recognition problem, the raw input data are $n \times n$ images, where $n \in [32, 1024]$ for typical applications. It is very hard to find an approximation of the target concept (digits) directly from values of n^2 pixels (attributes). The most popular approach to this problem is based on defining some additional, e.g., basic shapes, skeleton graph. These features must be easily extracted from images, and they are used to describe the target concept.

Efficiency: The fact that the complex concept can be decomposed into simpler one allows to decrease the complexity of the learning process. Each component can be learned separately on a piece of a data set and independent components can be learned in parallel. Moreover, dependencies between component concepts and their consequences can be approximated using domain knowledge and experimental data.

Expressiveness: Sometime, one can increase the readability of concept description by introducing some additional concepts. The description is more understandable, if it is expressed in natural language. For example, one can compare the readability of the following decision rules:

if *car speed is high* **and** *a distance to a preceding car is small* **then** *a traffic situation is dangerous*	**if** $car\_speed(X) > 176.7km/h$ **and** $distance\_to\_front\_car(X) < 11.4m$ **then** *a traffic situation is dangerous*

Layered learning [25] is an alternative approach to concept approximation. Given a hierarchical concept decomposition, the main idea is to synthesize a target concept gradually from simpler ones. One can imagine the decomposition hierarchy as a treelike structure (or acyclic graph structure) containing the target concept in the root. A learning process is performed through the hierarchy, from leaves to the root, layer by layer. At the lowest layer, basic concepts are approximated using feature values available from a data set. At the next layer more complex concepts are synthesized from basic concepts. This process is repeated for successive layers until the target concept is achieved.

The importance of hierarchical concept synthesis is now well recognized by researchers (see, e.g., [15, 14, 12]). An idea of hierarchical concept synthesis, in the rough mereological and granular computing frameworks has been developed (see, e.g., [15, 17, 18, 21]) and problems related to approximation compound concept are discussed, e.g., in [18, 22, 5, 24].

In this paper we concentrate on concepts that are specified by decision classes in decision systems [13]. The crucial factor in inducing concept approximations is to create the concepts in a way that makes it possible to maintain the acceptable level of precision all along the way from basic attributes to final decision. In this paper we discuss some strategies for concept composing founded on the rough set approach. We also examine effectiveness of the layered learning approach by comparison with the standard rule-based learning approach. The quality of the new approach will be verified relative to the following criteria: generality of concept approximation, preciseness of concept approximation, computation time required for concept induction, and concept description lengths. Experiments are carried out on an artificial data set generated by a road traffic simulator.

2 Concept approximation problem

In many real life situations, we are not able to give an exact definition of the concept. For example, frequently we are using adjectives such as “good”, “nice”, “young”, to describe some classes of peoples, but no one can give their exact

definition. The concept "young person" appears be easy to define by age, e.g., with the rule:

$$\textbf{if} age(X) \leq 30 \textbf{ then } X \text{ is young},$$

but it is very unnatural to explain that "Andy is not young because yesterday was his 30^{th} birthday". Such uncertain situations are caused either by the lack of information about the concept or by the richness of natural language.

Let us assume that there exists a concept X defined over the universe $\mathcal{U}$ of objects ($X \subseteq \mathcal{U}$). The problem is to find a description of the concept X, that can be expressed in a predefined descriptive language $\mathcal{L}$ consisting of formulas that are interpretable as subsets of $\mathcal{U}$. In general, the problem is to find a description of a concept X in a language $\mathcal{L}$ (e.g., consisting of boolean formulae defined over subset of attributes) assuming the concept is definable in another language $\mathcal{L}'$ (e.g., natural language, or defined by other attributes, called decision attributes).

Inductive learning is one of the most important approaches to concept approximation. This approach assumes that the concept X is specified partially, i.e., values of characteristic function of X are given only for objects from a *training sample* $U \subseteq \mathcal{U}$. Such information makes it possible to search for patterns in a given language $\mathcal{L}$ defined on the training sample sets included (or sufficiently included) into a given concept (or its complement). Observe that the approximations of a concept can not be defined uniquely from a given sample of objects. The approximations of the whole concept X are induced from given information on a sample U of objects (containing some positive examples from $X \cap U$ and negative examples from $U - X$). Hence, the quality of such approximations should be verified on new testing objects.

One should also consider uncertainty that may be caused by methods of object representation. Objects are perceived by some features (attributes). Hence, some objects become indiscernible with respect to these features. In practice, objects from $\mathcal{U}$ are perceived by means of vectors of attribute values (called information vectors or information signature). In this case, the language $\mathcal{L}$ consists of boolean formulas defined over accessible attributes such that their values are effectively measurable on objects. We assume that $\mathcal{L}$ is a set of formulas defining subsets of $\mathcal{U}$ and boolean combinations of formulas from $\mathcal{L}$ are expressible in $\mathcal{L}$.

Due to bounds on expressiveness of language $\mathcal{L}$ in the universe $\mathcal{U}$, we are forced to find some approximate rather than exact description of a given concept. There are different approaches to deal with uncertain and vague concepts like multi-valued logics, fuzzy set theory, or rough set theory. Using those approaches, concepts are defined by "multi-valued membership function" instead of classical "binary (crisp) membership relations" (set characteristic functions). In particular, rough set approach offers a way to establish membership functions that are data-grounded and significantly different from others.

In this paper, the input data set is represented in a form of information system or decision system. An *information system* [13] is a pair $\mathbb{S} = (U, A)$, where U is a non-empty, finite set of *objects* and A is a non-empty, finite set, of *attributes*. Each $a \in A$ corresponds to the function $a : U \to V_a$ called an

evaluation function, where V_a is called the *value set* of a. Elements of U can be interpreted as cases, states, patients, or observations.

For a given information system $\mathbb{S} = (U, A)$, we associate with any non-empty set of attributes $B \subseteq A$ *the B-information signature* for any object $x \in U$ by $inf_B(x) = \{(a, a(x)) : a \in B\}$. The set $\{inf_A(x) : x \in U\}$ is called the *A-information set* and it is denoted by $INF(\mathbb{S})$.

The above formal definition of information systems is very general and it covers many different systems such as database systems, or *information table* which is a two–dimensional array (matrix). In an information table, we usually associate its rows with objects (more precisely information vectors of objects), its columns with attributes and its cells with values of attributes.

In supervised learning, objects from a training set are pre-classified into several *categories* or *classes*. To deal with this type of data we use a special information systems called *decision systems* that are information systems of the form $\mathbb{S} = (U, A, dec)$, where $dec \notin A$ is a distinguished attribute called *decision*. The elements of attribute set A are called *conditions*.

In practice, decision systems contain a description of a finite sample U of objects from a larger (may be infinite) universe $\mathcal{U}$. Usually decision attribute is a characteristic function of an unknown concept or concepts (in the case of several classes). The main problem of learning theory is to generalize the decision function (concept description) partially defined on the sample U, to the universe $\mathcal{U}$. Without loss of generality for our considerations we assume that the domain V_{dec} of the decision dec is equal to $\{1, \ldots, d\}$. The decision dec determines a partition

$$U = CLASS_1 \cup \ldots \cup CLASS_d$$

of the universe U, where $CLASS_k = \{x \in U : dec(x) = k\}$ is called the k^{th} *decision class of* $\mathbb{S}$ for $1 \leq k \leq d$.

3 Concept approximation based on rough set theory

Rough set methodology for concept approximation can be described as follows (see [5]).

Definition 1. *Let $X \subseteq \mathcal{U}$ be a concept and let $U \subseteq \mathcal{U}$ be a finite sample of $\mathcal{U}$. Assume that for any $x \in U$ there is given information whether $x \in X \cap U$ or $x \in U - X$.* A rough approximation of the concept X *in a given language $\mathcal{L}$ (induced by the sample U) is any pair $(\mathbf{L}_\mathcal{L}, \mathbf{U}_\mathcal{L})$ satisfying the following conditions:*

1. $\mathbf{L}_\mathcal{L} \subseteq \mathbf{U}_\mathcal{L} \subseteq \mathcal{U}$,
2. $\mathbf{L}_\mathcal{L}, \mathbf{U}_\mathcal{L}$ *are expressible in the language $\mathcal{L}$, i.e., there exist two formulas $\phi_L, \phi_U \in \mathcal{L}$ such that $\mathbf{L}_\mathcal{L} = \{x \in \mathcal{U} : x$ satisfies $\phi_L\}$ and $\mathbf{U}_\mathcal{L} = \{x \in \mathcal{U} : x$ satisfies $\phi_U\}$,*
3. $\mathbf{L}_\mathcal{L} \cap U \subseteq X \cap U \subseteq \mathbf{U}_\mathcal{L} \cap U$;
4. *the set $\mathbf{L}_\mathcal{L}$ ($\mathbf{U}_\mathcal{L}$) is maximal (minimal) in the family of sets definable in $\mathcal{L}$ satisfying (3).*

The sets $\mathbf{L}_{\mathcal{L}}$ and $\mathbf{U}_{\mathcal{L}}$ are called the *lower approximation* and the *upper approximation* of the concept $X \subseteq \mathcal{U}$, respectively. The set $\mathbf{BN} = \mathbf{U}_{\mathcal{L}} \setminus \mathbf{L}_{\mathcal{L}}$ is called the *boundary region of approximation* of X. The set X is called *rough* with respect to its approximations $(\mathbf{L}_{\mathcal{L}}, \mathbf{U}_{\mathcal{L}})$ if $\mathbf{L}_{\mathcal{L}} \neq \mathbf{U}_{\mathcal{L}}$, otherwise X is called *crisp* in $\mathcal{U}$. The pair $(\mathbf{L}_{\mathcal{L}}, \mathbf{U}_{\mathcal{L}})$ is also called the *rough set* (for the concept X). Condition (3) in the above list can be substituted by inclusion to a degree to make it possible to induce approximations of higher quality of the concept on the whole universe $\mathcal{U}$. In practical applications the last condition in the above definition can be hard to satisfy. Hence, by using some heuristics we construct sub-optimal instead of maximal or minimal sets. Also, since during the process of approximation construction we only know U it may be necessary to change the approximation after we gain more information about new objects from $\mathcal{U}$. The rough approximation of concept can be also defined by means of a rough membership function.

Definition 2. *Let $X \subseteq \mathcal{U}$ be a concept and let $U \subseteq \mathcal{U}$ be a finite sample. A function $f : \mathcal{U} \to [0, 1]$ is called a rough membership function of the concept $X \subseteq \mathcal{U}$ if and only if $(\mathbf{L}_f, \mathbf{U}_f)$ is an approximation of X (induced from sample U) where $\mathbf{L}_f = \{x \in \mathcal{U} : f(x) = 1\}$ and $\mathbf{U}_f = \{x \in \mathcal{U} : f(x) > 0\}$.*

Note that the proposed approximations are not defined uniquely from information about X on the sample U. They are obtained by inducing the concept $X \subseteq \mathcal{U}$ approximations from such information. Hence, the quality of approximations should be verified on new objects and information about classifier performance on new objects can be used to improve gradually concept approximations. Parameterizations of rough membership functions corresponding to classifiers make it possible to discover new relevant patterns on the object universe extended by adding new (testing) objects. In the following sections we present illustrative examples of such parameterized patterns. By tuning parameters of such patterns one can obtain patterns relevant for concept approximation on the extended training sample by some testing objects.

3.1 Case-based rough approximations

For case-base reasoning methods, like kNN (k nearest neighbors) classifier [1, 6], we define a distance (similarity) function between objects $\delta : \mathcal{U} \times \mathcal{U} \to [0, \infty)$. The problem of determining the distance function from given data set is not trivial, but in this paper, we assume that such a distance function has been already defined for all object pairs.

In kNN classification methods (kNN classifiers), the decision for a new object $x \in \mathcal{U} - U$ is made by decisions of k objects from U that are nearest to x with respect to the distance function δ. Usually, k is a parameter defined by an expert or automatically constructed by experiments from data. Let us denote by $NN(x; k)$ the set of k nearest neighbors of x, and by $n_i(x) = |NN(x; k) \cap CLASS_i|$ the number of objects from $NN(x; k)$ that belong to i^{th} decision class. The kNN classifiers often use a voting algorithm for decision making, i.e.,

$$dec(x) = Voting(\langle n_1(x), \ldots, n_d(x) \rangle) = \arg\max_i n_i(x),$$

In case of imbalanced data, the vector $\langle n_1(x), \ldots, n_d(x)\rangle$ can be scaled with respect to the global class distribution before applying the voting algorithm.

Rough approximation based on the set $NN(x;k)$, that is, an extension of a kNN classifier can be defined as follows. Assume that $0 \leq t_1 < t_2 < k$ and let us consider for i^{th} decision class $CLASS_i \subseteq \mathcal{U}$ a function with parameters t_1, t_2 defined on any object $x \in \mathcal{U}$ by:

$$\mu^{t_1,t_2}_{CLASS_i}(x) = \begin{cases} 1 & \text{if } n_i(x) \geq t_2 \\ \frac{n_i(x)-t_1}{t2-t1} & \text{if } n_i(x) \in (t_1, t_2) \\ 0 & \text{if } n_i(x) \leq t_1, \end{cases} \tag{1}$$

where $n_i(x)$ is the i^{th} coordinate in the class distribution $ClassDist(NN(x;k)) = \langle n_1(x), \ldots, n_d(x)\rangle$ of $NN(x;k)$.

Let us assume that parameters t^o_1, t^o_2 have been chosen in such a way that the above function satisfies for every $x \in U$ the following conditions:

$$\text{if } \mu^{t^o_1,t^o_2}_{CLASS_i}(x) = 1 \text{ then } [x]_A \subseteq CLASS_i \cap U, \tag{2}$$

$$\text{if } \mu^{t^o_1,t^o_2}_{CLASS_i}(x) = 0 \text{ then } [x]_A \cap (CLASS_i \cap U) = \emptyset, \tag{3}$$

where $[x]_A = \{y \in U : inf_A(x) = inf_A(y)\}$ denotes the indiscernibility class defined by x relatively to a fixed set of attributes A.

Then the function $\mu^{t^o_1,t^o_2}_{CLASS_i}$ considered on $\mathcal{U}$ can be treated as the rough membership function of the i^{th} decision class. It is the result of induction on $\mathcal{U}$ of the rough membership function of i^{th} decision class restricted to the sample U. The function $\mu^{t^o_1,t^o_2}_{CLASS_i}$ defines a rough approximations $\mathbf{L}_{kNN}(CLASS_i)$ and $\mathbf{U}_{kNN}(CLASS_i)$ of i^{th} decision class $CLASS_i$. For any object $x \in \mathcal{U}$ we have

$$x \in \mathbf{L}_{kNN}(CLASS_i) \Leftrightarrow n_i(x) \geq t^o_2 \text{ and } x \in \mathbf{U}_{kNN}(CLASS_i) \Leftrightarrow n_i(x) \geq t^o_1.$$

Certainly, one can consider in conditions (2)-(3) an inclusion to a degree and equality to a degree instead of the crisp inclusion and the crisp equality. Such degrees parameterize additionally extracted patterns and by tuning them one can search for relevant patterns.

As we mentioned above kNN methods have some drawbacks. One of them is caused by the assumption that the distance function is defined *a priori* for all pairs of objects, that is not the case for many complex data sets. In the next section we present an alternative way to define rough approximations from data.

3.2 Rule-based rough approximations

In this section we describe the rule-based rough set approach to approximations.

Let $\mathbb{S} = (U, A, dec)$ be a decision table. A *decision rule* for the k^{th} decision class is any expression of the form

$$(a_{i_1} = v_1) \wedge \ldots \wedge (a_{i_m} = v_m) \Rightarrow (dec = k), \tag{4}$$

where $a_{i_j} \in A$ and $v_j \in V_{a_{i_j}}$. Any decision rule $\mathbf{r}$ of the form (4) can be characterized by following parameters:

– $length(\mathbf{r})$: the number of descriptors in the the left hand side of implication;
– $[\mathbf{r}]$ = *carrier of* $\mathbf{r}$, i.e., the set of objects satisfying the premise of $\mathbf{r}$,
– $support(\mathbf{r}) = card([\mathbf{r}] \cap CLASS_k)$,
– $confidence(\mathbf{r})$ introduced to measure the truth degree of the decision rule:

$$confidence(\mathbf{r}) = \frac{support(\mathbf{r})}{card([\mathbf{r}])}, \tag{5}$$

The decision rule $\mathbf{r}$ is called *consistent* with $\mathbb{S}$ if $confidence(\mathbf{r}) = 1$.

Among decision rule generation methods developed using the rough set approach one of the most interesting is related to *minimal consistent decision rules*. Given a decision table $\mathbb{S} = (U, A, dec)$, the rule $\mathbf{r}$ is called a *minimal consistent decision rule* (with $\mathbb{S}$) if is consistent with $\mathbb{S}$ and any decision rule $\mathbf{r}'$ created from $\mathbf{r}$ by removing any of descriptors from the left hand side of $\mathbf{r}$ is not consistent with $\mathbb{S}$. The set of all minimal consistent decision rules for a given decision table $\mathbb{S}$, denoted by $Min\_Cons\_Rules(\mathbb{S})$, can be computed by extracting from the decision table *object oriented reducts* (called also local reducts relative to objects) [3, 9, 26].

The elements of $Min\_Cons\_Rules(\mathbb{S})$ can be treated as interesting, valuable and useful patterns in data and used as a knowledge base in classification systems. Unfortunately, the number of such patterns can be exponential with respect to the size of a given decision table [3, 9, 26, 23]. In practice, we must apply some heuristics, like rule filtering or object covering, for selection of subsets of decision rules

Given a decision table $\mathbb{S} = (U, A, dec)$. Let us assume that $\mathbf{RULES}(\mathbb{S})$ is a set of decision rules induced by some rule extraction method. For any object $x \in \mathcal{U}$, let $MatchRules(\mathbb{S}, x)$ be the set of rules from $\mathbf{RULES}(\mathbb{S})$ supported by x. One can define the rough membership function $\mu_{CLASS_k} : \mathcal{U} \to [0, 1]$ for the concept determined by $CLASS_k$ as follows:

1. Let $\mathbf{R}_{yes}(x)$ be the set of all decision rules from $MatchRules(\mathbb{S}, x)$ for k^{th} class and let $\mathbf{R}_{no}(x) \subset MatchRules(\mathbb{S}, x)$ be the set of decision rules for other classes.
2. We define two real values $w_{yes}(x), w_{no}(x)$, called "for" and "against" weights for the object x by:

$$w_{yes}(x) = \sum_{\mathbf{r} \in \mathbf{R}_{yes}(x)} strength(\mathbf{r}) \qquad w_{no}(x) = \sum_{\mathbf{r} \in \mathbf{R}_{no}(x)} strength(\mathbf{r}) \tag{6}$$

   where $strength(\mathbf{r})$ is a normalized function depending on *length*, *support*, $confidence$ of $\mathbf{r}$ and some global information about the decision table $\mathbb{S}$ like table size, class distribution (see [3]).
3. One can define the value of $\mu_{CLASS_k}(x)$ by

$$\mu_{CLASS_k}(x) = \begin{cases} \text{undetermined} & \text{if } \max(w_{yes}(x), w_{no}(x)) < \omega \\ 0 & \text{if } w_{no}(x) - w_{yes}(x) \geq \theta \text{ and } w_{no}(x) > \omega \\ 1 & \text{if } w_{yes}(x) - w_{no}(x) \geq \theta \text{ and } w_{yes}(x) > \omega \\ \frac{\theta + (w_{yes}(x) - w_{no}(x))}{2\theta} & \text{in other cases} \end{cases}$$

where ω, θ are parameters set by user. These parameters make it possible in a flexible way to control the size of boundary region for the approximations established according to Definition 2.

Let us assume that for $\theta = \theta_o > 0$ the above function satisfies for every $x \in U$ the following conditions:

$$\text{if } \mu^{\theta_o}_{CLASS_k}(x) = 1 \text{ then } [x]_A \subseteq CLASS_k \cap U, \tag{7}$$

$$\text{if } \mu^{\theta_o}_{CLASS_k}(x) = 0 \text{ then } [x]_A \cap (CLASS_k \cap U) = \emptyset, \tag{8}$$

where $[x]_A = \{y \in U : inf_A(x) = inf_A(y)\}$ denotes the indiscernibility class defined by x with respect to the set of attributes A.

Then the function $\mu^{\theta_o}_{CLASS_k}$ considered on $\mathcal{U}$ can be treated as the rough membership function of the k^{th} decision class. It is the result of induction on $\mathcal{U}$ of the rough membership function of k^{th} decision class restricted to the sample U. The function $\mu^{\theta_o}_{CLASS_k}$ defines a rough approximations $\mathbf{L}_{rule}(CLASS_k)$ and $\mathbf{U}_{rule}(CLASS_k)$ of k^{th} decision class $CLASS_k$, where $\mathbf{L}_{rule}(CLASS_k) = \{x : w_{yes}(x) - w_{no}(x) \geq \theta_o\}$ and $\mathbf{U}_{rule}(CLASS_k) = \{x : w_{yes}(x) - w_{no}(x) \geq -\theta_o\}$.

4 Hierarchical scheme for concept synthesis

In this section we present general layered learning scheme for concept synthesizing. We recall the main principles of the layered learning paradigm [25].

1. Layered learning is designed for domains that are too complex for learning a mapping directly from the input to the output representation. The layered learning approach consists of breaking a problem down into several task layers. At each layer, a concept needs to be acquired. A learning algorithm solves the local concept-learning task.
2. Layered learning uses a bottom-up incremental approach to hierarchical concept decomposition. Starting with low-level concepts, the process of creating new sub-concepts continues until the high-level concepts, that deal with the full domain complexity, are reached. The appropriate learning granularity and sub-concepts to be learned are determined as a function of the specific domain. Concept decomposition in layered learning is not automated. The layers and concept dependencies are given as a background knowledge of the domain.
3. Sub-concepts may be learned independently and in parallel. Learning algorithms may be different for different sub-concepts in the decomposition hierarchy. Layered learning is effective for huge data sets and it is useful for adaptation when a training set changes dynamically.
4. The key characteristic of layered learning is that each learned layer directly affects learning at the next layer.

When using the layered learning paradigm, we assume that the target concept can be decomposed into simpler ones called sub-concepts. A hierarchy of concepts has a treelike structure. A higher level concept is constructed from those concepts in lower levels. We assume that a concept decomposition hierarchy is given by domain knowledge [18, 21]. However, one should observe that concepts and dependencies among them represented in domain knowledge are expressed often in natural language. Hence, there is a need to approximate such concepts and such dependencies as well as the whole reasoning. This issue is directly related to the computing with words paradigm [27, 28] and to rough-neural approach [12], in particular to rough mereological calculi on information granules (see, e.g., [15, 17, 16, 19, 18]).

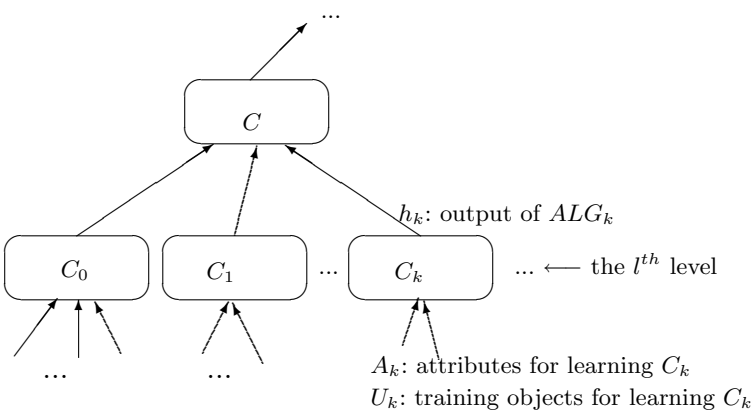

Fig. 1. Hierarchical scheme for concept approximation

The goal of a layered learning algorithm is to construct a scheme for concept composition. This scheme is a structure consisting of levels. Each level consists of concepts $(C_0, C_1, ..., C_n)$. Each concept C_k is defined as a tuple

$$C_k = (U_k, A_k, O_k, ALG_k, h_k), \tag{9}$$

where (Figure 1):

- U_k is a set of objects used for learning the concept C_k,
- A_k is the set of attributes relevant for learning the concept C_k,
- O_k is the set of outputs used to define the concept C_k,
- ALG_k is the algorithm used for learning the function mapping vector values over A_k into O_k,
- h_k is the hypothesis returned by the algorithm ALG_k as a result of its run on the training set U_k.

The hypothesis h_k of the concept C_k in a current level directly affects the next level in the following ways:

1. h_k is used to construct a set of training examples U of a concept C in the next level, if C is a direct ancestor of C_k in the decomposition hierarchy.
2. h_k is used to construct a set of features A of a concept C in the next level, if C is a direct ancestor of C_k in the decomposition hierarchy.

To construct a layered learning algorithm, for any concept C_k in the concept decomposition hierarchy, one must solve the following problems:

1. Define a set of training examples U_k used for learning C_k. A training set in the lowest level are subsets of an input data set. The training set U_k at the higher level is composed from training sets of sub-concepts of C_k.
2. Define an attribute set A_k relevant to approximate the concept C_k. In the lowest level the attribute set A_k is a subset of an available attribute set. In higher levels the set A_k is created from attribute sets of sub-concepts of C_k, from an attribute set of input data and/or they are newly created attributes. The attribute set A_k is chosen depending on the domain of the concept C_k.
3. Define an output set to describe the concept C_k.
4. Choose an algorithm to learn the concept C_k that is based on a diven object set and on the defined attribute set.

In the next section we discuss in detail methods for concept synthesis. The foundation of our methods is rough set theory. We have already presented some preliminaries of rough set theory as well as parameterized methods for basic concepts approximation. They are a generalization of existing rough set based methods. Let us describe strategies for concept composing from sub-concepts.

4.1 Approximation of compound concept

We assume that a concept hierarchy H is given. A training set is represented by decision table $\mathbb{S}_S = (U, A, D)$. D is a set of decision attributes. Among them are decision attributes corresponding to all basic concepts and a decision attribute for the target concept. Decision values indicate if an object belong to basic concepts or to the target concept, respectively.

Using information available from a concept hierarchy for each basic concept C_b, one can create a training decision system $\mathbb{S}_{C_b} = (U, A_{C_b}, dec_{C_b})$, where $A_{C_b} \subseteq A$, and $dec_{C_b} \in D$. To approximate the concept C_b one can apply any classical method (e.g., k-NN, supervised clustering, or rule-based approach [7, 11]) to the table $\mathbb{S}_{C_b}$. For example, one can use a case-based reasoning approach presented in Section 3.1 or a rule-based reasoning approach proposed in Section 3.2 for basic concept approximation. In further discussion we assume that basic concepts are approximated by rule based classifiers derived from relevant decision tables.

To avoid overly complicated notation let us limit ourselves to the case of constructing compound concept approximation on the basis of two simpler concept approximations. Assume we have two concepts C_1 and C_2 that are given to us in the form of rule-based approximations derived from decision systems $\mathbb{S}_{C_1} = (U, A_{C_1}, dec_{C_1})$ and $\mathbb{S}_{C_1} = (U, A_{C_1}, dec_{C_1})$. Henceforth, we are given two

rough membership functions $\mu_{C_1}(x)$, $\mu_{C_2}(x)$. These functions are determined with use of parameter sets $\{w_{yes}^{C_1}, w_{no}^{C_1}, \omega^{C_1}, \theta^{C_1}\}$ and $\{w_{yes}^{C_2}, w_{no}^{C_2}, \omega^{C_2}, \theta^{C_2}\}$, respectively. We want to establish a similar set of parameters $\{w_{yes}^{C}, w_{no}^{C}, \omega^{C}, \theta^{C}\}$ for the target concept C, which we want to describe with use of rough membership function μ_C. As previously noted, the parameters ω, θ controlling of the boundary region are user-configurable. But, we need to derive $\{w_{yes}^{C}, w_{no}^{C}\}$ from data. The issue is to define a decision system from which rules used to define approximations can be derived.

We assume that both simpler concepts C_1, C_2 and the target concept C are defined over the same universe of objects $\mathcal{U}$. Moreover, all of them are given on the same sample $U \subset \mathcal{U}$. To complete the construction of the decision system $\mathbb{S}_C = (U, A_C, dec_C)$, we need to specify the conditional attributes from A_C and the decision attribute dec_C. The decision attribute value $dec_C(x)$ is given for any object $x \in U$. For conditional attributes, we assume that they are either rough membership functions for simpler concepts (i.e., $A_C = \{\mu_{C_1}(x), \mu_{C_2}(x)\}$) or weights for simpler concepts (i.e., $A_C = \{w_{yes}^{C_1}, w_{no}^{C_1}, w_{yes}^{C_2}, w_{no}^{C_2}\}$). The output set O_i for each concept C_i, where $i = 1, 2$, consists of one attribute that is a rough membership function μ_{C_i} in the first case or two attributes $w_{yes}^{C_i}, w_{no}^{C_i}$ that describe fitting degrees of objects to the concept C_i and its complement, respectively. The rule-based approximations of the concept C are created by extracting rules from $\mathbb{S}_C$.

It is important to observe that such rules describing C use attributes that are in fact classifiers themselves. Therefore, in order to have a more readable and intuitively understandable description as well as more control over the quality of approximation (especially for new cases), it pays to stratify and interpret attribute domains for attributes in A_C. Instead of using just a value of a membership function or weight we would prefer to use linguistic statements such as *"the likelihood of the occurrence of C_1 is low"*. In order to do that we have to map the attribute value sets onto some limited family of subsets. Such subsets are then identified with notions such us *"certain", "low", "high"* etc. It is quite natural, especially in case of attributes being membership functions, to introduce linearly ordered subsets of attribute ranges, e.g., $\{negative, low, medium, high, positive\}$. That yields a fuzzy-like layout of attribute values. One may (and in some cases should) consider also the case when these subsets overlap. Then, there may be more linguistic value attached to attribute values since variables like *low* or *medium* appear.

Stratification of attribute values and introduction of a linguistic variable attached to the strata serves multiple purposes. First, it provides a way for representing knowledge in a more human-readable format since if we have a new situation (new object $x^* \in \mathcal{U} \setminus U$) to be classified (checked against compliance with concept C) we may use rules like:

If *compliance of x^* with C_1 is high or medium* ***and*** *compliance of x^* with C_2 is high* ***then*** $x^* \in C$.

Another advantage in imposing the division of attribute value sets lays in extended control over flexibility and validity of system constructed in this way.

As we may define the linguistic variables and corresponding intervals, we gain the ability of making a system more stable and inductively correct. In this way we control the general layout of boundary regions for simpler concepts that contribute to construction of the target concept. The process of setting the intervals for attribute values may be performed by hand, especially when additional background information about the nature of the described problem is available. One may also rely on some automated methods for such interval construction, such as, e.g., clustering, template analysis and discretization. Some extended discussion on the foundations of this approach, which is related to rough-neural computing [12, 18] and computing with words can be found in [24, 20].

Algorithm 1 Layered learning algorithm

Input: Decision system $\mathbb{S} = (U, A, d)$, concept hierarchy H;
Output: Scheme for concept composition

```
1: begin
2:   for l := 0 to max_level do
3:     for (any concept C_k at the level l in H) do
4:       if l = 0 then
5:         U_k := U;
6:         A_k := B;
             // where B ⊆ A is a set relevant to define C_k
7:       else
8:         U_k := U;
9:         A_k = ⋃ O_i;
             // for all sub-concepts C_i of C_k, where O_i is the output vector of C_i
10:        Generate a rule set RULE(C_k) to determine the approximation of C_k;
11:        for any object x ∈ U_k do
12:          generate the output vector (w_yes^{C_k}(x), w_no^{C_k}(x));
               // where w_yes^{C_k}(x) is a fitting degree of x to the concept C_k
               // and w_no^{C_i}(x) is a fitting degree of x to the complement of C_k.
13:        end for
14:      end if
15:    end for
16:  end for
17: end
```

Algorithm 1 is the layered learning algorithm used in our experiments.

5 Experimental results

To verify effectiveness of layered learning approach, we have implemented Algorithm 1 for concept composition presented in Section 4.1. The experiments were performed on data generated by road traffic simulator. In the following section we present a description of the simulator.

5.1 Road traffic simulator

The road simulator is a computer tool that generates data sets consisting of recording vehicle movements on the roads and at the crossroads. Such data sets are used to learn and test complex concept classifiers working on information coming from different devices and sensors monitoring the situation on the road.

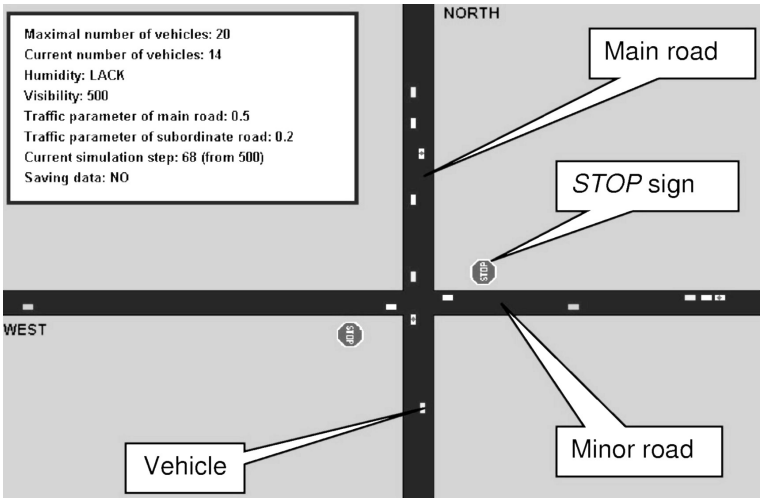

Fig. 2. Left: the board of simulation.

A driving simulation takes place on a board (see Figure 2) that presents a crossroads together with access roads. During the simulation the vehicles may enter the board from all four directions that is east, west, north and south. The vehicles coming to the crossroads form the south and north have the right of way in relation to the vehicles coming from the west and east. Each of the vehicles entering the board has only one goal - to drive through the crossroads safely and leave the board.

Both the entering and exiting roads of a given vehicle are determined at the beginning, that is, at the moment the vehicle enters the board. Each vehicle may perform the following maneuvers during the simulation like passing, overtaking, changing direction (at the crossroads), changing lane, entering the traffic from the minor road into the main road, stopping, pulling out.

Planning each vehicle's further steps takes place independently in each step of the simulation. Each vehicle, "observing" the surrounding situation on the road, keeping in mind its destination and its own parameters, makes an independent decision about its further steps; whether it should accelerate, decelerate and what (if any) maneuver should be commenced, continued, or stopped.

We associate the simulation parameters with the readouts of different measuring devices or technical equipment placed inside the vehicle or in the outside

environment (e.g., by the road, in a helicopter observing the situation on the road, in a police car). These are devices and equipment playing the role of detecting devices or converters meaning sensors (e.g., a thermometer, range finder, video camera, radar, image and sound converter). The attributes taking the simulation parameter values, by analogy to devices providing their values will be called sensors. Here are exemplary sensors: distance from the crossroads (in screen units), vehicle speed, acceleration and deceleration, etc.

Apart from sensors the simulator registers a few more attributes, whose values are determined based on the sensor's values in a way determined by an expert. These parameters in the present simulator version take over the binary values and are therefore called concepts. Concepts definitions are very often in a form of a question which one can answer YES, NO or DOES NOT CONCERN (NULL value). In Figure 3 there is an exemplary relationship diagram for some concepts that are used in our experiments.

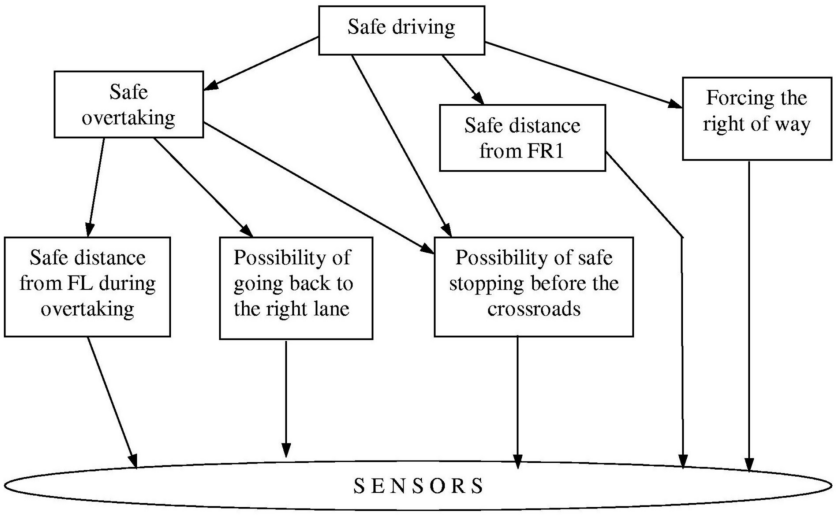

Fig. 3. The relationship diagram for exemplary concepts

During the simulation, when a new vehicle appears on the board, its so called driver's profile is determined and may not be changed until it disappears from the board. It may take one of the following values: a very careful driver, a careful driver and a careless driver. Driver's profile is the identity of the driver and according to this identity further decisions as to the way of driving are made. Depending on the driver's profile and weather conditions speed limits are determined, which cannot be exceeded. The humidity of the road influences the length of braking distance, for depending on humidity different speed changes take place within one simulation step, with the same braking mode. The driver's profile influences the speed limits dictated by visibility. If another vehicle is

invisible for a given vehicle, this vehicle is not taken into consideration in the independent planning of further driving by a given car. Because this may cause dangerous situations, depending on the driver's profile, there are speed limits for the vehicle.

During the simulation data may be generated and stored in a text file. The generated data are in a form of information table. Each line of the board depicts the situation of a single vehicle and the sensors' and concepts' values are registered for a given vehicle and its neighboring vehicles. Within each simulation step descriptions of situations of all the vehicles are saved to file.

5.2 Experiment description

A number of different data sets have been created with the road traffic simulator. They are named by *c*xx_*s*yyy, where xx is the number of cars and yyy is the number of time units of the simulation process. The following data sets: *c10_s100, c10_s200, c10_s300, c10_s400, c10_s500, c20_s500* have been generated for our experiments. Let us emphasize that the first data set consists of about 800 situations, whereas the last data set is the largest one, which can be generated by the simulator. This data set consists of about 10000 situations. Every data set has 100 attributes and has imbalanced class distribution, i.e., about $6\% \pm 2\%$ of situations are unsave.

Every data set *c*xx_*s*yyy was divided randomly into two subsets *c*xx_*s*yyy.trn and *c*xx_*s*yyy.test with proportion 80% and 20%, respectively. The data sets of form *c*xx_*s*yyy.trn are used in learning the concept approximations.

We consider two testing models called *testing for similar situations* and *testing for new situations*. They are described as follows:

Model I: *Testing for similar situations*. This model uses the data sets of the form *c*xx_*s*yyy.test for testing the quality of approximation algorithms. The situations, which are used in this testing model, are generated from the same simulation process as the training situations.

Model II: *Testing for new situations*. This model uses data from a new simulation process. In this model, we create new data sets using the simulator. They are named by *c10_s100N, c10_s200N, c10_s300N, c10_s400N, c10_s500N, c20_s500N*, respectively.

We compare the quality of two learning approaches called *RS rule-based learning* (**RS**) and *RS-layered learning* (**RS-L**). In the first approach, we employed the RSES system [4] to generate the set of minimal decision rules and classified the situations from testing data. The conflicts are resolved by simple voting strategy. The comparison analysis is performed with respect to the following criteria:

1. accuracy of classification,
2. covering rate of new cases (generality),
3. computing time necessary for classifier synthesis, and
4. size of rule set used for target concept approximation.

In the layered learning approach, from training table we create five sub-tables to learn five basic concepts (see Figure 3): C_1: "*safe distance from FL during overtaking*," C_2: "*possibility of safe stopping before crossroads*," C_3: "*possibility of going back to the right lane*," C_4: "*safe distance from preceding car*," C_5: "*forcing the right of way.*"

These tables are created using information available from the concept decomposition hierarchy. A concept in the next level is C_6: "*safe overtaking*". C_6 is located over the concepts C_1, C_2 and C_3 in the concept decomposition hierarchy. To approximate concept C_6, we create a table with three conditional attributes. These attributes describe fitting degrees of object to concepts C_1, C_2, C_3, respectively. The decision attribute has three values YES, NO, or $NULL$ corresponding to the cases of overtaking made by car: safe, not safe, not applicable.

The target concept C_7: "*safe driving*" is located at the third level of the concept decomposition hierarchy. The concept C_7 is obtained by composition from concepts C_4, C_5 and C_6. To approximate C_7 we also create a decision table with three attributes, representing fitting degrees of objects to the concepts C_4, C_5, C_6, respectively. The decision attribute has two possible values YES or NO if a car is satisfying global safety condition, or not, respectively.

Classification accuracy: As we mentioned before, the decision class "safe_driving = YES" is dominating in all training data sets. It takes above 90% of training sets. Sets of training examples belonging to the "NO" class are small relative to the training set size. Searching for approximation of the "NO" class with the high precision and generality is a challenge for learning algorithms. In experiments we concentrate on approximation of the "NO" class.

In Table 1 we present the classification accuracy of RS and RS-L classifiers for the first of testing models. It means that training sets and test sets are disjoint and samples are chosen from the same simulation data set.

Table 1. Classification accuracy for the first testing model

Testing model I	**Total accuracy**		**Accuracy of YES**		**Accuracy of NO**	
	RS	**RS-L**	**RS**	**RS-L**	**RS**	**RS-L**
$c10\_s100$	0.98	0.93	0.99	0.98	0.67	0
$c10\_s200$	0.99	0.99	1	0.99	0.90	1
$c10\_s300$	0.99	0.96	0.99	0.96	0.82	0.81
$c10\_s400$	0.99	0.97	0.99	0.98	0.88	0.85
$c10\_s500$	0.99	0.94	0.99	0.93	0.94	0.96
$c20\_s500$	0.99	0.93	0.99	0.94	0.91	0.91
Average	**0.99**	**0.95**	**0.99**	**0.96**	**0.85**	**0.75**

One can observe that the classification accuracy of the testing model I is higher, because the testing the training sets are chosen from the same data set.

Table 2. Classification accuracy for the second testing model

Testing model II	**Total accuracy**		**Accuracy of YES**		**Accuracy of NO**	
	RS	**RS-L**	**RS**	**RS-L**	**RS**	**RS-L**
$c10\_s100N$	0.94	0.97	1	1	0	0
$c10\_s200N$	0.99	0.96	1	0.98	0.75	0.60
$c10\_s300N$	0.99	0.98	1	0.98	0	0.78
$c10\_s400N$	0.96	0.77	0.96	0.77	0.57	0.64
$c10\_s500N$	0.96	0.89	0.99	0.90	0.30	0.80
$c20\_s500N$	0.99	0.89	0.99	0.88	0.44	0.93
Average	**0.97**	**0.91**	**0.99**	**0.92**	**0.34**	**0.63**

Although accuracy of the "YES" class is better than the "NO" class but accuracy of the "NO" class is quite satisfactory. In those experiments, the standard classifier shows a little bit better performance than hierarchical classifier. One can observe that when training sets reach a sufficient size (over 2500 objects) the accuracy on the class "NO" of both classifiers are comparable. To verify if classifier approximations are of high precision and generality, we use the second testing model, where training tables and testing tables are chosen from the new generated simulation data sets. One can observe that accuracy of the "NO" class strongly decreased. In this case the hierarchical classifier shows much better performance. In Table 2 we present the accuracy of the standard classifier and the hierarchical classifier using the second testing model.

Covering rate: Generality of classifiers usually is evaluated by the recognition ability of unseen objects. In this section we analyze covering rate of classifiers for new objects. In Table 3 we present coverage degrees using the first testing model. One can observe that the coverage degrees of standard and hierarchical classifiers are comparable in this case.

Table 3. Covering rate for the first testing model

Testing model I	**Total accuracy**		**Accuracy of YES**		**Accuracy of NO**	
	RS	**RS-L**	**RS**	**RS-L**	**RS**	**RS-L**
$c10\_s100$	0.97	0.96	0.98	0.96	0.85	1
$c10\_s200$	0.95	0.95	0.96	0.96	0.67	0.80
$c10\_s300$	0.94	0.93	0.97	0.95	0.59	0.55
$c10\_s400$	0.96	0.94	0.96	0.94	0.91	0.87
$c10\_s500$	0.96	0.95	0.97	0.96	0.84	0.86
$c20\_s500$	0.93	0.97	0.94	0.98	0.79	0.92
Average	**0.95**	**0.95**	**0.96**	**0.96**	**0.77**	**0.83**

We also examined the coverage degrees using the second testing model. We obtained the similar scenarios to the accuracy degree. The coverage rate for the

Table 4. Covering rate for the second testing model

Testing model II	**Total accuracy**		**Accuracy of YES**		**Accuracy of NO**	
	RS	**RS-L**	**RS**	**RS-L**	**RS**	**RS-L**
$c10\_s100N$	0.44	0.72	0.44	0.74	0.50	0.38
$c10\_s200N$	0.72	0.73	0.73	0.74	0.50	0.63
$c10\_s300N$	0.47	0.68	0.49	0.69	0.10	0.44
$c10\_s400N$	0.74	0.90	0.76	0.93	0.23	0.35
$c10\_s500N$	0.72	0.86	0.74	0.88	0.40	0.69
$c20\_s500N$	0.62	0.89	0.65	0.89	0.17	0.86
Average	**0.62**	**0.79**	**0.64**	**0.81**	**0.32**	**0.55**

both decision classes strongly decreases. Again the hierarchical classifier shows to be more stable than the standard classifier. The results are presented in Table 4.

Computing speed: A time computation necessary for concept approximation synthesis is one of the important features of learning algorithms. Quality of learning approach should be assessed not only by quality of the classifier. In many real-life situations it is necessary not only to make precise decisions but also to learn classifiers in a short time.

Table 5. Time for standard and hierarchical classifier generation

Table names	**RS**	**RS-L**	**Speed up ratio**
$c10\_s100$	94 s	2.3 s	40
$c10\_s200$	714 s	6.7 s	106
$c10\_s300$	1450 s	10.6 s	136
$c10\_s400$	2103 s	34.4 s	60
$c10\_s500$	3586 s	38.9 s	92
$c20\_s500$	10209 s	98s	104
Average			**90**

The layered learning approach shows a tremendous advantage in comparison with the standard learning approach with respect to computation time. In the case of standard classifier, computational time is measured as a time required for computing a rule set used for decision class approximation. In the case of hierarchical classifier computational time is equal to the total time required for all sub-concepts and target concept approximation. The experiments were performed on computer with processor AMD Athlon 1.4GHz. One can see in Table 5 that the speed up ratio of the layered learning approach over the standard one reaches from 40 to 130 times.

Description size: Now, we consider the complexity of concept description. We approximate concepts using decision rule sets. The size of a rule set is character-

ized by rule lengths and its cardinality. In Table 6 we present rule lengths and the number of decision rules generated by the standard learning approach. One can observe that rules generated by the standard approach are quite long. They contain above 40 descriptors (on average).

Table 6. Rule set size for the standard learning approach

Tables	**Rule length**	**# Rule set**
$c10\_s100$	34.1	12
$c10\_s200$	39.1	45
$c10\_s300$	44.7	94
$c10\_s400$	42.9	85
$c10\_s500$	47.6	132
$c20\_s500$	60.9	426
Average	**44.9**	

The size of rule sets generated by layered learning approach are presented in Tables 7, 8 and 9. One can notice that rules approximating sub-concepts are short. The average rule length is from 4 to 6.5 for the basic concepts and from 2 to 3.7 for the super-concepts. Therefore rules generated by layered learning approach are more understandable and easier to interpret than rules induced by the standard learning approach.

Table 7. Description length: C_1, C_2, C_3 for the hierarchical learning approach

	Concept C_1		**Concept** C_2		**Concept** C_3	
Tables	**Ave. rule l.**	**# Rules**	**Ave. rule l.**	**# Rules**	**Ave. rule l.**	**# Rules**
$c10\_s100$	5.0	10	5.3	22	4.5	22
$c10\_s200$	5.1	16	4.5	27	4.6	41
$c10\_s300$	5.2	18	6.6	61	4.1	78
$c10\_s400$	7.3	47	7.2	131	4.9	71
$c10\_s500$	5.6	21	7.5	101	4.7	87
$c20\_s500$	6.5	255	7.7	1107	5.8	249
Average	**5.8**		**6.5**		**4.8**	

Two concepts C_2 and C_4 are more complex than the others. The rule set induced for C_2 takes 28% and the rule set induced for C_4 takes above 27% of the number of rules generated for all seven concepts in the traffic road problem.

6 Conclusion

We presented a method for concept synthesis based on the layered learning approach. Unlike the traditional learning approach, in the layered learning approach the concept approximations are induced not only from accessed data sets

Table 8. Description length: C_4, C_5 for the hierarchical learning approach

	Concept C_4		**Concept** C_5	
Tables	**Rule length**	**# Rule set**	**Rule length**	**# Rule set**
$c10\_s100$	4.5	22	1.0	2
$c10\_s200$	4.6	42	4.7	14
$c10\_s300$	5.2	90	3.4	9
$c10\_s400$	6.0	98	4.7	16
$c10\_s500$	5.8	146	4.9	15
$c20\_s500$	5.4	554	5.3	25
Average	**5.2**		**4.0**	

Table 9. Description length: C_6, C_7, hierarchical learning approach

	Concept C_6		**Concept** C_7	
Tables	**Rule length**	**# Rule set**	**Rule length**	**# Rule set**
$c10\_s100$	2.2	6	3.5	8
$c10\_s200$	1.3	3	3.7	13
$c10\_s300$	2.4	7	3.6	18
$c10\_s400$	2.5	11	3.7	27
$c10\_s500$	2.6	8	3.7	30
$c20\_s500$	2.9	16	3.8	35
Average	**2.3**		**3.7**	

but also from expert's domain knowledge. In the paper, we assume that knowledge is represented by a concept dependency hierarchy. The layered learning approach proved to be promising for complex concept synthesis. Experimental results with road traffic simulation are showing advantages of this new approach in comparison to the standard learning approach. The main advantages of the layered learning approach can be summarized as follows:

1. High precision of concept approximation.
2. High generality of concept approximation.
3. Simplicity of concept description.
4. High computational speed.
5. Possibility of localization of sub-concepts that are difficult to approximate. It is important information, because is specifying a task on which we should concentrate to improve the quality of the target concept approximation.

In future we plan to investigate more advanced approaches for concept composition. One interesting possibility is to use patterns defined by rough approximations of concepts defined by different kinds of classifiers in synthesis of compound concepts. We also would like to develop methods for rough-fuzzy classifier's synthesis (see Section 4.1). In particular, the method mentioned in Section 4.1 based on rough-fuzzy classifiers introduces more flexibility for such composing because a richer class of patterns introduced by different layers of rough-fuzzy classifiers can lead to improving of the classifier quality [18]. On the

other hand, such a process is more complex and efficient heuristics for synthesis of rough-fuzzy classifiers should be developed.

We also plan to apply the layered learning approach to real-life problems especially when domain knowledge is specified in natural language. This can make further links with the computing with words paradigm [27, 28, 12]. This is in particular linked with the rough mereological approach (see, e.g., [15, 17]) and with the rough set approach for approximate reasoning in distributed environments [20, 21], in particular with methods of information system composition [20, 2].

Acknowledgements: The research has been partially supported by the grant 3T11C00226 from Ministry of Scientific Research and Information Technology of the Republic of Poland.

References

1. Aha, D.W.. The omnipresence of case-based reasoning in science and application. Knowledge-Based Systems, 11 (5-6) (1998) 261-273.
2. Barwise, J., Seligman, J., eds.: Information Flow: The Logic of Distributed Systems. Volume 44 of Tracts in Theoretical Computer Scienc. Cambridge University Press, Cambridge, UK (1997)
3. Bazan, J.G.: A comparison of dynamic and non-dynamic rough set methods for extracting laws from decision tables. In Polkowski, L., Skowron, A., eds.: Rough Sets in Knowledge Discovery 1: Methodology and Applications. Physica-Verlag, Heidelberg, Germany (1998) 321–365
4. Bazan, J.G., Szczuka, M.: RSES and RSESlib - a collection of tools for rough set computations. In Ziarko, W., Yao, Y., eds.: Second International Conference on Rough Sets and Current Trends in Computing RSCTC. LNAI **2005**. Banff, Canada, Springer-Verlag (2000) 106–113
5. Bazan, J., Nguyen, H.S., Skowron, A., Szczuka, M.: A view on rough set concept approximation. In Wang, G., Liu, Q., Yao, Y., Skowron, A., eds.: Proceedings of the Ninth International Conference on Rough Sets, Fuzzy Sets, Data Mining and Granular Computing (RSFDGrC'2003),Chongqing, China. LNAI **2639**. Heidelberg, Germany, Springer-Verlag (2003) 181–188
6. Cover, T.M. and Hart, P.E.: Nearest neighbor pattern classification. IEEE Transactions on Information Theory, 13 (1967) 21-27.
7. Friedman, J., Hastie, T., Tibshirani, R.: The Elements of Statistical Learning: Data Mining, Inference, and Prediction. Springer-Verlag, Heidelberg, Germany (2001)
8. Grzymała-Busse, J.: A new version of the rule induction system lers. Fundamenta Informaticae **31(1)** (1997) 27–39
9. Komorowski, J., Pawlak, Z., Polkowski, L., Skowron, A.: Rough sets: a tutorial. In Pal, S.K., Skowron, A., eds.: Rough Fuzzy Hybridization: A New Trend in Decision-Making. Springer-Verlag, Singapore (1999) 3–98
10. Kloesgen, W., Żytkow, J., eds.: Handbook of Knowledge Discovery and Data Mining. Oxford University Press, Oxford (2002)
11. Mitchell, T.: Machine Learning. Mc Graw Hill (1998)

12. Pal, S.K., Polkowski, L., Skowron, A., eds.: Rough-Neural Computing: Techniques for Computing with Words. Cognitive Technologies. Springer-Verlag, Heidelberg, Germany (2003)
13. Pawlak, Z.: Rough Sets: Theoretical Aspects of Reasoning about Data. Volume 9 of System Theory, Knowledge Engineering and Problem Solving. Kluwer Academic Publishers, Dordrecht, The Netherlands (1991)
14. Poggio, T., Smale, S.: The mathematics of learning: Dealing with data. Notices of the AMS **50** (2003) 537–544
15. Polkowski, L., Skowron, A.: Rough mereology: A new paradigm for approximate reasoning. International Journal of Approximate Reasoning **15** (1996) 333–365
16. Polkowski, L., Skowron, A.: Rough mereological calculi of granules: A rough set approach to computation. Computational Intelligence **17** (2001) 472–492
17. Polkowski, L., Skowron, A.: Towards adaptive calculus of granules. In Zadeh, L.A., Kacprzyk, J., eds.: Computing with Words in Information/Intelligent Systems, Heidelberg, Germany, Physica-Verlag (1999) 201–227
18. Skowron, A., Stepaniuk, J.: Information granules and rough-neural computing. [12] 43–84
19. Skowron, A., Stepaniuk, J.: Information granules: Towards foundations of granular computing. International Journal of Intelligent Systems **16** (2001) 57–86
20. Skowron, A., Stepaniuk, J.: Information granule decomposition. Fundamenta Informaticae **47(3-4)** (2001) 337–350
21. Skowron, A.: Approximate reasoning by agents in distributed environments. In Zhong, N., Liu, J., Ohsuga, S., Bradshaw, J., eds.: Intelligent Agent Technology Research and Development: Proceedings of the 2nd Asia-Pacific Conference on Intelligent Agent Technology IAT01, Maebashi, Japan, October 23-26. World Scientific, Singapore (2001) 28–39
22. Skowron, A.: Approximation spaces in rough neurocomputing. In Inuiguchi, M., Tsumoto, S., Hirano, S., eds.: Rough Set Theory and Granular Computing. Volume 125 of Studies in Fuzziness and Soft Computing. Springer-Verlag, Heidelberg, Germany (2003) 13–22
23. Skowron, A., Rauszer, C.: The discernibility matrices and functions in information systems. In Słowiński, R., ed.: Intelligent Decision Support - Handbook of Applications and Advances of the Rough Sets Theory. Volume 11 of D: System Theory, Knowledge Engineering and Problem Solving. Kluwer Academic Publishers, Dordrecht, Netherlands (1992) 331–362
24. Skowron, A., Szczuka, M.: Approximate reasoning schemes: Classifiers for computing with words. In: Proceedings of SMPS 2002. Advances in Soft Computing, Heidelberg, Canada, Springer-Verlag (2002) 338–345
25. Stone, P.: Layered Learning in Multi-Agent Systems: A Winning Approach to Robotic Soccer. The MIT Press, Cambridge, MA (2000)
26. Wróblewski, J.: Covering with reducts - a fast algorithm for rule generation. In Polkowski, L., Skowron, A., eds.: Proceedings of the First International Conference on Rough Sets and Current Trends in Computing (RSCTC'98), Warsaw, Poland. LNAI **1424**, Heidelberg, Germany, Springer-Verlag (1998) 402–407
27. Zadeh, L.A.: Fuzzy logic = computing with words. IEEE Transactions on Fuzzy Systems **4** (1996) 103–111
28. Zadeh, L.A.: A new direction in AI: Toward a computational theory of perceptions. AI Magazine **22** (2001) 73–84