**UDK 004.89**

**V. S. Lenko[1], V. V. Pasichnyk[1], Y. M. Shcherbyna[2]**
[1]Lviv Polytechnic National University
[2]Ivan Franko National University of Lviv

# KNOWLEDGE REPRESENTATION MODELS

**The paper addresses an issue of knowledge representation (KR) in the intelligent systems. The most widely-used techniques of KR are considered, including first-order logic, type theories, semantic networks, frames, scripts, production rules and ontologies. Each technique is investigated for the presence of advantages and limitations of its application on practical and theoretical levels. The relationship between different approaches of knowledge representation is discussed.**

**Key words: knowledge representation, first-order logic, type theories, semantic networks, frames, scripts, production rules, ontologies, knowledge systems.**

## Introduction

An intelligence that is exposed by human beings is one of the most mysterious phenomena to which the science still doesn't have a clear explanation. One of the definitions proposed by an artificial intelligence states that it as a process of logical inference (reasoning) that is applied to a stored knowledge. What is important about the definition is that it emphasizes an existence of two categories, knowledge and reasoning, which are interconnected; moreover the presence of knowledge is necessary for the reasoning process to be performed. Thus the problem of acquiring, storing and retrieving the knowledge appears to be the primary one in a task of building an intelligent system.

Once the knowledge is acquired, it should be stored in a knowledge base for the later use in reasoning. Knowledge base (KB) typically consists of the facts that are true by definition. The structure of the facts may vary from a simple propositional sentence to a complex ontology. The way the facts are organized in the KB significantly affects the inference mechanisms that could be applied to the stored knowledge. Knowledge representation study aims at researching the optimal techniques of knowledge description and structuring, which should facilitate the tasks of KB creation and reasoning. Among the variety of criteria that are used for evaluation of KR techniques, the expressiveness and inference capabilities are of a great significance.

## The Task Analysis

Knowledge representation models are typically divided into four classes: logic-based models, semantic networks, frame-based models and production-based models [1]. Logic-based models constitute of the formal systems that provide formal languages and deductive systems for knowledge description and reasoning respectively. The paper addresses first-order logic and type theories, also known as higher-order logics. Semantic networks follow more natural, graph approach to knowledge representation. Here graph nodes correspond to the concepts and graph arcs stand for the relationship between concepts. The most common kinds of semantic networks are: definitional, assertional, implicational, executable, learning and hybrid [2]. Frame-based models take an object-oriented approach, where the knowledge is structured around the concept of "an object". The presence of "an object" enables grouping of the related properties and procedures, therefore simplifies description and management of the knowledge. The paper also highlights a special kind of frames – scripts, which are used for describing a stereotyped sequence of events in a particular context [3]. Production-based models represent the knowledge as a collection of "if-then" rules and are supplied with the methods of an automated or semi-automated reasoning. Expert systems are the most famous implementation of this approach.

The most recent research is conducted in the areas of an ontology engineering and a type theory. Ontology as a KR technique combines the aforementioned models of knowledge representation and has many active implementations, including Protege, Cyc, Dublin Core, etc. Some optimization solutions for the issue of an ontology growing are proposed in [4]. Type theories are considered to be an important member in the formal logics. Since a Curry-Howard isomorphism discovery they are treated with a special interest. Nowadays a lot of efforts are dedicated to a homotopy type theory [5].

## Knowledge Representation Models

### Predicate logic (first-order logic)

First-order logic (FOL) is a collection of formal systems that consist of a formal language for statements description and a deductive system for reasoning. There are many variations of FOL, including intuitionistic, many-sorted, modal, etc.; here we will discuss the classical one. First-order logic is the standard for the formalization of mathematics into axioms and also plays a crucial role in representation of the knowledge of almost any kind. It appeared as a successor to a propositional logic, which does not possess a sufficient expressiveness to represent real-world knowledge. The main difference between first-order logic and propositional logic is basically in the ontological commitment: FOL, like natural language, assumes that world contains objects, relations and functions, while propositional logic operates only with facts. FOL is useful for showing logic relationships and their reasoning; moreover it allows logic statement to be split into words. As an example, let's consider two logical statements "Church is mathematician" and "Turing is mathematician". In propositional logic these statements are viewed as being unrelated, while in predicate logic we are allowed to introduce a predicate "Mathematician (x)" that makes a relationship between the statements explicit.

The formal language of FOL consists of syntax and semantics. Syntax specifies which collections of symbols from an alphabet are legal expressions, while semantics determines the meanings of these expressions. An alphabet is a set of symbols from which the strings of a formal language may be formed. In first-order logic the symbols of alphabet are divided into two groups, logical and non-logical. The difference is that logical symbols always have the same meaning, while the meaning of non-logical symbols depends on interpretation. The logical symbols in FOL are usually represented by quantifiers $(\forall, \exists)$, logical connectives $(\wedge, \vee, \rightarrow, \leftrightarrow, \neg)$, punctuation (parentheses), variables (infinite set), equality (=). Non-logical symbols consist of predicates and functions with a valence (number of arguments) greater or equal to zero. Predicate of valence 0 corresponds to a propositional variable and function of valence 0 is called constant. The formation rules define two types of legal syntactic expressions in FOL, terms and formulas. Terms are represented by variables and functions, while formulas consist of predicates (accept terms as arguments), equality symbol (operates on terms), logical connectives and quantifiers. A variable in formula might be bound or free (or both), depending on whether it lies within the scope of a quantifier or not. A formula with no free variables is called a first-order sentence. The first-order sentences are the formulas that will have well-defined truth values under an interpretation, therefore they are used to represent the knowledge; the rest is merely supporting syntactic machinery. The semantics of first-order logic is provided by an interpretation, which specifies the domain of discourse and denotation of the non-logical symbols.

The reasoning in FOL is performed within a deductive system, which is used to show that one formula is a logical consequence of another formula. Most deductive systems consist of logical axioms (possibly none) and inference rules that can be used to derive the theorems of the system. These systems possess an important feature – they are purely syntactic, which means derivations can be verified without considering any interpretation. As a result each derivation is a syntactic consequence of all the expressions that precede it. There are many deductive systems for the first-order logic, including Hilbert-style system, natural deduction, sequent calculus, tableaux method, resolution. A deductive system is called *sound* if it produces only logically valid formulas, those that are guaranteed to be a logical consequence. A deductive system is called *complete* if it is guaranteed to produce all logically valid formulas. Soundness and completeness are relative to a semantics suitable for the language of a deduction system. Unfortunately no automated reasoning process for FOL can be both sound and complete, since the problem of determining whether one sentence is a logical consequence of others in FOL is in general unsolvable [6].

First-order logic is a good starting point for the knowledge representation and reasoning, since it has many metalogical properties that stronger logics do not have. According to Lindström's theorem, FOL is a maximal logic that satisfies the downward Löwenheim-Skolem property and the countable compactness property [7]. Nevertheless it also suffers from the significant limitations. At first, it is affected by Skolem's paradox, which implies that infinite structures (i.e. real line, natural numbers) cannot be categorically axiomatized in first-order logic; this issue is solved in stronger logics such as second-order logic. At second, first-order logic is not sufficient for expressing some typical features of a natural language, like quantification over properties, which requires the presence of a quantifier over predicates. Finally, the computational difficulty of FOL is among the major factors that lead to consideration of various other options for knowledge representation and reasoning [6].

## Type theories (higher-order logics)

Initially presented by Russell, type theory plays a fundamental role in both logic and mathematics. It appeared as a response to the paradox discovered by Russell in a naive set theory, which demonstrates that the membership of "set of all sets that are not members of themselves" is a contradiction. Once the paradox was reproduced in the Frege's system through using extension of predicates as arguments to predicates, Frege pointed out that the expression "a predicate is predicated of itself" is not exact. Frege had a distinction between predicates and objects, so that predicate could only be applied to an object and cannot accept a predicate as an argument. The presence of distinction between objects, predicates, predicate of predicates was enough to block the paradox. This hierarchy is called "an extensional hierarchy of type" by Russell, and its necessity was recognized as the consequence of his paradox [8].

The fundamental work in formal logic "Principia Mathematica" defines the type structure as follows:

- $i$ is the type of individuals, () is the type of propositions;
- If $A_1,...,A_n$ are types then $(A_1,...,A_n)$ is the type of $n$-ary relation over objects of those types.

Since predicate is classified as $n$-ary relation of type $(A_1,...,A_n)$ it is obvious that it cannot accept arguments of type $(A)$, in particular predicates, thus avoiding the Russell's paradox. According to the specified structure it is possible to define also the types of binary relation $(i,i)$, binary connectives $((),())$ and quantifiers over individuals $((i))$. Church provided his own formulation of a type theory that is based on a simply-typed λ-calculus notation; it gave the opportunity to introduce functions as primitive objects, and use them in arguments and return values. The types in Church's formal system are defined as follows:

- $i$ is the type of individuals, $o$ is the type of propositions;
- If $A, B$ are types then $A \to B$ is a type of functions from $A$ to $B$.

According to the structure it is possible to form the next types: $i \to i$ (functions), $(i \to i) \to i$ (functionals), $i \to o$ (predicates), $(i \to o) \to o$ (predicate of predicates).

One of the most notable discoveries is a Curry-Howard correspondence that established relationship between two families of formalisms, proof systems and models of computation, in particular intuitionistic logic and simply-typed λ-calculus. In his paper Howard describes correspondence between propositional connectives and computational types, as well as introduces new types called "dependent types" that correspond to predicate quantifiers. Introduction of the dependent types, allows representing a proof in a predicate logic by the terms of a simply typed λ-calculus. As a consequence the concepts of "propositions as types", "proofs as programs", "simplification of proofs as evaluation of programs" have arisen [9]. It led to the development of a new kind of software called "proof assistant", with the most famous representatives named "Coq" and "NuPRL".

Lambek has shown the connection between proofs in intuitionistic logic and cartesian closed categories. Curry-Howard-Lambek correspondence made proof theory, type theory and category theory highly interconnected through the isomorphic interpretation of the entities of intuitionistic logic, typed λ-calculus and cartesian closed categories. Nowadays the most active research is conducted in the area of homotopy type theory (HoTT), which links topology to "propositions as types".

| Types | Logic | Sets | Homotopy |
|---|---|---|---|
| $A$ | proposition | set | space |
| $a:A$ | proof | element | point |
| $B(x)$ | predicate | family of sets | fibration |
| $b(x):B(x)$ | conditional proof | family of elements | section |
| $0,1$ | $\bot, \mathrm{T}$ | $\varnothing, \{\varnothing\}$ | $\varnothing, *$ |
| $A+B$ | $A \vee B$ | disjoin union | coproduct |
| $A \times B$ | $A \wedge B$ | set of pairs | product space |
| $A \to B$ | $A \Rightarrow B$ | set of functions | function space |
| $\sum_{(x:A)} B(x)$ | $\exists_{(x:A)} B(x)$ | disjoint sum | total space |
| $\prod_{(x:A)} B(x)$ | $\forall_{(x:A)} B(x)$ | product | space of sections |
| $Id_A$ | equality $=$ | $\{(x,x) \mid x \in A\}$ | path space $A^I$ |

*Fig. 1. Comparing points of view on type-theoretic operations; presented in [5]*

## Semantic networks

A semantic network is a knowledge representation technique that uses a graph notation to describe the structure of knowledge. According to it concepts, instances and properties correspond to a graph nodes, while relations between them (i.e. taxonomy) are specified via directed arcs. Formally, semantic network is defined as a tuple $< I, R_1, R_2, ..., R_n, M >$, where $I$ stands for information entities, $R_1, R_2, ..., R_n$ denotes types of relations between information entities, and $M$ corresponds to a map that establishes the relations between information entities [1]. The diversity of a nature of elements in the tuple and the operations they support allows distinguishing the following most common kinds of semantic networks: definitional, assertional, implicational, executable, learning and hybrid [2]. Despite the significant differences in information entities and networks application, all of them use a declarative graphic approach to the knowledge representation.

There exist various kinds of semantic networks, so let's look more precisely on them. Definitional networks are used to describe concepts, instances and relations between them. They heavily utilize the relations of types "is-a", "has-a", "instance-of" to form a taxonomy of knowledge that is essential for the external reasoning mechanism. Assertional network represents graphically the logical propositions, therefore its elements carry additional semantic roles that correspond to the logical operators, such as existential quantifier, conjunction, disjunction and others. Implicational networks are a special case of a propositional semantic network that uses implication as a primary relation for connecting nodes; other relations may be nested inside the propositional nodes, but they are ignored by the inference procedures. Executable networks contain mechanisms that can cause some change to the network itself. The most common executable mechanisms are message passing, attached procedures and graph transformations.

Learning networks adjust their internal representations in a response to new information, in order to perform effectively in their environment. A learning network could be adjusted in three ways: rote memory, changing weights and restructuring. Finally, hybrid networks are the ones that simultaneously incorporate multiple types of semantic networks, in order to reuse advantages and eliminate drawbacks of the aforementioned networks.

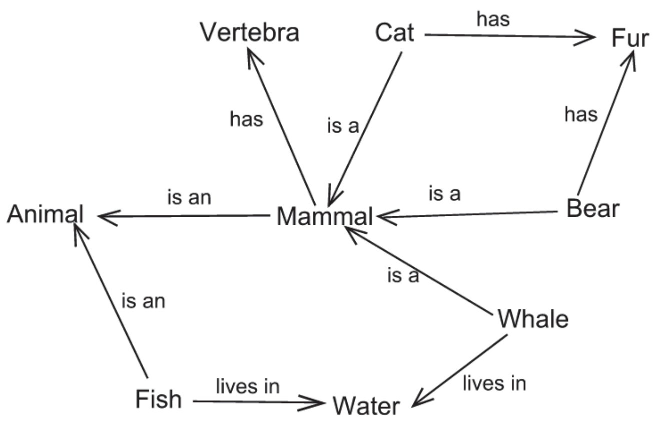

*Figure 2. An example of a definitional semantic network [10]*

Among the major advantages of semantic networks is the easiness of knowledge conception for a human, which is provided by a graphic representation. At first, graphic notation shows direct connections between the related concepts, while linear notation must rely on the repeated occurrences of variables or names to show the same connections. At second, graphic notations often have heuristic value in helping human readers to discover patterns that would be difficult or impossible to see in a linear form; this could be explained by the clustered appearance of a related knowledge. Speaking about the drawbacks of semantic networks it is important to highlight the lack of standardization of nodes and arcs values, which introduces a maintenance complexity. Other than that, some types of semantic networks suffer from specific issues like multiple inheritance errors, one-to-many relations, absence of a formal semantics, mixing the different levels of abstraction. Nevertheless, a "natural" approach to knowledge representation makes semantic networks a widely-used technique in the knowledge systems engineering.

## Frames

Frame is an object-centered form of knowledge representation that makes its ontological commitment to objects, properties, values and relations between them. As a result, it is often called an "object-property-value" representation. Objects are a natural way to organize the knowledge about physical objects and situations. What is of a great importance is that object enable grouping of procedures for determining its own properties, as well as the properties of other objects. In FOL statements about the world are scattered around, therefore it is difficult to control the consistency and comprehensiveness of the recorded knowledge. The object-centered representation could be reached by merging many properties of the object of the same type into one structure.

In frame-based systems, an object that contains properties filled by values corresponds to a "frame" that contains "slots" filled by "fillers"; therefore frame could be defined as a structural and relational representation of an object. Each frame has a name and consists of a set of slots with the associated fillers. The filler could take a value of the primitive types (number, string, date, etc.), complex types (set, list) or serve as a reference to another frame. Based on the nature of a description object, two types of frames are distinguished – individual and generic. Individual frame represents a single object, which is an instance of some category, while generic frame is used to describe abstract category. Consequently individual frames by default have a special slot called "INSTANCE-OF", where the filler is the name of (or a pointer to) an appropriate generic frame. At the same time, generic frames *may* have a special slot called "IS-A" with a filler that points to a more generic frame. The presence of these special slots is crucial, since they enable an inheritance mechanism.

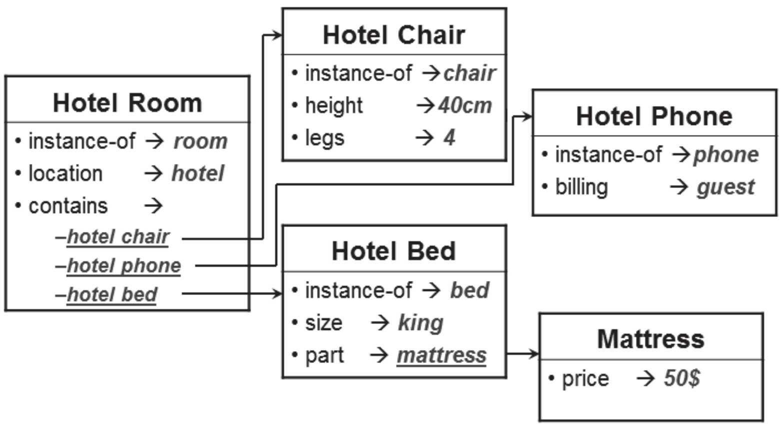

*Fig. 3. Set of frames that describe a hotel room [11]*

The inference process is typically controlled via procedures, which are associated with the slots in the generic frames. There are two types of procedures – "IF-NEEDED" and "IF-ADDED". The first one is executed when slot filler is absent and the value is needed; the second one is executed when slot filler is present and it might affect the other frames. The most common use cases for these procedures are: providing default values for empty slots, expressing constraints between slots, consistency maintenance. Procedures and fillers of more general frame are applicable to more specific frame through an inheritance. The reasoning in frame-based systems is significantly influenced by these procedures and usually has the following order: 1) user instantiates a new frame, in order to declare an existence of some object or situation; 2) slot fillers are inherited where possible; 3) inherited "IF-ADDED" procedures are executed, causing more frames to be instantiated and slots to be filled; 4) if the user or any procedure requires the slot filler that is missing, then an inherited "IF-NEEDED" procedure is run, potentially causing additional actions.

As usual frame representation has its own advantages and drawbacks. The major benefits come from the structural peculiarities. An object-centered approach enables grouping of properties and procedures related to the objects of the same type, simplifying the processes of inference and consistency maintenance. Slot fillers that point to the other frames, makes the frames implicitly associated with each other, thus keeping the related knowledge clustered. New properties and relations can be easily added via new slots, while the associated procedures can propagate default fillers for the introduced slots to the existing frames. Moreover, procedures make derived (computed) properties to look explicit and could avoid potentially unneeded computations. Frames allow both declarative and procedural control of the inference process, thus reducing the total efforts needed for a system creation. An elegant implementation of the inheritance mechanism highlights the expressiveness of a frame-based approach. Among the major drawbacks are: the absence of a formal reasoning (inference) mechanism; the lack of the standards for slot-filler values; user-dependency – a frame representing the same object, might significantly vary for two different users.

## Scripts

A script is a structured knowledge representation scheme, describing a stereotyped sequence of events in a particular context. An event corresponds to a time-dependent action that may indicate cause and effect relationships. Scripts are very similar to the frames, except the values that fill the slots have to be ordered; nevertheless the frames are considered to be more general technique of knowledge representation. Scripts are used in natural language understanding systems to organize a knowledge base in terms of the situations that the system should understand. They provide knowledge and expectations about specific events or experiences and can be applied to the new situations. Scripts are soundly beneficial, because the real world events do follow stereotyped patterns, as human beings use previous experiences to understand the new situations.

To describe a sequence of events, the script uses a series of slots containing information about the distinguished elements, which form the background or are involved in the events. Usually script has name and consists of the following slots: track, roles, props, entry conditions, scenes and results. Track is used to present the variations of a particular script. Roles describe the people involved in a script, while props refer to the objects that are used in the sequence of events. The entry conditions define premises that must be satisfied, before events in this script can occur or will be valid. Scenes describe the actual sequence of events that occur. Finally, the results refer to the conditions that exist after the events in the script have occurred. It worth to mention that some scenes in a script might be optional, thus increasing the general applicability of the script to various use cases.

The knowledge that is represented by a script is typically stored in a symbolic form. This can be easily achieved by using LISP or any other symbolic language. The main functionality of a script is to answer the questions related to the roles, objects and results of its execution in the context of a provided situation. The reasoning process in a script consists in an application of a search and pattern-matching methods that examine the script for answers. Even though some situations have not been observed, it allows to predict what will happen to whom and when. Once a script is triggered, the user may ask questions and receive accurate derived answers with little or no original input knowledge. This peculiarity highlights the power of stereotyped knowledge representation techniques.

To summarize, script allows building up a single coherent interpretation from a collection of observations. It provides a high degree of flexibility and the ability to predict events. Stereotyped nature of the scripts makes them applicable to a variety of the real world situations. The major limitation of a script is the difficulty in representation of complex concepts and relationships; moreover not all the knowledge can be represented as a script. Sometimes it is also hard to select a precise structure of a script to reach an optimal reasoning performance. As it was already mentioned, they are less general than frames and could be treated as a partial case of frames.

| | | | |
|---|---|---|---|
| Script:<br><br>Track:<br><br>Props:<br><br><br><br><br>Roles: | Going to a restaurant<br><br>Regular restaurant<br><br>Food<br>Tables<br>Menu<br>Money<br><br>Owner<br>Customer<br>Waiter<br>Cashier | Scene 1: | “Entering the restaurant”<br>Customer enters the restaurant<br>Scans the tables<br>Chooses the best one<br>Decides to sit there<br>Goes there<br>Occupies the seat |
| | | Scene 2: | “Ordering the food”<br>Customer asks for menu<br>Waiter brings it<br>Customer glances it<br>Chooses what to eat<br>Orders that item |
| Entry Conditions:<br>Customer is hungry<br>Customer has money<br>Owner has food | | Scene 3: | “Eating the food”<br>Waiter brings the food<br>Customer glances it<br>Customer eats it |
| Results: | Customer is not hungry<br>Owner has more money<br>Customer has less money<br>Owner has less food | Scene 4: | “Paying the bill”<br>Customer asks for the bill<br>Waiter brings it<br>Customer pays for it<br>Waiter hands the cash to the cashier<br>Waiter brings the balance amount<br>Customer tips him<br>Customer moves out of the restaurant |

*Fig. 4. Pseudo-form of a restaurant script [12]*

## Production rules

Production rules, also known as "IF-THEN" rules, are one of the most widely-used forms of knowledge representation, especially in the expert systems (ES) and automated decision support (ADS) systems. They are characterized by a high modularity, which brings flexibility to the process of KB instantiation and maintenance. Each rule defines a relatively small and independent piece of knowledge; moreover it can be easily added and deleted independently to the other rules. Simplicity of the syntax and a natural way of knowledge expression, results in a simple interpretability and ease of understanding. Typical production rule consists of the antecedent and consequent. Antecedent corresponds to a premise (condition, IF-clause) and consequent stands for an action (conclusion, THEN-clause). To enable a reasoning mechanism, the rules are usually grouped into a so called *production system*, which has its own structural and functional peculiarities.

```
IF    the stain of the organism is gram negative
AND   the morphology of the organism is rod
AND   the aerobiocity of the organism is gram anaerobic
THEN  there is strong evidence (0.8)
      that the class of the organism is enterobacteriaceae
```

*Fig. 5. Production rule from MYCIN expert system [13]*

Production system consists of a knowledge base and an inference engine, thus resulting in two types of rules – declarative and procedural. Declarative rules state all the facts and relationships about the problem, while procedural rules advice on how to solve the problem, given that certain facts are known. They also differ by a storage location: the first ones are stored in the KB and the second ones are incorporated into the inference mechanism. Knowledge base of a production system has its own specifics and is divided into a working memory and a rule base. Working memory consists of a set of facts, which are basically the statements about the world, but can also represent various data structures. Syntax and representation of the facts are not strictly formalized, thus may differ across the systems. A rule base is formed by a set of rules, which belong to a special type of "IF-THEN" rules, where an antecedent is represented by a conjunction of conditions and a consequent represents a sequence of actions. A structure of a random entry in a rule base is defined via the formula $p_1 \wedge p_2 \wedge ... \wedge p_n \Rightarrow a_1, a_2, ..., a_k$ . Antecedent members might take a form of the negated or non-negated predicate statements and conditions on a particular object or multiple objects. Consequent usually consists of the following actions: "ADD" the fact to the working memory, "REMOVE" the fact from the working memory, "MODIFY" an attribute field, "QUERY" the user for an input; i.e. $A(x) \wedge B(x) \wedge C(y) \Rightarrow REMOVE : D(x), ADD : B(y)$.

An inference mechanism in the production systems is typically implemented using the forward chaining method. It starts with the facts available in a working memory and uses a set of "active" rules from KB to extract more facts until a goal is reached. A rule is said to be active, if its antecedent is satisfied; it is a necessary condition for the consequent to be executed. It is a common situation when multiple rules are active at the same time, thus introducing a need to specify the order of their execution. In fact, the order of execution is important, since ADD and REMOVE actions can change the set of active rules, so that next active rule in a queue might be deactivated by a prior one. Strategy for selecting the rule to be executed from among possible candidates is called conflict resolution. Below are listed some of the most popular conflict resolution strategies:

- No duplication – do not execute the same rule twice;
- Recency – rules referring to facts newly added to the working memory take precedence;
- Specificity – rules that are more specific are preferred over more generic ones;
- Priority levels – assign priority levels to the rules; a rule with higher priority takes precedence.

Another issue that appears in the production systems reasoning is related to the process of unification. Some rules may require a large number of unifications, especially if they contain variables testing all the instances where the rules are active. Additionally, conditions of many rules may overlap, thus resulting in the multiple repeats of the same unifications. To address the issues of inefficient unifications and conflict resolution a Rete algorithm has been introduced. The key principles of its functioning are as follows: the rules are compiled into a network that merges conditions of multiple rules together, thus avoiding duplication; only valid unifications are propagated; only changed conditions are reevaluated.

To conclude, production rules provide a great modularity and flexibility to the task of knowledge representation. They are expressive, easy to understand and are supported by a powerful inference engine available in production systems. Moreover their utility has already been proven by the success of the vast commercial expert systems. Speaking about the drawbacks, it is important to mention that not all the knowledge could be expressed in the form of rules. Production rules are poor at representing structured descriptive knowledge and it is hard to follow the hierarchies. Large systems typically consist of thousands of rules, thus making the knowledge consolidation difficult and significantly decreasing the performance of an inference engine.

## Ontologies

Ontology is a modern technique of knowledge representation that is grounded on a conceptualization. Conceptualization is an abstract, simplified view of some selected part of the world, containing the objects, concepts, and other entities that are assumed to exist in some area of interest and the relationships that hold among them [14]. Since conceptualization can be implemented in multiple ways, it could be represented by several distinct ontologies. Thus, a widely accepted definition for ontology is "an explicit specification of a conceptualization". Along with the ontology comes the notion of "ontological commitment". It is said, that a theory is ontologically committed to a given object only if that object is common to all of the ontologies fulfilling the theory. Therefore, ontological commitment can be defined as the object or objects common to an ontology fulfilling some theory [15].

Formally ontology consists of the concepts organized in taxonomy, their properties or attributes, related axioms and inference rules [16]. An ontology together with a set of individual instances of classes constitutes a knowledge base [17]. To denote the common elements of ontology the following notation is typically used:

- class (concept) – stands for a concept in the domain of discourse;
- instance (individual) – represent an individual instance of a specific class;
- slot (property, role) – describes a feature or an attribute of concept or individual;
- facet (role restriction) – represents a restriction on a slot;
- relations – describes the way in which classes and instances can be related to each other;
- rules – "if-then" sentences that describe possible logical inferences from an assertion;
- axioms – assertions in a logical form that comprise an overall theory of a domain of discourse.

Ontology engineering requires the usage of all four major models of knowledge representations: frames are used for the knowledge declaration; semantic networks are used for the relations declaration; axioms are defined using second-order logic and production systems represent the rules of inference [16]. The process of an ontology building consists of four steps: defining classes in the ontology; arranging classes in taxonomy; defining slots and describing allowed values for them; filling in the slot values in the individuals [17]. Inference rules and axioms can be added at the third step, if appropriate.

Ontologies have many practical applications in the real world. At first, they serve as a mechanism for sharing common understanding of the structure of information. A person or a software agent can access published ontologies, in order to reuse the existing knowledge for building integrated, extended or dedicated ontology. It is common to distinguish domain-oriented, task-oriented and top-level ontologies. Domain-oriented ontologies are tied to a specific universe of discourse and model the corresponding domain knowledge, while top-level ones act as foundation for more specific domain ontologies providing definition for general-purpose concepts [18]. Task-oriented ontologies are also of a significant interest since they can be applied to different domains, avoiding the limitation of the ad-hoc ontologies [19]. At second, ontologies could be used for reasoning and inferring new knowledge. These activities are usually implemented according to the "tell and ask" functional interface, where a client interacts with a knowledge system by making logical assertions (tell) and posing queries (ask). Finally, ontologies possess some useful features, like separation of the domain knowledge from operational knowledge, explicit definition of domain assumptions, knowledge reusability, etc. [17]

Construction of an ontology with a purpose to solve real problem requires a significant amount of resources – domain experts, technical engineers, time. Since ontology development is an iterative process, maintenance burden is also inevitable. One could benefit from reusing the existing ontologies, but it doesn't eliminate the need for human involvement that still could be enormous. Therefore, it makes sense to search for automated or semi-automated methods of ontology creation and maintenance that could reduce the required resources to a minimum. Facilitating the construction of ontologies and populating them with instances of both concepts and relations commonly referred to an ontology learning [20]. There exist multiple types of ontology learning. The most popular is supposed to be an ontology learning from text; it generates lightweight taxonomies by the means of text mining and information extraction. The other types like linked data mining, concept learning in description logics and OWL, crowdsourcing ontologies are also investigated. Another important task that comes with the ontology learning is an evaluation of the ontology quality. In practice, the quality of a learned ontology often depends on the degree of automation. Typically the higher involvement of a human in a semi-automatic ontology generation leads to a higher quality. If an ontology is insufficient for the intended application in terms of quality, it will eventually require a significant amount of post-processing [20].

## Conclusions

The paper presents an overview of the various knowledge representation techniques, their advantages and limitations. The formal logic is discussed in the light of first-order logics and type theories. The expressiveness and metalogical properties of the FOL have been highlighted. The diversity of semantic network models and the benefits of graphical notation introduce a good alternative to the traditional formal logic; however the lack of standardization and multiple specific issues removes the status of a "silver bullet" from it. Stereotyped knowledge representation techniques are discussed in terms of frames and scripts. Frames bring the power of default values, explicit computed properties and a procedural control of the inheritance; nevertheless it suffers from the absence of a formal reasoning mechanism and impossibility to represent all the knowledge. Scripts are good at representing the real word events and applying previous experience to the new situations, however they are doing poorly at describing complex concepts and relationships. Finally, production rules that form the base of a production system are supported by a powerful formalized inference engine and provide a convenient understandable way of a knowledge representation.

To sum up, a precise analysis shows that every approach has its own advantages and limitations, which should be evaluated before an implementation of the intelligent knowledge-based systems. Nowadays a lot of research is conducted in the area of hybrid knowledge representation techniques, which try to combine the advantages of described approaches in a single scheme. In particular, the methods of building and using ontologies are of a great concern. A typical ontology consists of the combination of presented KR techniques, thus supporting the assumption about the usefulness of a hybrid approach to knowledge representation. The research of hybrid models lays the foundation for a development of the reasoning methods that could operate on those models.

An overview of the described knowledge representation models, aims to provide a solid background for the further research of building an intelligent system for personal knowledge management. The way the knowledge is organized in KB, significantly affects the productivity of an inference mechanism. The absence of an ideal solution raises a need to contribute the time and efforts to a development of the hybrid KR techniques. Another important issue to be considered is an interaction of a system and the user. Typically an intelligent knowledge management system would allow user to operate with the knowledge (create, modify, view, delete) and execute some reasoning activity. These functionalities should be as convenient as possible; otherwise the system will be abandoned because of being "too technical" or timeconsuming. Among a variety of supporting tools, the methods of natural language processing (NLP) are of exceptional use. They provide the means for transforming the sentences written in natural language to the structured data. A recent breakthrough in the area of deep learning networks resulted in a significant raise of interest to NLP techniques. In general, those and other challenges form a need for the further research in the selected topic.

*1. Глибовець М. М., Олецький О. В. Штучний інтелект. – К.: КМ Академія, 2002. – 366 с. 2. Sowa J. Semantic networks. – [Електронний ресурс]: http://www. jfsowa.com/pubs/semnet.htm. 3. C. Schank R., Abelson R.. Scripts, Plans, Goals and Understanding: an Inquiry into Human Knowledge Structures. – Lawrence Erlbaum Associates, 1977 – 256pp. 4. Литвин В. В. Задачі оптимізації структури та змісту онтології та методи їх розв'язування / В. В. Литвин // Вісник НУ "Львівська політехніка". Серія: Інформаційні системи та мережі. –2011. – № 715 – С. 203–214. 5. The Univalent Foundations Program. Homotopy Type Theory: Univalent Foundations of Mathematics. – https://homotopytypetheory.org/book, 2013. – 475 p. 6. Brachman R., Levesque H. Knowledge Representation and Reasoning. – Morgan Kaufmann Publishers, 2004. – 381 pp. 7. Väänänen J. Lindström's theorem. – [Електронний ресурс]: http://www.math.helsinki.fi/logic/opetus/lt/lindstrom_ theorem1.pdf. 8. Stanford Encyclopedia of Philosophy. Type Theory. – [Електронний ресурс]: https://plato. stanford.edu/entries/type-theory. 9. Wadler P. Propositions as Types // Communications of the ACM. – 2015. – Vol. 58, No 2. – p. 75–-84. 10. Semantic Network. – [Електронний ресурс]: https://upload.wikimedia.org/wikipedia/commons/6/67/Semantic_Net.svg. 11. McKeown K. Semantic Nets, Frames, World Representation. – [Електронний ресурс]: www1.cs.columbia.edu/~kathy/cs4701/ documents/frames.ppt. 12. Luger G. Artificial Intelligence, 6th Edition. – Pearson, 2008. – 784 p. 13. Durkin J. Expert Systems: Design and Development, 1st edition. – Macmillan Coll Div, 1994. – 800 p. 14. Floridi L. Philosophy of Computing and Information. – Blackwell, 2003. – 388pp. 15. Audi R. The Cambridge Dictionary of Philosophy, 2nd edition. – Cambridge University Press, 1999. – 1039 pp. 16. Литвин В. В. Оцінка новизни знань під час автоматичної розбудови онтологій / В. В. Литвин, А. С. Мельник, В. Я. Крайовський // Вісник НУ "Львівська політехніка". Серія: Інформаційні системи та мережі. – 2011. – № 699 – с. 343-352. 17. Noy N., McGuinness D. Ontology Development 101: A Guide to Creating Your First Ontology. – [Електронний ресурс]: http://protege.stanford.edu/publications/ ontology_development/ontology101-noy-mcguinness.html. 18. Jean S., Pierra G., Ait-Ameur Y. Domain Ontologies: A Database-Oriented Analysis / Proceedings of Web Information Systems and Technologies 2006. – [Електронний ресурс]: http://www.lias-lab.fr/publications/7412/2006-WEBIST-Jean.pdf. 19. Jurisica I., Mylopoulos J., Yu E. Ontologies for Knowledge Management: An Information Systems Perspective. – Knowledge and Information Systems, Volume 6, No. 4. – 2004. – 380-401 pp. 20. Lehmann J., Völker J. An Introduction to Ontology Learning. – [Електронний ресурс]: http://jens-lehmann.org/ files/2014/pol_introduction.pdf. 21. Epstein R. Classical mathematical logic: the semantic foundations of logic. – Princeton University Press, 2006. – 544 p. 22. Harmelen F., Lifschitz V., Porter B. Handbook of Knowledge Representation. – Elsevier Science, 2007. – 1005 p. 23. Russell S., Norvig P. Artificial Intelligence. A Modern Approach. 3rd edition. – Prentice Hall, 2009. – 1152 pp. 24. Keserwani P., Mishra A. Selecting integrated approach for knowledge representation by comparative study of knowledge representation schemes // International Journal of Scientific and Research Publications. – 2013. – Vol. 3, Issue 2. 25. Нікольський Ю. В., Пасічник В. В., Щербина Ю. М. Системи штучного інтелекту. – Львів.: Магнолія-2006, 2010. – 279 с. 26. Stanford Encyclopedia of Philosophy. Church's Type Theory. – [Електронний ресурс]: https://plato.stanford.edu/entries/type-theory-church. 27. Khosravi H. Knowledge Representation using first-order logic. – [Електронний ресурс]: http://www.cs.ubc.ca/~hkhosrav/ai/ slides/chapter8.pdf. 28. Hauskrecht M. Production systems. Frame-based representations. – [Електронний ресурс]: http://people.cs.pitt.edu/~milos/courses/cs2740/Lectures/class11.pdf. 29. Costea I. Scripts. – [Електронний ресурс]: http://www. csun.edu /~icostea/SP08/MSE614/Week_5_Session/ Scripts handout Week5.rtf. 30. Kerber M., Knowledge Representation I. – [Електронний ресурс]: ftp://ftp.cs.bham.ac.uk/pub/ authors/ M. Kerber/ vTeaching/AI/l6.pdf. 31. Baader F., Calvanese D., McGuinness D., Nardi D., Patel-*

*Schneider P. The Description Logic Handbook: Theory, Implementation, and Applications. – Cambridge University Press, 2003. – 505 p. 32. Baral C. Knowledge Representation, Reasoning and Declarative Problem Solving. – Cambridge University Press, 2003. – 544 p. 33. Davies J., Fensel D., Harmelen F. On-To-Knowledge: Content-Driven Knowledge-Management through Evolving Ontologies. – John Wiley & Sons, 2002. – 312 p. 34. Sowa J. Knowledge Representation: Logical, Philosophical, and Computational Foundations. – Blackwell, 1999. – 608 p. 35. Chater N., Oaksford M. The probabilistic mind: Prospects for Bayesian cognitive science. – Oxford University Press, 2008. – 534 p. 36. Tanwar P., Prasad T., Datta K. Hybrid technique for effective knowledge representation and a comparative study. – [Електронний ресурс]: https://arxiv.org/ftp/arxiv/ papers/1209/1209.3869.pdf. 37. Huyck C. Knowledge Representation, Semantic Nets, Frames, Scripts. – [Електронний ресурс]: http://www.cwa.mdx.ac.uk/bis2040/JohnLects/01bkb05p.ppt. 38. Bush V. As We May Think. – Atlantic Monthly, 1945. – 10 p. 39. Davies S., Allen S., Raphaelson J., Meng E., Engleman J., King R., Lewis C. Popcorn: the personal knowledge base / Proceedings of the 6th conference on Designing Interactive systems. – ACM, 2006. – 150–159 p. 40. Munn K., Smith J. Applied Ontology: An Introduction. – Ontos Verlag, 2009. – 342 p.*

**УДК 004.3; 004.7**

**В. Голембо, О. Бочкарьов**
Національний університет “Львівська політехніка”,
кафедра електронних обчислювальних машин

# ПІДХОДИ ДО ПОБУДОВИ КОНЦЕПТУАЛЬНИХ МОДЕЛЕЙ КІБЕРФІЗИЧНИХ СИСТЕМ

**Розглянуто підходи до побудови концептуальних моделей кіберфізичних систем (КФС). Запропоновано узагальнену схему взаємодії КФС з оточенням та визначено набір основних компонент КФС. Розглянуто організацію функціонування компонент КФС та запропоновано відповідну концептуальну модель. Визначено поняття контуру взаємодії КФС з фізичними процесами та запропоновано концептуальну модель контурів взаємодії. Розглянуто проблему узгодження взаємодії КФС з фізичними процесами. Запропоновано концептуальну модель інтегрування компонент КФС.**

**Ключові слова: кіберфізична система, концептуальна модель.**

**Approaches to the construction of conceptual models of cyber-physical systems are considered. A generalized scheme of interaction between the CPS and its environment is proposed. The set of basic CPS components is defined. The organization of CPS components functioning is considered. The corresponding conceptual model is proposed. The definition of the contour of interaction between CPS and physical processes is provided. The conceptual model of the contours of interaction is proposed. The problem of matching CPS interaction with the physical processes is considered. The conceptual model of integration of CPS components is proposed.**

**Key words*:* cyber-physical system, conceptual model.**

## Вступ

Під кіберфізичною системою (КФС) розуміють систему, яка складається з деякої зв’язної множини кібернетичних засобів (управління, обробки, передавання даних та ін.), та взаємодіє з фізичним середовищем за допомогою сенсорних і виконавчих систем за схемою зворотного зв’язку