Basic constructions of the computational model of support for access operations to the semantic network in the field of implementation of the best available technologies

Viacheslav E. Wolfengagen[1], Larisa Yu.Ismailova[1], Sergey V.Kosikov[2]

[1] National Research Nuclear University "MEPhI"
(Moscow Engineering Physics Institute),
Moscow, 115409 RF
{jir.vew@gmail.com, lyu.ismailova@gmail.com}
[2] Institute for Contemporary Education "JurInfoR-MGU",
Moscow, 119435 RF,
kosikov.s.v@gmail.com

The paper considers the approach to solving the task of storing data in the Web environment using semantic networks (SN). The control over the access to SN is identified as a critical task. An approach to the solution based on the use of the controlling SN is proposed. The rationale for the approach involves developing a computational model for supporting the access operations. The construction of a model based on intensional logic is proposed. The basic logical constructions, necessary for building a model, are considered. The testing of the model's constructions was performed when building the tools of semantic support for the implementation of the best available technologies (BAT).

Keywords: informational objects, semantics, computational model, semantic network, intensional logic, access operation

1. Introduction

As Web technologies are progressing [11], the task of developing the tools of organizing and storing data in the Web environment becomes increasingly significant. In the pre-network single-user environment, the predominant way to organize data was a file structure organized hierarchically. Although this method was convenient not for all tasks (in particular, it did not directly provide the possibility of simultaneous classification of various information objects according to different grounds), on the whole it was fairly convenient.

The situation changed with the arrival of network technologies [1,12], when information objects appeared to be assigned to different users, who when hosting, searching and processing exploited, in general, different principles [2]. The rigidly defined hierarchies, thus, are replaced by the network structures that determine the links of information resources, set in various ways. The links can occur on different grounds, which implies the need to process them in various ways, or to take into account the meaning of the data and their links during processing. Thus, the data acquire a semantic character. As a semantic measurement of data structures appears it leads to the need for a corresponding change in the methods for describing them, searching for in the environment of such structures and manipulating them.

One of the most important tasks in the processing of semantically oriented data is the maintaining of semantics in the course of working with resources. The resources, on the one hand, exist for a long time, which provides the possibility of their multiple use, on the other hand, they usually have a dynamic character, i.e., they can be modified, updated, etc. In the meantime it is possible to change both the data on individual facts, processes, and so on in the area, described by the considered set of resources, and general semantic characteristics of the data. It is also possible to change both the data itself and the links, i.e., the dynamic appearance of new links, a change in the semantics of existing links and other information objects.

Changes in semantics can both maintain the logical continuity of the network, and violate it. By logical continuity in this paper we mean the preservation of a set of general constraints (including informal ones) imposed on the network contents. The maintenance of logical continuity during the modification of the network requires dynamic verification of constraints when performing operations that change the network semantics in order to prevent actions destroying the semantics.

The essential fact when supporting the changes is that semantics can be changed both accidentally during work and maliciously. In the latter case the goal may be to gain unauthorized access to information – obtaining or modification (network espionage or fraud), hooliganism (network vandalism or "trolling"), etc. This is why supporting the semantic integrity of the network suggests, in particular, restricting accessibility to the information in the network [4]. Such a restriction may include both usual restrictions for recording or reading and also more complicated semantically motivated constructions, for example, the ability to change data in a network only in a strictly defined way, or to associate with each change the information, identifying the subject, who made the changes.

The development of support tools for network access operations involves the description of the semantics of a system of interconnected resources in a formalized manner, which brings to the need to introduce the concept of the semantic network (SN) as a formal analogue of the resource system [3, 13]. The capabilities of the network access tools and their constraints are expressed in the form of a model of access tools to the SN [8]. The need to implement support tools for access operations leads to the fact that the model should be computational in nature. The development of such a model as expected is to provide the possibility of constructing a semantically correct SN access support system, including the ability to set semantically coordinated access restrictions; and the latter makes the task of developing a computational model of support to access operations of current interest.

2. Task to develop a model of supporting the access operations

The organizational questions of access to semantic network constructions, in particular, support for access restrictions based on the identification of users (user classes) with different access rights have been intensively studied in various aspects.

The need to support semantic integrity involves researching new models for presenting the network information and access to it. The resulting model should ensure the access rights to the objects of the network from the point of the semantic relationships between them. Such a computational model potentially covers all the elements of the semantic network, and the appearing possibilities of manipulating the network can cause a violation of the information model consistency and, as a consequence, the destruction of the network integrity. To overcome this complexity it is required to restrict access to the semantic network in some regular way.

This paper proposes a solution for control access to the SN, based on the introduction of a metanet, which is a control semantic network. The proposed solution ensures homogeneity of representation of the problem-oriented and control structures of the SN, which reduces the scope of the software code of the supporting mechanisms and facilitates its maintenance (in particular, verification of correctness). The proposed solution has not been previously met in the literature and presents an element of novelty.

The study of the characteristics necessary to display the behavioural feature of the access tools to the SN, which are essential for solving the problem of describing access restrictions, leads to setting the task of developing a model of supporting access operations to the SN, which provides the following possibilities:

- modelling the behaviour of various subjects, interacting with the SN, through the mechanisms of reference management [10];
- modelling the dynamics of interaction through the mechanisms of tracing the objects changes history [5];
- description of sets of access tools and their limitations for specific subjects in the form of specialized information objects - access policies to the system [12];
- providing the capability of specifying access operations and describing their execution.

An essential feature of the model ensuring the achievement of the planned characteristics is the possibility of its integration with the applicative computing system. In the theoretical plan to achieve the integration a method is proposed based on nesting lambda-calculus constructions into the supporting formal system, and in practical terms - on the basis of immersing the model implementation into the applicative environment. The integration of the access support model with the application environment is also an element of the novelty of the model being developed.

3. Intensional logic – possibilities to use for supporting access operations to the SN

Solving the problem of developing a computational model for supporting access operations to the SN requires the use of advanced mathematical methods [13]. The necessity to take into account the dynamic character of the appearing constraints and their dependence on the subject causes the possibility of using methods of nonclassical logics, in particular, intensional logic. Let us consider the necessary logical constructions.

Intensional notions were discussed in logic for a long time beginning with Aristotle, but the first practical semantically oriented system was built by R. Montague [6, 7]. We generally follow his approach but use notions more appropriate for our task.

We use different logical means for treating the dependency of the interpretation of expressions on parameters and for abstractions. According to D.Scott [9] we consider the set of parameters (subjects in the problem domain, moments of time etc.) as assignment points. Each assignment point accumulates the values of the parameters relative to considering situation. The set of assignment points is denoted Asg. We consider assignment points as (abstract) indices and the interpreration of particular expressions can depend on these indices.

We propose a language L_0 for the description of the expressions denoting the access to the semantic network for a concrete subject. This language contains operators for control the access for different assignment points but does not contain means for abstraction on assignment points. Here is the strict definition.

The symbols of L_0 are contained in the following syntactical categories:

1. individual variables. We assume that all variables can be arranged in the sequence and refer to the number of the variable in this sequence as a

natural number of the variable. We denote variables by v, w, x, y, z, possibly with index;
2. individual constants. We denote them by a, b, c, d, possibly with index. In the examples we can use for the constants the unique names, which technically are not the constants but their denotations. But we can build the construction of the language by its denotation in a unique way, so this does not lead to the problems;
3. predicate constants. We denote them by P, Q, R, possibly with index;
4. logical constants ~, &, V, →, A, E, =, Ex;
5. syntactical delimiters: brackets and commas;
6. operators: N, possibly with index.

Individuals in the problem domain are considered as possible individuals. We write Ex(x) for the fact that x exists or x is an actual individual.

Operators in L_0 are considered as unary, so they make a formula Nφ from a formula φ. Operators with the different arity are not considered for simplicity.

Terms of L_0 are defined as individual constants or individual variables. Formulae of L_0 are defined as follows.

1. if $t_0, ..., t_{n-1}$ are terms and P is a n-ary predicate then $P(t_0, ..., t_{n-1})$ is a formula;
2. if s, t are terms then s = t is a formula;
3. if t is a term then Ex(t) is a formula;
4. if φ and ψ are formulae then ~φ, φ & ψ, φ V ψ, φ → ψ, Ax.φ, Ex.φ are formulae;
5. if φ is a formula and N is an operator then Nφ is a formula;
6. there are no other formulae.

Now we define an interpretation for the language L_0. We already have defined Asg as a set of assignment points. The interpretation distinguishes the intensional of the expression, which represents formally the sense of the expression, and the extensional of the expression, which represents its value corresponding to the assignment point.

The interpretation will be defined as follows. For every k ϵ Asg we define the set A_k of objects which we consider as exisiting or actual objects in k. For every individual or predicate constant we define its intensional. The intensional is a function with the domain Asg, so we define the extensional of the constant in k ϵ Asg as the value of this function on k. For the intepretation of the operators we assign to each operator the relation between assingment points and sets of assignment points.

We introduce now some auxiliary notions. The set of thuth values is denoted B. The type of the functions from the set S to the set T is denoted as S → T. We assume that functional types are right associative. We consider a relation on $U_0, ..., U_{n-1}$ as function $U_0 \times ... \times U_{n-1} \to B$ and denote its type as $<U_0, ..., U_{n-1}>$. We consider a predicate on $U_0, ..., U_{n-1}$ as function $Asg \to (U_0 \times ... \times U_{n-1} \to B)$ and denote its type as $<Asg; U_0, ..., U_{n-1}>$.

The important special case of these notion is n=1. Then $<U_0>$-relation is a function $U_0 \to B$ and corresponds to the subset of U_0. The $<Asg; U_0>$-predicate is a function $Asg \to (U_0 \to B)$ and corresponds to the property of the elements of U_0 depending on the assignment point.

The another important case is n=0. Then <>-relation is a function 1 → B where 1 is a cartesian product of the empty sequence of sets. So such relations corresponds to the elements of B and can be considered as logical values. The <Asg>-predicate is a function Asg → (1 → B) which is isomorphic to Asg → B. Such functions can be considered as judgments whose truth value depends on the assignment point.

Now we are ready to the formal definition. The interpretation of L_0 is a triple I = <A, F, R> where

1. A is a function. The domain of A is denoted Asg. The value of A on each argument k ϵ Asg the value A(k) (or A_k) is a set. We denote the union of A_k for all k ϵ Asg as D;
2. F is a function defined on the set of individual and predicate constants of L_0. If c is an individual constant of L_0, then F(c) is a function Asg → D. If P is n-ary predicate constant of L_0, then F(p) is n-ary <Asg; D^n>-predicate;
3. R is a function defined on the set of operators of L_0. If N is an operator of L_0, then R(N) is <Asg, P(Asg)>-relation, where P(Asg) is a set of all subsets of Asg.

In this formal definition we consider a domain of A as a set of assignment points for I. For k ϵ Asg we consider A(k) as a set of actual individuals in k, and D as a set of possible individuals. The individual constant corresponds to the possible individual and the predicate constant corresponds to the set of possible individuals in the given assignment point.

Now we have to establish the value of the expression of L_0 according to the interpretation I. We consider N as a set of natural numbers and the sequence of elements of D as a function N → D. It is safe to use the same letter N for a set and an operator because they occur in different contexts. We use the sequences of the elements of D for assigning values to variables.

We define Ext^I(t, k) where I = <A, F, R> is an interpretation, k is an assignment point and t is a term (an individual constant or a variable) of L_0 as follows. Ext^I(t, k) is a function from N → D to D. If t is a variable x, whose natural number is n, then Ext^I(x, k) (f) = f (n). If t is a constant c, then Ext^I(c, k) (f) = F(c) (k). In the former case Ext does not depend on F and in the later case it does not depend on f.

We define Ext^I(φ, k) where I and k is considered as above and φ is a formula of L_0 as follows. Ext^I(φ, k) is a function from N → D to B which we can consider as mapping φ to the subset of N → D.

1. If φ has the form of Ex(t) then Ext^I(φ, k) (f) = true if and only if Ext^I(t, k) (f) ϵ A(k);
2. If φ has the form of s = t then Ext^I(φ, k) (f) = true if and only if Ext^I(s, k) (f) = Ext^I(t, k) (f);
3. If φ has the form of P(t_0, ..., t_{n-1}) then Ext^I(φ, k) (f) = F(k)(P)(Ext^I(t_0, k), …, Ext^I(t_{n-1}, k));
4. If φ has the form of ~ψ then Ext^I(φ, k) (f) = true if and only if Ext^I(ψ, k) (f) = false;
5. If φ has the form of χ & ψ then Ext^I(φ, k) (f) = true if and only if Ext^I(χ, k) (f) = true and Ext^I(ψ, k) (f) = true;
6. If φ has the form of χ V ψ then Ext^I(φ, k) (f) = true if and only if Ext^I(χ, k) (f) = true or Ext^I(ψ, k) (f) = true;

7. If φ has the form of $\chi \rightarrow \psi$ then $Ext^I(\varphi, k)$ (f) = true if and only if $Ext^I(\chi, k)$ (f) = false or $Ext^I(\psi, k)$ (f) = true;
8. If φ has the form of A x. ψ then $Ext^I(\varphi, k)$ (f) = true if and only if for all y ϵ D $Ext^I(\psi, k)$ (f'_y) = true, where f'_y (i) = y, if i = n and f'_y (i) = f(i) otherwise, if n is a natural number of x;
9. If φ has the form of E x. ψ then $Ext^I(\varphi, k)$ (f) = true if and only if for some y ϵ D $Ext^I(\psi, k)$ (f'_y) = true, where f'_y is defined above;
10. If φ has the form of N ψ then $Ext^I(\varphi, k)$ (f) = true if and only if (k, {k' ϵ Asg | $Ext^I(\psi, k')$ (f) = true}) ϵ R(N).

The language L_0 is convenient for description of one subject having access to the semantic network. When we have multiple subjects, we must consider their interaction and the common restrictions for access system in general. So we need a language with more powerful abstraction means. We define now the language L_1 where it is possible to use (definite) descriptions among other constructions. This possibility allows to define a functional abstraction and express the general restrictions.

The language L_1 is defined as follows. The symbols of L_1 are contained in the following syntactical categories:

1. individual variables;
2. individual constants;
3. predicate constants;
4. logical constants;
5. syntactical delimiters;
6. n-ary predicate variables for every natural n. We denote them by $G^{(n)}_i$ where n is the arity;
7. operator □ (operator of the necessity);
8. operator I (operator of the description).

The first five categories are similar to the categories of L_0. We assume that the arguments of the predicate constants can belong not only to individual terms but also to predicate variables. The only operator is necessity, the other operators can be expressed through it. The description operator is used together with predicate variables for abstraction.

The set of terms of L_1 is analogous to L_0. The set of the formulae of L_1 is defined as follows.

1. if $t_0, ..., t_{n-1}$ are terms or predicate variables and P is a n-ary predicate constant then $P(t_0, ..., t_{n-1})$ is a formula;
2. if $t_0, ..., t_{n-1}$ are terms and $G^{(n)}_i$ is a n-ary predicate variable then $G^{(n)}_i(t_0, ..., t_{n-1})$ is a formula;
3. if s, t are terms then s = t is a formula;
4. if t is a term then Ex(t) is a formula;
5. if φ and ψ are formulae then $\sim\varphi$, φ & ψ, φ V ψ, $\varphi \rightarrow \psi$, Ax.φ, Ex.φ are formulae;
6. if φ is a formula and G is a predicate variable then AG.φ, EG.φ are formulae;
7. if φ is a formula and G is a predicate variable then φ'_G is a formula, where φ'_G is a result of the substitution of G not preceded by A, E, I, to IG.ψ ;
8. there are no other formulae.

The interpretation of L_1 will be defined as a pair I = <A, F> where A and F hold the points (1) and (2) of the interpretation of L_0, and also the following condition. If P is a predicate constant allowing the formula $P(t_0, ..., t_{n-1})$, then F(p) is n-ary $<Asg; U_0, ..., U_{n-1}>$-predicate, where $U_i = D$, if t_i is an individual constant or variable and U_i is a set of $<Asg; D^{k-1}>$-predicates if t_i is a k-ary predicate variable.

To establish the value of the expression of L_1 according to the interpretation I we define $Ext^I(t, k)$ where I = <A, F, R> is an interpretation, k is an assignment point and t is a term or a formula of L_1 as follows. The domain of Ext is a set of functions $N \times N \rightarrow U_i$, where for argument (n, k) $U_i = D$, if n = 0 and U_i is a set of $<Asg; D^{n-1}>$-predicates if n > 0. The codomain of Ext is D for terms and B for formulae.

1. If t is an individual constant c, then $Ext^I(c, k) (f) = F(c) (k)$;
2. If t is an individual variable x, whose natural number is i, then $Ext^I(x, k) (f) = f (0, i)$;
3. If t is a predicate variable $G^{(n)}{}_i$, then $Ext^I(x, k) (f) = f (n, i)$;
4. If t is a description IG.φ where φ is a formula, then $Ext^I(x, k) (f) = H$, where H is a function such that for all f $\{H (f)\} = \{P \mid x^{(n,k)}{}_P \in Ext^I(\varphi, k)\}$ or there is no such Q that $\{Q\} = \{P \mid x^{(n,k)}{}_P \in Ext^I(\varphi, k)\}$ and then $H(f) = Asg \times \{false\}$;
5. If φ has the form of $P(t_0, ..., t_{n-1})$ where P is a predicate constant and ti is a individual constant or individual variable or a term IG.φ where G is a predicate variable then $Ext^I(\varphi, k) (f) = F(k)(P)(Ext^I(t_0, k), \ldots, Ext^I(t_{n-1}, k))$;
6. If φ has one of the forms of Ex(t), s = t, ~ψ, χ & ψ, χ V ψ, χ → ψ, A x. ψ, E x. ψ, then the definition of $Ext^I(\varphi, k)$ is similar to the same definition for L_0;
7. If φ has the form of □ψ then $Ext^I(\varphi, k) (f) = true$ if and only if for all k ϵ Asg $Ext^I(\psi, k) (f) = true$.

The given definition allow to interpret the expressions of L_0 by L_1. We consider a brief example in the next section.

4. Model of the access to the semantic network

For modeling the restrictions for the access to the semantic network we adopt the languages L_0 and L_1 having the same Asg. Assignment points represent the states of the semantic network, including the states where the restrictions hold and the states where the restrictions are violated. We consider the set S where states are semantically correct (the restrictions hold). We assume the accessibility relation ρ between k and k' where k ρ k' when k' can be produced from k by one of the access control operation.

Then we assume that L_0 contains the operator N, for which $R(N) = \{(k, T) \mid T = \{k'\} \; \& \; k \rho k' \; \& \; k' \in S\}$. According to the definition of Ext we have Ext (Nφ, k) = true when Ext (φ, k') = true. We consider Ext (Nφ, k) = true as the formalization of the judgment that it is possible to satisfy φ in the semantically correct assignment point.

We can now investigate the properties of the system. For example, it it easy to show that in the assumed prepositions the following formulae have the truth value “true” in all assignment points:

1. Nφ, if φ is a propositional tautology;
2. $N(\varphi \rightarrow \psi) \rightarrow (N\varphi \rightarrow N\psi)$
3. $N (N(\varphi \rightarrow \psi) \rightarrow (N\varphi \rightarrow N\psi))$

4. $\sim N\varphi \rightarrow N(\sim N\varphi)$
5. $N(\sim N\varphi \rightarrow N(\sim N\varphi))$

These formulae can be considered as the general logical laws for access control system. For example, (1) shows that we can not make a requirement if it is self-contradictory. (2) allows to reason about the logical properties of the system inside the system itself. (3)-(5) are the variants of the introspection properties.

If we want to provide some special properties to the access control system, then we can formulate the appropriate conditions on ρ. For example we can propose that every assignment point is accessible from every other point. We can also accept more weak but more realistic condition that for every $k \in Asg$ exists $k' \in S$ such that $k \rho k'$. This condition correspond to the situation when there is no semantically incorrect states that can not be repaired. We can also take into account some computational properties and require for ρ reflexivity or transitivity. So the model gives the basis for implementation of the access control system.

5. Approbation of model constructions

The model is currently under development. Nevertheless, the proposed basic constructions of the model have been tested during the construction of separate components of information support for the preparation processes and implementation of the best available technologies (BAT).

The basic tools of access and manipulation for SN structures have been tested during the construction of tools to support the model for the BAT implementation in cement production. The model was oriented both to the computation of attribution for SN constructions with respect to correlation, and to the generation of expressions representing the attributions of SN constructions when they are stored in a relational database.

The computation of SN constructions took into account the correlations given in the input language of the **system. The possibility was ensured to specify constants representing individuals** in the subject domain and to include them into the extensions of the concepts forming the model. The extensions of complex expressions were specified functionally as functions of their integrated parts.

In the course of the computations it was possible to assign attributions to various parts of the expression when having various correlations. The syntactical means for this attribution assigning is to specify the context of the computations and associate it with a sub-expression (in the current implementation – with a term). Different sub-expressions can be associated with different correlations, which provide the computation of expressions that cannot be specified extensionally.

The performed possibilities of computations provide modelling of separate features of intensional character for the considered subject area. The number of the interesting features is enriched by the ability to specify the type of individual when working with the subject domain. The type specification is performed by establishing the correlation of the considered individual with the frame setting the specified type. After the correlation is established it becomes possible to assign attributions to the frames associated with the specified type.

The specification is significantly used in cases where measurements are required to assess the environmental situation at the BAT implementation facility. The measuring causes an expansion of knowledge about the subject domain, i.e., the transition from one correlation to another. Based on the measurement results the classification of the implementation facility as a whole or its parts is made or refined.

The testing of the possibilities to develop and apply the operational SN also took place. Such SN operates the constructions of object (subject-oriented) SN, i.e., contains the tools for their description, comparison and manipulation. As expected, the development of the operational SN will lead to the possibility of identifying typical patterns of user interaction with the SN and the organization of access to the SN in accordance with patterns that will include sets of access functions. The implementation of the supporting system is made in the Java language. The test sample for the development and testing of subject-oriented mechanisms contains about 200 concepts.

The elements of the technology of multi-level interacting SNs have been also developed during the construction of an instrumental system of visualization tools to support the tasks of information maintenance for the BAT implementation. The visualization tools system is an extension of the applicative computing system of the evaluator of conceptual objects; this system provides the construction of a conceptual model of the visualized subject domain in the form of the interrelated objects system. The structure of objects is given in the language of the conceptual objects evaluator system and can be researched and / or modified by applicative methods.

The access to objects is set through a function that is associated with the interface element. The function in the system is the first order object and is usually received by applying a function of higher order. A typical set of system functions provides for the verification of the validity of a particular access operation for a given user and its execution in the case of a positive test result.

The structure of the used objects provides a dynamic link to the objects of a given type of a set of functions for their processing. The resulting set of higher-order functions is similar to the operational SN both in terms of functionality and in place in the architecture of the system, and in fact can be considered as an implementation of the operational SN by the method of functional programming. The applicative nature of the implementation provides the possibility of integrating the visualization system with the model support system both at the level of the SN description and at the level of the applicative mechanisms used.

The visualization system has been used to prepare the fragments of documentation that form the description of the subject domain of the BAT implementation. The system is developed in the Java language. The test sample of the description of the visualized domain contains about 250 top-level concepts and about 5000 basic concepts.

6. Conclusion

The paper suggests an approach to solving the problem of data storage in the Web environment based on the SN use. It is proposed to solve the critical task of control over the access to SN objects, and, in particular, of access restrictions to the SN, basing on indentifying the operational SN, which processes the access restrictions to the SN; in particular, it provides setting the specialized managing objects - policies of access to the SN. The correctness of the supporting tools construction is provided by building a model of access to the SN based on intensional logic. The proposed method of construction provides, in particular, the following:

- increasing the flexibility of access to data by taking into account their semantic characteristics when accessing;
- ability to describe the access to data for various entities, including joint access to data;
- ability of the user's views concordance on the data, leading to the possibility of dynamic formation of users coalitions.

The elements of the proposed approach have been tested when developing the information systems that support the development of institutional foundations for the implementation of the best available technologies in the Russian Federation. The test demonstrated the possibility of achieving the set goals, which determines the practical significance of the proposed approach.

References

[1] Atzeni, P. et al. 2013.
The relational model is dead, SQL is dead, and I don't feel so good myself. SIGMOD Rec. 42, 1 (Jul. 2013), 64–68.
[2] Chernyshov A., Balandina A., Kostkina A., Klimov V. Intelligence Search Engine and Automatic Integration System for Web-Services and Cloud-Based Data Pro-viders Based on Semantics, Procedia Computer Science, 10.1016/j.procs.2016.07.449
[3] Conceptual Modeling. Proceedings of 35th International Conference by ed. Comyn-Wattiau, I. et al, ER 2016, Gifu, Japan, November 14-17, 2016, Lecture Notes in Computer Science, Volume 9974 2016, 2016, Springer
[4] Gabriella Castro Barbosa Costa. Using data provenance to improve software process enactment, monitoring and analysis. In Proceedings of the 38th International Conference on Software Engineering Companion, ICSE '16, pages 875-878, New York, NY, USA, 2016. ACM.
[5] Larisa Ismailova. Criteria for computational thinking in information and computational technologies. *Life Science Journal*, 11(9s):415-420, November 2014.
[6] Montague, R., 1968, "Pragmatics", in R. Klibansky (ed.), Contemporary Philosophy. A Survey, Florence: La Nuova Italia Editrice, 102–122. Reprinted in Thomason (ed.) 1974, pp. 95–118.
[7] Montague, R., 1970b, "Pragmatics and intensional logic", Synthese, 22: 68–94. Reprinted in Thomason (ed.) 1974, pp. 119–147.
[8] Population Modeling Working Group. Population modeling by examples (wip). In Proceedings of the Symposium on Modeling and Simulation in Medicine, MSM '15, pages 61–66, San Diego, CA, USA, 2015. Society for Computer Simulation International.
[9] Scott, D. "Advice in modal logic", in Philosophical Problems in Logic, ed. Karel Lambert. Reidel, 1970
[10] V. Wolfengagen et al. Migration of the Individuals Procedia Computer Science, Volume 88, 2016, pages 359–364, http://dx.doi.org/10.1016/j.procs.2016.07.449
[11] Wolfengagen,V.E., Ismailova,L.Y., Sergey V. Kosikov Computational Model of the Tangled Web Procedia Computer Science, 10.1016/j.procs.2016.07.440
[12] Wolfengagen,V.E. Ismailova,L.Y. et al Concordance in the Crowdsourcing Activity Procedia Computer Science, 10.1016/j.procs.2016.07.448
[13] Wolfengagen,V.E. Ismailova,L.Y. et al Evolutionary Domains for Varying Individuals Procedia Computer Science, 10.1016/j.procs.2016.07.447