Graph Representation of Declarative Languages as a Variant of Future Formal Specification Language

Ian ORLOVSKI
Technical University of Moldova, Chisinau, Moldova
orlovski@yandex.com

The paper includes brief review of formal specification languages, requirements and ideas for possible future developments. At the same time focusing attention to the languages based on first-order logic theory. For determining possible evolution directions of the existing specification methods we present generalized set of current and assumed requirements for systems of formal specification. As a potential solution for some lacks elimination concerned with specifications representation, was proposed an approach to graph representation of declarative languages. Was introduced and sight for future of specification languages as a result of current methods integration.
Keywords: *formal specification, graph representation, knowledge representation, formal methods*

1 Existing approaches and languages

Model languages are widely used in a modern systems development life cycle. One of the most known of then is UML language which oriented on system modeling by business-users. Graphical representation is one of the main advantages of UML, but from the other side, this language among others points of criticism has a principal problem consisting in lack of a base formal theory. As an alternate path of evolution, was developed a direction of formal specification languages. These languages allow designing mathematical models of the describing system. In addition of this at the last time was developed an approach of application domain oriented programming (for example DSL). This approach guess for simplify of complex systems developing to write own domain oriented language, which, in particularly, can be a specification language.

Formal specification language consists of three components: syntax, semantics and proof theory (or inference engine). For systems based on formal specifications languages and intended for systems design were determined next purposes: design of requirements, architecture, specifications; verifying of designed model; intercommunication of developers in the design process; reusing of well formulated specification elements and other. Different languages were developed for a broad range of system types. In this paper we narrow research area on software information systems direction, which is actual for programmers, system analyst, testers and other roles included in software development lifecycle.

Exists wide range of theories which were selected as base of specification languages: algebraic (for example CASL, OBJ) or logical approach, set theory, some of the functional languages theory and combination of this theories and approaches.

Settling on logic approach in our research we accepted next formal specification languages:

- RSL (RAISE Specification Language) — based on logic of particular functions,
- VDM – objects are noted by set theory, logic expressions based on first-order logic,
- Z – based on a typed first-order theory and a set theory

At the research of languages designed, or potentially suitable, for the formal specification description that also have a rigorous logical theory, we can distinguish the following groups of languages:

- Languages specially designed for use as formal specification languages, that distinguished by syntax and semantic (due to different logic theories from the subset of first-order logic) and application domain.

According to the previous section this languages are:

- VDM, RSL and Z;
- Declarative programming language – logic of functional-logic languages (like Prolog, Curry, Mercury);
- Conceptual graphs as visual knowledge representation language;
- Family of knowledge representation languages which can be used to represent the concept definitions of an application domain.
- In this article we focus on the formal specification languages applied to modeling of software information systems, consequently all approaches and languages will be reviewed from this point of view.

2 Requirements, evaluation criteria, criticism and request for future systems of formal specifications

Base requirements for modern formal specification languages results from requirements for specifications itself and consists in: unambiguous, completeness of specifications, high level of abstraction (specification must be free of realization algorithms), simplicity of understanding etc.

According to requirements and purpose of formal specification systems can be formulated next evaluation criteria which in details are described in [1]:

- Expressive power and level of coding required — low level of expressiveness requires more skills for knowledge expression;
- Constructability, manageability and evolution;
- Usability of design and presentation;
- Communicability — simplicity of representation;
- Powerful and efficient analysis — as a compromise with language expressiveness. High order logic improve expressive power but leads to potential impossibility of specification processing

Criticism of formal specification languages represent inadequate or poor level of some properties required for users and consist in:

- Limited scope — most of the languages oriented on to the functional description of the system;
- Poor separation of concerns — impossibility to distinguish internal properties of a system, assumptions about environment, properties of the application domain;
- Low-level ontology — excess number of details including low-level description of elements;
- Isolation from interconnected lifecycle studies of from other modules of the system;
- Poor guidance of the user in the process of design;
- Poor tool feedback — system only can establish problem but not suggest any resolutions.

Based on general requirements, listed lacks and problems, wishes for future systems presented in [1], we can pick out items interesting for us in terms of graph representation a possible perspective of research. These are possible requirements for future systems of formal specification:

- Integration of different system lifecycle phases and relation between specifications, reuse of specifications;
- Possibility of abstraction level separation;
- Simplicity of design and representation;
- Different modes of representation (multi-format specification) — graph, table, formal text notation, free text notation, diagrams, schemes etc.

3 Review of existing approaches

Formal specification languages

Most formal specification language based on algebraic model describing a data sets and functions on these sets. We examine modeling oriented languages such VDM, Z, and RSL as the development of VDM language, which has the opportunity to present in the form of another approach based on the properties description. These and similar to these languages usually use the theory of sets, and have a theoretical framework based on semantic of a first-order logic. Interpretation of the designed models

described by these languages can be used for translation to one of the programming languages or for the analysis of these models. VDM is one of the first languages of formal specification. Language focused on representation of discrete data types such as integers, Boolean values, characters, and other sets, including lists. Over these types can be declared functions described in a declarative - a functional style, or in the imperative style, designed to process conditions. For a given language developed many tools that allows to interpret the written specification with the possibility of analysis and verification of designed models, improving the specification, the generation of tests for specification.

Theoretical approach of the Z language is similar to VDM, but more focused on the description of systems based on the states. There were introduced additional expressive abilities for describing the state space and descriptions of operations for allowed in the system.

RSL - combines a paradigm-oriented modeling and description of properties, and allows formulating the specification in different kinds of representation. Specifications can be reused, can be parameterized. [2]

Visual representation in this language group focused on the already existing traditional system models representation. For example, the model in the Z language is a set of states and operations, i.e. transitions between states. Accordingly to this, the visual representation of the model uses existing techniques for system state representation, for example, Petri nets, state diagrams or sequence diagrams.

Thus, the specialized languages of formal specifications narrow the application domain and ways of presenting of system models, which limits the area of their use. At the same time expressive language within the domain can be reasonably high. Visual representation inherited from the already existing methods for representing these class systems.

4 Declarative languages

Declarative programming languages allow describing what is required to get a result without describing how to implement it. Often these requirements apply to the methods of description specifications (see the requirement of abstraction). Declarative languages are divided into two classes - the logical and functional. Functional languages are based on lambda calculus and have the expressiveness of the second-order logic. At the same time, using all the expressive features of language leads to complication of programs for understanding, as it is criticized, for example, in [7]. Theory of definitional programming, as one of the results of further development of declarative languages, and presented in [7], today hasn't widespread and there were no practical applications. Based on the foregoing, as expressive, but easily understandable language we choose the class of logic and functional-logic languages.

Prolog is the classic of logic programming has received not only academicals status language, but also as widely used in the development of real applications. Expressiveness of the language is limited to Horn clauses of first-order predicate logic. Improving the expressiveness of the logic elements of the second order is possible only by the addition of functional features that automatically leads to a new language of the category of functional-logic languages that are described below. Formulas of propositional logic that used in the literature in analyzing the mechanism of Prolog often are visually presented in the form of a decision tree. In addition, exist some attempts to the visual presentation of predicate logic. The proposed methods of visual presentation were not received distribution. At the same time the popularity of Prolog as a language available to ordinary business users (i.e. non-programmers and non-specialists in formal systems) after the project of 5-th generation computers is practically extinct, so the new practical developments in this area is not maintained. In this paper we consider the possibility of

developing graph representation of Horn clauses, i.e. in fact expressions of Prolog programs.
Functional-logical line is represented by Curry and Mercury languages. These languages have essentially two different approaches to the same problem. Curry is the development of one of the most powerful functional language Haskell, obtained by adding the logical properties. Mercury is the opposite side and is the development of logic programming language Prolog by adding functional features. The resulting «fusion» increases the expressiveness of the language at the expense of functional properties and at the same time has a descriptive, more accessible to understanding properties of predicate logic.
Ability to compile a program implies that presented in the language of declarative formal description of the model should have some resulting executable functions. Clearly, model specification, in most cases should not have similar functions. Contrary to the requirement of a high level of abstraction implies the absence of such low-level descriptions. Such elements may appear when you create a test function to test the constructed model. Ability to create mechanisms for verification, and automated refactoring to improve the description of the model implies the presence of an executable mechanism. Part of these functions implements a parser development environment or compiler system. Specialized semantic analyzers, despite their theoretical possibility, as we know at the moment are not created.

5 Conceptual graphs

Conceptual graph is a directed graph in which vertices are divided into two types of concepts and relationships between concepts and edges determine the connection between objects. The fundamental difference from the classical ideas of semantic networks consists in the transition of relations in a certain type of nodes. In this case, the connection only indicate the procedure of connection the various nodes into the one formula. The theory of conceptual graphs together various research relations in semantic networks and explicitly declares a set of possible types of relationships. Language expressiveness is limited to the classical first order logic. The advantage of formalism over other methods of knowledge representation is based on the «integrated» model visualization.
Like any formalism of knowledge representation conceptual graphs are intended to describe the high level of abstraction. Presentation of the model will be available for simple «nontechnical users», the graphical representation will improve the ergonomics of the description of the formal specification.

6 Description logics

Description logic is the general name for the languages of knowledge representation based on the idea of concepts and relations between them and their use for describing the application domain. As formal description languages may be used as the formal specification languages (i.e. for the description of specifications).
Description logics theory is the result of consolidation and development of the two paradigms of artificial intelligence domain that are semantic networks and frames. The theory has three ideas [5]:

- Basic syntactic elements may be of three types - concepts, roles and individuals, represented by the next logical elements respectively - unary predicates, binary predicates and constants.
- Expressiveness language is limited in the sense that the complex structures are built from small number of constructors
- Implicit knowledge of the concepts can be automatically derived using inference rules

One of the modern applications of the description logics is the ontology language for the Internet - Web Ontology Language, to be exact its subclass OWL-DL. The principal difference between the languages of classical logic languages (e.g. Prolog) consists in using a hypothesis open world, i.e. if the truth of a proposition cannot be proved, then

the assertion is true, in contrast to the hypothesis of a closed world, where the same assertion is false, respectively. The syntax of OWL is built on the XML markup language, and therefore it can be useful as a universal language for knowledge translation between different systems.

As one of the directions for the language ergonomics increase, community developed methods of visual representation, the so-called Visual OWL. Internet community provides review of visual presentations of requirements for information systems.

Existing development in the domain of visualization OWL language consist in tree representation of XML markup language, or a graph similar to the classical semantic networks.

7 First-order logic graph representation as an evolution direction of formal specification languages

From our point of view the problems associated with the requirements of the comfortable, user-friendly, multi-view presentation, as well as the problem of the user's control during the developing process are associated with a formalized graphic representation.

A variant of graph representation was presented in [3]. The proposed approach first of all provides an abstracted, free from implicit features of the text, where each element of syntax is a separate element of the graph.

Remind that the dictionary of the graph language was defined by following sets of nodes and edges

V = {N, E}, where:

N = {F, P, O, Q} (functional constants, predicate constants, logical operators, quantifiers);

E = {C, N, T} (connectivity edge, negation of the elements connectivity edge, term connectivity edge);

O = {AND, OR};

Q = {EMPTY, Existence, Universal};

F, P, T — are named by the corresponding formula elements.

Syntax was defined by the following set:

V_R = {(F, T, F), (F, T, P), (P, C, O), (P, N, O), (O, C, O), (O, N, O), (Q, T, F), (Q, T, P)},

where the structure in the parentheses consist of three relation elements — start node, relation edge, finish node.

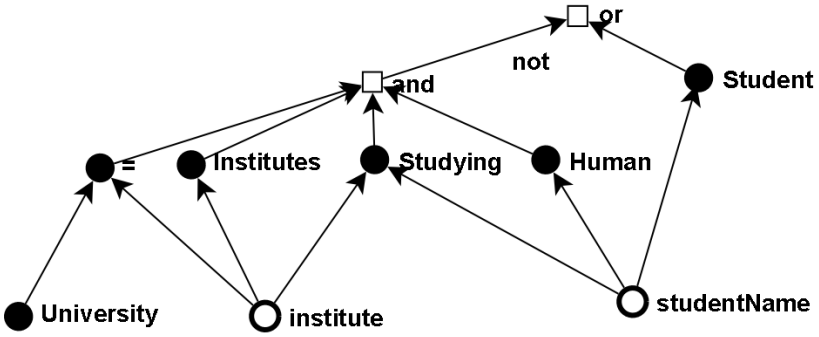

Fig. 1. Graph representation of IsStudent predicate

This graph representation gives easy possibility to trace interconnections between elements of logic formula. In example represented on figure 1 (this example not pretends to the sense presence) the connection element **institute** (a variable) is presented only once and with a clarity shows relation between the objects **Studying**, **Institutes** and **=**.

For example we present in the form of graph representation next formula of the first-order logic (presume that predicates Human, Studying, Institutes are already declared separately):

IF Human(studentName) AND Studying(studentName, institute) AND =(studentName, institute)

THEN IsStudent(studentName)

or in Prolog notation:

student(StudentName):-

human(StudentName),
studying(StudentName, Institute),
Institute = «University».

8 Evolution of first-order logic graph representation

Was developed a variant of models representation presented in the form based on Horn sentences and syntax of Prolog language.
In this variant the graph V is defined by the following elements:
N = {L, F, P, O, Q},
clause (positive part of the Horn clause), functional constants, predicate constants, logical operators, quantifiers,
E = {C, N, T},
connectivity edge, negation of the elements connectivity edge, term connectivity edge
O = {AND, OR}
Q = {EMPTY},
in this set, as in the Prolog language, presents only empty quantifier due to the existing in Prolog language notion of the variable domain
L, F, P, T — are named by the corresponding formula elements
A set of the graph syntax from the previous section was complemented by the three new elements describing relations with the new node type «clause» denoted by L:
V_R = {(F, T, L), (O, T, L), (Q, T, L), (F, T, F), (F, T, P), (P, C, O), (P, N, O), (O, C, O), (O, N, O), (Q, T, F), (Q, T, P)},
Hence, the example above can be represented by the following graph on the figure 2.

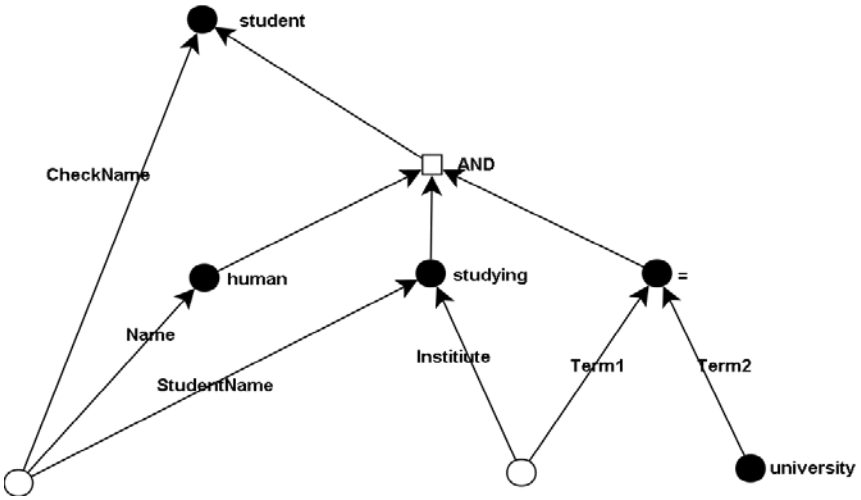

Fig. 2. Prolog graph representation of IsStudent predicate(complex)

In Prolog this phrase can be simplified by the next way:
student(StudentName):-
human(StudentName),
studying(StudentName, university).
According to this we can also simplify the graph representation in the figure 3.
To understand the fundamental differences with the textual notation we pay attention to the edges names of Term type (detailed description see in [3]):
1. Name of such edges is the necessarily property because the edge type «Term» is determined only by the presence of the title. In this way, also, is determined relation to the concrete argument of the predicate (as opposed to the argument order in the text written in Prolog program)
2. Name of the argument in the predicate «student» differs from the argument name of predicates «human» and «studying». This is done especially to show that the logic relation between two formula elements (arguments of different predicates) is defined by the connection to the same quantifier, not by the same variable name.

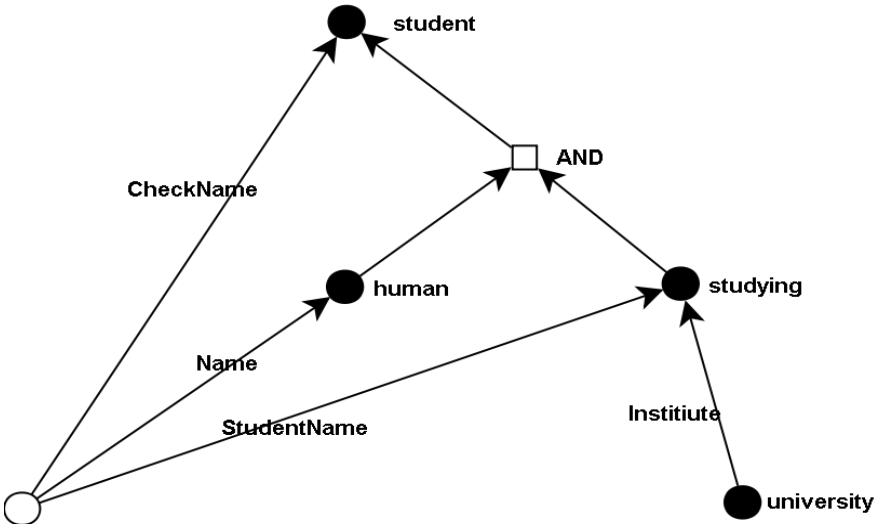

Fig. 3. Prolog graph representation of IsStudent predicate (simplified)

The proposed approach can and should evolve towards a more expressive language, as well as in the area of compatibility with existing programming languages that have effective interpretation tools. Research in the area of functional-logic graph representation will allow getting a graphical language with second-order logic features.

9 Comparison of formal specification methods approaches and perspective of their integration

Were compared the methods of describing the formal specifications in the context of the requirements and criteria that are defined in the corresponding part of the article.

Languages of formal specification through its specialization have high expressiveness in the selected area aimed at certain classes of systems or descriptions of the methodology (targeted at the system properties or description of its states). At the same time it increases the demands on the user's knowledge, and specialization of language restricts its ability to express knowledge about the describing system. Development environments or additional tools provide abilities for the analysis of the designed models, their improvement and verify. Methods graphical representation of models even more specialized and are essentially the adaptation of well-known types of diagrams for a particular language. The developed models are usually in too much detail and focuses on implementation, rather than declarative description of the information system.

Expressive representation using declarative programming languages affects the accessibility, i.e. understanding of the designed models and more dependent on the user, which is not limited to methods of formal specification languages. At the same time, the wide used specialized methodology can be realized as a certain kind of pattern, it will simplify the building process of special domain-oriented models. For the analysis of the developed model, the user must develop mechanisms to describe its validation and testing, which will be performed in the executable program, although the simplest errors can be identified by a development environment. Graphical representation of declarative languages in spite of some existing development is not widely applied. This class of programming languages is the closest to natural language therefore available to study a wide range of users, so that the creation of affordable and easy to use graphical representations and development environment allows using these languages to describe systems at different levels of abstraction and use design description on various stages of software lifecycle.

The expressiveness of conceptual graphs is limited to the expressiveness of first order predicate logic. Graph representation in our opinion is not enough abstract, as was

mentioned in [3] although it can be useful for knowledge representation in general and as a first, i.e. an intermediate, level for computer representation of phrases in natural language. Languages of the description logics family that are oriented on the description of properties of the system have different levels of abstraction. W3C Community today still being developed graphical representation which can be declared as a standard. With representation in XML, a dialect of OWL can be a unified interchange format for exchanging data between different development systems. Additionally, this area is dynamically developing and in the closest future there may be the new practical results in this direction

10 Idea of approaches integration

Unifying the different advantages of the described methods, based on graph representations of functional-logic languages, we can formulate the properties of proposed future system, or some kind of framework, which allow different kinds of users the following features:

- expressive declarative language that allows to develop a formal description of different interconnected levels of abstraction (Curry or Mercury for example);
- visual representation based on graph representation of declarative language;
- variants of graph syntax, focused on different application domains and phases of design (ideally, according the ideology of domain specific languages variants of syntax can be designed for every complex project). Certain for different views will be necessary to design also the mechanism of translation from one representation to any other necessary format. As the standard alternative format can be, for example: the documentation (textual or algorithm diagram representation of requirements), specification (representation in the form of block diagrams), interface (the relationship of elements of input / output), data structure (relational database diagram), etc.;
- as a consequence of the preceding paragraph availability for use by different categories of users in the process of software developing using translation mechanisms into a variety of representations (specialized or vice versa general form in the language of conceptual graphs);
- development environment with properties of online control of the syntax of designed specifications, possible semantic analysis of the contradictions and redundancy;
- use of related or external specifications, in the form of black boxes and called as existing propositions (may be described by the predicate);
- intersystem interactions based on developed standards for knowledge representation (e.g. based on XML and OWL).

11 Conclusion

The result of analysis of different approaches to the formal representation of specifications can be defined as necessity to establish methods presentation and writing of specifications includes the following features implemented in different approaches:

- accessibility to the user based on the functional-logical representation of knowledge
- expressiveness of the language on the level of second order logic (based on properties of functional languages)
- the possibility of automated analysis, conversion and translation into other languages developed a formal model
- ergonomic presentation of the model based on graphs and their graphical realization
- ability to describe different levels of abstraction
- formal unified format for interchange between different development systems

It was presented the idea of consolidation different approaches to obtain variant which will have distinctive positive features of different formal specification approaches.

A variant of graph representation logic programming language Prolog phrases, based on Horn clauses, was proposed.

Proposed to research possibility of:

- creating an integrated approach based on proposed graph representation extending it to the class of functional-logic languages;
- defining in the developed system a method of translation domain oriented description exploring the existing patterns and approaches;
- defining the interpretation tools for verification and analyzing the designed specifications.

References

[1] A. Van Lamsweerde, "Formal Specification: a Roadmap", *Département d'Ingénierie Informatique*, Université catholique de Louvain.

[2] D. Bjørner and M. C. Henson, *Logics of Specification Languages*, Springer, 2008.

[3] I. Orlovski, N. Pelin and A. Miron, "A Variant of Vocabulary and Syntax of Graphical Representation Method of First Order Predicate Logic Formulas", in *Economy Informatics*, Vol VIII, No. 1, 2008.

[4] D. Partridge, *Artificial Intelligence and Software Engineering: Understanding the Promise of the Future*, AMACOM, 1998.

[5] F. Baader, *The Description Logic Handbook*, Cambridge University Press, 2003.

[6] N. Duglas, T. Viguer, N. Leveson and M. A. Storey, "On the use of visualization in formal requirements specification," *Proceedings of the 10th Anniversary IEEE Joint International Conference on Requirements Engineering*, 2002, pp.71-80.

[6] O. Torgersson, "A Note on Declarative Programming Paradigms and the Future of Definitional Programming," *Proceedings of Das Wintermote 96*, 1996.

Ian ORLOVSKI has a master of science in Information Systems. Today is a graduate student at the Technical University of Moldova. Research in the domain of logic programming, knowledge representation and application of artificial intelligence methods since year 2000.