Mapping Teaching Knowledge

Tatiana Gavrilova, Vladimir Gorovoy, Elena Petrashen
Saint-Petersburg State University, Graduate School of Management, Information Technologies for Management Dpt, Volkhovsky per. 3, 199004
Saint-Petersburg, Russia
tgavrilova@gsom.pu.ru, gorovoy@gsom.pu.ru, eap367@nyu.edu

Abstract. This paper discusses the ontological approach to the knowledge categorization and structuring for e-learning portal design as it turns out to be efficient and relevant to current domain conditions. The role of course content ontology visualization is specifically highlighted, as that way materials are presented in a comprehensive but easily digestible way and the structure can be adapted for learners' needs easily. We describe the experience of developing Knowledge Engineering course ontology and "OntolingeWiki" tool, which permits to create ontology-based e-learning portals.

1 Introduction

In recent years, there has been a noticeable interest in means to increase efficiency of ways in which learning content can be presented to a learner. And as e-learning solutions development spans not only pedagogy but also psychology, sociology and human-computer interaction, interdisciplinary ideas and notions prove to be a fertile ground for such means. Given increasing importance of visual aspect of technology [13] and growth of economy's knowledge-intensive share, such concepts as just-in-time delivery, usability and task relevance become crucial for educative systems. To provide this kind of quality organization and presentation of learning materials would be as important as these materials' content.

Using ontologies in educational systems is not really a new concept as they have often been used to represent different concepts of the course [2].

However, the importance of specification and structuring the content as well as its visual presentation – followed with such issues as design, adaptation and usability has been underestimated to a certain extent until recent times as the researchers were far more concerned with means to educate (methods of instruction or reasoning over the content) than with approaches to present the object of the research (content specification and knowledge structure) [6]. So ontology as a mean to form content and/or navigation system is rather new and promising field. In recent years, there has been a growing interest in the development and use of domain ontologies, strongly motivated by the Semantic Web initiative.

In this paper we describe the experience of developing Knowledge Engineering ontology for one of undergraduate courses in Saint-Petersburg State Polytechnic University. So firstly we review the literature, and the discussion on reasons, that let us consider ontology-based conceptual domain modeling being so efficient, follows.

Third section describes ontology development in more details, and in the fourth one we describe OntolingeWiki, which is an ontology-based tool for creating educational portals. The paper concludes with summary and a brief discussion on implications for future research.

2 Ontological Framework: a brief overview

Numerous known definitions of "ontology" may be generalized by: "Ontology is a hierarchically structured vocabulary describing a domain that can be used as a skeletal foundation for a knowledge base" [4]. Generally, experts argue about the distinctions between the ontology and the user's conceptual model [6][7]. Our take on that would be considering that the ontology corresponds to the analyst's view of the conceptual model, but is not the de facto model. That way, definition above clarifies the ontological approach to knowledge structuring and provides sufficient freedom for open-ended, creative thinking.

For example, ontological engineering can provide a clear course structure, equipping a learner with a map of connected resources and their relationships. This is where the Semantic Web principles are vital and ontological reasoning is a promising tool, helping to provide a formal description for a shared domain conceptualization [15]. This conceptualization also leads to process of selection which things are relevant to be represented and which are not, so construction of top ontology leads to expressing knowledge [2]. Some other pros would be:

1. The relationships between entities can be more clearly expressed and it allows better reasoning. So it's possible to share common understanding of the structure of information among people or software agents [10] and to separate domain knowledge from the operational knowledge [11].

2. Content-oriented view could facilitate knowledge sharing and reuse [9]. That goes in line with two of the most current research issues in the e-learning community - specifying reusable blocks of learning content and defining an abstract way for designing different units (e.g. lessons) [7], which is motivate by unfortunate fact that even though many e-learning systems exist and keep being developed, knowledge reuse from one system to another is almost non-existent.
3. It also provides us with the ability to easily modify the course's structure adapting to different educational purposes. One-size-fits-all approach is often not good in learning environment as not only do learners are of varying degrees of proficiency and professionalism, but their goals, expectations, techniques and motivations for learning are also different.
4. Ontologies are good in making domain assumptions explicit so that it's possible to change these assumptions easily if knowledge about the domain changes [12].
5. Besides, as the Internet itself is so large-scale hypertext, getting a comprehensible and explicit overview of all the available information relevant to current user's needs and tasks is often difficult if not impossible as if the user is not fully experienced in the knowledge domain she is not fully able to define whether found content is entirely appropriate, up-to-date and relevant for the task [1]. Research shows that ontologies are good in providing intelligent, task-centered and relevant information support for solving problems and performing learning tasks[6].
6. A visual presentation of course's structure helps facing two very common problems in learning: loss of overview due to low information density of the medium and short attention spans [11]. The large picture is more comprehensible and digestible: that parallels the way it's much more easier to understand what jigsaw puzzle pieces are about when the puzzle is gathered, not separated.

3 Knowledge Engineering ontology creating

It has been mentioned that the majority of works in the field put emphasis on ontology specification, not capture, so we would want to stress the importance of latter a bit more.

To create Knowledge Engineering ontology we used a slightly modified version of Four-Step Algorithm, described in [4]:

1. Goals, strategy and boundary identification: The first step in ontology development should be to "identify the purpose of the ontology and the needs for the domain knowledge acquisition."
2. Meta-concept identification: gathering all the information relevant to the described domain - selecting and verbalizing all of the essential objects and concepts in the domain.
3. Laddering, including categorization and specification: forming categories by creating high-level concepts and/or breaking them into a set of detailed ones where it is needed. This could be done via a top-down strategy by trying to break the high level concept from the root of the previously built hierarchy, by detailing and specification of instance concepts. Another way is generalization via bottom-up structuring strategy, associating similar concepts to create meta-concepts of leaves.
4. Refinement: updating the visual structure by excluding any excessiveness, synonymy, and contradictions. As well-balanced ontological hierarchy equates to a strong and comprehensible representation of the domain's knowledge, it should have the following qualities[4][5][15]:
   - Accuracy
   - Completeness
   - Clarity in concepts and relationships
   - Cognitive adequacy
   - Conceptual balance

It's also important to exclude contradictions and to avoid the ontology being too complex and/or big. Thus, our decision was to introduce scalable GUI - with option to zoom and center on chosen fragments of ontology the way the screen is not overloaded with the information.

Hierarchy concepts are tied with self-contained chunks of training content – learning objects, targeting reusability since these learning objects can be reassembled later to create other courses or to personalize this one for different types of learners (e.g. different levels of proficiency). Visual form influences both analyzing and synthesizing procedures in ontology development process [16], helping to provide unambiguous and complete learning design [14]. The developed ontology will be also used as a table of contents for educational system.

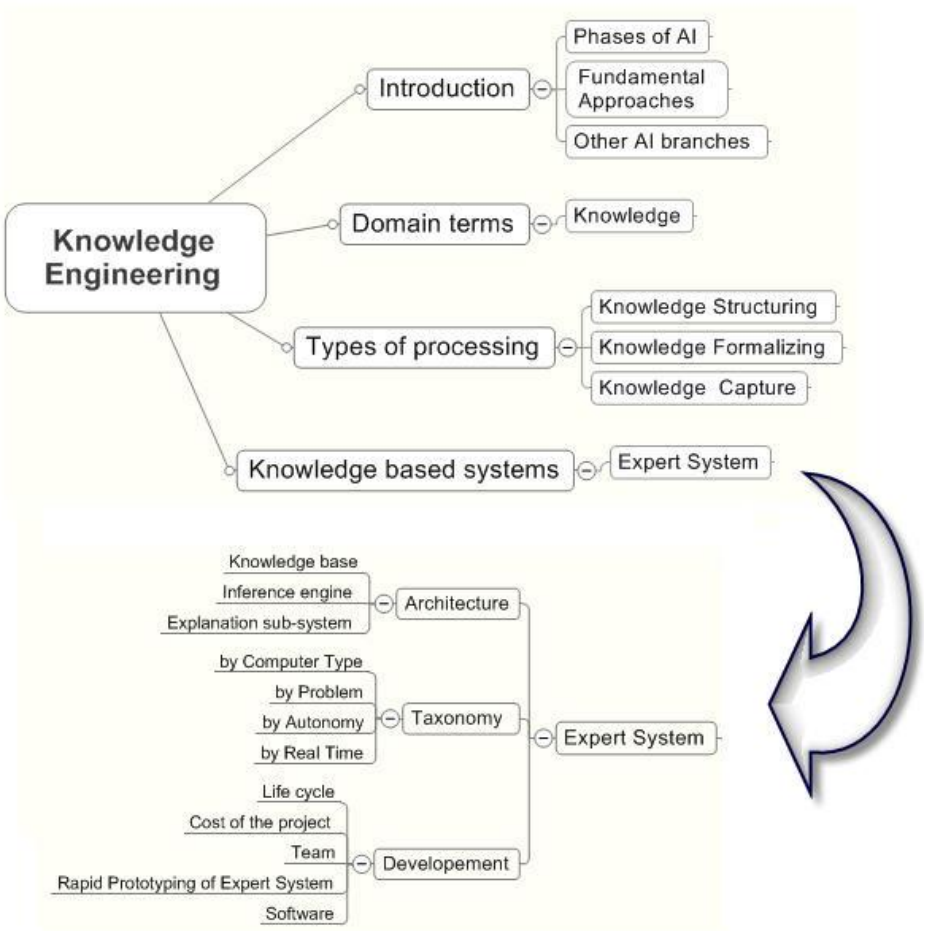


Fig. 1. Top Levels of the Educational Ontology of Knowledge Engineering (nodes can be opened to present the nested data)

4 OntolingeWiki – technology for creating e-learning systems

As we witness rise of crowdsourcing and other collaborative techniques in technology, it seems interesting to include collaboration in the conceptual modeling process as well [11] - so that students could not only browse the presented structure, but also modify it.

OntolingeWiki is a tool that takes advantage of both wiki-technology, which targets perfectly collaborative development and ontologies as a tremendous tool for knowledge structuring. It can import any ontology saved in OWL format and provide web-interface for its navigation with visualization based on hyper graph technology (http://hypergraph.sourceforge.net). Each concept of the ontology can be annotated with wiki-page created on demand (see http://ontowiki.org.ru:8180/ontolinge/dispatcher). OntolingeWiki was created based on Ontolinge-KAON system [3].

This technology can be used to create ontology-based educational portals in the way that doesn't require much time or technology savvy from the developer. It was successfully leveraged in the design of the ontology-based content management system for the virtual exposition in Saint-Petersburg Optical Technologies Museum. Digital teaching materials such as presentations, animations and java-applets were united in the virtual exposition, which introduces visitors to scientific principles and theories underlying museum's collection.

5 Summary and future work

As modern tendencies in e-learning are to allow easy sharing and reusing of educational systems' content and knowledge components, while not minimizing resources required, ontology proves itself to be a useful structuring tool. It is able to enrich the process of education, providing users of e-learning systems with an organizing axis to help them mentally mark their own personalized picture of domain knowledge [4].

We argue that knowledge categorization and laddering makes content more comprehensive for an individual and provides a technologically stable database. But the usability and appropriateness of this ontology should be further investigated and refined accordingly.

We consider described Knowledge Engineering ontology a promising start towards a fully functional e-learning system and OntolingeWiki a step forward in creating a quality technological environment for educational collaborative systems development.

References

1. Aroyo L., Dicheva, D.: The New Challenges for E-learning: The Educational Semantic Web. In: Educational Technology & Society, 7 (4), pp. 59-69 (2004)
2. Breuker J., Bredeweg B.: Ontological Modelling for Designing Educational Systems. AI-ED 99 Workshop on Ontologies for Educational Systems (1999)
3. Cristea A.: What can the Semantic Web do for Adaptive Educational Hypermedia? In: "Educational Technology and Society" 7(4) IEEE, LTSC, pp 40--58 (2004)
4. Gavrilova, T., Laird, D.: Practical Design Of Business Enterprise Ontologies. In: Industrial Applications of Semantic Web, Springer (2005)
5. Guarino N., Schneider L.: Ontology-Driven Conceptual Modeling. Lecture notes (2002)
6. Ikeda M., Hayashi Y., Lai J., Chen W., Bourdeau J., Seta K., Mizoguchi R.: An ontology more than a shared vocabulary. AI-ED 99 Workshop on Ontologies for Intelligent Educational Systems (1999)
7. Knight C., Gaševic D., Richards G.: Ontologies to integrate learning design and learning content. In: Journal of Interactive Media in Education (2005)
8. Jonassen D.: Computers in the Classroom: Mindtools for Critical Thinking, Englewood Cliffs, NJ: Prentice Hall (1996)
9. Mizoguchi R., Vanwelkenhuysen J., Ikeda M.: Task Ontology for reuse of problem solving knowledge. IOS Press (1995)
10. Musen M. Dimensions of knowledge sharing and reuse. In: Computers and Biomedical Research 25, pp. 435-467. (1992)
11. Naeve A, M. Nilsson, M. Palmer: E-Learning in the Semantic Age. The 2nd European Web-based Learning Environments Conference (2001)
12. Noy N., McGuinness D.: Ontology Development 101: A Guide to Creating Your First Ontology, Knowledge Systems Laboratory, Stanford (2001)
13. Prensky M.: Digital Natives, Digital Immigrants. NCB Univ. Press, Vol. 9 (2001)
14. Psyché V., Mendes O, Bourdeau J. : Apport de l'ingénierie ontologique aux environnements de formation à distance, Revue STICEF, Volume 10 (2003)
15. Snae C., M. Brueckner: Ontology-Driven E-Learning System Based on Roles and Activities for Thai Learning Environment. In: Interdisciplinary Journal of Knowledge and Learning Objects, Volume 3 (2007)
16. Sosnovsky S., Gavrilova T. Development of Educational Ontology for C-programming. In: Information Theories & Applications, Volume 13 (2005)

Prefix Rewriting Systems as Object-Oriented Data Models*

Alexander E. Gutman

Sobolev Institute of Mathematics,
Acad. Koptyug av. 4, 630090 Novosibirsk, Russia
http://math.nsc.ru/

Novosibirsk State University,
Pirogova 2, 630090 Novosibirsk, Russia
http://www.nsu.ru/

Abstract. A deterministic longest-prefix rewriting system is a rewriting system such that there are no rewriting rules $X \to Y$, $X \to Z$ with $Y \neq Z$, and only longest prefixes of words are subject to rewriting. Given such a system, analogs are defined and examined of some concepts related to object-oriented data systems: inheritance of classes and objects, instances of classes, class and instance attributes, conceptual dependence and consistency, conceptual scheme, types and subtypes, etc. A special attention is paid to the effective verification of various properties of the rewriting systems under consideration. In particular, the algorithms are presented for answering the following questions: Are all words finitely rewritable? Do there exist recurrent words? Is the system conceptually consistent? Given two words X and Y, does X conceptually depend on Y? Does the type of X coincide with that of Y? Is the type of X a subtype of that of Y?

Key words: prefix rewriting, term rewriting, object-oriented data system, information system, consistency verification, ontology of a data model.

1 Introduction

The classical object-oriented approach to describing structured data employs the two primary relations, *has* (or "*has a*") and *is* (or "*is a*").

The *has* relation links objects (and classes) with their attributes. By saying "`X` *has a* `Y`" we mean that the object (or class) `X` possesses an attribute named `Y`, and we thus can speak of "*the* `Y` *of* `X`" or "`X`*'s* `Y`" as a property of `X` conventionally denoted by `X.Y`. For instance, if a web page has a submit button whose style assumes a border of a particular width, we can speak of "the width of the border of the style of the submit button of the page" and thus arrive at the object `page.submitButton.style.border.width`.

The *is* relation can be used for (1) instantiating objects from classes; (2) inheriting classes from classes; and (3) assigning values to attributes. By saying "`40` *is an* `Integer`" we associate the object `40` with the class `Integer` and mean that `40` is an instance of `Integer`. The phrase "`Integer` *is* `Number`" means that the class `Integer` inherits from the class `Number`. By claiming that "`John.age` *is* `40`" we assign the value `40` to the `age` attribute of `John`.

As is seen from the above examples, the wide interpretation of the *is* relation makes it possible to eliminate the difference between objects and classes. A single data system can syndicate the class declarations ("metadata") and the object instantiations and initializations ("data"). We do not assert that data and metadata are worth more combined than separated; nevertheless, this approach allows us to unify data analysis and develop a common tool for verifying conceptual and semantical consistency.

The *is* and *has* relations are naturally connected. By interpreting "*is*" as "inherits," we assume that if "`X` is `Y`" then all the attributes of `Y` are inherited by `X`. In particular, if "`X` is `Y`" and "`Y` has a `Z`" then "`X` has a `Z`." Moreover, if "`X` is `Y`" is the only explicit information on `X`, we can conclude that "`X.Z` is `Y.Z`." (By the explicit information we mean the *is* rules which form the data system under consideration.) In doing so, we derive an *implicit* information on `X` and say that "`X.Z` *is* `Y.Z` *implicitly*." Therefore, when evaluating the object `X.Z`, we rewrite its prefix `X` with `Y` according to the explicit rule "`X` is `Y`." The same is applicable to objects of any length. For instance, if we know explicitly that "`A.B` is `P.Q.R`" then `A.B.C.D` rewrites implicitly to `P.Q.R.C.D`.

* The work is supported by the Russian Science Support Foundation.

It is clear that the explicit rules supersede any implicit derivatives; therefore, if "`X` is `Y`," "`Y` has a `Z`," and the data system contains the explicit rule "`X.Z` is `A`," the latter wins over the implicit "`X.Z` is `Y.Z`." However, a conflict of another kind is possible in case several explicit rules are simultaneously applicable. Consider the following fragment of a data system:

```
block.style.color  is  blue
header.style.color is  red
button             is  block
button.style       is  header.style
```

Let us try to evaluate the button's style color, i.e., `button.style.color`. Since `button` is a `block`, we might conclude that `button.style.color` is `block.style.color`, which is `blue`. On the other hand, `button.style` is `header.style`; therefore, `button.style.color` is `header.style.color`, which is `red`. Intuitively, the latter evaluation should win, since "`button.style` is `header.style`" seems to take priority over "`button` is `block`." The reason is not the fact that the former rule occurs next to the latter (we treat a data system as an unordered set of *is* rules). The key point is that the rule "`button.style` is `header.style`" is more *concrete* as it evaluates a longer object, `button.style` rather than `button`. Therefore, when evaluating an object, we should *rewrite the longest prefix* (i.e., use the most concrete rule applicable).

We will now dwell on *data consistency*. Obviously, when designing a set of definitions, conceptual cycles should be avoided. By saying "`man` is `man`" we define nothing, since evaluation of `man` results in a dead cycle. However, conceptual consistency in no way outlaws recursion. For instance, the rule "`man.son` is `man`" is quite legal. On the other hand, the rule "`man` is `man.son`" seems incorrect: we still do not know what *man* is unless *man*'s *son* is defined, while the latter is senseless prior to defining *man*. Furthermore, the rules "`man` is `Adam`" and "`Adam.rib` is `man.rib`" form an inconsistent pair, since `Adam.rib` is `man.rib`, while the latter implicitly rewrites to `Adam.rib`. Such examples justify the need for a formal definition of conceptual consistency and the search for the corresponding effective verification. (This is similar to analyzing the ontology of a data system as a set of concept definitions.)

It is clear that, prior to defining a set of concepts (classes or objects), we need at least one concept which does not require definition. In general, there can be several primary concepts; however, a single "generic object" is sufficient. We denote the latter by ω. Given a data system and a word X of the form $\texttt{entity.attr}_1\texttt{.attr}_2\texttt{.}\cdots\texttt{.attr}_n$, we rewrite X by applying the most concrete *is* rule, thus obtaining a new word, and continue rewriting the longest prefixes of the subsequent words until ω is reached. In this case we conclude that the initial word X is an *object* (or a *concept*). Otherwise, if the rewriting process either (1) ends with a nonrewritable word other than ω or (2) never terminates, we claim that X is senseless. The possibility of (2) makes the analysis nontrivial and justifies the search for an effective verification if a given word makes sense. (This is close to analyzing the ontology of a concept within a data system.)

Given an object X and a word δ of the form $\texttt{attr}_1\texttt{.attr}_2\texttt{.}\cdots\texttt{.attr}_n$, say that δ is a *detail* of X if $X\texttt{.}\delta$ makes sense. The set $\|X\|$ of all details of X can be regarded as the *type* of X. Whenever an algorithmic procedure assumes a formal argument A, the body of the procedure contains A along with some words $A\texttt{.}\delta_i$. For the procedure to operate correctly with X substituted for A, it is necessary (and probably sufficient) that all the words $X\texttt{.}\delta_i$ make sense. This results in the requirement that X be of an appropriate type. Therefore, we need an algorithm for comparing object types: given two objects X and Y, we should be able to effectively compare the types of X and Y, i.e., determine which of the relations $\|X\| = \|Y\|$, $\|X\| \subseteq \|Y\|$, $\|X\| \supseteq \|Y\|$ hold. The problem is not trivial if for no other reason than the fact that the type of an object can be infinite. (For instance, given the rule "`man.son` is `man`," the type of `man` contains all the words `son`, `son.son`, `son.son.son`, ...)

In what follows we give formal definitions for the notions under consideration, state some results, and present algorithms for all the problems mentioned above. Unfortunately, the length restrictions on paper submissions forced us to exclude the proofs of the theorems and justifications of the algorithms. (All the details, including various examples, will be published elsewhere.)

To make notation less cumbersome, we treat the names of entities and attributes as single symbols (letters) of some alphabet $\mathbb{A}$ and agree to write the property paths $\alpha_1\texttt{.}\alpha_2\texttt{.}\cdots\texttt{.}\alpha_n$ as $\alpha_1\alpha_2\ldots\alpha_n$ thus making them words over $\mathbb{A}$. The explicit rules "X *is* Y" will be written as $X \to Y$.

2 Definitions and main results

Throughout the paper, $\mathbb{A}$ is a finite alphabet and $\mathbb{A}^*$ (resp. $\mathbb{A}^+$) is the set of all (all nonempty) words over $\mathbb{A}$. The elements of $\mathbb{A}$ are called *letters*. We conventionally identify the letters with the corresponding single-letter

words. Say that X is a *prefix* (resp. a *proper prefix*) of $Y \in \mathbb{A}^+$ and write $X \sqsubseteq Y$ or $Y \sqsupseteq X$ ($X \sqsubset Y$ or $Y \sqsupset X$) if $X \in \mathbb{A}^+$ and $Y = XS$ for some $S \in \mathbb{A}^*$ ($S \in \mathbb{A}^+$). The length of a word X is denoted by $|X|$. Given an integer $n \geqslant 1$ and a word $X \in \mathbb{A}^+$ such that $|X| \geqslant n$, define $X\restriction_n \in \mathbb{A}^+$ so that $X\restriction_n \sqsubseteq X$ and $\left|X\restriction_n\right| = n$. For brevity, in the sequel we say "word" instead of "nonempty word over $\mathbb{A}$."

Given any binary relation $\rightsquigarrow$, we conventionally denote by $\overset{+}{\rightsquigarrow}$ the transitive closure of $\rightsquigarrow$ and by $\overset{*}{\rightsquigarrow}$, the reflexive transitive closure of $\rightsquigarrow$.

Consider a finite binary relation $\to$ on $\mathbb{A}^+$ (i.e., a finite subset of $\mathbb{A}^+ \times \mathbb{A}^+$) and a letter $\omega \in \mathbb{A}$. Say that the pair $\langle \to, \omega \rangle$ is a *deterministic longest-prefix rewriting system*, or a *system* for short, if $\to$ is nonempty and the following hold:

(a) $X \to Y$ and $X \to Z$ imply $Y = Z$;

(b) there are no $S, Y \in \mathbb{A}^*$ such that $\omega S \to Y$.

Put $\mathbb{E} := \{X : X \to Y \text{ for some } Y\}$ and call the elements of $\mathbb{E}$ *explicit words*. Say that E is an *explicit prefix* of X if $E \in \mathbb{E}$ and $E \sqsubseteq X$. Say that a word X is *rewritable* if X has an explicit prefix.

As is easily seen, condition (a) means that, for each $E \in \mathbb{E}$, there is a unique word E' such that $E \to E'$, while condition (b) amounts to the fact that all words of the form ωS, with $S \in \mathbb{A}^*$, are not rewritable.

Given a rewritable word X, consider the longest explicit prefix E of X, determine the suffix $S \in \mathbb{A}^*$ so that $X = ES$, and put $X' := E'S$. We call X' *the rewrite* of X. Introduce the binary relation $\Rightarrow$ on $\mathbb{A}^+$ by setting $X \Rightarrow Y$ if and only if X is a rewritable word and $Y = X'$.

By way of recursion, put $\mathbb{W}_0 := \mathbb{A}^+$, $X^{(0)} := X$ for $X \in \mathbb{W}_0$ and, for each $n \geqslant 1$, put $\mathbb{W}_n := \{X \in \mathbb{W}_{n-1} : X^{(n-1)} \text{ is rewritable}\}$ and $X^{(n)} := (X^{(n-1)})'$ for $X \in \mathbb{W}_n$. The word $X^{(n)}$ is called the *nth rewrite* of X. It is clear that, for each $X \in \mathbb{W}_n$, we have $X = X^{(0)} \Rightarrow X^{(1)} \Rightarrow \cdots \Rightarrow X^{(n)}$, $X \overset{+}{\Rightarrow} X^{(n)}$ for $n > 0$, and $X \overset{*}{\Rightarrow} X^{(n)}$ for $n \geqslant 0$.

The elements of $\bigcap_{n=1}^{\infty} \mathbb{W}_n$ are called *infinitely rewritable words*. The other words are *finitely rewritable*. Given a word X, call the maximal (finite or infinite) sequence of the form $\langle X^{(0)}, X^{(1)}, X^{(2)}, \dots \rangle$ the *rewriting sequence* of X. Therefore, a word X is finitely (infinitely) rewritable if and only if the rewriting sequence of X is finite (infinite).

Say that $X \in \mathbb{A}^+$ is an *object* if $X \overset{*}{\Rightarrow} \omega$. Let $\mathbb{O}$ be the set of all objects. Note that the rewriting sequence of every object $X \in \mathbb{O} \backslash \{\omega\}$ has the form $X = X^{(0)} \Rightarrow \cdots \Rightarrow X^{(n)} \to \omega$, where $n \geqslant 0$.

¿From the above notation and definitions it is clear that we treat a system $\langle \to, \omega \rangle$ as a rewriting system and assume that only longest prefixes of words are subject to rewriting. The system is then regarded as a recognition device, with $\mathbb{O}$ the accepted language (see [1]). We have called such a system "deterministic," since every rewritable word has a unique rewrite.

We may regard the notion of an object as isolating "concepts" from "senseless words." An object is a word X possessing a "meaning," the rewrite $X' = X^{(1)}$, which also possesses a meaning, $(X')' = X^{(2)}$, and so on up to the final rewrite, the "generic object" ω, whose meaning is assumed predefined. The relation $\to$ is thus treated as conceptual definition, and a rule $X \to Y$ is regarded as a definition of X via Y: "X is a Y." Next, a rule $X\alpha \to Z$ is an attribute definition, "the α of X is a Z," while $X\alpha\beta \to Z$ means "the β of the α of X is a Z," etc. In this respect, condition (a) imposed on the relation $\to$ amounts to conceptual unambiguity (no concept can have several meanings).

We may also treat the relation $\to$ as object-oriented inheritance or instantiation and regard a rule $X \to Y$ as an explicit indication of the fact that "class X directly inherits class Y" or "object X is an instance of class Y." Next, a rule $X\alpha \to Z$ may be regarded as an attribute declaration or property evaluation: "class X has attribute α of class Z" or "the property $X\alpha$ has value Z" or "the property $X\alpha$ is an instance of class Z." In this respect, having imposed condition (a) on the relation $\to$, we thereby disallowed multiple inheritance (therefore, no object can belong to several incomparable classes).

Introduce the binary relation $\Rightarrow_{\mathrm{w}}$ on $\mathbb{A}^+$ by setting $X \Rightarrow_{\mathrm{w}} Y$ if and only if $X = ES$ and $Y = E'S$ for some $E \in \mathbb{E}$, $S \in \mathbb{A}^*$. Therefore, $\Rightarrow_{\mathrm{w}}$ is the rewriting corresponding to the system $\langle \to, \omega \rangle$ regarded as an ordinary prefix rewriting system rather than a longest-prefix rewriting system. (It is clear that $X \to Y$ implies $X \Rightarrow Y$, and $X \Rightarrow Y$ implies $X \Rightarrow_{\mathrm{w}} Y$. We may thus read the formulas $X \overset{+}{\Rightarrow} Y$ and $X \overset{+}{\Rightarrow}_{\mathrm{w}} Y$ as "X *rewrites to* Y" and "X *weakly rewrites to* Y." The formula $X \to Y$ can be read as "X *explicitly rewrites to* Y.")

Introduce the binary relation $\rightarrowtail$ on $\mathbb{A}^+$ by setting $X \rightarrowtail Y$ if and only if $X \sqsupset Y$ or $X \Rightarrow_{\mathrm{w}} Y$. As is easily seen, the transitive closure $\overset{+}{\rightarrowtail}$ is the least transitive relation on $\mathbb{A}^+$ possessing the following three properties for all $X, Y, S \in \mathbb{A}^+$:

$$\text{if } X \to Y \text{ then } X \overset{+}{\rightarrowtail} Y; \qquad \text{if } X \to Y \text{ then } XS \overset{+}{\rightarrowtail} YS; \qquad XS \overset{+}{\rightarrowtail} X.$$

In case $X \overset{+}{\rightarrowtail} Y$ we say that X *depends on* Y. A word X is *well-defined* if X does not depend on X. Say that a system $\langle \to, \omega \rangle$ under consideration is *conceptually consistent* if all words are well-defined, i.e., no word

depends on itself. For brevity, introduce the following named condition:

The system is conceptually consistent. (**Con**)

The above terminology is justified by our informal treatment of a rewriting rule $X \to Y$ as a definition of X via Y ("X is a Y") and regarding a rule $XS \to Z$ as a detail definition ("the S of X is a Z"). Therefore, informally, the relation $X \overset{+}{\rightarrowtail} Y$ can be understood as follows: the definition of X explicitly or implicitly employs Y; in particular, when subsequently describing a conceptual scheme, the concept Y should be introduced before X, otherwise X becomes ill-defined.

If $\rightsquigarrow$ is a binary relation on $\mathbb{A}^+$, put $|\rightsquigarrow| := \{X \in \mathbb{A}^+ : E \overset{*}{\rightsquigarrow} X \text{ for some } E \in \mathbb{E}\}$ and denote by $[\rightsquigarrow]$ the directed graph whose nodes are the words in $|\rightsquigarrow|$ and arcs are the pairs $\langle X, Y\rangle$ such that $X, Y \in |\rightsquigarrow|$ and $X \rightsquigarrow Y$.

Given a system, we call $[\rightarrowtail]$ the *conceptual scheme* and $[\Rightarrow_{\mathrm{w}}]$ the *weak rewriting scheme*. Since $X \Rightarrow_{\mathrm{w}} Y$ implies $X \rightarrowtail Y$, the weak rewriting scheme is a subgraph of the conceptual scheme.

Proposition 1. *Given $X, Y \in \mathbb{A}^+$, we have $X \overset{+}{\rightarrowtail} Y$ if and only if $X \sqsupset Y$ or $X \overset{+}{\Rightarrow}_{\mathrm{w}} YS$ for some $S \in \mathbb{A}^*$.*

Say that a word X is *weakly recurrent* if $X \overset{+}{\Rightarrow}_{\mathrm{w}} XS$ for some $S \in \mathbb{A}^*$.

Corollary 2. *A word is well-defined if and only if it is not weakly recurrent.*

Theorem 3. *The following properties of a system are equivalent:*
(1) *all words are well-defined, i.e., (Con) holds;*
(2) *each explicit word is well-defined;*
(3) *there are no weakly recurrent explicit words;*
(4) *there are no weakly recurrent words;*
(5) *the conceptual scheme is acyclic;*
(6) *the conceptual scheme is acyclic and finite;*
(7) *the weak rewriting scheme is acyclic and finite.*

Say that a word X is *recurrent* if $X \overset{+}{\Rightarrow} XS$ for some $S \in \mathbb{A}^*$. Introduce the following named condition:

There are no recurrent words. (**Rec**)

Proposition 4. *(Con) implies (Rec).*

Put $\mathbb{A}_{\mathbb{E}} := \min\{A \subseteq \mathbb{A} : \mathbb{E} \subseteq A^+\}$, i.e., $\mathbb{A}_{\mathbb{E}}$ is the *explicit alphabet*, the set of all letters occurred in explicit words. In addition, put $\mu := \max\{|E| : E \in \mathbb{E}\}$.

Theorem 5. *If each word $X \in \mathbb{A}_{\mathbb{E}}^+$ with $|X| \leqslant \mu$ is not recurrent then all words are not recurrent, i.e., (Rec) holds.*

Remark 6. Let $B(\mathcal{S})$ be a set of words defined via a system $\mathcal{S}$ by some condition. Say that $B(\mathcal{S})$ is a *recurrence basis* if, given an arbitrary system $\mathcal{S}$, nonrecurrence of all words in $B(\mathcal{S})$ implies nonrecurrence of all words. Theorem 5 states that the set $\{X \in \mathbb{A}_{\mathbb{E}}^+ : |X| \leqslant \mu\}$ is a recurrence basis. Despite its finiteness, the set can be rather large. However, we are not aware of conditions which determine considerably smaller recurrence bases. (There are examples showing that neither the set $\mathbb{E}$ of explicit words, nor the set of all prefixes of the explicit words can serve as a recurrence basis.)

Introduce the following named condition:

All words are finitely rewritable. (**Fin**)

Theorem 7. *If each explicit word is finitely rewritable then all words are finitely rewritable, i.e., (Fin) holds.*

Theorem 8. *(Rec) implies (Fin).*

Therefore, according to Proposition 4 and Theorem 8, we have the implications (Con) $\Rightarrow$ (Rec) $\Rightarrow$ (Fin). As examples show, the converse implications are not true in general.

Introduce the following two named conditions:

$$\text{All explicit words are objects, i.e., } \mathbb{E} \subseteq \mathbb{O}. \qquad (\textbf{Obj})$$

$$\text{If } X \in \mathbb{A}^+,\ \alpha \in \mathbb{A}, \text{ and } X\alpha \in \mathbb{E} \text{ then } X \in \mathbb{O}. \qquad (\textbf{PreObj})$$

Proposition 9.
(1) *Let $X, Y \in \mathbb{A}^+$, $X \overset{*}{\Rightarrow} Y$. Then $X \in \mathbb{O}$ if and only if $Y \in \mathbb{O}$.*
(2) *Assume (Obj). Then, given $X \in \mathbb{A}^+$, we have $X \in \mathbb{O}\backslash\{\omega\}$ if and only if $X \overset{*}{\Rightarrow} E$ for some $E \in \mathbb{E}$.*
(3) *Assume (PreObj). If $X, S \in \mathbb{A}^+$ and $XS \in \mathbb{O}$ then $X \in \mathbb{O}$.*

Let $X \in \mathbb{O}$ and $\alpha \in \mathbb{A}$. Say that α is an *attribute* of X if $X\alpha \in \mathbb{O}$. Denote by $\|X\|_1$ the set of all attributes of X. It is clear that $\|\omega\|_1 = \varnothing$.

Say that α is an *explicit attribute* (resp. *implicit attribute*) of X if $X\alpha \in \mathbb{O} \cap \mathbb{E}$ ($X\alpha \in \mathbb{O}\backslash\mathbb{E}$).

Say that α is an *overriding attribute* (resp. *added attribute*) of X if $X\alpha \in \mathbb{O} \cap \mathbb{E}$ and, in addition, $X'\alpha \in \mathbb{O}$ ($X'\alpha \notin \mathbb{O}$). If α is an overriding attribute of X, we say that $X\alpha$ *overrides* $X'\alpha$.

Therefore, every attribute is either explicit or implicit, and every explicit attribute is either overriding or added.

Proposition 10. *For all $X \in \mathbb{O}\backslash\{\omega\}$ and $\alpha \in \mathbb{A}$ the following hold:*
(1) *α is an implicit attribute of X if and only if $X\alpha \notin \mathbb{E}$ and $X'\alpha \in \mathbb{O}$;*
(2) *α is an added attribute of X if and only if $X\alpha \in \mathbb{O}$ and $X'\alpha \notin \mathbb{O}$.*

Proposition 11. *Assume (Obj). If $X, Y \in \mathbb{A}^+$, $\alpha \in \mathbb{A}$, $X \overset{*}{\Rightarrow} Y$, and $Y\alpha \in \mathbb{O}$ then $X\alpha \in \mathbb{O}$.*

Proposition 12. *Assume (Obj). Consider the rewriting sequence $X = X^{(0)} \Rightarrow \cdots \Rightarrow X^{(n)} = \omega$ of an object X. If $\alpha \in \|X\|_1$ then there is a number $0 \leqslant i \leqslant n$ such that $X^{(0)}\alpha, \ldots, X^{(i)}\alpha \in \mathbb{O}$, $X^{(i+1)}\alpha, \ldots, X^{(n)}\alpha \notin \mathbb{O}$, and α is an added attribute of $X^{(i)}$.*

Corollary 13. *Assume (Obj). A letter α is an attribute of an object X if and only if there is a number $n \geqslant 0$ such that $X^{(n)}\alpha \in \mathbb{E}$.*

Given an object X, say that δ is a *detail* of X if $\delta \in \mathbb{A}^+$ and $X\delta \in \mathbb{O}$. Denote by $\|X\|$ the set of all details of X and call $\|X\|$ the *type of X*. (It is clear that $\|\omega\| = \varnothing$.) Note that the set $\mathbb{O}$ of all objects can be infinite and, moreover, some object types $\|X\|$ can be infinite. On the other hand, we will see that the set $\{\|X\| : X \in \mathbb{O}\}$ of all object types is always finite (see Theorem 27).

Proposition 14. *For all objects X and Y we have*
(1) *if $\|X\| = \|Y\|$ then $\|X\delta\| = \|Y\delta\|$ for all $\delta \in \|X\|$;*
(2) *if $\|X\| \subseteq \|Y\|$ then $\|X\delta\| \subseteq \|Y\delta\|$ for all $\delta \in \|X\|$.*

On assuming (PreObj), we also have
(3) *$\|X\| = \|Y\|$ if and only if $\|Y\|_1 = \|X\|_1$ and $\|X\alpha\| = \|Y\alpha\|$ for all $\alpha \in \|X\|_1$;*
(4) *$\|X\| \subseteq \|Y\|$ if and only if $\|X\|_1 \subseteq \|Y\|_1$ and $\|X\alpha\| \subseteq \|Y\alpha\|$ for all $\alpha \in \|X\|_1$.*

Proposition 15. *If $X, Y \in \mathbb{O}$, $X \Rightarrow Y$, $XS \notin \mathbb{E}$ for all $S \in \mathbb{A}^+$ then $\|X\| = \|Y\|$.*

If we informally interpret the relation $X \overset{+}{\Rightarrow} Y$ as "X inherits Y" (or "X is a particular case of Y" or "X is a Y") and treat the objects $X\alpha$ as "properties of X," then in case $X \overset{+}{\Rightarrow} Y$ the object X should in a sense inherit the properties of Y and optionally make them more concrete and enlarge their totality. Formally, this requirement amounts to the following:

$$\text{if } X, Y \in \mathbb{O} \text{ and } X \overset{+}{\Rightarrow} Y \text{ then } \|X\| \supseteq \|Y\|. \qquad (*)$$

Introduce the following named condition:

$$\begin{array}{l}\text{If } X, Y \in \mathbb{A}^+,\ \alpha \in \mathbb{A},\ X\alpha \in \mathbb{E},\ Y\alpha \in \mathbb{O}, \text{ and } X \Rightarrow Y \\ \text{then } X\alpha \in \mathbb{O} \text{ and } \|X\alpha\| \supseteq \|Y\alpha\|.\end{array} \qquad (\textbf{CoInh})$$

Theorem 16. *Assume (PreObj). The following are equivalent:*
(1) *condition (CoInh) is satisfied;*
(2) *if $X, Y \in \mathbb{O}\backslash\{\omega\}$ and $X \Rightarrow Y$ then $\|X\| \supseteq \|Y\|$;*
(3) *if $X, Y \in \mathbb{O}$ and $X \overset{*}{\Rightarrow} Y$ then $\|X\| \supseteq \|Y\|$.*

Therefore, with (PreObj) satisfied, $(*)$ is equivalent to (CoInh).

Corollary 17. *Assume (PreObj) and (CoInh). If $X, Y, S \in \mathbb{A}^+$, $X \overset{*}{\Rightarrow} Y$, and $YS \in \mathbb{O}$ then $XS \in \mathbb{O}$ and $\|XS\| \supseteq \|YS\|$. In particular, if $Y \in \mathbb{O}$ and $X \overset{*}{\Rightarrow} Y$ then $\|Y\|_1 \supseteq \|X\|_1$ and $\|X\alpha\| \supseteq \|Y\alpha\|$.*

Object systems with attribute value typing usually satisfy the following (or analogous) requirements: Suppose that a class Y has a declared attribute α with value type τ. If x is an instance of Y then x has attribute α whose value type is equal to or more concrete than τ. Similarly, if X is a class inherited from Y then X has the inherited attribute α whose value type is equal to or more concrete than τ. If we interpret the relation $X \Rightarrow Y$ as "the object X is an instance of the class Y" or "the class X directly inherits the class Y" and treat the relation $X\alpha \to V$ as "V is the explicit value of the property $X\alpha$" or "V is the declared value class of the attribute α within the class X," then the above requirements can be formalized by the following named condition:

$$\begin{array}{l}\text{If } X, Y \in \mathbb{A}^+,\ \alpha \in \mathbb{A},\ Y\alpha \in \mathbb{O},\ X\alpha \to V, \text{ and } X \Rightarrow Y\\ \text{then } V \in \mathbb{O} \text{ and } \|V\| \supseteq \|Y\alpha\|.\end{array} \qquad (\textbf{CoVal})$$

Theorem 18. *Conditions (PreObj) and (CoVal) imply (CoInh).*

The conceptual dependence was introduced above as a relation on the set of all words. As soon as non-object words are regarded as "senseless," it is reasonable to describe dependence between objects involving objects only; namely, if a concept X depends on a concept Y, then there should be a chain of concepts (rather than arbitrary words) connecting X with Y. This principle is justified by the following theorem (see also Corollary 20).

Theorem 19. *Assume (Obj), (PreObj), and (CoInh). Given $X, Y \in \mathbb{O}$, X depends on Y if and only if there exist $X_1, \ldots, X_n \in \mathbb{O}$, $n \geqslant 1$, such that $X \rightarrowtail X_1 \rightarrowtail \cdots \rightarrowtail X_n = Y$.*

Corollary 20. *Assume (Obj), (PreObj), and (CoInh). The restriction of $\overset{+}{\rightarrowtail}$ onto $\mathbb{O}$ is the least transitive relation on $\mathbb{O}$ possessing the following three properties for all $X, Y \in \mathbb{O}$ and $\delta \in \mathbb{A}^+$:*

if $X \to Y$ then $X \overset{+}{\rightarrowtail} Y$;
if $X \to Y$ and $\delta \in \|X\| \cap \|Y\|$ then $X\delta \overset{+}{\rightarrowtail} Y\delta$;
if $\delta \in \|X\|$ then $X\delta \overset{+}{\rightarrowtail} X$.

The last assertion shows that, with (Obj), (PreObj), and (CoInh) satisfied, the conceptual dependence relation between objects can be described in full conformity with the initial definition of this relation on the set of all words. The only distinction consists in the fact that the latter description does not go beyond the set of objects.

Remark 21. In Theorem 19 and Corollary 20, none of the conditions (Obj), (PreObj), or (CoInh) can be omitted.

3 Algorithmization

The rest of the paper is devoted to the effective verification of various properties of rewriting systems under consideration, and the following theorem is the main step in this direction.

Theorem 22. *Given a system, put $\mu := \max\{|E| : E \in \mathbb{E}\}$. A word X is infinitely rewritable if and only if one of the following two (mutually exclusive) conditions holds:*

(a) *there are integers $n \geqslant 0$ and $r > 0$ such that $X^{(n)} = X^{(n+r)}$;*
(b) *there are integers $n \geqslant 0$ and $r > 0$ such that*

$$\mu \leqslant \left|X^{(n)}\right| \leqslant \left|X^{(n+1)}\right|, \ldots, \left|X^{(n+r)}\right|,$$
$$X^{(n)} \neq X^{(n+r)} \qquad X^{(n)}\restriction_\mu = X^{(n+r)}\restriction_\mu.$$

In case (a) we have $X \overset{*}{\Rightarrow} \underbrace{X^{(n)} \Rightarrow \cdots \Rightarrow X^{(n+r-1)}}_{period} \Rightarrow X^{(\#)} \Rightarrow \cdots$.

In case (b) put $Y = X^{(n)}\restriction_\mu$ and let $S \in \mathbb{A}^*$ be such that $X^{(n)} = YS$. Then there is a word $R \in \mathbb{A}^+$ such that $Y^{(r)} = YR$ and the rewriting sequence $\langle X^{(0)}, X^{(1)}, \ldots \rangle$ contains a subsequence constituted by the words $X^{(n+mr)} = YR^mS$, $m \geqslant 0$, each of which starts a regular "growth period" of length r:

$$\begin{array}{lllll} X \overset{*}{\Rightarrow} X^{(n)} & = YS & \Rightarrow Y^{(1)}S & \Rightarrow \cdots \Rightarrow & Y^{(r-1)}S \\ \Rightarrow X^{(n+r)} & = YRS & \Rightarrow Y^{(1)}RS & \Rightarrow \cdots \Rightarrow & Y^{(r-1)}RS \\ \Rightarrow X^{(n+2r)} & = YR^2S & \Rightarrow Y^{(1)}R^2S & \Rightarrow \cdots \Rightarrow & Y^{(r-1)}R^2S \\ \Rightarrow \cdots & & & & \\ \Rightarrow X^{(n+mr)} & = YR^mS & \Rightarrow Y^{(1)}R^mS & \Rightarrow \cdots \Rightarrow & Y^{(r-1)}R^mS \\ \Rightarrow \cdots . & & & & \end{array}$$

In particular, $\{X^{(n)}, X^{(n+1)}, \dots\} = \{Y^{(j)}R^mS : 0 \leqslant j < r,\ m \geqslant 0\}$.

Let $\mathcal{P}$ be an arbitrary set of "constructive entities" (i.e., a set whose elements can be used as inputs for algorithms) and let $C(Y,p)$ be a condition imposed on words $Y \in \mathbb{A}^+$ with additional parameters in $\mathcal{P}$. Formally we may assume that C is a subset of $\mathbb{A}^+ \times \mathcal{P}$ and, for all $Y \in \mathbb{A}^+$, $p \in \mathcal{P}$, the expression $C(Y,p)$ means the containment $\langle Y,p\rangle \in C$.

Given $C(Y,p)$ as above, introduce the condition $C'(Y,R,S,p)$ for $Y,R \in \mathbb{A}^+$, $S \in \mathbb{A}^*$, $p \in \mathcal{P}$ as follows:

$C'(Y,R,S,p)$ if and only if there is an $m \geqslant 1$ such that $C(YR^mS,p)$.

Say that $C(Y,p)$ is *cyclically decidable* if the following two conditions hold:

(a) there is an algorithm verifying $C(Y,p)$ for $Y \in \mathbb{A}^+$, $p \in \mathcal{P}$;

(b) there is an algorithm verifying $C'(Y,R,S,p)$ for $Y,R \in \mathbb{A}^+$, $S \in \mathbb{A}^*$, $p \in \mathcal{P}$.

Note that (a) does not in general imply (b), which fact can be derived from existence of a recursive set $C \subseteq \mathbb{N}^2$ such that the set $\{n \in \mathbb{N} : (\exists\, m \in \mathbb{N})\ \langle m,n\rangle \in C\}$ is not recursive (see, for instance, [2, Chapter C.1, §6]).

Let $C(Y,p)$ be a cyclically decidable condition. Theorem 22 justifies the following simple algorithm which, given a system, a word $X \in \mathbb{A}^+$, and a parameter p, verifies existence of a word $Y \in \mathbb{A}^+$ such that $X \overset{*}{\Rightarrow} Y$ and $C(Y,p)$.

Algorithm 23.

- *If $C(X,p)$, return* ***Yes****. If X is not rewritable, return* ***No****.*
- *Otherwise, subsequently calculate the rewrites $X^{(1)},\dots,X^{(i)},\dots$ and, at each step $i \geqslant 1$, subsequently analyze the fragments $\langle X^{(n)}, X^{(n+1)}, \dots, X^{(n+r)}\rangle$ for $0 \leqslant n < n+r = i$, as follows:*
  - *If $C\big(X^{(n+r)},p\big)$, return* ***Yes****. If $X^{(n)} = X^{(n+r)}$, return* ***No****.*
  - *If $\langle X^{(n)}, \dots, X^{(n+r)}\rangle$ satisfies condition (b) of Theorem 22,*
    put $Y := X^{(n)}\restriction_\mu$; let $S \in \mathbb{A}^*$ be such that $X^{(n)} = YS$;
    let $R \in \mathbb{A}^+$ be such that $Y^{(r)} = YR$ (such an R exists by Theorem 22)*;*
    if $C'(Y,R,S,p)$, return ***Yes****; otherwise return* ***No****.*
  - *If $X^{(n+r)}$ is not rewritable, return* ***No****. Otherwise proceed to the next step, $i+1$.*

By Theorem 22, the above procedure terminates for every input.

As is easily seen, given a system $\mathcal{S}$, the condition $C(Y,\mathcal{S}) =$ "Y is not rewritable within $\mathcal{S}$" is cyclically decidable. Therefore, specialized with this condition, Algorithm 23 verifies finite rewritability of a given word within a given system. A simplified version is presented below.

Algorithm 24.

Start subsequent calculation of the rewrites $X^{(0)}, X^{(1)}, \dots$. If a nonrewritable word $X^{(i)}$ occurs, return ***Yes****. Otherwise, according to Theorem 22, a fragment $\langle X^{(n)}, X^{(n+1)}, \dots, X^{(n+r)}\rangle$ will occur which satisfies (a) or (b) of Theorem 22; in this case return* ***No****.*

Since the set $\mathbb{E}$ of explicit words is finite, Theorem 7 implies that *(Fin) is effectively verifiable*: it suffices to apply Algorithm 24 to all explicit words.

The specialization of Algorithm 23 with the condition $C(Y) =$ "$Y = \omega$" checks *if a given word is an object*. (This can be also verified by a slight modification of Algorithm 24.) It is now clear that *(Obj) and (PreObj) are effectively verifiable*.

Note that if (Fin) holds, the containment $X \in \mathbb{O}$ can be trivially verified: just check if the (finite) rewriting sequence of X ends with ω. In addition, if (Obj) holds, we can stop calculating the rewriting sequence of X if $X^{(n)} \in \mathbb{E}$ at some step $n \geqslant 0$; furthermore, if both (Obj) and (PreObj) hold, the calculation can be terminated if some $X^{(n)}$ becomes a prefix of any explicit word (see Proposition 9).

As is easily seen, the condition $C(Y,X) =$ "$Y \sqsupseteq X$" is cyclically decidable. Therefore a properly specialized version of Algorithm 23 checks *if a given word X is recurrent*. By Theorem 5, we conclude that *(Rec) is effectively verifiable*: to check if all words are not recurrent, it suffices to apply the algorithm to all words over $\mathbb{A}_\mathbb{E}$ of length at most μ. (However, the resultant verification occurs exponential-time; see Remark 6.)

Another approach to verifying (Rec) can be based on Theorem 8 which states that (Rec) implies (Fin). Check (Fin) first. If it fails then (Rec) also fails. If (Fin) holds true, condition (Rec) can be verified by processing all words X over $\mathbb{A}_\mathbb{E}$ of length at most μ and returning **Yes** whenever an X occurs such that $X \sqsubseteq X^{(n)}$ for some $n \geqslant 1$.

By Theorem 3, condition (Con) can be effectively verified by constructing the conceptual scheme and checking (during the construction) if the scheme is acyclic. The algorithm described below checks if a given system is conceptually consistent. If it is so, the algorithm returns the conceptual scheme of the system; otherwise it returns

an example of a cycle in the conceptual scheme. The algorithm uses a variable directed graph $\Gamma = \langle \Gamma_{\mathrm{N}}, \Gamma_{\mathrm{A}} \rangle$ (with Γ_{N} the nodes and Γ_{A} the arcs) and a variable $\mathcal{A}$ whose values are finite subsets of $\mathbb{A}^+ \times \mathbb{A}^+$.

Algorithm 25.

(1) *Put* $\Gamma_{\mathrm{N}} := \mathbb{E}$, $\Gamma_{\mathrm{A}} := \varnothing$.
(2) *Put* $\mathcal{A} := \{\langle X, Y \rangle : X$ *is a sink of* Γ *and* $X \rightarrowtail Y\}$.
(3) *If* $\mathcal{A} = \varnothing$, *claim that the system is* ***conceptually consistent*** *and return* Γ *as the* ***conceptual scheme****.*
(4) *Otherwise do the following for each pair* $\langle X, Y \rangle \in \mathcal{A}$*:*
    if $X = Y$ *or* Γ *contains a path from* Y *to* X*,*
        claim that the system is ***conceptually inconsistent*** *and return a* ***cycle****:* $X \rightarrowtail X$ *or* $Y \rightarrowtail \cdots \rightarrowtail X \rightarrowtail Y$*;*
    otherwise put $\Gamma_{\mathrm{N}} := \Gamma_{\mathrm{N}} \cup \{Y\}$, $\Gamma_{\mathrm{A}} := \Gamma_{\mathrm{A}} \cup \{\langle X, Y \rangle\}$.
(5) *Go to (2).*

When applied to a conceptually inconsistent system, Algorithm 25 returns an example of a cycle $X_0 \rightarrowtail X_1 \rightarrowtail \cdots \rightarrowtail X_m = X_0$ in the conceptual scheme, but it is not guaranteed that all words X_i in the cycle are objects (even in case X_0 is an object). On the other hand, Theorem 19 implies that, with (Obj), (PreObj), and (CoInh) satisfied, every path $X_0 \rightarrowtail X_1 \rightarrowtail \cdots \rightarrowtail X_m$ between objects X_0, X_m can be transformed into a path of *objects* $Y_0 \rightarrowtail Y_1 \rightarrowtail \cdots \rightarrowtail Y_n$, with $Y_0 = X_0$ and $Y_n = X_m$. We note that such a transformation can be performed *effectively*.

By slightly modifying Algorithm 25, we can obtain a procedure for constructing the weak rewriting scheme rather than the conceptual scheme. The algorithm presented below checks in an arbitrary system if there are weakly recurrent words, and either returns an example of such a word or constructs the weak rewriting scheme of the system.

Algorithm 26.

(1) *Put* $\Gamma_{\mathrm{N}} := \mathbb{E}$, $\Gamma_{\mathrm{A}} := \varnothing$.
(2) *Put* $\mathcal{A} := \{\langle X, Y \rangle : X$ *is a sink of* Γ *and* $X \Rightarrow_{\mathrm{w}} Y\}$.
(3) *If* $\mathcal{A} = \varnothing$, *claim that* ***there are no weakly recurrent words*** *and return* Γ *as the* ***weak rewriting scheme****.*
(4) *Otherwise do the following for each pair* $\langle X, Y \rangle \in \mathcal{A}$*:*
    if $X \sqsubseteq Y$ *or* Γ *contains a path from a prefix* $Y_0 \sqsubseteq Y$ *to* X*,*
        return a ***weakly recurrent word****:*
        $X \Rightarrow_{\mathrm{w}} Y \sqsupseteq X$ *or* $Y_0 \Rightarrow_{\mathrm{w}} \cdots \Rightarrow_{\mathrm{w}} X \Rightarrow_{\mathrm{w}} Y \sqsupseteq Y_0$*;*
    otherwise put $\Gamma_{\mathrm{N}} := \Gamma_{\mathrm{N}} \cup \{Y\}$, $\Gamma_{\mathrm{A}} := \Gamma_{\mathrm{A}} \cup \{\langle X, Y \rangle\}$.
(5) *Go to (2).*

According to Theorem 3, the weak rewriting scheme of a conceptually inconsistent system includes a path $X_0 \overset{+}{\Rightarrow}_{\mathrm{w}} X_n \sqsupseteq X_0$ with $X_0 \in \mathbb{E}$. Given an inconsistent system, Algorithm 26 returns an example of a path $Y_0 \overset{+}{\Rightarrow}_{\mathrm{w}} Y_n \sqsupseteq Y_0$, but the leading word Y_0 need not be explicit. In this connection it is worth noting that every path of the form $Y_0 \overset{+}{\Rightarrow}_{\mathrm{w}} Y_n \sqsupseteq Y_0$ can be *effectively* transformed into a path $X_0 \overset{+}{\Rightarrow}_{\mathrm{w}} X_n \sqsupseteq X_0$ (of the same length) with $X_0 \in \mathbb{E}$.

To the author's opinion, the most convincing indication of conceptual inconsistency is an example of some cycle $X_0 \rightarrowtail \cdots \rightarrowtail X_n = X_0$ which is constituted by *objects* and starts with an *explicit* word X_0. We note that, if an inconsistent system satisfies (Obj), (PreObj), and (CoInh), then a cycle of this kind can be found *effectively*.

We now turn to the effective analysis of object types.

Say that $C \in \mathbb{O}$ is a *character* if either $C = \omega$ or $C \sqsubset E$ for some $E \in \mathbb{E}$. Let $\mathbb{C}$ be the set of all characters. It is clear that $\mathbb{C}$ is finite.

Given an object X, consider the rewriting sequence $X^{(0)} \Rightarrow \cdots \Rightarrow X^{(n)} = \omega$ and denote by $\mathrm{ch}(X)$ the first character in the sequence $\langle X^{(0)}, \ldots, X^{(n)} \rangle$. Call $\mathrm{ch}(X)$ the *character of* X.

It is clear that $\mathrm{ch}(C) = C$ for all $C \in \mathbb{C}$. Therefore, $\mathbb{C} = \{\mathrm{ch}(X) : X \in \mathbb{O}\}$.

Theorem 27. *For every object* X *we have* $\|X\| = \|\mathrm{ch}(X)\|$.
Consequently, $\{\|X\| : X \in \mathbb{O}\} = \{\|C\| : C \in \mathbb{C}\}$, *and the set* $\{\|X\| : X \in \mathbb{O}\}$ *is finite.*

By the *type comparison problem* we mean the following: given two objects X and Y, effectively compare the types of X and Y, i.e., determine which of the relations $\|X\| = \|Y\|$, $\|X\| \subseteq \|Y\|$, $\|X\| \supseteq \|Y\|$ hold.

It is clear that the character of an object can be effectively determined. Therefore, due to Theorem 27, the type comparison problem reduces to effective comparison of the character types.

Given $X \in \mathbb{C}$ and $\alpha \in \|X\|_1$, introduce the notation $X_\alpha := \mathrm{ch}(X\alpha)$ and, given an arbitrary function $\tau : \mathbb{C} \to \{1, \dots, N\}$, define the following equivalence relation on $\mathbb{C}$:

$$X \underset{\tau}{\sim} Y \quad \text{if and only if} \quad \|X\|_1 = \|Y\|_1 \text{ and } \tau(X_\alpha) = \tau(Y_\alpha) \text{ for all } \alpha \in \|X\|_1.$$

The following algorithm uses two variable functions $\tau, \tilde{\tau} : \mathbb{C} \to \{1, 2, \dots\}$ (each of which can be encoded as an array of naturals indexed by the characters).

Algorithm 28.

(1) *Define* τ *by putting* $\tau(X) := 1$ *for all* $X \in \mathbb{C}$.
(2) *If* $X \underset{\tau}{\sim} Y$ *for all* $X, Y \in \mathbb{C}$ *such that* $\tau(X) = \tau(Y)$, ***return*** τ.
(3) *Assign* $\tilde{\tau}$ *a copy of* τ.
(4) *For each* $k \in \big\{\tau(X) : (\exists\, Y \in \mathbb{C})\big(X \underset{\tau}{\nsim} Y \;\&\; \tau(X) = \tau(Y)\big)\big\}$ *do the following:*
    arbitrarily enumerate the set $\{X \in \mathbb{C} : \tau(X) = k\}$ *thus making it a sequence* $\langle X_1, \dots, X_n\rangle$, $n \geqslant 2$, *and, for each* $i = 2, \dots, n$, *do the following:*
        if $X_i \underset{\tau}{\sim} X_j$ *for some* $1 \leqslant j < i$, *reassign* $\tilde{\tau}(X_i) := \tilde{\tau}(X_j)$;
        otherwise reassign $\tilde{\tau}(X_i) := \max\{\tilde{\tau}(X) : X \in \mathbb{C}\} + 1$.
(5) *Assign* $\tau := \tilde{\tau}$ *and go to (2).*

Theorem 29. *Algorithm 28 halts for any system and, if the system meets (PreObj), the resultant function* $\tau : \mathbb{C} \to \{1, \dots, N\}$ *is such that* $\tau(X) = \tau(Y)$ *if and only if* $\|X\| = \|Y\|$.

Remark 30. Algorithm 28 is analogous to Vizing's algorithm of partitioning the vertex set of a graph into classes of similar vertices (see [3]). The author is grateful to S. V. Avgustinovich for discovering the analogy.

The following algorithm uses two variable sets $\Sigma, \Sigma_0 \subseteq \mathbb{C}^2$.

Algorithm 31.

(1) *Put* $\Sigma := \mathbb{C}^2$.
(2) *Put* $\Sigma_0 := \big\{\langle X, Y\rangle \in \Sigma : \|X\|_1 \subseteq \|Y\|_1 \;\&\; (\forall\, \alpha \in \|X\|_1)\, \langle X_\alpha, Y_\alpha\rangle \in \Sigma\big\}$.
(3) *If* $\Sigma_0 = \Sigma$, ***return*** Σ. *Otherwise reassign* $\Sigma := \Sigma_0$ *and go to (2).*

Theorem 32. *Given any system, Algorithm 31 always halts and, if the system meets (PreObj), the resultant set* Σ *equals* $\{\langle X, Y\rangle \in \mathbb{C}^2 : \|X\| \subseteq \|Y\|\}$.

Therefore, given a system subject to (PreObj), we can effectively verify the relation $\|X\| \subseteq \|Y\|$ for $X, Y \in \mathbb{O}$. (As is easily seen, this implies that *(CoInh) and (CoVal) are effectively verifiable.*) However, the mere claim "$\|X\| \nsubseteq \|Y\|$" is not always sufficient, and one may require a particular reason why the inclusion fails. Within a bulky system, it may occur nontrivial to find a particular detail $\delta \in \|X\| \backslash \|Y\|$, and a corresponding algorithm would thus be a useful troubleshooting tool. Recall that, due to Theorem 27, it suffices to automate the solution for characters only.

The following algorithm uses a variable set $\Delta \subseteq \mathbb{C}^2$ and a variable function $\delta : \Delta \to \mathbb{A}^+$.

Algorithm 33.

(1) *Put* $\Delta := \varnothing$ *and* $n := 1$.
(2) *For all pairwise distinct* $X, Y \in \mathbb{C}$ *do the following:*
    if there is an $\alpha \in \|X\|_1 \backslash \|Y\|_1$, *add* $\langle X, Y\rangle$ *to* Δ *and assign* $\delta(X, Y) := \alpha$.
(3) *For all pairwise distinct* $X, Y \in \mathbb{C}$ *such that* $\langle X, Y\rangle \notin \Delta$ *do the following:*
    if there is an $\alpha \in \|X\|_1$ *such that* $\langle X_\alpha, Y_\alpha\rangle \in \Delta$ *and* $|\delta(X_\alpha, Y_\alpha)| = n$,
    add $\langle X, Y\rangle$ *to* Δ *and assign* $\delta(X, Y) := \alpha\, \delta(X_\alpha, Y_\alpha)$.
(4) *If there were no assignments at step (3),* ***return*** δ.
    Otherwise put $n := n + 1$ *and go to (3).*

Theorem 34. *Given any system, Algorithm 33 always halts and, if the system meets (PreObj), the resultant function* $\delta : \Delta \to \mathbb{A}^+$ *is such that*

$\Delta = \{\langle X, Y\rangle \in \mathbb{C}^2 : \|X\| \nsubseteq \|Y\|\}$,
$\delta(X, Y) \in \|X\| \backslash \|Y\|$ *for all* $\langle X, Y\rangle \in \Delta$,
$|\delta(X, Y)| = \min\{|\delta'| : \delta' \in \|X\| \backslash \|Y\|\}$ *for all* $\langle X, Y\rangle \in \Delta$.

Acknowledgments. The general idea of representing objects by means of property paths and evaluating them with prefix rewriting belongs to Alexander Nikolaevich Ageev. The author is also grateful to Sergei Vladimirovich Avgustinovich for fruitful discussions.

References

1. Salomaa, A.: Formal Languages. Academic Press, New York (1973)
2. Barwise, J. (ed.): Handbook of Mathematical Logic. North-Holland, Amsterdam (1977)
3. Vizing, V.G.: Distributive Coloring of Graph Vertices. Diskretn. Anal. Issled. Oper. 2(4), 3–12 (1995)

Knowledge-based System for Software Requirements Analysis and Management

Natalja Pustovalova, Maxim Bakaev, Tatiana Avdeenko.

Novosibirsk State Technical University, Department of Economic Informatics
+7-383-346-06-79
natalja.ru@gmail.com, maxis81@gmail.com, tavdeenko@mail.ru

Abstract. The paper describes the work in progress towards the development of knowledge-based system (KBS) to support developers in ensuring the quality of software requirements. Ambiguity, contradictoriness, incompleteness, poor traceability and adjustability were identified as currently most severe problems with requirements, as well as certain ignorance towards existing validation and verification techniques. In an approach to solving these problems, frame-based model has been chosen for knowledge base. Two main advantages of frame-based model are declarative nature and procedural features. First lets to set requirements and relations, second allows for determine procedures of models' conversations. An example of frame instance is presented.

Keywords: requirement management tools, frame model, knowledge engineering, information extraction

1 Introduction

It has almost become a commonplace that a significant share of software development projects fails or exceeds their budget. There is no lack of research on the causes of runaway projects, and unstable or incomplete requirements and poor estimations are justifiably identified as the two major ones [1; 2, p.5]. The former of the two is clearly the primary factor, and it is also widely noted that requirements errors are the most expensive to correct late in the development process or when the product is released.

Certain desired properties for well-defined requirements are described in international standards related to software development: correspondence to the needs and expectations of users and stakeholders, unambiguity for different domain experts, consistency, lack of contradictions, completeness and etc. [3]. An extensive list of approaches and methodologies concerning requirements analysis and management includes SWEBOK, PMBOOK, RUP, DOORS, CORE and so on (review is available in [3]), however there seem to be certain factors hindering their ultimate effectiveness:

1. The discrepancy on the conceptual level. There are different definitions of requirements, compiled by different experts from their diverse experience [4, 5]. Thus, methodologies are discriminate in their understanding of a requirement, so they may mark out different levels and types of requirements, leading to radical distinctions in the development process.
2. The absence of reliable means for choosing the most appropriate approach for a given project. To make a justified choice, developer would have to be an expert in these various theoretical paradigms, as well as software development methodologies. In practice, the choice depends either on the developers' previous experience, or even on requests from stakeholders, who may be quite inept in software engineering.
3. The lack of accordance on procedures and criteria for assessing the final quality of the requirements. Although the methods for requirements' validation and verification exist, they hardly provide clear control points where the quality of the requirements set could and should be estimated before the construction of the product or its prototype has started.

These factors, in the light of objective complexity of software engineering domain, lead to such generally recognized problems related to requirements:

1. Requirements elicitation and formalization that result in ambiguous or unobvious requirements. Omitted requirements are, probably, even greater problem, as they result in omitted logic in the final product, and this kind of error, according to some estimates [1, p.75], constitute a prominent 30% of all persistent software errors.
2. Poor requirements traceability and adjustability, due to lack of established relationships between them. Nowadays, unstable requirements are less and less viewed as the problem of weak management, but more accepted as objective reality [1, p.69] – thus proper requirements management techniques have to deal with introducing changes to the requirement set.
3. Persistence of requirements errors, which significantly increase risks and costs for the project. Many requirements validation and verification techniques, such as reviews, consistency checks, etc., rely heavily on expertise, however developers are often pressed by project estimates and schedules that generally allocate few time on their involvement in requirement-cleansing [1, p.72].

As an approach to solving these problems we propose knowledge-based system (KBS) for requirements analysis.

The main peculiarity of the system is that it uses frame-based model both to represent declarative knowledge about requirements in natural form and to use procedural possibilities of this model for determining transformations between different types of requirements.

2 Knowledge representation model

Currently, popular requirements management tools, such as Rational IBM or Borland products, offer means for managing requirements in the course of the project and integrating interim results of the development. However, effective use of these tools, including requirements quality assurance, implies relatively high level of expertise, as domain knowledge is mostly presented in user manual or online help only. Thus, the application of such products is not accessible for all developers or, given their considerable cost, economically feasible in all projects.

As we believe the requirements analysis and management domain matches generally accepted premises for expert systems deployment (e.g., see in [6, p.36]), we suggest the application of knowledge engineering methods to develop a knowledge-based system (KBS) aiding developers in ensuring the quality of requirements. The system, used by developers (requirements analysts) should be able to decrease the probability of errors committed by inexperienced developers as well as alleviate workload on skilled ones, by providing support in requirements elicitation, specification, validation and their management throughout the development cycle.

As a model of knowledge representation for KBS of the requirements analysis, a frame-based model has been selected. Frame is knowledge representation scheme, focused on the inclusion into a strictly organized data structure the implicit information relationships existing in the domain. This arrangement supports the organization of knowledge into more complex units, which represent the structure of domain objects.

In accordance with [7], frame can be considered a static data structure used to represent well-understood stereotypical situations. Perhaps our own knowledge about the world is also organized as frame-like structures. We adapt to each new situation, recalling information based on experience. Then we adapt this knowledge according to this new situation.

Why do we consider frame-based model the most suitable for implementation of KBS for requirements analysis? First, the frames, as well as semantic networks, let to organize the hierarchy of knowledge, which allows a natural way to describe the levels of requirements recommended by experts. Generally, three levels of requirements are marked out [8] (See Appendix).

1. At the top level, we have the so-called business requirements. Business requirements are usually formulated by top managers or shareholders of the company.
2. The next level – the level of user requirements. Example of user requirements: the system should provide interactive tools for entering complete information about the order, followed by saving it in database and routing information about the order to the person responsible for planning and execution. User requirements are often poorly structured, duplicative, or contradictory. Therefore, the third level is very important in which the formalization of requirements is carried out.
3. Third level – functional and technical requirements, it depends on the system scale. Example of functional requirements (or simply functions): the order can be created, edited, deleted and moved.

It is clear that the above levels can be represented as a hierarchy of requirements with the relevant property inheritance. However, the relations between the individual information units may not necessarily be relations on inheritance, there are different kinds of them. For example, there may exist relations “Conflicting requirements”, “Related requirements”, etc., between instances of the frame “Requirement”. Such relations, as well as hierarchical ones, may be represented by the semantic network model. But the use of frame-based model can make it in the most natural way. This uses the internal structure of the frame (the so-called slots). For this purpose, special slots of relations are used. These types of slots have the names that are the names of relations, and values are the names of the frames with which a given frame is connected. Frame model allows a convenient way to organize working with scripts. For example, KBS for requirements analysis may be arranged in the form of a script as follows. In the initial moment of time slots of frames of the knowledge base (KB) are either empty or filled with default values. In the process of accumulating information from stakeholders and users of the designed system, the KB is filled with instances of requirements. In this case, along with the declarative semantics it uses procedural semantics of frame-based model. For example, the value of a slot may be a name of a program, checking for conflict of the newly introduced requirement with requirements already existing in the KB. This program begins to work automatically after the confirmation of entering the new requirement into the system. The result of this work may be, for example, the modification of slots of the frame “Conflicting requirements”.

3 Results

3.1 Software structure and frame model

Figure 1 illustrates relations between different models and views that are used in the process of software development. Frame model of requirements and domain problem model, both are crucial, because they are the base for others. All model transformations are defined as procedures and receive data from slots of frame instances.

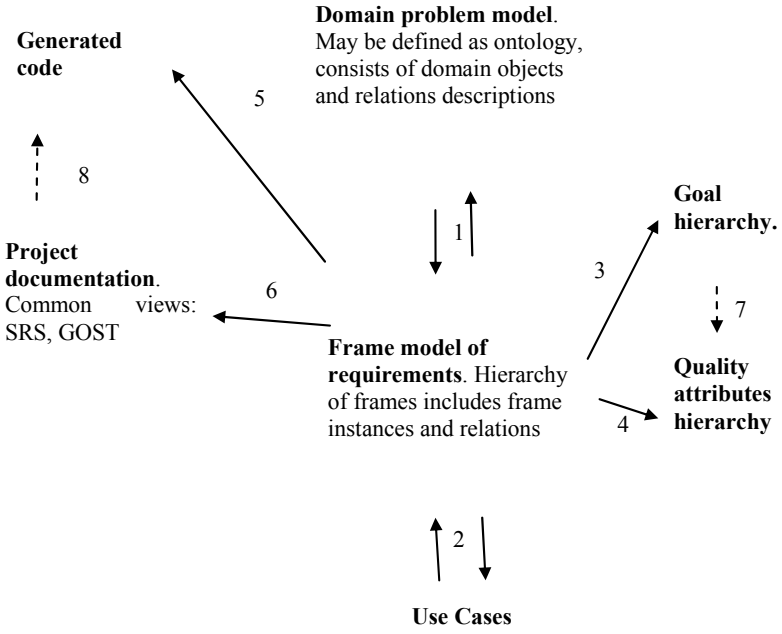


Fig. 1. Infological model

Fig. 2 illustrates common structure of frame instance. Slots of this structure are used for procedural transformation models. For example, data from slot *Detailed requirement formulation* may be used for creation of Use Cases.

Slot name	
Common requirement formulation	
Index	
Objective	
Detailed requirement formulation	Actors
	Process
	Documents
	Database
Parent-child requirements	
Related requirements	Conflicting
	Duplicated
Requirement owner	
Requirement holder	
Requirement attributes	Core/ periphery
	Complexity of implementation
	Priority
	Criticality of implementation
	Costs
Objective achievement measure	
Lead time estimate	Designing deadline
	Programming deadline
	Testing and debugging deadline
	Documentation deadline

	Total time
Lead time actual	Designing deadline
	Programming deadline
	Testing and debugging deadline
	Documentation deadline
	Total time
Resources estimate	Designing resources
	Programming resources
	Testing and debugging resources
	Documentation resources
	Total resources
Resources actual	Designing resources
	Programming resources
	Testing and debugging resources
	Documentation resources
	Total resources
Current point of implementation	Life-cycle phases
	Level of implementation
Versions and revisions	Version number
	Changes
	Changes owner
Optionally	

Fig. 2. Common frame structure

3.2 The system capabilities

The system has the following principal capabilities:

1. Controls the fill-in of the essential attributes for requirements, allows setting pre-defined relations between requirements.
2. Class inheritance offers possibilities for reusing and customizing frame hierarchy for users' needs and project context. (See Figure 3 for class "Requirement" structure).

Class «Requirements»
- Subclass «Concept»
- Subclass «Business requirement»
- Subclass «User requirement»
- Subclass «Technical requirement»

Fig. 3. Structure of "Requirements" class

3. Allows modifications to the requirements, traces change history, notifies on possible effects on related requirements.
4. Provides visualization for the requirements set (tables, tree view, technical documentation, etc.) as well as for related concepts, in particular for running requirements reviews with stakeholders.
5. Helps tracking the degree of the requirements' fulfillment during modeling, simulation or prototyping.
6. Checks for inconsistencies in requirements priorities, development order, pre-requisite requirements; offer conflict resolution strategies (e.g. based on priorities of involved stakeholders).
7. Tracks development estimations vs. real work efforts, builds up statistics for analysis and improved estimating and product release date forecasting.

3.3 Requirements analysis

Further development of the system implies the extension of the procedural component for requirements analysis, so that the system: auto-discovers such relationships between requirements as *Child-parent*, *Related*, *Contradicting*, *Duplicate*; hints on possibly omitted requirements; checks for ambiguous wordings in requirements descriptions.

For that, we plan to utilize information extraction, text mining and concept mining techniques, with external domain or common sense ontologies or vocabularies (such as Russian WordNet) providing thesaurus (see [9], [10]).

4 Conclusions

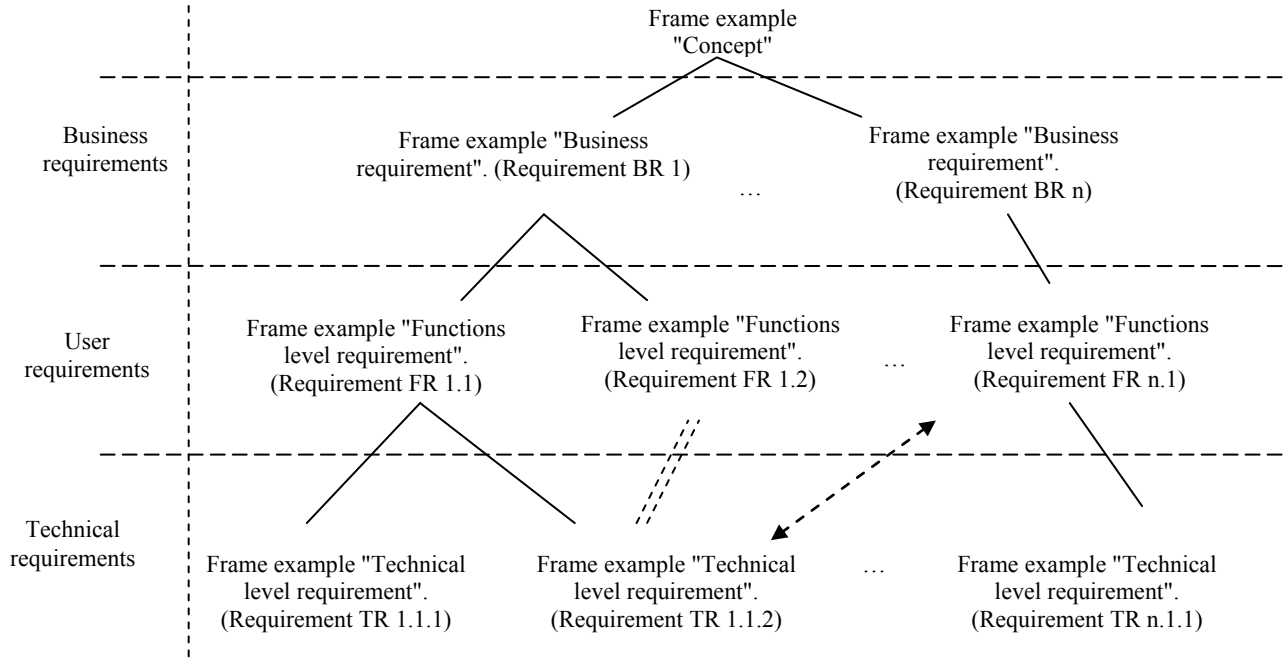
In our paper, we sought to reason that application of AI methods in requirements analysis and management domain could contribute to resolving problems persisting in software development, such as ambiguous, contradicting or incomplete requirements, poorly suited for adjusting or tracing. The proposed KBS built based on frame model includes capabilities to aid in control of the requirements quality and provides means for better traceability that leads to advantages on various stages of the development. Further, with respect to requirements validation and verification, the system both offers abundant visualization aimed on ensuring stakeholder involvement (which is major factor for the project success [2, p.4]), and supplies the foundation for automatic requirements analysis.

The developed frame-based knowledge structure for the KBS may serve as conceptual model of requirements analysis domain, and in turn get used by other systems related to software development. So, the *Requirement* frame and related concepts were utilized in an adjacent project in Novosibirsk State Technical University: KBS used in solving practical tasks in web interface design, based on such requirements as business goals of the project and target user attributes [11].

References

1. R.L. Glass. Facts and Fallacies of Software Engineering. Addison Wesley, 2003.
2. Standish Group International. The Chaos Report, 1995.
3. V. Kulyamin, N. Pakulin, O. Petrenko, A. Sortov, A. Khoroshilov. Formalizatsiya trebovanii na praktike (in Russian). Preprint 13, ISP RAN, Moscow, 2006.
4. E. Hull, K. Jackson, J. Dick. Requirements Engineering. SpringerVerlag, 2004.
5. D. Leffingwell, D. Widring. Managing software requirements: a unified approach. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, 1999.
6. T.A. Gavrilova, V.F. Khoroshevsky. Bazi znanii intellektualnikh sistem (in Russian). Piter, St.Petersburg, Russia, 2000.
7. . Minsky. A framework for representing knowledge. In The Psychology of Computer Vision. McGraw-Hill: P. Winston, 1975.
8. Y.A. Maglinets. Requirements analysis to automated information systems, http://www.intuit.ru/department/itmngt/analisis/2/.
9. A. Maedche, S. Staab. Mining Ontologies from Text. Lecture Notes in Computer Science, 1937: 169-189, 2000.
10. I. Spasic, S. Ananiadou, J. McNaught, and A. Kumar. Text mining and ontologies in biomedicine: Making sense of raw text. Briefings in Bioinformatics 6(3): 239 – 251, 2005.
11. M.Bakaev, T. Avdeenko. Knowledge-Based System for Web Interface Design. Proc. of 2nd International Conference on Information and Multimedia Technology (ICIMT 2010), Hong Kong, China, Dec 28-30, 2010, vol. 3, pp. 262-266.

Appendix


Frame example
"Concept"
Business
requirements
Frame example "Business
requirement". (Requirement BR 1)
…
Frame example "Business
requirement".
(Requirement BR n)
User
requirements
Frame example "Functions
level requirement".
(Requirement FR 1.1)
Frame example "Functions
level requirement".
(Requirement FR 1.2)
…
Frame example "Functions
level requirement".
(Requirement FR n.1)
Technical
requirements
Frame example "Technical
level requirement".
(Requirement TR 1.1.1)
Frame example "Technical
level requirement".
(Requirement TR 1.1.2)
…
Frame example "Technical
level requirement".
(Requirement TR n.1.1)

Rational Agents at the Marketplace*
(extended abstract)

N.V. Shilov and N.O. Garanina

A.P. Ershov Institute of Informatics Systems,
Lavren'ev av., 6, Novosibirsk 630090, Russia,
{garanina, shilov}@iis.nsk.su

Abstract. We study multiagent algorithms for the following problem: There are buyers (customers) and salesmen at the marketplace; all customers are rational agents; in contrast, all salesman are not agents, but every salesman has individual price for every buyer; the problem is to define rationality-based protocol of pairwise negotiation, flips/swaps (of salesmen) between agents that leads every agent (1) to knowledge about its individual salesman, and (2) to knowledge that it is impossible to improve total price to be paid by swapping salesmen. We call this problem *Rational Agents at the Marketplace*. The problem is related to the well-known *Assignment Problem* in Graph Theory, and to *Mars Robot Puzzle* that we have examined to some extend previously.

1 Introduction

A *multiagent system* is a distributed system [5] that consists of *agents*. An agent is an autonomous reactive object (in OO-sense) whose internal states can be characterized in terms of *Believes* (B), *Desires* (D), and *Intentions* (I). Agent's Believes represent its "knowledge" about itself, other agents and an "environment"; this "knowledge" may be incomplete, inconsistent, and (even) incorrect. Agent's Desires represent its long-term aims, obligations and purposes (that may be controversial). Agent's Intensions are used for a short-term planning. *Reactivity* means that every agent can change its Believes, Desires, and Intentions after communication and interaction with other agents (including environment), but every agent is autonomous, i.e. change of "personal" Believes, Desires and Intensions is not decreed by other agents. Agents of the described kind are usually called BDI-agents [6]. A *rational* agent has clear "preferences" and always chooses the action (in feasible actions) that leads to the "best" outcome for itself; *bounded rationality* is "decision making" limited by the cognitive abilities of agents (e.g. the finite amount of time they have to make decisions). Multiagent algorithm is a distributed algorithm [5] that solves some problem by means of cooperative work of agents in a multiagent system.

In this paper we exam the following problem that we call *Rational Agents at the Marketplace* (RAM-problem or RAMP).

> There are $m \geq 1$ customers (buyers) and $n \geq 1$ salesmen (traders) at the marketplace. Every salesman has a (single) unit of some indivisible good for sale, and every buyer wants to buy exactly one unit of the good. All salesmen are offering their units of the good individually to buyers by individual prices $\{p_{s,b} \in \mathcal{N} : 1 \leq b \leq m \text{ is a buyer}, 1 \leq s \leq n \text{ is a salesman}\}$. All buyers are rational agents that can communicate, negotiate, make concessions and swap their individual salesmen pairwise (and only pairwise) in peer-to-peer (P2P) manner so that all concessions and swaps must be rational for both participating agents.
> Problem: Design a multiagent algorithm (protocol) for negotiations and concessions, changes ("flips") of salesmen, exchanges ("swaps") of salesmen by buyer that eventually leads every buyer to knowledge about its a individual salesman with whom it can make a beneficial (i.e. rational) deal
> 1. without conflicts (concurrency) with other buyers, and
> 2. no pairwise swap can not improve (reduce) a total price payed by buyers.

This problem grew up from our study of the following *Mars Robot Puzzle* (MRP) [3].

> There are $n > 1$ autonomous agents ("robots") and (the same) number of shelters on a plane part of Mars. Locations of all shelters are fixed and known to all robots. Every robot can communicate with any other robot in P2P manner, but is not aware about the current positions of other robots.

* The research has been supported by Russian Foundation for Basic Research (grant 10-01-00532-a) and by Siberia Branch of Russian Academy of Science (Integration Grant n.2/12).

At some moment all robots fix their current positions, and have to select individual shelters to move at by a straight route. Assume that there are no any obstacle (like rocks, holes, robots and shelters, etc.) between any robot and any shelter. Definitely, robots should not collide. Hence, every individual robot can move to its shelter only when it knows for sure that it will not collide with any other robot on the route.
Problem: Design a multiagent algorithm (protocol) that guarantees that every robot will eventually know that its route to the selected shelter does not intersect with routes of other robots (and hence robots will not collide in a motion).

The difference between RAMP and MRP is manyfold. First, in RAMP agents are assumed to be rational, while in MRP agents do not care about their benefits (preferences) at all. Next, MRP has a clear geometric interpretation, but it has not clear from the very beginning, why any set intersection-free routs exists, and, hence it is not obvious that a desired protocol may exist. Fortunately, the existence of these routes can be proved by contradiction, or by reduction to the *assignment problem* in Graph Theory [1], or to the convex hull problem in Combinatorial Geometry. At the same time MRP is closely related to the path-planning problem in Artificial Intelligence. In contrast, RAMP has no geometric interpretation, but it is seems á priori, that some protocol may exist.

Nevertheless RAMP and MRP are closely related due to the core protocol SMEx for MRP [3]. This protocol is a multiagent variant of a local search algorithm for a combinatorial geometry problem suggested by E.W. Dijkstra [4]. We presented in the paper [3] a series of modifications of SMEx protocol that solve MRP problem, and proved (manually) their correctness. All these modifications belong to a class of so-called *wave algorithms* [5], since they meet the following properties:

- termination: each computation is finite;
- decision: each computation contains at least one decide event;
- dependence: each event potentially influences all computations.

The rest of the paper is organized as follows. The next section 2 sketches a protocol for "initial" choices of salesmen by buyers (i.e. that satisfies the first constraint of RAMP). The following section 3 presents a protocol for the most "rational" for buyers distribution of salesmen that is possible to arrange by pairwise swaps (i.e. that satisfies the second constraint of RAMP). Finally, we conclude in the last section 4 with a discussion of further research topics.

All our algorithms in this paper rely upon the following fairness communication assumption [3]: communication (in a multiagent system) is said to be *fair*, if every agent which would like to communicate with any other agent will communicate eventually[1]. We also assume, that the total number of customers m and the complete set of customers is *common knowledge* in this group of agents (customers) [2], and that every customer $b \in [1..m]$ knows its individual price-list (i.e. the set $P_b = \{p_{s,b} \in \mathcal{N} : 1 \leq s \leq n\}$) sorted in the ascending order (i.e. the cheapest first, the most expensive – the last). We interpret this price-list as buyer's preferences, its rationale for flipping/swapping traders. At the same time we assume bounded rationality as follows: a buyer never mind to remember data of other customers and always is looking for the most rational *local* action (i.e. just one step ahead). We use pseudocode a-lá [5] to present algorithms (protocols) in this paper.

2 Looking for a salesman

In this section we design and prove a multiagent algorithm (protocol) for rational negotiations between buyers that eventually leads every buyer to knowledge about its a individual salesman with whom it can make a deal without conflicts (concurrency) with other buyers. We assume that all customers (buyers) arrive to marketplace *simultaneously*. One may consider alternative opportunity: buyers arrive one by one or (more generally) group by group (one by one or by a single group in particular). But this "more general" situation may be reduced to the simultaneous arrival: since m (the total number of customers) is common knowledge, all buyers may just wait until all have arrived to the marketplace.

The protocol consists of the algorithm LSM (*Look for a SalesMan*) for an individual agents and a communication scheduler. At every moment every buyer has some salesman (current salesman) as its current intention; at the very beginning this intention is the salesman with the best (cheapest) price for this customer. Believes of every buyer are represented by two integer counters: NC for *Number of Conflicts* and CF for *Conflict-Free* buyers; NC represents agent's upper estimation of number of buyers with whom it could have conflicts, and,

[1] In terms of Linear Temporal Logic: **F**((agent 1 wants to talk to agent 2) **U** (agent 1 and agent 2 are talking))

respectively, CF represents its lower estimation of number of agents that have no conflicts at all; in particular, the agent believes that it does not compete for the current salesman with any other buyer as soon as $NC = 0$; the agent believes that there are competition for any salesman at the marketplace as soon as $NC = 0$ and $CF = 2 \times (m - 1)$, it.e. it believes that it has no rivals, and it checks twice that all buyers believe that they do not compete for their salesmen. But in the case when two agents (say i and j) compete for a salesman, then they play the following static game with complete information that we denote as $Flip\_Game(l_i, f_i, l_j, f_j)$:

$i \setminus j$	*bid*	*flip*
bid	(f_i, f_j)	$(0, l_j)$
flip	$(l_i, 0)$	(f_i, f_j)

Two moves (strategies) in this game are *bid* for the same salesman and *flip* to the next salesman. Before the game every participating agent $b \in \{i, j\}$ sends to the partner two integer values: l_b is its loss[2], if it flips, and its "fine" f_b for simultaneous bidding or flipping with the partner [3]. Then both agents compute mixed strategy Nash equilibrium (that is a rational behavior) and play corresponding mixed strategies until they make different moves (i.e. one flips, another bids).

Description of a fair scheduler is out of scope of our paper, pseudocode of the LSM-algorithm is presented in the Appendix A. Some important properties of this LSM-algorithm are accumulated in the following lemmas 1, 2 and 3. Their proofs will be published in a forthcoming full research report.

Lemma 1. *Assume that two agents i and j play $Flip\_Game(l_i, f_i, l_j, f_j)$ where all values l_i,f_i, l_j and f_j are negative[4]. Then Nash equilibrium in mixed strategies in $Flip\_Game(l_i, f_i, l_j, f_j)$ exists (and is unique) iff $f_1 < l_1$ and $f_2 < l_2$.*

Let remark that lines 13 – 19 of the above algorithm LSM contains game-theoretic procedure for conflict resolution between the agent and a partner. Let LSM′ be a variant of the algorithm LSM, with any procedure (non-deterministic, probabalistic or deterministic) in these lines for conflict resolution such that either the agent flips to its next salesman XOR[5] the partner flips to its next salesman.

Lemma 2. *Assume that it is common knowledge in a system that all agents execute one and the same algorithm LSM′. Then every agent in this system at every moment of time after execution of line 2 knows that all salesmen that gives smaller prise to the agent than its current salesman are intensions of some other agents.*

Let us recall that communication is said to be *fair* in a multiagent system, if every agent which would like to communicate with any other agent will communicate eventually.

Lemma 3. *Assume that all agents of a system execute one and the same algorithm LSM′, and that communication is fair in the system. Then the system eventually terminates.*

The desire of every buyer is to select in a rational way an individual salesman and to know that it can make a deal with the salesman without competition with other buyers. Lemmas 1, 2 and 3 together imply the following proposition.

Proposition 1.

If *a system with fair communication consists of $m > 0$ buyers each of which would like to make an individual deal with some of $n \geq m$ salesmen, it is common knowledge (in the system) that all buyers are agents that execute algorithm LSM, and for every buyer b its* `my_fine` *is less then* $\min\{p_{s',b} - p_{s'',b} : s', s'' \in [1..n]$ *and s'' has the next price to the price of s' in $P_b\}$,*

then *every agent will eventually terminate, it will know upon termination that nobody in the system will never compete for its current salesman* `cur_sman`*, and hence it will be able to make a deal with this salesman.*

Proof of this proposition will be published in a forthcoming full research report.

Unfortunately, we do not know yet whether the outcome of the LSM-algorithm is a Pareto-optimal assignment or not. This topic requires further research.

---

[2] This value must be negative, since the agent flip better salesman to worse salesman.

[3] This value must be negative, since *time is money.*

[4] This assumption is very natural, since losses l_i and l_j are due to choice of the next price in the ascending order, and fines f_i and f_j are due to simultaneous bidding/flipping, i.e. due to loss of time (but *time is money*).

[5] The exclusive disjunction, *eXclusive OR.*

3 Exchange by salesmen

In this section we design and prove a multiagent algorithm (protocol) for rational negotiations between buyers (who already have their individual salesmen) that eventually leads every buyer to knowledge that it is impossible to improve (reduce) by pairwise swapping salesmen the total price all buyers have to pay. Again the protocol comprises two algorithms: SWP (*SWaPping*) for an individual agents and a fair communication scheduler; description of a scheduler is out of scope of our paper, SWP-algorithm is similar to the above LSM-algorithm, it is discussed and presented in the sequel.

At every moment every buyer knows an individual salesman (current salesman) as its current intention (and it knows that nobody compete for this salesman); at the very beginning this current salesman is known *initial* salesman (for example due to the previous LSM-algorithm). Again, believes of every buyer are represented by two integer counters: NS for *Number of Swaps* and SF for *Swap-Free* buyers; these counters NS and SF have informal semantics similar to semantics of counters NC and CF in the algorithm LSM: NS represents agent's upper estimation of number of buyers with whom it can have swaps, and, respectively, SF represents its lower estimation of number of agents that believe they have no swaps at all; in particular, the agent believes that it will not swap the current salesman with any other buyer as soon as $NS = 0$; the agent believes that there are competition for any salesman at the marketplace as soon as $NS = 0$ and $SF = 2 \times (m - 1)$, it.e. it believes that it will no swaps, and it checks twice that all buyers believe that they will not swap their salesmen. But (in contrast to LSM-algorithm) the are two options how two agents can solve whether to swap their salesmen or not.

The first is similar to *Flip_Game* in the LSM-algorithm: two agents (say i and j) play the following static game with complete information that we denote as $Swap\_Game(g_i, f_i, g_j, f_j)$:

$i \setminus j$	*decline*	*swap*
decline	$(0, 0)$	(f_i, f_j)
swap	(f_i, f_j)	(g_i, g_j)

Two moves (strategies) in this game are *decline* and *swap* of the salesmen. Before the game every participating agent $b \in \{i, j\}$ sends to the partner two integer values: its gain[6] g_b, if agents swap their salesmen, and its "fine" f_b for disagreement on declining and swapping with the partner[7]. Then both agents compute mixed strategy Nash equilibrium (that is a rational behavior) and play corresponding mixed strategies until they make correlated beneficial moves. Study of this option is a topic for further research.

The second option is inspirited by SMEx-algorithm that solves MRP under assumption of fair communication [3]. This time two agents (say i and j) also play a static game with complete information that we denote by $Coop\_Game_\alpha(g_i, g_j)$:

$i \setminus j$	*decline*	*swap*
decline	$(0, 0)$	$(0, (1 - \alpha) * g)$
swap	$(\alpha * g, 0)$	$(\alpha * g, (1 - \alpha) * g)$

In this game $\alpha, (1 - \alpha) \in (0, 1)$ represent share (concession) of the total gain $g = g_i + g_j$, Two moves (strategies) in this game are *decline* and *swap* of the salesmen. Before the game every participating agent $b \in \{i, j\}$ sends to the partner its gain g_b in the case of swap[8]. This game has Nash equilibrium in pure strategies: (decline, decline) if $g < 0$, and (swap, swap) if $g > 0$. We call this game *Cooperative Game* since rational agents that swap their salesmen iff it is beneficial and then they share the total gain in some proportion (in contrast to *Swap_Game*, where everyone gets its own gain of loss even in the case when the total gain is positive).

Pseudocode of the SWP-algorithm is presented in the Appendix B. Some important properties of this SWP-algorithm are accumulated in the following two lemmas 4 and 5. Their proofs will be published in a forthcoming full research report.

Lemma 4. *Assume that it is common knowledge in a system that all agents execute one and the same algorithm SWP, and that every agent in the system has known its initial individual salesman before execution of the algorithm. Then every agent in this system at every moment of time after execution of line 2 knows some individual salesman that is someone in the set of all initial individual salesmen.*

[6] In contrast to l_b in *Flip_Game*, this value can be positive as well as negative.
[7] This value must be negative as in the *Flip_Game*.
[8] This value can be positive as well as negative as in the *Swap_Game*.

Lemma 5. *Assume that all agents execute one and the same algorithm SWP, that every agent had initial individual salesman before execution of the algorithm, and that communication is fair in the system. Then the system eventually terminates.*

Lemmas 4 and 5 together imply the following proposition.

Proposition 2.

If *a system with fair communication consists of $m > 0$ buyers each of which knows some initial individual salesman among $n \geq m$ salesmen, it is common knowledge (in the system) that all buyers are agents that execute algorithm SWP,*

then *every agent will eventually terminate, it will know upon termination an individual salesman (that is someone in the set of all initial individual salesmen), and it will know that it is impossible to reduce the total price all buyers have to pay by pairwise swapping salesmen.*

Proof of this proposition will be published in a forthcoming full research report.

In the Conclusion we demonstrate that the above algorithm SWP can not solve the graph-theoretic *assignment problem* that is (in our settings) to compute an assignment of buyers to robots with the *cheapest* total price [1]. But we do not know whether our two multiagent algorithms LSM and SWP coupled together can solve the problem or not. This topic also requires further research.

4 Conclusion

Let us remark, that both algorithms LSM and SWP presented in this paper (as well as all algorithms in the paper [3]) belong to a class of so-called *wave algorithms* [5], since they meet the following properties:

- termination: each computation is finite;
- decision: each computation contains at least one decide event;
- dependence: each decide event potentially influences all computations.

But what is more interesting for us in the case of LSM and SWP algorithms, is the following phenomenon: the common knowledge in a system of agents executing these algorithms emerges after double-check of individual beliefs. This observation leads us to the following topic for further research: When multiple (but finite and bounded) double-check of individual beliefs leads to common knowledge?

Couple of other topics for further research have been mentioned at the end of sections 2 and 3. Whether the outcome of the LSM-algorithm is a Pareto-optimal assignment? Whether algorithms LSM and SWP coupled together can solve the assignment problem?

Unfortunately, SWP-algorithms alone cannot solve the assignment problem as follows from a the counterexample below.

$b \setminus s$	s_1	s_2	s_3
b_1	0	1	3
b_2	3	0	1
b_3	1	3	0

This table presents weighted bipartite graph that consists of the vertices $\{b_1, b_2, b_3\}$ for buyers and $\{s_1, s_2, s_3\}$ for producers. Assume also that the initial assignment of producers to consumers is $b_1 \leftarrow s_2$, $b_2 \leftarrow s_3$ and $b_3 \leftarrow s_1$. In these settings, the exchange in line 12 of the algorithm SWP cannot work and hence the buyerswill terminate with the unchanged assignment, but this assignment is not optimal.

References

1. **Burkard R., Dell'Amico M., Martello S.** Assignment Problems. — SIAM, 2009.
2. **Fagin R., Halpern J.Y., Moses Y., Vardi M.Y.** Reasoning about Knowledge. — London: MIT Press, 1995.
3. **Shilov N., Garanina N. and Bodin E.** Multiagent approach to a Dijkstra problem. — In Proceedings of Workshop on Concurrency, Specification and Programming CS&P'2010 (Helenenau, September 27 – 29, 2010). — Humboldt-Universitat zu Berlin. Informatik-Bericht Nr.237, v.1, p.73–84. (Available at `http://www2.informatik.hu-berlin.de/ki/CSP2010/proceedings2010.zip`, visited May 10, 2011.)
4. **Shilov N.V., Shilova S.O.** Etude on theme of Dijkstra. ACM SIGACT News, 2004, v35(3), p.102–108.
5. **Tel** Introduction to Distributed Algorithms. — Cambridge University Press, 2nd Edition, 2000.
6. **Wooldridge M.** An Introduction to Multiagent Systems. — John Willey & Sons Ltd, 2002.

A Pseudocode of *Look for a Salesman* Algorithm

```
algorithm LSM::
const Me : integer in [1..m]                                   // Personal number.
const my_fine : integer;
                                  // A personal fine for simultaneous bidding/flipping.
var NC : integer in [1..m];
var CF : integer in [1..2*m];
var partner : integer in [1..m];
var cur_sman : integer in [1..n];                              // Current salesman.
var next_sman : integer in [1..n];                             // Next salesman.
var par_sman : integer in [1..n];                              // Partner's salesman.
var par_bel : boolean;
                          // Partner's belief that it (the partner) is conflict-free.
var contacts : set of [1..m];                                  // A set of buyers.
var my_loss : integer;                                // A loss due to flipping salesmen.
var par_loss : integer; //                          Partner's loss due to flipping salesmen.
var par_fine : integer;
                                  // Partner's fine for simultaneous bidding/flipping.
var pro_bid : real in [0..1];                         // Probability of bidding strategy.
begin
1:  NC:= (m - 1); CF:= 0;
2:  cur_sman := the salesman with the cheapest price in P_Me;
3:  repeat
4:   if NC > 0 then NC:= (m - 1);
5:   contacts:= set of all buyers but Me;
6:   repeat
7:    partner := a buyer in the contacts ready to communicate;
                                                       // Scheduler resolves this request.
8:    start communication session with the partner:
9:    fork
       send <cur_sman>, <NC=0> to partner ||
       receive <par_sman>, <par_bel> from partner
      join;
10:   if cur_sman = par_sman
11:   then
      begin
12:    next_sman:= a salesman with the next to cur_sman's price
                                    according to P_Me, if it exists, cur_sman otherwise;
13:    my_loss:= P_{cur_sman,Me} - P_{next_sman,Me};
14:    fork
        send <my_loss>, <my_fine> to partner ||
        receive <par_loss>, <par_fine> from partner
       join;
15:    pro_bid := probability of My bid in Nash equilibria
                                       in Flip_Game(my_loss, my_fine, par_loss, par_fine);
                                            // Solves the Flip_Game in mixed strategies.
16:    repeat                                                  // Play Flip_Game;
17:     my_move:= some_in {cur_sman:pro_bid,
                                                          next_sman: (1 - pro_bid)};
                                // Select My next move according to assigned probabilities.
18:     fork
         send <my_move> to partner ||
         receive <par_move> from partner
        join;
19:    until (my_move=cur_sman ∧ par_sman≠par_move) ∨
                                                  (my_move≠cur_sman ∧ par_sman=par_move);
```

```
20:    cur_sman:= my_move; NC:= (m - 1); CF:= 0
21:   end
22:   else if NC > 0
23:       then {NC:= (NC - 1); CF:= 0}
24:       else if par_bel
25:           then CF:= CF + 1
26:           else {NC:= (m - 1); CF:= 0}
27:   close communication session with partner;
28:  contacts:= remove partner from contacts
29:  until contacts becomes empty
30: until (NC = 0 ∧ CF = 2*(m - 1) )
end.
```

B Pseudocode of *Swapping salesmen* Algorithm

```
algorithm SWP::
const Me : integer in [1..m]                                          // Personal number.
const Intit : integer in [1..n]                                       // Initial salesman.
var NS : integer in [1..m];
var SF : integer in [1..2*m];
var partner : integer in [1..m];
var cur_sman : integer in [1..n];                                     // Current salesman.
var par_sman : integer in [1..n];                                     // Partner's salesman.
var par_bel : boolean;
                              // Partner's belief that it (the partner) has nothing to swap.
var contacts : set of [1..m];                                         // A set of buyers.
var my_gain : integer;                                  // A gain due to swapping salesmen.
var par_gain : integer; //                             Partner's gain due to swapping salesmen.
begin
1:  NS:= (m - 1); SF:= 0;
2:  cur_sman := Init;
3:  repeat
4:   if NC > 0 then NC:= (m - 1);
5:   contacts:= set of all buyers but Me;
6:   repeat
7:    partner := a buyer in the contacts ready to communicate;
                                                    // Scheduler resolves this request.
8:    start communication session with the partner:
9:    fork
       send <cur_sman>, <NS=0> to partner ||
       receive <par_sman>, <par_bel> from partner
      join;
10:   my_gain:= P_{cur_sman,Me} − P_{par_sman,Me};                  // Compute personal gain.
11:   fork
       send <my_gain> to partner ||
       receive <par_gain> from partner
      join;
12:   if my_gain + par_gain > 0                                     // Solve Coop_Game.
       then {cur_sman:= par_sman; NS:= (m - 1); SF:= 0}
13:    else if NS > 0
14:        then {NS:= (NS - 1); SF:= 0}
15:        else if par_bel
16:            then SF:= SF + 1
17:            else {NS:= (m - 1); SF:= 0}
18:   close communication session with partner;
19:  contacts:= remove partner from contacts
```

```
20:  until contacts becomes empty
21: until (NC = 0 ∧ CF = 2*(m - 1) )
end.
```

Средства визуализации онтологий и информационного наполнения научных порталов и методы построения жгутов ребер

З.В. Апанович[1], Т.А. Кислицына[2]

[1]Институт систем информатики СО РАН
630090, Новосибирск, проспект Лаврентьева, 6, Россия
apanovich@iis.nsk.su
тел: +7 (383) 330-33-44, факс: +7 (383) 332-34-94
[2]Новосибирский государственный университет,
630090, Новосибирск, ул. Пирогова, 2, Россия

Ключевые слова: портал знаний, онтология, информационное наполнение, анализ информации, методы визуализации информации, силовой алгоритм, радиальный алгоритм, иерархические и геометрические жгуты ребер, поуровневое изображение ориентированного графа, сети цитирования, Open Linked Data

Введение

В связи с бурно развивающимся направлением Semantic Web и его новой ветвью Linked Open Data[1] в Интернете становятся доступными большие объемы информации, посвященной различным научным направлениям. В число таких ресурсов входят информационные системы, цифровые библиотеки и специализированные порталы, посвященные научным направлениям и основанные на онтологиях. Важные научные направления, самые активные и влиятельные исследователи, организации, в которых они работают, и места, в которых расположены научные организации – вся эта информация становится доступной в rdf/xml формате. Наиболее достоверным источником информации, посвященной любому научному направлению, являются собственно научные публикации, составляющие основное наполнение научных порталов и цифровых библиотек. Важно также отметить, что эта информация эволюционирует во времени и стремительно увеличивается в объеме. Исследование и анализ этих данных необходимы как в процессе разработки Интернет ресурсов, так и на протяжении всего их жизненного цикла. Если при разработке Интернет ресурсов наиболее важной задачей является поиск и устранение ошибок, то в последующем более важным становится наукометрический анализ накапливаемых данных, позволяющий оптимизировать процессы управления научными исследованиями. Для обеспечения понимания этих стремительно увеличивающихся в объеме данных нужны новые инструменты.

Одним из таких общепризнанных инструментов является визуализация информации с применением графовых моделей. Существенной проблемой при визуализации онтологий, а в особенности, информационного наполнения порталов, является большой объем данных, ведущий к визуальной перегруженности изображения. Одним из возможных способов решения этой проблемы является объединение ребер в «жгуты». В данной работе рассматриваются различные стратегии применения визуализации на основе жгутов ребер для онтологий и информационного наполнения порталов, основанных на онтологиях.

1. Визуализация онтологий и информационного наполнения при помощи метода иерархических жгутов ребер.

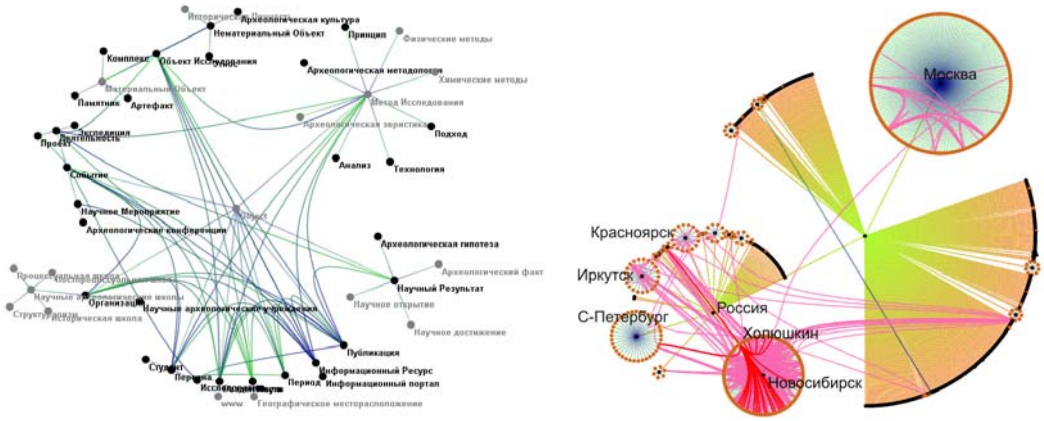
Онтология представляет собой граф, вершинами которого являются классы, а ребра соответствуют отношению наследования между классами и ассоциативным отношениям. В работе [2] уже была представлена техника совместного изображения таксономии и ассоциативных связей, но прежние возможности этой техники были весьма ограничены, так как быстро возникала перегруженность изображения. Поэтому нас заинтересовала идея использования для этих целей метода иерархических жгутов ребер. Этот метод позволяет извлекать из онтологии отношение наследования и накладывать на его изображение изображения ассоциативных отношений. Для отображения иерархической структуры, индуцируемой отношением наследования, можно использовать любой алгоритм визуализации деревьев, а каждое ребро, соответствующее ассоциативному отношению, – изображать в виде кубического β-сплайна (кривой).

Мы строим изображение дерева при помощи либо радиального, либо кругового алгоритма. Затем координаты вершин построенного дерева используются в качестве контрольных точек, через которые проводится каждое ребро, соответствующее ассоциативному отношению. Таким образом, каждое ребро ассоциативного отношения, представляет собой многоугольный путь по изображению дерева, описываемый последовательностью точек P_0, P_1,…, P_{N-1}. Зная эти контрольные точки, можно построить кривую, изображающую ребро ассоциативного отношения по формуле:

$$P' = \beta \cdot P_i + (1-\beta)(P_0 + \frac{i}{N-i}(P_{N-1} - P_0)\,,\ \text{где}$$

N – это количество контрольных точек,
i – номер контрольной точки, $i \in \{0, \ldots, N-1\}$,
P_i – позиция i-ой контрольной точки
β –сила связывания, $\beta \in [0, 1]$.

На рисунке 1(а) показана онтология археологического портала [3]. Отношение наследования показано при помощи кругового алгоритма. Ассоциативные отношения показаны при помощи иерархических жгутов ребер[9]. Достоинством этого типа изображения является также то, что оно позволяет увидеть ВСЕ ассоциативные связи данной онтологии одновременно. Кроме этого, с первого взгляда видны также подклассы, не имеющие собственных отношений. Исходя из обнаруженных свойств, можно рекомендовать инженеру знаний либо дополнить такие подклассы отношениями, отличающими их от родительских классов, либо удалить их из множества классов.



(а) (б)

Рис. 1. (а)- Визуализация всех существующих ассоциативных отношений археологического портала в комбинации с отношением наследования. (б)- Отношение соавторства между научными сотрудниками, работающими в раз личных географических областях

В процессе экспериментов с наполнением информационных порталов мы заметили также, что они содержат большое количество отношений партономии, задающих иерархию объектов. К таким отношениям относятся Метод-Исследования-Включает, Организация-Включает Место-Включает, и др. Очевидно, что метод иерархических жгутов ребер позволяет строить комбинированные изображения отношений партономии и любых других отношений, имеющихся в онтологии. Возможности этого метода можно расширить, если визуализировать не только отношения, содержащиеся в онтологии в явном виде, но и вновь определяемые отношения. В частности, на основе информации о научных публикациях, составляющих наибольшую часть любого научного портала, можно сгенерировать, а затем визуализировать сети соавторства, сети цитирования и ко-цитирования. Для порталов, описанных в RDF/OWL формате, генерация таких сетей осуществляется при помощи запросов sparql. Для других форматов генерация новых отношений осуществляется программно. Важность визуализации таких сетей для анализа тенденция развития научных направления обсуждалась в работах [12,13].

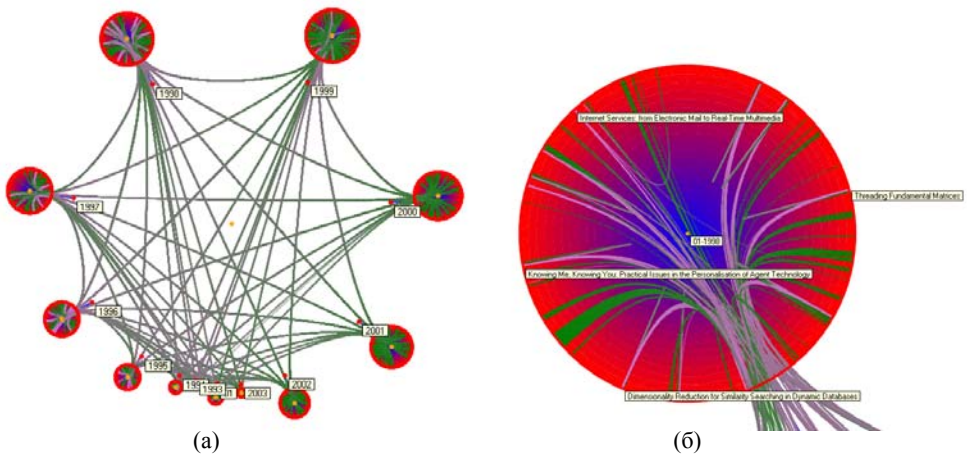
На рисунке 1(б) изображено отношение соавторства между научными сотрудниками, работающими в различных географических областях, полученное при помощи метода иерархических жгутов ребер. Это отношение было сгенерировано программно на основе информации об авторстве публикаций археологического портала. В качестве основы изображения используется круговое изображение дерева, соответствующего отношению *Место-включает*. Маленькие черные окружности изображают географические объекты, такие как страна, город, поселок, и т.д., прямолинейные ребра соответствуют отношению *Место-включает*. То есть, наличие ребра между объектом Россия и объектом Иркутск соответствует тому, что Иркутск находится в России. Маленькие светлые окружности изображают отдельных научных сотрудников, а прямолинейные ребра, соединяющие вершины-сотрудники с вершинами-местами соответствуют факту проживания определенного научного сотрудника в той или иной местности. Светлые криволинейные шлейфы изображают отношения соавторства между исследователями из разных городов, а более тонкие и более темные криволинейные шлейфы показывают отношения сотрудничества одного выбранного исследователя (Холюшкин). Видно, что у него есть соавторы в Новосибирске, Санкт-Петербурге, Красноярске и др., но нет соавторов в Москве. Это изображение является также примером того, как выглядят недоработки при введении данных. Например, все российские города располагаются на окружности, центром которой является вершина с названием Россия. Но город Москва, расположен по

периметру другой окружности. Значит, в тестовых данных не хватает информации о том, что Москва находится в России. Таким же способом можно исследовать зависимость отношения соавторства между исследователями, работающими в разных разделах науки, в различных научных организациях, применяющих разные методы исследования, и т.д.

2. Визуализация сетей цитирования, извлекаемых из облака Linked Open Data

Следующая группа экспериментов связана с генерацией и визуализацией сетей цитирования. В качестве исходных данных для этих визуализаций использовались порталы, входящие в облако Linked Open Data (LOD)[6-9], в которое входят такие известные порталы как DBLP, Citeseer, CORDIS, NSF, EPSRC, ACM, IEEE и др. Данные предоставляются в формате RDF и имеют весьма внушительные объемы. Например, RDF-данные, предоставленные порталом Citeseer, содержат 8 146 852 троек RDF, данные портала ACM насчитывают 12 402 336 троек RDF, портал DBLP предоставил 28 384 790 троек RDF. Пользователь может либо скачивать файлы в формате RDF, либо генерировать данные при помощи запросов sparql. Все данные этих порталов переведены на единую онтологию AKT Reference Ontology [6], представляющую собой объединение нескольких онтологий. Надо сказать, что хотя отношение цитирования (akt:cites-publication-reference) имеется в AKT Reference Ontology, то на уровне информационных объектов такие отношения описаны только для порталов Siteceer и ACM[6, 8]. Также следует отметить, что нами были затрачены весьма значительные дополнительные усилия для генерации информативных сетей цитирования. В случае портала Siteseer нами применялась двухуровневая схема генерации сетей цитирования, а в случае портала ACM дополнительно использовалась собственная онтология этого портала, позволяющая выбирать публикации, относящиеся к определенному разделу науки.

Наконец, следует сказать, что методы, применяемые при визуализации сетей соавторства, оказались мало пригодными для визуализации сетей цитирования. Прежде всего, сеть цитирования является ориентированным графом, поэтому для понятного изображения этой сети желательно, чтобы все ребра были направлены в одну сторону. Направление ребер может соответствовать хронологическому порядку публикаций. В принципе, метод изображения иерархических жгутов ребер, соответствует этому требованию. Но применение стандартного метода иерархических жгутов ребер затруднено тем фактом, что в такой базе данных как Citeseer нет достаточно глубокой иерархии, на которую можно было бы наложить сети цитирования. На основе информации о публикациях, мы в своих экспериментах строили иерархию дат публикаций, которая имела всего 2 уровня: год публикации-месяц публикации. В результате получалось изображение, достаточно разреженное в центре и сильно перегруженное на периферии, как это можно видеть на Рисунке 2. На этом рисунке показано изображение сети цитирования из 20 000 вершин, извлеченной из базы данных портала Citeseer. Временной период этих публикаций с 1993 по 2003 год. На рисунке 2(а) показан общий план изображения этого множества данных. Поскольку ребра в этой сети ориентированные, для облегчения задачи определения направления ребер их концы раскрашены в разные цвета. Входной конец ребра (инцидентный цитируемой вершине-публикации) раскрашен сиреневым цветом, а выходной конец ребра (инцидентный цитирующей публикации) раскрашен зеленым цветом. Можно заметить, что наибольшее количество публикаций в этом множестве приходится на 1998 и 1989 годы. При этом можно рассмотреть достаточно много ссылок на публикации этих лет (жгуты сиреневого цвета), а также заметить, что публикации 2003 года весьма немногочисленны, и от них идут жгуты зеленого цвета – ссылки на более ранние публикации. Для более детального изучения надо рассматривать это изображение фрагментами. При возрастании размеров сети цитирования, в особенности, при увеличении временного интервала, которому принадлежат публикации сети цитирования, эта задача становится весьма трудной для данного алгоритма изображения (Рисунок 2(б).)

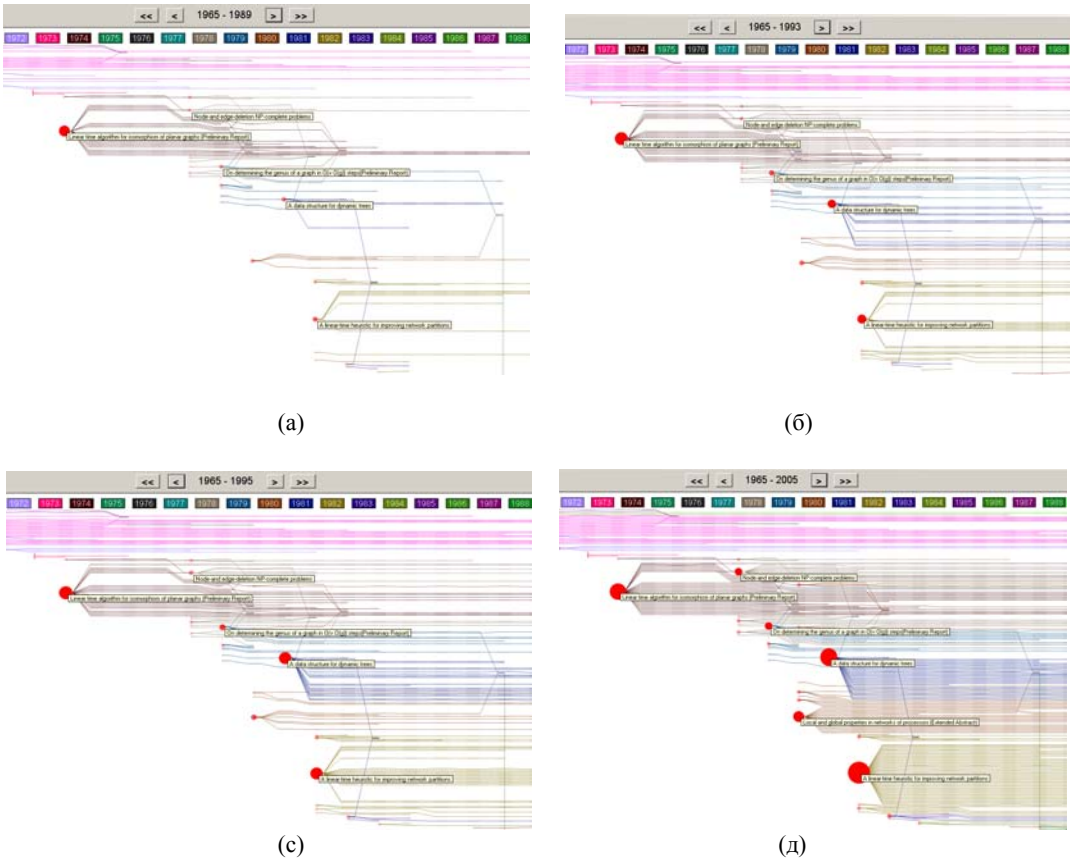


(а) (б)

Рис. 2. Изображение сети цитирования, извлеченной из RDF-данных портала Siteseer и содержащей 20 000 вершин. (а) общий план изображения, (б) публикации за один месяц 1998 года

Для того чтобы сделать возможным просмотр и анализ изменения сетей цитирования на больших промежутках времени, нами был реализован метод поуровневого размещения ориентированного графа с минимизацией пересечений ребер [11]. Суть данного метода состоит в том, что вершины-публикации разбиваются на слои, соответствующие различным годам. Индекс цитирования публикации, т.е. ее значимость, отображается площадью вершины и интенсивностью ее цвета, что позволяет сразу увидеть самые важные публикации за определенный интервал времени.

На рисунке 3 показано изображение сети цитирования, полученное при помощи поуровневого метода размещения. Вершины этой сети, соответствующие отдельным публикациям, упорядочены хронологически по годам публикаций. Годы публикаций показаны прямоугольниками разного цвета в верхней части изображения. Все публикации, появившиеся в одном году, располагаются в вертикальном столбце, соответствующем этому году. Ребра этой сети соответствуют отношению цитирования. Каждое ребро сети цитирования соответствует отношению akt:cites-publication-reference и ориентировано справа налево. Чем больше ссылок в сети цитирования имеется на некоторую публикацию, тем больше входных ребер имеет соответствующая вершина, и тем больше ее радиус. Цвет каждого ребра, соответствует цвету года цитирующей публикации. Для того чтобы легче было отследить количество ссылок на одну и ту же публикацию, используется процедура минимизации количества пересечений ребер. В каждом вертикальном ряду осуществляется сортировка вершин, переставляющая каждую вершину в центр тяжести вершин, расположенных в ближайшем к ней ряду слева, с которыми она связана ребром цитирования. Для того чтобы такие перестановки были возможны, каждое длинное ребро цитирования разбивается фиктивными вершинами на короткие ребра. Длинными являются ребра-ссылки на публикации, с момента появления которых до рассматриваемого момента прошло несколько лет. Каждое короткое ребро соединяет вершины, расположенные в соседних вертикальных рядах. Благодаря этой трансформации, ребра цитирования одной и той же публикации образуют хорошо различимые на рисунке жгуты. Также, в программе реализована возможность отслеживания динамики цитирования по годам. Для этого в верхней части экрана расположены кнопки, позволяющие перемещаться по изображению с заданными интервалами времени. В данный момент размер минимального интервала равен одному году. При нажатии кнопки « >> » изображается вся имеющаяся сеть цитирования, а при нажатии « << » происходит очистка изображения.



(а) (б)

(с) (д)

Рис. 3. Изменение значимости публикаций во времени.

Перемещение по изображению осуществляется при помощи кнопок « <» и « > », позволяя наблюдать изменение сети цитирования во времени. Технически, эта возможность реализована при помощи фильтрации

вершин и ребер сети цитирования.

На рисунке 3 показана изменяющаяся во времени сеть цитирования для публикаций по теории графов. Четыре рисунка покрывают фрагменты временного интервала с 1965 по 2005 год. В период с 1965 по 1989 (Рис.3(а)) среди публикаций по теории графов доминирует «Linear-time algorithm for isomorphism of planar graphs». Эта вершина имеет самый большой радиус и большой коричневый шлейф. К 1993 году (Рис.3(б)) возрастает цитируемость публикации «A data structure for dynamic trees» и «A linear-time heuristic for improving network partition». К 1995 году (Рис.3(с)), они имеют такую же значимость, как и публикация «Linear-time algorithm for isomorphism of planar graphs». А в 2005 году (Рис.3(д)) публикация «A linear-time heuristic for improving network partition» является самой цитируемой. Можно так же видеть, как появляется интерес к публикации «Node-and-edge-deletion NP-complete problems», причем она ссылается на ранее доминировавшую публикацию «Linear-time algorithm for isomorphism of planar graphs», т.е. образуется цепочка значимых связанных публикаций. Помимо всего прочего, такой способ визуализации, позволяет обнаруживать ошибки и неточности в библиографических данных.

3. Геометрический метод построения жгутов ребер

Проблемой с применением обычного поуровневого метода является то, что очень быстро возникает перегруженность изображения, а применение фильтрации удаляющей малозначимые публикации, искажает реальность: малозначимые публикации вносят основной вклад при определении значимости других публикаций. Поэтому возникла необходимость в алгоритме визуализации, который уменьшал бы визуальную загруженность изображения, формируя жгуты ребер на основе их собственной геометрии [14], а не привнесенной извне иерархии. Общая схема алгоритма выглядит следующим образом:

8. Сгенерировать прямоугольную сетку размера NxN и отобразить на эту сетку изображение графа, построенное любым способом;
9. Для каждой ячейки прямоугольной сетки вычислить основное направление ребер, пересекающих эту ячейку;
10. Объединить соседние ячейки с направлениями, отличающимися не более чем на пороговое значение *a*, в зоны;
11. Вычислить основное направление в каждой зоне, и перпендикуляр к основному направлению зоны;
12. Построить отрезки, проходящие перпендикулярно направлению зоны до пересечения с границей зоны;
13. Использовать полученные точки пересечения с границей каждой зоны для построения новой сетки при помощи триангуляции;
14. Для каждого ребра построенной триангуляции найти точки пересечений с ребрами исходного изображения графа. Вычислить центр среди этих точек;
15. Для каждого ребра графа G построить b-spline, проходящий через центральные точки ребер контрольной сетки, которые пересекает ребро графа G.

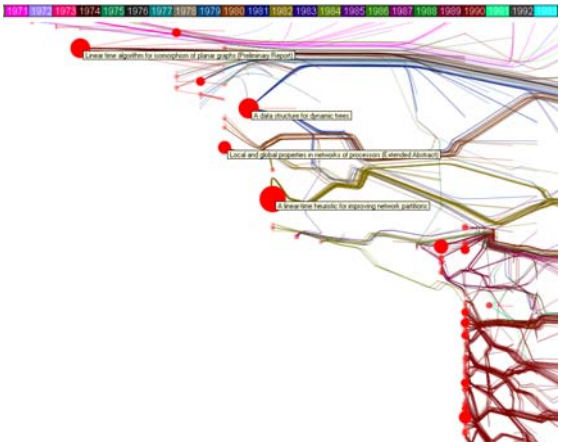


Рис. 4. Применение идеологии жгутов ребер к сети цитирования

На Рис. 4. показано применение алгоритма геометрических жгутов ребер к изображению, полученному при помощи поуровневого изображения сети цитирования, показанной на Рис. 3(д).

Следует заметить, что этот алгоритм находится в настоящий момент в экспериментальной эксплуатации, и у нас имеется гораздо больше вопросов, связанных с этим методом, чем ответов на них. Как наилучшим обра-

зом выбрать направление прямоугольной сетки? Как зависит направление жгутов ребер от размера сетки? Как выбрать наилучшее направление внутри каждой зоны? Тем не менее, даже в настоящий момент можно констатировать, что этот алгоритм существенно уменьшает загруженность изображения и, мы надеемся его развить до состояния, когда с его помощью можно будет диагностировать тенденции развития научного направления.

Заключение

В данной работе рассмотрены методы построения изображений, направленные на уменьшение перегруженности изображений больших графов. Эти методы опробованы как на небольших порталах научного направления, так и на порталах, входящих в облако Linked Open Data.

Генерируемые при помощи наших методов изображения позволяют обнаруживать и корректировать ошибки, возникающие на этапе разработки порталов, основанных на онтологии, а также наглядно представляют наукометрическую информацию в течение жизненного цикла таких порталов.

Список литературы

1. Bizer, C., Heath, T. and Berners-Lee, T. Linked Data - The Story So Far. //Int. J. Semantic Web Inf. Syst., 5 (3) .— 2009.— pp. 1-22.
2. Апанович З.В. Винокуров П.С. Анализ онтологии и информационного наполнения портала знаний при помощи методов визуализации информации // Проблемы управления и моделирования в сложных системах: Труды XI Международной конференции (Самара, 22-24 июня 2009 г.) .— 2009.— С. 556-562.
3. Загорулько Ю.А., Боровикова О.И., Холюшкин Ю.П. Построение предметной онтологии для археологического портала научных знаний//Информационные технологии в гуманитарных исследованиях. 2006. N10.
4. Апанович З.В., Кислицына Т.А. Расширение подсистемы визуализации наполнения информационного портала средствами визуальной аналитики // Проблемы управления и моделирования в сложных системах: Труды XII Международной конференции (Самара, 21-23 июня 2010 г.) .— 2010.— С. 518-525.
5. Apanovich Z. V., Vinokurov P. S. Ontology based portals and visual analysis of scientific communities//First Russia and Pacific Conference on Computer Technology and Applications, 6-9 September, 2010.— Vladivostok, Russia.— 2010.— pp. 7-11.
6. http://www.w3.org/wiki/TaskForces/CommunityProjects/LinkingOpenData/DataSets.
7. Описание онтологии AKT: http://www.aktors.org/ontology.
8. Данные портала CiteSeer: http://citeseer.rkbexplorer.com/.
9. Данные портала ACM: http://acm.rkbexplorer.com/.
10. Holten D. , Hierarchical Edge Bundles: Visualization of Adjacency Relations in Hierarchical Data// IEEE Transactions on Visualization and Computer Graphics.— v.12, n.5.— 2006. — pp. 741-748.
11. Sugiyama K., Tagawa S., Toda M. Methods for Visual Understanding of Hierarchical System Structures, //IEEE Trans. Systems, Man, and Cybernetics.— 1981.— pp. 109-125.
12. Chen Ch., Song I-Y., Weizhong Zhu W. Trends in conceptual modeling: Citation analysis of the ER conference papers (1979-2005). //Proceedings of the 11th International Conference on the International Society for Scientometrics and Informatrics. CSIC, Madrid, Spain. June 25-27, 2007. pp. 189-200.
13. Henry Small. Visualizing Science by Citation Mapping //Journal of the American Society for Information Science.— Vol. 50(9) .— 1999 .— pp. 799-813.
14. Cui W., Zhou H., Wong P., Li X., Geometry-based edge clustering for graph visualization. IEEE Transactions on Visualization and Computer Graphics 14, 6 (2008), 1277–1284.

Информационный и криптографический аспекты задачи о роботах на Марсе

А.Ю. Бернштейн

Ранее Н.В. Шилов, Н.О. Гаранина и В.Е. Бодин описали и исследовали следующую задачу «о роботах на Марсе» (MRP). Она состоит в том, что $n>1$ автономным роботам требуется без постороннего вмешательства договориться о таком распределении укрытий, при котором прямолинейные пути роботов до соответствующих им укрытий не будут иметь попарных пересечений. Для достижения этой цели роботы могут попарно обмениваться сообщениями. Однако были исследованы вопросы о существовании протоколов, служащих для решения этой задачи, при следующих допущениях: роботы в состоянии передать друг другу информацию, касающуюся их геометрического положения (1), не разглашая своих точных координат (2). Настоящая работа посвящена исследованию этих двух аспектов этой задачи: информационному и криптографическому. В рамках исследования информационного аспекта было доказано, что не существует протокола, решающего MRP, который бы требовал передачи ограниченного количества бит. В рамках исследования криптографического аспекта был предложен протокол, позволяющих двум роботам проверить, пересекаются ли их пути, не разглашая дополнительных сведений о своём положении.

1. Введение

1.1. Постановка задачи и краткий обзор результатов

Начнём с неформальной постановки задач, решению которых посвящена данная работа. Мы будем рассматривать мультиагентные алгоритмы (протоколы), решающие следующую *задачу о роботах на Марсе*.

«На плоском участке поверхности Марса находятся $n > 1$ автономных роботов и столько же укрытий в общем положении. Местоположение всех укрытий фиксировано и известно каждому роботу. Каждый робот знает о существовании всех остальных роботов, но не знает их координаты. Роботы могут взаимодействовать каждый с каждым попарно. В некоторый момент времени все роботы останавливаются, фиксируют свои текущие позиции, и затем они все должны выбрать укрытия, чтобы двинуться к ним по прямому маршруту. Ясно, что роботам нельзя сталкиваться; поэтому каждый отдельный робот может двигаться к укрытию только когда он знает (абсолютно уверен), что на своём пути он не столкнётся с другим роботом. Задача: разработать мультиагентный алгоритм, гарантирующий, что каждый робот когда-нибудь будет точно знать, что его маршрут к выбранному укрытию не пересекается с маршрутами остальных роботов.» (Цит. по [1].)

Задача носит неформальный характер, однако её можно разумным образом формализовать, как это сделано, например, в уже цитировавшейся работе [1], и построить протоколы, решающие эту задачу. Возникает вопрос: как выбрать из них какой-то один, в некотором смысле лучший? Ясно, что такой выбор будет зависеть от имеющихся целей и приоритетов, то есть от того, что именно делает один протокол предпочтительнее другого. Обычно различные протоколы сравниваются по временной сложности (количеству итераций, требующихся для выполнения протокола). Однако этот вопрос требует дополнительных исследований. В данной работе исследуется сложность передачи отдельных сообщений, которые необходимы для работы протокола. В этом состоит информационная постановка задачи.

Криптографическая постановка задачи состоит в следующем. В исходном протоколе агенты сообщали друг другу достаточно много сведений о себе, которые могли бы позволить каждому отдельному роботу решить задачу сразу за всех роботов. В данной работе решается задача сокрытия сведений о положении роботов.

Теперь кратко сформулируем основные результаты, полученные нами в решении этих двух задач.

В решении информационной постановки задачи получено несколько результатов, наиболее типичный из которых состоит в том, что не существует протокола, решающего задачу с передачей ограниченного числа бит даже в случае целочисленных координат на плоскости. Иными словами, для любого протокола при любом $n > 1$ можно так расположить роботов, чтобы им пришлось переслать друг другу в соответствии с этим протоколом больше N бит, где N — произвольное наперёд заданное число.

В рамках криптографической постановки задачи условие того, что пути роботов не пересекаются, удалось сформулировать в таком виде, что решение задачи свелось к вычислению логической схемы, размер которой линейно зависит от количества бит в координатах роботов. При этом тайное вычисление этой схемы с применением выбранного нами протокола оказывается очень удобным (так как схема при этом несколько упрощается). Таким образом, как и требовалось, мы получили эффективный протокол проверки пересечения путей двух роботов, при котором они не получают никаких дополнительных сведений о координатах друг друга.

1.2. Основные протоколы

В этом пункте мы фиксируем некоторые обозначения, которыми будем пользоваться на протяжении всей работы.

Протоколом *распределения укрытий между* n *участниками* (или просто протоколом распределения) будем называть протокол, удовлетворяющий следующим условиям.

1. В протоколе участвуют n сторон R_1, …, R_n, называемых *роботами*.
2. Каждый робот R_i знает два числа x_i и y_i, неизвестные остальным роботам. Эти числа называются его *координатами*.
3. Имеются также $2n$ констант, известных всем роботам: x_1^0, y_1^0, …, x_n^0, y_n^0. Числа x_i^0 и y_i^0 называются *координатами i-ого укрытия*.
4. Кроме того, изначально инициализированы ещё n глобальных переменных, значения которых известны всем роботам: u_1, …, u_n. Значения этих переменных принадлежат множеству $\{1,2,\ldots,n\}$, причём изначально они попарно различны. Смысл переменной u_i состоит в том, что в текущий момент роботу R_i соответствует укрытие с номером u_i. Значения переменных u_i могут быть особым образом изменены (см. далее).
5. Между каждыми двумя роботами существует канал связи, по которому они могут обмениваться сообщениями.
6. В случае, если того требует протокол, любые два робота R_i и R_j могут *обменяться укрытиями*, то есть обменять значениями переменные u_i и u_j.
7. Гарантируется, что среди $2n$ точек на плоскости с координатами (x_1,y_1), …, (x_n,y_n), (x_1^0,y_1^0), …, (x_n^0,y_n^0) никакие три не лежат на одной прямой.
8. Обозначим через P_i отрезок на плоскости, координатами концов которого являются точки (x_i,y_i) и $(x_{u_i}^0,y_{u_i}^0)$, и назовём его *путём* робота R_i. Тогда по завершении протокола (а он должен завершиться за конечное время при любых допустимых значениях входных данных) любые два отрезка P_i и P_j, где $i \neq j$, должны не пересекаться.

По сути, протокол распределения укрытий — это протокол, решающий MRP. Именно различные протоколы распределения укрытий будут представлять для нас основной интерес.

Назовём протокол, участниками которого являются два робота R_i и R_j, позволяющий этим роботам проверить, нужно ли им обмениваться укрытиями, *модифицированным протоколом щелчка с обменом* (или модифицированным протоколом щелчка, или просто модифицированным щелчком), если он удовлетворяет следующим двум условиям:

1. Если текущие пути роботов пересекаются, то они должны совершить обмен укрытиями.
2. После обмена укрытиями суммарная длина путей роботов всегда строго уменьшается.

Второе условие гарантирует, что с помощью модифицированного протокола щелчка можно совершить лишь конечное число обменов укрытиями. В свою очередь, первое условие означает, что если с его помощью нельзя совершить ни одного обмена укрытиями, то текущие пути роботов попарно не пересекаются. Эти два свойства позволяют построить на основе любого модифицированного протокола щелчка протокол распределения укрытий, как показано в работе [1]. В рамках данного исследования для нас не важно, как именно строится этот протокол. Единственное существенное для нас обстоятельство заключается в том, что при исполнении этого протокола роботы сообщают друг другу что-либо о своём положении только во время выполнения модифицированных щелчков.

Для удобства для модифицированных протоколов щелчка мы вводим следующие обозначения. Роботы-участники протокола обозначаются A и B, так же обозначаются точки на плоскости, соответствующие координатам этих роботов. Точки на плоскости, соответствующие координатам их укрытий, обозначаются A^0 и B^0 соответственно. Координаты роботов обозначаются (x_A,y_A), (x_B,y_B), координаты укрытий — (x_A^0,y_A^0) и (x_B^0,y_B^0).

Приведём два основных примера модифицированных протоколов щелчка. При исполнении первого из них, *протокола щелчка*, роботы обмениваются укрытиями в том и только в том случае, когда их текущие пути пересекаются. При исполнении второго, *протокола сравнения расстояний*, роботы обмениваются укрытиями в том и только в том случае, когда после обмена суммарная длина их путей строго уменьшится. Почти всюду далее нас будет интересовать только протокол щелчка.

2. Информационная постановка задачи

В рамках информационной постановки задачи там, где не оговорено обратное, в качестве входных данных для протоколов будут рассматриваться действительные числа. При этом нас не будет интересовать то, как роботы хранят их в памяти. Кроме того, мы будем считать, что роботы могут выполнять с ними с абсолютной точностью произвольные операции: складывать, вычитать, умножать, логарифмировать и так далее. Наши дальнейшие рассуждения останутся в силе, даже если считать, что робот может вычислять произвольные функции от действительных чисел. Таким образом, единственной проблемной частью протоколов становится передача роботами результатов своих вычислений друг другу. Наконец, все протоколы в этой части работы мы будем считать детерминированными, то есть такими, в которых действия участников однозначно определяются предыдущим ходом исполнения протокола и входными данными. (Любой протокол можно превратить в детерминированный, однозначно зафиксировав исход каждого случайного выбора, осуществляемого его участниками. При этом количество бит, передаваемое при его исполнении в худшем случае не увеличится, а только оно для нас существенно.)

2.1. Характеристики протоколов

Основная величина, которая будет нас интересовать, — это общее количество бит, переданное участниками друг другу в ходе исполнения протокола. Её мы будем называть *временем работы протокола*[1]. В зависимости от того, как можно оценить время работы данного протокола, мы будем давать ему различные характеристики.

Будем называть протокол *финитным*, если при любых допустимых значениях входных данных его работа завершается после передачи конечного количества бит.

Будем называть протокол *ограниченным*, если существует такая константа N, что при любых допустимых значениях входных данных его работа завершается после передачи не более N бит. Ограниченные протоколы — самые «хорошие» в смысле передачи информации. Однако мы покажем, что большинство протоколов, которые мы будем исследовать, не являются ограниченными.

Будем называть протокол *частично ограниченным*, если время его работы можно оценить сверху некоторой функцией от входных данных всех участников, кроме какого-то одного. Поскольку в протоколе распределения укрытий роли всех участников одинаковы, для него без ограничения общности можно считать, что «свободными» остаются только входные данные первого участника. Тогда частичная ограниченность протокола распределения укрытий означает, что существует функция $f\colon \mathbb{R}^{4n-2} \to \mathbb{R}$, такая, что при любых допустимых значениях входных данных время работы протокола не превосходит $f(x_2, y_2, \ldots, x_n, y_n, x_1^0, y_1^0, \ldots, x_n^0, y_n^0)$. Очевидно, что любой ограниченный протокол является также и частично ограниченным (в этом случае функцию f можно положить тождественно равной ограничивающей константе). Поэтому ограниченность — более сильное свойство, чем частичная ограниченность.

Ещё один способ ослабить свойство ограниченности мы получим, если будем вместо количества переданных бит рассматривать только количество переданных сообщений. Именно, будем называть протокол *ограниченным по сообщениям*, если существует такая константа N, что при любых допустимых значениях входных данных его работа завершается после передачи не более N сообщений. Поскольку каждое сообщение содержит по меньшей мере один бит, любой ограниченный протокол является также и ограниченным по сообщениям.

Наконец, по аналогии с частично ограниченными протоколами будем называть протокол *частично ограниченным по сообщениям*, если количество сообщений, передаваемое участниками за время его работы, можно оценить сверху некоторой функцией от входных данных всех участников, кроме какого-то одного. Свойство частичной ограниченности по сообщениям — самое слабое из тех, которые мы будем исследовать.

2.2. Основные утверждения

В этом пункте мы сформулируем основные результаты, касающиеся протокола распределения с n участниками.

Нам потребуется ещё одно определение. *Сужением* протокола π на множество I будем называть протокол, отличающийся от π только тем, что входные данные для него должны принадлежать множеству I. В частности, сужение протокола распределения на множество $M \subset \mathbb{R}^2$ — это протокол распределения, в котором все точки на плоскости, соответствующие координатам роботов и укрытий, должны принадлежать множеству M.

Теорема 1. Для любого $n \geq 2$ существует финитный протокол распределения укрытий между n участниками.

Доказательство теоремы 1 опирается на следующую лемму.

[1] Это наименование вполне естественно, если считать, что все вычисления роботы выполняют мгновенно и время при исполнении протокола тратится только на передачу информации по каналам связи.

Лемма 1. Существует финитный протокол щелчка.

Эскиз доказательства. Идея искомого финитного протокола состоит в следующем. Участник A отправляет участнику B свои координаты с точностью до одного знака после запятой. Если после этого участник B ещё не может определить, каков должен быть результат исполнения протокола, он извещает об этом участника A, и тот в ответ отправляет свои координаты с точностью до двух знаков после запятой. Если полученной информации участнику B всё ещё недостаточно, A отправляет ему всё более и более точные свои координаты, пока B не сможет вычислить результат работы протокола. Когда это происходит, B остаётся только известить об этом результате A. Поскольку никакие три из точек A, B, A^0, B^0 не лежат на одной прямой, можно показать, что подобный протокол действительно финитен. □

Доказательство теоремы. Поскольку протокол щелчка — это частный случай модифицированного протокола щелчка, то на основе финитного протокола щелчка строится финитный протокол распределения укрытий (см. п. 1.2). □

Поскольку, как утверждает теорема 1, существует финитный протокол распределения укрытий, возникает вопрос о том, какими из характеристик, описанных в пункте 2.1, он может обладать. Следующая теорема, являющаяся нашим главным результатом в рамках информационной постановки задачи, даёт ответ на этот вопрос.

Теорема 2. Пусть множество S точек на плоскости обладает следующими свойствами.

1. Никакие три точки множества S не лежат на одной прямой.
2. Существует непрерывная кривая Φ, ограничивающая[2] выпуклое множество, такая, что все точки множества S принадлежат Φ.

Кроме того, пусть множество S бесконечно (более чем счётно). Тогда сужение протокола распределения укрытий между $n \geq 2$ участниками на множество S не может быть ограниченным (соответственно частично ограниченным по сообщениям).

Следствие 1. Не существует частично ограниченного по сообщениям протокола распределения укрытий между $n \geq 2$ участниками.

Доказательство. Если бы такой протокол существовал, то существовал бы и частично ограниченный по сообщениям протокол, являющийся ограничением протокола распределения укрытий между n участниками на единичную окружность. Но окружность, очевидно, удовлетворяет условиям теоремы 2. □

Следствие 2. Не существует ограниченного протокола, являющегося сужением протокола распределения укрытий между $n \geq 2$ участниками на множество точек с целыми координатами.

Доказательство. В качестве множества S из условия теоремы 2 достаточно рассмотреть, например, точки с целыми координатами, лежащие на параболе $y = x^2$. □

Замечание. Рассмотрим следующее сужение протокола распределения укрытий между $n \geq 2$ участниками на множество точек с целыми координатами. Все роботы, кроме первого, отправляют свои координаты первому, после чего он самостоятельно подбирает правильное распределение укрытий и рассылает его описание остальным роботам. Как и утверждается в следствии 2, этот протокол не является ограниченным, поскольку точная передача произвольного целого числа может потребовать сколь угодно большого числа бит. Тем не менее, этот протокол, очевидно, частично ограничен, а также ограничен по сообщениям.

План доказательства теоремы 2 будет представлен в следующих пунктах.

2.3. Вспомогательные протоколы

Доказательство теоремы 2 основано на сведении вопросов, касающихся протокола распределения, к изучению свойств более простых вспомогательных протоколов.

Будем называть *протоколом проверки равенства* протокол, удовлетворяющий следующим условиям.

1. В протоколе участвуют две стороны: A и B.
2. Участник (сторона) A знает действительное число x_A, неизвестное участнику B. Участник B знает действительное число x_B, неизвестное участнику A.
3. Протокол позволяет участникам выяснить, верно ли, что $x_A = x_B$.

Будем называть *протоколом сравнения* протокол, удовлетворяющий следующим условиям.

1. В протоколе участвуют две стороны: A и B.

[2] Мы используем слово «ограничивающая» в следующем смысле. Пусть кривая Φ разделяет плоскость на две части. Тогда про каждую из этих частей мы говорим, что Φ её ограничивает. (В частности, кривая Φ может ограничивать неограниченное множество).

2. Участник (сторона) A знает действительное число x_A, неизвестное участнику B. Участник B знает действительное число x_B, неизвестное участнику A. Числа x_A и x_B заведомо неравны.
3. Протокол позволяет участникам за конечное время выяснить, какое из чисел x_A, x_B больше.

Эскиз доказательства теоремы 2. Покажем, как, используя сужение протокола распределения укрытий на множество S, построить сужение протокола сравнения на множество той же мощности, что и S. Рассмотрим такие точки X_1, X_2, …, X_{2n-4} из множества S, что для любых $1 \le i < j \le 2n-2$ точки X_{i+1}, …, X_{j-1} принадлежат одной и той же дуге X_iX_j кривой Φ. Множество таких точек $X \in S$, что все точки X_2, …, X_{2n-2} принадлежат одной и той же дуге X_1X кривой Φ, обозначим через $\uparrow\{X_1, \dots, X_{2n-2}\}$. Можно показать, что точки X_1, …, X_{2n-2} можно выбрать так, чтобы множество $\uparrow\{X_1, \dots, X_{2n-2}\}$ было равномощно множеству S. Зафиксируем какой-нибудь такой набор точек X_1, …, X_{2n-2}. Введём обозначения: $S_1 := X_1$, $S_2 := X_2$, …, $S_n := X_n$, $R_n := X_{n+1}$, …, $R_3 := X_{2n-2}$. Выберем $R_1, R_2 \in \uparrow\{X_1, \dots, X_{2n-2}\}$ и выполним протокол распределения укрытий с роботами, находящимися в точках R_1, R_2, …, R_n, и укрытиями в точках S_1, S_2, …, S_n. Тогда в силу того, что кривая Φ ограничивает выпуклое множество, в результате работы протокола роботу R_i будет поставлено в соответствие укрытие S_i для всех $3 \le i \le n$. При этом роботу R_1 будет поставлено в соответствие укрытие S_1 тогда и только тогда, когда R_2 находится на кривой Φ между R_1 и R_3 (то есть на той же дуге S_1R_1, что и все остальные R_i и S_i, см. рис. 1). Введя на кривой Φ координаты, можно убедиться, что работа такого протокола эквивалентна сужению протокола сравнения на множество координат точек из $\uparrow\{X_1, \dots, X_{2n-2}\}$. Поскольку оно имеет ту же мощность, что и множество S, то в силу свойств протокола сравнения (см. пункт 2.4) если S бесконечно, такой протокол не может быть ограниченным, а если S более чем счётно — частично ограниченным по сообщениям. □

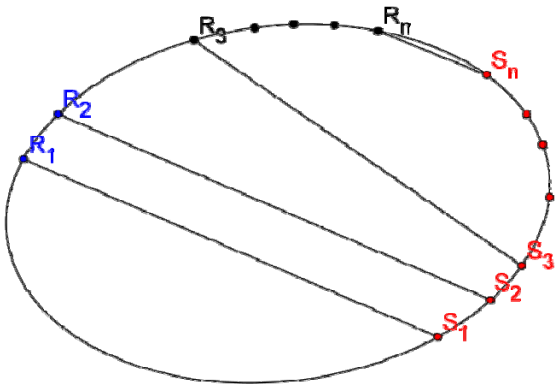


Рис. 1

2.4. Вспомогательные утверждения

Как было показано в пункте 2.3, для доказательства теоремы 2 требуется анализ свойств протокола сравнения и его сужений на различные множества. В этом пункте мы приведём полученные нами результаты этого анализа, а в пункте 2.5 в общих чертах изложим методы, применявшиеся нами при доказательстве следующих утверждений.

Утверждение 1. Не существует финитного протокола проверки равенства.

Утверждение 2. Не существует ограниченного протокола, являющегося сужением протокола сравнения на бесконечное множество I.

Утверждение 3. Не существует частично ограниченного по сообщениям протокола, являющегося сужением протокола сравнения на более чем счётное множество I.

Для доказательства теоремы 2 достаточно утверждений 2 и 3. Однако для полноты исследования мы рассмотрели также сужения протоколов сравнения и проверки равенства на конечные множества. Очевидно, что на конечном множестве любой детерминированный финитный протокол ограничен. Утверждения 4 и 5 содержат ответ на вопрос о точном значении границы необходимого числа бит для протоколов сравнения и проверки равенства на n-элементном множестве.

Утверждение 4. Пусть $n > 1$. Тогда для работы протокола проверки равенства на n-элементном множест-

ве в худшем случае потребуется передача не менее $\left\lceil \frac{1+\log_2 n}{2} \right\rceil + 1$ бит, причём эта оценка достигается.

Утверждение 5. Пусть $n > 2$. Тогда для работы протокола сравнения на n-элементном множестве в худшем случае потребуется передача не менее $\left\lceil \frac{\log_2 n}{2} \right\rceil + 1$ бит, причём эта оценка достигается.

2.5. Эскизы доказательств

Основное понятие, использованное нами при доказательстве утверждений о вспомогательных протоколах — это понятие диалога. *Диалог* $d = \langle m_1, \ldots, m_n \rangle$ —это конечная последовательность конечных битовых строк $m_1, \ldots, m_n$. Мы будем рассматривать диалоги, которые получаются при исполнении того или иного протокола с двумя участниками, состоящие из сообщений, которые участники отправляют друг другу. Например, если первый участник отправил сообщение «001», второй ответил «10», первый ответил «0101», то текущий диалог имеет вид $\langle 001, 10, 0101 \rangle$.

Поскольку все рассматриваемые протоколы считаются детерминированными, то результат работы протокола или следующее сообщение, которое должен отправить участник, однозначно определяется известными ему входными данными и текущим диалогом. Это обстоятельство играет ключевую роль в доказательстве утверждений 1–5. В качестве примеров приведём здесь полные доказательства утверждений 1 и 2.

Доказательство утверждения 1. Предположим противное. Пусть существует финитный протокол проверки равенства. Для данного действительного x через d_x обозначим диалог, которым завершится этот протокол, если входными данными будут $x_A = x_B = x$. Значит, при входном данном x и текущем диалоге d_x участник (неважно, A или B) делает вывод о том, что $x_A = x_B$. Заметим, что множество всех диалогов счётно. Поэтому найдутся такие числа x и y, что $x \neq y$, но $d_x = d_y$. Обозначим $d := d_x$. Рассмотрим работу протокола для входных данных $x_A = x$ и $x_B = y$.

Докажем по индукции, что на каждом этапе работы протокола текущий диалог в этом случае будет совпадать с началом диалога d. В начале текущий диалог пуст, так что база индукции верна. Пусть текущий диалог d' является началом диалога d и, для определённости, следующее сообщение отправляет участник A (для участника B рассуждение аналогично). Тогда следующее сообщение однозначно определяется числом x и диалогом d'. Но, значит, оно должно совпадать со следующим после d' сообщением в d, ведь именно такое сообщение должен отправить участник A после диалога d', если $x_A = x$ (и $x_B = x$). Шаг индукции доказан. Итак, на каждом шаге протокола текущий диалог будет совпадать с началом диалога d. Но, значит, после некоторого числа шагов текущий диалог станет равен d. При этом и участник A с $x_A = x$, и участник B с $x_B = y$ из этого должны сделать вывод, что $x_A = x_B$. Но $x \neq y$, поэтому протокол в этом случае даёт неверный результат. Полученное противоречие завершает доказательство. □

Для доказательства утверждения 2 нам потребуется ещё одно новое понятие. Назовём *финальным* диалог, после которого уже не требуется отправка новых сообщений, то есть после которого протокол завершается. Тогда ограниченность протокола равносильна тому, что он финитен и каждый его финальный диалог содержит не более N бит, где N — некая константа.

Доказательство утверждения 2. Предположим противное. Пусть существует финитный протокол, являющийся сужением протокола сравнения на бесконечное множество I, каждый финальный диалог которого в общей сложности содержит не более N бит. Для неравных действительных чисел $x, y \in I$ обозначим через $d(x, y)$ финальный диалог, который получается при исполнении данного протокола с начальными данными $x_A = x$, $x_B = y$. Через $D(x)$ обозначим множество тех диалогов d, для которых найдётся $y \neq x$ из множества I такое, что $d = d(x, y)$. Поскольку диалогов, содержащих не более N бит, конечное число, множество всех подмножеств множества таких диалогов тоже конечно. В частности, $D(x)$ принимает лишь конечное число значений. Поэтому в бесконечном множестве I найдутся такие $x_1 < y < x_2$, что $D(x_1) = D(y) = D(x_2)$.

Покажем, что $d(x_1, y) = d(x_2, y)$. Доказательство, как и в случае утверждения 1, по индукции от числа отправленных сообщений. Изначально текущие протоколы в обоих случаях пусты, так что база индукции очевидна. Пусть на каком-то шаге текущие протоколы равны d'. Если теперь очередь участника B отправлять сообщение, то он, исходя из того, что $x_B = y$, а текущий диалог равен d', отправит одно и то же сообщение, вне зависимости от того, чему равно x_A. Пусть теперь очередь участника A отправлять сообщение. Предположим, что при $x_A = x_1$ он отправляет сообщение m. Значит, в множестве $D(x_1)$ есть диалог, начинающийся с

(d', m). Но $D(x_1) = D(x_2)$, поэтому такой же диалог есть и в $D(x_2)$. По определению, это означает, что существует такое z, что при $x_A = x_2$, $x_B = z$ и текущем диалоге d' участник A должен отправить сообщение m. Но сообщение, отправляемое A, зависит только от x_A и текущего диалога. Значит, при $x_A = x_2$ после диалога d' первый участник всегда отправляет сообщение m, как и при $x_A = x_1$. Шаг индукции доказан.

Вывод, который делает участник B о результате работы протокола, зависит только от x_B и полученного финального диалога. Значит, если $x_B = y$, а $x_A = x_1$ или $x_A = x_2$, вывод, который участник B сделает о результате работы протокола, будет один и тот же. Но в первом случае $x_A < x_B$, а во втором $x_A > x_B$. Полученное противоречие завершает доказательство. □

Полные доказательства утверждений 3–5 мы здесь не приводим, так как они во многом аналогичны доказательствам утверждений 1 и 2. Лишь кратко опишем использованные в них идеи.

Утверждение 3 доказывается в два этапа. Сначала доказывается, что не существует ограниченного по сообщениям протокола, являющегося сужением протокола сравнения на более чем счётное множество I, а затем из этого факта выводится невозможность и частичной ограниченности такого протокола. При этом, в отличие от доказательства утверждения 2, здесь существенно используется то, что речь идёт не только о протоколе сравнения, но и о его сужениях на различные множества.

Утверждения 4 и 5 доказываются с помощью выписывания некоего рекуррентного неравенства, указывающего нижнюю границу для искомого числа бит.

2.6. Нестрогие протоколы

Назовём *нестрогим протоколом распределения укрытий между* $n \geq 1$ *участниками* (или просто нестрогим протоколом распределения) протокол, отличающийся от протокола распределения заменой того условия, что никакие три из точек (x_1, y_1), …, (x_n, y_n), (x_1^0, y_1^0), …, (x_n^0, y_n^0) не лежат на одной прямой, на требование, чтобы все эти точки были попарно различны и никакие *четыре* из них не лежали на одной прямой. (Последнее условие гарантирует, что распределение укрытий, при котором пути роботов не пересекаются, существует). Аналогично определяются *нестрогий протокол щелчка* и *нестрогий модифицированный протокол щелчка*.

Заметим, что наше доказательство леммы 1 о существовании финитного протокола щелчка существенно использовало тот факт, что никакие три из точек A, B, A^0, B^0 не лежат на одной прямой. Как показывает следующая лемма, без этого условия финитный протокол построить не удаётся.

Лемма 2. Не существует финитного нестрогого протокола щелчка.

Таким образом, не существует протокола, который бы позволил двум роботам проверить, пересекаются ли их пути, за конечное время. Тем не менее, верна следующая теорема.

Теорема 3. Для любого $n \geq 2$ существует финитный нестрогий протокол распределения укрытий между n участниками.

Доказательство теоремы 3 опирается на следующую лемму.

Лемма 3. Существует финитный нестрогий модифицированный протокол щелчка.

Можно показать, что финитного нестрогого протокола сравнения расстояний, как и финитного нестрогого протокола щелчка, не существует. Поэтому для доказательства леммы 3 приходится строить некую комбинацию этих двух протоколов.

Доказательство леммы 2 состоит в сведении нестрогого протокола щелчка к *нестрогому протоколу сравнения* — протоколу, отличающемуся от протокола сравнения отсутствием условия $x_A \neq x_B$ и позволяющему участникам проверить, верно ли, что $x_A \leq x_B$. В отношении нестрогого протокола сравнения верно следующее утверждение.

Утверждение 6. Не существует финитного нестрогого протокола сравнения.

Доказательство утверждения 6 аналогично доказательству утверждения 1. Интересный результат даёт попытка перенести это доказательство на случай обычного протокола сравнения (который, очевидно, может быть финитным). Результатом является следующее утверждение.

Утверждение 7. Зафиксируем некий финитный протокол сравнения и диалог d. Пусть S — множество таких пар (x, y), что $d(x, y) = d$ (в обозначениях доказательства утверждения 2, см. п. 2.5). Предположим, что множество S содержит более двух элементов. Тогда для всех пар (x, y) из S неравенство $x < y$ или одновременно выполняется, или одновременно не выполняется. Кроме того, существует такое число z, что для всех пар (x, y) из S число z лежит между x и y.

3. Криптографическая постановка задачи

В отличие от информационной постановки теперь мы будем считать, что входные данные (координаты роботов и укрытий) берутся из некоторого конечного множества. В качестве основного примера мы будем рассматривать множество знаковых чисел с фиксированной запятой, в представлении которых участвуют N бит. Заметим, что в этом случае смысл протокола распределения укрытий состоит в вычислении некоторой функции координат роботов и укрытий, значением которой является искомое распределение. В силу теоремы о конфиденциальных вычислениях («secure multi-party computation», см. [2,3]), существует способ вычисления этой функции, при котором участники не получат никаких дополнительных сведений о входных данных друг друга. К сожалению, прямое описание этого способа чрезмерно громоздко и вряд ли выполнимо на практике. Поэтому мы несколько упростим себе задачу. Именно, будем считать, что участники пользуются каким-то протоколом распределения укрытий, основанном на щелчках с обменом (см. пункт 1.2). Наша цель, таким образом, состоит в построении *протокола щелчка*, при котором участники бы не раскрывали друг другу никаких дополнительных сведений о своих координатах. Участников мы будем считать получестными («честными, но любопытными»). Это означает, что они строго следуют протоколу, хотя и пытаются из сообщений друг друга извлечь как можно больше дополнительной информации. Такой подход полностью согласуется с интерпретацией участников как роботов, которые строго следуют заложенной в них программе и, кроме того, заинтересованы в правильности работы протокола.

3.1. Протокол конфиденциального вычисления схемы

Известны различные протоколы конфиденциального вычисления логических схем (см. [2,3]). Мы будем пользоваться одним из них, который окажется очень удобен для наших целей. Здесь мы изложим его для случая двух участников (поскольку именно протоколы с двумя участниками будут нас интересовать).

Пусть требуется вычислить некоторую логическую функцию f, часть аргументов которой известны участнику A, а часть — участнику B. При этом участники не хотят раскрывать друг другу сведений о своих входных данных. Функция f записывается с использованием только операций конъюнкции, отрицания и исключающего «или». Затем её вычисление представляется в виде логической схемы, на входы которой подаются аргументы функции, а на выходе получается её значение. Теперь будут последовательно вычисляться значения в узлах схемы, но не полностью, а «в разделённом виде». Это значит, что для каждого узла i у участников получатся биты a_i и b_i такие, что значение в узле равно $a_i \oplus b_i$. При этом бит a_i известен только участнику A, а бит b_i — только участнику B. Для этого участники делают следующее. Для каждого своего бита p участник создаёт два случайных бита a и b, такие что $a \oplus b = p$. Один из этих битов он посылает второму участнику. Таким образом оказываются «разделены» входные данные. Теперь осталось описать, как вычислять результаты логических операций. С отрицанием и исключающим «или» всё просто — чтобы вычислить отрицание, надо взять отрицание соответствующего бита первого участника, а чтобы вычислить исключающее «или», каждый участник должен вычислить исключающее «или» для своей «части» значения (см. рис. 2).

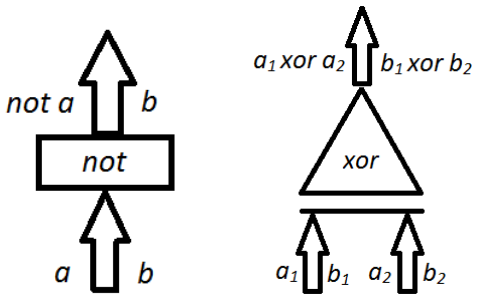


Рис. 2

Некоторая сложность возникает при вычислении конъюнкции. Пусть на входе имеются биты a_1, b_1, a_2 и b_2. Требуется получить такие биты a и b, что $a \oplus b = (a_1 \oplus b_1)\&(a_2 \oplus b_2)$. Поскольку исключающее «или» и конъюнкция — это не что иное как сложение и умножение по модулю 2 соответственно, то мы получаем, что по модулю 2 $a + b = a_1a_2 + a_1b_2 + b_1a_2 + b_1b_2$. Поскольку бит a_1a_2 известен первому участнику, а бит b_1b_2 — второму, им теперь нужно вычислить биты c и d такие, что $c + d = a_1b_2 + b_1a_2$. Тогда можно положить $a := a_1a_2 + c$ и $b := b_1b_2 + d$. Биты c и d вычисляются с помощью следующего протокола. Участ-

ник A генерирует случайный бит x. Затем он полагает $s_{00} := x$, $s_{01} := x + a_1$, $s_{10} := x + a_2$, $s_{11} := x + a_1 + a_2$. С помощью протокола передачи данных вслепую «1 из 4» (она же забывчивая передача, передача с забыванием, oblivious transfer, см. [2,3]) участник B узнаёт значение бита $s_{b_2 b_1}$. Теперь нужно положить $c := x$, $d := s_{b_2 b_1}$.

После того, как выходные биты оказываются вычислены в «разделённом» виде, участникам остаётся только обменяться своими «частями» результата.

Важным для нас свойством этого протокола является то, что при его использовании очень легко вычисляются отрицания и исключающие «или». Именно этот факт делает его удобным для наших целей.

3.2. Конфиденциальный протокол щелчка

Для начала выразим условие, при котором отрезки AA^0 и BB^0 пересекаются. Поскольку из точек A, A^0, B, B^0 никакие три не лежат на одной прямой, отрезки AA^0 и BB^0 пересекаются тогда и только тогда, когда точки A и A^0 лежат по разные стороны от прямой BB^0, а точки B и B^0 — по разные стороны от прямой AA^0. Будем считать, что наш протокол возвращает 0, если отрезки не пересекаются, и 1, если пересекаются. Обозначим через $sides(A, B, C, D)$, где A, B, C, D — некоторые точки на плоскости, утверждение «точки A и B лежат по разные стороны от прямой CD». Тогда задача протокола щелчка — вычислить истинность конъюнкции $sides(A, A^0, B, B^0) \& sides(B, B^0, A, A^0)$.

Опишем, как вычислять $sides(A, A^0, B, B^0)$ (ясно, что для $sides(B, B^0, A, A^0)$ способ вычисления аналогичен). Переместим начало координат в точку B^0. Обозначим координаты точек в этой системе следующим образом: $A(x_2, y_2)$, $A^0(x_0, y_0)$, $B(x_1, y_1)$, $B^0(0,0)$. Уравнение прямой BB^0 в этой системе координат есть $y_1 x - x_1 y = 0$. Поэтому точки A и A^0 лежат от этой прямой по разные стороны, если и только если знаки чисел $y_1 x_2 - x_1 y_2$ и $y_1 x_0 - x_1 y_0$ различны.

Теперь заметим, что $x_0 = x_A^0 - x_B^0$, $y_0 = y_A^0 - y_B^0$, $x_1 = x_B - x_B^0$, $y_1 = y_B - y_B^0$, $x_2 = x_A - x_B^0$, $y_2 = y_A - y_B^0$. Поэтому числа x_0, y_0 могут вычислить оба участника, числа x_1, y_1 может вычислить участник B, а числа x_2, y_2 — участник A. При этом понятно, что эти числа также принадлежат некоторому конечному множеству. В нашем основном примере это множество знаковых чисел с фиксированной запятой, в представлении которых участвует $N + 1$ бит (при этом количество бит после запятой не изменяется).

Итак, участники вычислили новые координаты точек. Далее мы будем поэтому использовать их в качестве основных параметров вместо исходных координат. Теперь им нужно узнать $\mathrm{sign}(y_1 x_2 - x_1 y_2) \oplus \mathrm{sign}(y_1 x_0 - x_1 y_0)$ ($\mathrm{sign}(x)$ по определению полагается равным 1, если $x > 0$ и 0, если $x < 0$). Заметим, что $\mathrm{sign}(y_1 x_0 - x_1 y_0)$ участник B может вычислить самостоятельно. Рассмотрим теперь $\mathrm{sign}(y_1 x_2 - x_1 y_2)$. Требуется установить, истинно ли неравенство $y_1 x_2 - x_1 y_2 > 0$. Оно эквивалентно $y_1 x_2 > x_1 y_2$. Это, в свою очередь, эквивалентно следующему: $\left(\frac{y_2}{x_2} > \frac{y_1}{x_1}\right) \oplus (x_1 < 0) \oplus (x_2 < 0)$, поскольку при делении на отрицательное число неравенство изменяет знак. Заметим, что если в этом выражении при необходимости деления на 0 результат положить условно равным ∞ или $-\infty$ (в зависимости от знака числителя), то значение выражения также окажется равным $\mathrm{sign}(y_1 x_2 - x_1 y_2)$.

Положим теперь $err_A := (x_2 < 0)$, $err_B := (x_1 < 0) \oplus (y_1 x_0 - x_1 y_0 > 0)$. Заметим, что участник A может самостоятельно вычислить err_A, а участник B — err_B. Таким образом, задача сводится к вычислению $\left(\frac{y_2}{x_2} > \frac{y_1}{x_1}\right) \oplus err_A \oplus err_B$.

Теперь рассмотрим числа $\frac{y_2}{x_2}$ и $\frac{y_1}{x_1}$. Множество значений, которые они могут принимать, конечно. Поэтому все эти значения можно монотонно занумеровать натуральными числами. Для дальнейшего не принципиально, как именно происходит эта нумерация (понятно, что способ нумерации зависит от представления чисел в памяти). Покажем, как можно сделать это в нашем основном примере.

Пусть x, y — знаковые числа с фиксированной запятой, в представлении которых используется $N + 1$ бит. Наша цель получить натуральный номер для числа $\frac{x}{y}$, причём монотонно возрастающий с ростом этой дроби. Можно показать, что если $y \neq 0$, то, вычислив число $\frac{x}{y}$ с точностью до $2N$ знаков после запятой, мы сможем по результату его однозначно восстановить. При этом для хранения целой части этого числа потребуется

ещё N бит, плюс один бит для знака. Заметим, что наибольшие по модулю отрицательное и положительное числа, которые можно хранить подобным образом, не могут получиться в результате таких вычислений. Поэтому эти два числа можно считать записью ∞ и $-\infty$. Итак, мы монотонно занумеровали дроби $\frac{x}{y}$ знаковыми числами с фиксированной запятой. Убрав запятую, мы получим знаковые целые числа, которые также монотонно нумеруют дроби $\frac{x}{y}$. Чтобы превратить их в натуральные, осталось увеличить их на модуль наибольшего по модулю отрицательного числа, которое есть среди них. Поскольку хранить знак больше не нужно, количество бит в числе при этой операции не увеличится. Таким образом, мы монотонно занумеровали дроби $\frac{x}{y}$ натуральными числами, содержащими $3N + 1$ бит.

Пусть натуральное число, соответствующее $\frac{y_1}{x_1}$, равно n_B, а число, соответствующее $\frac{y_2}{x_2}$, равно n_A. Участник A самостоятельно вычисляет n_A, а участник B — n_B. Таким образом,

$$sides(A, A^0, B, B^0) = (n_B > n_A) \oplus err_A \oplus err_B.$$

Аналогично, $sides(B, B^0, A, A^0) = (\tilde{n}_A > \tilde{n}_B) \oplus \tilde{err}_A \oplus \tilde{err}_B$, где переменные с волнами определяются как и переменные без волн с той лишь разницей, что участник A и его входные данные заменяется всюду участником B и его входными данными, и наоборот. Тогда результат щелчка есть

$$\big((n_B > n_A) \oplus err_A \oplus err_B\big)\&\big((\tilde{n}_A > \tilde{n}_B) \oplus \tilde{err}_A \oplus \tilde{err}_B\big).$$

Осталось представить сравнение двух натуральных чисел как логическую функцию. Пусть первое число есть $d_1 d_2 \ldots d_k$, а второе $e_1 e_2 \ldots e_k$ (в двоичной системе). Напомним, что в нашем основном примере $k = 3N + 1$. Тогда первое число больше второго тогда и только тогда, когда существует такое $i \le k$, что $d_1 = e_1$, …, $d_{i-1} = e_{i-1}$, а $d_i = 1$, $e_i = 0$. Положим $x_i := (\neg d_i) \oplus e_i$. Биты x_i можно считать входными данными, которые уже «разделены» между участниками, поскольку $x_i = (\neg d_i) \oplus e_i$. Тогда первое число больше второго тогда и только тогда, когда

$$(\neg x_1 \& \neg e_1) \vee (x_1 \& \neg x_2 \& \neg e_2) \vee (x_1 \& x_2 \& \neg x_3 \& \neg e_3) \vee \ldots \vee \big((x_1 \& x_2 \& \ldots \& x_{k-1} \& \neg x_k \& \neg e_k)\big).$$

Заметим, что из всех альтернатив, перечисленных в этом выражении, может иметь место не более одной, поэтому все дизъюнкции можно заменить на исключающие «или». Имеем:

$$(\neg x_1 \& \neg e_1) \oplus (x_1 \& \neg x_2 \& \neg e_2) \oplus (x_1 \& x_2 \& \neg x_3 \& \neg e_3) \oplus \ldots \oplus \big((x_1 \& x_2 \& \ldots \& x_{k-1} \& \neg x_k \& \neg e_k)\big).$$

Мы не будем здесь рисовать саму логическую схему, а просто опишем, как она устроена. Чтобы минимизировать количество конъюнкций, сперва следует вычислить значения $y_i := x_1 \& \ldots \& x_i$ (вычисляя y_{i+1} как $y_i \& x_{i+1}$) для всех i от 1 до $k - 1$. Для этого нужно совершить $k - 2$ конъюнкции. Затем нужно вычислить $z_i := y_{i-1} \& \neg x_i \& \neg e_i$ (полагая $z_1 := \neg x_1 \& \neg e_1$) для всех i от 1 до k. Для этого потребуется ещё $2k - 1$ конъюнкция. Затем вычисляется окончательное значение, равное $z_1 \oplus \ldots \oplus z_k$. Как видим, в общей сложности нам потребовалось $3k - 3$ операции конъюнкции. В нашем основном примере это даёт $9N$ операций. Заметим, что это количество линейно зависит от k, то есть от длины сравниваемых чисел.

Поскольку нашей целью является не просто сравнить два числа, а выполнить щелчок, участники не должны останавливаться на вычислении $z_1 \oplus \ldots \oplus z_k$, а тем же способом вычислить значение второго неравенства, а затем с помощью двух исключающих «или» и одной конъюнкции получить окончательное значение $\big((n_B > n_A) \oplus err_A \oplus err_B\big)\&\big((\tilde{n}_A > \tilde{n}_B) \oplus \tilde{err}_A \oplus \tilde{err}_B\big)$, рассматривая $err := err_A \oplus err_B$ и $\tilde{err} := \tilde{err}_A \oplus \tilde{err}_B$ как две новые уже «разделённые» переменные. Таким образом, мы описали протокол щелчка, при котором участники не сообщают друг другу никаких дополнительных сведений о своих координатах, то есть решили поставленную задачу.

3.3. Протокол распределения укрытий для двух участников

В качестве дополнения построим на основе описанного выше протокола щелчка протокол конфиденциального распределения укрытий между двумя участниками (напомним, что до этого мы рассматривали не произвольный протокол распределения, а протокол распределения, основанный на щелчках).

Дважды выполним протокол щелчка: сначала для одного распределения укрытий, потом для второго. При этом в конце работы протоколов участники не должны обмениваться своими результатами. Таким образом, результат работы одного протокола будет равен $r_0 = a_0 \oplus b_0$, а другого — $r_1 = a_1 \oplus b_1$, причём биты a_0 и a_1 известны первому участнику, а биты b_0 и b_1 — второму. Искомый протокол должен возвращать 0, если

$r_1 = 1$, 1, если $r_0 = 1$, и случайный бит, если $r_0 = r_1 = 0$ (заметим, что не может быть, чтобы $r_0 = r_1 = 1$). Для этого первый участник генерирует случайный бит c_1, а второй — случайный бит c_2. Обозначим $c := c_1 \oplus c_2$. Как видим, значение c уже «разделено» между участниками. Положим результат работы протокола равным $r_0 \oplus (\neg(r_0 \oplus r_1) \& c)$. Тогда при $r_0 = 1$ он будет равен 1, при $r_1 = 1$ — 0, а при $r_0 = r_1 = 0$ — c, случайному биту, неизвестному ни одному из участников, что и требовалось.

3.4. Конфиденциальный модифицированный протоколы щелчка

В пункте 3.2 был описан конфиденциальный протокол щелчка. Однако, как отмечалось в пункте 1.2, протокол распределения укрытий можно построить не только на основе протокола щелчка, но и на основе любого модифицированного протокола щелчка. Напомним свойства, которым должен удовлетворять такой протокол.

1. Если текущие пути роботов пересекаются, то они должны совершить обмен укрытиями.
2. После обмена укрытиями суммарная длина путей роботов всегда строго уменьшается.

Очевидно, в смысле сокрытия информации лучшим модифицированным протоколом щелчка был бы такой, который удовлетворяет следующим условиям.

1. Если пути роботов пересекаются, то происходит обмен.
2. Если после обмена укрытиями суммарная длина путей не уменьшится, то обмена не происходит.
3. Если не выполнены условия (I), (II), то случайным образом, непредсказуемым ни для какого участника, определяется, происходит ли обмен.

В этом пункте мы опишем, как должен может быть устроен такой протокол. Он будет использовать протокол щелчка, описанный в пункте 3.2, а также конфиденциальный протокол сравнения расстояний, который мы опишем в пункте 3.5.

Участники выполняют протоколы щелчка и сравнения расстояний, но не обмениваются результатами. Обозначим эти результаты $c = c_A \oplus c_B$ ($c = 1$ тогда и только тогда, когда текущие пути роботов пересекаются) и $d = d_A \oplus d_B$ ($d = 1$ тогда и только тогда, когда после обмена суммарная длина путей уменьшится). Теперь участники генерируют случайные биты x_A и x_B. Тогда $x = x_A \oplus x_B$ — случайный бит, неизвестный никому из участников и уже «разделённый» между ними. Нам нужно указать логическую функцию $f(c, d, x)$ такую, что если $c = 1$, то $f = 1$, если $d = 0$, то $f = 0$, а если $c = 0$ и $d = 1$, то $f = x$ (заметим, что невозможно, чтобы $c = 1$ и $d = 0$ одновременно). Обмен нужно совершать тогда и только тогда, когда $f = 1$. Можно положить $f = c \oplus ((c \oplus d) \& x)$. В самом деле, при $c = 1$ получаем $f = 1$, при $d = 0$ получаем $f = 0$, а при $c = 0$ и $d = 1$ получаем $f = x$, что и требовалось.

Отметим, что, хотя сам по себе описанный выше протокол обеспечивает лучшую сохранность информации, чем протокол щелчка, требуется дополнительное исследование вопроса о том, какой из них более целесообразно применять в протоколе распределения укрытий, поскольку данный протокол может потребовать больше итераций.

3.4. Конфиденциальный протокол сравнения расстояний

Протокол конфиденциального сравнения расстояний устроен проще, чем протокол щелчка. Заметим, что условие $|AA^0| + |BB^0| > |AB^0| + |BA^0|$ равносильно такому $|AA^0| - |AB^0| > |BA^0| - |BB^0|$. Левую часть этого неравенства может самостоятельно вычислить участник A, а правую — участник B. При этом обе они могут принимать лишь конечное число значений, так как координаты точек берутся из конечного множества. Поэтому их можно монотонно занумеровать натуральными числами. Таким образом, задача свелась к конфиденциальному сравнению двух натуральных чисел. Эта задача подробно обсуждалась в пункте 3.2.

Опишем теперь, как монотонно занумеровать натуральными числами сравниваемые выражения в нашем основном примере, то есть когда координаты точек — это знаковые числа с фиксированной запятой, в представлении которых участвуют N бит. Для удобства можно считать координаты целыми, поскольку, убрав запятую, мы изменим все их в одно и то же число раз. Запишем выражение в левой части неравенства в координатах (второе записывается аналогично): $\sqrt{(x_A - x_A^0)^2 + (y_A - y_A^0)^2} - \sqrt{(x_A - x_B^0)^2 + (y_A - y_B^0)^2}$. Разности координат вычисляются точно. Для их представления требуется $N + 1$ бит. Для точного представления их квадратов достаточно $2N$ бит. Сумма двух таких чисел также вычисляется точно и требует $2N + 1$ бит. Пусть

$x > y$ — два таких числа. Тогда $\sqrt{x} - \sqrt{y} = \frac{x-y}{\sqrt{x}+\sqrt{y}} > \frac{1}{2\sqrt{2^{2N+1}}} > \frac{1}{2^{N+2}}$. Поэтому корень из натурального числа, представимого $2N + 1$ битом, достаточно вычислять с точностью до $N + 2$ знаков после запятой. При этом на хранение целой части этого корня достаточно отвести $N + 1$ бит, так как она заведомо меньше, чем 2^{N+1}. Итак, для хранения каждого из корней в виде беззнакового числа с фиксированной запятой нам требуется $2N + 3$ бита. Модуль их разности тогда потребует $2N + 4$ бита. Поскольку разность уже может быть отрицательной, нам осталось добавить один бит для знака. Итак, мы представили сравниваемые числа в виде знаковых чисел с фиксированной запятой, в представлении которых участвуют $2N + 5$ бит. Чтобы сделать их натуральными, мы поступим так же, как и в пункте 3.2: сначала уберём запятую, а потом увеличим их на модуль наибольшего по модулю отрицательного числа, которое есть среди них. Итак, мы монотонно занумеровали сравниваемые выражения натуральными числами, содержащими $2N + 5$ бит.

4. Заключение

В заключение ещё раз перечислим наши основные результаты по каждому направлению исследований.

В рамках информационной постановки задачи была доказана теорема, утверждающая, что для широкого класса множеств сужение протокола распределения укрытий на эти множества не может быть ограниченным или даже частично ограниченным по сообщениям.

В рамках криптографической постановки задачи был построен протокол, позволяющий роботам проверить, пересекаются ли их текущие пути, без раскрытия дополнительной информации о своём положении. Также был построен другой протокол, который может использоваться в протоколе распределения укрытий и при однократном повторении позволяет роботам скрыть больше информации о своих координатах, чем протокол щелчка.

Список литературы

[1] Бодин Е.В., Гаранина Н.О., Шилов Н.В. Задача о роботах на Марсе (мультиагентный подход к задаче Дейкстры).

[2] Б. Шнайер. Прикладная криптография. Протоколы, алгоритмы, исходные тексты на языке Си. — М.: Издательство ТРИУМФ, 2002—816 с.:ил.

[3] O. Goldreich. “Foundations of Cryptography”.

Система типов как онтология

Д.Ю. Власов

Институт математики им. С.Л. Соболева СО РАН
Россия, 630090, Новосибирск
пр. Акад. Коптюга, 4
vlasov@academ.org

Аннотация Система типов языка формальной математики Russell может рассматриваться как онтология, на которой помимо родо-видового отношения задается структура, индуцированная грамматикой рассматриваемого языка и системой определений, полностью описывающих содержание производных понятий. В силу того, что на языке Russell может быть представлено не только математическое знание, но вообще говоря произвольное, систему типов Russell можно считать ещё одной формальной моделью онтологии.

1 Онтология математики

Традиционный взгляд на математику как на формальную систему утверждений, выводимых их некоего базового набора аксиом по заданным правилам вывода, не предполагал явной осуществимости процедуры формального представления сколь-нибудь значительной части реальной математики в формальном виде. Процедура формализации рассматривалась чисто теоретически, как предельная форма ясности и точности математического знания, что позволяло проводить метаматематические рассуждения и математически обосновывать те или иные формы оснований математики при помощи аппеляции к хорошим метаматематическим свойствам этих оснований. Поэтому такое понятие как "определение" (имеется в виду математическое определение нового объекта через уже известные на формальном уровне) не рассматривалось как принципиально важное. Как правило, все сводилось к интуитивно очевидной возможности замены определяемого на его определение, и поэтому было теоретически элиминируемым. Понимание определения как "сокращения записи" относится именно к этому подходу.

Ситуация начала меняться с развитием компьютерной техники, когда появилась реальная возможность оперировать огромными массивами данных, в частности, формализованными теориями. Старая постановка вопроса об основаниях математики претерпела изменения: если раньше явная формализация математического знания представлялась чисто умозрительной процедурой, то теперь встал вопрос о реальном представлении всего математического знания в формальном виде на некотором языке, допускающем как минимум процедуру автоматической проверки всех доказательств. При этом старая неформальная интерпретация определения как "сокращения записи" стала неудовлетворительной. Встал вопрос о том, как точно сформулировать понятие "определения", о статусе определений и о математических свойствах определений.

Несмотря на то, что до сих пор идут споры о том, что понимать под онтологией, есть общее семантическое ядро этого понятия, которое описывает онтологию как систему понятий и их отношений. В этом смысле, терминологический аппарат математики, а именно система математических понятий и некая структура, определенная на них, может считаться онтологией математики. При этом можно отделить "базовый уровень" математики как набор аксиом и правил вывода для данной формализации, и "терминологическую надстройку", которая есть система понятий, определяемых через исходные понятия базового уровня. При этом в онтологии также выделяются эти два уровня: базовые понятия и производные.

Очевидно необходимым требованием к производным понятиям является требование консервативности расширения формальной системы посредством добавления определений (такие расширения будем называть дефинициальными). Свойство консервативности заключается в следующем: если некоторое утверждение, не содержащее вхождений некоторого понятия выводится из множества утверждений также не содержащих этого понятия, и вывод включает в себя вхождения этого понятия, то должен также существовать вывод данного утверждения не содержащий вождений рассматриваемого понятия. Это свойство влечет, например, гарантию непротиворечивости дефинициальных расширений исчисления предикатов, что крайне важно с прикладной точки зрения. Действительно, если не гарантировать непротиворечивости дефинициальных расширений, то формальная система может стать случайно (или намеренно) противоречивой, когда разные контрибьюторы в эту систему введут противоречивые определения.

2 Уровни онтологии в Russell

Язык Russell - это металогический язык для описания формальных дедуктивных систем, представимых в виде схем аксиом и правил вывода. В частности, в Russell можно определить исчисление предикатов и сформулировать аксиомы теории множеств, то есть можно представить математику с классическим основанием в виде аксиоматической теории множеств. Пакет, содержащий исходные тексты программной реализации языка Russell, базу теорем и набор тестовых скриптов можно загрузить с сайта по адресу `http://russellmath.org/`. Опишем онтологическую систему языка Russell. В языке Russell можно выделить три уровня онтологии.

Первый уровень онтологии - система типов. Это самый высокий уровень, на котором определяются сами типы и базовое отношение частичного порядка между ними. Система типов в Russell - это статическая система, с отношением супертип-подтип. Можно считать, что типы в Russell - это и есть онтологические единицы, то есть "понятия". Отношение супертип-подтип в данном случае есть в точности отношение "общее-частное". Следует отметить, что в отличии от языков программирования, в которых правильное понимание отношения подтип-супертип предполагает динамический аспект (т.е. наследуется поведение объекта), в языке Russell отношение подтип-супертип не предполагает никакой динамики, то есть оно чисто статическое. Поэтому в Russell тип "круг" естественным образом будет подтипом для типа "эллипс", в то время как данные типы представляют собой классический пример неудачного применения наследования в ООП. В дальнейшем систему типов будем считать частично упорядоченной структурой $(T, \leq_{\mathrm{t}})$, где T - множество типов и $\leq_{\mathrm{t}}$ - отношение подтип-супертип.

Второй уровень онтологии - правила грамматики. На этом уровне определяются синтаксические структуры, задающие понятия как выражения. Выражения - это базовые единицы, из которых строятся все остальные конструкции языка Russell: утверждения, доказательства, определения. Поскольку Russell - это язык с типами, то каждое выражение имеет тип, включая переменные. На этом уровне онтологии происходит важное отождествление: типы отождествляются с нетерминальными символами грамматики. Поэтому введение правил грамматики задает дополнительную структуру на систему типов (т.е. онтологию), которая определяет синтаксис выражений. Следует отметить, что на этом уровне содержание понятий ещё не определено.

Третий уровень онтологии - определения. Как уже отмечалось выше, если некоторе понятие не является базовым, то оно должно как то формально определяться через другие, и в конечном итоге все должно сводиться к базовым понятиям. В Russell для определений вводится специальная синтаксическая конструкция, которая выделяет в определении три объекта:

- определяемое (defiendum),
- определение (definiens),
- форму определения.

Определяемое d_m - это новая синтаксическая конструкция, тип t_m которой представляет собой новое понятие, вводимое в онтологию. Определение d_s - это некоторый терм, представляющий собой определение этой новой синтаксической конструкции. При этом тип определяемого t_s должен быть подтипом определения, то есть между соответствующими понятиями должно быть отношение подтип-супертип. Форма определения d_f - это выражение с двумя переменными v_m и v_s, причем типы этих переменных должны быть супертипами для t_m и t_s соответственно. В языке Russell в конечном итоге определения преобразуются в аксиомы при помощи подстановки выражений d_m и d_s вместо переменных v_m и v_s в выражении d_f.

Поясним, что такое определение в языке Russell, на примере. Рассмотрим классическое исчисление высказываний, в котором связки $\neg, \vee$ и $\leftrightarrow$ определены стандартно. Мы хотим добавить в эту формальную систему связку $\rightarrow$ (импликацию). Пусть система типов исходной формальной системы состояла из единственного типа wff - правильно построенная базовая формула. Введем новый тип $wff_{\rightarrow}$ - правильно построенная при помощи импликации формула, и тогда расширенная система типов состоит из двух типов: $T = \{wff, wff_{\rightarrow}\}$, при этом имеет место $wff_{\rightarrow} \leq_{\mathrm{t}} wff$. Теперь определим компоненты определения:

- определяемое: $d_m = (A \rightarrow B)$ - типа $wff_{\rightarrow}$, переменные A и B - типа wff,
- определение: $d_s = (\neg A \vee B)$ - типа wff и переменные A и B - тоже типа wff,
- форма определения: $d_f = (V_m \leftrightarrow V_s)$ - типа wff, здесь переменные V_m и V_s имеют тип wff.

Тогда при подстановке в выражение d_f вместо переменной V_m выражения d_m, и вместо переменной V_s выражения d_s получим следующее выражение: $((A \rightarrow B) \leftrightarrow (\neg A \vee B))$, которое в виде аксиомы описывает присоединение символа импликации к исходной (базовой) формальной системе.

Таким образом, помимо базового отношения $\leq_t$ и структуры формальной грамматики, на системе типов (т.е. на множестве понятий онтологии) задается ещё одна структура - структура определений, которая описывает содержательный смысл этих понятий. Поскольку структура определений описывает понятия онтологии дедуктивной системы исчерпывающим образом, дальнейшего углубления уровней онтологии не требуется.

Определения языка Russell обладают следующим хорошим свойством: при выполнении набора простых условий, гарантируется консервативность дефинициального расширения исчисления. Эти условия сводятся к проверке выражения d_m на простоту (то есть в выражении d_m не должно быть собственных подвыражений кроме переменных), к проверке того, что d_m не входит в выражение d_s и к проверке того, что если в выражении d_f обе переменных заменить на d_s, то полученная аксиома будет входит в список доказанных утверждений.

3 Система понятий и поиск доказательств

Следует отметить, что разветвленная система типов (т.е. разветвленная онтология) не является необходимым условием формализации математики. А именно, для формализации математики в теории множеств ZFC можно ограничиться всего двумя типами: формула и множество. Если использовать теорию множеств с классами (например систему аксиом Гёделя-Бернайса GB), потребуется для типа множества ввести супертип для типа множество - класс. Но в любом случае, такая система типов (онтология) будет очень бедна. Существующая формальная теория для математики на базе исчисления предикатов и теории множеств с классами, оттранслированная из базы теорем языка metamath, насчитывает в точнсти три типа: формула, класс и множество.

Помимо концептуальной функции отражения системы понятий реальной математики в системе типов, наличие богатой системы типов (онтологии) важно и для такого прикладного вопроса, как автоматический поиск доказательств. Дело в том, что если система типов крайне бедна, то при построении дерева вариантов, когда производится унификация текущего выражения с заключениями уже доказанных утверждений, будет найдено огромное количество вариантов, формально подходящих для унификации, но реально избыточных. Например, пусть в выражении из текущего узла дерева вариантов есть переменная, неформально соответствующая типу “группа”, но заключение некотрой теоремы из теории топологических пространств формально унифицируется с данным выражением, причем переменной неформального типа “топологическое пространство” соответствует переменная типа неформального “группа”. На уровне языка обе переменные буду типа “множество”, поэтому такая унификация будет корректной. Между тем очевидно, что если бы система типов включала бы в себя типы (понятия) “группа” и “топологическое пространство”, причем между этими типами (понятиями) не было бы отношения подтип-супертип, то такая унификация уже бы не прошла.

Поэтому разветвленная система типов может существенно облегчить задачу автоматического поиска доказательств за счет раннего отсечения лишних ветвлений на основе анализа типов выражений.

4 Применение Russell для представления знаний

Несмотря на то, что изначально язык Russell разрабатывался для формального представления математического знания, он может быть использован и как дедуктивная система для представления иных типов знания. При этом, система типов будет точно также представлять систему понятий формализуемой предметной области, и такие хорошие свойства, как консервативность дефинициальных аксиоматических расширений, сохранится.

В данном контексте возникает новая задача, явно не встающая при формальном представлении математики: задача объединения нескольких дедуктивных систем (модульное представление знаний). Ясно, что дизъюнктное объединение дедуктивных систем не представляет большого интереса. Поэтому предлагается ввести операцию синтеза дедуктивных систем при помощи отождествления типов (понятий) этих систем. Поясним на примере. Предположим, что у нас есть две формальные дедуктивные системы, представляющие физику и химию. Изначально будем считать их дизъюнктными. Поэтому из механическое объединение не даст ничего нового. Пусть в обоих системах есть тип (понятие) “атом”. Что произойдет, если мы отождествим тип “атом” в физической части объединенной системы с типом “атом” в химической части системы? Во-первых, если тип “атом” был связан отношением подтип-супертип с некоторым другим типом в одной части, то тип “атом” в другой станет также связан этим отношением. Во-вторых, при

постоении синтаксических конструкций, в нетерминал типа “атом” одной системы мы сможем подставлять выражение типа “атом” другой системы. Ну и в-третьих, если у нас есть содержательная аксиома из одной части, в которой есть переменная типа “атом”, то опять-таки, мы сможем использовать эту аксиому и для выражения типа “атом” из другой части. Далее, если в обоих системах есть понятия “масса” и “плотность”, то мы можем отождествить и их соотвественно. Таким образом, отождествляя пары типов из дизъюнктных объединений дедуктивных систем, можно получить нетривиальные слияния этих систем.

Онтологический подход к построению и развитию информационных Wiki-систем

Ю.А. Загорулько[1], В.К. Шестаков[2]

[1] Институт систем информатики имени А.П. Ершова СО РАН, г. Новосибирск, Россия
[2] Новосибирский государственный университет, Новосибирск, Россия
zagor@iis.nsk.su, zfc@ngs.ru

Аннотация. В работе рассматривается подход к разработке и сопровождению Wiki-систем, базирующийся на использовании онтологий. Проблема состоит в том, что существующие средства построения Wiki-систем отслеживают в создаваемых системах только структурную целостность ссылок, но не логическую целостность и семантическую согласованность используемых в них понятий (категорий). Идея предлагаемого подхода состоит в создании инструментария, который бы обеспечивал построение Wiki-систем с согласованной системой понятий (семантически согласованных Wiki-систем). Wiki-систему с такими свойствами можно получить, если строить ее на основе логически непротиворечивой онтологии, описывающей предметную область будущей системы. В докладе рассматривается подход к построению информационных Wiki-систем на основе онтологий предметных областей, а также необходимый для полноценной реализации этого подхода метод извлечения онтологий из существующих Wiki-систем.
Ключевые слова: онтология, Wiki-система, семантические технологии, построение информационных систем.

1 Введение

Для удовлетворения все возрастающих информационных потребностей пользователей разработаны разнообразные средства построения информационных систем. Одним из таких удобных и простых в использовании средств сбора и хранения информации являются Wiki-технологии. Эти технологии позволяют работать не только с текстовым, но и с мультимедийным контентом, имеют удобный и интуитивно понятный интерфейс, просты в освоении. Однако большим недостатком этих технологий является то, что они позволяют отслеживать в создаваемых информационных системах только структурную целостность ссылок, не обеспечивая при этом логической целостности и семантической согласованности используемых в них понятий (категорий). Идея предлагаемого подхода состоит в создании инструментария, который бы обеспечивал построение Wiki-систем с согласованной системой понятий (семантически согласованных Wiki-систем). Wiki-системы с такими свойствами можно получить, если строить ее на основе логически непротиворечивой онтологии, описывающей предметную область будущей системы.

В докладе рассматривается подход к построению информационных Wiki-систем на основе онтологий предметных областей. Чтобы этот подход можно было использовать для реинжиниринга, сопровождения и развития уже существующих Wiki-систем, он дополняется обратной процедурой, т.е. методом извлечения онтологий из Wiki-систем.

2 Обзор существующих подходов

2.1 Обзор средств создания информационных Wiki-систем с использованием семантических технологий

На сегодняшний день существует достаточно много проектов создания информационных Wiki-систем, так или иначе связанных с семантическими технологиями. Хотя эти проекты нацелены на решение различных задач, общим для них является использование онтологий на тех или иных этапах построения или функционирования информационных систем. Так как на данный момент существует большое количество средств создания Wiki-систем, то в этом обзоре мы сосредоточим наше внимание только на тех из них, которые в той или иной мере используют семантические технологии.

Сначала рассмотрим проект из Санкт-Петербурга [1], в котором в одном из первых начали применяться онтологии при построении таких информационных систем, как корпоративные Wiki-порталы. Исходя из того, что онтологии позволяют создавать более концептуально-ясные и целостные модели предметных областей, разработчиками было решено привязать контент Wiki-системы к онтологии, чтобы избежать проблем с плохой структурой портала, т.к. она в этом случае будет такого же качества, как и определяющая ее онтология. В рамках это проекта был разработан прототип инструментария OntolingeWiki, который содержит внутри себя Wiki-

движок и для каждого концепта онтологии автоматически создает Wiki-страницу, в которую эксперт вносит информацию об описываемом понятии. При этом конечные пользователи видят онтологию, представленную в виде графа, могут осуществлять навигацию по ее понятиям и видеть содержимое Wiki-страниц, соответствующих выбранному понятию.

Недостатком OntolingeWiki является то, что онтология, лежащая в основе созданной с помощью нее Wiki-системы, не может быть изменена или дополнена средствами самой Wiki-системы.

Наиболее распространенным инструментальным средством для создания Wiki-систем является популярный Wiki-движок MediaWiki [2]. Именно для этого универсального движка больше всего строится расширений, позволяющих использовать средства семантических технологий, Одной из первых была создана надстройка над MediaWiki, получившая название Semantic MediaWiki [3]. Это расширение допускает возможность импорта онтологии [4] и позволяет использовать в Wiki-системе структуры и связи, определенные в ней. Однако предоставляемый Semantic MediaWiki модуль импорта онтологий на данный момент находится в стадии бета-версии и не включен в состав новых версий данного расширения. В связи с этим реализацию импорта онтологий пользователи Semantic MediaWiki должны выполнять самостоятельно.

Другим расширением MediaWiki является BOWiki [5] — основанный на онтологии семантический Wiki-движок. Первоначально он базировался на Semantic MediaWiki, а потом был реализован как «чистое» расширение MediaWiki. Достоинством BOWiki является то, что он позволяет пользователям рассматривать сущности, описанные Wiki-страницами, в качестве экземпляров онтологических категорий, определять новые отношения в Wiki-контенте, связывать Wiki-страницы, в том числе n-арными семантическими отношениями, запрашивать Wiki-страницы, удовлетворяющие определенным условиям, допускает импорт еще нескольких био-онтологий для ограниченного использования и обеспечивает экспорт Wiki-контента в OWL-представление.

BOWiki разрабатывался как средство для совместного создания и интеграции знаний в области биоинформатики, поэтому его функционирование основано на онтологии высокого уровня General Formal Ontology (GFO) [6] в OWL-версии. В своей работе он использует также две онтологии, основанные на GFO, — базовую биомедицинскую онтологию (GFO-Bio) и онтологию функций (Ontology of Functions или OF). Как было сказано выше, возможность использования других (внешних) онтологий существенно ограничена, поэтому BOWiki не может рассматриваться в качестве инструмента для построения Wiki-систем произвольного вида.

Также для MediaWiki существуют расширения, работающие совместно с Semantic MediaWiki, и добавляющие дополнительную функциональность по импорту семантической информации. Одним из таких является RDFIO [7]. Оно позволяет импортировать произвольные RDF-триплеты [8]. Но это расширение пока имеет статус бета-версии и еще не готово для массового использования. Еще одним подобным расширением является LinkedWiki [9]. Оно применяется для получения данных с внешнего сервиса и отображения этих данных в Wiki-системе. Это расширение не может изменить структуру существующей системы, а только дополняет ее, поэтому имеет довольно ограниченную область применения.

2.2 Обзор подходов к извлечению онтологий из Wiki-систем

Существует много подходов к извлечению знаний из Wiki-систем в виде онтологии, в частности, из Википедии. Одними из первых решением этого вопроса занялась группа немецких ученых из Института информатики общества Макса Планка [10]. Их целью было автоматическое построение общей онтологии значительного объема, включающую факты, извлеченные из Википедии с высокой точностью. При этом предполагалось совместно использовать знания из Википедии и WordNet [11]. Эта задача была успешно решена, В результате была получена онтологическая база знаний YAGO. В данном подходе источниками информации в Википедии служат не сами статьи целиком, а система категорий этой энциклопедии и так называемые «шаблоны-карточки» («infoboxes) — стандартизованные таблицы, содержащие основную информацию о предмете, описываемом в статье. При этом из Википедии извлекаются индивиды, большая часть отношений и категории-листья, а остальная иерархия категорий и отношений достраивается благодаря WordNet [12]. В основе YAGO лежит собственная модель, которая является небольшим расширением RDFS. По заявлению авторов, несмотря на достаточную выразительность модели, она остается разрешимой (в отношении проверки ее непротиворечивости). В проведенном ими оценочном исследовании точность информации, содержащейся в онтологии, составила 95% — намного выше, чем у всех других автоматически построенных общих онтологий. Кроме того, данная онтология имеет гораздо больший по сравнению с упомянутыми выше онтологиями размер. Она используется во многих приложениях и является составной частью многих других семантических проектов. Дальнейшее развитие этой базы знаний — YAGO2 — использует еще один источник данных (GeoNames) и содержит еще более внушительный объем информации [13].

На основе Википедии строят не только общие онтологии, но и специализированные. В частности, группа японских ученых разработала метод для построения крупномасштабной онтологии людей [14]. В нем иерархия категорий, а также сами экземпляры, извлекаются из энциклопедии с помощью машинно-обучаемого классификатора и с использованием японского тезауруса Nihongo Goi-Taikei. Существенной частью метода является использование «похожих категорий» — это категории «родители», «дети» или «сестры», у которых последнее слово совпадает с последним словом целевой категории. Эксперименты, проведенные авторами этой работы, показывают высокую точность и полноту полученной онтологии (они превосходят предшествующие сущест-

вующие методы), а также существенность использования тезауруса и «похожих категорий». Еще одним преимуществом перед другими методами названа возможность извлечения категорий, не имеющих совпадений в тезаурусе, и категорий с неоднозначными названиями.

Еще одним подходом к построению предметно-ориентированных онтологий занимается совместный проект университетов Инсбрука и Флориды [15]. Его участники утверждают, что основной недостаток существующих онтологий в том, что они разрабатываются только небольшими группами людей, а большинство потенциальных пользователей исключено из процесса разработки онтологий и их мнения не учитываются. Поэтому предлагается использовать стандартные Wiki-технологии, с помощью которых тысячи пользователей уже вовлечены в создание такого огромного по объему источника знаний как электронная энциклопедия Википедия, в качестве среды для разработки онтологий. В работе доказывается, что URI этой энциклопедии могут служить надежными идентификаторами для онтологических концептов, и демонстрируется применимость этого подхода на практике.

Помимо непосредственного извлечения знаний в виде онтологий, на основе Википедии решают и вспомогательные задачи в этой области. Например, при конструировании онтологии обычно требуется корпус из конкретной предметной области для построения соответствующей иерархии концептов. При этом он должен обеспечивать хорошее покрытие этой области с достаточной степенью качества. Этим вопросом как раз и занимаются ученые из Гонконгского политехнического университета [16]. Они используют в качестве источника Википедию и предлагают новый подход для классификации статей по предметным областям. Основная идея заключается в генерации иерархии предметной области на основе гипертекстовых связей страниц. И только статьи, тесно связанные с этой иерархией, выбираются в качестве кандидатов. Предлагаемый метод пока использует только информацию о категориях Википедии. Далее выполняется ранжирование и фильтрация выбранных страниц. Результаты эксперимента по оценке, проведенного авторами, показывают, что Википедия является хорошим ресурсом для получения корпуса конкретной предметной области сравнительно высокого качества.

Конечно же, существуют проекты, в которых данные, извлеченные из Википедии, используются не просто для построения онтологии или для решения отдельно взятой вспомогательной задачи, а применяются как часть крупной информационной системы, тесно взаимодействующей с другими системами (примерами взаимодействия являются автоматический сбор данных из других систем, объедение с другими онтологиями, отображение содержимого в другую систему и т.п.). Для иллюстрации рассмотрим проект под названием «Построение и использование геопространственной онтологии в наблюдающей системе BioCaster», разрабатываемый группой японских и вьетнамских ученых [17]. Для начала они планируют построить геопространственную онтологию, используя информацию из Википедии, а затем использовать ее в своей системе. База данных будет содержать названия стран и самых крупных городов, а также отношения типа «часть-целое» между странами и их частями. Процесс построения проходит в полуавтоматическом режиме: автоматическое извлечение данных, а затем верификация с участием человека. На последнем этапе геопространственная иерархия из Википедии объединяется с онтологией BioCaster. Эта система предназначается для выявления и отслеживания вспышек инфекционных заболеваний на основе новостных сообщений. В дальнейшем эта полученная информация визуализируется с использованием Google Maps.

Еще одним проектом, в котором построение онтологии на основе Википедии используется в качестве составной части, занимаются в Наньянском технологическом университете [18]. В его рамках была предпринята попытка автоматического построения крупномасштабной мульти-модальной онтологии для классификации веб-изображений. Для текстовой части были использованы преимущества структурных и контентных возможностей Википедии, и объекты реального мира были формализованы в терминах концептов и отношений. Для визуальной части производилось обучение классификатора и по результатам генерировались концепты промежуточного уровня. В дальнейшем использовалось ассоциативное правило поискового алгоритма для улучшения построенной онтологии. Посредством эксперимента была доказана высокая точность работы метода.

Википедия — не единственный источник информации, используемый для извлечения знаний. В проекте разработки портала знаний о классификации компьютерных языков [19] используется подход, основанный на извлечении знаний из «шаблонов-карточек», а в качестве источников информации используются Progopedia [20] и Freebase [21].

3 Построение информационной Wiki-системы на основе онтологии

Предлагаемая в рамках данной работы инструментальная система позволяет на основе ранее разработанной онтологии построить Wiki-систему, структура и содержание которой будет определяться этой онтологий. В дальнейшем построенная Wiki-система может расширяться как обычная Wiki-система с использованием традиционных средств Wiki-технологии.

Общая схема работы инструментальной системы представлена на рис. 1. Согласно ней разработка Wiki-системы в рассматриваемой инструментальной системе включает следующие этапы.

На первом шаге онтология, полученная в готовом виде или разработанная в Protégé [22] или каком-либо другом редакторе онтологий, сохраняется или конвертируется в OWL-формат [23]. После этого файл со специ-

фикацией онтологии подается на вход специально разработанному в рамках данного проекта программному модулю Onto2Wiki, который разбирает его с использованием библиотеки RDFLib [24], а затем, при помощи среды Python WikipediaBot Framework [25], создает каркас информационной системы на основе пустого Wiki-сайта, работающего на базе MediaWiki с расширением Semantic MediaWiki. При этом в Wiki-систему добавляются нужные страницы, для которых указываются соответствующие атрибуты, расставляются категории и прописываются нужные связи. После выполнения этих действий Wiki-система готова к использованию.

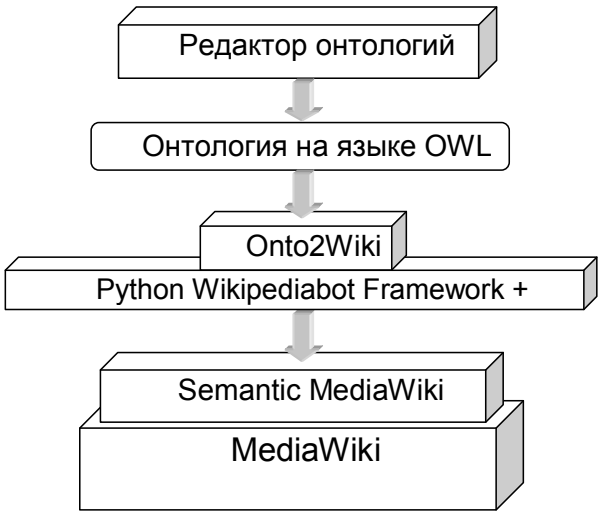


Рис. 1. Общая схема работы инструментальной системы

В таблице 1 представлено соответствие между конструкциями языка OWL и семантической Wiki, которое используется при отображении онтологии в структуры и содержание Wiki-системы. В соответствии с этой таблицей для каждого класса онтологии будет создана своя категория, которая будет проставлена на каждой подкатегории, соответствующей подклассу данного класса, и на каждой странице, соответствующей экземпляру данного класса. Также все атрибуты экземпляра онтологии будут указаны в качестве атрибутов на странице, соответствующей этому экземпляру. При этом все отношения онтологии отобразятся в типизированные ссылки [26] между соответствующими страницами и категориями.

Таблица 1. Соответствие конструкций языка OWL и Semantic MediaWiki

Конструкция OWL	Semantic MediaWiki
Класс	Категория
Подкласс	Подкатегория
Экземпляр	Страница
Атрибут	Атрибут
Отношение	Типизированная ссылка

Созданную с помощью данного инструмента систему можно дорабатывать в соответствии с возникающими потребностями, взаимодействуя с ней как с обычной Wiki-системой. Т.е. можно создавать новые страницы, прописывать новые связи, указывать новые категории и т.д.

4 Извлечение онтологий из существующих Wiki-систем

Как было сказано выше, подход к построению информационных Wiki-систем на основе онтологий предметных областей кроме механизма «отображения» онтологий должен включать и обратную процедуру, т.е. метод извлечения онтологий из Wiki-систем. Причем этот метод может иметь гораздо больше применений, чем это может показаться на первый взгляд.

4.1 Метод извлечения онтологий

Задача извлечения онтологии из Wiki-системы решается с помощью тех же инструментов, которые применялись при ее «отображении». Разница заключается в использовании вместо модуля Onto2Wiki другого программного модуля — Wiki2Onto, также разработанного в рамках данного проекта. При этом все операции выполняются в обратном порядке. Т.е. сначала модуль Wiki2Onto при помощи Python WikipediaBot Framework извлекает онтологию из Wiki-системы, а затем с использованием библиотеки RDFLib сохраняет в файл на языке OWL. При этом используется та же самая таблица соответствия конструкций языка OWL и семантической Wiki (см. таблицу 1).

Рассмотрим этот процесс более подробно. Онтология из Wiki-системы извлекается в следующем порядке. Сначала извлекаются все классы, при этом каждому классу соответствует одна категория Wiki-системы, а структура вложенности категорий Wiki-системы определяет иерархию классов. Затем извлекаются все страницы как экземпляры соответствующих классов. Для пустых страниц, на которые в Wiki-системе имеются ссылки, заводится специальный служебный класс «Несуществующие страницы». После этого просматриваются все ссылки на каждой странице. Для начала определяется, является ли ссылка обычной или семантической. Если ссылка обычная, то для соответствующего экземпляра класса в OWL-онтологии заводится объектное свойство «Ссылается на» со значением в виде экземпляра, имя которого совпадает с именем страницы, на которую указывает ссылка. Если ссылка семантическая, то она имеет следующую структуру *<название свойства*, *значение свойства>*, и для нее сначала определяется тип ее свойства. Если свойство имеет тип *«Страница»* или его тип не указан, то в OWL-онтологии заводится объектное свойство с соответствующими именем (*название свойства*) и значением (*значение свойства*). (Заметим, что по умолчанию свойство ссылки имеет тип *«Страница»*.) Если же свойство имеет какой-то другой стандартный тип, то тип свойства данных в OWL-онтологии определяется согласно таблице 2. Для пользовательских типов свойств создается собственный тип свойства данных.

Таблица 2. Соответствие при извлечении стандартных типов

Тип свойства	Тип OWL
Строка	string
Число	double
Булево	boolean
Дата	dateTime
Текст	string
Код	string
Телефонный номер	Annotation property
URL	Annotation property
Почта	Annotation property
URI аннотации	Annotation property

Так как реализация модуля Wiki2Onto еще не доведена до финальной стадии, то пока поддерживаются не все стандартные типы свойств Semantic MediaWiki, а только те, что представлены в таблице 2.

Следует отметить, что Wiki-система, из которой извлекается онтология, не обязательно должна функционировать на расширении Semantic MediaWiki. Однако в случае использования Wiki-системы без этого расширения извлекаемая онтология будет гораздо беднее, так как в ней не будет присутствовать специальная семантическая информация. В частности, нельзя будет извлечь атрибуты, разнообразие отношений также будет невелико. Правда, путем индивидуальной настройки на данную Wiki-систему объем извлекаемой из нее информации можно увеличить.

4.2 Варианты применения метода извлечения онтологий

Самое очевидное из применений это, конечно же, посмотреть на то, как изменилась онтология после того, эксперты поработали над содержимым Wiki-сайта, созданного на базе первоначальной онтологии. Это может понадобиться не только ради обычного любопытства, но и для вполне серьезных целей: например, отслеживание развития проекта, верификация получаемой онтологии, проверка качества и сбалансированности получаемой структуры данных, координация в развитии отдельных частей предметной области. Также к полученной онтологии можно применить машину вывода для получения неявных знаний.

Кроме того, извлекать онтологии можно не только из тех Wiki-систем, куда она была предварительно «отображена», но из любых других. Это можно применять в тех случаях, когда у нас уже есть система, где содержатся некоторые данные по нужной нам предметной области, а мы хотим построить онтологии этой области. Вместо того, чтобы вручную строить всю онтологию «с нуля», можно извлечь ее из системы, получив пред-

варительный, черновой вариант, а потом уже дорабатывать его, что уже гораздо проще. Это особенно актуально для построения тезаурусов.

Еще одно применение заключается в объединении нескольких Wiki-систем по схожим предметным областям. Непосредственное объединение может быть довольно сложным и потребовать очень много ручной работы, потому что крайне затруднительно отследить все связи и пересечения между двумя системами. Используя предлагаемый подход, можно поступить гораздо проще: извлечь онтологию из каждой системы, а потом провести их слияние. После этого по полученной онтологии строить общую Wiki-систему.

5 Заключение

В данной работе рассмотрен подход к построению информационных систем на основе Wiki-технологии и онтологий предметных областей. В рамках этого подхода предложены метод построения Wiki-систем на основе онтологий и метод извлечения онтологий из существующих Wiki-систем, разработан прототип инструментальной системы, реализующий данные методы в объеме, указанном в разделах 3 и 4.1.

Разработка метода построения Wiki-систем на основе онтологий позволяет строить Wiki-системы с хорошей структурой и согласованной системой понятий.

Наличие метода извлечения онтологий из Wiki-систем открывает следующие возможности в построении информационных систем и онтологий:

1) **контроль качества построенной на основе онтологии Wiki-системы в течение всего ее жизненного цикла:** Wiki-система, построенная на основе онтологии в рамках нашего подхода, дополняется новыми категориями и страницами; периодически из этой системы извлекается онтология, верифицируется, исправляется; после этого по обновленной онтологии заново строится Wiki-система, обладающая хорошим качеством;
2) **возможность реинжиниринга информационных Wiki-систем**: сначала извлекается онтология Wiki-системы; затем эта онтология верифицируется, (при необходимости) дополняется необходимыми элементами; после этого по новой онтологии строится непротиворечивая система с новыми свойствами;
3) **возможность построения обобщенной информационной Wiki-системы на основе нескольких близких по тематике Wiki-систем**: из каждой Wiki-системы извлекается онтология, выполняется слияние полученных онтологий, на основе объединенной онтологии строится обобщенная Wiki-система;
4) **возможность построения полных онтологий предметных областей**: извлекаются онтологии нескольких Wiki-систем, относящихся к одной тематике; после верификации полученных онтологий, они объединяются в одну (используя методы выравнивания и слияния онтологий).

Список литературы

[1] Муромцев Д.И., Горовой В.А., Малинин А.А., Гаврилова Т.А., Злобин А.Н., Катков Ю.В. Интеграция wiki-технологии и онтологического моделирования в задаче управления знаниями предприятия // Труды 11-ой национальной конференции по искусственному интеллекту с международным участием КИИ-2008 (г. Дубна, Россия). — М.: ЛЕНАНД, 2008. — Т.3. — С. 360-368.
[2] MediaWiki http://mediawiki.org.
[3] Semantic MediaWiki http://semantic-mediawiki.org.
[4] Ontology import http://semantic-mediawiki.org/wiki/Help:Ontology_import.
[5] Backhaus M., Kelso J., Bacher J., Herre H., Hoehndorf R., Loebe F., Visagie J. BOWiki – a collaborative annotation and ontology curation framework // *In Proceedings of the Workshop on Social and Collaborative Construction of Structured Knowledge*, CKC Banff, Canada, May 8, of CEUR Workshop Proceedings. Volume 273. Edited by Noy N., Alani H., Stumme G., Mika P., Sure Y., Vrandecic D. Aachen, Germany: CEUR-WS.org, 2007.
[6] Herre H., Heller B., Burek P., Hoehndorf R., Loebe F., Michalek H. General Formal Ontology (GFO) – A foundational ontology integrating objects and processes [Version 1.0]. Onto-Med Report 8, Research Group Ontologies in Medicine, Institute of Medical Informatics, Statistics and Epidemiology, University of Leipzig, Leipzig (2006)
[7] RDFIO http://www.mediawiki.org/wiki/Extension:RDFIO.
[8] RDF http://www.w3.org/RDF/.
[9] LinkedWiki http://www.mediawiki.org/wiki/Extension:LinkedWiki.
[10] Suchanek F.M., Kasneci G., Weikum G. YAGO: A Core of Semantic Knowledge Unifying WordNet and Wikipedia // *In Proceedings of the 16th International Conference on World Wide Web* (Banff, Alberta, Canada, May 8–12, 2007). WWW '07. ACM Press, New York, NY, pp. 697– 706, 2007.
[11] Fellbaum C., editor. WordNet: An Electronic Lexical Database. MIT Press, 1998.
[12] Suchanek F.M., Kasneci G., Weikum G. YAGO: A Large Ontology from Wikipedia and WordNet // Web Semantics: Science, Services and Agents on the World Wide Web, Volume 6, Issue 3, September 2008, pp. 203–217.

[13] Hoffart J., Suchanek F.M., Berberich K., Weikum G. YAGO2: A Spatially and Temporally Enhanced Knowledge Base from Wikipedia // Research Report MPI-I-2010-5-007, Max-Planck-Institut für Informatik, November 2010.
[14] Shibaki Y., Nagata M., Yamamoto K. Constructing Large-Scale Person Ontology from Wikipedia // *In Proceedings of the 2nd Workshop on "Collaboratively Constructed Semantic Resources"*, Coling, 2010, pp. 1–9.
[15] Hepp M., Bachlechner D., Siorpaes K. Harvesting Wiki Consensus – Using Wikipedia Entries as Ontology Elements // *In Proceedings of the First Workshop on Semantic Wikis – From Wiki to Semantics, co-located with the 3rd Annual European Semantic Web Conference* (ESWC 2006), 2006, pp. 124–138.
[16] Cui G.Y., Lu Q., Li W.J. and Chen Y.R. Corpus Exploitation from Wikipedia for Ontology Construction // *In Proceedings of the Sixth International Language Resources and Evaluation* (LREC 2008), Marrakech, 2008, pp. 2125–2132.
[17] Doan, Son, Ngo, Quoc-Hung, Kawazoe, Ai, and Collier, Nigel, "Building and Using Geospatial Ontology in the BioCaster Surveillance System", Available from Nature Precedings, <http://dx.doi.org/10.1038/npre.2008.2110.1> (2008).
[18] Huan Wang, Xing Jiang, Liang-Tien Chia, and Ah-Hwee Tan Wikipedia2Onto --- Adding Wikipedia Semantics to Web Image Retrieval // *In Proceedings of the WebSci'09: Society On-Line*, 18-20 March 2009, Athens, Greece.
[19] Shilov N.V., Akinin A.A., Zubkov A.V., Idrisov R.I. Development of the Computer Language Classification Knowledge Portal // *In Proceedings of the 8th Int. Ershov Informatics Conference* (Novosibirsk, Russia, June 27 – July 01, 2011). Novosibirsk: A.P. Ershov Institute of Informatics Systems, 2011, pp. 255–260.
[20] Progopedia http://progopedia.ru.
[21] Freebase http://freebase.com.
[22] Protégé http://protege.stanford.edu.
[23] OWL http://w3.org/2004/OWL/.
[24] RDFLib http://rdflib.net.
[25] Python WikipediaBot Framework http://pywikipediabot.sourceforge.net.
[26] Völkel M., Krötzsch M., Vrandecic D., Haller H. and Studer R. Semantic Wikipedia // *In Proceedings of the 15th International Conference on World Wide Web* (Edinburgh, Scotland, May 23–26, 2006). WWW '06. ACM Press, New York, NY, pp. 585–594, 2006, <http://doi.acm.org/10.1145/1135777.1135863>.

Моделирование социальных процессов на основе мультиагентного подхода

Н.С. Копылова, Ф.А. Мурзин, И.А. Курков

Институт систем информатики им. А.П. Ершова СО РАН
630090, Новосибирск, проспект Лаврентьева, 6, Россия
[1]sibirierkatze@gmail.com, [2]murzin@iis.nsk.su, [3]ikurkov@gmail.com

Введение

В условиях высоких темпов информационного развития современного общества и нарастания неопределенности социальных изменений возникает потребность в методах диагностики и анализа самой социальной среды, социальных взаимодействий, прогнозирования будущего и оценки рисков. Однако далеко не всегда существует возможность разработать строгие математические модели социальной системы или сформировать запрос на данные исследования специалистам. Одним из решений данной проблемы является разработка комплекса гибких моделей социальных процессов, позволяющих выделить основные направления и характеристики социальной реальности, упрощая тем самым сложность и формируя целостный образ, необходимый для поставленных задач: детального анализа, адекватной оценки ситуации, принятия управленческих решений, постановки задач на более строгие, количественные исследования.

Социальные процессы труднее оценивать и моделировать, так как результаты скорее психологические и социологические, чем физические. Хотя средства моделирования социальных систем очень медленно разрабатываются, все же существует несколько групп, добившихся некоторого успеха в разработке агентных моделей социальных сетей. Компьютерное моделирование с использованием агентов представляется наиболее эффективным для моделирования сложных социальных процессов.

В основе такого подхода лежит понятие интеллектуального агента. Агент – это сущность, которая находится в некоторой среде, получает данные, которые отражают события, происходящие в среде, интерпретирует их и исполняет команды, воздействующие на среду. Мультиагентные системы характеризуются тем, что, во-первых, каждый агент владеет информацией, недостаточной для решения всей задачи, во-вторых, нет глобального контроля, то есть каждый агент принимает решения самостоятельно, в-третьих, данные децентрализованы, и наконец, вычисления могут быть асинхронны.

Основной целью работы является описание социологической модели, построение математической модели, для программной реализации этой модели - моделировании в системе RePast.

1. Идея модели

В модели присутствуют агенты нескольких типов (работники, чиновники, собственники). Кроме того, возможно деление на различные типы внутри одной группы. Например, рабочие, добывающие ресурс, и рабочие, перерабатывающие ресурс. Агенты могут переходить из одной группы в другую, при выполнении некоторых заранее определенных условий. Время жизни агента не ограничено, считается, что агент представляет не одного человека, а некоторое множество (группу) людей. При благоприятном состоянии окружающей среды возможно появление новых агентов внутри группы. Можно рассмотреть модели, где появление агентов внутри группы возможно для всех групп, и где это возможно только для некоторых групп. Например, все новые агенты являются рабочими, затем агент из группы рабочих может переходить в другие группы. [1]

Каждый агент описывается набором свойств и атрибутов. Свойства – это, например, географическое положение, возможность выполнять некоторые действия. Атрибут – некоторая измеряемая и изменяемая величина (шкала). Уровень того или иного атрибута влияет на поведение агента. Базовый набор атрибутов одинаков для всех типов агентов. В зависимости от типа агента атрибут может находиться в активном или пассивном состоянии (когда его уровень не влияет на поведение агента). Например, можно рассмотреть атрибут авторитетность, для агента-чиновника это активный атрибут, определяющий его радиус влияния, для агента-работника это пассивный атрибут. Если работник станет чиновником, то атрибут перейдет в активное состояние.

Агенты взаимодействуют друг с другом и оказывают друг на друга влияние. Способы влияния зависят от типа взаимодействующих агентов. Есть несколько базовых типов взаимодействия. Во-первых, через изменения атрибутов, так, агенты-чиновники собирают налоги с агентов-работников, тем самым уменьшая уровень атрибута накопления работника. Во-вторых, через указания к действиям. Например, агент-собственник может отправить подчиненных ему работников на новое место работы или дать им новое задание (перекинуть часть работников с добычи ресурса на его переработку, например).

Цель агента – улучшение его показателей-атрибутов.

2. Среда расположения агентов

Агенты находятся в некотором пространстве. В этой среде имеется ресурс, добываемый и перерабатываемый агентами (рабочими). Ресурсы могут быть различных типов. Некоторые ресурсы являются возобновляемыми, другие нет. Количество доступного ресурса в точке определяется насыщенностью (*RESOURCE DEPTH*). От насыщенности зависит и величина возможной добычи. Кроме того, для каждой точки определяется сложность добычи ресурса (*EXTRACTION DIFFICULTY*). Чем выше уровень трудности, тем больше усилий затрачивается на добычу ресурса.

3. Атрибуты агентов

Каждый агент имеет следующий набор атрибутов: *ENERGY*, *OBEDIENCE*, *AUTHORITY*, *FUNDS*, *STOCK*, *PRODUCTS*.

Энергия (*ENERGY*) – это жизненные силы агента. Уровень энергии определяет здоровье агента, влияет на производительность агента.

$$ENERGY(n+1) = ENERGY(n) + ENERGY_+(n) - ENERGY_-(n)$$

Каждый шаг некоторое количество энергии тратится на поддержание жизни агента, а также для выполнения различных действий. Количество затрачиваемой энергии зависит от типа агента и от выполняемых им действий. Нет действий, которые производились бы без затрат энергии. Если уровень энергии агента ниже, чем количество, необходимое на выполнение действия, то агент не может выполнить это действие. Если уровень энергии ниже, чем количество, необходимое для поддержания жизни, то агент умирает.

$$ENERGY_-(n) = ENERGY_{life}(n) + \sum_{actions} ENERGY_{act}(n)$$

$$ENERGY_{life}(n) = E_{life}$$

$$ENERGY_{act} \geq 0$$

Энергия пополняется за счет накоплений (т.е. на деньги агенты «покупает продукты питания», «съедает» их, восстанавливая жизненную энергию). Но за один шаг энергия не может пополниться более чем некоторую определенную величину.

$$ENERGY_+(n) = ENERGY_{feed}(n)$$

$$0 \leq ENERGY_{feed}(n) < E_{feed}$$

Для атрибута энергия есть несколько граничных точек. Величина границ зависят от типа агента. E_{life} – минимальное количество энергии, необходимое на поддержание жизни агента. E_{max} – максимально возможное количество энергии.

Послушание (*OBEDIENCE*) определяет насколько легко агент подчиняется указаниям других агентов. То есть, введена степень случайности в то, будет ли выполнено указание. И вероятность выполнения прямо пропорциональна *OBEDIENCE*. И чем выше послушание, тем легче агент выполняет требования других агентов. Можно рассматривать уровень послушания по отношению к разным группам. Уровень послушания снижается, если агенту «плохо» живётся. Правило понижения уровня послушания зависят от типа агента. Уровень послушания увеличивается, если агенты «хорошо» живётся. Правило повышения уровня послушания зависят от типа агента. Если уровень послушания падает до некоторой критической величины O_{disob}, то агент выходит из повиновения.

Авторитетность (*AUTHORITY*) определяет возможность агента управлять другими агентами. Для агента-чиновника – чем выше авторитет, тем больше территория, которой чиновник может управлять. Для агента-собственника – чем больше авторитет, тем больше рабочих он может контролировать. Если агент «плохо» справляется со своей задачей, то уровень его авторитетности падает. Правило понижения уровня авторитетности зависят от типа агента. Если агент «хорошо» справляется со своей задачей, то уровень его авторитетности возрастает. Правило повышения уровня авторитетности зависят от типа агента.

Накопления (*FUNDS*) – это денежные накопления агента. Накопления агента могут быть его личными накоплениями (агенты-рабочие, агенты-собственники) или общими накоплениями группы (агенты-чиновники). Агент тратит средства на поддержание своей жизни (на увеличение энергии). Кроме того, разные типы агентов могут тратить накопления на различные цели, характерные для этих типов, например налоговые выплаты или выплата заработной платы.

$$FUNDS_-(n) \Leftarrow FUNDS_{feed}(n)$$

Увеличение накоплений происходит за счёт разных источников, которые зависят от типа агента, например,

увеличение накоплений агентов-чиновников происходит за счёт поступаемых налогов, увеличение накоплений агентов-собственников – за счёт продажи производимой продукции. Уровень накоплений не ограничен.

Запас ресурсов ($STOCK$) – количество ресурсов, имеющихся у агента. Способы пополнения запаса зависят от типа агента. Например, агент-рабочий может пополнить запас, добывая ресурс из среды, агент-собственник – получая ресурсы от своих рабочих или покупая их. Уменьшение запаса происходит, когда агент продаёт или отдаёт другим агентам имеющиеся у него ресурсы. Максимальное количество ресурсов, которое может иметь агент, зависит от типа агента. Например, агент-собственник может хранить при себе количество в несколько раз больше, чем агент-рабочий.

$$STOCK(n) \leq S_{max}$$

Запас продуктов ($PRODUCTS$) – количество продуктов, имеющихся у агента. Способы пополнения запаса зависят от типа агента. Например, агент-рабочий может пополнить запас, перерабатывая сырьё в продукт, агент-собственник – получая продукты от своих рабочих. Уменьшение запаса происходит, когда агент продаёт или отдаёт другим агентам имеющиеся у него продукты. Максимальное количество продуктов, которое может иметь агент, зависит от типа агента. Например, агент-собственник может хранить при себе количество в несколько раз больше, чем агент-рабочий.

$$PRODUCTS(n) \leq P_{max}$$

4. Свойства агентов

Положение агента в пространстве описывается свойством $COORDINATES$. Агенты могут перемещаться из одной точки пространства в другую либо по собственному желанию, либо по указанию других агентов. При перемещении агент тратит энергию, пропорциональную перемещению.

$$ENERGY_{path} = PATH \times E_{path}.$$

В случае перемещения по собственному желанию агент проходит за один временной шаг расстояние (считаем, что из энергии уже вычли энергию на поддержание жизни)

$$PATH = integer(ENERGY \times PATH_{norm}).$$

В случае перемещения по указанию агент проходит за один временной шаг расстояние

$$PATH = integer(ENERGY \times OBEDIENCE \times PATH_{norm}).$$

При выполнении соотношения

$$PATH_{norm} \times E_{path} \leq 1,$$

затраченная энергия всегда меньше энергии агента (с учетом того, что некоторую ее часть уже потратили на поддержание жизни агента).

$CREW\ SIZE$ (численность) – свойство агента, определяет количество людей, представляемых одним агентом.

Также агент может анализировать окружающую среду. Свойство $VIEW$ описывает расстояние, на которое видит агент. При этом агент тратит некоторое количество энергии E_{view} (одинаковое для агентов одного типа).

$CONTROL\ AREA$ (подконтрольная территория) – свойство агента, определяет территория находящаяся под контролем агента. Для агента-рабочего является неактивным.

5. Агенты-рабочие

Агенты-рабочие могут быть свободными (работают на себя) и несвободными (работают на собственника). Агенты могут добывать или перерабатывать ресурс. Причем агент не может делать оба действия в один шаг.

Агенты-рабочие могут добывать ресурс, находящийся в среде.

$$RESOURCE \rightarrow STOCK$$

Количество добываемого ресурса зависит от уровня энергии агента, от количества ресурса в точке, где располагается агент, от доступности ресурса и от его уровня послушания агента. Добывая ресурс, агент тратит энергию. Количество затраченной энергии зависит от трудности добычи и от количества добытого ресурса ($STOCK_{ext}$), для добычи единицы ресурса при минимальной сложности добычи тратится E_{piece} энергии. При этом зафиксирована норма (S_{norm}), т.е. количество ресурса, которое добывает агент с фиксированным набором атрибутов ($\{ENERGY, OBEDIENCE\} = (1,1)$).

Предлагается использовать линейную зависимость от энергии и послушания, обратно пропорциональную от трудности добычи. Потраченная энергия пропорциональна количеству ресурса и сложности добычи. Будем считать, что от $ENERGY$ уже отняли количество E_{life}, необходимое для поддержания жизни агента.

$$STOCK_{ext} = \min\left\{ENERGY \times OBEDIENCE \times f_s(EXTRACTION\ DIFFICULTY) \times S_{norm}, \frac{RESOURCE\ DEPTH}{S_{max}}\right\}$$

$$ENERGY_{ext} = STOCK_{ext} \times f_e(EXTRACTION\ DIFFICULTY) \times E_{piece}$$

Отметим, что при выполнении соотношения

$$S_{norm} \times E_{piece} \le \left(\max_{EXT\ DIF} f_s(EXT\ DIF) \times f_e(EXT\ DIF)\right)^{-1},$$

затраченная энергия всегда меньше энергии агента (с учетом того, что некоторую ее часть уже потратили на поддержание жизни агента).

Функции f_s и f_e могут иметь вид, указанный на рисунке 1. Кроме того, исходя из смысла величин S_{norm} и E_{unit}, если сложность добычи минимальна, то значения этих функций должны быть равны единице: $f_e(easy) = 1$, $f_s(easy) = 1$

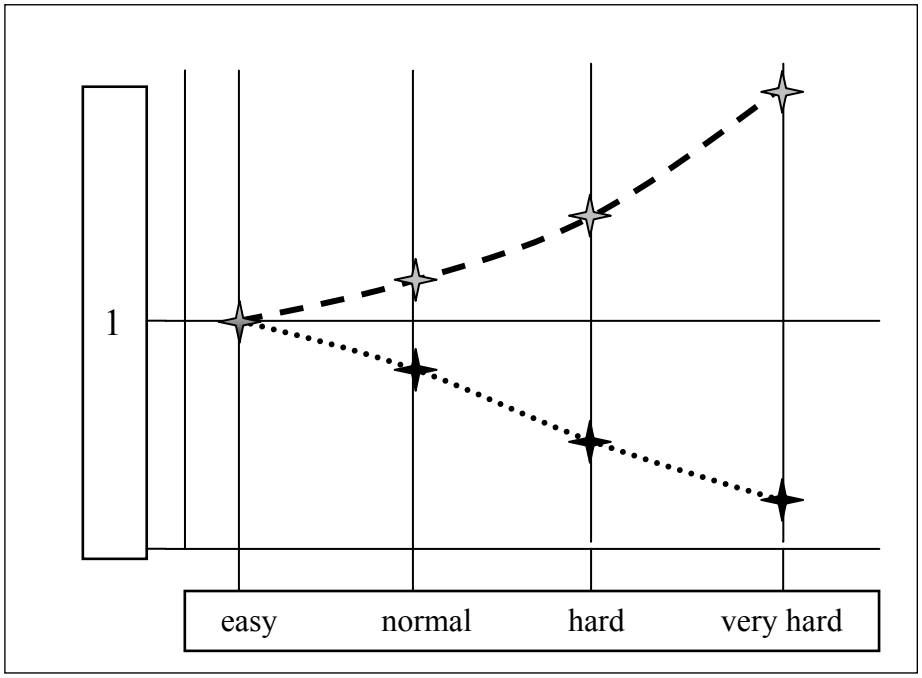


Рис. 1. Функции f_e (пунктир) и f_s (точки)

Агенты-рабочие могут перерабатывать ресурсы в продукты.

$STOCK \rightarrow PRODUCTS$

Количество ресурса, которое агент может переработать зависит от энергии агента и от его уровня послушания агента. Количество произведенного продукта пропорционально потраченному ресурсу. Перерабатывая ресурс, агент тратит энергию. Количество затраченной энергии зависит от количества произведённых предметов, на один предмет агент тратит энергию E_{unit}. При этом зафиксирована норма (S_{rate}), т.е. количество ресурса, которое добывает агент с зафиксированным набором атрибутов ($\{ENERGY, OBEDIENCE\} = \{1,1\}$). Из единицы ресурса агент может произвести P_{piece} предметов.

Предлагается использовать линейную зависимость от энергии и послушания. Потраченная энергия пропорциональна количеству произведённых предметов. Будем считать, что от $ENERGY$ уже отняли количество E_{life}, необходимое для поддержания жизни агента.

$$STOCK_{proc} = \min \left\{ \begin{array}{c} STOCK \\ ENERGY \times OBEDIENCE \times S_{rate} \\ P_{max} / P_{piece} \end{array} \right\}$$

$$PRODUCTS_{proc} = P_{piece} \times STOCK_{proc}$$

$$ENERGY_{proc} = PRODUCTS_{proc} \times E_{unit}$$

Отметим, что при выполнении соотношения

$S_{rate} \times E_{unit} \leq 1$

затраченная энергия всегда меньше энергии агента (с учетом того, что некоторую ее часть уже потратили на поддержание жизни агента).

Свободный агент-рабочий добытые ресурсы и произведённые товары продаёт на «рынке», получая за них некоторую плату в зависимости от проданного количества, несвободный агент-рабочий отдаёт их контролирующему его агенту-собственнику в обмен на заработанную плату.

$$\left\{ \begin{array}{l} STOCK\_ \Leftarrow STOCK_{market} \\ PRODUCTS\_ \leftarrow PRODUCTS_{market} \\ FUNDS_{+} \Leftarrow FUNDS_{market} \end{array} \right. \quad \left\{ \begin{array}{l} STOCK\_ \Leftarrow STOCK_{wage} \\ PRODUCTS\_ \Leftarrow PRODUCTS_{wage} \\ FUNDS_{+} \Leftarrow FUNDS_{wage} \end{array} \right.$$

Кроме того, свободные рабочие должны выплачивать определённый налог чиновнику. Мы считаем, что за рабочих, контролируемых собственником, налоги выплачивает сам собственник.

$FUNDS\_ \Leftarrow FUNDS_{tax}$

В зависимости от физического состояния агента может изменяться и уровень его послушания. Если уровень энергии падает ниже определённой величины, то уровень послушания снижается. Если же, наоборот, уровень энергии достаточно высокий, то уровень послушания возрастает.

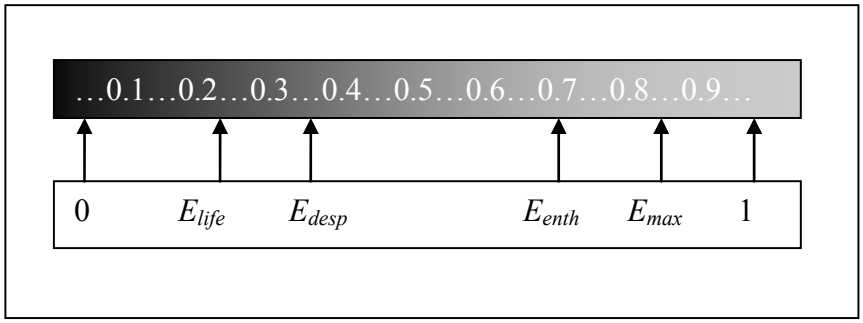


Рис. 2. Энергия, граничные точки

$$ENERGY < E_{desp} \Rightarrow OBEDIENCE \downarrow$$

$$ENERGY > E_{enth} \Rightarrow OBEDIENCE \uparrow$$

Атрибут $AUTHORITY$ является пассивным для агентов-рабочих. Несмотря на это, агенты могут повышать или понижать уровень своего авторитета. Например, если количество добытых ресурсов больше, чем среднее количество в некоторой окрестности, то авторитет растёт – принцип «лучшего работника». В другом случае можно сравнивать количество накоплений со средним по территории – принцип «самого успешного».

$$\left\{ \begin{array}{l} \Lambda_S = STOCK(n) - \overline{STOCK} \geq S_{auth} \\ \Rightarrow AUTHORITY(n+1) = \min\{AUTHORITY(n) + g_S(\Lambda_S), 1\} \\ \Lambda_S = STOCK(n) - \overline{STOCK} \leq -S_{auth} \\ \Rightarrow AUTHORITY(n+1) = \max\{AUTHORITY(n) - g_S(\Lambda_S), 0\} \end{array} \right.$$

$$\begin{cases} \Lambda_P = PRODUCTS(n) - PRODUCTS \geq P_{auth} \\ \Rightarrow AUTHORITY(n+1) = \min\{AUTHORITY(n) + g_P(\Lambda_P), 1\} \\ \Lambda_P = PRODUCTS(n) - \overline{PRODUCTS} \leq -P_{auth} \\ \Rightarrow AUTHORITY(n+1) = \max\{AUTHORITY(n) - g_P(\Lambda_P), 0\} \end{cases}$$

$$\begin{cases} \Lambda_F = FUNDS(n) - FUNDS \geq F_{auth} \\ \Rightarrow AUTHORITY(n+1) = \min\{AUTHORITY(n) + g_F(\Lambda_F), 1\} \\ \Lambda_F = FUNDS(n) - \overline{FUNDS} \leq -F_{auth} \\ \Rightarrow AUTHORITY(n+1) = \max\{AUTHORITY(n) - g_F(\Lambda_F), 0\} \end{cases}$$

Можно рассматривать эти условия как по отдельности, так и в сочетании друг с другом. Отметим, что все величины S_{auth}, P_{auth}, F_{auth} положительны, как и функции g_S, g_P, g_F. Возможный вид функций представлен на графике. [2]

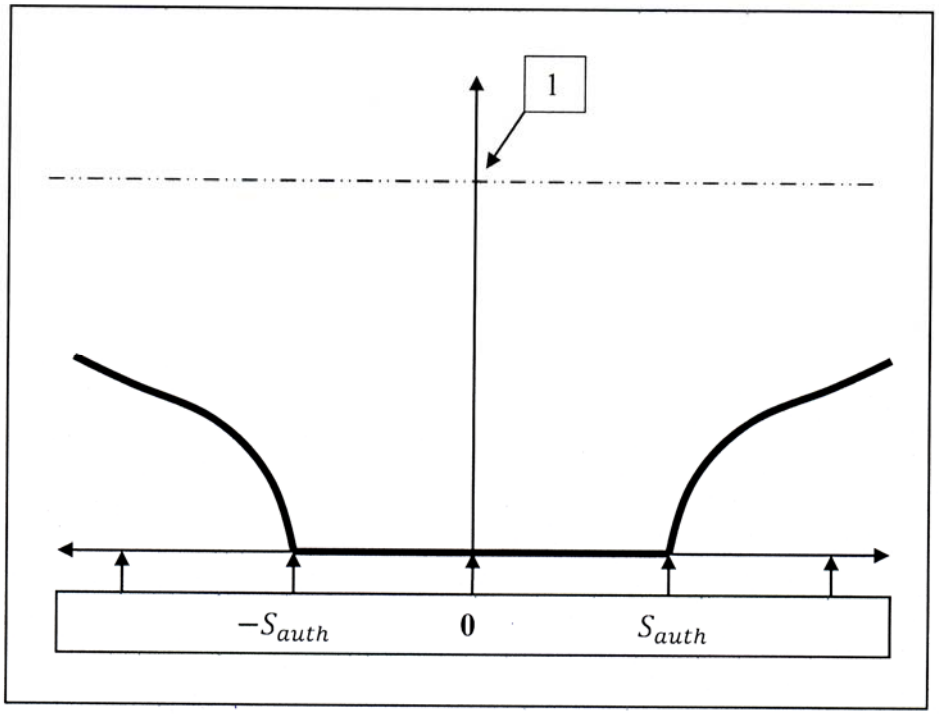


Рис. 3. Примерный вид функций функции g_s

Работник может перемещаться по территории по указанию чиновника или собственника, на которого он работает, а также в «поисках лучшей жизни», т.е там где средний уровень энергии выше на некоторую величину чем у него.

6. Агенты-чиновники

Каждый чиновник управляет некоторой территорией, чиновник контролирует других агентов, находящихся на этой территории. Величина контролируемой территории зависит от уровня авторитета чиновника. Функция d_c должна быть монотонно возрастающей и ограниченной.

$$CONTROL\,AREA(n;t) = \left\{ (x,y) \middle| \begin{matrix} d\big(COORDINATES(n;t),(x,y)\big) \leq d_c(AUTHORITY(n;t)) \\ \wedge\ (x,y) \in CONTROL\,AREA(n;f)\ if\ f \neq t \end{matrix} \right\}$$

Кроме того в некоторых моделях возможна иерархия чиновников. Один чиновник может контролировать других чиновников. В этом случае, для представления связей можно использовать дерево подчинения. Чиновники, находящиеся на более высоких уровнях, могут не иметь привязки к определенной территории, но у чиновников, представляемых листьями дерева, всегда есть подконтрольная им территория.

Если работники на территории чиновника живут хорошо, то авторитет чиновника растёт. Можно считать, что уровень повышение или понижение авторитета зависит от среднего уровня послушания работников за некоторый промежуток времени (аналогично можно рассмотреть зависимости от среднего уровня энергии или

накоплений). [1] Увеличение и уменьшение уровня авторитета происходит не каждый шаг, а через некоторые интервалы времени.

$$\begin{cases} Y_S = \overline{OBEDIENCE} \geq O_{hap} \Rightarrow AUTHORITY(n+1) = \min\{AUTHORITY(n) + h_O(Y_S), 1\} \\ Y_S = \overline{OBEDIENCE} \leq O_{mis} \Rightarrow AUTHORITY(n+1) = \max\{AUTHORITY(n) - h_O(Y_S), 0\} \end{cases}$$

$$0 \leq O_{disob} \leq O_{mis} \leq O_{hap} \leq O_{max} \leq 1$$

Функция h_O имеет вид аналогичный функциям g_S, g_P, g_F.

Если уровень авторитетности падает, то часть территорий может выйти из-под контроля чиновника. Если авторитет позволяет контролировать большие территории, то чиновник стремится захватить свободную местность или отобрать часть территории у чиновника с меньшим авторитетом. Также возможны модели, где чиновника назначают для управления территорией, тогда при повышении авторитета чиновника могут переводить на другие бóльшие территории, при снижении авторитета снимать с должности.

Чиновник собирает налоги с подконтрольных ему территорий – с собственников и со свободных работников (не работающих у собственника). Налоги могут представлять как фиксированную выплату, так и некоторый процент с заработанных средств. Чиновник может контролировать величину налогов в зависимости от состояния работников на подчинённой ему территории.

$$FUNDS_+(n) \Leftarrow \sum_{t \in O} FUNDS_{tax}(n;t)$$

$$FUNDS_{tax}(n;t) = r(*_+(n;t), ENERGIE(n;t)), \qquad * \in (STOCK, PRODUCTS, FUNDS)$$

Накопления чиновника считаем государственными деньгами. Часть денег чиновник тратит на поддержание своей энергии, какую-то часть, возможно, присваивает себе. Возможно, он отдаёт все накопления или их часть вышестоящему чиновнику. Деньги могут тратиться чиновником на поддержание населения, оказавшегося в «трудной ситуации».

$$FUNDS\_(n) \Leftarrow \sum_{t \in O} FUNDS_{help}(n;t)$$

$$FUNDS_{help}(n;t) = s(ENERGIE(n;t))$$

Чиновник также может давать некоторые указания агентам, находящимся на его территории. Например, анализировать окружающую среду. Полученные данные он может использовать для определения территорий, где например, наиболее выгодна добыча ресурсов, и отправлять туда агентов.

7. Взаимодействие агентов.

Агенты могут взаимодействовать как внутри группы, так и между группами.

Рабочие могут общаться друг с другом, узнавать о состоянии своих соседей, условиях окружающей среды, и в дальнейшем на основе этой информации делать вывод о наиболее предпочтительном местоположении.

Чиновник имеет возможность запрашивать у рабочих информацию о ресурсах в области их видимости. На основе полученной информации может отдавать рабочим приказание двигаться к более насыщенному источнику ресурсов.

Чиновник имеет право взимать часть средств, заработанных рабочими за ход. В случае, отсутствия у рабочих накоплений для поддержания жизни и энергии может выдавать им необходимое количество денег.

Чиновники могут взаимодействовать между собой. Например, в случае, когда на его территории находится много ресурсов, но мало людей, чиновник может спросить у своего соседа-чиновника дополнительных работников. Если же наоборот мало ресурсов и много людей, он может попытаться найти новые свободные территории или отдать людей другому чиновнику.

8. Появление новых агентов, переход агентов из одной группы в другую

Если средний уровень жизни достаточно высок, то количество агентов-рабочих может увеличиться. Уровень жизни может оцениваться по уровню энергии, послушания и количеству накоплений. Высокий уровень энергии означает хорошее физическое состояние агента, высокий уровень послушания – хорошее психологическое состояние (высокий уровень послушания говорит о том, что в течение длительного периода состояние агента было хорошим). Высокий уровень накоплений означает, что работники в данном месте не отдают все свои средст-

ва на налоги, есть некоторый излишек ресурса, т.е. на данной территории может содержаться ещё несколько агентов.

Для агентов-чиновников можно рассматривать модели, в которых новые агенты появляются только из класса рабочих, и модели, где чиновники могут появляться внутри своей группы. [1] Во втором случае необходимо уточнить, какие территории достаются «новоиспечённому» чиновнику. Можно считать, что если у некоторого чиновника уровень авторитетности очень высок, то он может (но не обязательно) породить нового чиновника (его протеже). Тогда новому чиновнику достаются некоторые свободные территории, если они имеются, или часть территорий его покровителя. Кроме того возможно образование иерархии внутри группы. Например, при появлении нескольких новых чиновников, они делят всю территорию чиновника, породившего их, а он сам становится их главой.

Если уровень авторитетности агента-рабочего высок, то работник может стать чиновником. Т.е. если $AUTHORITY \geq A_{w\ to}$, то с какой-то вероятностью p работник может стать чиновников. Если поблизости есть свободная территория вне зоны влияния других чиновников, то агент с достаточным уровнем авторитетности станет чиновником и займёт её. Кроме того, если авторитет агента выше, чем авторитет чиновника на данной территории, то он может отобрать часть территории. Можно также рассмотреть модель, где из нескольких подходящих кандидатов чиновники выбирается один.

9. Передвижение рабочих

Пространство, в котором располагаются агенты, разбивается на некоторые территории (в зависимости, например, от предполагаемого ландшафта, социального устройства или от того, как чиновники контролируют эти территории).

На каждой территории определяется геометрический «центр тяжести». На каждом шаге берутся все агенты-работники, находящиеся на этой территории и определяется E - средняя энергия по территории. Вообще говоря, вместо средней энергии можно использовать средние накопления, но мы все таки будем использовать энергию, как более наглядный атрибут для других рабочих. А, например, если рассматривать передвижение собственников, логичнее выбрать не среднюю энергию, а средние накопления.

В простейшем случае средняя энергия по территории определяется просто как среднее арифметическое всех агентов. $\bar{E}_k = \frac{1}{N_k}\sum_{i \in I_k} E_i$, где I_k - множество агентов находящихся на этой территории k, а N_k - их количество.

Теперь рассмотрим агента-работника. Важно определить границу видимости для него, то есть территорию, которую он будет рассматривать при передвижении. Можно взять простое соседство – территория, где находится агент, и все смежные с ней территории. Также можно рассмотреть расширенное соседство – смежные территории «второго» порядка.

(а) (б)

Рис. 4. Простое (а) и расширенное соседство (б)

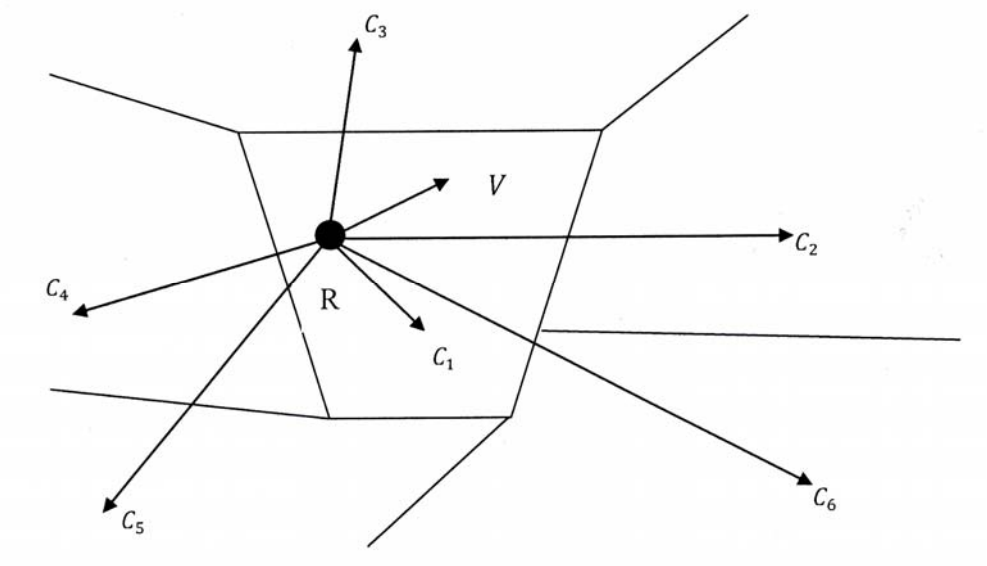


Рис. 5. Направление движения агента

В данном случае агент-работник R находится на территории T_1. Берем простое соседство, то есть рассмотрим также смежные территории $T_i (i \in [2,6])$. Из векторов направленных от агента к центрам тяжести C_j территорий длины E_j получаем результирующий вектор V, по направлению которого двигается агент.

Может возникнуть проблема, если, например, ресурсы в какой-то момент времени сосредоточены на границах территории, а агенты, несмотря на это, будут стягиваться к центру. Можно видоизменить способ определения средней энергии, учитывая близость агентов к центру территории: $\bar{E}_k = \frac{1}{N_k} \sum_{t \in I_k} E_t / |x_t - x_c|^{\alpha}$, где x_t, x_c - координаты агентов и центра, α – степень ослабления расстояния.

Также возможен другой вариант определения средней энергии, когда в качестве коэффициентов берутся не расстояния, а уровень авторитета. Чем больше вес в обществе (то есть больше на виду), тем большее впечатление на соседей. $\bar{E}_k = \frac{1}{N_k} \sum_{t \in I_k} E_t \times A_t$. Но так как авторитет, вообще говоря, зависит, как и энергия, от накоплений, то этот вариант представляется не самым лучшим.

Кроме того, чиновники, к которым приписан данный агент, также задает вектор направления (например, приказание двигаться к более насыщенному источнику ресурсов). Длина этого вектора зависит от величины авторитетности чиновника (обычно больше величин $\bar{E}_k$) и итоговое направление движение агента определяется суммой векторов от агента к центрам тяжести смежных территорий и направлением чиновника.

Поскольку работники не могут находиться в постоянном передвижении, можно ввести степень случайности в то, будет ли он перемещаться. Вообще говоря, логично ввести степень случайности для принятия работником решения, прислушаться ли к требованию чиновника двигаться в том или ином направлении. Ведь если мы не рассматриваем случай рабства, работникам необходимы определенные стимулы для переезда, и они не могут быть уверены в правильности указаний чиновника. То есть перед каждым перемещением, которое обусловлено вектором, случайным образом с определенной вероятностью происходит выбор – состоится ли это перемещение.

8. Программная реализация

Для реализации вышеприведенных моделей можно использовать несколько инструментов таких, как Swarm, RePast, NetLogo. Нами был выбран RePast, как наиболее оптимальный для социального моделирования инструмент. Основными критериями выбора:

- • Актуальность и регулярная поддержка (разработка продукта).
- • Наличие хорошей и доступной документации.
- • Легкость в расширении.
- • Возможность подключения внешних инструментов (интеграция с базами данных, математическими пакетами и т.д.)

На данном этапе реализована одна из простых моделей. На сетке случайным образом распределяются ресурсы, а также агенты двух типов: рабочие и чиновники. У чиновников есть возможность управления передви-

жениями и добычей ресурса рабочими. Рабочие добывают ресурс, исследуют окружающие пространство и отчисляют чиновникам-владельцам часть добытых и переработанных ресурсов. При высоком стремлении чиновников к обогащению происходит вымирание рабочих.

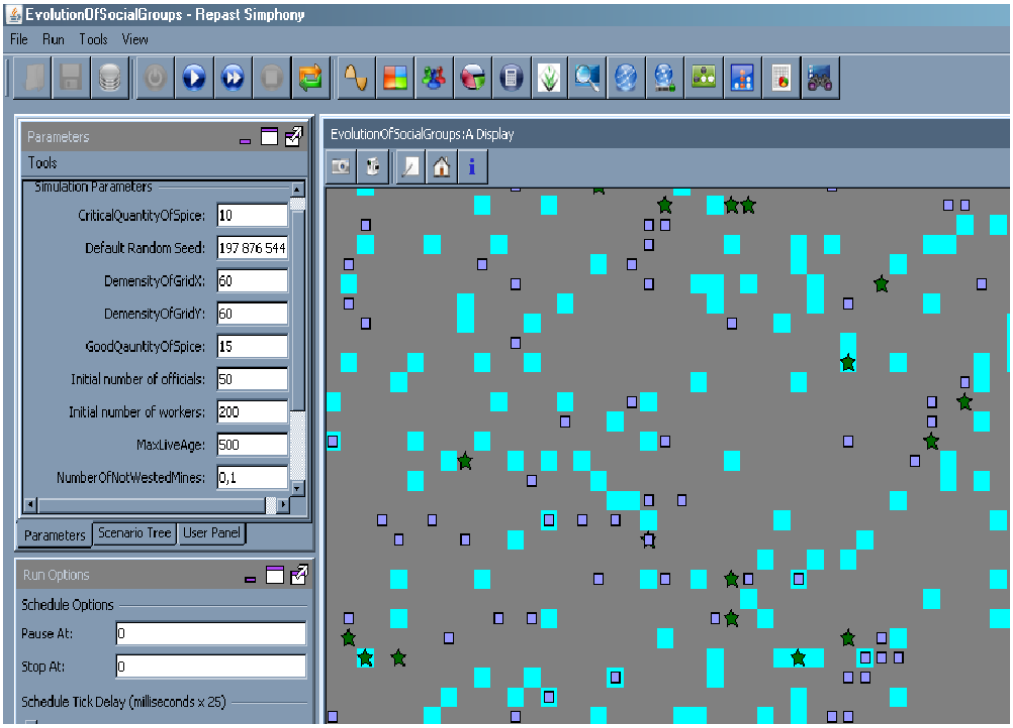


Рис. 6. Начальное положение агентов

Голубые клетки – ресурсы, зеленые звездочки – агенты-чиновники и, соответственно, сиреневые квадраты – агенты-рабочие.

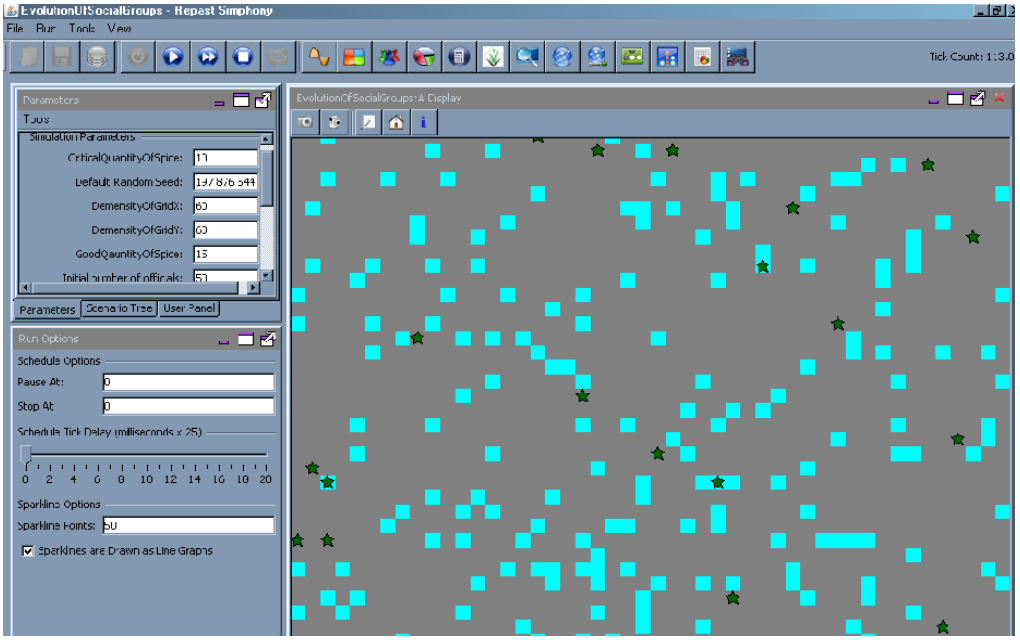


Рис. 7. Результат работы программы

На следующем рисунке представлен график зависимости количества добытых ресурсов от времени. Такой

график строится в реальном времени при моделировании, вследствие чего есть не только возможность наблюдения за передвижениями агентов, но также есть возможность анализа ситуации.

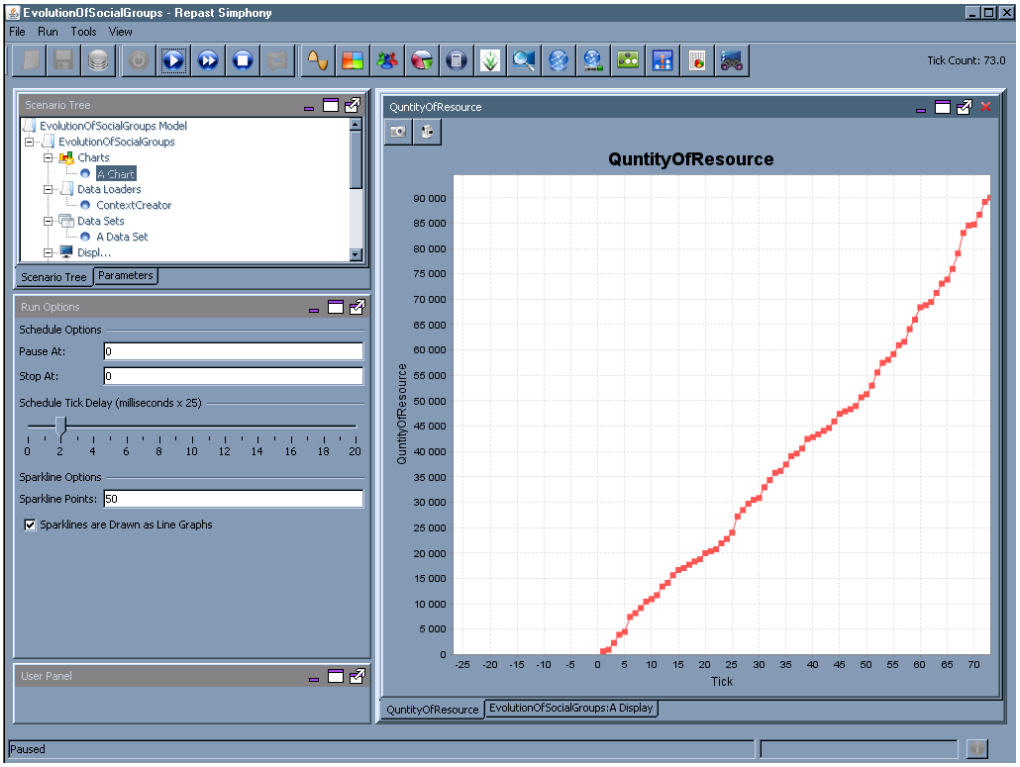


Рис. 8. График зависимости количества добытых ресурсов от времени

Заключение

Две наиболее трудоемкие части социального моделирования – выбор показательного примера целевой группы для формирования базового набора данных и развитие обоснованного набора правил поведения и вероятностных характеристик, влияющих на поведение агентов. Если набор данных и правила поведения хорошо разработаны, то моделирование, имитирующее социальные процессы, будет более надежным, реалистичным и заслуживающим доверие.

Более детализированные наборы данных, описывающих демографические показатели, правила поведения в группе и сеть социальных связей, могут быть полезными при дальнейшем построении моделей.

Представляется перспективным применение данного продукта для анализа демографических процессов, построения моделей управления рисками, анализа социальных сетей.

В ходе выполнения работы были изучены и проанализированы различные наборы инструментов для мультиагентного моделирования социальных процессов, проанализированы их достоинства и недостатки.

Построена система являющаяся шаблоном для моделирования различных социальных процессов в рамках исследуемой области, позволяющая учитывать динамические факторы, влияющие на социальный процесс.

Частично реализована разработанная математическая модель, однако планируется дальнейшее развитие, с использованием теории Латане и PIC-метода, представленными в данной работе, добавление возможности эволюции модели общества, смешивания социальных групп и построения неоднозначных моделей, продолжение исследование различных социальных процессов с помощью данной системы.

Список литературы

1. Малков С.Ю., Селунская Н.Б., Сергеев А.В. Социально-экономические и демократические процессы в аграрном обществе как объект математического моделирования. // "История и синергетика. Математическое моделирование социальной динамики". – М., «Ком Книга», 2005 г.

2. Marsella, S. C., Pynadath, D. V., Read, S. J. PsychSim: Agent-based modeling of social interactions and influence. // Paper in the Proceedings of the 6th International Conference on Cognitive Modeling, Carnegie Mellon University, Pittsburgh, PA, 2004.
3. Копылова Н.С., Мурзин Ф.А. Моделирование механизмов социального влияния на основе мультиагентного подхода // Труды 11-ой национальной конференции по искусственному интеллекту с международным участием КИИ-2008 (г.Дубна, Россия). –М.: ЛЕНАНД, 2008. –Т.3. –С.226-234.
4. Копылова Н.С., Мурзин Ф.А. Моделирование механизмов социального влияния на основе мультиагентного подхода // Вопросы искусственного интеллекта (Вестник НСМИИ РАН), – 2009. – С. 173–183.
5. Белогубова М.В., Копылова Н.С. Моделирование механизмов социального влияния с помощью мультиагентного похода // Материалы XLVI международной научной студенческой конференции Студент и научно-технический прогресс»,, Новосибирск, 2008, С.132-133.
6. Белогубова М.В., Копылова Н.С. Моделирование социальных процессов на основе мультиагентного подхода // Материалы XLVIII международной научной студенческой конференции «Студент и научно-технический прогресс», Новосибирск, 2010, С.194.

Формирование интегральной olap-модели предметной области на основе формального концептуального анализа*

А.В. Коробко, Т.Г. Пенькова

Институт Вычислительного Моделирования СО РАН,
Академгородок, 50, стр.44, 660036, Красноярск, Россия
{Lynx, Penkova_t}@icm.krasn.ru

Аннотация. Предложен метод концептуального OLAP-моделирования предметной области. Представлено описание формирования интегральной аналитической модели в виде формальной концептуальной решетки OLAP-кубов. Выполнено построение концептуальной аналитической модели размещения муниципального заказа.

Ключевые слова: интегральная OLAP-модель, оперативная аналитическая обработка данных, формальный концептуальный анализ.

Эффективность управления административными ресурсами во многом определяется своевременностью предоставления аналитической информации. Для поддержки принятия управленческих решений широко используется технология OLAP (On-line analytical processing) [1-5]. Необходимость оперативной аналитической обработки больших объемов данных в задачах организационного управления (территориального, отраслевого, корпоративного и т.п.) требует создания новых подходов к реализации технологии OLAP. Качество анализа данных на основе OLAP во многом определяется доступностью исходных данных и прозрачностью аналитической модели предметной области. Как правило, аналитическая модель представляет собой множество отдельных OLAP-моделей предназначенных для решения частных задач [1-3]. Формирование такой фрагментарной модели происходит из-за необходимости привлечения специалиста со знанием структуры и состава исходных данных для решения каждой новой аналитической задачи. Применение специализированного хранилища данных [6-8] в качестве источника исходной информации позволяет частично избежать ошибок согласованности анализируемых данных, но не решает проблему отсутствия возможности оперировать всеми объектами анализа предметной области. Актуальной становится задача формирования интегральной OLAP-модели предметной области. В ряде случаев [4, 9] интегральный подход реализуется построением каталога показателей, позволяющего систематизировать объекты анализа, но не обеспечивающего поддержку их совместной аналитической обработки. Необходимо разработать метод формирования интегральной OLAP-модели на основе структурирования экспертных знаний об объектах анализа предметной области и возможности их совместной аналитической обработки.

Предлагается метод формирования интегральной OLAP-модели, основанный на формальном концептуальном анализе показателей и измерений предметной области. Описан процесс построения интегральной аналитической модели предметной области в виде концептуальной решетки OLAP-кубов. Интегральная OLAP-модель позволяет оперировать всеми объектами анализа и охватывает максимальное число решаемых аналитических задач. Применение интегральной OLAP-модели предметной области для поддержки принятия управленческих решений позволяет повысить эффективность оперативной аналитической обработки многомерных данных.

OLAP-моделирование предметной области

Технология OLAP представляет собой современную концепцию анализа данных, описанную совокупностью требований к программным продуктам, обеспечивающим оперативную аналитическую обработку и представление данных. Впервые принципы OLAP были сформулированы основоположником теории реляционных баз данных Е. Коддом [10]. OLAP обеспечивает пользователя естественной, интуитивно понятной моделью данных, организуя их в виде многомерных кубов.

OLAP-куб можно рассматривать как гиперкуб $G = <D, F>$ – модель логического многомерного представления данных, характеризующаяся двумя наборами параметров: показателями и измерениями (рисунок 1);

$F = <f_1, f_2, \ldots, f_n>$ – показатели (меры) гиперкуба: каждый показатель имеет множество значений, количественно характеризующих анализируемый процесс;

$D = <d_1, d_2, \ldots, d_m>$ – измерения гиперкуба: каждое измерение представляет собой упорядоченное множество значений определенного типа. Измерения могут быть организованы в виде упорядоченной иерархической структуры. Множество измерений образует оси гиперкуба. В ячейках куба, находятся аналитические показатели.

* Работа выполнена при поддержке гранта ФЦП «Научные и научно-педагогические кадры инновационной России» на 2009-2013 годы (ГК № 02.740.11.0621 от 29.03.2010 г.).

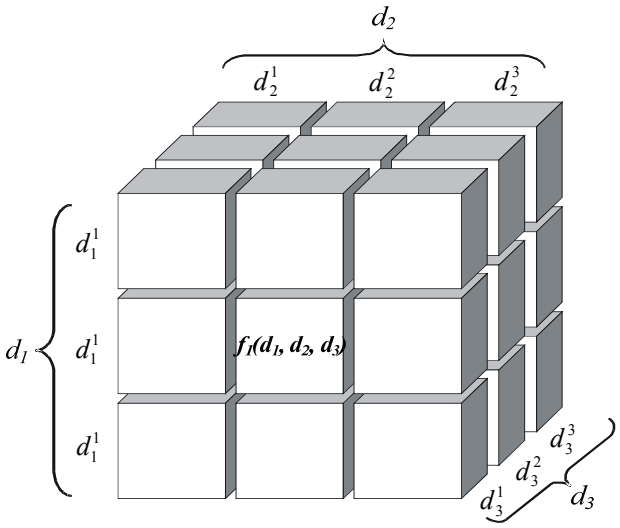


Рис. 1. OLAP-куб

Одно из основных требований технологии OLAP – «прозрачность»: готовый многомерный куб должен быть представлен конечному пользователю в удобном для него виде, инструменты манипулирования кубом должны быть интуитивно понятны, наименования объектов анализа должны соответствовать терминологии предметной области. Процесс OLAP-анализа представляется совокупностью операций с многомерными данными – детализации, консолидации (группировки), формирования среза и поворота. Операции консолидации определяют переход от детального представления данных к агрегированному, а в случае детализации – обратный переход. Формирование среза куба заключается в фиксации значения (значений) определенного измерения, при этом сокращается размерность куба. Срез представляет собой подкуб, в который входят все остальные измерения. Операция поворота заключается в изменении положения осей куба – измерений. В результате вращения меняется «точка зрения» на данные.

Для конечного пользователя применение OLAP обеспечивает высокую гибкость получаемых решений за счет возможности изменения отображения результата. Аналитик получает не жестко регламентированный отчет, а инструментарий для творческого исследования задачи. Возможность свободной манипуляции данными упрощает получение необходимых наборов данных. Важным преимуществом OLAP является предоставление пользователю возможности оперировать знакомыми терминами из предметной области [11].

OLAP-куб можно рассматривать как абстрактное представление выбранного подмножества реляционной базы данных [12]. Процесс формирования OLAP-куба включает выбор необходимых таблиц, расстановку связей между ними, выбор полей данных и сопоставление их с терминами предметной области, что требует специальных знаний о предметной области и структуре исходных данных. Количество и состав многомерных кубов определяется решаемыми задачами. Построение OLAP-куба для каждой частной аналитической задачи ведет к формированию фрагментарной аналитической модели предметной области. В результате наблюдается многократное использование одних и тех же объектов анализа (показателей и измерений) для построения различных кубов в рамках одной модели. Расширение круга аналитических задач ведет к необходимости объединения существующих кубов с обязательной проверкой сопоставимости показателей и измерений. Для повышения эффективности оперативной аналитической обработки данных необходима интегральная OLAP-модель, построенная на основе всех объектов анализа предметной области. Возможность манипулирования всеми объектами анализа предметной области одновременно сопряжена с необходимостью использования экспертных знаний об объектах анализа и о возможности их совместной аналитической обработки.

Формирование интегральной OLAP-модели предметной области на основе формального концептуального анализа

С целью формирования интегральной OLAP-модели предметной области на основе всех объектов анализа необходимо выделить группы объектов, имеющих общие структурные признаки, исходя из отношения сопоставимости и возможности их совместной аналитической обработки. Для этого целесообразно использовать методы бинарной объектно-признаковой кластеризации, в которых сходство объединяемых в один кластер объектов выражается через общие элементы описания всех объектов из данного кластера [13]. К таким методам относятся методы, основанные на формальных концептах и решетках формальных концептов [14].

Формальный концептуальный анализ (Formal Concept Analysis) впервые был предложен R. Wille в 1981 го-

ду [15] и активно развивается сегодня. Метод заключается в следующем. Формальным контекстом называется тройка $K = (G, M, I)$, где G – множество объектов, M – множество атрибутов, $I \subseteq G \times M$ – отношение такое, что gIm, где $g \in G$, $m \in M$ означает, что объект g обладает атрибутом m. Формальный контекст может быть представлен в виде бинарной матрицы, строки которой помечены именами объектов, а столбцы – значениями атрибутов. Для произвольных $A \subseteq G$ и $B \subseteq M$ определяются A' и B':

$A' = \{m \in M \mid gIm$ для всех $g \in A\}$,

$B' = \{g \in G \mid gIm$ для всех $m \in B\}$.

Пара множеств (A, B) таких, что $A \subseteq G$, $B \subseteq M$, $A' = B$ и $B' = A$, называется формальным концептом контекста K. Множество объектов A представляет объем формального концепта, а множество атрибутов B – содержание формального концепта. Таким образом, формальный концепт – это множество объектов предметной области, каждый из которых обладает всеми атрибутами из некоторого подмножества атрибутов, присущих всем этим объектам.

Применение формального концептуального анализа к объектам оперативной аналитической обработки многомерных данных позволяет построить интегральную OLAP- модель на основе экспертных знаний об объектах анализа предметной области и возможности их совместной аналитической обработки. На основе интеграции технологии OLAP и формального концептуального анализа предложен метод формирования концептуальной аналитической модели предметной области в виде формальной решетки многомерных кубов [16]. На рисунке 2 представлена контекстная диаграмма IDEF0 формирования интегральной OLAP-модели предметной области.

Рис. 2. Формирование интегральной OLAP-модели предметной области

Предлагаемый метод включает следующие основные этапы:

1. Формирование множества объектов анализа предметной области;
2. Построение формального контекста;
3. Определение формальных кубов-концептов;
4. Построение формальной концептуальной решетки OLAP-кубов.

На первом этапе путем интервьюирования конечного пользователя и изучения отчетных форм эксперт определяет аналитические задачи предметной области, формирует множество аналитических запросов. Сформулированные запросы позволяют определить множество терминов предметной области – объектов анализа, которые используются для построения OLAP-кубов концептуальной аналитической модели. Объекты концептуального анализа, в соответствии с моделью многомерного представления данных, делятся на множество измерений $D = \{d_1, d_2, \ldots, d_n\}$ и множество показателей $F = \{f_1, f_2, \ldots, f_m\}$. Термины, определяющие аспекты анализа предметной области, образуют множество измерений, а термины, представляющие количественные характери-

стики анализируемого процесса, образуют множество показателей. Затем на основе структуры хранилища данных выполняется сопоставление объектов анализа с полями таблиц, определяя тем самым физическую составляющую измерений и показателей. Объекты анализа могут быть связаны с полями таблиц хранилища данных напрямую, или рассчитываться на основе нескольких полей по заданному алгоритму расчета.

На втором этапе формирования интегральной OLAP-модели экспертом структурируется множество объектов анализа, сформированное на предыдущем этапе. На основе экспертных знаний об аналитических особенностях информационных объектов и функциональных зависимостей хранилища данных определяется сопоставимость показателей и измерений – возможность их совместной аналитической обработки.

Отношение сопоставимости между элементами множеств F и D обозначим R. $R \subseteq F \times D$, $(f_i, d_j) \in R$, если показатель f_i может быть проанализирован по измерению d_j. Тройка (F, D, R), в соответствии с теорией формального концептуального анализа, представляет собой формальный контекст K. Формальный контекст отражает знания эксперта об объектах анализа предметной области и о возможности их совместной аналитической обработки. Формальный контекст может быть представлен в виде бинарной матрицы, строки которой соответствуют показателям, а столбцы – измерениям.

На третьем этапе на основе сформированного формального контекста определяется множество кубов-концептов по признаку сопоставимости объектов анализа.

Теория формального концептуального анализа позволяет объединять характеристики анализируемого процесса по признаку общности аспектов анализа в кластеры – концепты. Множество показателей одинаковой размерности A, которые могут быть проанализированы по всем измерениям из B, образуют куб-концепт (A, B). Множество показателей A представляет объем формального куба-концепта, а множество измерений B – содержание формального куба-концепта.

В соответствии с моделью многомерного представления данных, формальный куб-концепт – это аналитический многомерный куб, полный относительно добавления показателей той же размерности и состава измерений. Это означает, что невозможно включить в такой OLAP-куб дополнительный показатель без уменьшения числа измерений, то есть в рамках построенного формального контекста не существует других показателей, сопоставимых с тем же набором измерений.

На заключительном этапе строится решетка кубов-концептов, которая позволяет оперировать всеми объектами анализа и охватывает максимально возможное число решаемых аналитических задач рассматриваемой предметной области.

Согласно методу формального концептуального анализа, множество всех концептов частично упорядочено отношением подконцепт-надконцепт: $(A_1, B_1) \leq (A_2, B_2)$ если $A_1 \subseteq A_2$ (что эквивалентно $B_2 \subseteq B_1$). В этом случае (A_1, B_1) называют *подконцептом* (A_2, B_2), а (A_2, B_2) – *надконцептом* (A_1, B_1). Упорядоченное множество всех концептов контекста образует полную решетку – решетку концептов [17].

Для концептуальной OLAP-модели отношение подконцепт-надконцепт определяется как подкуб-надкуб: множество показателей родительского куба включает множество показателей дочернего куба, а, в свою очередь, множество измерений дочернего куба включает множество измерений родительского куба. Решетка кубов-концептов представляет собой визуализацию интегральной OLAP-модели предметной области.

Интегральная OLAP-модель размещения муниципального заказа

Рассмотрим формирование аналитической модели, основанной на формальном концептуальном анализе, на примере размещения муниципального заказа.

Муниципальный заказ – это заказ на поставку товаров, выполнение работ, оказание услуг, формируемый от имени органов местного самоуправления, финансируемый из местного бюджета и направленный на удовлетворение муниципальных нужд [18]. Размещение муниципального заказа является одной из важнейших организационных форм территориального управления и представляет инструмент эффективного использования бюджетных средств.

Процедура размещения заказа включает три основных этапа:

- планирование – распределение и контроль объемов бюджетных ассигнований и закупаемой продукции;
- размещение – организация и проведение торгов;
- контроль – заключение контрактов и управление исполнением.

Анализ эффективности размещения муниципального заказа на каждом из этапов связан с решением различных аналитических задач:

- анализ дефектных ведомостей муниципальных потребностей;
- анализ распределения объемов закупаемой продукции;
- анализ результатов проведения торгов;
- мониторинг возврата обеспечения;
- анализ активности участников размещения заказа;
- оценка эффективности распределения бюджетных средств;

– мониторинг исполнения контрактов;
– оценка эффективности деятельности уполномоченного органа;
– и т. д.

Решение задач связано с выполнением следующих запросов:

– сумма бюджетных средств, выделяемая на удовлетворение муниципальных нужд бюджетополучателей;
– объем бюджетных ассигнований на приобретение отдельных категорий продукции;
– количество конкурсов, аукционов, запросов котировок, проводимых за месяц, квартал, год;
– количество процедур, проводимых на поставку товаров, выполнение работ, оказание услуг;
– количество лотов, несостоявшихся по причинам несоответствия заявки требованиям заказчика, непредоставления обязательных документов, невнесения денежных средств в качестве обеспечения, в связи с отсутствием заявок и т.д.;
– количество заявок по различным типам размещения заказа, поданных участниками, имеющими преимущества;
– начальная цена контракта и цена по итогам проведения конкурсов, аукционов, запросов котировок на поставку товаров, выполнение работ, оказание услуг по главным распорядителям бюджетных средств;
– количество контрактов, заключенных с участниками размещения заказа, занявшими первое и второе место;
– количество неисполненных контактов с указанием суммы денежных средств по бюджетополучателям и главным распорядителям бюджетных средств;
– и т.д.

Анализ запросов позволяет выделить термины предметной области и на их основе определить множество объектов анализа размещения заказа:

– показателей: количество процедур, количество лотов, количество заявок, количество контрактов начальная цена контракта, цена по итогам размещения, объем бюджетных ассигнований, размер обеспечения заявки, размер обеспечения контракта и т. д.
– измерений: способ размещения заказа, состояние закупки, категория продукции, категории преимущества заказа, главный распорядитель, главные распорядители, наименование заказчика, тип заказчика, состояние лота, статус заявки, состояние заявки, организационно-правовая форма участника, источник финансирования, форма оплаты, состояние контракта и т.д.

		d1 Способ размещения заказа	d2 Состояние закупки	d3 Категория продукции	d4 Категории преимущества заказа	d5 Главные распорядители	d6 Наименование заказчика	d7 Тип заказчика	d8 Состояние лота	d9 Статус заявки	d10 Состояние заявки	d11 Присвоенное место	d12 Наименование участника	d13 Категория участника	d14 Организационно-правовая форма участника	d15 Источник финансирования	d16 Форма оплаты	d17 Входящий номер закупки	d18 Наименование закупки	d19 Наименование лота	d20 Состояние контракта	d21 Тип сведений в реестре контрактов	d22 Состояние записи реестра контрактов	d23 Вид работы по исполнению контракта	d24 Номенклатура товаров	d25 Код бюджетной классификации	d26 Группа реестра потребностей	d27 Район г. Красноярска	d28 Месяц	d29 Квартал	d30 Год	d31 Ответственное лицо
f1	Количество процедур	×	×	×	×	×	×	×								×	×								×	×		×	×	×	×	×
f2	Количество лотов	×		×	×	×	×	×	×							×	×	×	×		×				×	×		×	×	×	×	
f3	Количество заявок	×		×		×	×	×	×	×	×	×	×	×	×	×	×	×	×	×					×			×	×	×	×	
f4	Количество контрактов	×		×	×	×	×	×				×	×		×	×	×				×	×	×	×	×	×			×	×	×	
f5	Начальная цена контракта	×	×	×	×	×	×	×	×							×	×	×	×	×							×		×	×	×	
f6	Цена по итогам размещения	×	×	×	×	×	×	×	×					×		×	×	×	×	×	×						×		×	×	×	
f7	Объем бюджетных ассигнований	×		×	×	×	×	×								×									×	×	×		×	×	×	
f8	Размер обеспечения заявки	×							×		×	×	×	×				×	×	×									×	×	×	
f9	Размер обеспечения контракта	×										×	×	×				×	×	×	×								×	×	×	

Рис. 3. Формальный контекст размещения муниципального заказа

С учетом сопоставимости показателей и измерений (совместной аналитической обработки), с точки зрения эксперта, определяется формальный контекст. На рисунке 3 в виде таблицы представлен формальный контекст размещения муниципального заказа. Определены следующие элементы множества F = {*количество процедур*, *количество лотов*, *количество заявок*, *количество контрактов*, *начальная цена контракта*, *цена по итогам размещения*, *объем бюджетных ассигнований*, *размер обеспечения заявки*, *размер обеспечения контракта*} и элементы множества D = {*способ размещения заказа*, *состояние закупки*, *категория продукции*, *категории преимущества заказа*, *главные распорядители*, *наименование заказчика*, *тип заказчика*, *состояние лота*, *статус заявки*, *состояние заявки*, *присвоенное место*, *наименование участника*, *категория участника*, *организационно-*

правовая форма участника, *источник финансирования*, *форма оплаты*, *входящий номер закупки*, *наименование закупки*, *наименование лота*, *состояние контракта*, *тип сведений в реестре контрактов*, *состояние записи реестра контрактов*, *вид работы по исполнению контракта*, *номенклатура товаров*, *код бюджетной классификации*, *группа реестра потребностей*, *район г. Красноярска*, *месяц*, *квартал*, *год*, *ответственный исполнитель уполномоченного органа*}.

Используя сокращенные обозначения, получим соответственно: $F = \{f_1, f_2, f_3, f_4, f_5, f_6, f_7, f_8, f_9\}$ и $D = \{d_1, d_2, d_3, d_4, d_5, d_6, d_7, d_8, d_9, d_{10}, d_{11}, d_{12}, d_{13}, d_{14}, d_{15}, d_{16}, d_{17}, d_{18}, d_{19}, d_{20}, d_{21}, d_{22}, d_{23}, d_{24}, d_{25}, d_{26}, d_{27}, d_{28}, d_{29}, d_{30}, d_{31}\}$.

Отношение R записывается следующим образом: $R = \{(f_1, d_1), (f_1, d_2), (f_1, d_3), (f_1, d_4), (f_1, d_5), (f_1, d_{14}), (f_1, d_{14}), (f_1, d_{24}), (f_1, d_{25}), (f_1, d_{27}), (f_1, d_{28}), (f_1, d_{29}), (f_1, d_{30}), (f_1, d_{31}), \ldots, (f_9, d_{30})\}$. В рамках рассматриваемого примера, показатели $A = \{f_1, f_2, f_4, f_7\}$ одновременно сопоставимы только с измерениями из $B = \{d_1, d_3, d_4, d_5, d_6, d_7, d_{15}, d_{24}, d_{25}, d_{28}, d_{29}, d_{30}\}$. Пара (A, B) образует формальный концепт – полный аналитический куб размещения заказов. Множество всех кубов-концептов, упорядоченное отношением подкуб-надкуб образует решетку кубов. На рисунке 4 приведена формальная концептуальная решетка OLAP-кубов, представляющая интегральную OLAP-модель размещения муниципального заказа.

Предложенный метод позволяет формировать интегральную аналитическую модель предметной области в виде решетки OLAP-кубов на основе экспертных знаний об объектах анализа и возможности их совместной аналитической обработки. Интегральная модель дает возможность оперировать всеми объектами анализа одновременно и охватывает максимальное число решаемых аналитических задач. Применение интегральной OLAP-модели для поддержки принятия управленческих решений позволит повысить эффективность оперативной аналитической обработки многомерных данных. Практическим результатом работы стало построение интегральной OLAP-модели размещения муниципального заказа.

Дальнейшее развитие метода OLAP-моделирования сопряжено с необходимостью разработки алгоритмов обнаружения дополнительных показателей и измерений, исходя из свойств концептуальной решетки и начальных условий решаемой аналитической задачи. Планируется разработка метода формирования базы знаний на основе формальной концептуальной решетки OLAP-кубов и создание программных средств поддержки адаптивного манипулирования объектами анализа при оперативной аналитической обработке многомерных данных.

Список литературы

1. Горелов Б.А., Горелов Б.Б. Разработка модели данных для целей оперативной аналитической обработки финансовой информации университета // Университетское управление. № 4(23). 2002. С. 33-46.
2. Ноженкова Л.Ф., Шайдуров В.В. OLAP-технологии оперативной информационно-аналитической поддержки организационного управления // Информационные технологии и вычислительные системы. № 2. 2010. С. 15-27.
3. Palaniappan S. and Ling C., Clinical Decision Support Using OLAP With Data Mining, International Journal of Computer Science and Network Security, VOL.8, No.9, 2008. pp. 290-296.
4. Javed A., Shaikh M. and Bhatti B. Conceptual Model for Decision Support System Based Business Intelligence OLAP Tool for Universities in Context of E-Learning, Proc. World Congress on Engineering and Computer Science, 2008.
5. Vasilecas O., Smaizys A. Business rule based data analysis for decision support and automation, Proc. International Conference on Computer Systems and Technologies, Vol. II, 2006.
6. Inmon W. Willey John & Sons, Building the Data Warehouse, New York, 1992.
7. Sperley E. Enterprise data warehouse: planning, building and implementation. Vol.1 Prentice Hall PTR, Upper Saddle River, NJ 07458, 2001. 400 p.
8. Бадмаева К.В. Методика адаптивного проектирования специализированных хранилищ данных // Труды XIV Байкальской конференции Часть III. Иркутск: ИСЭМ СО РАН, 2009. С. 214-221.
9. Боярский Н.А., Шовкун А.В. Построение аналитической части корпоративной информационно-аналитической системы средствами Oracle OLAP Option и BI Beans. [Электронный ресурс]: статья, Oracle Magazine RE, апрель-май, 2004. – Режим доступа: http://www.interface.ru/home.asp?artId=9604.
10. Codd E.F. Providing OLAP to user-analysts: An IT mandate. Codd and Associates, 1993.
11. Korobko A., Penkova T. OLAP-modeling of municipal procurement automation support problem // Proc. International Conference on Conceptual Structures (ICCS'09). 2009. pp. 87-91.
12. Gray J., Bosworth A., Layman A., Priahesh H. Data Cube: A Relational Aggregation Operator Generalizing Group-By, Cross-Tab, and Sub-Totals // Proceedings of the 12th International Conference on Data Engineering. IEEE, 1995. pp. 152-159.
13. Kedrov S., Kuznetsov S., Study of Groups of Web Users with Methods Based on Formal Concept Analysis and Data Mining, Business-informatics, №1, 2007. pp. 45-51.
14. Ganter B., Wille R. Formal Concept Analysis: mathematical Foundations. Springer-Verlag. Berlin Heidelberg New York, 1999.
15. Wille, R. Restructuring Lattice Theory: an approach based on hierarchies of concept. Reidel, Dordrecht-Boston, 1982. pp. 445-470.
16. Korobko A., Penkova T. On-line analytical processing based on Formal concept analysis // Procedia Computer Science.Vol. 1. Iss. 1., ELSEVIER. pp. 2305-2311.
17. Биркгоф Г. Теория решеток. М. : Наука. 1984. 568 с.
18. Храмкин А.А., Воробьева О.М., Вдовина В.В., Волосатова А.В., Ермаков В.А. Настольная книга госзаказчика – М.: Юриспруденция, 2006.– 312 с.

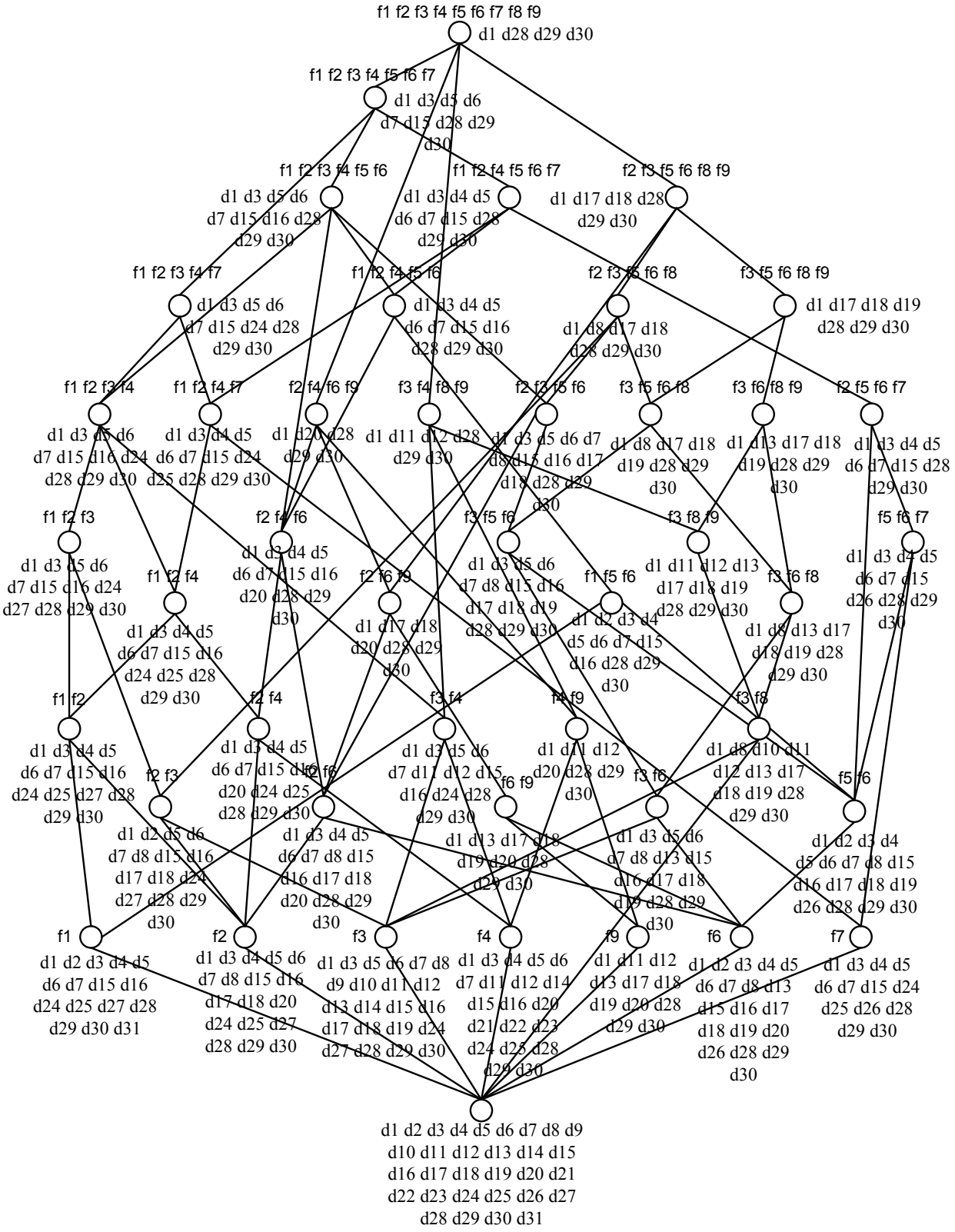


Рис. 4. Решетка формальных концептов размещения муниципального заказа

В поисках идеального языка
(философские предпосылки появления формальных языков)

А.Н. Савостьянов

1. В европейской философской традиции многократно предпринимались попытки создания «идеального» языка, свободного от «недостатков» естественных языков.
2. Первые намеки на такой язык содержатся в диалоге Платона «Кратил». В диалоге высказывается идея, согласно которой имя вещи должно полностью отражать сущность вещи. В таком случае, зная имя, человек может узнать все свойства вещи, которую это имя называет. По мнению участников диалога, в прошлом существовал язык, отвечающий требованиям идеального, но он был искажен. Современный же язык (т.е. язык эпохи Платона) лишь частично удовлетворяет таким требованиям. Поэтому, восстановление идеального языка мыслится как возвращение к золотому веку. Однако в конце диалога содержится и критика идеального языка. Один из участников диалога Кратил высказывает мнение, что изменчивые вещи не могут быть описаны точно. Представление изменчивых свойств мира в языке ведет к логическим противоречиям. Общее платоновское решение проблемы сводится к тому, что идеальный язык применим к описанию неизменного мира эйдосов, и лишь частично применим для описания изменчивого космоса.
3. Концепция «универсального языка» разрабатывалась в рамках философской системы Г. Лейбница. По его мнению, такой язык необходим для осуществления научных исследований. Лейбниц стремится исключить из языка оценочные высказывания, что, вероятно, связано с попытками избежать конфликтные дискуссии. Универсальный язык должен позволить философам избегать споров, т.к. любая проблема, сформулированная на этом языке, будет разрешаться при помощи вычислений. Лейбниц создает теорию вычислений, которую можно рассматривать как одного из предшественников современной информатики. Однако в поздние годы жизни Лейбниц приходит к выводу, что универсальный язык не применим для описания человеческого поведения, поскольку поведение опирается на субъективные оценки происходящих событий.
4. Наиболее полно, представления об идеальном языке представлены в исследованиях, проводимых в рамках неопозитивистского направления философии начала XX века. В частности, работы раннего Витгенштейна дают теоретическое обоснование для разработки формальных языков. Формальный язык, по мнению Л. Витгенштейна, не должен допускать многозначности высказываний, но должен иметь логический синтаксис (т.е. синтаксис предложения должен быть жестко связан с семантикой). Также, предложения на формальном языке должны распадаться на «логические атомы». Наличие такого языка позволит избегать псевдовысказываний, т. е. высказываний, не имеющих процедуры их проверки.
5. Информатика в середине XX века формировалась под влиянием идей раннего Витгенштейна. Н. Винер и А. Тьюринг некоторое время были студентами Витгенштейна и использовали его философский подход для разработки информационной теории. Можно утверждать, что языки программирования во многом удовлетворяют тем требованиям к «идеальному языку», которые высказывались европейскими философами.
6. Критика лингвистического формализма в работах позднего Витгенштейна может быть перенесена на языки программирования. В частности, можно принять его положение о том, что формальный язык не позволяет «создавать новое», т.е. машина, «мыслящая» на таком языке неспособна к творчеству. Поздний Витгенштейн предложил новый подход к философии естественного языка - «теорию языковых игр». Этот подход уходит от попыток построить «идеальный язык», но концентрируется на понимании реального человеческого языка.
7. Современная информатика основывается на философских воззрениях об идеальном языке, имеющих длительную историческую традицию. Однако, все попытки создать такой язык наталкивались на проблему его ограниченной применимости в реальных жизненных условиях. Можно предположить, что новый виток развития информатики будет связан с переходом от построения идеальных языков к использованию естественных языков в качестве средств представления информации.

In searching of ideal language (the philosophical preconditions of appearance of formal languages)

A.N. Savostyanov

1. In European philosophical tradition there were the several attempts to create “ideal language”, i.e. the artificial language without “disadvantages” of natural languages.
2. The earliest idea of such language can be obtained from the Plato’s dialog “Cratylus”. According to this dialog, the thing's name should completely represent a thing’s essence. It means that if person knows the name, he knows all properties of thing, which is named by this name. According to the view of dialog’s participants, the ideal language was in the past, but it was deformed. The modern language (i.e. the language of Plato’s epoch) is just partly corresponding to such requirements. In this reason, the reconstruction of ideal language was presented as the returning to Golden century. However, the critical comments to ideal language conception also contained in the end of dialog. One of participants, Cratylus, told the point of view that changeable things cannot be described by words exactly. The representation of variable properties of world in the language leads to logical contradictions. In general, according to Plato’s position, the ideal language can be applied for description of the immutable eidos world, and just partly can be applied for description of mutable cosmos.
3. The conception of “universal language” was suggested in philosophical system of G. Leibniz. According to his view, such language can be used in scientific studies. Leibniz tried to delete from language all estimated statements. Probably, it was related with attempts to avoid the disputed discussions. The universal language should give opportunity to avoid the discussions between philosophers. Any problem, which is formulated in this language, will be solved by means of computing. Leibniz suggested the theory of computing, which was one of predecessors of modern informatics. However, in the later part of his life Leibniz made conclusion, that universal language cannot be used for description of human behavior, because behavior depends on subjective estimations of events.
4. In the first half of XXth century, the conception of ideal language was presented in the studies of neo-positivism philosophers. In particular, the studies of “earlier” L. Wittgenstein give theoretical basis for elaboration of formal languages. According to Wittgenstein’s view, formal language should not include ambiguous sentences. Such language should have “logical syntaxes” (i.e. the syntactical structure of sentence should be hardly related with semantic structure). Also, the sentences of formal language should be decomposed into “logical atoms”. If we will have such language, we can avoid “pseudo-sentences”, i.e. the sentences, which have not the procedure of verification.
5. In the 40th years of XXth century, the ideas of “earlier” Wittgenstein influenced to formation of informatics. Norbert Wiener and Alan Turing attended lectures of Ludwig Wittgenstein and used his philosophical approach for elaboration of informatics. It is possible to suggest, that the computer languages in strong degree correspond to requirements of “ideal language”, which were formulated by European philosophers.
6. The criticism of linguistic formalism in the studies of “later” Wittgenstein can be transferred to programming languages. In particular, it is possible to accept his idea that the formal language doesn’t allow “to create new”. The machine, which is “thinking” in such language, is unable to creation of new ideas. “Later” Wittgenstein suggested new approach to philosophy of language – “theory of linguistic games”. This approach doesn’t try to create “ideal language”, but focused on understanding of real human language.
7. The modern informatics is based on the philosophical conceptions of ideal language, which have long-term history. However, all attempts to create such language have some limitations in application of ideal language in real-life conditions. It is possible to make hypothesis, that new period of development of informatics will be related with transition from elaboration of ideal languages to using of natural languages as means of information representation.

Вопросы создания прикладных лингвистических онтологий*

Е.А.Сидорова

Институт систем информатики им. А.П. Ершова
630090, Новосибирск, пр. Лаврентьева 6
lena@iis.nsk.su

В работе описывается лингвистическая онтология, используемая для анализа текстов на естественном языке в ограниченной предметной области. Особенностью предложенного подхода является включение в онтологию помимо классических словарей и тезауруса дополнительных компонент, предназначенных для распознавания контекста терминов в тексте и последующего связывания найденных элементов с понятиями и отношениями предметной области. Также рассматриваться лингвистические задачи, решаемые в процессе создания информационных систем, использующих сервисы анализа текста для автоматического наполнения и изменения содержания системы. Исследуется роль знаний, представленных онтологией информационной системы, в процессе обработки текста, т.е. как именно знания используются и влияют на извлекаемую из текста информацию и как именно результаты, полученные при обработке текста, влияют на сами знания.

Введение

Когда мы говорим о лингвистической онтологии, подразумевается, что речь идет о естественном языке, о его устройстве и то, как он связан с окружающим нас миром. В практических задачах рассматривается ограниченная часть мира или предметная область (ПО) и соответствующий данной области ограниченный язык.

Главной характеристикой лингвистических онтологий (ЛО) является то, что эти онтологии связаны со значениями языковых выражений (слов, именных групп и т.п.). Существуют несколько способов организации взаимодействия между знаниями о предметной области и языке [Нариньяни, 2002], из которых можно выделить следующие два подхода.

Первый подход, принятый в лингвистике, заключается в создании лексических онтологий или тезаурусов (например, WordNet), которые охватывают большинство терминов ПО, и одновременно имеют онтологическую структуру, проявляющуюся в отношениях между терминами. Этот подход основан на онтологической гипотезе Сепира-Уорфа, то есть то, что не описывается словами, не может быть отражено в онтологии.

Второй подход четко разделяет онтологию и словарь (например, Микрокосмос, OntoSem). Онтология, по мнению приверженцев данного подхода, должна быть максимально независимой от конкретного языка, в то же время понятия онтологии должны иметь свое непосредственное отражение в языковых значениях [Добров и др., 2006]. Словарная статья термина в таком подходе может иметь и связь с понятием онтологии и особенности конкретной лексической единицы.

Несмотря на все удобства последнего подхода, следует отметить, что понятия онтологии часто невозможно связать «напрямую» с элементами словаря, как правило, они выражаются более сложным образом: словосочетаниями или фразами, части которых могут быть разнесены по тексту. Это связано с проблемой лексической многозначности слова [Поляков, 2004] и требует для уточнения исследования контекста. Знания, связанные с описанием возможных контекстов словарных единиц, должны стать неотъемлемой частью лингвистической онтологии. Таким образом, выделяется промежуточный элемент между словарем и онтологией предметной области.

В данной статье рассматривается подход к построению лингвистической онтологии, который достаточно полно отражает процесс связывания словаря и онтологии. По принципу «разделяй и властвуй» выделяются максимально-независимые компоненты ЛО, что с технологической точки зрения дает все преимущества модульного подхода создания программных систем: независимая и распределенная реализация, повторное использование, прозрачность, легкость расширения и т.п.

Отметим, что мы будем рассматривать те прикладные лингвистические онтологии, которые используются в информационных системах для автоматической обработки текста. Назначение таких онтологий — обеспечить сопоставление естественно-языковым выражениям, встретившихся в тексте, элементов онтологии предметной области. Параллельно будут рассматриваться все знания, хранимые в информационной системе (ИнС), и то, какую роль они играют в процессе обработки текста.

* Работа выполнена при финансовой поддержке РФФИ (проект № 09-07-00400) и Президиума РАН (Интеграционный проект СО РАН № 2/12 в рамках программы фундаментальных исследований РАН № 2).

1. Онтология информационной системы

Современным подходом к представлению знаний информационной системы является онтология [Хорошевский и др., 2001], основное назначение которой состоит в том, чтобы облегчить пользователю поиск информации в большом наборе ресурсов путем структурирования и систематизации знаний, создания единой иерархии понятий, унификации терминов. Онтология ИнС наряду с традиционным описанием предметной области содержит соотнесенное с ним описание структуры и типологии соответствующих хранилищ данных и сетевых ресурсов [Васильев и др, 2003; Боровикова и др, 2002].

На этапе разработки ИнС онтология играет важную роль при анализе требований и концептуальном моделировании, особенно, если она интегрируется с лексическими ресурсами, используемыми для поддержки анализа текстов на естественном языке. В задачах извлечения информации из текстов с целью автоматического наполнения информационного пространства системы [Рубашкин, 2006] онтология ИнС играет определяющую роль (можно говорить, что анализ осуществляется под управлением онтологии).

Онтология ИнС рассматривается как трехуровневая система знаний:

- метаонтология описывает структуру предметной онтологии, т.е. фиксирует те структурные элементы, с помощью которых будут представляться понятия предметной области, обычно к ним относят такие элементы как: понятие (класс), иерархическое отношение наследования между классами, бинарное отношение (ассоциативное, часть-целое), тройка атрибут–тип–значение и т.п.;
- онтология предметной области описывает предметную область ИнС в терминах понятий и отношений;
- онтология нижнего уровня включает экземпляры понятий и отношений, образующих информационное наполнение ИнС.

В данной статье онтология ИнС будет рассматриваться с точки зрения того, как именно знания, хранящиеся в системе, используются и влияют на извлекаемую из текста информацию и как именно результаты, полученные при обработке текста, влияют на саму онтологию.

1.1. Прикладные лингвистические задачи

Разработка онтологии ИнС – это обязательно итеративный процесс, поэтому можно рассматривать жизненный цикл онтологии в контексте использования средств анализа текста для ее развития. Под жизненным циклом онтологии понимается непрерывный процесс, который начинается с момента принятия решения о необходимости создания онтологии и заканчивается в момент полного прекращения ее поддержки. Таким образом, жизненный цикл онтологии, как неотъемлемая часть жизненного цикла самой системы, охватывает все стадии и этапы ее создания, сопровождения и развития:

1. Формирование онтологии верхнего уровня (проектирование) – на данном этапе вводится метаонтология, а также основные понятия предметной области;
2. Формирование онтологии предметной области (ПО) – на данном этапе описывается предметная область в терминах понятий и отношений, формируются иерархии понятий, фиксируется атрибутная структура понятий и ассоциативных отношений, создаются списки доменных значений;
3. Добавление справочных знаний – к справочному знанию относятся, например, расшифровка обозначений, используемых на производстве, или заранее известный список объектов строительства в строительной области, эти знания представляются в виде экземпляров понятий и отношений онтологии ПО и поэтому их можно отнести к онтологии нижнего уровня. Еще один вид справочной информации – толкования понятий ПО; возможность представления такой информации должна предусматриваться на этапе проектирования.
4. Наполнение онтологии нижнего уровня, т.е. добавление в систему экземпляров понятий и отношений онтологии;
5. Эксплуатация онтологии и ее сопровождение, на данном этапе поступающая в систему новая информация должна согласовываться с уже имеющейся;
6. Расширение онтологии системы – этот этап наступает тогда, когда знания в системе «устаревают» и не устраивают пользователя в том виде, в котором они представлены.

Отметим, что задачи решаемые сервисом анализа текста на разных этапах, на самом деле различны, даже типы или жанры документов, с которыми приходится работать могут быть неодинаковы.

Онтологический анализ ПО (этап 2) обычно начинается с создания словаря терминов, который используется при обсуждении и исследовании характеристик объектов и процессов, составляющих рассматриваемую ПО, также выделяются основные логические взаимосвязи между понятиями, которые соответствуют введенным терминам. Результатом этого анализа является онтология ПО системы. Таким образом, встает задача автоматического извлечения предметной терминологии, которая включает как однословные, так и многословные термины [Лукашевич и др, 2008; Сидорова, 2008]. Для решения этой задачи используется подборка текстов по данной тематике (жанр текстов для данной задачи практически не имеет значения). Далее, используя различные методы кластеризации (как, например, в [Крижановский, 2006]) можно автоматизировать построение иерархичных отношений между терминами, а также сформировать списки синонимов для дальнейшего анализа. Ус-

ловием применения таких методов является наличие обучающего корпуса текстов, специальным образом размеченного.

На третьем этапе рассматривается задача автоматического добавления справочной информации в базу знаний системы. Знания, хранимые в справочниках, должны иметь естественную проекцию в онтологии ПО. Справочные ресурсы – это, как правило, хорошо структурированные тексты. Использование формальной жанровой модели таких ресурсов может значительно упростить процесс анализа текста, а также ускорить его настройку. Еще одной особенностью данного типа ресурсов является наличие в тексте значимых несловарных единиц, выражаемых буквенно-числовыми конструкциями (например, *5 м/с*, *103-105 км, корпус 2а* и т.п.). Извлечение таких единиц возможно на основе шаблонов, описываемых на специальном языке, см. [Жигалов и др, 2002].

Четвертый этап является основным приложением средств, предлагаемых сервисом анализа текстов. Задачей данного этапа является извлечение значимой для системы информации. Онтология ПО определяет формат данных, хранимых в ИнС и, следовательно, определяет, какую именно информацию необходимо извлекать из текста документа, а какую можно проигнорировать. Результат анализа документа представляется в виде семантической сети объектов, являющихся экземплярами понятий и отношений, заданных онтологией предметной области. Данная семантическая сеть добавляется в базу данных ИнС и, таким образом, преобразуется в знания, которыми в дальнейшем может оперировать система.

Когда в системе накоплено достаточное количество данных возникает ситуация, когда при поступлении новой информации, ее требуется согласовывать с уже имеющейся в системе (этап 5). С другой стороны, присутствует обратная связь – знания накопленные системой могут помочь разрешить неоднозначные ситуации в тексте. На данном этапе возникает много интересных задач, часть которых будут рассматриваться в данной статье: отождествление информационных объектов (обозначающих один и тот же объект действительности), поиск объекта в базе данных системы по неполному набору характеристик (критерии выборы наиболее подходящего объекта), изменения свойств объектов БД (в связи с естественным изменением характеристик обозначаемого объекта в реальном мире), поддержка актуальности данных, обработка противоречивых данных и т.п.

Внесение изменений в онтологию ПО (а также, в онтологию верхнего уровня – этап 6) возможно либо при изменении требований к системе со стороны пользователя, либо при накоплении достаточного количества фактов сигнализирующих о наличие неполноты в системе описания ПО. Данные факты могут извлекаться из текста по специальным правилам с обязательным требованием высокой точности.

1. 2. Лингвистические знания и ресурсы

Очевидно, что знаний о предметной области, хранящихся в онтологии, недостаточно для автоматического извлечения информации из текста – требуются дополнительные знания о языке, на котором эта информация представляется. Рассмотрим подробнее, какие знания могут использоваться при анализе текста.

Минимальной единицей анализа является словоформа (слово), число или специальная символьная конструкция. Каждая единица идентифицируется – слово ищется в словаре с учетом его парадигмы, для прочих символьных выражений определяется их тип. Процесс идентификации слов осуществляется на основе знаний, хранящихся в словарях. Дальнейшая обработка заключается в последовательном анализе контекста найденных лексем (и других минимальных единиц). Можно выделить следующие типы контекста:

- Устойчивое словосочетание (словокомплекс), характеризующееся наличием определенной синтаксической связи между контактно расположенными словами и высокой частотностью в анализируемом подъязыке, – для выявления контекста данного типа обычно составляются словари, которые содержат знания, необходимые для идентификации словокомплексов в тексте;
- Фраза – к данному типу относятся неустойчивые словосочетания, для которых не задан строгий порядок слов, не частотные в анализируемом подъязыке, а также фразы, которые формируются нерегулярным образом: могут быть разрывными, используют различные элементы языковой отсылки (анафорические и дейктические выражения;
- Множество связанных фраз – данный контекст характеризуется определенным линейным порядком связанных частей, использованием различных видов языковой редукции (сочинение и другие виды эллипсиса, анафора) и позволяет извлекать из текста информационные объекты со сложной структурой и связи;
- Текст – важным свойством данного контекста является жанр текста. Жанр играет важную роль в определении формальной структуры создаваемого автором текста, в которой, помимо логического уровня, т.е. уровня разбиения на предложения, абзацы, параграфы и т.п., выделяется уровень концептуальный, или собственно жанровый (наличие в тексте таких разделов, как заголовок, резюме, основной текст) [Жигалов и др, 2002];
- Ранее обработанные тексты – знания, извлеченные из ранее обработанных документов, хранятся в базе знаний ИнС (как альтернатива, они могут включаться в разметку корпуса текстов, используемого в задачах автоматического обучения).

Таким образом, для поддержки всех способов обработки текста могут потребоваться следующие лингвистические ресурсы:

1. Корпус текстов – подборка текстов определенного жанра, тематика которых соответствует заданной предметной области. Корпус может содержать лингвистическую разметку, представляющую собой ин-

формацию, полученную автоматически при анализе текстов, либо приписанную экспертом вручную. Основное назначение корпуса – автоматизация создания других лингвистических ресурсов.

2. Словник или перечень минимальных единиц языка, используемых при описании значимой для ИнС информации.
3. Набор специфичных для заданного языка лингвистических знаний: морфологические классы (определение лексемы для найденной в тексте словоформы), правила согласования терминов языка и т.п. Эти знания могут быть заданы с разной степенью подробности в зависимости от требований и возможностей разработчиков ИнС.
4. Словарь словокомплексов.
5. Знания о согласовании имеющихся лингвистических знаний с предметными знаниями, заданными онтологией ИнС. С этой целью термины группируются в семантические группы (или им приписываются семантические ориентации), которые в свою очередь также согласуются с элементами онтологии либо непосредственно, либо в соответствии с определенной схемой.
6. Набор описаний жанровых структур текста, соотнесенных с тем или иным типом текстовых ресурсов, хранящихся в ИнС.

Дополнительно может рассматриваться коммуникативно-прагматический контекст, и в этом случае в качестве лингвистического ресурса может фигурировать проблемная модель, зависящая от цели, которую преследуют разработчики системы анализа текста (например, модель коммуникации для вопросно-ответных диалоговых систем).

Предложенный набор лингвистических ресурсов предполагает схему анализа, описание которой можно найти в [Сидорова, 2007]. В данной схеме рассматривается только одна проблемная модель – модель извлечения фактов для фактографического анализа текста и сопоставления фактов элементам онтологии. Эта модель может считаться базовой для всех остальных задач и может быть расширена дополнительными компонентами.

2. Лингвистическая онтология

Под лингвистической онтологией далее понимается вся совокупность лингвистических и экспертных знаний, необходимая для анализа текста на естественном языке. Дадим формальное определение.

Лингвистическая онтология, для которой задана онтология O и ее информационное наполнение O_I [Васильев и др, 2003], которое содержит экземпляры понятий и отношений онтологии O, определяется пятеркой вида $<V, W, T, F, D>$, где

V – словарь, включающий минимальные единицы текста – лексемы и лексические конструкции,

W – словарь устойчивых словосочетаний (словокомплексов),

T – семантический словарь (тезаурус), который устанавливает классические тезаурусные отношения между элементами словарей V и W,

F – множество упорядоченных наборов схем фактов,

D – множество моделей документов, для каждой из которых может быть определен собственный набор схем фактов.

Рассмотрим подробнее эти компоненты.

2.1. Словарь-Тезаурус

Словарь можно отнести к лингвистической онтологии лишь условно. Словарные единицы или термины в той же степени относятся к ЛО, что и экземпляры понятий к онтологии ПО. Интерес представляет онтологическая информация, т.е. то, какие признаки приписываются терминам и как термины связаны между собой. Поскольку вся эта информация в рамках нашего подхода хранится в словарной статье термина, а относится она как непосредственно к словарю, так и к тезаурусу, то удобно дать интегральное определение Словаря-Тезауруса.

Словарь-Тезаурус – это объем лексики, организованной по тематическому (семантическому) принципу с отражением определенного набора базовых семантических отношений. Наличие групп, объединяющих в себе множество синонимических выражений одного понятия, с одной стороны, и объектная ориентированность словаря, с другой – позволяют словарю выполнять функции тезауруса.

Можно дать определение Словаря-Тезауруса аналогично определению онтологии с выделенными морфологическими типами и синонимическими отношениями, где базовыми понятиями являются термины из словарей V и W нашей лингвистической онтологии.

Таким образом, словарь-Тезаурус – это знаковая система

$<V, W, C_T, M, G_{syn}, R_T>$, где

$V \cup W$ – конечное множество терминов (лексем, словокомплексов),

C_T – конечное множество семантических и тематических признаков,

M – конечное множество морфологических классов,

$G_{syn} = \{G_1,\ldots,G_k\}$ – конечное множество подмножеств терминов, связанных отношением синонимии (синсеты),

$R_T \subseteq (V \cup W) \times (V \cup W)$ – конечное множество бинарных отношений $r_i(t_x, t_y)$, $t \in (V \cup W)$ между терминами, выделяются следующие типы отношений:

- отношение «общее-частное», антисимметричное, транзитивное, нерефлексивное бинарное отношение, задающее частичный порядок на множестве терминов,
- отношение «часть-целое», антисимметричное, транзитивное бинарное отношение на множестве терминов,
- конечное множество дополнительно определяемых пользователем отношений (например, антонимия).

Технология, позволяющая создавать такие тезаурусы, описана в [Сидорова, 2008].

Рассмотрим, какие элементы должны присутствовать в словаре на примере описания ФИО человека – *Андрей Петрович Ершов*. Словарь *V* содержит:

Ершов: t∈M {часть речи: сущ, одушевленность: од, род: мр, имя собственное: фам},
Андрей: u∈M {часть речи: сущ, одушевленность: од, род: мр, имя собственное: тип имени},
Петр: // парадигма мужского имени содержит отчество
u∈M {часть речи: сущ, одушевленность: од, род: мр, имя собственное: тип имени}.

Словарь W может содержать словокомплекс:
Ершов А П: Персона∈C_T,
связан отношением *включает* с:
лексемой *Ершов*,
лексической конструкцией *А.П.*
входит в синсет *Андрей Петрович Ершов* (т.е. синонимичен терминам:
А П Ершов, Андрей Петрович Ершов, Ершов Андрей Петрович).

2.2. Модель фактов

Факт представляет собой высказывание из предметной области, имеющее определенную синтаксическую и семантическую структуру, базовые элементы которой хранятся в словаре. Для описания модели факта (далее схема фактов) определяется упорядоченный набор $< t_1,\ldots,t_n >$ терминов, которые должны удовлетворять определенным условиям или ограничениям. Факт, покрывая часть текста, связывает термины в более сложные структуры, которые также могут быть использованы при описании других схем фактов. Одним из способов описание структуры значимых фактов и их связей с онтологией является декларативное представление схем фактов.

Схема факта – это тройка вида $< Arg, Res, C_F >$, где

Arg – множество аргументов факта, где аргумент может быть задан термином или семантическим признаком словаря, а также типом объекта, получаемого с помощью другой схемы фактов.

$Res = < c_i, op(c_i), P >$ – результат применения схемы факта, где

c_i – задает тип результирующего объекта,

$op(c_i)$ – тип операции, применяемой, если все условия схемы факта C_F выполнены (создание объекта или редактирование аргумента),

P – множество правил для формирования значений атрибутов результирующего объекта. Каждое правило ставит в соответствие атрибуту результирующего объекта один из следующих элементов:

- значение по умолчанию,
- значение атрибута одного из аргументов,
- значение, заданное таблично в таблице сочетаемости *Sem*.

Для того, чтобы применить схему факта ее аргументы должны удовлетворять заданным ограничениям на сочетаемость аргументов. Выделяются семантические и структурные ограничения.

$C_F = < Sem, St >$, где *Sem* – набор семантических ограничений, *St* – множество структурных ограничений. Семантические ограничения накладывают условия на атрибуты аргументов факта, а также могут задаваться таблично, определяя сочетаемость семантических характеристик [Кононенко и др., 2002]. Структурные ограничения накладывают условия на взаиморасположение элементов факта в тексте и их характер. К структурным ограничениям, в частности, относятся жанровые ограничения и условия на синтаксическую сочетаемость элементов.

Если в качестве аргументов факта выступают термины словаря, то такая схема предназначена для исследования контекста 2-го типа. Если аргумент – это результат применения другой схемы, то речь уже идет о контексте 3-го типа. Например,

*arg*1: персона (термин) + *arg*2: звание (термин) : ‘контактно’∈*St*
-> Персона (ФИО: *arg*1.*name*, звание: *arg*2.*name*)

контекст 2-го типа, а более сложная структура:

*arg*1: Персона (объект) + *arg*2: Организация (объект): ‘в одном сегменте’∈*St* -> Сотрудник (кто: *arg*1, работает_где: *arg*2)

контекст 3-го типа, т.к. извлекаются связи между информационными объектами.

2.3. Модель документа

Выраженную формальную жанровую структуру текста имеют многие документы: анкеты, деловые письма, научные статьи, программы конференций, информационные сообщения и т.п. Такую структуру можно формально представить с помощью иерархии сегментов. Сегмент – это фрагмент текста, удовлетворяющий условиям, которые определяют границы фрагмента и его структурную организацию (позицию относительно других фрагментов). Сегменты характеризуются либо “полиграфическими” элементами (абзац, строка и т.п.), либо определенной лексикой, и могут реализовываться в рамках формальных сегментов других типов.

Т.о. сегмент s – это тройка <*l*, *Mark*, *a*>, где *l* – уникальное имя сегмента, *Mark* – множество лексических маркеров (либо элемент словаря, либо произвольная строка, либо символ), *a* – свойство, определяющее порядок и правила инициализации сегментов.

Модель документа определяется набором сегментов, порядком их следования и вложенностью. Т.е. модель документа *D* – это тройка вида <*Seg*, R_S, R_I>, где

Seg – множество формальных сегментов,

R_S – отношение следования, заданное на множестве сегментов *Seg*,

R_I – отношение вложенности, заданное на множестве сегментов *Seg*,

Например, текст делового письма имеет следующие жанровые разделы:
заголовок: отправитель, адресат, резюме и обращение,
основной раздел: текст письма, примечания и приложения,
заключительный раздел: подпись, исполнитель и бланкодержатель.

3. Внедрение информации в пространство системы

При добавлении информации, полученной в результате анализа текста, в информационное пространство системы необходимо выполнить ряд операций, обеспечивающий корректность, уникальность, актуальность, непротиворечивость, целостность и согласованность добавляемых данных.

Добавляемая информация связывается с документом и образует его контент. Для того чтобы сформировать контент документа необходимо:

- обеспечить корректность и целостность информационных объектов, полученных в результате анализа;
- идентифицировать объекты, образующие контент;
- обеспечить актуальность добавляемой информации.

3.1. Корректность объектов

Объект считается корректным, если для него определены обязательные атрибуты класса и значения атрибутов принадлежат соответствующим им доменам.

Корректность значений атрибутов объектов обеспечивается на уровне словаря и правил извлечения фактов. На этапе конструирования словаря эксперт должен либо внести термины, обозначающие доменные значения в словари, либо предусмотреть возможность создания объектов (из других словарных терминов), атрибуты которых позволят воссоздать название доменного значения. На этапе анализа в результате применения соответствующего правила термины преобразуются в строковые (доменные) значения атрибутов.

Поскольку описание одного и того же доменного значения в тексте может осуществляться с помощью различных терминов, то необходимо предусмотреть механизм сопоставления разных терминов одному значению. Для этого в словарной подсистеме определяется синонимичная группа (синсет) – набор терминов, альтернативных для данного значения.

Целостность информации — соответствие информации внутренней структуре онтологии и всем явно заданным аксиомам (правилам, ограничениям). Примеры таких аксиом: возраст персоны должен быть положительным и не больше 130; количество знаков в телефонном номере не должно превышать 15; возраст родителей не может быть меньше возраста их ребёнка и т.д. Задача создателя онтологии — возможно более полно выявить все имеющиеся ограничения и аксиомы.

Целостность информации не гарантирует ее достоверности, но обеспечивает по крайней мере правдоподобность этой информации, отвергая заведомо невероятные, невозможные значения.

3.2. Идентификация объектов

Объект считается идентифицированным, если для него определен класс и набор ключевых атрибутов данного класса. Данное свойство позволяет однозначно выделить данный объект из множества других объектов, т.е. обеспечивает его уникальность в БД системы. Отметим, что набор ключевых атрибутов включает все обязательные атрибуты класса.

В процессе идентификации объектов осуществляется уточнение полученных при анализе объектов (уточнение атрибутов объектов и их связей), "склеивание" одинаковых объектов, на основе использования локального и глобального контекста, а также поиск объектов в информационном пространстве системы.

Под локальным контекстом понимается содержание анализируемого документа, которое используется для решения следующих задач:

– определение референта для местоимений и неоднозначных именных групп;

– отождествление объектов, имеющих один и тот же референт (который обозначает один и тот же объект действительности), на основании частично совпадающего и непротиворечивого набора атрибутов и связей.

В данном случае уместно говорить о задаче разрешения анафоры, которая стала весьма популярной в области компьютерной обработки текста. Например, алгоритмы автоматического разрешения анафоры описываются в работах таких авторов, как Г.Хирст, Ш.Лаппин, Р.Митков, М.Поэсио и др.

Глобальный контекст представлен всем информационным пространством системы, в котором осуществляется поиск объекта. Он содержит как результаты ранее проанализированных документов, так и справочную информацию, вводимую пользователем вручную или поступающую из справочных ресурсов (на 3 этапе жизненного цикла онтологии).

Использование глобального контекста возможно в случае, когда набор ключевых атрибутов задан не полностью. В этом случае осуществляется поиск максимально похожего объекта в БД системы по известному набору атрибутов и классу объекта, а также его связям.

Предложенный способ заключается в построении фокусного множества объекта, найденного в тексте, которое включает все объекты, непосредственно связанные с данным, и сопоставлении его с фокусными множествами объектов, найденных в БД системы. Отметим, что в фокусное множество имеет смысл включать только уже идентифицированные объекты, найденные в БД.

Отметим еще две возможности использования глобального контекста.

Во-первых, уточнение класса объекта по иерархии понятий. Если объект с тем же набором ключевых атрибутов нашелся в БД с другим – более точным классом, то класс объекта, обнаруженного в тексте можно уточнить, а сам объект проассоциировать с найденным.

Во-вторых, идентификация и уточнение объектов с помощью восстановления иерархии по отношению "часть-целое" (иерархии вложенности понятий). Это применимо, например, в случае, когда объект подчинен другому объекту и имеет сложную структуру, представленную линейными цепочками наименований, совокупность которых образует дерево (множество деревьев) информационных объектов. Часто такие объекты являются параметризованными, что требует специальной обработки [Кононенко и др., 1987]. Эти объекты могут встречаться в справочниках, например, дороги, – каждый участок, если он не имеет уникального имени, идентифицируются верхним участком (*сибирское направление*) и числовым значением (*3333 км*) или именным участком (*станция Комаровка*) и расстоянием от него. Для идентификации такого объекта требуется восстановить иерархию вложенности объектов документа данного типа путем сравнения с эталонной иерархией объектов из БД. Каждая пара объектов, удовлетворяющая определенным требованиям порядка слов, проверяется на предмет наличия между ними отношения вложенности (с учетом транзитивности). Результирующими являются те объекты, которые соответствуют листьям полученных древесных структур.

3.3. Актуальность информации

Актуальность информации — это степень соответствия информации текущему моменту времени. Достоверность (или истинность) есть соответствие фактов реальной действительности. В какой-то степени новые документы отражают изменения, происходящие в реальном мире, и, поэтому, могут осуществлять контроль достоверности информации, хранящейся в системе.

Существует простой подход (с точки зрения здравого смысла), когда новая поступающая информация считается более актуальной, чем уже имеющаяся. Однако, когда мы имеем дело с системами автоматической обработки текста, нет никакой гарантии, что информация извлечена правильно, кроме того информация, поступающая из разных источников, зачастую противоречива, либо вообще представлена в виде гипотез или обрывочных данных (например, исторические сведения или текущие исследования, как постоянно обновляющиеся результаты работ). Поэтому все полученные конструкции являются предположениями, а не фактами.

Таким образом, необходимо учитывать, что если полученная из новых документов информация не согласуется с уже имеющейся, то либо информация в системе устарела (потеряла актуальность) и требует обновления, либо новая информация ошибочна.

Для решения этой задачи без потери полноты требуется привлечение статистики (как говорил Сегалович И.В., один из основателей Яndex: «Статистика может все»). Т.е. задача поддержки актуальности знаний в системе сводится к накоплению статистики изменения информации и, в соответствии с запросом пользователя, выдается либо самая достоверная на текущий момент информация, либо история ее изменений.

Для решения задачи необходимы структуры данных, которые могли бы накапливать статистику, сопряженную с временными характеристиками. Был предложен следующий подход. Каждый минимальный факт в системе (связь объекта с его атрибутом, точнее с одним из значений атрибута) снабжается структурой включающей следующие характеристики:

– дата начала (X_1) и дата конца (X_2) актуальности данного параметра в реальном мире,
– дата начала (Y_1) и дата конца (Y_2) актуальности данного параметра в системе (в какой промежуток времени система думала, что это так),
– набор источников или документов, в которых присутствует информация о данном факте.

В частности, факт переименования в онтологии отражается с помощью введения для соответствующих атрибутов множества имен с датами начала и конца их действия. Например,

struct Data(Data_begin: data, Data_end: data, Data_begin_loc: data, Data_end_loc: data);
struct Naming(Name: string, Время_действия: Data);

class *Персона* (Фамилия: Naming; Имя: string; Отчество: string; …)
class *Организация* (Название: Naming; Аббревиатура: Naming)

Можно рассмотреть следующие возможные ситуации.

а) $X(X_1,X_2)$ не определено – типичная ситуация когда в тексте ничего не сказано о времени действия факта. Параметр Y определяется датой документа, в котором факт встретился впервые для Y_1 и датой последнего документа для Y_2.

б) $Y>X$ в тексте указана информация о наступившем событии.

в) $Y<X$ в тексте указано будущее событие.

г) В тексте может быть указана косвенная дата, например, *во время Z состоялось событие, на котором обсуждались факты* 1,2… В этом случае косвенная дата выступает в качестве реальной $X_1 = X_2 = Z$ до тех пор, пока не появится уточняющая информация.

Ссылки на источник информации (обработанный документ) используют не только для контроля достоверности поступающих данных, но и для индексирования документа (множество всех ссылок связывает документ с его контентом).

3.4. Изменение ключевых характеристик информационного объекта

Изменение ключевых характеристик информационного объекта требует отдельного рассмотрения, т.к. оно влияет на уникальные относительно БД параметры объекта. К ключевым характеристикам относятся ключевые атрибуты (или альтернативные наборы ключевых атрибутов), задаваемые в онтологии ПО, и класс – понятие или отношение ПО.

Такое изменение возможно, только если это явным образом обозначено в тексте (в виде факта) и удалось с большой точностью выявить этот факт. Например, тот же факт переименования, отмеченный выше: *институт А был переименован …, институт А был объединен с институтом В, студент Х закончил университет и поступил на должность научного сотрудника в …, персона Х сменила фамилию* и т.п.

Механизм представления изменений ключевых атрибутов такой же, как и для простых атрибутов, но при поиске множество значений ключевого атрибута выступает как ИЛИ-множество – множество альтернатив, это позволяет, например, идентифицировать объект по любому его наименованию в прошлом.

Кроме того, упоминание в тексте объекта с «устаревшими» параметрами позволяет выдвинуть гипотезу о косвенной дате всех событий описанных в тексте (если нет указания явной даты).

Изменение класса объекта отслеживать труднее (отметим, что изменение класса нарушает концепцию объектной ориентированности), кроме того, в общем случае при изменении класса может нарушаться целостность объекта, происходить потеря данных. Поэтому на данную операцию накладывается ограничение – объект может лишь уточняться по иерархии классов или обобщаться (если нет потери данных) в случае, если предыдущее изменение принято за ошибочное.

Заключение

Модель знаний, используемая в процессе анализа текста включает такие знания как онтология предметной области, лингвистические знания о языке и жанровых особенностях анализируемых документов, знания и данные, накопленные информационной системой в процессе эксплуатации. Объем данной статьи не позволяет привести подробные примеры применения данного подхода, однако их можно найти в следующих источниках: использование схем фактов и модели документов жанра «деловое письмо» в системе документооборота InDoc [Загорулько и др, 2004], использование цепочки тезаурус – онтология для анализа текста для портала знаний по археологии и этнографии [Андреева и др., 2005], использование тезауруса, схем фактов и онтологии для анализа сообщений из архива «Хроники СО РАН» [Сидорова и др., 2009].

В статье подробно рассматривались вопросы внедрения информации, поскольку именно на этом этапе обеспечивает взаимодействие подсистемы анализа с контекстом самой ИнС и позволяет существенно расширить возможности анализа, связанные с уточнением и корректировкой как получаемой из документа информации, так и информации, хранящейся в самой системе.

Список литературы

[Андреева и др., 2005] Андреева О.А., Боровикова О.И., Загорулько Ю.А., Кононенко И.С., Сидорова Е.А. Коллекционер онтологической информации для портала знаний по археологии и этнографии // Информационные технологии в гуманитарных исследованиях. Вып. 9. – Новосибирск: НГУ, 2005. –С. 39–47.

[Боровикова и др, 2002] Боровикова О.И., Загорулько Ю.А. Организация порталов знаний на основе онтологий. // Труды международного семинара Диалог'2002 по компьютерной лингвистике и ее приложениям. – М.: Наука, 2002. – С.76–82.

[Васильев и др, 2003] Васильев И.А., Тузовский А.Ф. Структура системы управления знаниями // Труды международного симпозиума «Информационные и системные технологии в индустрии, образовании и науке. – Караганда: КарГТУ, 2003. – С. 286–288.

[Добров и др., 2006] Добров Б.В., Лукашевич Н.В.: Онтологии для автоматической обработки текстов: описание понятий и лексических значений // Труды международной конференции Диалог'2006 «Компьютерная лингвистика и интеллектуальные технологии». – М.: РГГУ, 2006. – С. 138–142.

[Жигалов и др, 2002] Жигалов В.А., Жигалов Д.В., Жуков А.А., Кононенко И.С., Соколова Е.Г., Толдова С.Ю. Система ALEX как средство для многоцелевой автоматизированной обработки текстов Труды международного семинара Диалог'2002 по компьютерной лингвистике и ее приложениям. – М.: Наука, 2002. – Т.2. – С. 192.–208.

[Загорулько и др, 2004] Загорулько Ю.А., Кононенко И.С., Костов Ю.В., Сидорова Е.А. Подход к интеллектуализации документооборота // Информационные технологии, № 11, 2004. –С. 2–11.

[Кононенко и др., 1987] Кононенко И.С., Першина Е.Л. Синтез числовых параметрических конструкций. // В Сб. научных трудов под ред. А.Е. Кибрика и А.С. Нариньяни. Моделирование языковой деятельности в интеллектуальных системах. – М.:Наука, 1987. – С. 220–256.

[Кононенко и др., 2002] Кононенко И.С., Сидорова Е.А. Обработка делового письма в системе документооборота // Труды международного семинара Диалог'2002 по компьютерной лингвистике и ее приложениям. – М.: Наука, 2002. – Т.2. – С. 299–310.

[Крижановский, 2006] Крижановский А.А. Автоматизированное построение списков семантически близких слов на основе рейтинга текстов в корпусе с гиперссылками и категориями // Компьютерная лингвистика и интеллектуальные технологии: Труды международной конференции Диалог'2006 "Компьютерная лингвистика и интеллектуальные технологии". – М.: РГГУ, 2006. – С. 297-302.

[Лукашевич и др, 2008] Лукашевич Н.В., Добров Б.В., Чуйко Д.С. Отбор словосочетаний для словаря системы автоматической обработки текстов // Компьютерная лингвистика и интеллектуальные технологии: Труды международной конференции «Диалог 2008». –М.: РГГУ, 2008. –С. 339–344.

[Нариньяни, 2002] Нариньяни А.С.: ТЕОН-2: от Тезауруса к Онтологии и обратно // Труды международного семинара Диалог'2002 по компьютерной лингвистике и ее приложениям. – М.: Наука, 2002. – Т.1. – С. 307–313.

[Поляков, 2004] Поляков В.Н. Использование технологий, ориентированных на лексическое значение, в задачах поиска и классификации // Проблемы прикладной лингвистики. Вып.2. Сборник статей/Отв. ред. Н.В. Васильева. М.:Азбуковник, 2004. С. 101–117. http://virtualcoglab.cs.msu.su/html/polyak.html

[Рубашкин, 2006] Рубашкин В.Ш. Семантический компонент в системах понимания текста // Труды десятой национальной конференции по искусственному интеллекту с международным участием КИИ-2006. – М.: Физматлит, 2006. – Т.2. – С. 455–463.

[Сидорова, 2007] Сидорова Е.А. Использование онтологии при извлечении информации из текстовых ресурсов // Труды IX международной конференции "Проблемы управления и моделирования в сложных системах". – Самара: Самарский Научный Центр РАН, 2007. – С. 455-461.

[Сидорова, 2008] Сидорова Е.А. Многоцелевая словарная подсистема извлечения предметной лексики // Труды международной конференции Диалог'2008 «Компьютерная лингвистика и интеллектуальные технологии». – М.: РГГУ, 2008. –С. 475-481.

[Сидорова и др., 2009] Кононенко И.С., Сидорова Е.А. Подход к извлечению фактов из текста на основе онтологии // Компьютерная лингвистика и интеллектуальные технологии: По материалам ежегодной Международной конференции «Диалог 2009». Вып. 8 (15). М.: РГГУ, 2009. –С. 451-457.

[Хорошевский и др., 2001] Гаврилова Т.А., Хорошевский В.Ф. Базы знаний интеллектуальных систем. Учебник. СПб.:Питер, 2001.

Онтология и планирование текстов для генерации описаний изображений в среде DEMLinG

Е.Г. Соколова[1], М.В. Болдасов[2]

[1] Российский государственный гуманитарный университет, Москва, Россия
[2] Российский государственный гуманитарный университет, Москва, Россия
eg.sokolova@gmail.com, boldasov@nm.ru, boldasov@gmail.com

В статье излагается ход исследований по созданию онтологии для системы генерации текстов – описаний изображений. В качестве источника знаний о содержании изображений рассматриваются тексты экспериментального корпуса описаний изображений. Исследование проводится в экспериментальной среде DEMLinG. Описываются составляющие модели содержания изображения, эксперимент с системой SemTalk2 и эскиз онтологии DEMLinG. Обсуждаются две основных составляющих генерации текста – создание плана текста и введение предикатов.

1 Введение

В последнее десятилетие задача поиска в ресурсах, содержащих изображения, стала актуальной в связи с появлением в Интернете большого количества таких мультимедийных ресурсов, в частности, поиск фотографий, фрагментов кинофильмов, а также появление программ-собеседников, обсуждающих изображения. Задача распознавания решается в четырех направлениях - распознавание: а) конкретных людей по лицам; б) конкретных предметов, например, Собор парижской Богоматери; в) объектов определенного типа, например, танк на местности; г) типов сцен, например, «офис», «на кухне», например, (Reineking et al., 2009). При распознавании конкретных объектов могут использоваться пары «изображение - текст». В этом случае рассматриваются специальные виды текстов - подписи под фотографиями, короткие газетные заметки с фотографиями, описания именованных объектов, представляющих культурный и политический интерес, из Википедии и других Интернет источников (Gornostay, Aker, 2009). Изображение объекта и его название (имя) могут сопоставляться путем вычисления выделенности объектов в тексте и на изображении (Deschacht, Moens, 2007). Все перечисленные направления действуют методами искусственного интеллекта без использования знаний.

Исследование в данной статье связано с распознаванием композиции объектов в изображенных связных сценах и направлено на решение задачи автоматической генерации текстов, описывающих содержание таких изображений. При этом в процессе распознавания и генерации текстов предполагается использование знаний. Фокусом интереса является использование онтологии как компонента знаний при распознавании и планировании текста. В статье затрагивается только вторая часть – генерация текстов из искусственно созданных входных представлений, содержащих информацию об изображенных объектах и композиции изображения. Задача распознавания изображений в данной работе не рассматривается.

Подход основывается на нашем опыте по генерации текстов двумя разными методами: а) метод, основанный на системно-функциональной грамматике М.Хэллидея, реализованный в проекте AGILE (Kruijff et al., 2000) с нашим участием для генерации текстов инструкций, и б) трансформационный метод, который мы развивали в собственной программно-компьютерной среде DEMLinG (Болдасов М.В., 2003) для следующих типов текстов: запросы к БД (Boldasov, Sokolova, 2002), статистические отчеты из БД и описания фотографий, (Соколова, Болдасов, 2007).

В проекте AGILE использовалось две онтологии – Upper-Model http://www.fb10.uni-bremen.de/anglistik/langpro/kpml/um89/um89-root.htm, классифицирующая типы значений слов ЕЯ и использующаяся как посредник между моделью предметной области (ПО) и системой генерации текстов, и модель ПО для области графических редакторов, которая использовалась в виде всплывающих меню для выбора понятий пользователем в интерфейсе при построении входного представления для системы генерации. Фактически последовательность фраз текста и вся необходимая информация для генерации содержалась во входном представлении. Этап планирования в системе AGILE позволял получить только небольшие стилистические вариации и разную полноту содержания генерируемых текстов. Хотелось развести предметное и текстовое представления. Наш трансформационный метод позволял определить процессы преобразования структур знаний и лингвистических структур. Эксперименты со средой DEMLinG показали, что можно достичь приемлемых результатов на этапах синтаксического и морфологического синтеза, форматирования, но для планирования текста трансформационный метод мало пригоден из-за разнообразия возможных входных предметных представлений. Таким образом, в данной статье мы обсуждаем систему, в которую мы вернули знания в виде онтологии для использования, в частности, при планировании текстов.

Статья представляет собой изложение хода исследований и состоит из следующих разделов: В р.2 мы представляем экспериментальный корпус описаний изображений и тексты описаний изображений, которые служат источником информации о содержании изображения; в р. 3 описываются элементы формализованного пред-

ставления содержания изображения; в р. 4 мы описываем эксперимент в системой SemTalk2, проведенный в поисках реализации онтологии для нашей системы; в р. 5 кратко представлена создаваемая онтология среды DEMLinG, в частности, ее содержания для генератора описания изображений; в р. 6 рассматриваются вопросы планирования текста.

2 Экспериментальный корпус описаний изображений

В наших экспериментах «Модель содержания изображения» (МСИ) создавались вручную. В качестве материала взяты цветные фотографии из коллекции С.М. Прокудина-Горского, выставленные на сайте в Интернете http://www.prokudin-gorsky.ru/. Поскольку единственным объективным источником знаний о том, как должны выглядеть МСИ, являются тексты описаний фотографий на ЕЯ, нами был создан экспериментальный корпус описаний изображений, в котором каждому изображению сопоставляются тексты, описывающие его содержание, выполненные разными людьми. В корпусе описываются цветные фотографии России из коллекции С.М. Прокудина-Горского, сделанные автором в 1900-1917 годах при финансовой поддержке царя Николая II, выставленные на сайте www.prokudin-gorsky.ru. Фотографии характеризуются статичностью и определенным масштабом – в основном общие виды местности или съемки стоящих людей или отдельных крупных предметов. Можно выделить следующие виды фотографий:

- общий план видовые - улица, аллея, гора, часть портовой зоны, поле и т.п.,
- средний план фотографии отдельных объектов - церковь, часовня, дом, катер и т.д.), человек или группа людей,
- средний план жанровые фотографии, например, шашлычная, крестьяне в поле,
- крупный план отдельные объекты – цветущий куст, икона, витраж;
- крупный план множественные объекты - кувшинки на пруду, васильки в поле.

В данной статье мы рассматриваем фотографии общим и средним планом.

Тексты описаний созданы студентами 4-го курса РГГУ. Корпус текстов содержит около 250 описаний для 100 фотографий, так как большинство фотографии имеет два (в редких случая три-четыре) описания, сделанные разными людьми. Вот пример изображения – фотография 01081 «Девятины» и текста-описания к нему:

Рис. 1

На фотографии изображен вид села в ясную погоду. Небо голубое, есть небольшие облачка. Рельеф местности холмистый. На переднем плане часть луга, по которому протекает фиолетовая речка. Через речку проходит мост, укрепленный камнями и досками. Под мостом, вероятно, находится плотина - мы видим, что уровень воды понижается, вода бурлит и пенится. У реки видны две деревянные избы слева. Рядом с избами большие огороды, огражденные заборами. В ближнем к нам огороде стоят 2 дерева, слева ещё край дерева. За мостом ещё одна небольшая изба. На горизонте справа ещё изба. Слева на горизонте виднеется 1 пятиглавый собор, слева от него одноглавый храм, слева от него звонница с шатровой крышей. Все объекты храмового комплекса белого цвета, крыши и купола зеленоватые.

Тексты в корпусе имеют следующие особенности:

1. Это спонтанные описания без заранее подготовленного плана, поэтому они отражают движение внимания автора, следующего определенной стратегии осмотра изображения. Такие стратегии, называемые когнитивными стратегиями, рассматривались, например, в работе (Кобозева, 1997).
2. Часто только часть текста непосредственно связана с изображенными объектами, другая часть текста передает фантазии и эмоции автора, переплетенные с информацией, почерпнутой автором из заголовка фотографии – название места, имя, профессия изображенного человека и т.п. В вышеприведенном тексте таких фантазий автора нет;
3. Обычно для одного изображения имеется два описания, сделанных разными людьми, что позволяет сравнивать мнение двух, иногда 3 – 4 людей в вопросе о том, что и как изображено на фотографии. Второе описание данной фотографии см. в р.6.

3 Элементы модели содержания изображений

Онтологии для описания изображений применялись, например, (Hollink et al., 2003). При этом авторы опирались на существующие базы данных и лексико-семантические базы – WordNet, БД по биографиям художников и др. Особенность таких описаний состояла в том, что по традиции компьютерной лингвистики и описания представлялись как совокупность языковых предикатов с местами, в частности, триад, обозначающих действие с актантами, типа (agent-process-object) и его атрибутов (settings) - (time, location). Такие описания предложены в (Tam et al., 2001) и использованы в (Schreiber et al., 2001), (Hollink et al., 2003). Для наших фотографий такие описания содержания невозможны по двум причинам:

1. Объекты реальности присутствующие на фотографиях в виде образов, например, «дом», «человек», «поле», занимают определенные фрагменты поверхности фотографии. На них можно прямо указывать, их свойства - цвет, форму, яркость, размер, отображать в описании. В отличие от них предикаты могут появиться в описании только на основе косвенной информации о свойствах объекта, например, позы живого существа (*человек сидит, бежит*, …), об отношениях между объектами, в частности, местоположении объектов (*у реки стоит дом, на поле находится стог)* и др. В названиях фотографий на сайте языковые предикаты встречаются редко и обозначают состояние, например: «Плоты, сидящие на перекатах у деревни Курья».
2. Существующие базы знаний и базы данных, в частности, основанные на онтологиях, не предназначались для описания визуальных данных, поэтому не содержат важной для интерпретации изображений информации – зрительный образ, ориентация в пространстве, например, *столб* – объект с вертикальной ориентацией, и др., а содержат мало применимую для этой задачи энциклопедическую информацию, например, «дверь» в WordNet толкуется через понятие «барьер»: «DOOR is-a “movable barrier (a barrier that can be moved to allow passage)».

Основой наших моделей содержания изображений мы считаем виртуальную матрицу пикселов, на которой объекты определяются по их границам на плоском изображении и распознаются по их зрительным образам. Плоское изображение заключает композиционные отношения объектов, если один объект находится на фотографии внутри границ другого объекта, например, *крыльцо* входит в границы изображения *дома*. Если границы объектов на изображении не вкладываются один в другой, мы говорим об отношениях пространственной ориентации, например, «*дерево находится около/справа от дома*». В тех случаях, когда границы объектов пересекаются, мы говорим об отношении локализации, например, «*дерево растет на берегу реки*» или о том, что один объект находится позади другого. Таким образом, для представления знаний мы нуждаемся в двух видах исходных отношений: отношениях композиции и отношениях пространственной ориентации объектов друг относительно друга включая отношение локализации.

Мы стремимся отразить в МСИ преимущественно визуальную информацию и основанные на ней отношения между объектами. Уровень детализации описания можно установить так: мы упоминаем те объекты и части объектов, о которых можем или хотим сообщить некоторые подробности, если не хотим, тогда не упоминаем. Например, мы упоминаем голову человека, если хотим сказать, что на голове – шляпа и т.п. Инвентарь понятий для МСИ задается в онтологии объектов, которая определяет как элементы, выделяемые на плоском изображении, так и онтологические знания о свойствах объектов и возможных отношениях между ними. Эксперимент по созданию такой онтологии описывается в следующем разделе.

4 В поисках средства реализации онтологии: эксперимент с Semtalk2

Мы стремились использовать язык OWL (Web Ontology Language) как специально созданный для редактирования онтологий. Он базируется на парадигме XML, удобной для автоматической обработки на компьютере. Стандартное решение – использование описаний Altova SemanticWorks показалось нам недостаточно удобным для нашей задачи. Мы провели эксперимент с системой SemTalk2. Преимущество SemTalk2 состоит в том, что она позволяет использовать одно и то же средство для редактирования онтологии и создания представлений из элементов этой онтологии. При этом она обладает удобным интерфейсом и редактором онтологий и представлений. SemTalk поддерживает два типа диаграмм: диаграмму классов – для описания онтологии и диаграмму экземпляров – для описания МСИ и позволяет импортировать внешние МСИ.

В эксперименте с системой SemTalk2 мы построили экспериментальную онтологию, в которой определили классы объектов: HOUSE, TREE, CHAPEL, SMOKE-STACK, BRANCH, FIELD, WINDOW и др., их свойства: COLOR, SIZE, SHAPE и др. и значения свойств: COLOR- BLACK, RED, и т.д.; SIZE - BIG, SMALLinWIDTH, BIGinHIGHT, и т.д.; SHAPE - SQUARE, ROND и т.д.

Мы определили одно композиционное отношение вложения – INCLUDES и два семантических отношения: PART-OF и LOCALIZATION. INCLUDES – это формальное отношение между объектами – отношение вхождения одного объекта в границы другого. Оно же представляет свойства объектов. Например, содержание текста «*На небе солнце. По небу летит птица с большими крыльями*» можно изобразить, выразив отношение вложения в виде скобок:

(SKY (BIRD (FLYING)(BIG WINGS)) SUN)).

На основании знаний оно может быть интерпретировано через отношение PART-OF или LOCALIZATION, например, PART-OF (BIG WINGS, BIRD), LOCALIZATION (SKY, SUN).

Пространственные отношения представлены в онтологии как «обратимые»:

– To-the-left(X, Y) / To-the-right(Y, X)
– Near(X, Y) / Near(Y, X)
– Around(Y, X) / In-the-centre(Y, X) и др.

Это позволяет использовать разные стратегии описания, в которых объекты могут следовать в представлении в прямом и обратном порядке: X, Y или Y, X.

Описание в среде SemTalk2 создается перетаскиванием элементов онтологии (классов, свойств и отношений) в область формирования МСИ. На рис. 2 показан скриншот системы SemTalk2. В правой части видна онтология, в левой – МСИ, построенная из экземпляров классов. В левом нижнем углу приведена описываемая фотография 00039.

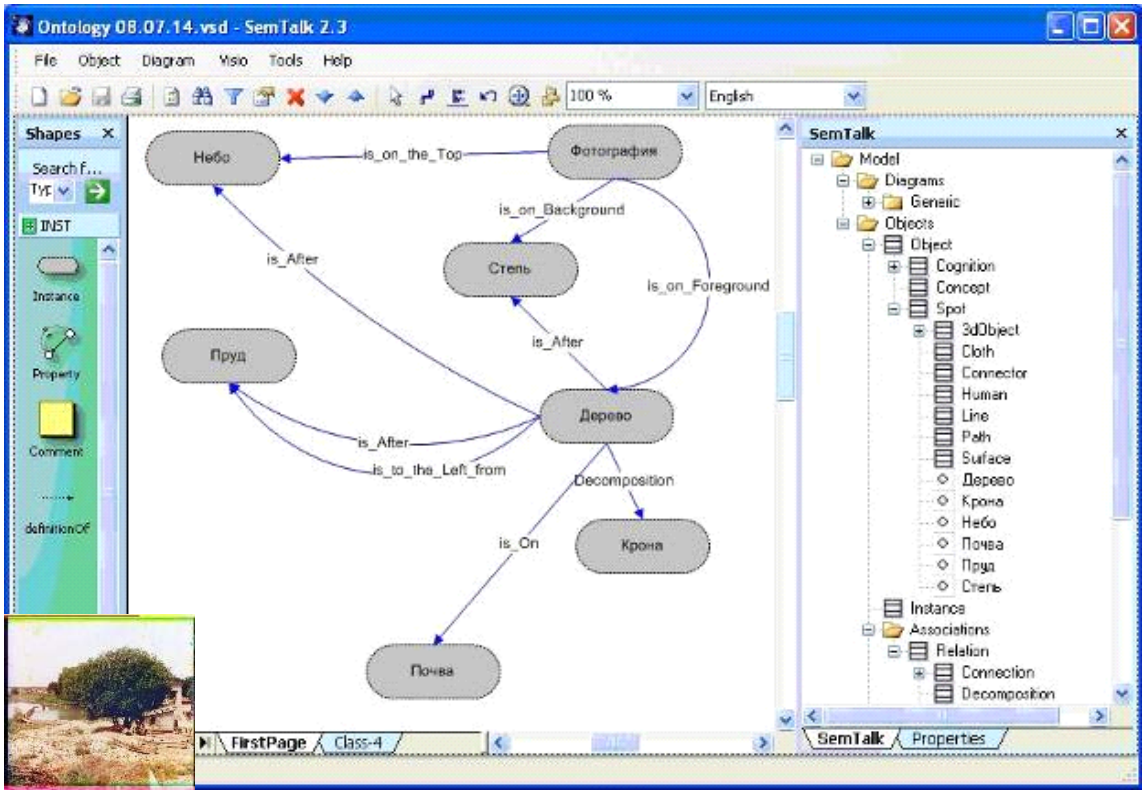


Рис. 2: Скриншот SemTalk2показывает модель содержания фотографии 00039, созданную из элементов онтологии, показанной справа

Модель содержания изображения 00039 создана на основе следующих двух студенческих описаний:
1. На фотографии изображено большое зелёное дерево на фоне чистого неба. Дерево находится на каменистой почве. Стоит ясная погода. Справа от дерева виднеется пруд.
2. Центр фотографии занимает раскидистое дерево с широкой кроной. На фоне – степь. Левее и чуть дальше дерева – небольшое озеро. Примерно две пятых фона - белое небо.

Эксперимент с SemTalk2 показал, что при указанных преимуществах, свободно доступная в Интернет версия обладает также существенными недостатками:

- большая часть заявленного функционала не работает, если его использовать в контексте нашей задачи;
- ключевые концепции построения онтологии, необходимость которых была нами осознана в процессе эксперимента, отсутствует в текущей реализации системы, в частности, множественное наследование, именование экземпляров сущностей и др.

В результате мы начали разрабатывать собственную модель онтологии, которая кратко представлена в следующем разделе.

5 Онтология DEMLinG

В настоящее время создается новая среда для моделирования онтологий и МСИ, позволяющая множественное наследование и имеющая некоторые другие полезные свойства. Она находится в процессе разработки и отладки, одновременно в ней проводятся эксперименты по созданию онтологии и ее использованию для построения МСИ. На рис. 3 представлен скриншот этой среды с одним из вариантов разрабатываемой онтологии.

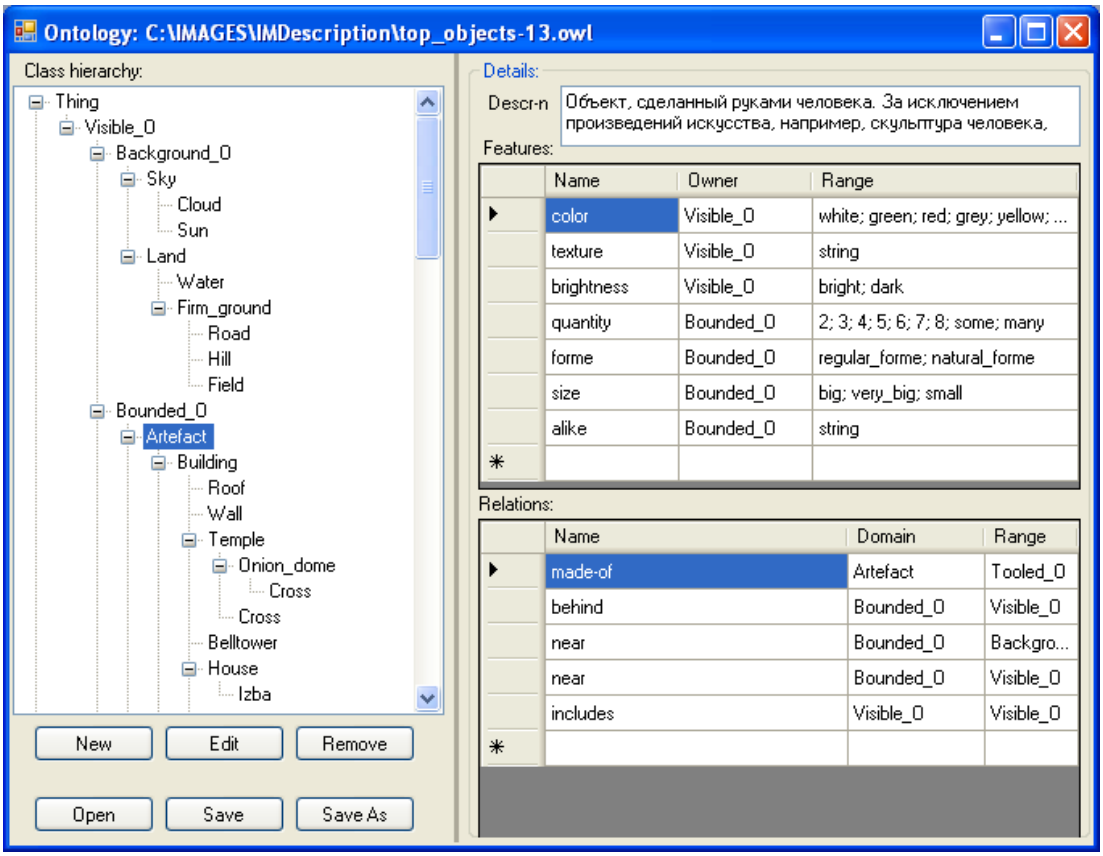


Рис. 3. Онтология DEMLinG

Относительно содержания онтологии, мы пока не пытались последовательно описывать визуальные свойства объектов, такие как, например, что окно обычно имеет прямоугольную форму и размер с человека или в несколько раз меньше. Также мы не пытались рассматривать возможности ее участия в процессе распознавания изображений. Для генерации нужны свойства объектов, влияющие на их участие в процессах, обозначаемых определенными глаголами ЕЯ. Эти свойства имеют когнитивную природу и используются именно в области зрительного восприятия человека, в других областях, например, «общение с БД», они не актуальны. Некоторые из них важны для построения МСИ, другие - для взаимодействия с лексической семантикой глаголов. В настоящее время нами разрабатывается версия «визуальной» онтологии, которая включает следующие типы объ-

ектов, распределенные по двум группам – реальные объекты и мнимые объекты. Реальные объекты, те, которые представлены на фотографиях, имеют границы, могут быть распознаны по своему образу и визуальным свойствам.

Реальные объекты:

– носитель изображения - фото

– фоновые объекты – небо, земля;

– части земли – поле, лес, река, дорога и т.д.

– совокупные объекты – поселок, улица, и др.

– объекты – дерево, дом, человек и т.д.;

– части объектов – дверь, окно, ветка, рука, шляпа и т.д.

Выделение типов мотивируется когнитивными соображениями следующего этапа – собственно генерации текста, а именно, теми функциями, которые объекты обычно выполняют с точки зрения нашего восприятия: фоновые объекты – обычно локализации, объекты – обычно локализуемые, части объектов – на изображении вложены в границы других объектов.

Свойства реальных объектов включают вышеупомянутые визуальные свойства и ту информацию о времени, месте и невизуальных свойствах изображенных объектов, которая может быть извлечена из названия фотографии, например, национальность, профессия и место работы изображенного человека.

Мнимые объекты:

– горизонт; горизонт воспринимается как линия, хотя на самом деле является только границей между двумя фактурами – неба и земли;

– группы объектов –совокупности однородных объектов, образующие единое пятно на изображении, например, комплекс монастырских построек, группа людей при описании конкретных изображений. В онтологии этот класс не имеет подклассов, а используется для группировки объектов;

– планы – характеризует группы объектов часто композиционно объединенных, нижняя граница которых находится на изображении приблизительно на одной горизонтальной линии. Обычно в описаниях используется «передний план», который располагается в центре или нижней части фотографии;

– части фотографии, относительно которых ориентируются объекты – центр, углы (прав/верх, прав/ниж, лев/верх, лев/ниж), части – (прав, лев, верн, нижн).

6 Планирование текста

Ключевым процессом генерации текста на основе МСИ является процесс планирования. Общие методы планирования, например, метод план-операторов (Болдасов, Соколова, 2005 и 2006), строятся на основе предикатов, поэтому не применимы к нашей задаче. Более адекватным является аппарат дискурсивных стратегий К. Маккьюин. Мы можем моделировать определенный порядок обхода МСИ, сопровождающийся темарематическими чередованиями в линейном плане текста. В настоящее время речь может идти только о проведении исследования. Исследование проводится в двух направлениях: 1) исследование организации текстов описания изображения, для которого мы имеем материал нашего корпуса; 2) исследования соответствий между значениями глаголов и конфигурациями объектов, которые они реализуют в текстах.

Предварительное исследование организации текстов показывает, что в них заложены некоторые когнитивные схема. Такие схемы описывались, в частности, в (Кобозева, 1997, 2000) и др. Для видовых фотографий вырисовывается несколько иной подход. Содержание фотографии состоит из выделенных объектов, к которым относятся сама фотография как носитель изображения, фоновые объекты, горизонт, передний план и некоторые выделяющиеся своими размерами или положением на изображении объекты. Текст организован как цепочка объектов, которая начинается от одного выделенного объекта, обычно это сама фотография, затем либо формирует цепочку, в которой перечисляются объекты, примыкающие к исходному объекту, и/или происходит скачок к следующему выделенному объекту. Назовем цепочку объектов между двумя выделенными объектами – «пассаж». В пассаже фокус внимания автора текста плавно переходит с одного объекта на другой по композиционным и пространственным отношениям, пока все объекты, составляющие данную часть композиции изображения, не будут исчерпаны. Разберем для примера конкретный текст, описывающий фотографию, приведенную выше на рис. 1. Текст разбит на предложения, где жирным шрифтом выделены объекты, на которых происходит скачок и подчеркнуты объекты, которые составляют пассажи данного плана текста:

- **На фотографии** изображен вид на окраину поселка летним днем.
- **На переднем плане** небольшая речка, чуть подальше плотина и мост, укрепленный бревнами.
- Слева и справа изображено несколько маленьких деревенских домиков, некоторые с участком земли, огороженным дощатым забором.
- Между домами **на самой верхушке холма** стоит большой белый монастырь.
- К нему ведет деревянная лестница на холме.
- **На картине** изображено много проселочных дорог, которые расходятся в разные стороны.

имеем следующий порядок упоминания объектов:

1. «**Фото**» → «поселок»
2. «**Передний план**» → «речка» → «плотина» → «мост» → «бревна»
3.(слева и справа от «моста») → «домики» →»участки» →»заборы»
4. (между «домиками»)
4.1. **Верхушка Холма (=горизонт)** → «монастырь»
5.(к нему (монастырю)) → «лестница» (на холме)
6. «**Картина (=фото)**» → «дороги».

Порядок упоминания объектов может рассматриваться как некая проекция плана текста. Если мы сможем выстраивать такие «порядки», вытягивая наше входное представление – МСИ в линию, то вторая задача в процессе генерации текста состоит в том, чтобы решить, каким образом эта цепочка распадается на предложения текста, и какие из отношений, стоящих за скачками и стрелками должны быть реализованы глаголами.

Исследование предполагает изучение семантики глаголов в области «визуального восприятия» и, в результате, построения семантических шаблонов, отображающих семантику глаголов в этой области. Например, глаголы зрительного восприятия имеют в семантическом шаблоне позицию «наблюдатель», которая может выражаться в области «визуального восприятия» местоимением «*мы*». Объектами глаголов зрительного восприятия могут быть любые объекты, кроме мнимых, за исключением понятия «горизонта». Человек считает, что он видит горизонт, но вряд ли скажет «мы видим горизонт» в отличие от: «*мы видим фото, небо, дом, руку и т.д.*». В этом случае речь идет обычно о «линии горизонта». Глаголы изображения более ограничены в выборе объектов, реализующих позиции их семантического шаблона. Эти глаголы предполагают роль «создателя», которая очень ограниченно может присутствовать в описаниях. Обычно же «субъектом» - синтаксическим подлежащим, является носитель изображения – *фотография*, *картина*, а объектом – независимый объект (не часть объекта) и т.д.

Кроме свойств глаголов существуют еще свойства самих объектов, предсказывающие процессы, которые могут быть с ними ассоциированы. В частности, объект «река» может проявлять себя в тексте как точечный объект, например, «*Вдали виднеется река*», как поверхность, например, «*По реке плывет пароход*» и как зеркало, например, «*Дом и деревья отражаются в реке*». Изучение семантических шаблонов глаголов в области зрительного восприятия требует подробного исследования на корпусе текстов и МСИ изображений. Первые шаги в этом направлении сделаны в (Соколова, 2007) и (Заикина, 2010).

Список литературы

1. Boldasov M.V., Sokolova E.G. QGen – generation module for the register restricted InBASE system // Computational linguistics and intelligent text processing (A.Gelbukh ed.). – Proceedings of the 4th International conference, CICLing 2003, Mexico City, Mexico, September 2002. Springer Berlin Heidelberg, Germany, 2003. Pages: 465-476.
2. Boldasov M.V., Sokolova E.G. User query understanding by the InBASE system as a source for Multilingual NL generation module // Text, Speech and Dialogue (P. Sojka, I. Kopeček and K. Pala eds.). – Proceedings of the 5th International conference, TSD 2002, Brno, Czech Republic, September 2002. Springer-Verlag Berlin Heidelberg, Germany, 2002. Pages: 33-40.
3. Deschacht K., Moens M-F. Text analysis for automatic image annotation // The 45th Annual meeting of the Association for Computational Linguistics, Prague, June 2007. http://acl.ldc.upenn.edu/P/P07/P07-1126.pdf
4. Gornostay T., Aker A. Development and implementation of multilingual object type toponym-referenced text corpora for optimizing automatic image description genereation // Proceedings of the Conference on computational linguistics and intellectual technologies DIALOG'2009 - 2009, Bekasovo, Russia May 27-31, 2009. Pages: 580 – 587.
5. Hollink L., Shreiber G., Wielemaker J., Wielinga B. Semantic annotation of image collections // S. Handschuh, M. Koivunen, R. Dieng, and S. Staab, editors, Knowledge Capture 2003. Proceedings Knowledge Markup and Semantic Annotation Workshop, Florida, USA, October 2003. P. 41-48. http://www.cs.vu.nl/~guus/papers/Hollink03b.pdf
6. Kruijff G-J., Bateman J., Teich E., Sharof S., Sokolova L., Kruijff-Korbayova I., Skoumalova H., Staykova K., Hana J., Hartley T. Multilinguality in a text generation system for three Slavic languages // Proceedings of the 18th conference on Computational linguistics - Volume 1, 2000, Saarbrъcken, Germany July 31 - August 04, 2000. Pages: 474 – 480.
7. Reineking Th., Schult N., Hois J. Evidential combination of ontological and statistical information for active scene classification // International Joint Conference on Knowledge Discovery, Knowledge Engineering and knowledge management, Final program and book of abstracts. Portugal, Funchal-Madeira, October 6-8, 2009, p. 59.
8. Schreiber A.Th., Dubbeldam B., Wielemaker J., Wielinga B.J. Ontology-based photo annotation //IEEE Intelligent systems, 16(3):66-74, May-June 2001. http://www.cs.vu.nl/~guus/papers/Schreiber01a.pdf
9. Tam, A.M. Leung, C.H.C. Structured Natural-Language description for semantic content retrieval // Journal of the American Society for Information Science and Technology Vol. 52(11), September 2001. Pages: 930 – 937.
10. Болдасов М.В. Парадигмы генерации ЕЯ текстов в инструментальной среде DEMLinG // Труды международной конференции по компьютерной лингвистике и интеллектуальным технологиям ДИАЛОГ'2003. Протвино: 2003. С. 66-75. www.dialog-21.ru/dialog2007/materials/html/78.htm
11. Болдасов М.В., Соколова Е.Г. Генерация текстов на естественном языке - состояние вопроса и прикладные системы // НТИ, Серия 2, №10, 2005, с.12-22.
12. Болдасов М.В., Соколова Е.Г. Генерация текстов на естественном языке - теории, методы, технологии // НТИ, Серия 2, №7, 2006, с.1-15.

13. Заикина Т.А. Глаголы зрительного вопсриятия и изображения вв перспективе генерации текстов – описаний фотографий – дипломная работа, защищена в РГГУ на кафедре ТиПЛ в 2010 г.
14. Кобозева И.М. Как мы описываем пространство, которое видим: композиционные стратегии // Труды Международного семинара Диалог'97 по компьютерной лингвистике и ее приложениям. Ясная Поляна, 10-15 июня. М. 1997. С. 132-136.
15. Кобозева И.М. Как мы описываем пространство, которое видим: форма объектов.// Труды международной конференции по компьютерной лингвистике и интеллектуальным технологиям ДИАЛОГ'2000.
16. Кобозева И.М. Представление знаний о физических объектах для систем типа «Рисунок - Текст» // Категоризация мира: пространство и время. Материалы конференции. Москва: МГУ, 1997.
17. Лобанов Б.М., Сизонов О.Г. Квазиречевой видеонавигатор для слепых // Речевые технологии №4, 2008. С. 103-110. http://speechtechnology.ru/files/4-2008.pdf
18. Соколова Е.Г., Болдасов М.В. Планирование текстов в системах генерации на естественном языке // Компьютерная лингвистика и интеллектуальные технологии. Труды Международной конференции Диалог'2005, "Звенигородский", 2005, с. 450-457.
19. Соколова Е.Г., Болдасов М.В. Формализованное описание содержания изображения как данные для генерации текста // Труды Международной конференции Диалог'2007 по компьютерной лингвистике и интеллектуальным технологиям. Бекасово, 30 мая – 3 июня, 2007. С.: 508 – 515.
20. Соколова Е.Г. Об использовании семантических отношений для описания изображений // Вестник РГГУ №8/07, Серия «Языкознание», Москва: РГГУ, 2007. С.: 131 – 144.

Представление обыденного знания в форме шаблонных ситуаций

И.С. Сысоев

Аннотация. Распространенные в наше время способы представления знаний обнаруживают определенные слабые стороны, сталкиваясь с разнообразием и расплывчатостью обыденного человеческого знания. Это делает осмысленным поиск альтернативных методик. В данной работе предлагается оригинальный подход к представлению знаний при помощи набора шаблонных ситуаций (заданных на нужном уровне абстракции). Рассуждения осуществляются путем поиска соответствий текущего образа мира с шаблонными ситуациями из базы знаний и переноса на текущий образ черт этих ситуаций. Целью работы является представление основных положений подхода. Однако в качестве иллюстрации будет представлена пока незаконченные проработки подхода.

1. Введение

На протяжении многолетней истории исследований в области искусственного интеллекта были разработаны весьма разнообразные способы представления знаний. Создатели многих из них имели в виду весьма амбициозные цели: создание машин, мыслящих в полном смысле этого слова; таких, чей интеллект не уступал бы человеческому. Излишне говорить, что этот великий вызов современной технологии так и не был достойно встречен, несмотря на многочисленные предпринятые попытки.

Наиболее развитой системой представления обыденного знания сегодня называют Cyc[1] [1]. В одном из своих проектов автор работы использовал эту систему. Согласно сложившемуся у него впечатлению, принятый в Cyc формально-логический способ слабо пригоден для представления повседневного человеческого знания. Неудовлетворенность результатами применения Cyc побудила автора искать пути преодоления его недостатков. На этих страницах он пытается представить результат своих размышлений.

Читателю данная работа может показаться весьма неформальной. Во-первых, это связано с выбранной областью – обыденным знанием – о которой затруднительно рассуждать формально по причине отсутствия строгой теории человеческого мышления. В частности поэтому, говоря о недостатках существующих подходов, автор не будет давать никаких оценок, но лишь поделится своим опытом. Во-вторых, автор стремится избежать излишней конкретики, поскольку его собственный подход еще весьма нов и его детали не определились полностью. Тем не менее, в разделе 4 некие промежуточные наработки будет даны в качестве примера, чтобы читатель смог получить лучшее представление о предмете.

Неформальный стиль изложения соответствует задаче данного текста. Автор не пытался продемонстрировать результаты законченной работы. Целью текста было ознакомить читателя с идеей рассуждений на основе шаблонных ситуаций и, возможно, заинтересовать его этой идеей.

2. Проблемы существующих подходов

С точки зрения автора, основная проблема принятого в Cyc подхода, как и многих других методик, заключается в монолитности больших онтологий. Вводя каждое новое понятие, в таких системах приходится предусмотреть все возможные варианты его дальнейшего использования. В результате базы знаний, подобные Cyc, напоминают монументальное сооружение, возводить которое приходится с огромной осторожностью по тщательно разработанному плану. Богатство и разнообразие ситуаций реальной жизни плохо вписывается в строгие рамки заранее составленных планов. Кроме того, для того, чтобы ввести какой-либо новый факт или даже задать запрос системе, требуется тщательно ознакомиться с представлением в ней такого рода знаний; порой это ознакомление занимает часы. Это делает использование системы весьма трудным.

Мы можем представить себе, как в идеале должно происходить наполнение знаниями интеллектуальной машины. Этот процесс должен напоминать воспитание родителями ребенка: хотя и сложную задачу, но все же не требующую ни специальных навыков, ни предварительного проектирования базы знаний и осуществимую в течение одной человеческой жизни.

По всей видимости, на пути к достижению этой цели нам следует наделить представления знаний свойством локальности. Большая база знаний в таком случае уже не будет напоминать монолит, но станет похожа на лоскутное одеяло. В каждой ситуации объекты и отношения будут выглядеть несколько по-другому, представленные наиболее подходящим образом. Переключение между различными модами представления будет происходить автоматически. В этом случае, если имеющиеся способы представления не позволяют выразить какой-либо факт, будет просто сформирована новая мода представления, в которой запись факта становится возможной. Есть основания полагать, что мышление людей устроено «лоскутным», а не «монолитным» образом. В [2,

[1] http://www.cerebromente.org.br/n07/opiniao/minsky/minsky_i.htm – интервью с Марвином Минским

глава «Представление Знаний»], одновременно с демонстрацией весьма изощренных способов представления некоторых знаний, высказывается также сомнение в «сходимости» множества специализированных онтологий к некоторой онтологии общего назначения. Похоже, что люди и не имеют в своих головах никакого аналога такой онтологии, а вместо этого пользуются преимуществами мультимодального представления понятий.

Другим уязвимым местом существующих подходов являются рассуждения над не-тождественно-истинными утверждениями. В этом случае необходимо каким-либо образом делать выбор между противоречивыми суждениями, неизбежно возникающими в ходе рассуждений. Предпринимались попытки преодолеть это затруднение путем введения логик, допускающих исключения (логика косвенного описания и логика умолчания, [2]). В частности, Cyc использует один из вариантов логики с исключениями и поддерживает две степени силы высказывания: monotonic (тождественно истинное) и default (истинное по умолчанию). Однако на практике это приводит к тому, что практически всем утверждениям приходится давать силу default (в самом деле, тождественно истинные высказывания – редкое явление в сфере обыденного знания). Если противоречие получено в результате применения двух default-утверждений, то Cyc не может определить, какой вывод истинен. Следовательно, данный подход не обеспечивает достаточной гибкости.

Другим распространенным способом разрешения конфликтов вывода является использование коэффициентов уверенности высказываний. В [1] приведена обоснованная критика такого подхода. Она сводится к сложности согласования коэффициентов уверенности в масштабных базах знаний, что связано с наличием определенной произвольности в выборе их значений.

Вероятностные подходы к представлению знаний подвержены похожей проблеме. Говоря точнее, чтобы назначить безусловные и условные вероятности для событий в базе знаний, мы должны сделать что-либо из следующего:

- точно измерить эти вероятности. Помимо того, что это в любом случае крайне трудоемко, не всегда ясно даже, каким образом можно провести такое измерение;
- назначить вероятности, исходя из интуитивных представлений. Тогда мы сталкиваемся с той же проблемой произвольности, что и для коэффициентов уверенности;
- позволить системе обучаться. На современном уровне развития технологии мы не можем отправить систему самостоятельно исследовать мир. Значит, создание учебных выборок является задачей специалиста по знаниям. Это не только чрезвычайно трудоемко, но и делает результат обучения зависимым от субъективных представлений человека, т.е. в завуалированном виде ставит нас перед все той же проблемой произвольности.

Проблемой существующих подходов автор считает и ту, что ни один из них не использует ассоциации, играющие важную роль в человеческом мышлении. Человеку достаточно усвоить картину, что охотники имеют ружья, и охотник у него будет ассоциироваться с ружьем, а ружье с охотником. Так, рассматривая некоторое ружье, он будет автоматически предполагать, что оно принадлежит некоторому охотнику. Причем, что важно, он будет автоматически делать такое предположение ровно до тех пор, пока не узнает про какие-либо другие группы обладателей ружей. В существующих методиках подобного поведения можно достигнуть, только отдельно задав утверждения вида «X – охотник → Y – ружье, что X имеет Y» и «Y – ружье → X – охотник, что X имеет Y» и вручную проследив их корректность (например, что никто кроме охотников в нашем мире не имеет ружья). Это утомительная, чреватая ошибками и требующая глобальной осведомленности о всей базе знаний задача. Как мы увидим, существует возможность отказаться от этой работы.

Подход, описанный на этих страницах, представляет собой попытку справиться с указанными недостатками.

3. Идея подхода

Идеей данного подхода является представление знаний в виде совокупности образов стереотипных случаев. Образ представляет собой набор участников и отношений между ними. Как мы увидим далее, подход допускает и описание ситуаций, развивающихся во времени. Описываются не конкретные случаи, а некоторые абстрагированные шаблонные ситуации: «охотники имеют ружья».

Имея набор шаблонных случаев и образ некоей текущей ситуации, мы можем выполнять два важных действия: делать предположения относительно ненаблюдаемых аспектов ситуации («что должно находиться за кустом»), и предсказывать развитие ситуации в будущем (или предполагать, что могло предшествовать ей в прошлом). Согласно теории «память-предсказание», именно эти две процедуры и являются слагаемыми понимания [3].

Делать такие рассуждения мы можем следующим путем. Имея начальный образ некоторой ситуации, система отыскивает в памяти другие образы, которые (по определенным правилам) можно сопоставить с текущим. Затем на текущий образ переносятся их черты. Процесс повторяется итеративно до тех пор, пока ситуации невозможно будет сопоставить никаких новых образов. В этом смысле ход рассуждений напоминает процесс прямого логического вывода.

Мы можем усмотреть здесь сходство с концепцией автоассоциативной памяти из теории нейронных сетей

[4]: нашей задачей является восстановление полной картины по частичному образу ситуации. Разумеется, модели автоассоциативной памяти решают совсем другие задачи, но данный пример является хорошей иллюстрацией.

По-видимому, первым идею об использовании запомненных образов в рассуждениях высказал Витгенштейн в «Философских исследованиях». Кун, определяя понятие парадигмы, говорит: «в своем установившемся употреблении понятие парадигмы означает принятую модель или образец»[2] - и широко использует этот оттенок смысла данного слова. Впервые модель рассуждений через запомненные ситуации предложил Шенк [5]. Его работы положили начало области рассуждения на основе прецедентов (case-based reasoning, CBR), обзор которой дается в [6]. Схожими чертами обладает и модель «память-предсказание» Хокинса [3].

Подход, описанный в данной работе, идейно близок двум вышеупомянутым подходам, но относится несколько к другой области. Он специализируется на задании знаний человеком в форме абстрактных ситуаций, а не на обучении машины на конкретных ситуациях. Специфическими для рассматриваемого подхода являются также итеративный способ проведения рассуждений и методы разрешения противоречий.

4. Возможная реализация подхода

В данном разделе будет приведена возможная реализация вышеописанной идеи. Детали реализации еще не являются чем-то оформившимся и устоявшимся, поэтому она приводится здесь лишь в качестве примера, а не как описание результатов работы.

4.1. Образ ситуации

Как было сказано выше, образ шаблонной ситуации представляет собой набор абстрактно заданных участников и отношений между ними. Например, нижеследующие два предложения задают образ небольшой шаблонной ситуации:

have(knight, sword)
use(knight, sword, for fighting)

Рассмотрим данный формат записи.

Слово knight здесь описывает объект: некоего абстрактного рыцаря, одного и того же для всех предложений конкретного образа. Описание объекта может быть и более длинным, например brave knight, при этом knight считается основным типом, a brave – дополнительным признаком. Если в ситуации участвует более одного рыцаря, необходимо различать их по дополнительным признакам («fat knight» и «grim knight») или пронумеровать («knight 1» и «knight 2»).

Между участниками ситуации заданы два отношения. Отношения могут включать «дополнения», помеченные ключевыми словами, такие как «for fighting» в данном примере. Отношение одного и того же вида может иметь разный набор дополнений в разных образах ситуаций. С точки зрения автора, нефиксированный набор дополнений позволяет дополнительно повысить гибкость системы.

Объектам шаблонной ситуации могут быть приписаны дополнительные квалификаторы, такие как «!» - квалификатор единственности. Например, wear(man, !hat) – означает, что человек носит одну шляпу. Квалификаторы могут иметь и предложения. Так, квалификатор «@» (от «arbitrary») означает, что предложение является опциональным для данной ситуации. Если данное предложение в контексте текущей ситуации должно быть отброшено согласно правилам разрешения конфликтов, то это не приводит к выводу о неприменимости всей шаблонной ситуации.

В образах ситуаций фигурируют абстрактные объекты, принадлежащие к определенным типам. В системе задана иерархия этих типов: они находятся в отношении наследования, которое является частичным порядком. В настоящий момент автор исследует также применение иерархий с нарушениями транзитивности. Для такой иерархии можно было бы, например, сказать, что тип «лучник» является наследником типа «воин», а затем (в контексте средневековья) для типа «охотник» указать, что он является «лучником», но не «воином». Иерархии с нарушением транзитивности позволили бы дополнительно сократить необходимость в предварительном проектировании базы знаний.

На типах объектов из иерархии может быть задано отношение disjoint – что означает, что типы не пересекаются (как, например, dog и cat). По умолчанию наследники одного типа являются непересекающимися типами, что позволяет легко строить классификации. Это правило можно обойти, используя специальный квалификатор при наследовании.

Иерархия наследования вводится и на отношениях.

[2] Т.Кун. Структура научных революций. Глава 3.

4.2. Применимость шаблонной ситуации

Шаблонная ситуация применима к текущей, если ее объекты и отношения в определенном смысле отождествимы с объектами и отношениями текущей ситуации. При этом типы объектов и отношений в текущей ситуации могут быть недостаточно точными (и тогда в результате применения шаблонной ситуации они будут уточнены), а набор отношений в текущей ситуации может быть не полон (и тогда он будет дополнен). Но все же элементы структуры отношений применяемой ситуации должны присутствовать и в текущей ситуации: необходимо, чтобы соответственные объекты были соединены соответственными отношениями в один связный граф. Конечно же, если некоторые объекты и отношения в текущей ситуации уже определены более точно, чем в применяемой ситуации, это не должно мешать применению шаблонной ситуации.

Исходя из этого, введем понятие отождествимости.

Объект A и объект B отождествимы тогда и только тогда, когда основной тип объекта A наследует основной тип объекта B или наоборот, а никакие дополнительные признаки объектов не находятся в отношении disjoint.

Отношение A и отношение B отождествимы тогда и только тогда, когда тип отношения A является типом отношения B или наоборот, а соответствующие аргументы отношений являются соответственными объектами.

4.3. Применение шаблонной ситуации

Итак, в результате применения шаблонной ситуации могут быть уточнены типы объектов и отношений текущей ситуации и добавлены новые отношения между сопоставленными объектами. И новые типы, и новые отношения попросту копируются из шаблонной ситуации. Читатель, знакомый с рассуждениями по аналогии [6] – одной из разновидностей CBR – может поинтересоваться, не было ли бы полезным ввести возможность модификации по определенным правилам элементов шаблонной ситуации при применении его к текущей ситуации. На самом деле подобную возможность при необходимости можно реализовать при помощи некоторых приемов написания шаблонных ситуаций.

Интересной возможностью является введение дополнительных объектов при применении шаблонных ситуаций. Например, рассуждая о феодале, мы можем вспомнить, что ему должен принадлежать замок. В текущей реализации объекты, которые можно вводить в ходе рассуждений, помечаются квалификатором «@» (от «arbitrary»).

4.4. Отношение «общее-частное» на ситуациях

При определении приоритетов тех или иных выводов в задаче разрешении конфликтов, а также в некоторых других ситуациях, оказывается весьма полезным сравнивать шаблонные ситуации по критерию «общее-частное». Пусть имеется ситуация A, ситуация B и некоторая взаимнооднозначная функция отождествления eq из подмножества объектов и отношений A в подмножество объектов и отношений B. Ситуация A является более общей, чем B, при отождествлении eq, если:

- всякому объекту в A сопоставляется объект в B, всякому отношению в A сопоставляется отношение в B;
- некоторые объекты в A состоят в некотором отношении, то образы этих объектов состоят в образе этого отношения;
- объект из A является представителем некоторого типа, то его образ в B является представителем *наследника* этого типа;
- отношение в A относится к некоторому типу, то его образ в B относится к *наследнику* этого типа.

Ситуация A является более общей, чем B, если существует отождествление eq, что A более общая, чем B, при отождествлении eq.

Поскольку типы объектов и отношений образуют частичный порядок, можно показать, что отношение «общее-частное» на шаблонных ситуациях также будет частичным порядком. Это обстоятельство весьма значимо, поскольку позволяет использовать ситуации в качестве объектов в других ситуациях, используя те же самые методы вывода и разрешения противоречий. Такое отображение языка в себя имеет несколько полезных применений:

- рассуждения об убеждениях [2, раздел «формальная теория убеждений»];
- сценарии: цепочки ситуаций, где каждая ситуация является «кадром» процесса, развивающегося во времени (подобные скриптлетам и MOP у Шенка [5]);
- метарассуждения: рассуждения над ситуациями. Например, можно организовать отдельную базу знаний для рассуждения о приоритетах шаблонных ситуаций при разрешении противоречий.

4.5. Разрешение противоречий

В ходе рассуждений над обыденным знанием, со свойственным ему наличием большого числа исключений, вполне естественно появление противоречий между выводами. Примерами противоречий могут быть наруше-

ние единственности для объектов с квалификатором «!», отнесение одного объекта к двум disjoint-типам, появление одних и тех же объектов в одном и том же отношении с отрицанием и без него. Противоречия должны тем или иным образом разрешаться, и менее приоритетное суждение должно удаляться.

Во всех случаях при разрешении противоречий сравниваются две шаблонные ситуации, которые имели отношение к получению противоречивых выводов. По тем или иным правилам выбирается более «сильная» из них, и соответствующему выводу отдается приоритет. «Проигравший» вывод или вся «проигравшая» шаблонная ситуация удаляется (это управляется квалификаторами «@» при отношениях в ситуации). На основе сравнения двух шаблонных ситуаций можно делать более аккуратные суждения о приоритетах выводов, нежели на основе сравнения двух мало о чем говорящих чисел.

Автор исследует пять различных правил разрешения противоречий. Поскольку данная работа не преследует цель детального описания подхода, мы рассмотрим лишь два правила.

Назовем проекцией текущей ситуации на шаблонную ситуацию совокупность объектов и отношений текущей ситуации, которые были отождествлены с объектами и отношениями шаблонной ситуации. Формально проекция является шаблонной ситуацией, и мы можем сравнивать проекции друг с другом согласно правилам из пункта 4.4.

Пусть мы сравниваем две шаблонные ситуации: A и B. Пусть в результате применения A мы имеем вывод α, а в результате применения B – вывод в, противоречащий б. Рассмотрим проекции текущей ситуации на A и B: A’ и B’ соответственно. Рассмотрим естественное отождествление eq объектов и отношений в A и B: отождествляются общие элементы текущей ситуации.

Правило 1 звучит так: если A’ является более частной ситуацией, чем B’, то A является «победителем» в сравнении. Это выглядит логичным: данная ситуация имеет лучшее, более точное соответствие текущей ситуации.

Согласно второму правилу, если A’ и B’ в точности равны, то обе ситуации признаются «проигравшими».

Если A’ и B’ несравнимы, в действие вступают другие правила.

5. Мультимодальное представление знаний

Важным преимуществом, которое автор надеется получить за счет применения данного подхода, является удобство мультимодального представления знаний, речь о котором шла в разделе 2. Шаблонные ситуации устроены достаточно просто, чтобы позволить их автоматическую перезапись. Более того, наработки по вышеприведенному способу рассуждений могут оказаться полезными для манипуляции над самим представлением текущей ситуации в целях переформулировки ее в другие представления.

Стратегия рекурсивной переформулировки входных данных оказалась очень эффективной для чатботов, таких как ALICE[3]. Входными данными чатбота являются фразы, над которыми эти программы осуществляют чисто синтаксические манипуляции - но вполне вероятно, что похожим образом можно манипулировать и осмысленными конструкциями.

В этом случае мультимодальное представление знаний можно было бы организовать следующим образом. Система проведения рассуждений состояла бы из двух частей: модуля метарассуждений и ядра вывода. Модуль метарассуждений осуществлял бы переформулировку элементов текущей ситуации в различные формы представления. Сопоставляя затем текущую ситуацию шаблонным ситуациям, записанным в различных представлениях, ядро вывода осуществляло бы собственно рассуждения.

6. Заключение

Подход на основе образов ситуаций обещает некоторые важные преимущества в области представления обыденного знания:

- такой способ представления может быть более естественным для человека. Это должно значительно повысить скорость наполнения баз знаний;
- метод задействует ассоциации, то есть эмулирует другой важный элемент человеческого мышления (собственно, шаблонная ситуация как раз и является ассоциативной связью между ее объектами). Если ход рассуждений системы будет подобен ходу рассуждений человека, то и выводы системы должны быть более естественными;
- метод предоставляет гибкие способы разрешения конфликтов вывода;
- ожидается, что метод хорошо поддерживает мультимодальное представление знаний.

Хотя эти пункты являются более или менее спорными, представляется, что данный способ представления знаний заслуживает внимания и дальнейшего развития.

[3] A. L. I. C. E. and AIML Documentation. Dr. Richard S. Wallace

В настоящее время в процессе написания находится программа-прототип, реализующая описанный способ ведения рассуждений. Автор собирается применить прототип в текстовой игре со сложными правилами, которую он планирует написать совместно с другим магистрантом НГУ.

Список литературы

1. Lenat D.B. Cyc: A large-scale investment in knowledge infrastructure. Communications of ACM, № 38.
2. С. Рассел, П. Норвиг. Искусственный Интеллект: Современный Подход.
3. Д. Хокинс. Об интеллекте.
4. Ф. Уоссермен. Нейрокомпьютерная техника: Теория и практика. Глава 6.
5. R. Schank. Dynamic Memory: A Theory of Learning in Computers and People
6. Aamodt, E. Plaza: Case-Based Reasoning: Foundational Issues, Methodological Variations, and System Approaches. Artificial Intelligence Communications, IOS Press, Vol. 7:1, pp. 39–59

Онтологический подход для создания автоматической классификации информационных ресурсов для естественных наук
(Тезисы)

А.З. Фазлиев

Институт оптики атмосферы СО РАН, Томск

Представление знаний является дисциплиной объединяющей логику, онтологию и вычисления [1]. Исследователь, использующий представление знаний в явной или неявной форме, выделяет набор вычислений необходимый для решения своих задач. Набор таких задач является основой для построения концептуализации (онтологии) предметной области. Для построения логической теории [2] в предметной области выбирается логика, обладающая необходимой выразительностью. Такой подход можно применять и для задач классификации информационных ресурсов.

В вводной части доклада представлены классификация наук, сделанная Пирсом [3] и категории верхнего уровня [1]. Последние используются в задачах интеграции разнородных информационных ресурсов. В частности, обсуждается подход J.Sowa к классификации математических теорий.

В докладе рассмотрена задача автоматической классификации информационных ресурсов в предметных областях связанных с решением прямых и обратных задач. Такие задачи возникают при измерениях и предсказании результатов экспериментов в естественных науках, содержащих значительную количественную компоненту.

В качестве примера рассмотрена классификация информационных ресурсов в спектроскопии. Дана интерпретация модели предметной области, представленной в виде двух связанных цепей задач. В рамках подхода Semantic web определены классы задач и рассмотрен подход к автоматическому построению таксономии классов, характеризующих информационные ресурсы в предметной области. Для опубликованных информационных ресурсов таксономия классов строится с помощью баз данных, содержащих перечни концептов предметной области.

Список литературы

1. Sowa J.F. Knowledge Representation: Logical, Philosophical, and Computational Foundations, Brooks Cole Publishing Co., Pacific Grove, CA, 2000. 594 p.
2. Зиновьев А.А. Основы логической теории знаний. М.: Наука, 1967. 260 с.
3. Charles Sanders Peirce, The Collected Papers Vol. I.: Principles of Philosophy (1931) Book II. The Classification of the Sciences