1. Introduction

Modern monitoring and control systems are equipped, as a rule, with object parameter analyzing blocks which facilitate drawing conclusions on the emerging situations and carrying out control on this basis. It is advisable to develop knowledge-based systems for comprehensive object analysis which allows the decision maker (DM) to change the analysis rules [1]. Such systems based on facts and rules make it possible to describe states of the controlled objects and the conditions under which they arise [2].

For interactive work with control rules, special structuring of rules and such mathematical model as the AND/OR graph are used [3]. Due to the better visualization of rule presentation, this approach enables the expert to form knowledge for control systems at early stages of construction of the knowledge field.

When forming control rules, unforeseen errors may occur, so it is important to be able to find them in an automated mode. Therefore, development and improvement of methods for verification of control rules and elaboration of a procedure for their use both in searching for and elimination of errors is a focal problem.

2. Literature review and problem statement

Theory is inconsistent if there is such an assertion that both ensues from the theory and is negated by it: $T \rightarrow \phi$ and $T \rightarrow \Phi$ where $T$ is theory and $\phi$ is assertion.

In [4], inconsistencies in the knowledge-based system are divided into external (the inconsistencies between the production system and the world model) and internal inconsistencies in the production system. The latter mean that (1)
there are rules that contain inconsistencies inside themselves and (2) there are two or more rules that are consistent per se but one rule is inconsistent with another or several rules.

Thus, in a system with inconsistent knowledge, both assertion and its negation may be drawn from the same premises. As a result, the system can make wrong inferences or do not come to any decision at all and the final recommendations of the system will depend on the strategy of rule choice (on the method of resolving the collision).

A theory \( T \) is complete if one can deduce from any assertion \( \phi \) that it is either correct or incorrect: \( T \vdash \phi \) or \( T \vdash \neg \phi \).

Completeness (incompleteness) for a knowledge-based system means that knowledge is adequate (inadequate) for solving problems using this system.

Incompleteness is defined in [4] as such a defect that is more substantive than formal in nature and is expressed in inability of the system to make inferences for a number of certain initial situations. The criterion of completeness determines how much the set of rules allows the system to involve all possible combinations of initial data.

Formally, incompleteness manifests itself in different ways. Firstly, facts can be omitted, for example, because of “obviousness” of some knowledge expert cannot represent it in an explicit form. Secondly, when rules are missing, deadlocks in the logical inference chains may occur. Certain requirements are imposed to the correctly formed set of rules. In particular, every rule conclusion attribute must either be a target attribute or be in the conditions of other rules. Each attribute of the rule condition must be either terminal or be the conclusion of another rule. Terminal attributes are the attributes with their values either entered by the user or requested in the database, or read from a measuring device, or calculated by a procedure. Also, the rule should not contain unreachable goals, that is, such goals that none of the possible logical inference chains leads to a given value of the target attribute. In [5], for a case of inductive rule construction, incompleteness implies absence of the representativeness property in the training set.

A method for searching for inconsistencies in the rules at the stage of testing the knowledgebase (KB) using an inference machine was proposed earlier [6]. However, rule verification and search for anomalies are most effective at early stages when description of the basic concepts and relationships of the subject area is made. This kind of approach applied with no use of the inference machine is called static verification.

However, most of the existing methods of static rule verification are not used widely. This relates to the fact that, firstly, before verification, the developer should in fact describe all possible errors in a form of setting constraints that may arise in the rules of this domain or specifically this set of rules. It is a laborious, often impracticable task. Secondly, after detecting errors using the developed methods, it is necessary to involve an expert for additional analysis of the detected anomalies [7].

For different methods, constraints are set differently. Methods based on label generation suggest construction of an analytical representation of all possible inference chains from the rules base with formation of a set of environments (labels) leading to the inference of relevant goals [8]. The methods based on conceptual graphs [9], decision tables [10], hypergraphs [11] allow the developer to represent the rules more obviously than in a text form which helps him to visually detect anomalies. However, for further analysis, the meta-knowledge set by the developer prior to verification is also required.

Improved methods of anomaly search have been developed for the new knowledge representation model in a form of a logical network [12] but they are not suitable for rule verification.

As can be seen, most of the existing methods of static verification for general rules require additional efforts from the user for constructing problem-oriented constraints, manual analysis of the detected anomalies associated with the KB inconsistencies and incompleteness, that is, these methods are not performed automatically. Thus, it is necessary for a narrower class, namely control rules, to offer new methods of static verification that are specific for them and enable, on the one hand, to verify without or with minimal preliminary preparation of meta-knowledge on the rules and, on the other hand, automatically detect errors without or with minimal necessity of their analysis by the developer. What is especially important, assistance is necessary in correcting each error type.

3. The aim and objectives of the study

The study objective was to reduce errors occurring in the creation of control rules by developing methods for their verification and correction.

To achieve this goal, the following tasks were set:
- to improve methods for checking reachability in the rules of the controlled object states, verification for inconsistency and completeness of the rule sets;
- to determine components and stages of functioning of the knowledge-based control systems with built-in verification and correction of rules;
- to develop a rule editor and valid prototypes of the knowledge-based control systems for two domains and investigate their effectiveness when used in learning management systems.

4. Development of methods for verification and correction of control rules

4.1. Models of control rules in a form of AND/OR graph and Boolean expressions

A model of control rules was proposed in a form of the AND/OR graph for interactive work, that is input and editing of knowledge: the rules are grouped according to the state of the object under control, and an individual AND/OR graph is constructed for each group [13]. The vertices of the graph corresponding to the rule premises set values of the control parameters, and one vertex in each graph corresponds to the state of the object being controlled. Edges are marked with rule numbers which makes it possible to convert the graph paths that correspond to the rules into logical expressions. If there is a large number of rules in the group, it is divided into semantically weakly related subgroups. A general form of the AND/OR graph for a subgroup of control rules is shown in Fig. 1.

Two models of control rules are proposed.
The first model of control rules in a form of an AND/OR graph is an oriented graph without cycles, with all its vertices divided into three disjoint sets: AND vertices, OR vertices and terminal (or target) vertices. This kind of model...
Control processes

is used in visualizing the rules and interactive work with them. Unlike the existing graphical model for general rules, the proposed model for control rules is based on dividing the rules into groups according to the states of the controlled object. The graph contains special markup: in particular, state vertices are marked, and edges are marked with rule numbers [13]. The latter allows to convert the graph paths that correspond to the rules into logical expressions used further in verification of the rules.

In Fig. 1 the premise vertices of each rule are linked with the conclusion vertex ($c_{ij}$ is the $j$-th premise of the $i$-th rule).

The AND/OR graph is suggested to be converted to Boolean expressions so that the path in the graph represents a direct set of rules [14]:

$$p_i \rightarrow \overset{\sim}{w}, \quad p_i \rightarrow \overset{\sim}{w}, \ldots, \quad p_i \rightarrow \overset{\sim}{w}, \quad (1)$$

where $p_i$ is conjunction of the premises of the rules $i$, $i=1,...,k$ is the number of rules;

$$\quad ci,j \quad \begin{array}{c} 1 \quad \ldots \quad \ldots \quad 1 \end{array} \quad \begin{array}{c} c_{1i} \quad \ldots \quad \ldots \quad c_{ni} \end{array}$$

$\quad $i, w$ is the object state out of range; $w$ is normal state of the object.

All premises of the rules are linked by the AND operation. If the rule has one or more OR operations, they are divided into several rules containing only AND vertices. Such transformations can always be made since DNF can be obtained from any Boolean formula. Also, if there are non-terminal rules in a subgroup, then premises of these rules can be included in the terminal rules. For this minimization of the Boolean formulas is used, and MDNF is obtained.

Thus, all rules of a subgroup have the same conclusions in a case of their correct construction and the premises of all rules are connected by the AND operation.

Further, the following formulas were obtained from (1):

- a general formula for a direct rule set

$$p_1 \lor p_2 \lor \ldots \lor p_k \rightarrow \overset{\sim}{w}, \quad (2)$$

- a general formula for an “inverse” rule set

$$p_1 \lor p_2 \lor \ldots \lor p_k \leftrightarrow \overset{\sim}{w}, \quad (3)$$

- an “inverse” rule set

$$p'_1 \rightarrow w, \quad p'_2 \rightarrow w, \quad \ldots, \quad p'_k \rightarrow w. \quad (4)$$

Note that it is impossible to obtain the following model in a form of rules (2)-(4) for any rules with different conclusions. It is only applicable to control rules, that is the rules of special kind when the conclusions of all group rules are the same. It is on the basis of these formulas that the proposed effective methods for rule verification can be constructed.

4. 2. Methods for verification of control rules

Three methods for verification of control rules based on the models presented in 4. 1 are proposed.

4. 2. 1. The method for verification of state reachability for the controlled object

The proposed improved method of verifying reachability of the object’s state vertex from the rules has a set of rules at the input and includes two steps:

1. Obtaining of formula (1) from the AND/OR graph for a direct rule set:

$$p_1 \rightarrow \overset{\sim}{w}, \quad p_2 \rightarrow \overset{\sim}{w}, \ldots, \quad p_k \rightarrow \overset{\sim}{w}. \quad (5)$$

2. If the right part of the implication in any Boolean expression does not contain a variable corresponding to the vertex that is responsible for the state of the controlled object, it means that there is a rule in which the vertex of the object state is unreachable.

It is expedient to detect the rules, that is the connected component from which the vertex of the controlled object state is unreachable and either change them or delete since they are not used for the object control.

At the output of the method, either a verdict is derived that a state of the controlled object is reachable from all chains of the set rules or the chains (connected components) with unreachability of the state vertex are determined.

Fig. 2 shows an example of the rules in a form of AND/OR graph for computer network control. One can see
that the terminal state vertex “problem” is unreachable from
the graph connected component containing rules 5 and 4.
These rules must be deleted or one more premise linking
rule 4 with the state vertex have to be added to rule 1.

4.2.2. The method for verifying control rules for inco-
 consistency
The proposed method for verifying premises of the con-
 trol rules for inconsistency has a set of rules at the input and
 includes three stages [14].
1. Verification is completed at the first stage if there are
 inconsistencies within each rule. For this type of verifica-
 tion, the left side of the implication of the Boolean for-
 mula (2) is sent as input to the SAT (SATisfiability problem).
2. Verification of the second stage is completed if there
 are inconsistencies in at least one rule. To do this, this the
 left part of implication of each rule (1) is sent to the input of
 the SAT problem.
3. Inconsistencies between the premises of two or more
 rules can be found at the third stage. To do this, the left part
 of the implication of formula (3) is sent to the input of the
 SAT problem.
At the output of the method, either a verdict is output
 that no inconsistency between the rule premises was found
 or the rules in which inconsistencies were found are shown.
Let us consider the examples illustrating detection of
 inconsistencies in the rule premises.

Example 1
If \( U \geq 1000 \text{ V} \) and \( U < 1000 \text{ V} \), then the works cannot be performed

There is an inconsistency between the premises of one
rule. Such a rule will never be fulfilled. The method will be
finished at the second stage.

Example 2
If \( U \geq 1000 \text{ V} \), then the works cannot be performed
If \( U < 1000 \text{ V} \), then the works cannot be performed

There is an inconsistency between the premises of dif-
f erent rules. Such a system of rules does not make sense
because in any case, a verdict will be output that it is im-
 possible to perform works. The method will be finished at
the third stage.

The main advantages of the proposed method are that
verification for inconsistency of the rule premises is per-
formed automatically and with taking into account specifics
of the control task. Such verification cannot be performed
for general rules since formulas (2)–(4) cannot be obtained
for them.

The method for verifying the rule premises for incon-
sistency allows one to find both the inconsistencies that
were introduced during construction of the rules and those
present in the original texts, in particular, in the normative
documents.

4.2.3. The method for verifying control rules for com-
 pleteness
The proposed method for verifying the rules for com-
 pleteness has a set of rules at the input and includes three
steps.
1. The user (expert) gets a graphical representation of
the generated AND/OR graph of the direct rule set (1), that
is, the conditions under which the controlled object is in the
emergency state.
2. The user gets a graphical representation of the
AND/OR graph of the “inverse” set of rules (4), that is, the
conditions under which the state of the controlled object
is normal. This kind of graph is obtained automatically from
a direct set of rules (1).
3. When analyzing the so-called “inverse” AND/OR
graph, the user can see which rules are missing or in which
rules conditions are specified incompletely.

At the output of the method, a verdict is given that the
set of control rules is complete if the user has viewed both
the direct and “inverse” rule sets and did not see missing
rules. Either otherwise, a verdict is given that the set of rules
is incomplete. In the latter case, it is potentially possible to
assist the user in construction of the missing rules as shown
later in 4.3.

Let us consider an example illustrating detection of in-
completeness of the rules base. For simplicity of the further
presentation, let the entire set of rules consist of one rule:

If \( U \geq 1000 \) V, and there is no supervisor, and the room is
dangerous, then the works cannot be performed

Introduce notations: \( a \) – \( U \geq 1000 \text{ V} \); \( b \) – there is no su-
pervisor; \( c \) – the room is dangerous, \( w \) – the works can be
performed. Representation and converting of the Boolean
formulas are given below:

\[
\begin{align*}
\quad a \land b \land c & \leftrightarrow \bar{a}, \\
\quad a \land b \land c & \leftrightarrow \bar{a} \lor \bar{b} \land \bar{c} \leftrightarrow \bar{w}, \\
\quad \bar{a} \leftrightarrow \bar{w}, & \quad \bar{b} \leftrightarrow \bar{w}, \quad \bar{c} \leftrightarrow \bar{w}.
\end{align*}
\]

Interpretation of one of the inverse rules: if \( U < 1000 \text{ V} \),
then the works can be performed. An incompleteness of the
rule set is found: it is obvious that there are no rules in the rules base for the case when \( U < \) 1000 V, so they shall be added to the rules base.

The proposed method works in the same way as the "proof by contradiction" method which is carried out as follows: to prove the assertion \( A \), it is assumed that it is incorrect and then proved that a certain incorrect assertion \( B \) follows from \( \neg A \). In the proposed method, \( A \) is a "direct" set of rules, \( \neg A \) is the "inverse" set, \( B \) is the "inverse" rule which is incorrect from the expert’s point of view.

Such completeness verifications can only be performed for each group of control rules separately and are not applicable to general rules because formulas (2)–(4) cannot be obtained for them.

4. 2. 4. Analysis of complexity of the algorithms used in the methods for rule verification

Although some of the problems considered are NP-complete, in particular, general SAT problems, this is not a limitation for application in analyzing quality of the control rules for several reasons:

- there are effective algorithms of solving the SAT problem enabling achievement of acceptable efficiency for real problems (the problems such as planning, scheduling, decision synthesis were considered);
- in the case of high-dimensional data, it is natural to solve complicated problems with the help of distributed systems of a great processing power;
- there are approximate algorithms that solve the SAT problems in a polynomial time;
- such verifications are performed not at the stage of object control when the end user is working with the system and where high efficiency is needed but at the stages of creation and correction of the rules or in training;
- the number of rules in each group for which verification is performed is usually small and ranges from 6 to 20 rules.

4. 3. The procedure for verification and correction of control rules

The next step after verification of the rules using the methods described in 4.2 is to eliminate the errors found.

A procedure for verification and correction of control rules was proposed. It consists of the following stages:

1. Formation of the AND/OR graph for the direct set of rules in the conclusions of which there is the emergency state of the controlled object.
2. Verification of the graph for reachability of the vertex of the controlled object state. If an inference chain was found that does not lead to the state vertex, then the user is given a choice:
   - either delete the chain of rules;
   - or replace the last rule conclusion of the incorrect chain with the object state;
   - or append the chain to another, correct one.
After corrections, transition to paragraph 1 is made.
3. Verification of the direct set of rules for inconsistency. If an inconsistency was found inside the rules or between the premises of several rules, then the user is given the rules for corrections. After corrections, transition to paragraph 1 is made.
4. Automatic conversion of the direct set of rules to an "inverse" set in conclusions of which normal state of the controlled object is found.
5. Presentation of the "inverse" set of rules to the expert as an AND/OR graph.
6. Is the set of rules complete from the expert’s point of view? If yes, go to the end, that is, to paragraph 9.
7. Automated construction of new direct and "inverse" rules by the expert or correction of existing ones. The steps for supplementing the rules base:
   - the expert selects each problematic "inverse" rule;
   - the system shows all possible ways of supplementing to completeness;
   - the expert forms both "direct" and "inverse" rules by choosing the desired ones from the presented premises and noting the type of state (emergency or normal).
8. Formation of a new set of direct rules by conversion from "inverse" and new direct rules. Following the corrections, transition to paragraph 1 is made.
9. The end.
Such procedure of rule construction is performed iteratively until the expert is satisfied with the constructed set of rules.

The proposed procedure for verification and correction of control rules helps to bring together all sorts of verifications, place them in a correct order and correct errors in an automated mode.

5. Knowledge-based control systems with built-in verification and correction of rules

The proposed components of the knowledge-based control systems, their functions, as well as input and output data are presented in Table 1.

| Control processes | 47 |
The developed systems were used as part of the learning management systems [16] as simulators for the dispatchers dealing with safe operation with electric installations and the system administrators of computer networks. Materials for simulators were developed by experts with the help of the rule editor.

During learning, the trainee observes and analyzes how the dispatcher should act. Using the example of a system simulating the work of an experienced expert, he/she tries different situations and makes his own inferences on this basis.

The learning procedure includes the following steps:

1. Acquaintance with the theory and normative documents.
2. Starting the system in autonomous mode at various input parameters. As a result, states of the controlled object are automatically determined.
3. The trainee makes his own inferences concerning the parameters at which the object enters the emergency mode and at which it works normally and forms on his/her own the rules for one or more groups using the AND/OR graph and in a text format with the help of the rule editor.
4. The rule editor verifies the constructed rules for inconsistency, completeness, reachability of states.
5. The created graphs are compared with the correct ones and the learning results are evaluated on this basis. Next, the correct graph is presented to the trainee for comparison.

### 6. Discussion of the results obtained in the studies aimed at development of knowledge-based control systems

#### 6.1. Control systems and rule editor for teaching decision makers

Control systems are designed for dispatchers or administrators, in general, for decision makers using the control results. The rule editor is used by experts when constructing control rules of the knowledge-based systems. However, their application was extended to the field of training: on the one hand, for training specialists in a specific domain, that is DM, and on the other hand, for developers learning the methods for creating knowledge-oriented systems.
The consultation subsystem (Fig. 4) allows the trainee to start consultations on the possible input parameters. It displays results on the object states corresponding to the parameters entered.

Based on the teaching of network administrators and electric safety dispatchers, experiments were conducted to evaluate work of the simulator. The results related to verification of the rules in teaching the decision makers are shown in Table 2. Of the 244 rules that were created, errors were found in 11.5 % of the rules when testing for inconsistency, in 4.95 % of the rules when testing for reachability of states. In completeness tests, the rules base was increased by an average of 13.05 %.

6.2. The control system and the rule editor in teaching students knowledge-based systems development

The rule editor was used in teaching the students of the Institute of Computer Systems at Odessa National Polytechnic University, Ukraine, during writing their term papers in intellectual data analysis. The tasks consisted in development of demonstrational prototypes of the control systems for various domains, in particular, subgroups for monitoring work of a computer network and safe operation of electric installations. Since students are not experts in the proposed fields but study the information technologies, they were offered description of basic information on the subject areas necessary for work. The purpose of the term paper was to acquire skills in construction of knowledge-based systems. The number of students who completed the term papers on time increased on average by 8 % when using the rule editor, as shown in Fig. 5 where the X-axis is the number of weeks of the semester and the Y-axis is percentage of students who passed their papers. Thus, due to better visualization of rules presentation in a form of AND/OR graph as well as effective assistance given to the expert during verification and correction at the early stages of rules construction, the rules quality has improved. The KB contained fewer errors and inaccuracies which lead to an improvement in quality of the decisions made when using the system and ultimately leads to a reduction in the number of emergency situations.

Such results were achieved due to:
- the application of knowledge in a form of rules for assessing the object states;
- the proposed rule models for the groups describing one state each and not the general rules but for the control tasks;
- the development of methods and the procedure for verification and correction of control rules based on the models of control rules.

The earlier approaches to static rule verification were developed mainly for general rules containing premises, linked by conjunction and disjunction operations, and conclusions. Therefore, in order to conduct verification, the developer had to make a preliminary description of his/her class of rules and domain. In this work, models and verification methods were developed for a narrower class of rules, the control rules. This approach has allowed to reduce preparation time, verify rules automatically in most cases, find and correct the errors that could not be found using general approaches.

For the proposed knowledge-based control systems, rules are created entirely by experts in an automated mode using the rule editor based on their knowledge and experience. Nevertheless, such an approach imposes restrictions connected mainly with laboriousness of performing such actions and subjectivity of expert knowledge. At present, machine learning is used in many fields to obtain effective models and an automatic derivation of patterns from examples. However,
such models are sometimes a “black box”. Therefore, the process of solving problems becomes uncontrollable. That is why it is important to obtain knowledge from examples in such a way that it can be used not only for solving problems but also to visualize, correct, interpret and control them.

Thus, further studies should be aimed at improvement of the proposed control systems. It is advisable not only add abilities of deriving control rules from examples but also integrate both approaches to acquire knowledge. This will require conversion of the “raw” patterns derived from the examples into rules available to the experts for their viewing, edition, etc. In other words, it will be necessary to create new models and methods that will support this kind of integration.

7. Conclusions

1. Development of the method for verifying vertices of the controlled object states reachability in the AND/OR graph has created the basis enabling search for and deletion of the connected components that do not contain such vertices. Due to the developed method of verification for inconsistency of the control rules premises based on the SAT problem, it became possible to find inconsistencies between the premises of each rule and between rules. The proposed verification method of the control rules for completeness based on the expert’s visualization of “inverse” rules has made it possible to evaluate what rules are missing.

2. A procedure was proposed that regulates the order of verification and the methods of correction various types of errors in the control rules. When conducting experiments to verify reachability of the controlled object state in the rules, an average of 5.4% of the rules containing errors of this kind were found and corrected. When verified for inconsistency of the rule premises, errors in 11.5% of the rules were found and corrected. When verified for completeness of the KB, it was expanded by an average of 12.3% through adding missing rules.

3. In development of knowledge-based control systems, it was proposed to divide them in two components because of their use at different stages and by different people: a rule editor to be used by knowledge engineers or experts and a control system for work of decision makers, i.e. dispatchers and administrators. Functions, the input and output data for both components, as well as methods of integration with existing control systems were presented.

4. An interactive work with control rules when using the developed editor was provided: creation, edition, visualization and verification. Control systems for two fields were created: control of safe work with electric installations and control of computer networks. As a result of the conducted experiments, it was shown that when using knowledge-based control systems in training, the time spent on task execution was reduced by an average of 8% in comparison with conventional methods, without loss of quality.

Acknowledgment

The work was carried out within the framework of the state budgetary themes of the Ministry of Education and Science of Ukraine, the research work No. 114-73 (State Registration No. 0116U004528): Methods, Models and Means of Software Engineering conducted by the Department of System Software, ONPU.

References

15. NetXMS. Open source network monitoring system. URL: https://www.netxms.org/