UDC 004.415:004.82 **doi: 10.32620/reks.2023.4.05**

Oleksandr KARATAIEV, Ihor SHUBIN

National University of Radioelectronics, Kharkiv, Ukraine

FORMAL MODEL OF MULTI-AGENT ARCHITECTURE OF A SOFTWARE SYSTEM BASED ON KNOWLEDGE INTERPRETATION

*The use of agents across diverse domains within computer science and artificial intelligence is experiencing a notable surge in response to the imperatives of adaptability, efficiency, and scalability. The **subject** of this study is the application of formal methods to furnish a framework for knowledge interpretation with a specific focus on the agent-based paradigm in software engineering. This study aims to advance a formal approach to knowledge interpretation by leveraging the agent-based paradigm. The **objectives** are as follows: 1) to examine the current state of the agent-based paradigm in software engineering; 2) to describe the basic concepts of the knowledge interpretation approach; 3) to study the general structure of the rule extraction task; 4) to develop the reference structure of knowledge interpretation; 5) to develop a multi-agent system architecture; 6) and to discuss the research results. This study employs formal **methods**, including the use of closed path rules and predicate logic. Specifically, the integration of closed path rules contributes to the extraction and explication of facts from extensive knowledge bases. The obtained **results** encompass the following: 1) a rule mining approach grounded in closed path rules and tailored for processing extensive datasets; 2) a formalization of relevance that facilitates the scrutiny and automated exclusion of irrelevant fragments from the explanatory framework; and 3) the realization of a multi-agent system predicated on the synergy among five distinct types of agents, dedicated to rule extraction and the interpretation of acquired knowledge. This paper provides an example of the application of the proposed formal tenets, demonstrating their practical context. The **conclusion** underscores that the agent-based paradigm, with its emphasis on decentralized and autonomous entities, presents an innovative framework for handling the intricacies of knowledge processing. It extends to the retrieval of facts and rules. By distributing functions across multiple agents, the framework offers a dynamic and scalable solution to effectively interpret vast knowledge repositories. This approach is particularly valuable in scenarios where traditional methods may struggle to cope with the volume and complexity of information.*

Keywords: *multi-agent system; software engineering; formal model; knowledge interpretation; closed path rules.*

Introduction

Agent-based systems represent one of the most dynamic and pivotal domains in recent years, garnering substantial attention in information technology development [1, 2]. Constituting intricate information systems, multi-agent systems feature many interacting agents, each endowed with distinct attributes, capabilities, and objectives, fostering collaborative interactions within the system. The versatility of multi-agent systems finds application across diverse domains, including the modeling of market behavior in economics, the simulation of biological and social systems in scientific contexts, and the representation of complex technical systems in engineering [3].

The ubiquity of agent-based systems transcends their initial roots in artificial intelligence, evolving into a foundational computing technology of paramount significance. The advent of information and related technologies aligns seamlessly with the utility of multi-agent systems. Concurrently, advancements in distributed object technologies, exemplified by the CORBA (Common Object Request Broker Architecture) distributed computing platform [4], which facilitates low-level interaction among heterogeneous distributed components, are instrumental contributors to the maturation of the agent-based approach. Notably, these technologies obviate the need for extensive redevelopment of foundational methods.

In the contemporary landscape, the imperatives of software demand flexibility, reliability, and efficiency. In response, an agent-based paradigm is gaining prominence within the field of software engineering. While prevailing efforts in the realm of agents often gravitate toward either practical application development or the intricate construction of reasoning logic, the development of formal models for agent systems assumes paramount importance. Such formal models substantiate the overarching computational objective of realizing tangible agent systems.

The modeling of intelligent functions, which encompass knowledge processing, can be effectively realized through a network of agents engaging with both the external environment and each other to address

ISSN 1814-4225 (print)
ISSN 2663-2012 (online)

intricate intellectual tasks. In this context, knowledge assumes the role of information conveyed by an observer to an intelligent agent, empowering the agent to rationalize its behavior in pursuit of predetermined goals derived from observational learning. The contemporary objective of knowledge bases is to facilitate the sharing and reuse of knowledge. Notably, the creation of expansive knowledge bases containing millions of facts about diverse entities has emerged as a focal point, proving immensely beneficial for intelligent web search, question comprehension, contextual advertising, social media analysis, and advancements in biomedicine.

Consequently, the application of an agent-based paradigm for knowledge processing, spanning the interpretation of information within large knowledge bases, as well as the search for facts and rules, holds promise for optimizing the efficient utilization of available information resources.

This article specifically addresses the development of a multi-agent system grounded in a formal description of agent architecture and underscores the modeling of pertinent explanations derived from knowledge bases. The primary objective of this study is to devise a formal approach to knowledge interpretation, leveraging the agent-based software development paradigm.

The research questions guiding this endeavor are as follows:

1. What properties influence the utility of information garnered from a knowledge base and how can they be employed in the interpretation of knowledge?

2. In what manner can the agent-based software development paradigm be effectively applied to a knowledge-based software system?

The subsequent sections are organized as follows: an examination of the current state of the agent-based paradigm in software engineering, an elucidation of the research methodology and used materials, a depiction of the approach to formalizing agent architecture, and its practical implementation for knowledge base interactions. The article concludes with a discussion of formal knowledge interpretation, the proposal of a multi-agent system architecture, and an illustrative application of the developed formal tenets, followed by conclusive remarks and a discussion of the results.

1. State of the art and problem statement

Until recently, the creation of intelligent systems predominantly followed two principal methods: the algorithmic approach and the structural-functional approach. A more recent paradigm, known as the functional-structural method, has emerged in the domain of intelligent system creation [5]. The algorithmic method entails constructing intelligent systems that emulate human intelligence functions described in linguistic terms, with subsequent algorithmic implementation [2]. Conversely, the structural-functional method involves initially establishing a network structure based on artificial neurons, followed by a learning process to replicate specific intelligence functions [6]. A third method, distinct from the former two, describes intelligence functions as a system of logical equations in predicate algebra. Subsequently, chains of switches are created on the basis of these equations, endowing the method with the advantage of purposeful structure formation to represent a designated intelligence function [5, 7].

Within the realm of artificial intelligence, the conceptualization of agents has emerged as a response to the challenges encountered when attempting to solve problems without accounting for the real external environment or the entity involved in the problem-solving process. Agents, designed to operate flexibly and adaptively in complex real-world environments, receive data from their environment through sensor devices and influence the environment through specialized effectors. This concept has gained swift and widespread adoption across various computer science fields because of its versatility and practical utility [1, 8].

The diversity of agent definitions has proliferated, spanning generic autonomous agents, software agents, intelligent agents, and more specialized categories such as interface agents, virtual agents, information agents, and mobile agents [9, 10]. Agents find application in a myriad of domains, including operating system interfaces, satellite image data processing, power distribution management, air traffic control, business process management, e-commerce, computer games, and beyond, attesting to their broad utility and adaptability [11, 12].

The agent metaphor, with its versatility, manifests both strength and weakness due to the absence of a universally agreed-upon definition. Consequently, many researchers provide their own interpretations, often characterizing agents by specific parameters. For instance, Wooldridge and Jennings [1] established a foundational notion of agents that encompasses autonomy (the ability to function without interference), sociality (interaction with other agents), reactivity (perception and response to a changing environment), and proactivity (purposeful behavior), which are widely acknowledged as pivotal qualities defining "agency."

In the context of software engineering, an agent is defined as an autonomous software entity that demonstrates a prolonged lifespan and adaptive behavior in response to environmental changes, facilitating interaction with other agents [1]. Within artificial intelligence systems, an agent is conceptualized as an entity capable of perceiving information via sensors and influencing

the environment through actuators [2]. The key features of agents include autonomy, interactivity, and learning abilities, enabling independent decision-making for goal attainment, and facilitating interactions akin to social dynamics. The ability to learn is particularly associated with intelligent agents within artificial intelligence systems.

The agent-based approach to software design seamlessly integrates features from both object-oriented and component-based methodologies, enabling the incorporation of specific agent properties [13, 14]. This methodology empowers the design of multi-agent systems in which agents collaborate to achieve individual and collective goals [15, 16].

For the effective design and implementation of agent systems, the establishment of a formal framework for describing and specifying their behavior is crucial. An agent system serves as a platform capable of creating, interpreting, executing, communicating, and terminating agents [3, 10]. The underpinning of software development relies on the use of formal models, a practice that has also been extended to the domain of multi-agent systems [14, 17]. Noteworthy formal approaches encompass equation-based modeling [18], game theory [19], discrete systems [20], classifiers, and other methodologies [21].

In the context of knowledge reuse, the foundational assumption posits that knowledge is replicable and transportable across different contexts [22, 23]. Knowledge, conceptualized as an abstraction beyond tangible representation, cannot be transcribed or physically grasped. Various formal languages, such as predicate calculus, are employed to represent knowledge in information systems. Predicate calculus, notable for its unambiguous formal semantics and operational support through a flawless inference mechanism, serves as a prominent tool for expressing and manipulating knowledge in a systematic manner [5, 7].

A Knowledge Graph (KG) constitutes an extensive collection of binary facts expressed in the form of subject-predicate-object triples [22, 24]. This assembly of facts forms a directed graph in which each fact is an edge, with the predicate serving as the label, connecting a subject entity to an object entity. A Knowledge Graph is a specialized form of a Knowledge Base (KB), in which a KB is a broader category encompassing general repositories of knowledge without imposing specific restrictions on the pattern of facts or the level of knowledge abstraction. A Knowledge Database (KD) has the capacity to incorporate not only general rules but also facts [22, 25].

Given the sheer volume of data, the manual construction of large Knowledge Graphs is impractical. Consequently, a primary challenge in building a Knowledge Graph lies in developing scalable methods for the automated learning of new entities, their properties, and relationships [24]. Some researchers argue that a Knowledge Graph transcends the conventional definition of a database [22, 25]. Specifically, it is asserted that a Knowledge Graph must possess a level of conceptual knowledge, often represented as a set of rules. Rules, constituting explicit knowledge, offer human-readable explanations for the outcomes derived from them. Traditional rule learning methods face limitations in this context, as they lack the scalability to handle vast amounts of data, and Knowledge Graphs do not explicitly articulate negative examples, which are crucial for many data mining tools.

Interpretation is a pivotal factor in ensuring effective interaction between users and software, with explanation playing a crucial role in fostering trust in the outcomes of intelligent systems. Typically, an explanation is perceived as a lucid delineation of the results obtained. However, providing an explanation necessitates a delicate balance between relevance and completeness. Demonstrating the relevance of knowledge poses a challenge, given that the perception of information is inherently subjective. What may be intriguing to one user might be entirely inconsequential to another. Consequently, the interpretation of knowledge can be realized through the explanations offered by intelligent systems to users.

In this context, employing the agent paradigm as the foundation for developing intelligent system software holds promise, particularly in terms of scalability and flexibility in crafting solutions. The imperative of processing large knowledge bases underscores the need for scalability in rule learning methods. Simultaneously, the extraction of rules, exploration of new entities, and understanding of their properties form the bedrock for constructing user-understandable explanations. This collective approach enables the development of intelligent systems grounded in knowledge interpretation.

2. The rule extraction task

Formally, a knowledge graph (KG) contains facts about entities. An entity e can represent a place, person, etc.; a fact is an RDF triple (e, P, e'). In other words, an entity e is related to another entity e' through a binary predicate P. Following the convention of knowledge representation, we denote this fact as $p(e, e')$. Thus, formally, a knowledge graph is a pair $K = (E, F)$where E - is a set of entities, and F - is a set of facts.

Let r denote the rule. Next, we consider rules that have the following form

$$p(x_0, x_n) \leftarrow p_1(x_0, x_1) \wedge \cdots \wedge p_n(x_{n-1}, x_n),$$

where everyone x_i $(0 \leq i \leq n)$ is a variable.

ISSN 1814-4225 (print)
ISSN 2663-2012 (online)

Intuitively, the rule r says that if $p_1(x_0, x_1), \cdots, p_n(x_{n-1}, x_n)$ is true, then $p(x_0, x_n)$ is also true. A rule of this type is usually called a closed-path rule (CPR) because the sequence of predicates in the body of the rule forms a path from the subject argument to the object argument of the main predicate [24].

For a given rule r, a pair of entities (e, e') satisfies the body of the rule r, denoted as $b(e, e')$, if there is a way to replace the variables in b with entities from the knowledge base (KB) such that: (1) all atoms in $b(e, e')$ (after replacement) are facts in the KB, and (2) x_0 i x_n are replaced by e and e', respectively. A pair of entities is said to be (e, e') is said to satisfy the head of the rule r, denoted as $h(e, e')$if $h(e, e')$ is also a fact in the knowledge base [24]. Then the degree of support for rule r is defined as the number of pairs of entities that satisfy both the head and the body of rule r:

$$S(r) = \#(e, e') : b(e, e') \wedge h(e, e').$$

The degree of confidence of the rule r SB(r) and the head coverage of the rule r SH(r) are defined as forms of normalization for the degree of support S(r):

$$SB(r) = \frac{S(r)}{\#(e, e') : b(e, e')}$$

$$SH(r) = \frac{S(r)}{\#(e, e') : h(e, e')}$$

So, SB(r) is a normalization S(r) by the number of entity pairs that satisfy the body, whereas SH(r) is a normalization S(r) by the number of pairs of entities that satisfy the head. The degree of support for a rule and its normalization are used as criteria for evaluating the extracted rules.

Rule extraction constitutes a central task in inductive logic programming [22, 24]. However, traditional logic programming systems are ill-suited for large-scale KG because they typically rely on negative facts, whereas KGs predominantly contain positive facts, and these methods are generally incapable of handling the immense volume of data present in a KG. Conversely, agents, integral to the agent-based paradigm, offer a solution by facilitating the construction of intelligent systems that meet the demands of flexibility and scalability—crucial elements for rule extraction and interpretation.

3. Formal agent architecture

To articulate the potential actions of an agent and its interactions with the external environment, a formal tool is indispensable for precisely describing the agent's behavior. The formal architecture of an agent serves as the foundation for understanding and designing an agent's behavior through clear and rigorous methods. This formal architecture is delineated through a description of the agent's operating environment, the agent's perception of this environment, and its corresponding actions [14, 26].

Let's denote the agent's external environment by the set of states S. The possible actions of the agent are described by the set of actions A [1]. Abstractly, an agent can be represented as a function of

$$g_s: S \to A$$

that is, the agent chooses a specific action from a set of possible actions based on the current state of the environment $s_i \in S$. At the same time, the agent's actions can influence the environment, but not completely control it.

To represent an agent, it is convenient to use a model of perception of the external environment. To do this, we introduce a set of possible perceptions P and a function $f: S \to P$ that describes how certain states of the environment are perceived by the agent. The agent is then represented by the function

$$g_P: P \to A$$

that is, the agent's action is determined in the general case by the current perception of the state of the environment $p_j \in P$. The agent model with perception is equivalent to the basic model. However, it allows us to introduce the following additional property of the agent: different states of the environment can be perceived in the same way and vice versa - one state can be perceived differently by the agent.

Another option for solving the problem of including previous actions when choosing a current action is to introduce the concept of the agent state. In this case, it is assumed that the agent has certain internal data structures that it modifies depending on the perception of the current state of the environment, and based on the results obtained, it chooses an action. To formalize this process, a set of I of internal states of the agent and the internal state update function, which is responsible for updating the internal state in accordance with the current perception of the environment:

$$h: I \times P \to I.$$

Then the agent is described using the function

$$g_I: I \to A$$

that is, the action is selected based on the current state

of the agent. To correctly describe the behavior of an agent with a state, it is necessary to determine the initial state $i_0 \epsilon I$. Such agent architecture has one significant drawback, namely that an agent defined in this manner does not receive information about its actions, which limits its ability to gain experience and analyze the potential consequences of its actions. One possible way to overcome this disadvantage is to present information about the agent's actions as part of the information about the external environment; however, this approach is not visual and intuitive. A more correct solution to this problem is to include information about the actions performed explicitly in the input data of the action selection function:

$$g_A: (P \times A)^* \to A.$$

In this configuration, the agent explicitly acquires information about previously executed actions. When selecting an action, it draws upon the perception of environmental states. In the case of an agent equipped with a state, the information regarding prior actions plays a role in the state update function:

$$h: I \times P \times A \to I.$$

The parameter of the state update function is not the sequence of all agent actions, but only the last action performed.

An agent that chooses an action based on the current perception, ignoring the entire history of previous perceptions, is a simple reflexive agent [2]. This type of agent is quite simple. In practice, simple reflexive agents are often sufficient to implement agent-based software systems [27]. In this case, we assume that each type of agent implements one function from the specification.

Let us define $S_I = \{s_0, s_1, s_2, \dots, s_n\}$ - is the set of possible states of a software system that characterize the interaction of software agents and components at a certain point in time, which together ensure the achievement of functional requirements in accordance with the specification.

Given $A_F = \{a_{F1}, \dots, a_{Fj}, \dots, a_{FS}\}$, the set representing agent actions aligning with the realization of functional requirements, the formal agent model is delineated as follows:

$$g_A: (S_I \times A_F) \to A_F.$$

Consequently, it becomes feasible to characterize the agents comprising a multi-agent system through a formal architecture that defines the agent's function as the selection of an action based on the current state of the software system and information regarding previously executed actions.

In the realm of using a multi-agent system to instantiate an intelligent system grounded in knowledge interpretation, this approach exhibits scalability and adeptness in handling substantial volumes of data.

Thus, addressing the second research question leads to the conclusion that software agents play a pivotal role in extracting rules from the knowledge base, processing, and interpreting them. This is accomplished through the implementation of an agent-based platform for parallel data processing, in which each agent is constructed in accordance with the aforementioned formal model.

4. Reference structure of knowledge interpretation

Elucidating or interpreting the behavior of an intelligent system represents a significant societal demand and a major challenge for both practitioners and theorists in the field of artificial intelligence. Despite the paramount importance of this task, there exists a notable absence of a logical formalization that captures these ubiquitous ideas. The formalization of knowledge, particularly in the form of rules that adhere to the definition of a closed path rule, has emerged as a potent tool for constructing a coherent and logical explanation.

It is essential to recognize that people may not find explanations compelling under certain conditions. First, when the extracted information holds true for scenarios beyond the one being explained, especially if this information is universally applicable. Second, individuals may exhibit disinterest if the extracted information merely reaffirms what they already know. In navigating the delicate balance of providing meaningful and pertinent explanations, the formalization of knowledge through closed path rules offers a structured and systematic approach to meet this pressing demand.

To formalize the explanation, we establish several foundational assumptions. We introduce the concepts of a deterministic system C_{sys}, denoted as the system under consideration, and input data I. The system C_{sys} processes the input data I and produces the desired result R requested by the user.

In the realm of knowledge interpretation, formalizing the concept of relevance in an explanation poses a challenge for logical formulation. Effective explanations must inherently consider both the context and the end-user, that is, the individual receiving the explanation. As a solution, this study introduces the concept of an "irrelevant explanation" as a formalism for evaluating interpretation results. Furthermore, a logical structure is presented to characterize the irrelevance concerning both the context and the user. This structured approach enhances the precision and clarity of the knowledge interpretation process.

ISSN 1814-4225 (print)
ISSN 2663-2012 (online)

Let's formalize the assumption.

Assumption 1: It is always feasible to represent the inputs and outputs of a system in a formal language. The input and output, or their respective descriptions, can be expressed as formulas denoted as I and R respectively.

Assumption 2: The characteristics of system C_{sys} can be represented by a logical theory T_S such that $T_S \cup \{I\} \leftarrow R$, i.e., inputs and outputs can be logically connected by T_S.

Assumption 3: The knowledge possessed by the user, denoted as the information consciously known to them, can be represented as another logical theory T_e. This theory encompasses both general knowledge and specific knowledge held by the user.

Assumption 4: The explanation consists of n sentences (fragments), each of which indicates the truth of the logical formula $E_i, i = 1, \ldots, n$ so that the entire explanation E can be represented as a conjunction $E = E_1 \wedge \cdots \wedge E_n$.

Assumption 5: The explanation is always true with respect to the specific behavior of the system it explains, i.e. $T_S \cup \{I\} \leftarrow E$.

Let's define the concept of an irrelevant explanation on the basis of the provided conditions.

1 (General irrelevance) An explanation E is considered irrelevant for the outcome R if there exist alternative inputs I' for which the system C_{sys} produces a different result R', such that *E* also explains the result R'. In essence, explanations that are overly general and fail to convincingly account for the outcome are deemed irrelevant.

2 (Partial irrelevance) An explanation E is considered irrelevant to the user if all $i \in \{1, \ldots, n\}$ holds $T_e \leftarrow E$, i.e., none of the conjuncts of E are unknown to the user. This consideration allows us to account for the specific knowledge possessed by individual users.

Partial irrelevance acknowledges that explanations can extend beyond a single phrase, encompassing a chain of reasoning from input to output. In this context, it becomes pivotal that at least one conjunct, denoted as E_i, forming the explanation remains unknown to the user. It is noteworthy that excluding parts of the explanation that are already known is not advocated, as these components contribute to the overall argument, and their omission would undermine the coherence of the entire explanation.

Therefore, a relevant explanation is one that incorporates at least one conjunct in explanation E that is unknown to the user. The constituents of E are fragments of rules extracted from the knowledge base, comprising individual facts that collectively develop a rule.

This definition of irrelevance serves as the foundation for constructing explanations derived from knowledge base information. The proposed formalization facilitates the automatic exclusion of irrelevant fragments or extracted rules from the comprehensive explanation, guided by the logical representation of the user's knowledge, such as through an ontology. In this context, the assessment of irrelevance can be conducted systematically.

Thus, we propose a formal logical framework for identifying irrelevant fragments within a complex explanation, operating under the premise that irrelevance is more amenable to logical formalization than the inherently subjective nature of relevance.

5. Multi-agent system architecture development

To determine the relevance of the explanations, we must first form an explanation E based on rule extraction from the knowledge base. In the following section, we consider a knowledge base as a knowledge graph formed by a set of entities and facts. We assume that there is a known knowledge graph $K = (E, F)$. Let us define P_t - the target predicate, i.e. the fact that needs to be interpreted. The task is to find the rules, conjunctions, and interpretation for a given target predicate P_t to find rules whose conjuncts form an explanation. In this study, we consider closed path rules. Note that attributive facts can be represented as binary predicates, as shown in [7]. Thus, let's consider how to study rules that have a target predicate in their head P_t. The proposed approach is based on the research of [24], which considers large amounts of data in the KG and uses an efficient algorithm to identify candidates for rule formation.

Let's delve into the structure of a multi-agent system designed for rule extraction. This approach involves categorizing agents according to the functional characteristics of the fundamental steps in the algorithm. Let's explore the key principles underlying the process of rule extraction.

1. We assume that the knowledge graph is given $K = (E, F)$ and the target predicate P_t, the maximum length of the rules L, the minimum values of the rule confidence $SB(r)$, the head coverage of the rule $SH(r)$ and the degree of support $S(r)$.

2. Due to the large size of the input data of the knowledge graph $K = (E, F)$ it is necessary to first sample the data. To achieve this goal, we propose to implement the function Sampling() to obtain a smaller KG $K' = (E', F')$ (where $E' \subset E, F' \subset F$)), which contains only those entities and facts relevant to the target predicate P_t.

3. To study the maximum length rules L we propose to build a sample of entities as follows:

$$E_0 = \{e \in E | \exists e' \in E: P_t(e, e') \in F \vee P_t(e', e) \in F\},$$

$$E_i = \{e \in E | \exists e' \in E_{i-1}: P(e, e') \in F \vee P(e', e) \in F\},$$
$$\forall i \in \{1, \dots, L-1\},$$

$$E' = \bigcup_{i=0}^{L} E_i,$$
$$F' = \{P(e_1, e_2) | e_1, e_2 \in E', P(e_1, e_2) \in F\}.$$

As a result, we have a set of entities E' and a set of facts F' representing the subgraph K' tangent to the target predicate P_t to a depth not exceeding L.

4 To proceed with the development of the candidate rules set, along with their subsequent evaluation and selection, it becomes imperative to implement the function Embeddings() that transforms them into a vector representation of predicates and their corresponding arguments. In essence, the implementation of this function is crucial for the effective representation and analysis of the extracted rules.

5 Searching for candidates for a set of rules in the form of CPRs, i.e. in the form of

$$p(x_0, x_{L-1}) \leftarrow p_1(x_0, x_1) \wedge \dots \wedge p_{L-1}(x_{L-2}, x_{L-1}),$$

actually, it comes down to finding sequences of predicates $P_1, P_2, \dots, P_{L-1}$ and their inverses. This is achieved by using the rule search function.

6 Resulting set of candidate rules $Cand = \{r_1, r_2, \dots, r_p\}$ should be evaluated. As a criterion for rule acceptance, a measure of similarity between the body of the candidate rule and the target predicate in the context of their vector representation can be used. For further selection, it is advisable to apply the minimum values of the rule confidence SB(r), rule head coverage SH(r) and the degree of support S(r) respectively, $SB_{min}, SH_{min}, S_{min}$ which reduces the number of rules obtained.

To implement these functions, we propose a multi-agent architecture, as illustrated in Figure 1. The following types of agents are defined:

A0: Agent-Ruler (coordinator), responsible for coordinating functions and serving as an interface for communication between the rule extraction module and other components. This includes interfacing with a Knowledge Interpretation Agent, which assesses the relevance of the obtained rules explaining the target predicate.

A1: Sampling Agent is a search space reduction agent, tasked with implementing the Sampling() function. This agent facilitates the construction of a knowledge subgraph $K' = (E', F')$, contributing to the reduction of the search space.

A2: Embedding Agent is an agent for vector representation of predicates and their arguments, a pivotal component for processing large volumes of facts within the KG. This agent enables the comparison of the head and body of a rule, thereby assessing their compatibility.

A3: Agent-Searcher, dedicated to constructing a set of candidate rules by considering all possible paths attainable on the subgraph K'.

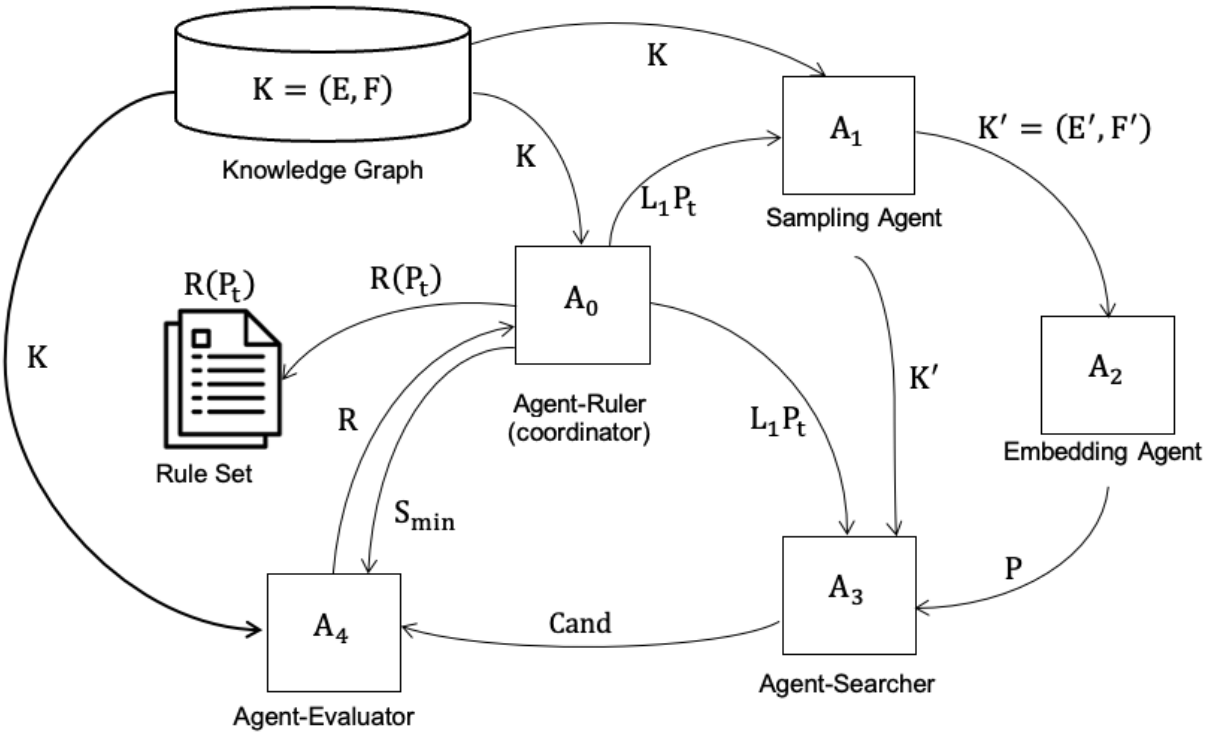

Fig. 1. Multi-agent implementation of rule mining

ISSN 1814-4225 (print)
ISSN 2663-2012 (online)

A4: Agent- Evaluator, responsible for implementing functions to evaluate the similarity of vectors in the body and head of the rule. In addition, it assesses the rule's confidence level SB(r), the coverage of the rule head SH(r) and the degree of support S(r).

Therefore, through the orchestrated interaction of the five types of agents, a sophisticated rule extraction algorithm can be effectively implemented. The actions of these agents are governed by corresponding functions, and their execution is contingent on the prevailing state of the software system. The resultant rules collectively compose an explanation and necessitate evaluation for relevance, both in a general sense and with respect to specific criteria.

6. Case-Study

Let's delve into an example illustrating the application of a multi-agent system for deriving rules and interpreting knowledge. The knowledge graph in focus comprises facts derived from electronic trading platforms, specifically encompassing attribute facts detailing various laptop models. As an illustration, consider the following attributes:

a – type of data storage that can accept values:
- a_1 – "HDD";
- a_2 – "SSD";
- a_3 – "SSD+HDD";

h – the RAM size:
- h_1 – "less than 16 GB";
- h_2 – "more than 16 GB";

i – the diagonal of the display:
- i_1 – 9'-12,5';
- i_2 – 13'-14';
- i_3 – 15'-15,6';
- i_4 – larger than 15.6'.

By capturing and interpreting various attributes, a comprehensive description of laptop features emerges, allowing for convenient categorization into distinct classes, as depicted in Figure 2. This classification system facilitates streamlined product discovery on e-commerce platforms, enhancing user experience and aiding in the identification of the most suitable products.

Let's designate, for instance, specific classes of laptops as follows:

c_1 – office laptops with a small monitor.

c_2 – powerful, large laptops.

It's important to note that the delineation of these classes is somewhat arbitrary and serves illustrative purposes. In practice, the formulation of classes results from the application of specific rules based on feature values. Alternatively, these classes can originate from the intellectual choices made by users, documented in the knowledge base as facts regarding which laptops were selected for particular search purposes. Subsequently, by leveraging the information about class matching and the particulars of features for a given model, it becomes feasible to develop rules for classifying a laptop into a specific category, offering a pertinent explanation.

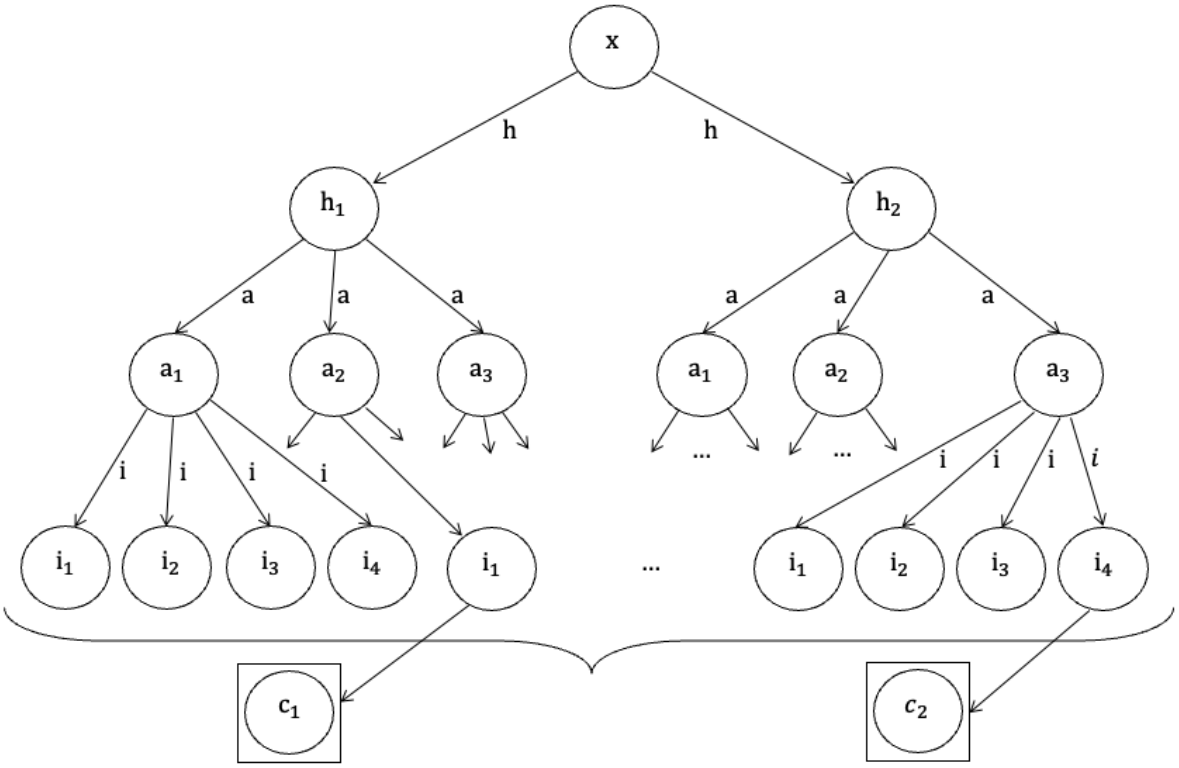

Fig. 2. Tree of features of laptop models (fragment)

Let's define a set of laptops $L = \{l_1, l_2, l_3\}$ for which the values of their attributes have been determined and the corresponding triplets in the KG have been formed (Fig. 3). Then let us denote the predicates that reflect the attribute facts as follows:

$h(x, y)$ – RAM size;
$i(x, y)$ – display diagonal;
$a(x, y)$ – type of data storage.

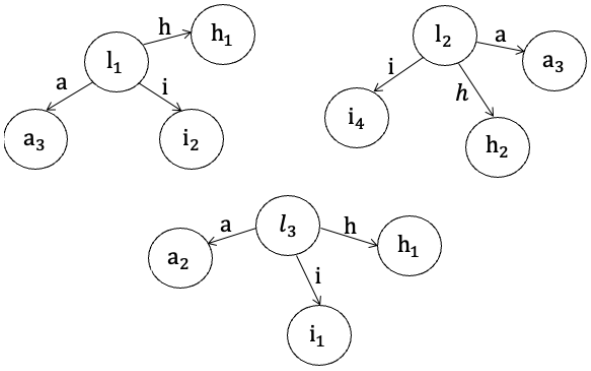

Fig. 3. Attribute facts in the KG (fragment)

Thus, for example, $h(l_1, h_1)$ – determines the fact that the laptop l_1 has less than 16 GB of RAM. All facts about the values of the laptop features can be written in triplets and corresponding predicates:

$$\begin{array}{ll} \langle l_1; a; a_3\rangle & a(l_1; a_3), \\ \langle l_1; h; h_1\rangle & h(l_1; h_1), \\ \langle l_1; i; i_2\rangle & i(l_1; i_2), \\ \langle l_2; a; a_3\rangle & a(l_2; a_3), \\ \multicolumn{2}{c}{\ldots} \\ \langle l_3; i; i_1\rangle & i(l_3; i_1). \end{array}$$

We assume that these laptops are divided into two classes: c_1 - office laptops with a small monitor; c_2 - powerful, large laptops. A fragment of the knowledge graph that reflects the known facts about the values of the features is depicted in Figure 4.

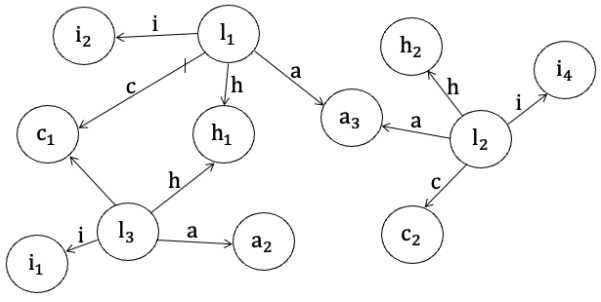

Fig. 4. Fragment of the knowledge graph

If the target predicate for rule retrieval is $c(x, y)$ then we can extract the following rules:

$$c(l_1; c_1) \leftarrow a(l_1; a_3)h(l_1; h_1)i(l_1; i_2);$$
$$c(l_3; c_1) \leftarrow a(l_3; a_2)h(l_3; h_1)i(l_3; i_1);$$
$$c(l_2; c_2) \leftarrow a(l_2; a_3)h(l_2; h_2)i(l_2; i_4).$$

From this, we can derive the following rules:

$$c(x; c_1) \leftarrow a(x; a_3)h(x; h_1)i(x; i_2) \vee$$
$$\vee\, a(x; a_2)h(x; h_1)i(x; i_1),$$

$$c(x; c_1) \leftarrow h(x; h_1)a((x; a_3)i(x; i_2) \vee$$
$$\vee\, a(x; a_2)i(x; i_1)).$$

This rule defines that the class of "office laptops with a small monitor" (c_1) can be classified as a laptop if its RAM is less than 16 GB and it has a 13'-14' or 9'-12.5' display and HDD or HDD+SSD storage.

If you want to find all laptops that match a class rule, you can set the target predicate to $c(x; c_1)$ then

$$c(l_1; c_1) \leftarrow a(l_1; a_3)h(l_1; h_1)i(l_1; i_2);$$
$$c(l_3; c_1) \leftarrow a(l_3; a_2)h(l_3; h_1)i(l_3; i_1).$$

From this, you can determine the value of the features of the laptop

$$l_1\colon\ a_3h_1i_2$$
$$l_3\colon\ a_2h_1i_3.$$

Thus, the multi-agent model for rule extraction and knowledge interpretation emerges as a robust and promising framework. The formalization of agent interactions facilitates the extraction of meaningful rules from a complex KG. The proposed approach, which involves distinct agent types performing specialized functions, enables efficient rule generation, evaluation, and interpretation.

7. Discussion

The presented example, where a KG derived from electronic trading platforms is utilized, showcases the practical application of the multi-agent model. By categorizing laptops on the basis of attributes and forming classes, the model effectively interprets and extracts relevant rules for improved system understanding.

The incorporation of contracts during the design and realization phases ensures the correctness of system implementation. Research on formal models of agent-based systems, including logical formalism and model checking, further contributes to the reliability and scalability of the multi-agent approach [28, 29]. The formal methodology for developing multi-agent systems typically comprises three stages:

1. Analysis Stage: This involves collecting informal requirements for the system.

2. Design Phase: This phase is accompanied by the creation and refinement of a model-based specification based on the requirements from the analysis phase.

3. Implementation Stage: This stage involves the actual realization of the system.

ISSN 1814-4225 (print)
ISSN 2663-2012 (online)

The concept of a contract becomes prominent during the design and realization stages. The design stage culminates in the creation of a model-oriented specification. A system is considered to have a correct implementation if it precisely adheres to the contract terms outlined in the specification.

In the last five years, numerous frameworks, platforms, and models within the agent-based approach have emerged, as discussed in [30]. Agent coordination and collaboration, particularly in terms of the platforms that facilitate them, are active research areas [31]. Despite the regular introduction of extensions and new languages, many remain limited to formal descriptions, lacking practical implementation to support their theoretical frameworks [30]. The primary challenge lies in their usability, which is compounded by a lack of effective evaluation methods. Consequently, a crucial research question arises as to how to compare existing languages and frameworks to enhance their functionality and broaden their applicability in real-world scenarios.

Although agent platforms offer versatility across various domains, several challenges persist. Notably, most existing platforms lack support for communication between agents developed using different frameworks, hindering interoperability [31]. To address this, an area of improvement involves incorporating common standards and protocols into agent platforms, facilitating seamless implementation across a broader range of applications.

Despite ongoing challenges related to diverse agent systems and the accommodation of self-organized and emergent behavior, the proposed framework provides a foundational approach to address these issues. Future advancements in multi-agent systems should prioritize certifying agent components for correctness, ensuring adaptability to system changes, and maintaining ongoing system integrity. Techniques such as dependency analysis and high-level contracts can significantly contribute to achieving these objectives.

Conclusions

This study contributes to the advancement of multi-agent systems by introducing an agent-formalized approach to the extraction of meaningful rules from complex knowledge graphs. In addressing the first research question, we determine that the resultant rules collectively form an explanation, necessitating evaluation for relevance in both a general sense and specific criteria. To address this issue, we propose a formal logical framework for identifying irrelevant fragments within a complex explanation, grounded in the premise that irrelevance is more amenable to logical formalization than the inherently subjective nature of relevance.

The resolution of the second research question leads to the conclusion that software agents play a pivotal role in extracting rules from the knowledge base, processing, and interpreting them. This is achieved through the implementation of an agent-based platform for parallel data processing, where each agent is constructed in accordance with the formal model. The proposed model, which involves the coordinated interaction of five distinct agent types, facilitates the generation, evaluation, and interpretation of rules. Consequently, it emerges as a robust and promising framework.

While the concept of an agent acting autonomously in the world is intuitively straightforward, the formal analysis of systems containing multiple agents is inherently complex. Understanding the properties of systems with multiple actors requires robust modeling and reasoning techniques to capture potential system evolutions. Such methods are essential for effectively modeling and analyzing agents and systems of agents through computation.

Research in formal models of agent-based systems aims to represent and comprehend system properties using logical formalism that describe both the mental states of individual agents and possible interactions within the system. Frequently employed logics include belief logics or other modalities, often incorporating temporal modalities. Efficient algorithms for theorem proving or model checking are required when dealing with large-scale problems. Recent efforts have used logical formalism to represent social properties such as agent coalitions, preferences, and game-type properties.

Formal methods such as model checking are crucial for testing, debugging, and verifying the properties of implemented multi-agent systems. Despite progress, challenges persist stemming from variations in agent systems, including paradigms, programming languages, and particularly the design of self-organized and emergent behavior. The latter may necessitate a programming paradigm that supports the automatic verification of both functional and non-functional properties. This leads to the need for certifying agent components for correctness against their specifications. Certification can be achieved by selecting components already verified using traditional methods or by automatically generating code from specifications.

In addition, methods are essential to ensure that the system continues to operate acceptably or safely during the adaptation process. This may involve techniques such as dependency analysis or high-level contracts and invariants to monitor the correctness of the system before, during, and after adaptation.

Contributions of authors: conceptualization, methodology, formulation of tasks – **Ihor Shubin;** analysis – **Oleksandr Karataiev;** development of mod-

el, software, verification – **Oleksandr Karataiev**; analysis of results, visualization – **Oleksandr Karataiev;** writing – original draft preparation visualization – **Oleksandr Karataiev;** writing – review and editing – **Ihor Shubin.**

All authors have read and agreed to the published version of this manuscript.

References

1. Wooldridge, M. *An Introduction to Multi-Agent Systems*. 2nd Ed., Wiley Publ., 2009. 488 p.
2. Russell, S. J., & Norvig, P. *Artificial Intelligence: A Modern Approach*. 4th Ed., Pearson, 2020. 1166 p. Available at: https://aima.cs.berkeley.edu/index.html. (accessed 12.07.2023).
3. Alkhateeb, F., Al Maghayreh, E., & Abu Doush, I. *Multi-Agent Systems – Modeling, Interactions, Simulations and Case Studies*. InTech, 2011. 514 p. DOI: 10.5772/1936.
4. *About the Common Object Request Broker Architecture Specification Version 2.0.* Available at: https://www.omg.org/spec/CORBA/2.0/About-CORBA (accessed 05.06.2023).
5. Bondarenko, M. F., & Shabanov-Kushnarenko, U. P. *Theory of intelligence: a Handbook*. Kharkiv, SMIT Company Publ., 2006.
6. Hinkelmann, F., Murrugarra, D., Jarrah, A. S., & Laubenbacher, R. C. A mathematical framework for agent based models of complex biological networks. *Bulletin of Mathematical Biology*, 2011, vol. 73, iss. 7, pp. 1583-1602. DOI: 10.1007/s11538-010-9582-8.
7. Sharonova, N., Doroshenko, A., & Cherednichenko, O. Issues of Fact-based Information Analysis. *International Conference on Computational Linguistics and Intelligent Systems*. CEUR Workshop Proceedings, 2018, vol. 2136, pp. 11-19. Available at: https://ceur-ws.org/Vol-2136/10000011.pdf (accessed 05.06.2023).
8. Botti, V., Mariani, S., Omicini, A., & Julian, V. *Multi-Agent Systems*. MDPI – Multidisciplinary Digital Publishing Institute, 2019. 392 p. ISBN 3038979252.
9. Wang, J., Deng, X., Guo, J., & Zeng, Z. Resilient Consensus Control for Multi-Agent Systems: A Comparative Survey. *Sensors*, 2023, vol. 23, iss. 6, article no. 2904. DOI: 10.3390/s23062904.
10. Niazi, M. A., & Hussain, A. A novel agent-based simulation framework for sensing in complex adaptive environments. *IEEE Sensors Journal*, 2011, vol. 11, iss. 2, pp. 404-412. DOI: 10.1109/JSEN.2010.2068044.
11. Nufer, G., & Muth, M. Artificial Intelligence in Marketing Analytics: The Application of Artificial Neural Networks for Brand Image Measurement. *Journal of Marketing Development and Competitiveness.* 2022, vol. 16, iss. 1. DOI: 10.33423/jmdc.v16i1.5027.
12. Dorri, A., Kanhere, S. S., & Jurdak, R. Multi-Agent Systems: A Survey. *IEEE Access*, 2018, vol. 6, pp. 28573-28593. DOI: 10.1109/ACCESS.2018.2831228.
13. Abar, S., Theodoropoulos, G. K., Lemarinier, P., & O'Hare, G. M. P. Agent Based Modelling and Simulation tools: A review of the state-of-art software. *Computer Science Review,* 2017, vol. 24, pp. 13-33. DOI: 10.1016/j.cosrev.2017.03.001.
14. North, M. J. A theoretical formalism for analyzing agent-based models. *Complex Adaptive Systems Modeling,* 2014, vol. 2, article no. 3. DOI: 10.1186/2194-3206-2-3.
15. Piccoli, B. Control of multi-agent systems: Results, open problems, and applications. *Open Mathematics*, 2023, vol. 21, iss. 1. DOI: 10.1515/math-2022-0585.
16. Houhamdi, Z. Multi-Agent System Testing: A Survey. *International Journal of Advanced Computer Science and Applications*, 2011, vol. 2, iss. 6. DOI: 10.14569/ijacsa.2011.020620.
17. Roggenbach, M., Cerone, A., Schlingloff, B.-H., Schneider, G., & Shaikh, S. *Formal Methods for Software Engineering: Languages, Methods, Application Domains*. Springer Cham Publ., 2021. 524 p. DOI: 10.1007/978-3-030-38800-3.
18. Smale, S. Differentiable dynamical systems. *Bulletin of the American Mathematical Society*, 1967, vol. 73, pp. 747-817. Available at: https://www.ams.org/journals/bull/1967-73-06/S0002-9904-1967-11798-1/S0002-9904-1967-11798-1.pdf (accessed 05.06.2023).
19. Nisan, N., Roughgarden, T., Tardos, E., & Vazirani, V. V. *Algorithmic Game Theory*. Cambridge, UK: Cambridge University Press, 2007. 754 p. Available at: https://www.cs.cmu.edu/~sandholm/cs15-892F13/algorithmic-game-theory.pdf (accessed 05.06.2023).
20. Cassandras, C. G., & Lafortune, S. *Introduction to Discrete Event Systems*. 2nd Ed., New York, USA, Springer Publ., 2010. 772 p. DOI: 10.1007/978-0-387-68612-7.
21. Dorofeenko, V., & Shorish, J. Dynamical Modeling of the Demographic Prisoner's Dilemma. *Reihe Ökonomie / Economics Series*, Vienna, Austria, Institute for Advanced Studies, 2002, no. 124. 25 p. Available at: https://www.econstor.eu/bitstream/10419/71200/1/740824198.pdf (accessed 05.06.2023).
22. Karataiev, O., & Shubin, I. Problemy povtornoho vykorystannya znan′ u protsesi proyektuvannya prohramnykh system [Reuse of information based on the interpretation of knowledge]. *Suchasnyy stan naukovykh doslidzhen′ ta tekhnolohiy v promyslovosti – Innovative technologies and scientific solutions for industries,* 2023, no. 2 (24), pp. 62-71, DOI: 10.30837/ITSSI.2023.24.062. (In Ukrainian).
23. Jarrahi, M. H., Lutz, C., & Newlands, G. Artificial intelligence, human intelligence and hybrid intelligence based on mutual augmentation. *Big Data & Society*, 2022, vol. 9, iss. 2. DOI: 10.1177/20539517221142824.
24. Omran, P. G., Wang, Z., & Wang, K. Learning Rules with Attributes and Relations in Knowledge Graphs. *AAAI Spring Symposium: MAKE*, 2022, vol.

ISSN 1814-4225 (print)
ISSN 2663-2012 (online)

3121. Available at: https://ceur-ws.org/Vol-3121/paper10.pdf (accessed 05.06.2023).

25. Brown, D. G., Riolo, R., Robinson, D. T., North, M. J., & Rand, W. Spatial process and data models: toward integration of agent-based models and GIS. *Journal of Geographical Systems*, 2005, vol. 7, pp. 25-47. DOI: 10.1007/s10109-005-0148-5.

26. Epstein, J. M. *Generative Social Science: Studies in Agent-Based Computational Modeling*. Princeton, NJ USA, Princeton University Press, 2006. 352 p. Available at: https://www.jstor.org/stable/j.ctt7rxj1 (accessed 05.06.2023).

27. Leombruni, R., & Richiardi, M. Why are economists sceptical about agent-based simulations? *Physica A: Statistical Mechanics and its Applications*, 2005, vol. 355, iss. 1, pp. 103-109. DOI: 10.1016/j.physa.2005.02.072.

28. Korteling, J. E. (Hans), van de Boer-Visschedijk, G. C., Blankendaal, R. A. M., Boonekamp, R. C., & Eikelboom, A. R. Human- versus Artificial Intelligence. *Frontiers in Artificial Intelligence*, 2021, vol. 4, article no. 622364. DOI: 10.3389/frai.2021.622364.

29. Strilets, V., Donets, V., Ugryumov, M., Artiuch, S., Zelenskyi, R., & Goncharova, T. Agent-oriented data clustering for medical monitoring. *Radioelectronic and Computer Systems*, 2022, no. 1, pp. 103-114. DOI: 10.32620/reks.2022.1.08.

30. Cardoso, R. C., & Ferrando, A. A Review of Agent-Based Programming for Multi-Agent Systems. *Computers*, 2021, vol. 10, article no. 16. DOI: 10.3390/computers10020016.

31. Wrona, Z., Buchwald, W., Ganzha, M., Paprzycki, M., Leon, F., Noor, N., & Pal, C.-V. Overview of Software Agent Platforms Available in 2023. *Information*, 2023, vol. 14, article no. 348. DOI: 10.3390/info14060348.

Received 17.07.2023, Accepted 20.11.2023

ФОРМАЛЬНА МОДЕЛЬ МУЛЬТИАГЕНТНОЇ АРХІТЕКТУРИ ПРОГРАМНОЇ СИСТЕМИ НА ОСНОВІ ІНТЕРПРЕТАЦІЇ ЗНАНЬ

Олександр Каратаєв, Ігор Шубін

Використання агентів у різних областях інформатики та штучного інтелекту переживає помітний сплеск у відповідь на вимоги адаптивності, ефективності та масштабованості. **Предметом** цього дослідження є застосування формальних методів для створення основи для інтерпретації знань у контексті агентної парадигми в розробці програмного забезпечення. **Метою дослідження** є розвиток формального підходу до інтерпретації знань шляхом використання агентної парадигми. **Завдання**: 1) вивчити поточний стан агентної парадигми в розробці програмного забезпечення; 2) описати основні поняття підходу до інтерпретації знань; 3) вивчити загальну структуру задачі видобування правил; 4) розробити еталонну структуру інтерпретації знань; 5) розробити мультиагентну архітектуру системи; 6) та обговорити результати дослідження. Дослідження використовує формальні **методи**, включаючи використання правил замкнутого шляху та логіку предикатів. Зокрема, інтеграція правил замкнутого шляху сприяє вилученню та поясненню фактів із баз знань. Отримані **результати**: 1) підхід до видобування правил, заснований на правилах замкнутого шляху та розроблений для обробки великих наборів даних; 2) модель релевантності, яка полегшує перевірку та автоматичне виключення нерелевантних фрагментів із пояснювальної структури; 3) багатоагентна система з п'яти типів агентів, призначених для видобування правил та інтерпретації отриманих знань. У статті наведено приклад застосування запропонованих принципів, демонструючи їх практичний контекст. **Висновок** підкреслює, що агентна парадигма з акцентом на децентралізацію та автономію представляє інноваційну структуру для вирішення завдань обробки знань. Це поширюється на пошук як фактів, так і правил. Розподіляючи функції між кількома агентами, пропонується динамічне та масштабоване рішення для ефективної інтерпретації величезних сховищ знань. Цей підхід стає особливо цінним у сценаріях, коли традиційні методи не можуть впоратися з обсягом і складністю інформації.

Ключові слова: мультиагентна система; програмна інженерія; формальна модель; інтерпретація знань; правило замкнутого шляху.

Каратаєв Олександр Анатолійович – асп. каф. програмної інженерії, Харківський національний університет радіоелектроніки, Харків, Україна.

Шубін Ігор Юрійович – канд. техн. наук, доц., проф. каф. програмної інженерії, Харківський національний університет радіоелектроніки, Харків, Україна.

Oleksandr Karataiev – PhD Student of the Department of Software Engineering, Kharkiv National University of Radioelectronics, Kharkiv, Ukraine,
e-mail: tosanik@gmail.com.

Ihor Shubin – Candidate of Technical Sciences, Associate Professor, Professor at the Department of Software Engineering, Kharkiv National University of Radioelectronics, Kharkiv, Ukraine,
e-mail: igor.shubin@nure.ua, ORCID: 0000-0002-1073-023X, Scopus Author ID: 57188703184.