About possible methodology of information systems modeling using graph logic language

Ian ORLOVSKI
Technical University of Moldova
ian.orlovski@gmail.com

***Abstract* —In the article will be examined variant of application graph logic language for modeling information systems. With this purpose after comparison of some existing methodologies will be proposed ways for language use, possible adaptation and extension. Proposed language evolution provides possibilities to represent models with expressive power comparable or even more expressive than existing modeling methods. For comparative examination were selected methodologies of IDEF family, UML and Z languages**

***Index Terms* — logical representation, formal specification, information system modeling**

I. INTRODUCTION

Information system design process includes different type models development, including such models as: information flow model, functional model, data model and others. In this article will examine some model representation methods used in different modeling methodologies. After that will be made analysis of ways to realize this methods on graph logic language developed by the author.

Graph language for first order logic Horn clauses representation was presented by the author in [1]. Purpose of developed language consist in visual representation of first-order logic formulas, saving formal semantics and with possibility to translate graph language phrases to Prolog language. Due to possibility of representation models without rules of execution sequence, translation of such models need to be used parallel version of Prolog language.

Language is declared by vocabulary V={N, E}. N = {F, P, O, Q} is a set of nodes and consist of functional and predicate constants, logic operations and quantifiers. E = {C, N, T} and represents connection, negative connection, and term connection edges. O={AND, OR} is a set of possible logic operations. Q={Empty, Existence, Universal} is a set of possible quantifiers.

Language constructions consist of vocabulary elements and relations in the form of (nodeFromType edgeType nodeToType) and can be represented by the next statements:

terms t ={Q, F, (t T F)}
predicate form fp = {P, (t T P)}
formula f = {fp, (fp C O), (fp N O)}
Horn clause fh = {fp, (f C fp), (f N fp)}

In this article, after presentation of models in existing methodologies will be analyzed variants for using graph language for presented models types. As an abstract problem for information system modeling will be used problem of students accounting. Some aspects of this problem will be used as simple examples for models representation.

Major examples will be presented in Prolog, with comments for visual representation of particularities. Also some examples will be represented on figures with possibility to make comparison with existing methods.

II. PENDING MODELING METHODOLOGIES

For information systems modeling and software development were proposed a variety of different approaches and methodologies. Will examine some well-known approaches, including those that have visualization possibilities.

IDEF is a family of methodologies for modeling complex information systems. Major part of this methodologies based on visual syntax for model representation. This family includes: IDEF0 is standard for functional modeling, IDEF1 is standard for information flow modeling based on ER (Entity-Relation) models, and IDEF1X is an extension of IDEF1 for data structure modeling. Syntax of IDEF0 and IDEF1 is quite different, therefore different visions of information system are independent one from another so user need to study syntax of many different languages.

Rational Unified Process software development framework applicable for whole software lifecycle, an using Unified Modeling Language for development vision, structure, and behavior models of information system [2]. UML has visual syntax but doesn't has formal semantics for syntactic constructions. As a partial solution of this disadvantage in second version of the language was introduced OCL symbolic language for declaring formal semantics applicable for whole model.

Contrary for visual languages exists symbolic (textual) languages for formal specifications. Z is one of well-known languages of such type. This language was selected by many researches as formal base for complex languages that have visual part (for example UML) and formal part using Z-language[3]. For example Alloy is one of this complex languages. Z language bases on first order predicate logic and theory of sets.

All presented methodologies have different problems for integration of different model types. Graph language presented by the author can be used as unified language for information system model declaration. Examination of

such possibility will be made by overlaying existing methodologies on proposed graph language. Identifying problems and lacks indicates direction for language adaptation, modification and evolution.

III. REPRESENTATION OF DATA FLOW MODEL

Data flow model is the simple model for conceptual description of a system. At this level need to be defined substantive processes and information flows between them. This diagram can be extended on other levels of development.

In IDEF family vision of the system can be represented by the IDEF0 diagram or by the IDEF3 diagrams. Detailed description of IDEF0 will be presented below in functional modeling part. Basic principle of IDEF0 from architectural point consist of idea that every function can be decomposed on the set of connected child functions which have input and output of the parent function. IDEF3 in process description uses logic operations such as AND, OR, XOR etc. for defining structure of object interconnection.

In UML high level model of the system can be represented by the component diagram. Components are represented by the rectangle shapes but edges define connection between them. Also this type of diagram has possibility to define interfaces between components.

In Z language schemes represent objects, but relations between objects can be interpreted as simple data flow model. Scope of this language to define formal specifications but not architecture of the system.

In proposed graph logic language system can be represented as a set of predicates that denote processes and quantified variables that connect this processes and can be interpreted as data flow. Also predicates are connected by the logic operators such as OR, AND. If necessary a set of logic operators can be extended with some more complex operators, for example XOR. As an example of a system can be presented process of student registration. Student as a person have personal data, system for registration need to check existence of such person in the database. If person doesn't have record in database occurs person registration and after that registration this person as a student. In Prolog language this discourse can be represented in the next form:

```
studentAdd(PersData, StudentID):-
        ((personNotExists(PersData),
          personAdd(PersData,PersonID))
        ;personExists(PersData, PersonID)),
        studentRegistration(PersonID, StudentID).
```

This clause can be interpreted as a simple data flow diagram with basic blocks presented by predicates and variables as data connections between them. Logic operations also can be used similarly with IDEF3 notation.

IV. REPRESENTATION OF FUNCTIONAL MODEL

IDEF0 approach suggests decomposition of the system on functions. Every function can be connected to other by the one of four types: input, output, control and mechanism. Directed connection edges define order of the processes. Every function can be decomposed on the set of child functions. An example of alternative student registration process is presented on the fig.1.

Fig. 1 IDEF0 representation of student registration process

In UML for function modeling Use-Case diagrams are proposed. This diagram describes all participants of the process (actors) and their roles. Processes can be inherited from the other processes, for example include or extend them. Inheritance type is defined by connection type (include, extend).

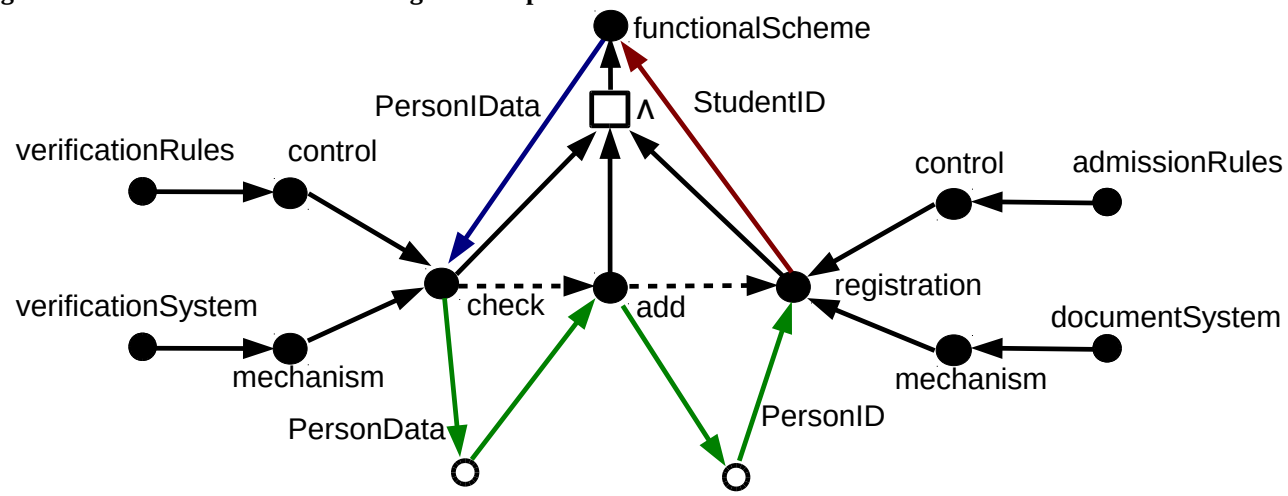
Functional model in Z language is defined by the set of schemas in which exists reference on objects [4]. So functional model can be restored after searching of all schemes referenced to the same objects.

In graph logic language function model can be extended from the ideas presented in previous paragraph. For mechanism and control flows we need define logic functions which we connect to the predicates. Mechanism and control entities are denoted by the functional constants.

Also we need to resolve problem of flow direction. In standard Prolog language direction is defined by the direction of reading, but in parallel versions sequence of execution can't be defined. Since proposed graph logic language has properties of parallel Prolog [5], we need to implicitly declare sequence of processes execution. Another requirement consists in objects flows direction.

For direction indication possibilities we need to introduce some syntax extensions and changes. First, new edge type will define sequence of the execution of predicates of one nesting level. Level of nesting is defined by the number of logic operators from the head of the Horn clause to the examined predicate. Second, direction of the term connection edge can be changeable (in contrast of current syntax where direction is always from the term) or even edge can be undirected in the case of undefined direction. This change need to change definition of the language from directed graph to the mixed graph. For comparison with IDEF0 representation of student registration process on Fig.2 presented version in graph logic language with described changes. Execution sequence edges are denoted by the dot line. Using of quantifiers as data flows are controlled by the directed term edges. In comparison with IDEF0 in model can be uses logic operators for define algorithm for system functions interconnection.

Fig. 2 Functional scheme of student registration process



V. Representation of Data Model

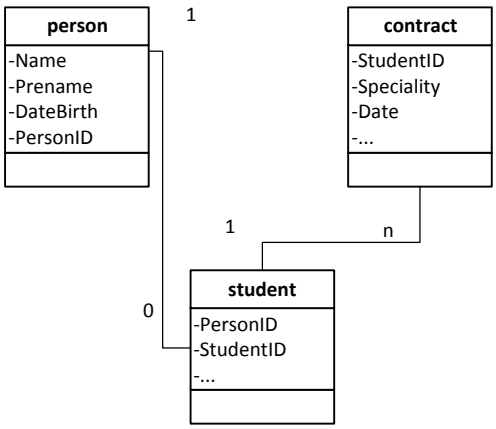
For conceptual data scheme description widely use ER-models (Entity-Relations) and ER-diagrams for their visualization. This diagram defines a set of logic entities, internal structure of this entities and relations between entities. Relation defines not only fact of the connection between entities, also contains information about cardinality of this connection (for example one-to-one, optional one-to-one, one-to-many etc.). Structure of the entity is defined by the set of its attributes.

IDEFX1 standard extends ER model from the one side and extends syntax of IDEF1 standard from the other side. In IDEFX1 has been added next possibilities: declaration of attributes data type, definition of entities identifiers and identifiers of referenced attributes. Thus described extensions leads IDEF1X definition to the physic model of relational database.

Class diagrams of UML have more important role, but not only system structure definition. Class definition defines attributes, their datatype, connectors defines cardinality of relations between classes. Moreover connections in class diagrams defines levels of inheritance between classes. Also class definition contains methods of its use. This approach tends from purpose for declaration models closest to the program code. So expressive power for defining data structure is equal in IDEFX1 and UML languages.

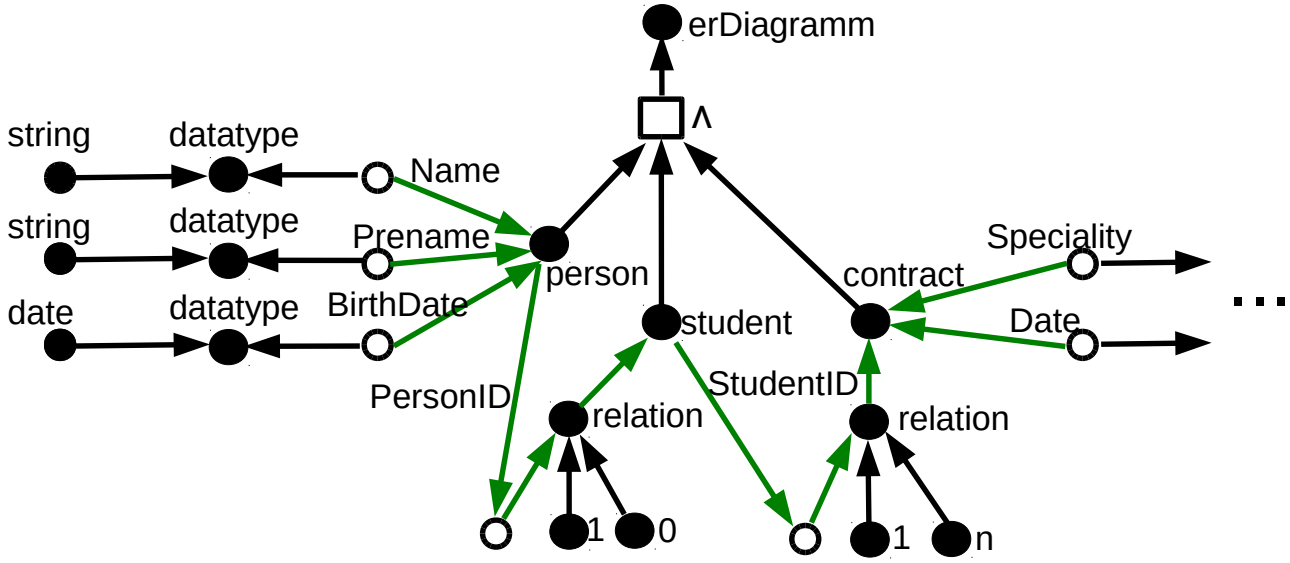
Using Z language entity can be described as scheme with attributes and their data types. Relations between entities can be defined by set relations, with matching between relation types in UML and Z [3]. Further relations between schemes are allocated in special section of specification. From user point of view this feature of symbolic languages tends to loss of connection between entity declaration and relations between them, while in visual methodologies presented before relations representation is base part of diagram syntax. An example of UML version of ER diagram presented on the Fig. 3

Fig. 3 UML representation of ER model



Simple ER diagram in graph language can be represented as a set of predicates which represents relations between terms which represent entities in the form of functional constants. But for an adequate description of data model we need to extend graph language and to introduce ways for description for next features: entities attributes including attributes data types, relations cardinality properties. With this approach predicates represents entities, quantifiers connect functions named “datatype” with entity predicates. Next will introduce function “relation” which has quantifier for entity identifier on input and cardinality properties of input and output. This function connects to predicate which represent entity to which connected this relation. Moreover this function can be named with any identifier for declaring relation between entities of such type. Example of student ER diagram is represented on Fig. 4.

Fig. 4 Representation of ER diagram in graph logic language



VI. CONCLUSION

In this article has been reviewed existing methodologies for some types of information systems models. After comparative analysis of different representation variants were selected those most expressive, which have been used as a base for adaptation and extension of graph logic language for modeling purpose.

Was proposed introduction of execution direction (sequence) edge between predicates of same level of nesting. Also has been changes the principle of term edge direction. Currently this edge indicates data flow direction (flow pattern) and even can be undirected, leads to change the type of graph from directed to the mixed graph.

For possibility of description of some model features was proposed to introduce specific functions, for example data types declaration. This principle can be interpreted as a substantive part of methodology which give possibilities to extend language with ease for it application in different modeling processes.

Distinctive property of using one language for different model types gives advantages for users that deals with different models in process of development. They need to know only one language with special predicates or functions for different variants of modeling.

REFERENCES

[1] I. Orlovski, A Variant of Vocabulary and Syntax of Graphical Representation Method of First Order Predicate Logic Formulas, Economy Informatics, Vol VIII, No. 1, 2008

[2] Noran O. Business modeling UML vs. IDEF.: Griffith.: Griffith University, 2002. 53 p.

[3] L.E.G. Martins, An Empirical Study Using Z and UML for the Requirements Specification of an Information System, In: EMPIRICAL SOFTWARE ENGINEERING LATIN AMERICAN WORKSHOP (ESELAW), Uberlândia. Proceedings of the 2nd Empirical Software Engineering Latin American Workshop. 2005.

[4] J. M. Spivey, The Z Notation: A Reference Manual, Oriel College, Oxford, England, 1998

[5] Орловский Я.С., К вопросу о разработке и реализации формального графового языка для представления логических моделей, DSMSI-2011, Abstracts of conference reports, Kyiv, Ukraine, May 25-27, 2011