Труды семинара «Знания и Онтологии *ELSEWHERE* 2009», ассоциированного с 17-й международной конференцией по понятийным структурам

Proceedings of the «Knowledge and Ontology *ELSEWHERE*» Workshop, in conjunction with the 17th International Conference on Conceptual Structures

Государственный Университет –Высшая школа экономики
Москва
2009

University Higher School of Economics,
Moscow, 2009

Preface

The first inaugural workshop "Knowledge and Ontology *ELSEWHERE*" took place October 03, 2008, in affiliation with 11th Russian National Conference on Artificial Intelligence (CAI-08) in Dubna near Moscow. The present "Knowledge and Ontology *ELSEWHERE*" (*ELSEWHERE*2009) is the second edition of the workshop, and we are happy that this year it is affiliated with 17th International Conference on Conceptual Structures (ICCS'09), hosted by State University Higher School of Economics (SU HSE, Moscow, Russia). On behalf of *ELSEWHERE*2009 Program Committee I would like to thank organizers of the Conference and hosting University for the great opportunity.

The basic idea behind both workshops is some negative effect in mass-consciousness caused by rapid growth of Semantic Web technologies in the New Millennium. In particular, "knowledge" and "ontology" became just technical terms of Description Logic, Web Ontology Language or Machine Learning. But "ontology" is not just a technical term of Semantic Web, and "knowledge" is not a content. Both terms have everlasting science and philosophy meaning. Ontology is a branch of philosophy that examines entities and their ties (relations), classes of entities and classes of their ties. Knowledge (in accordance with Plato) is true belief, and epistemology is another branch of philosophy that studies methodology of belief validation. Just due to these arguments "knowledge" and "ontology" are very important notions ELSEWHERE outside Semantic Web.

In particular, *ELSEWHERE*2009 includes the following 2 invited talks, 8 contributed talks and 2 industrial posters (that were selected by program committee from 15 submitted papers).

1. Invited talks:

1.1. Konstantin Vorontsov (Institution of Russian Academy of Sciences – Dorodnicyn Computing Centre) Machine learning techniques based on rule induction. (In Russian).

1.2. Jack Stecher (Norwegian School of Economics and Business Administration) Knowledge and Backward Induction in Repeated Games. (In English, not in proceedings).

2. Contributed talks:

2.1. Alexander Akinin, Nikolay Shilov, Alexey Zubkov A. Towards Ontology for Classification of Computer Languages. (In English)

2.2. Zinaida Apanovich and Pavel Vinokurov Tools for visual analysis of ontology and content of a knowledge portal (In Russian)

2.3. Evgeny Beniaminov, Vladimir Lapshin, Dmitry Perov EZOP – Web 2.0 service of ontologies' construction. (In Russian)

2.4. Michael Boldasov and Elena Sokolova Ontology for Image Annotating . (In English)

2.5. Natalia Garanina Can autonomous robots solve assignment problem? (In Russian)

2.6. Tatiana Gavrilova, Vladimir Gorovoy, Elena Petrashen, Dmitry Mouromtsev Categorizing Knowledge Patterns into Ontological Framework. (In English)

2.7. Nikolay Skvortsov Specificity of ontology mapping approaches. (In Russian)

2.8. Gennady Sokolov Survey of modelling of Information and Material resources by means of decision-making. (In Russian)

3. Industrial posters:

3.1. Anna Korobko and Tatyana Penkova OLAP-modelling of Municipal Procurement Automation Support Problem. (In Russian)

3.2. Nikota Scherbakov and A. Andrichenko Management of the Standard/Reference Data and Technological Knowledge in the Industry. (In Russian)

On behalf of Program Committee *Nikolay V. Shilov*

22 June, 2009
Novosibirsk, Russia
shilov@iis.nsk.su

*ELSEWHERE*2009 Program Committee

Table of Contents

Towards Ontology for Classification of Computer Languages 1
Alexander Akinin, Nikolay Shilov, Alexey Zubkov

Ontology for Image Annotating 13
Michael Boldasov, Elena Sokolova

Categorizing Knowledge Patterns into Ontological Framework 22
Tatiana Gavrilova, Vladimir Gorovoy, Elena Petrashen, Dmitry Mouromtsev

Management of the Standard/Reference Data and Technological Knowledge in the Industry 28
A. Andrichenko, N. Scherbakov

Tools for Visual Analysis of Ontology and Content of a Knowledge Portal 33
Zinaida Apanovich, Pavel Vinokurov

EZOP – Web 2.0 service of ontologies' construction 47
Evgeny Beniaminov, Vladimir Lapshin, Dmitry Perov

Machine Learning Techniques Based on Rule Induction 57
Konstantin Vorontsov

Can Robots Solve an Assignment Problem? 72
Natalia Garanina

OLAP-modelling of Municipal Procurement Automation Support Problem 87
Anna Korobko, Tatyana Penkova

Specificity of Ontology Mapping Approaches 91
Nikolay Skvortsov

Survey of Modelling of Information and Material Resources by Means of Decision-Making 105
Gennady Sokolov

Author Index 112

Towards Ontology for Classification of Computer Languages[1]

Alexander Akinin[1], Nikolay Shilov[2], Alexey Zubkov[3]

[1]Novosibirsk State University
Pirogova str. 2, Novosibirsk 630090, Russia
akinin3113@gmail.com
[2]A.P. Ershov Institute of Informatics Systems
Lavrentev av. 6, Novosibirsk 630090, Russia
shilov@iis.nsk.su
[3]Novosibirsk State University
Pirogova str. 2, Novosibirsk 630090, Russia
ortoslon@gmail.com

Abstract. During the semicentennial history of Computer Science and Information Technologies, several thousands of computer languages have been created. The computer language universe includes languages for different purposes: programming languages, specification languages, modeling languages, languages for knowledge representation, etc. In each of these branches of computer languages it is possible to track several approaches (imperative, declarative, object-oriented, etc.), disciplines of processing (sequential, non-deterministic, parallel, distributed, etc.), and formalized models (from Turing machines up to logic inference machines). Computer language paradigms are the basis for classification of the computer languages. They are based on joint attributes, which allow us to differentiate branches in the computer language universe. Currently the number of essentially different paradigms is close to several dozens. The study and precise specification of computer language paradigms (including new ones) are called to improve the choice of appropriate computer languages for new Software projects and information technologies. This position paper presents an approach to computer languages paradigmatization (i.e. paradigm specification) and classification that is based on the unified approach to formal semantics, and an open ontology (wiki-styled) for pragmatics, formal syntax and informal "style".

1 The Problem of Computer Language Classification

We understand by a computer language any language that is designed or used for automatic "information processing", i.e. data (including process) representation, handling and management. A classification of some universe (the universe of computer languages in particular) consists in means for class(es) identification, separation and navigation.

[1] Research is supported by grant of Russian Basic Research Foundation 08-01-00899-a.

During the semicentennial history of development of programming and information technologies, several thousands of computer languages have been created[2]. Due to the number of the existing computer languages alone, there is a necessity for their systematization or, more precisely, for their classification. At the same time, classification of already developed and new computer languages is a very important problem for computer science, since it could benefit software engineering and information technology by a sound framework for computer language choice for components of new program and information systems.

At the initial stage of programming and information technology history (years 1950–65), it was possible to classify computer languages chronologically with annotations a-la Herodotus' "History", i.e. including lists of authors, their intentions, personal stories, etc. (Refer to [11] for a story of this kind.) The matter is that at the first stage all (almost) computer languages were languages of imperative programming for von Neumann's computers.

But since the late 1960-s the approach in style of the "Father of History" became unacceptable. Since this time the variety of computer languages included not only programming languages, but also specification languages, data representation languages, etc. Some of these branches since the late 1960-s include not only imperative, but also declarative languages (functional in particular).

Between the middle of 1970-s and the early 1980-s, new approaches to computer language design appeared (logical and object-oriented for example). Drawing an analogy between Computer Science and other sciences, we could say that since late 60s, the classification of computer languages could be done either in "Linnaeus style" (i.e. a taxonomy like: Kingdom – Phylum – Class – Order – Family – Subfamily – Genus – Species) or in "Mendeleyev style" (i.e. as a chemical periodic table where the rows represent periods, and the columns represent groups). For example, look at[3] *Taxonomic system for computer languages* at `http://hopl.murdoch.edu.au/taxonomy.html`.

However, the 1990-s and the beginning of a new millennium became the time of rapid growth of existing and new branches of computer languages. For example, knowledge representation languages, languages for parallel/concurrent computing, languages for distributed and multi-agent systems, etc. Each of these new computer languages has its own syntax (sometimes a very specific), a certain model of information processing (i.e. semantics or a virtual machine), and its pragmatics (i.e. the sphere of application and distribution). And though there were rather small groups of computer languages (e.g., Hardware Description Languages), many groups have been already "crowded" (e.g., Specification Languages) and some of them went through the period of explosion and migration (e.g., Markup Languages). Sometimes

[2] The *History of Programming Languages* poster by O'REILLY (http://www.oreilly.com/news/graphics/prog_lang_poster.pdf) is well known. The full-scale version of the poster is about 6 m long and contains 2500 entries. The chronology is represented by the temporal axis placed at the top of the poster, version history of individual languages are shown with colored lines, and the influences of programming languages are depicted by weak grey lines. Please refer Appendix B for scaled copy of the poster and zoomed fragment of it.

[3] Please refer Appendix C for scaled copy of the poster and zoomed fragment of it.

computer language "Gurus" have difficulties in putting some languages into the one definite paradigm[4] or to any paradigm (e.g. Language of Temporal Ordering Specification LOTOS, Business Process Modeling Notation BPMN). Rapid generation of new computer languages will continue while new spheres of human activities will be computerized. Thereby the situation in computer languages radically differs from that of the natural sciences: in biology or chemistry the situation is much more static, while in computer languages the situation is rather dynamic. Due to these arguments alone, the natural sciences analogies cannot be adequately applied to the classification of computer languages.

We define paradigms of computer languages as specifications of alternative approaches to information processing, accumulated and fixed in the form of computer languages. Computer language paradigms should base on joint attributes, which allow us to differentiate classes in the computer language universe. Paradigmatization is an approach to the specification of paradigms.

Let us remark that Robert Floyd was the first who had explicitly used the concept of "paradigm" in Computer Science, but in a different context and with another meaning. He discussed "paradigms of Programming" in his Turing Award Lecture in 1978 [5]. Floyd referred to Thomas Kuhn's well-known book [7], published just 8 years before. According to T. Kuhn, a paradigm is a method, an approach to the formulation of problems and the ways to solve them. R. Floyd had a similar understanding of programming paradigms. In particular, he wrote: "To the designer of programming languages, I say: unless you can support the paradigms I use when I program, or at least support my extending your language into one that does support my programming methods, I don't need your shiny new languages" [5]. Currently, the number of essentially different paradigms of programming is already several dozens (see, for example, the list of "programming paradigms" at [13]).

2 Introducing Syntactic-Semantic-Pragmatic View (approach)

For natural and artificial languages (including computer languages), the terms "syntax", "semantics" and "pragmatics" are used to categorize descriptions of language characteristics. Syntax is the orthography of the language. The meaning of syntactically-correct constructs is provided through language semantics. Pragmatics is the practice of use of meaningful syntactically-correct constructs. Therefore the approach based on "specific features" of syntax, semantics and pragmatics, could be natural for the specification of paradigms and the classification of computer languages.

The syntactic aspect of computer language classification should take into account formal syntax as well as the human perspective. Certainly, it is very important for the compiler implementation whether a particular language has regular, context-free or context-sensitive syntax. Thus formal characteristics of computer languages could be

[4] For example, programming language *Ruby*. "Its creator, Yukihiro "matz", blended parts of his favorite languages (Perl, Smalltalk, Eiffel, Ada, and Lisp) to form a new language that balanced functional programming with imperative programming" (http://www.ruby-lang.org/en/about/).

attributes for their classification. These attributes can be brought from formal language theory in accordance with Chomsky hierarchy or any other formal classification of formal languages. But the classification based on formal syntax has lost its original value by virtue of development of effective and powerful parsers. In contrast, informal annotations (attributes or characteristics) like "flexibility", "naturalness", "style" (supported by a library of good-style examples), "understandability" from a human standpoint (including a portion of "syntactic sugar") become much more important.

The role of formal semantics for the classification of computer languages is also well known. The major problem with semantics of computer languages is different formalisms and different level of formalization that is adopted for particular computer languages. In programming languages, for example,

- functional language LISP is based on very precise denotational semantics in terms of λ-calculus,
- the structured subset of a high-level imperative language Pascal has operational and axiomatic semantics [6],
- but formal semantics for imperative languages of C-family is still a research challenge [9, 10].

This difference makes it extremely hard to compare semantics of different computer languages. Nevertheless, we believe that sound unification of approaches to semantics is essential for better classification of computer languages and a proper paradigm definition. We think that the problem can be solved by two-dimensional stratification of paradigmatic[5] computer languages. Each of these languages should be stratified into levels and layers.

- Level hierarchy is the human-friendly semantic (and partially syntactic) representation. It should comprise 2-3 levels that could be called as elementary, basic, and full. Elementary level should be an educational dialect of the language for the first time study of its basic concepts and features. Basic level should be a subset for regular users of the language which requires skills and experience. Full level is the language itself, it is for advanced and experienced users.
- Layer hierarchy is the formal-oriented semantic representation. It could comprise up to 3 layers for basic level and (optionally) for other levels. These layers could be called kernel, intermediate and complete. The kernel layer should have an virtual machine semantics and provide tools for the implementation of the intermediate layer; the intermediate layer in turn should provide tools for the complete layer. Implementation of intermediate layer should be of semantics-preserving transformation. Please refer [9] for an example of 3-layer hierarchy for programming language C#.

In contrast to highly mathematical formal semantics, pragmatics relies upon highly informal expertise and experience of people that are involved in the compute language life cycle (i.e. design, implementation, promotion, usage and evolution). In other words, we need to formalize expert "knowledge" (views) about computer languages, related concepts, and relations between computer languages. It naturally

[5] We discuss what is "paradigmatic computer language" in the next section. To be short, paradigmatic languages are the most typical ones for a particular paradigm (class).

leads to an idea to represent this knowledge about computer language pragmatics as ontology.

"Ontology is the theory of objects and their ties. Ontology provides criteria for distinguishing various types of objects (concrete and abstract, existent and non-existent, real and ideal, independent and dependent) and their ties (relations, dependencies and predication)" [4]. Roughly speaking, an ontology is a partial formalization of a "knowledge" about a particular problem domain (computer languages for instance). This "knowledge" could be an empirical fact, a mathematical theorem, a personal belief, an expert resolution, a shared viewpoint of a group.

The most popular computer language for ontology representation is Web Ontology Language OWL. It provides an opportunity to use Description Logic (DL) reasoners for automatic consistency checking. Consistency is very important for open evolving temporal ontology. Openness means that the ontology is open for access and editing. Temporal means that the ontology change in time, admits temporal queries and assertions, and that all entries of the ontology have time-stamp. Evolving means that the ontology tracks all its changes. Wikipedia, the free encyclopedia [14], is a good example of open, evolving, and temporal ontology.

3 Towards Open Temporal Evolving Ontology for Classification of Computer Languages

Formalization of expert knowledge for pragmatics of computer languages should be open evolving temporal ontology that includes syntactic and semantic (formal and informal) knowledge in the form of annotations and attributes.

Let us remark, that the *History of Programming Languages* poster by O'RELLY (see Appendix B) already defines an ontology of programming languages in which the class differentiation and navigation method is explicit enumeration of languages, "influence lines" and chronology. The same holds for HOPL (History of Programming Languages at http://hopl.murdoch.edu.au/, see Appendix C). This project represents historical and implementation information about an impressive number (8512) of programming languages. Unfortunately HOPL is not open for editing, is not evolving since 2006, and does not deal with any other inter-language relations than language–dialect–variant–implementation. Situation is different with the Progopedia the wiki-like encyclopedia of programming languages (http://progopedia.ru/). It is open for editing, traces its history. But both HOPL and Progopedia does not deal with programming paradigms, have restricted temporal navigation. While the HOPL provides some taxonomy instruments, the Progopedia has only a trivial one (language–dialect–variant–implementation). In comparison with HOPL and O'REILLY poster, Progopedia is relatively small. At present it contains information about 51 language, 80 dialects, 187 implementations, and 485 versions. All these "ontologies" do not have means for constructing classes by users and support only manual navigation among the classes. We believe, that a more comprehensive ontology could solve the problem of computer languages classification, i.e. identification and separation of classes of computer languages and navigation among them.

In the proposed ontology for computer languages, objects should be computer languages (including their levels and layers also as objects), concepts/classes (in terms of DL/OWL) – collections of computer languages that can be specified by concept terms (in DL), ties (DL-roles or OWL-properties) – relations between computer languages. For example, Pascal, LISP, PROLOG, SDL, LOTOS, BPMN, UMLT, as well as C, C-light and C-kernel, OWL-Lite, OWL-DL and OWL-full should be objects of the ontology. Since we understand computer language paradigms as specifications of classes of computer languages, and we consider classes of computer languages as DL-concepts/OWL-classes, then we have to adopt DL concept terms as paradigms of computer languages. In these settings computer language paradigms and classification will not be taxonomy trees based on property inheritance from sup-class to sub-class. Objects (i.e. computer languages) of the proposed ontology could be attached by different formal attributes (e.g., formal syntax properties) and informal annotations (e.g., libraries of samples of good style).

Let us remark that the list of formal attributes and informal annotations is not fixed but open for modifications and extensions. Nevertheless it makes sense to provide certain attributes and annotations for all objects (e.g., an attribute “date of birth” with different time granularity, or annotation “URL of external link” for references) but allow indefinite value for them. In contrast, some other attributes and annotations will be very specific to objects. For example, “try” annotation with a link to easy to install or web-based small implementation (that can be freeware or shareware) makes sense for elementary levels or kernel layers.

We have already discussed a number of examples of concepts/classes in the proposed ontology: “has context-free syntax”, “functional languages”, “specification languages”, “executable languages”, “static typing”, “dynamic binding”, etc. (Other examples can be found at [17].) All listed examples should be elementary DL-concepts/OWL-classes. All elementary DL-concepts/OWL-classes should be explicitly annotated by appropriate attributes (“has context-free syntax”, “is functional language”, “is specification language”, etc.). Nonelementary concepts/classes should be specified by means that are supported by OWL and DL (by standard set-theoretic operations “union” and “intersection” in particular). For example, “executable specification languages” is the intersection of “executable languages” and “specification languages”. We have some doubts about “complement”, since the proposed ontology will always be open-world ontology with incomplete information. For example, if a language has no explicitly attached attribute “has context-free syntax”, it does not mean that the language has no CF-syntax. At present we adopt a temporary solution to use explicit positive (e.g., “has context-free syntax”, “is functional language”, “is specification language”, etc.) and negative attributes (that are counterparts of positive one, e.g., “has NOT context-free syntax”, “is NOT functional language”, “is NOT specification language”, etc.) and use of positive DL concept terms for paradigm specification (i.e. concept terms without complement).

But the proposed ontology should have a special elementary concept/class for “paradigmatic computer languages” that comprises few (but one at least) representative for every elementary concept/class. Of course, all elementary concepts/classes (including “paradigmatic languages”) should be formatted on base of expert knowledge and be open for editing. A special requirement for the proposed

ontology should be the following constraint: every legal ("well-formed") non-empty concept/class must contain a paradigmatic language (one at least). A background intuition is straightforward: if expert can not point out any representative example of a paradigm, then paradigm should be empty.

Roles/properties in the proposed ontology could be very natural also: "is dialect of", "is layer of", "uses syntax of", etc. For example: "REAL is dialect of SDL", "C-light is layer of C", "OWL uses syntax of XML". All listed examples are elementary DL-roles/OWL-properties. Standard (positive) relational algebra operations "union", "intersection", "composition" and "transitive closure" can be used and are meaningful for construction of new roles/properties. For example, "uses syntax of dialect of" is the composition of "uses syntax of" and "is a dialect of": REAL [8] executional specifications "uses syntax of dialect of" SDL [12]. Again we have some doubts about use of "complement" and "inverse" and, maybe, we will adopt use of explicit complement and inverted for elementary DL-roles/OWL-properties.

Let us remark that computer language domain has four domain-specific ties between languages: "is dialect of", "is variant of", "is version of", and "is implementation of" [15]. Of course these ties must be presented in the proposed ontology as elementary DL-roles/OWL-properties. But, unfortunately, there is no consensus about definition of these ties. For example, some people [18] consider that an implementation can have a version, while some other people [17] have an opposite view that a version can have an implementation. Detailed discussion of this issue is a topic for further research, but currently we adopt the following definition. Dialects are languages with joint elementary level. Variants are languages with joint basic level. Versions are linearly ordered collections of variants such that later earlier versions are compatible subsets of later versions. Implementation is platform-dependent variant of a language.

Universal and existential quantifier restrictions that are used in OWL and DL for construction of new classes/concepts also could get a natural and useful meaning. An example of existential restriction (in DL notation): a concept $\exists$ (uses syntax of : ((markup language) $\cap \neg\{XML\}$) consists of all computer languages that are markup languages but do not use syntax of Extensible Markup Language XML; an example of a language of this kind is LaTeX. An example of universal restriction and terminological sentence (in DL notation also) follows: a sentence XML: $\forall$ (is dialect of) . ($\neg\{ML\}$) expresses a fact that XML is a dialect of any computer language but a functional programming "Meta Language" ML.

We would like to emphasize that a proposed ontology for pragmatics of computer languages should be open evolving ontology. Openness of the ontology will be supported by wiki technology for editing. Evolution will be supported by the automatic timestamping and history of all edits. Temporality will be supported by temporal extensions of Description Logic for paradigm specification.

Recently we have started implementation of a prototype of a computer languages classification knowledge portal that eventually (we hope) will evolve into Open Temporal Evolving Ontology for Classification of Computer Languages (see Appendix A). We believe that it will provide Computer Language researchers with a sound and easy to maintain and update framework for new language design and language choice/selection tools for new Software engineers and Information Technology managers.

References

1. Anureev I.S. Ontological Transition Systems. Joint NCC&IIS Bulletin, Series Computer Science, 2007, v. 26. (Accepted for publication.)
2. Baader F., Calvanese D., Nardi D., McGuinness D., and Patel-Schneider P., editors. The Description Logic Handbook: Theory,Implementation and Applications. Cambridge University Press, 2003.
3. Borger E. and Stark R. Abstract State Machines: A Method for High-Level System Design and Analysis. Springer-Verlag, 2003.
4. Corazzon R. Ontology. A Resource Guide for Philosophers. At http://www.formalontology.it/.
5. Floyd R.W. The paradigms of Programming. Communications of ACM. v.22, 1979, p.455-460.
6. Hoare, C.A.R. and Wirth N. An Axiomatic Definition of the Programming Language PASCAL. Acta Informatica, 2, 1973, p.335-355.
7. Kuhn T.S. The structure of Scientific Revolutions. Univ. of Chicago Press, 1970.
8. Nepomniaschy V.A., Shilov N.V., Bodin E.V., Kozura V.E. Basic-REAL: integrated approach for design, specification and verification of distributed systems. Springer Lecture Notes in Computer Science, v.2335, 2002.
9. Nepomniaschy V.A., Anureev I.S., Dubranovskii, I.V. Promsky A.V. Towards verification of C# programs: A three-level approach. Programming and Computer Software. 2006, 32(4), p. 190-202.
10. Norrish M. C formalised in HOL. PhD Thesis. University of Cambridge, Computer Laboratory, Technical Report 453, 1998.
11. Ritchie D.M. The development of the C language. ACM SIGPLAN Notices, v.28, n.3, 1993, p.201-208.
12. Turner K. J. (Editor) Using Formal Description Techniques: An Introduction to Estelle, LOTOS and SDL. John Wiley and Sons, 1993
13. Programming paradigm. From Wikipedia, the free encyclopedia. At http://en.wikipedia.org/wiki/Programming paradigm.
14. Wikipedia, the free encyclopedia. At http://en.wikipedia.org.
15. Pigott D. HOPL: an interactive Roster of Programming Languages. 1995-2006. At http://hopl.murdoch.edu.au/
16. OWL Web Ontology Language Guide, W3C Recommendation, February 10, 2004. At http://www.w3.org/TR/owl-guide/
17. Kinnersley W. The Language List. At http://people.ku.edu/~nkinners/LangList/Extras/langlist.htm.
18. Progopedia. Encyclopedia Yazykov Programmirovaniya. (In Russian) At http://progopedia.ru/.

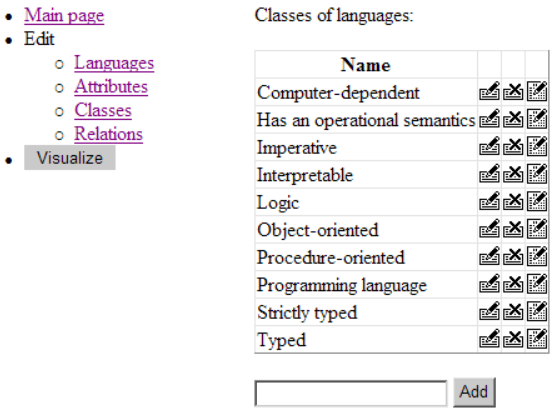
Appendix A: a prototype of a computer languages classification knowledge portal

A prototype of a knowledge portal for computer languages classification has been designed for small-scale experimentation purposes. So it does not support full functionality. The prototype is implemented as a web application, so "experts" (i.e. members of the laboratory) can enter it with a web browser. The interface allows users to observe and edit the information represented by the portal, which is formed as an ontology.

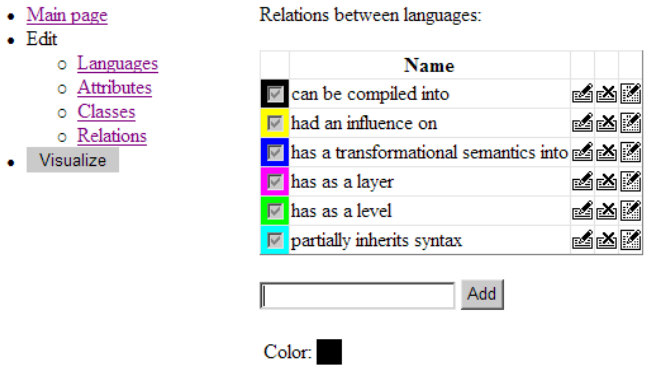
The main elements of the ontology are computer languages (objects of the ontology), elementary classes of languages (arbitrary, explicitly user-specified subsets of the set of objects), relations between the languages (binary relations over the set of objects), attributes (mappings from the set of languages to some external data types, e.g. text strings, URL's) and axioms (Description Logic statements).

Each of these types of entities form a list which can be viewed and modified directly by the user. The pictures below illustrate this process.

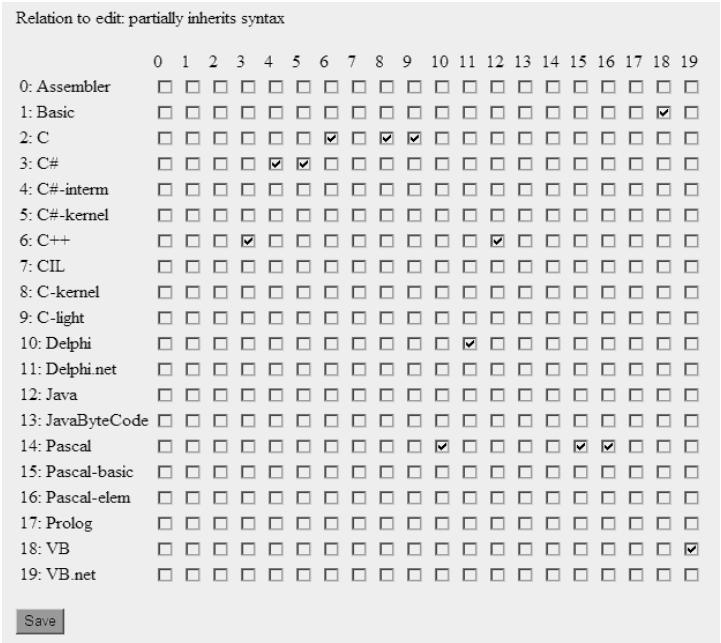
1. Example of list of language classes:



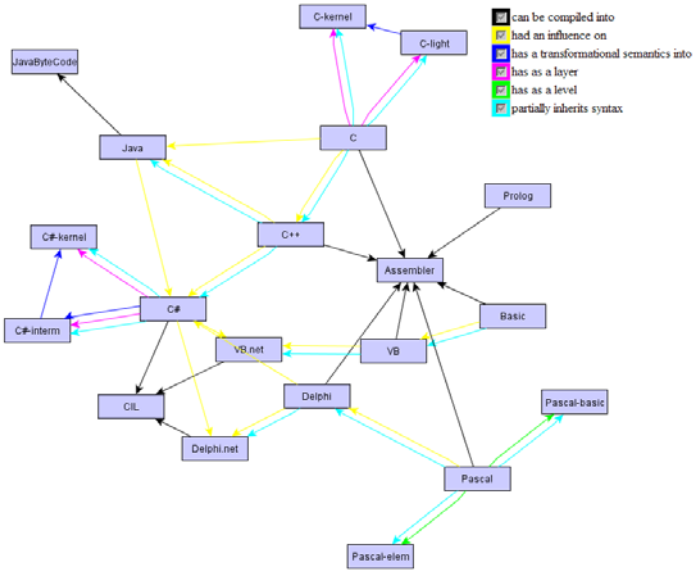
2. Example of a list of relations:



3. Example of relation editing:

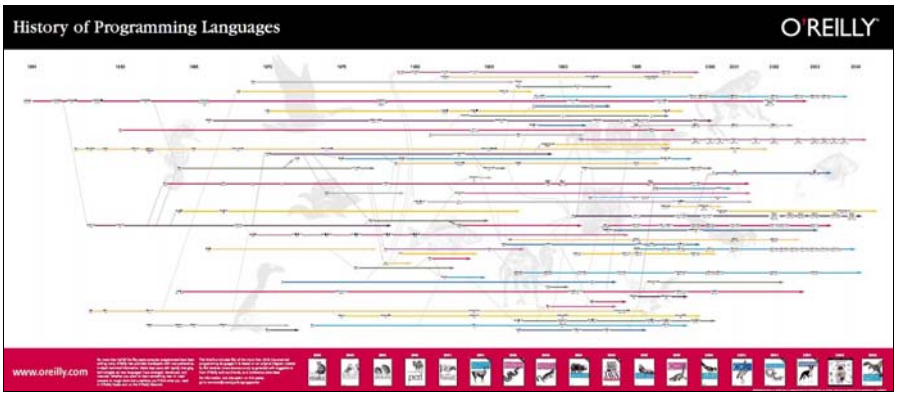


The portal prototype also allows to visualize selected relations between languages. It draws a graph where the vertices are computer languages marked by their names and the edges are the relations between the languages marked by their color.

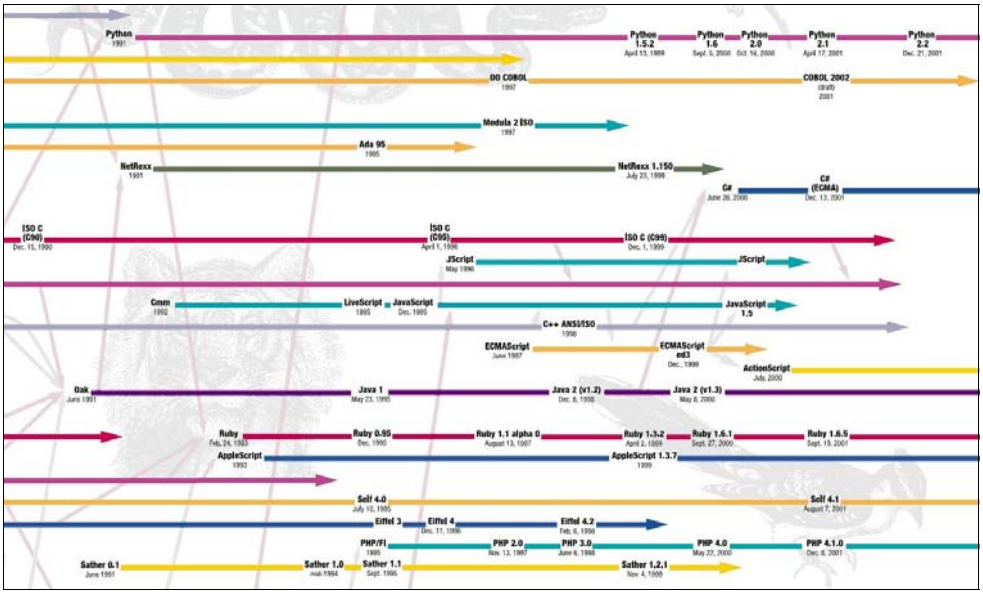


The data in the prototype is represented by an OWL-language database (RDF-repository). In future it will allow us to use some OWL-oriented reasoning tool to perform the consistency control and process the user queries.

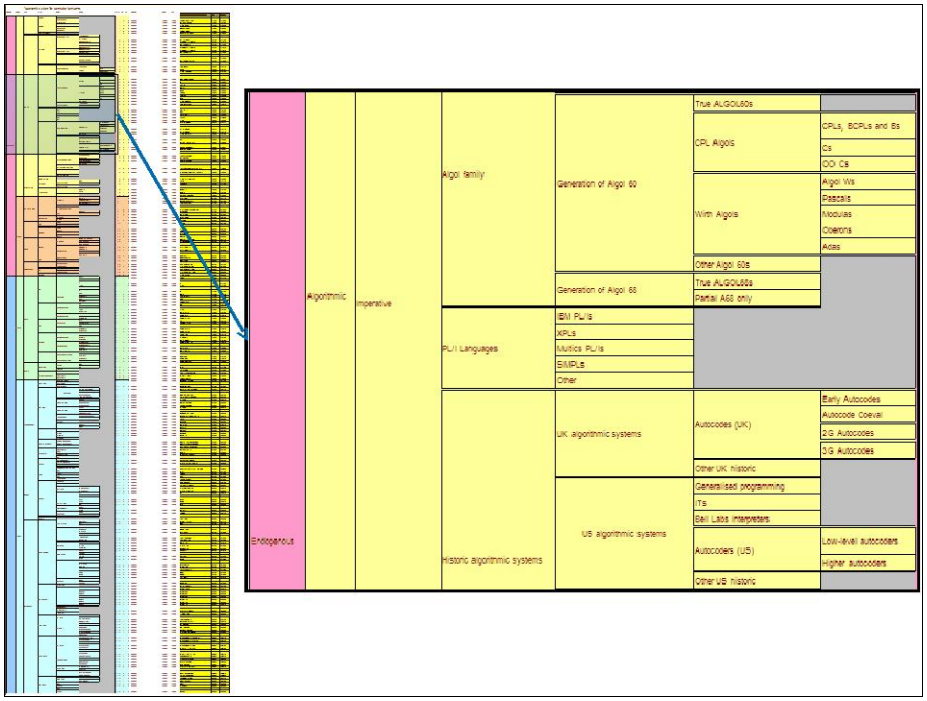
The prototype is at the beginning stage of development now. It is planned to expand its functionality by providing a more effective and reliable mechanisms to deal with big quantities of data elements and users, to solve complex queries.



Appendix B: History of Programming Languages by O'REILLY (http://www.oreilly.com/news/graphics/prog_lang_poster.pdf)



Appendix C: HOPL' taxonomic system for computer languages (http://hopl.murdoch.edu.au/taxonomy.html)



Ontology for Image Annotating

Michael Boldasov[1], Elena Sokolova[2]

[1] Luxoft Professional Ltd.
boldasov@nm.ru
[2] Russian State University for Humanities
minegot@rambler.ru

Abstract. In our paper we consider ontology that is used for semantic annotating of landscapes. Such annotations are high valuable for the task of image retrieval to allow their automatic processing. In article we discuss the main principles of organization of ontology for image annotating, consider its usage for the task of image retrieval, and describe our experiment to prepare a fragment of such ontology using SemTalk2 software tool.

Introduction

Last decennium finding an image suitable for a particular purpose received a considerable attention. In image retrieval textual indexes and annotations are used which are manually written or automatically found (or summarized from the texts) on the web. This approach is valid for retrieval of named entities in the images, especially of persons and toponyms. The method explores statistical salience of objects in the image and in the text and then mentioned and depicted objects are matched, for example (Deschacht, Moens, 2007). This approach is not valid in the case of art collections or photograph collections or cinema records where not named entities are presented, for example: "Mid shot a man walking between two lanes". In the latter case user can be interested in types of depicted objects, their features, composition and spatial relations not having in mind particular named entity. This approach to annotating can be named "knowledge-based" since the objects thyself should be described in some source of knowledge – knowledge base or ontology.

The system that implements the knowledge-based approach is supposed to generate texts of image descriptions automatically based directly on results of image recognition. Such system can be logically divided into two parts: Recognizer that prepares so called Image Models (Semantic Annotations that are based on concepts from ontology – results of Image Recognition), and Generator that prepares Image Description in a given Natural Language (NL). Prepared texts of picture annotations can be used further for sophisticated image retrieval.

The goal of current research is to model the Generator component of such system. Since we have no valid algorithms of Image Recognition that can automate recognition process for such system, we need to invent the main principles of composition of Image Model that is the main point of attention in current article. Knowledge-based methodology of representation an Image Model is already tried by

(Hollink et al., 2003), where it is described, how to create a formalized model of a picture using the existing Knowledge Bases (KB) and lexico-semantic databases.

Our experimental materials were images from the collection of photographs of Prokudin-Gorsky (www.prokudin-gorsky.ru – colored photographs, dated from the period of 1900-1917), and associated texts of image descriptions, manually created by students of the Russian State University for Humanities that describe just what is depicted in the photos. It is planned to prepare the first working prototype of the Generator system to solve the task of multilingual image retrieval above resources of that collection. For a moment we have a working dummy prototype of NLG system that can prepare basic texts of image descriptions in English and Russian languages from Image Models prepared by hand, though some questions of organization of Image Model are still open. Image Model is not semantic representation or text plan. It has visual nature and can serve as input for the text planning component of generation model.

In section 2 we consider the notion of image model. In section 3 we consider the types of knowledge and sources of knowledge and sketch the contours of Ontology unifying visual features and ontological features of visible objects and their nominations in concrete natural languages. In sections 4 and 5 we discuss subject matter description and Ontology. Section 6 describes our experiment to formalize Ontology and use it for the image models creation by means of SemTalk2 project. We discuss there the problems we have met during the experiment. Finally section 7 presents conclusions of the performed research.

1 Image Model

We consider the Image Model (Semantic Annotation) as a set of Objects with particular characteristics, and a number of relations between objects. Domain Model of all possible Objects and Relations is provided by Ontology specially organized for the current research. This is the point that differs our approach from the approach based on using for Image Model of existing sources like ontologies, thesauri, databases (DB) or knowledge bases. For example, in (Hollink et al., 2003) knowledge Sources for Image Model are presented by WordNet and some other linguistic ontologies and art databases. Preparation of annotation is based in this approach on linguistic information – triads (agent-process-object) and settings (time, location and artist) (Tam et al., 2001), (Hollink et al., 2003). Using that approach for our task, we would have two problems:

1. Differently to objects, actions have no area on the surface of photograph, they are only indirect knowledge that ought to be inferred from location of objects, postures and gestures of humans and animals;
2. Linguistic ontologies and art databases are not developed to present visual information.

Therefore we need a special Ontology to present ontological and visual features of objects. Image Model is amount of concepts of depicted objects related by spatial and ontological relations.

2 Corpus and Annotation

We use the corpus of photo descriptions as a source of information about presented objects and their composition. From 1902 photos in the web-site of Prokudin-Gorsky collection we considered 100 ones to include to our corpus. NL descriptions of these photos were made by students of the Russian State University for Humanities in Moscow. Every photo has two or more descriptions that are made by different students. Prepared descriptions contain approximately 2 to 15 lines of texts. We consider NL descriptions as some result of analysis of visual information in the picture. The descriptions are used to explore the following things:

– What objects and relations were noticed by author, hence what concepts ought to be in our ontology and what relations will form the Image Model composition;
– How the objects are expressed in the text. We use these wordings in our "concept-Russian" dictionary;

Photographs of concerned collection contain few objects and sometimes they are not of high quality. Here is description of photo 00957:

В сельской местности на возвышенности, среди *вековых* ***елей и сосен расположена*** *уютная* ***небольшая часовня. Сверкает на солнце ее серебряный купол, двери ее распахнуты, а на ступенях примостился одинокий посетитель,*** *пришедший с прилежащей пашни. Возведена часовня по особому случаю:* ***ее крыльцо обращено к старой сосне,*** *в ветвях которой жителям прилежащей деревушки явился образ иконы Божьей Матери. И сейчас* ***мужчина, сидящий на ступенях****, вглядывается в мохнатые ветви сосны, надеясь вновь увидеть чудесное явление.*

Figure 1. Materiki. Chapel of the Mother of God and the pine tree on which the icon appeared ¶

In this description (and in its gloss[1]) bold are words describing just visual objects and their features and relations which ought to be included into the Subject matter annotation. The rest of the text concerns knowledge, hypothesis, and impressions of the author. Many photos present one focus object – a chapel, a church, a mill, one or two people, a flower bush etc. Most of them are landscapes. There are also multiple-object pieces, for example, "Cornflowers among the rye". There are also genre-pieces,

1 In a country land on a hill between secular spruces and pine trees a little snug chapel is situated. Shining in the sun its silver cupola, doors are broken open and on the stairs a someone attendant nestles on the stairs came from the proximate tillage. The chapel is build on the occasion of a particular case: its porch exposes to the old pine tree in which an image of the Madonna icon turned up to the residents of the proximate village. And now the man sitting on stairs peers to the pine brunches hoping to see the miraculous icon again.

for example, “Peasants on a hay field”, “Shashlik stand” and some images of icons and stained-glass. In this paper we explore landscapes.

Proposed in (Schreiber et al., 2001), general structure of the photograph annotation consists of 4 parts:

- Photo features (Vantage point, Photographer, Exact location, Exact time);
- Medium features (Resolution, Format);
- Settings (Terrain type, Time of day, Season);
- Subject matter description (Passive agent, Active agent).

First two types of information can be retrieved from the bibliographic and medium information of photos in the web-site (e.g. exact location, exact time), from Section of the Collection and sometimes from the name of photo, for example, “The town of Kirillov. View from Maura Hill”.

Since we consider the paradigm of image recognition, Settings are not considered as disclosed from the Subject matter description, for example, Terrain type – hill, field and so on. Other types can be result of information inference, for example, Time of day ← light sky, the sun in the sky, Season ← color of grass or leaves in the trees.

We use the students’ descriptions to explore the types of information they see in the photos. We have found one more type of information there – impression, for example, “photo in blue keys”, “very calm” and so on – hence we add “Art features”. So we have 5 parts of annotation structure:

- Photo features (Name of photo, Vantage point, Photographer, Exact location(Toponym), Exact time));
- Medium features (Resolution, Format and so on));
- Settings (Time of day, Season, Weather);
- Art features (Color key, Impression);
- Subject matter description (Objects and their features, their spatial and composition relations).

3 Subject Matter Description

Description template that was proposed in (Tam et al., 2001) and used in (Schreiber et al., 2001), (Hollink et al., 2003) consists of triads (agent, action and object) and settings. It presents interpretation of image in terms of actions. Usage of features of subject matter as acting agents is absolutely problematic for our corpus since our images are static, and they have usually no action at all. The only types of physical processes that are mentioned in the names of photos are like the following: “Rafts sitting on the rocks at the village of Kurya”.

Ontologies and KBs are the sources of knowledge about the objects – names, membership, parts, process participants and so on. Photo itself includes composition of objects and particularities of every object (its parts and features). These relations cannot be directly presented by grammatical relations “common to many languages” (Tam et al., 2001). We need spatial relations and composition relations.

Composition relations are presented by single relation INCLUDES**.** The surface of photo is divided into areas that are formed by boundaries of the depicted objects. Area of one object can enclose area of another object, for example, (SKY ((FLYING-BIRD

(WINGS)) SUN)). Objects that are inside the area of another object are included in it. To describe relations of partly included objects, in particular, OBJECTS standing on GROUND, we need to use our knowledge about what is "standing on the ground and what object can play the role of localization".

Describing a picture in NL we often divide it into layers – groups of objects that are in approximately equal distance from the viewer, for example, "*in the foreground we see a group of people, on the back – a street*". We use the concept LAYER to present these groups of objects. Practically people use from 0 to 2 layers ("foreground" and "background") but generally picture description can contain as many layers as it is needed for narration (foreground, second plan, third plan, … background).

Objects whose areas are not intercrossed or that are intercrossed not as "an object standing on the GROUND" are related by spatial relations, which are usually bidirectional, for example:

¶"To-the-left(X, Y)",
"To-the-right(Y, X)";
"Near(X, Y)",
"Near(Y, X)";
"Around(Y, X)",
"In-the-centre(Y, X)", etc.

¶We consider IM as a kind of visual specification with elements of semantic interpretation. Both can be of different level of discriminating – general vs. more detailed description. Concepts of IM are described in ontology where they are supported by the above mentioned concepts and semantic concepts which can be used for their further interpretation, for example, relations as "Part", "Super class".

4 Ontology

We define **classes of objects** (their types) and design them by English words, for example, HOUSE, TREE, CHAPEL, SMOKE-STACK, BRANCH, FIELD, WINDOW, and so on. In Image Models we use instances of classes and their specific features (color, size, shape etc.) that should be added to the models. So we need to have in our ontology concepts of visual parameters and their **meanings**, for example, COLOR: BLACK, RED, etc.; SIZE: BIG, SMALLinWIDTH, BIGinHIGHT, etc.; SHAPE: SQUARE, ROND and so on.

For the needs of NLG we also need superclasses which would be presented by hypernyms for the names of classes. They can be used in the situations of visual haziness or for the second mention of the same entity in the text. For example, for classes EDIFICE, CHURCH, CHAPEL, HOUSE, BARN and CABIN valid superclass can be BUILDING, which is related with them by **is-a relation**. Hypernyms can be extracted from dictionaries or ontologies, for example, from WordNet. But WordNet descriptions themselves are not valid for our purpose since they are mostly functional descriptions, for example:

– BUILDING is-a "a structure that has a roof and walls and stands more or less permanently in one place";

– WALL is-a “partition, divider (a vertical structure that divides or separates)”;
– DOOR is-a “movable barrier (a barrier that can be moved to allow passage)”.

We can also use ontological relations “part holonym” – “part-meronym”. For classes WALL, ROOF, WINDOW and DOOR (through DOORWAY) part holonym class is BUILDING, EDIFICE. For BUILDING among its part meronyms are WALL, ROOF, PORCH, and FLOOR. So we need **Classes** and **Superclasses** that correspond to the “is-a” relation, one **Composition relation** “part” that has two terminals - part-holonym and part-meronym, and a number of **Spatial relations**.

Formalization of concepts and relations that are used in image model is obligatory requirement because the prepared image model should be processed automatically. So we need to add to ontology the complete set of classes, whose instances can be met in described photographs, supporting them with all possible visual parameters[2] and ontological features, and to add to ontology descriptions of all possible relations between instances of the classes. On the other hand, the resulted ontology should not be too “heavy”, i.e. it should be reasonable easy to use it for composing IM. So, it is not a good idea to make it possible e.g. to choose a visual parameter of an object from the whole set of visual parameters of any class, or to choose possible relation from the whole set of relations between any of classes from ontology. Thus we need to invent a kind of filter that controls that the proper object is supported with a proper set of visual parameters and relations.

Such filtering presents description of subject matter as information, which is prepared for communication, where every object is presented in some **cognitive perspective (CP)**. CPs are the containers of visual parameters and participants of relations. Class consists of one or several CPs, e.g. class RIVER consists of aspects SURFACE, VALUE and MIRROR. Class instance can be assigned in image model manually with an extra CP if it performs not typical role in a picture, e.g. if a SCARF is used as a SKIRT we need to combine in the instance of image model these two CPs both.

The described above ontology can be used in two paradigms: image recognition and NLG of picture descriptions. Here we pay attention only on NLG paradigm. The objects and their parts are given us in photo. For recognition process we need to know what object can include as its pats and we need defeasible reasoning[3].

5 Experiment

It was decided to realize ontology in terms of OWL (Web Ontology Language) because it is based on paradigm of XML – so it is very convenient for machine processing. Moreover OWL is specially prepared to describe ontologies.

[2] Visual parameters can be used in object recognition. We do not discuss this type of information in the paper.

[3] Image recognition paradigm needs more information and defeasible rules, for example, to recognize an object as a BARN we need information that its image is similar to the image of a “building for men enter in it” (building, edifice) and that “barn” usually has no “window” or has a small one in difference to “house” and is bigger than a cabin.

It is a very attractive idea to use one (possibly visual) environment for distributed management of ontology as well as for preparation of particular image models – so we tried to find appropriate tool trough Internet. The standard solution to manage OWL descriptions by Altova SemanticWorks failed, because the system is not optimized enough to specifics of our particular task.

As a real candidate of desired environment we have tried SemTalk2 that has a user-friendly editor for Semantic Web Ontologies and Processes. SemTalk2 is based on Visio Diagrams that are used to introduce any new object to the ontology or image model. It supports two types of diagrams: Class Diagrams – for description of ontology, and Instance Diagrams – for description of image models. Maintenance of ontology is supported not only through Diagrams but also through hierarchic View that is convenient for usage. Moreover SemTalk2 supports concept of distribution of development of ontology and image models – it allows importing as many external models as you like.

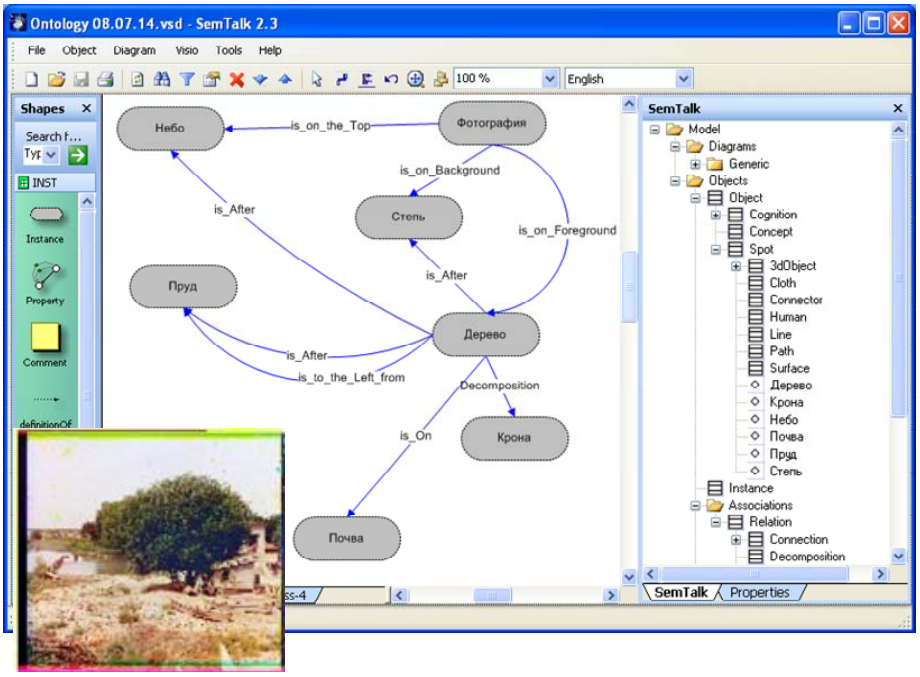


Figure 2. Screenshot of the opened IM in SemTalk2 application, supported with the photo 00039 that is described on the opened IM. IM is displayed on the diagram on the center if the screen; KB that was used to construct IM is displayed in the hierarchy to the right of the diagram

Screenshot in **Fig. 2** displays an opened image model in SemTalk2 application. Image model that is displayed on diagram in the center of the screen is prepared based on two student's descriptions of photo 00039:

– “На фотографии изображено большое зелёное дерево на фоне чистого неба. Дерево находится на каменистой почве. Стоит ясная погода. Справа от дерева виднеется пруд”
– “Центр фотографии занимает раскидистое дерево с широкой кроной. На фоне – степь. Левее и чуть дальше дерева – небольшое озеро. Примерно две пятых фона - белое небо” (see the English translation of these descriptions [4])

Ontology that was used to prepare the IM is placed on the right part of the Screenshot in **Fig.2**. It is a hierarchy with possibility to drag-and-drop desired classes to the diagram of image model.

Evaluated application SemTalk2 satisfies to our task much better than Altova SemanticWorks. But using the free version of SemTalk2 that is available in Web we have met the following lacks of the application:

– The application is not enough bug-fixed. When we tried to use it not by scenario that was described in user guide, we got errors and data inconsistency.
– The idea of Cognitive Perspectives is not supported.
– The displayed name of instances is an identifier that is obligatory and should be set by user. And from our point of view it would be better if an Individual was displayed by name of its Class (or CP!) and all its valued parameters, and its identifier is set automatically and it is not obligatory to be displayed.

Another possible decision is to use Altova SemanticWorks for management of ontology, and to use our own XML-based notation to describe Image Models, which will be optimized for the task of NLG.

Conclusion and Further Research

In our paper we sketched a method of semi-automatic Image Model creation. We discuss theoretical issues of interaction between visual data and human knowledge, as well as appropriate theoretical aspects that should be used as a base for ontology and IM. Prepared ontology and IM can be used:

– in the process of image recognition providing information about composition of objects in the picture and their ontological relations,
– in the images retrieval, for example, in art collections and other sources,
– in NLG of picture descriptions.

In our experiment we have created a fragment of ontology and used it for construction of some Image Models by means of the SemTalk2 system. The experiment showed that the system is valuable for this purpose in general, but the free version has some lacks for our task.

[4] Here is the English translation of the descriptions:

- “The photo shows a large green tree on background of pure sky. The tree is located on stony ground. The weather is fine. To the right from the tree there is seen a pond.”
- “The center of a photo borrows a sprawling tree with a broad crown. On background is steppe. More to the left and a little further of a tree is a small lake. About two-fifths of the background takes a white sky.”

Our previous experience shows that transformational approach to text planning from some specifications is possible (5, 6) but it is not the best method of NLG. Our experience in systemic functional modeling in KPML for the AGILE system (8) can be used and presents the area of further research.

References

1. 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
2. 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. Также доступна в интенет http://www.cs.vu.nl/~guus/papers/Hollink03b.pdf
3. 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
4. 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.
5. 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. Pp. 33-40.
6. 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. Pp. 465-476.
7. Hunter J. Adding multimedia to the semantic Web – building an MPEG-7 ontology // Proceedings of International Semantic Web Working Symposium (SWWS), Stanford, July 30 - August 1, 2001 http://archive.dstc.edu.au/RDU/staff/jane-hunter/semweb/paper.html
8. G-J. Kruijff, J. Bateman, E. Teich, S. Sharof, L.Sokolova, I. Kruijff-Korbayovб, H. Skoumalovб, K. Staykova, J.Hana, T. Hartley. Multilinguality in a text generation system for three Slavic languages // Proceedings of the 18th conference on Computational linguistics - Volume 1, 2000, Saarbrьcken, Germany July 31 - August 04, 2000. Pages: 474 - 480

Categorizing Knowledge Patterns into Ontological Framework

Tatiana Gavrilova[1], Vladimir Gorovoy[1], Elena Petrashen[2], Dmitry Mouromtsev[2]

[1] Saint-Petersburg State University, Graduate School of Management, Information Technologies for Management Dpt. 199004, Volkhovsky, per. 3, St. Petersburg, Russia
[2] Saint-Petersburg State Polytechnic University, 29 Polytechnicheskaya str. St. Petersburg 195251
tgavrilova@gsom.pu.ru, gorovoy@gsom.pu.ru, elena.petrashen@gmail.com

Abstract. The paper describes the ontological approach to the knowledge categorization and structuring for the e-learning portal design as it turns out to be efficient and relevant to current domain conditions. It is primarily based on visual ontology-based description of content of the learning materials and this helps to provide productive and personalized access to these materials. After describing the experience of ontology developing for Knowledge Engineering courses based on educational course for both undergraduate and master students of Saint-Petersburg State Polytechnic University and Saint-Petersburg State University we will also mention "OntolingeWiki" tool for creating ontology-based e-learning portals containing different knowledge patterns.

1 Introduction

As e-Learning is a cross discipline artefact that spans sociology, philosophy, psychology, pedagogy, artificial intelligence and human computer interaction, a technical solution which is a result of a system development should be carefully processed from the point of view of mentioned disciplines. The society becomes more and more visually dominated[17], new economy requires efficiency, just-in-time delivery and task relevance – this is why educative systems need to become more up-to-date, flexible and adequate. And often pedagogical and psychological construction and delivery of contents are even more important than the actual content.

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

However, the importance of specification and structuring the content and its visual presentation – followed with such connected issues as design, adaptation and usability has been underestimated to a certain extent until recent times as the researchers were far more concerned about how to educate (with methods of instruction or reasoning over the content) than how to present the object of the research (content specification and knowledge structure) [6]. So constructing ontologies to form content and/or navigation system, improving navigation usability and level of knowledge acquisition 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 categorization of knowledge patterns and ontology developing for Knowledge Engineering courses based on educational course for undergraduate Computer Science students of Saint-Petersburg State Polytechnic University. And firstly it seems important to discuss related work and specify why ontology-based conceptual domain modeling is so efficient for needs and purposes of e-learning using researches applying our own considerations.

Next section gives details of goals we wanted to achieve while starting to develop the ontology and also presents its top level. In the fourth section we describe OntolingeWiki, ontology-based tool for creating educational portals. The paper concludes with summary and future work discussion.

2 Ontological Framework: A Brief Overview

In the context of e-learning the conceptual structure of the content is an important part of the learning material. It helps learner to integrate the concepts that he/she is trying to learn, understand them and include them into his/her own image of domain. This is where the Semantic Web principles are vital [4] and ontological reasoning is a promising tool to provide a formal description for a shared domain conceptualization [15]. Conceptualization also select which things are relevant to be represented and which are not, so construction of top ontology leads to expressing knowledge [2].

A formal representation of a set of concepts as a visual presentation of course's structure has several advantages in e-learning. By using an ontology, the relationships between entities can be more clearly expressed and it allows better reasoning, 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]. Also two of the most current research issues in the e-learning community are specifying reusable blocks of learning content and defining an abstract way for designing different units (e.g. lessons) [7]. Despite the fact that many e-learning systems exist and keep being developed, knowledge reuse from one system to another is almost non-existent. Content-oriented view could facilitate knowledge sharing and reuse [9].

It also provides the ability to modify easily the course's structure for different educational purposes adapting to current needs, to utilize existing concepts defined in alternate ontologies. Just like we can build castle for a princess and space atomic Coca-Cola factory using the same Lego blocks. So it will be possible to use different structures to overcome the one-size-fits-all approach and personalize learning process as not only do learners are of varying degrees of proficiency and professionalism, but their goals, expectations and techniques and motivations for learning are also different.

Ontologies are also good to make domain assumptions explicit so that it would be possible to change these assumptions easily if knowledge about the domain changes [12].

Besides, the web itself is a very large scale hypertext information space where users can search and find information in different domains. But considering the constant increasing of resources on the Web getting an overview of all the available information relevant to their current needs and tasks becomes pretty much impossible. And if this user is not fully experienced in the knowledge domain as it happens in the majority of cases he or she is not totally able to define for sure whether found content is entirely appropriate and useful for their cognitive state, tasks and level or accurate, conforming exactly to truth or to a up-to-date standard [1]. Within the class of Web-based educational systems, a major role in various instructional contexts play the Educational Information Systems that are aimed at providing intelligent, task-centered information support for solving problems and performing learning tasks. And ontology is really good for maintaining functionality required for those ones [6].

A visual presentation of course's structure which is presented by a top level ontology permits also to boost quality of learning process. As two major problems perception in e-learning are loss of overview due to low information density of the medium and short attention spans due to fast fatigue of perception structure and presentation of learning material it will be a lot more effective if it reflects the characteristics of hypermedia and the web [11]. It's simpler and more effective to perceive and analyze domain knowledge when it's clearer how all small pieces of information connected between each another are. It can be compared with having a picture of a complete puzzle before and while inspecting each of the separated pieces. It's easy: we know how this piece is connected with others, what's pictured on it and how we can use it. But when the puzzle is undone and the only thing we know it's that this is a picture of a pretty blonde princess, a lot of time can be spent while deciding whether that little blue piece is sky or princess's dress.

3 Knowledge Engineering Ontology

Our first step was creating a visual representation of the top level of ontology as a powerful mind tool in structuring process. Its concepts are tied with self-contained chunks of training content – learning objects. This targets reusability since these learning objects can be reassembled later to create other courses just as well to personalize this one for different types of learners (e.g. for different proficiency degrees students etc). Visual form influences both analyzing and synthesizing procedures in ontology development process [16]. Also this is important as IES require unambiguous and complete learning design [14]. The developed ontology will be also used as a table of contents for educational system.

The required qualities for the ontology to be used as a domain knowledge representation model are:

- Accuracy
- Completeness
- Cognitive adequacy [5]
- Conceptual balance [15]

It's also important to exclude excessiveness and contradictions and to avoid it being too complex and/or big. The solution how to combine first and last demands is

to make it more scalable – e.g. when user chooses the leave of top level ontology it proves to be the root of another one – the screen is not overloaded so the information is perceivable yet all the advantages of using ontology are kept.

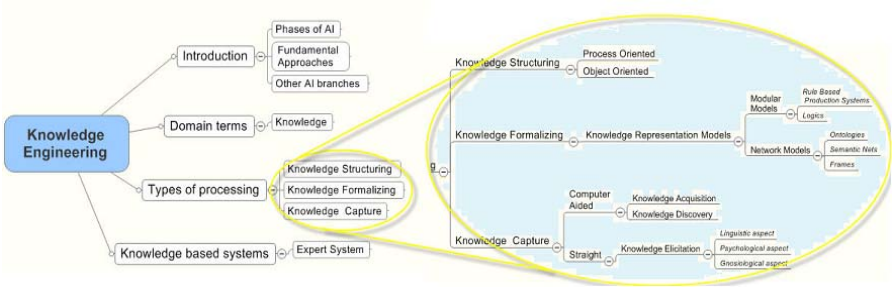


Figure 1. Top Levels of the Educational Ontology of Knowledge Engineering (nodes can be opened to present the nested data)

4 OntolingeWiki – technology for creating e-learning systems

It seems important not only to present information as a structure and mutual dependence of a set of learning objects but also to give learners opportunity not only to consume it, but to create. Often students are just presented with some kind of well-structured material which could be supposed to prepare them for a standardized test, however, new and interesting techniques for collaboration and conceptual modeling are not used [11] .

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 take any ontology saved in OWL format as an input and provide web-interface for ontology navigation with visualization based on hypergraph 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 on the base of Ontolinge-KAON system [3].

This technology can be used for creating ontology-based educational portals and was successfully leveraged in the design of the ontology-based content management system for the virtual exposition of the optical technologies museum in Saint-Petersburg State University of Information Technologies, Mechanics and Optics. Many electronic teaching materials such as presentations, animations or java-applets were united in the virtual exposition which introduces a visitor with optics according to the chosen ontology model.

5 Summary and Future Work

The main purpose of modern development of e-learning is to open up, share and reuse educational systems' content and knowledge components. It has to be effective, well designed, not expensive and fast in development.

The domain ontology proposed in this paper has a wider objective to provide a framework to describe the course as ontology-based conceptual modeling makes content more comprehensive for an individual and technologically it provides concrete and stable database. Knowledge categorization and laddereing permits to boost quality [12]. But the usability and appropriateness of this ontology should be further investigated and refined accordingly.

We observe the developed ontology as a promising starting point towards achieving a fully functional adequate and up-to-date computer-based educational system where it's going to be used as a domain knowledge representation model.

We consider OntolingeWiki tool a step forward in creating a useful technological environment for creating ontology-based educational portals supporting collaboration.

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 Intelligent Educational Systems, Le Mans, France (2009)
3. Cristea A.: What can the Semantic Web do for Adaptive Educational Hypermedia? International Peer-Reviewed On-line Journal "Educational Technology and Society" 7(4), Special Issue on Ontologies and the Semantic Web for E-learning, IEEE, LTSC, pp 40--58 (2004)
4. Gorovoy V., Gavrilova T.: Technology for ontological engineering lifecycle support. In: International Journal "Information Theories & Applications", Vol.14, pp. 19-25. (2007)
5. Guarino N., Schneider L.: Ontology-Driven Conceptual Modelling. 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, Le Mans, France (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, Technical Report KSL-01-05 (2001)
13. Prensky M.: Digital Natives, Digital Immigrants. NCB Univ. Press, Vol. 9 No. 5 (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 // Information Theories & Applications – 13 (4), pp.. 303–308. (2005)
17. Trifonov T.: Cultural Heritage Studies – the Need for a New Integrated Educational Approach based on Visual Comparative Interpretation of the Cultural Past, www.ncd.matf.bg.ac.yu/casopis/04/d013/download.pdf (2004)

Управление справочными данными и технологическими знаниями в промышленности
Management of the Standard/Reference Data and Technological Knowledge in the Industry

А.Н. Андриченко, Н.А. Щербаков
A. Andrichenko, N. Scherbakov

АСКОН, Москва
andrichenko@asconm.ru, scherbakov@asconm.ru

Аннотация. Ведение нормативно-справочной информации (НСИ) – одна из ключевых задач в обеспечении работы современного предприятия. Корпоративные информационные системы конкурируют между собой за обладание источниками НСИ. Комплексное решение проблемы возможно на основе объектной модели данных, позволяющей хранить информацию об объектах НСИ вместе с правилами их взаимодействия. Возможности данного подхода: описание знаний – семантических связей между объектами; доступность объектов справочных данных всем прикладным системам по единому протоколу; согласованное хранение онтологий предметных областей всех приложений; варьирование содержания и объема доступных справочных данных в зависимости от контекста их использования. Перспективы: повышение уровня автоматизации принятия технологических и управленческих решений. Имеется практический опыт реализации описанных принципов в серийном программном продукте.

Ключевые слова: справочники, классификаторы, управление знаниями, НСИ, объектная модель, поддержка принятия решений, семантическая сеть, онтология, контекст, MDM, Master Data Management, Semantic Web.

1 Введение

Справочники и классификаторы средств производства, материалов, товаров, работ, а также система правил взаимодействия этих объектов – основа для принятия технологических и управленческих решений на производстве.

Автоматизируя отдельные направления своей деятельности, многие предприятия параллельно эксплуатируют системы от различных поставщиков, как глобальные (ERP, PLM), так и нишевые (CAD, CRM и пр.). Каждое из этих приложений использует свою собственную модель данных, свой набор справочников и классификаторов.

Такое многообразие порождает следующие трудности:

- необходимость сопровождения одновременно нескольких справочных баз данных и повышение стоимости владения ПО за счет дублирования работ;
- дублирование одной и той же информации в различных средах и возникающие из-за этого несогласованность и избыточность справочных данных;
- большая трудоемкость и нетехнологичность операций синхронизации справочных данных между взаимодействующими приложениями;
- отсутствие единого централизованного хранилища справочных данных, обеспечивающего отраслевые и корпоративные стандарты именования, атрибутирования и классификации объектов;
- отсутствие целостного взгляда на корпоративную информацию, отражающего все аспекты бизнеса предприятия.

Таким образом, раздробленность справочных данных представляет собой серьезную проблему для многих предприятий. Наиболее естественный и надежный способ решения этой проблемы – построить на предприятии единую централизованную систему управления НСИ.

2 Решение: единые справочники и знания

Несколько лет назад появился класс систем, позволяющих решить ряд перечисленных проблем. *Master Data Management (MDM)* – совокупность методологий и инструментов, специально предназначенных для управления НСИ. Данное направление активно развивается и на сегодняшний день является одним из самых перспективных в мировой ИТ-индустрии. Объем рынка MDM, по данным ведущих аналитических агентств, составляет порядка миллиарда долларов и имеет тенденцию к интенсивному росту.

Дальнейшее развитие идей MDM – *система интеллектуального управления НСИ*, базирующаяся на следующих принципах:

- *консолидация НСИ* – объединение всех справочных данных в единую информационную среду;
- *объектная модель данных* – хранение справочных данных и обмен ими в виде информационных объектов;
- *онтологическое представление НСИ* – использование семантических моделей предметных областей для хранения объектов НСИ;
- *контекстность видения НСИ* – представление объектов исключительно в связи с определенной точкой зрения на их состав и взаимосвязи;
- *ориентированность на знания* – перенос знаний (правил взаимодействия объектов) из бизнес-логики приложений в объектную базу НСИ.

2.1 Консолидация НСИ

Проблемы, непосредственно связанные с дублированием и несогласованностью справочных данных, решаются созданием единого пространства справочных данных, одинаково доступного всем прикладным системам.

2.2 Объектная модель данных

Объектная модель данных – фундамент интеллектуальной MDM-системы. В отличие от реляционной модели, она позволяет хранить информацию об объектах вместе с правилами их взаимодействия. MDM-система является поставщиком объектов НСИ для любых специализированных внешних приложений, способных поддерживать принятый протокол обмена данными.

Применение объектной модели избавляет пользователя, ответственного за ведение НСИ, от необходимости разбираться в базе данных на физическом уровне. Конфигуратор модели позволяет пользователю оперировать общедоступными понятиями: объект, класс, атрибут, метод.

Объектная платформа системы НСИ может использоваться в качестве интеграционной среды прочими информационными ресурсами предприятия.

2.3 Онтологическое представление НСИ

Эксперты строят онтологии или семантические модели своих предметных областей: описывают, “что из чего состоит и каким бывает” в данной области и как одно может быть связано с другим.

Классификация объектов в интеллектуальной MDM-системе служит основанием для упорядочивания правил, определяющих их поведение. Так, правило расчета массы детали на основе ее геометрических размеров и плотности материала принадлежит объекту *Деталь*, а знание о том, что спиральным сверлом можно получить круглое отверстие, лежит на пересечении объектов *Сверло* и *Отверстие*.

Все частные онтологии сшиваются в единую универсальную онтологию, составляющую единое пространство понятий. Этим обеспечивается фиксация всех возможных точек зрения на структуру, состав и возможности взаимодействия объектов. Появляется возможность многократно использовать справочные данные из единого источника в различных прикладных областях.

2.4 Контекстность видения НСИ

Контекстность – зависимость содержания и объема доступных справочных данных от цели их использования. Это свойство отражает тот факт, что знания всегда связаны с определенной областью деятельности, видом задач, которые мы хотим решить. Связи, заданные в одном контексте, не обязательно имеют место в других контекстах. Контекст естественным образом соответствует какой-либо функции информационного объекта.

Контексты позволяют различным группам пользователей по-разному видеть объекты НСИ на разных этапах их жизненного цикла. Например, контекстная точка зрения на металлорежущий станок позволит технологу видеть в структуре этого объекта механизмы перемещения заготовки и режущего инструмента, а механику – узлы и детали, подлежащие профилактическому осмотру.

2.5 Ориентированность на знания

Ключевая особенность единой системы – возможность оперировать семантическими связями между объектами, т.е. знаниями. Правила принятия решений, традиционно находившиеся в алгоритмах приложений, переносятся на уровень моделей данных. Знания, во-первых, становятся доступными другим приложениям и, во-вторых, за счет переноса на более низкий уровень повышается эффективность их обработки.

При этом в качестве критериев отбора объектов можно задавать их атрибуты и взаимосвязи с другими объектами. Например, при поиске сверла в классификаторе режущих инструментов можно указать не только его длину и диаметр, но и материал обрабатываемой детали, схему обработки и металлорежущий станок. MDM-система подберет все сверла соответствующего размера, совместимые с заданными объектами.

Итак, алгоритм работы интеллектуальной MDM-системы, построенной по изложенным принципам, выглядит следующим образом. В систему собираются справочные данные от всех приложений. При этом в одном объекте смешиваются атрибуты, представляющие интерес для различных групп пользователей. Контекстность видения объекта помогает рационально использовать его в частных задачах. А сеть правил взаимодействия, охватывающая все объекты, обеспечивает автоматизацию принятия решений везде, где это возможно, без обращения к бизнес-логике прикладных систем.

3 Промышленный прототип интеллектуальной системы управления НСИ

Коллективом разработчиков АСКОН накоплен многолетний уникальный опыт по созданию и развитию системы управления НСИ для машиностроения.

АСКОН производит комплекс программных решений для автоматизации конструкторско-технологической подготовки производства. За управление НСИ в комплексе отвечает набор специальных компонентов, один из которых – *Универсальный технологический справочник*.

За пять лет своей истории справочник прошел эволюцию от простого классификатора технологических данных до полноценной объектно-реляционной системы управления произвольной НСИ. Выпущено четыре коммерческие версии продукта, более двух тысяч его копий работают на производственных предприятиях, среди которых «ПО «Севмаш», «РСК «МиГ», ОАО «Вагонмаш» и др.

Система реализует объектную модель данных, которая является логической надстройкой над реляционной СУБД (поддерживаются Microsoft SQL Server, Oracle, InterBase).

Ключевые особенности системы:

- развитые функции поиска в массиве справочной информации;
- ведение неограниченного количества многоуровневых справочников;
- поставка клиентским приложениям данных в виде объектов;

– механизм установления взаимосвязей между объектами справочников;
– импорт, экспорт данных в различные форматы, включая XML;
– сохранение истории изменений и использования данных;
– развитый API-функционал и многое другое.

С системой поставляются обширные базы данных по машиностроительному оборудованию, станкам, инструментам и материалам. Предусмотрены два варианта поставки справочника: в качестве самостоятельного приложения – интеллектуального хранилища данных с функциями информационно-поисковой системы, и как поставщика справочных данных внешним приложениям: САПР, PDM, ERP и др.

В отличие от традиционных MDM-решений, предназначенных в первую очередь для автоматизации бизнес-процессов продаж, поставок и маркетинга, наша система ориентирована на производство как таковое и способна учесть все его аспекты: от проектирования изделий до принятия управленческих решений.

4 Заключение

Интеллектуальная MDM-система – это в первую очередь продвинутые возможности по работе с практическим смыслом – семантикой хранимой информации. Семантика уже активно используется в интернет-индустрии, нефтегазовой промышленности, здравоохранении – областях, где знания составляют основное содержание или стоят особенно дорого.

При построении семантической модели предметной области в рамках локальной MDM-системы приходится оперировать терминами и определениями различных областей знаний. С развитием Semantic Web – “семантического интернета” – многочисленные модели предметных областей будут созданы в удаленных центрах компетенции и распространены в глобальной Сети. Предприятия смогут получать информацию “из первых рук”, обращаясь напрямую к самым актуальным базам инструмента, оборудования, материалов на сайтах их производителей.

Внедрение MDM-системы в сочетании с семантическими технологиями откроет предприятиям перспективу свободного участия в глобальном обмене знаниями, обещающем стать стандартом уже в ближайшее десятилетие.

Средства визуального анализа онтологии и информационного наполнения портала знаний[1]
Tools for Visual Analysis of Ontology and Content of a Knowledge Portal

З.В Апанович[1], П.С.Винокуров[2]
Zinaida Apanovich[1], Pavel Vinokurov[2]

[1] Институт систем информатики им. А.П. Ершова СО РАН
630090, Новосибирск, проспект Лаврентьева, 6, Россия
A.P. Ershov Institute of Informatics Systems SB RAS
тел: +7 (383) 330-93-44, факс: +7 (383) 332-34-94
6, Acad. Lavrentjev pr., Novosibirsk 630090, Russia
apanovich@iis.nsk.su
Phone: +7 (383) 330-93-44, Fax: +7 (383) 332-34-94
[2] Новосибирский государственный университет
630090, Новосибирск, ул. Пирогова, 2, Россия
Novosibirsk State University,
2 Pirogova Street, Novosibirsk, 630090, Russia

Abstract. The process of development of an ontology-based knowledge portal and creation of its content is time-consuming and labor-intensive. It is very desirable to have special analysis facilities for maintenance and development of such a portal. The subject of our paper is a tool for visual analysis of content and ontology of a knowledge portal.

The basis of our approach is applying graph generation and graph visualization methods. Browsing of the content of a knowledge portal is organized as a multi-level stepwise process. Appropriate placement algorithms are used at each step of this process. They take into account certain types of ontological relations and their combinations. Radial, circular and layered tree placement algorithms are used for visualization of the inheritance relation. Several force-directed algorithms are used for visualization of role relations between different classes as well as between class instances. Finally, we have two special placement algorithms for visualization of the superposition of partonomy and role relations. A number of useful graph transformations are realized. They essentially improve understandability of a portal under investigation.

This approach provides a user with general understanding of ontology and possibility of visual estimation of the sizes of classes and relations. It allows the user to select and analyze specific relations between classes and between objects as well as to identify errors. The program was tested on the real knowledge portals and appeared to be rather helpful.

Key words: knowledge portal, ontology, content, information visualization, force-directed placement, tree placement.

---

[1] Работа выполнена при финансовой поддержке Российского фонда фундаментальных исследований (грант № 09-07-00400).

Ключевые слова: портал знаний, онтология, информационное наполнение, анализ информации, методы визуализации информации, силовой алгоритм, радиальный алгоритм

Введение

Процесс разработки онтологии и информационного наполнения портала является весьма длительным и трудоемким и требует усилий большого коллектива разработчиков. Многие данные вводятся вручную, что потенциально опасно ошибками ввода, которые нелегко обнаружить, просматривая одна за другой текстовые формы, показывающие связи конкретного объекта. Поэтому задача поддержания долговременного функционирования и развития такого портала в течение всего жизненного цикла является весьма актуальной и требует разработки специализированных средств. Для решения этой задачи важно обеспечить, в частности, одного из главных разработчиков системы – инженера знаний – инструментарием для анализа информационного наполнения портала, который значительно упростит его понимание. Обеспечение «понимаемости» онтологий также существенно при повторном использовании уже созданной онтологии и базы знаний в разработке других порталов, так как гораздо проще исследовать и развивать существующую онтологию, чем разрабатывать новую с нуля. Наконец, такие средства анализа могут служить основой для автоматического пополнения портала новыми знаниями. Общепризнанным инструментом, обеспечивающим понимание больших объемов абстрактной информации, являются методы визуализации информации [1–3]. Онтология, составляющая основу информационного портала, может быть представлена в виде графа, вершины которого изображают сущности, такие как классы, объекты и атрибуты онтологии, а ребра изображают отношения между этими сущностями.

Следует отметить, что алгоритмы визуализации постоянно обновляются, что приводит к необходимости развития подсистемы визуализации конкретного информационного портала. Также, для поддержания долговременного функционирования портала знаний, нужны средства не только визуализации, но и анализа накопленных знаний.

В данной работе методы визуализации рассматриваются на примере наполнения археологического портала знаний[4, 5], хотя в процессе работы, аналогичные эксперименты были проведены и с другими данными.

1 Входные данные и панель управления подсистемы визуализации

Подсистема визуализации принимает входные данные в виде двух xml-файлов, первый из которых (ontology.xml) содержит информацию о классах и отношениях классов, а второй файл (data.xml) содержит информацию о конкретных объектах и их отношениях. Эти данные преобразуются во внутренний формат, после чего на экране появляется окно управления визуализацией. Центральную

часть этого окна занимает панель визуализации, а верхняя и левая часть окна занята панелью управления. Слева находятся компоненты выбора классов и отношений, имеющихся во входных файлах, а вверху расположены компоненты, позволяющие выбирать режим визуализации. В настоящий момент в подсистеме имеется 3 режима визуализации: ”Наследование классов”, “Отношения между классами”, “Отношения между объектами”. В режиме «Наследование классов» осуществляется генерация изображения дерева, вершинами которого являются все классы онтологии, а ребра изображают отношение наследования между классами онтологии. В режиме « Классы» к изображению отношения наследования добавляется изображение ассоциативных отношений между классами, выбранными пользователем. В режиме «Объекты» строится изображение выбранных пользователем отношений между объектами выбранных классов.

Каждому режиму визуализации соответствует свое множество алгоритмов визуализации, компонента выбора алгоритмов также расположена вверху. Наконец, в верхнем левом углу имеется компонента выбора атрибутов объектов произвольного класса. Названия выбранных атрибутов изображаются при визуализации отношений между отдельными объектами.

2 Изображение отношения наследования между всеми классами онтологии

Знакомство с онтологией предлагается начинать с общего плана (изображения) всех классов и отношений наследования между этими классами. Для получения такого изображения достаточно выбрать в списке «Элементы отрисовки» вариант «Наследование классов» и нажать кнопку «Нарисовать». В результате будет вызвана подпрограмма, которая извлечет из онтологии подграф, соответствующий отношению наследования между классами, и изобразит его одним из алгоритмов, имеющемся в списке «Алгоритм отрисовки». В настоящий момент для изображения отношения наследования имеются поуровневый, радиальный и круговой алгоритм изображения дерева. Используемая в качестве примера онтология археологии не содержит множественных наследований, поэтому подграф наследования является деревом весьма умеренного объема. Высота дерева равна трем, в нем имеется тринадцать вершин первого уровня, большинство вершин имеет глубину два и только три вершины имеет глубину три, что позволяет вполне комфортно отобразить на одном экране и структуру и метки при всех вершинах. Поэтому для визуализации отношений наследования между классами археологического портала вполне достаточно статической версии радиального алгоритма для деревьев [6]. Эксперименты с различными онтологиями показали, что наличие пустых классов, без объектов, является обычным явлением. Поэтому такие классы высвечиваются серым для привлечения к ним внимания пользователя

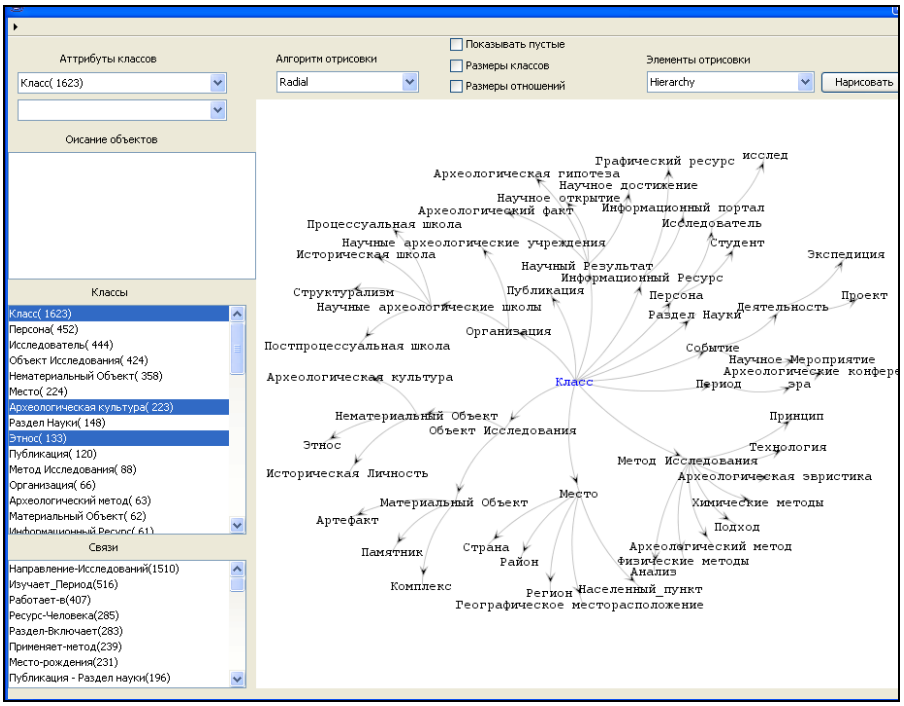


Рисунок 1. Окно визуализации и радиальное изображение отношения наследования между классами онтологии археологии

В качестве дополнительной возможности, в подсистеме визуализации разработана версия радиального алгоритма, в которой размер секторного сегмента, изображающего каждый класс, пропорционален количеству объектов в этом классе. Этот вариант позволяет визуально оценить количество объектов каждого класса.

Рисунок 2. Изображение, использующее геометрическую вложенность для визуальной оценки количества объектов

3 Изображение ассоциативных отношений между классами

Существенной особенностью информационного наполнения археологического портала является наличие большого количества различных видов отношений между классами объектов. Помимо отношения наследования в онтологии информационного портала используются отношения партономии, большой спектр транзитивных отношений (к которым, в частности, относится отношение включения), а также большое разнообразие ассоциативных отношений. Возможность просмотра ассоциативных отношений между классами в дополнение к уже изображенному отношению наследования является весьма важной для понимания структуры портала. В настоящий момент эта возможность предоставляется пользователю в двух режимах отрисовки.

В режиме «Наследование классов» достаточно выбрать при помощи мыши произвольный класс на изображении всей онтологии, и в результате этого действия к изображению будут добавлены ребра с метками, изображающие ассоциативные связи выбранного класса, как это показано на рисунке 3. Одновременно, на панели «Связи», расположенной слева от окна отрисовки, будут высвечены названия этих отношений с указанием количества экземпляров данного отношения.

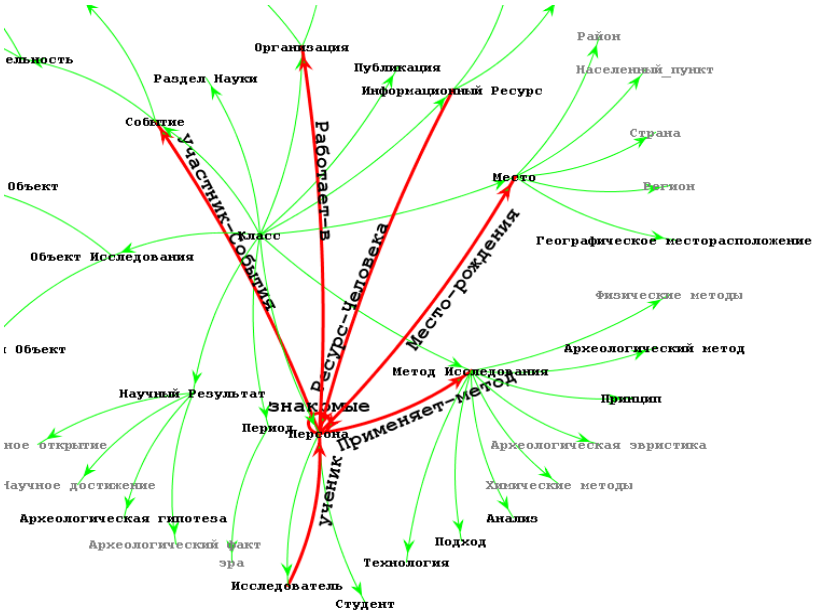


Рисунок 3. Иерархия классов с высвеченными ассоциативными отношениями

Этот тип изображения является весьма полезным инструментом при исследовании свойств существующей онтологии. В частности, при разработке онтологии не всегда легко отделить классы и подклассы, связанные отношениями наследования, от объектов одного класса, связанных ассоциативным отношением

включения. При анализе мы исходим из предположения, что подкласс имеет право на самостоятельное существование, если он имеет отношения и атрибуты, отсутствующие у класса-отца.

Например, онтология археологического портала содержит класс Персона, у которого есть подклассы Студент и Исследователь. Объектами всех этих классов являются физические лица. При этом класс Персона связан с другими классами следующими ассоциативными отношениями :

«Знакомые»(класс Персона)», «Участник-события»(класс Событие), «Ресурс-Человека»(класс Информационный Ресурс), «Применяет-Метод»(класс Метод_Исследования),

«Ученик»(класс Исследователь).

Его подкласс Исследователь, помимо наследуемых отношений от класса Персона, имеет дополнительные отношения с другими классами: «Направление-исследований»(класс РазделНауки), «Автор» (класс Публикация), «Изучает-Период»(класс Период), «Участник-Проекта»(класс Проект), «Ученик» (класс Персона).

В то же время, подкласс Студент класса Персона своих собственных отношений с другими классами не имеет. Это свойство легко обнаруживается при выделении мышью данного подкласса в режиме «Наследование классов».

Исходя из обнаруженных свойств, можно рекомендовать инженеру знаний либо дополнить подкласс Студент отношениями, отличающими его от классов Персона и класса Исследователь, либо удалить его из множества классов. Аналогичным образом, можно быстро исследовать все подклассы имеющейся онтологии. Ввиду небольшого количества классов в исследуемой онтологии, такая проверка требует всего несколько минут.

При исследовании онтологии достаточно важной является также возможность увидеть полную картину отношений между классами, интересующими пользователя. С этой целью в системе визуализации предусмотрена возможность выбора произвольного подмножества классов в списке «Классы» панели управления и их визуализации, как это показано на рисунке 4. В этом случае можно получить изображение выбранных классов и связей этих классов с другими классами. Такие изображения могут быть удачной подсказкой для разработчика онтологии. Например, на рисунке 4 изображены связи между классами Персона и Исследователь. Зеленое ребро соответствует отношению наследования между этими классами. При этом у пользователя может возникнуть вопрос, почему отношение «Применяет-Метод» связывает класс Метод Исследования с классом Персона, а не с классом Исследователь.

В качестве дополнительной возможности, можно получить изображение, в котором размер вершин и толщина ребер пропорциональны количеству объектов в соответствующих классах и количеству связей между объектами этих классов. Такое изображение показано на рисунке 5. Оно позволяет заранее оценить объем изображения при переходе на уровень объектов.

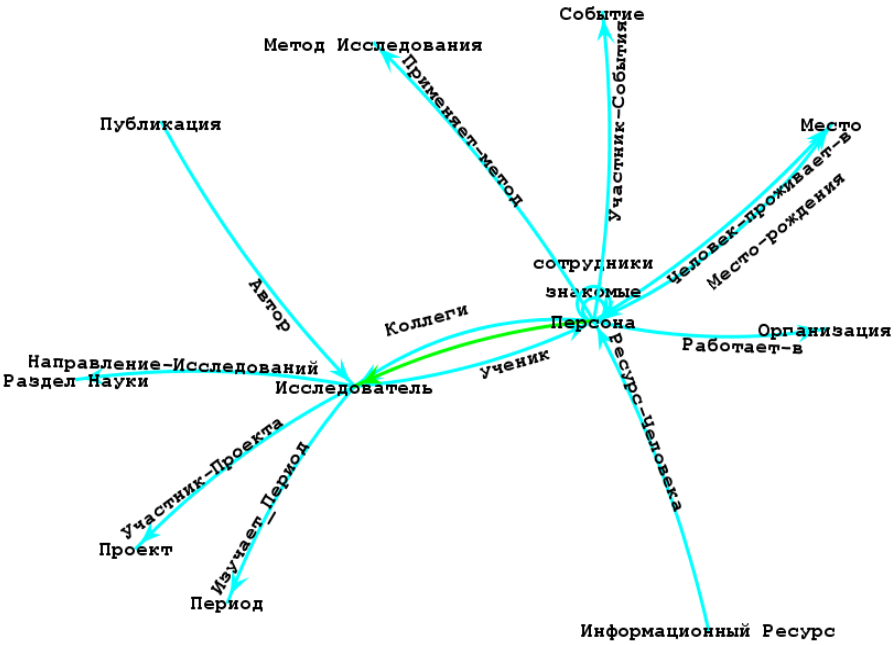


Рисунок 4. Изображение связей между классами Исследователь и Персона

Рисунок 5. Изображение в котором размер вершин и толщина ребер пропорциональны количеству объектов в соответствующих классах и количеству связей между объектами.

4 Изображение объектов и связей между ними

Режим отрисовки «Объекты» предназначен для изображения объектов, принадлежащих классам, выбранных пользователем, и типов связей, также выбранных пользователем.

Для построения изображения объектов и связей между ними можно выбрать при помощи мыши интересующие отношения на диаграмме классов и перейти в режим визуализации отношений между объектами. Можно также выбрать нужные отношения из списка имеющихся отношений, перечисленных в компоненте выбора отношений. Программа разделяет граф на компоненты связности, а затем применяет к ним один из имеющихся алгоритмов размещения. Наиболее удобными при визуализации имеющихся данных оказались алгоритмы Kamada-Kawai[9] и Fruchterman-Reingolda[7]. Они позволяют быстро оценить объем и структуру визуализируемых данных. На рисунке 6 изображена визуализация отношения “Объект датирован” между объектами класса “Период” и “Объект исследования” . Можно видеть несколько компонент связности. Центром каждой компоненты связности является объект класса период, с которым ребрами соединены объекты исследования, датируемые соответствующим периодом.

Рисунок 6. Изображение отношения “Объект датирован” между объектами класса “Период” и “Объект исследования”

Помимо обычных силовых алгоритмов, имеется несколько несколько специфических методов визуализации, учитывающих специфику конкретных отношений между объектами. Например, тестовая онтология содержит значительное количество отношений партономии, таких как «Метод-Включает», «Раздел-Включает» и т.д. При выделении подграфов, соответствующих этим отношени-

ям обнаруживается значительное количество транзитивных ребер. Наличие этих ребер влечет применение силового алгоритма визуализации, в результате которого получается малопонятное изображение. В таких случаях программа позволяет извлекать дерево вложенностей при помощи удаления транзитивных отношений и строить изображение одним из алгоритмов визуализации деревьев. На рисунке 7 изображена визуализация отношения “Метод-Включает” для объектов из класса “Метод исследования”до и после удаления транзитивных ребер.

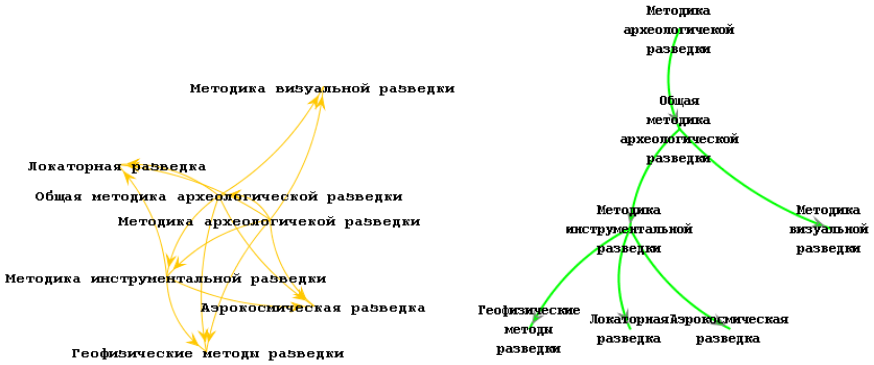


Рисунок 7. Визуализация отношения партономии между объектами класса Метод Исследования

Еще одна особенность связана с тем, что в тестовой структуре имеется большое количество ассоциативных отношений связывающих объекты, которые являются частью иерархической структуры, индуцированной отношением вложенности (партономии). Например, имеется большая иерархия методов исследования, связанных отношением “Метод-Включает”, а пользователь желает увидеть персон, применяющих эти методы. Если визуализировать соответствующий граф без учета отношения вложенности, изображение может оказаться весьма запутанным, как это показано на рисунке 8.

Поэтому возникает задача построения специализированной визуализации ассоциативного отношения и отношения партономии на уровне объектов. Мы разработали и реализовали два алгоритма визуализации такой комбинации.

Первый алгоритм размещает дерево вложенностей при помощи геометрической вложенности окружностей. В каждой окружности размещаются объекты, связанные ассоциативным отношением с соответствующей вершиной дерева вложенностей.

Вершины, изображающие эти объекты, размещаются алгоритмом Kamada&Kawai[9].

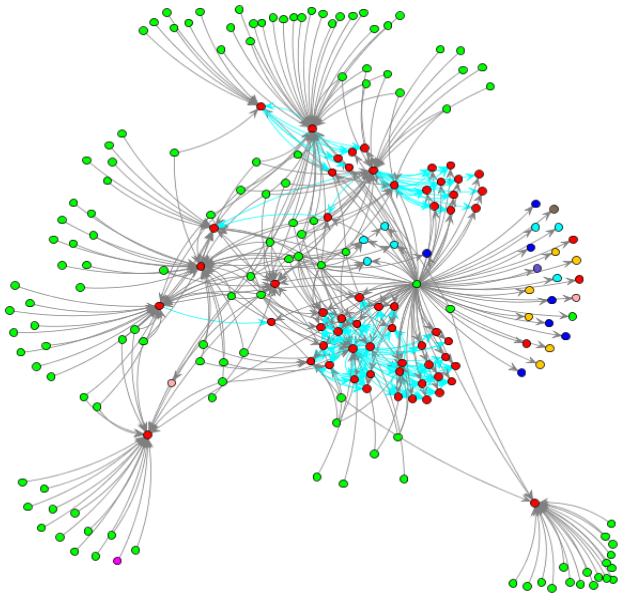


Рисунок 8. Совместное изображение отношения партономии и одного из ассоциативных отношений при помощи силового алгоритма

При выделении некоторого объекта к данному изображению добавляются связи, показывающие все вершины дерева вложенностей, связанные с выбранным объектом. На рисунке 9 изображено дерево вложенностей отношения “Метод исследования-включает”. Центральная окружность соответствует объекту ”Методика раскопок”. Методы, которые включает ”Методика раскопок” соответствуют вложенным окружностям. Каждая окружность содержит вершины, изображающие персон, связанные с этой структурой отношением “Применяет метод”.

Достоинством такой визуализации является компактность изображения. Красные линии показывают все методы исследования, применяемые выбранной персоной.

Второй алгоритм визуализации сначала изображает дерево вложенности, соответствующее одному из классов объектов. А затем на это изображение накладывается ассоциативное отношение. При это высота каждой вершины равна количеству ассоциативных отношений соответствующего объекта и его под объектов. На рисунке 10 изображена визуализация отношения “Применяет метод” вместе с отношением «Метод-Включает». При выборе одного из объектов «Метод исследования» на экран выводится список объектов “Персона”, применяющих этот метод, а также изображаются стрелки, указывающие на другие методы, применяемые указанными Персонами.

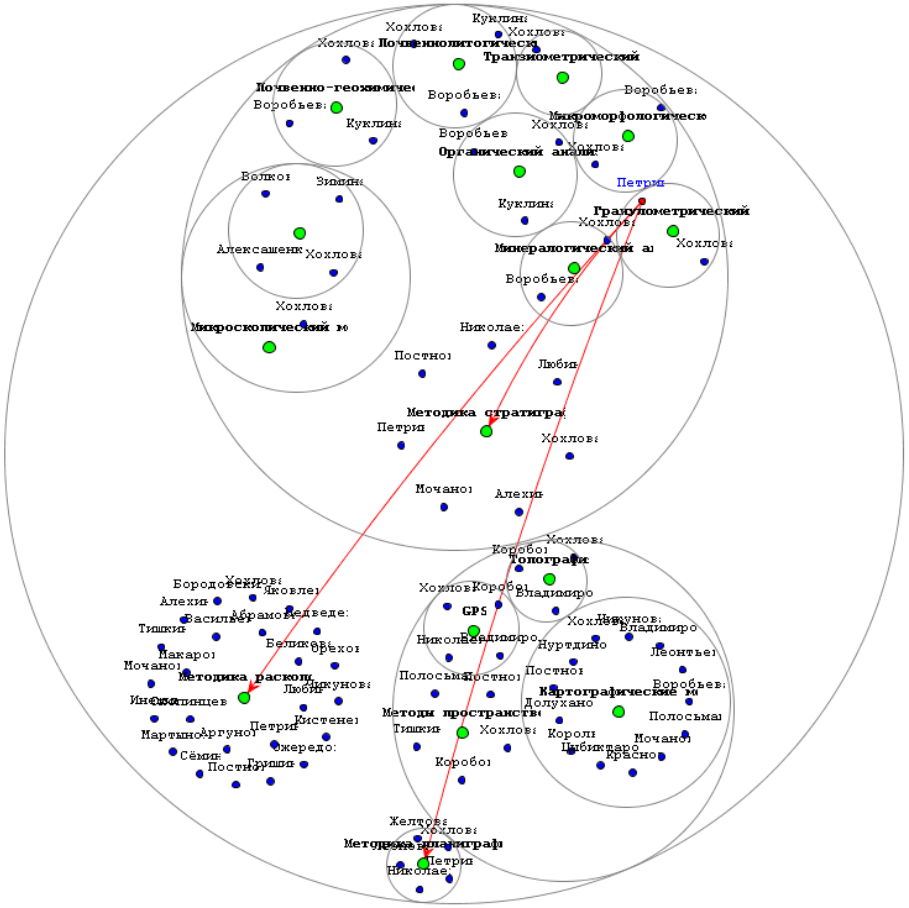


Рисунок 9. Совместное изображение отношения партономии и одного из ассоциативных отношений при помощи вложеных окружностей и силового алгоритма

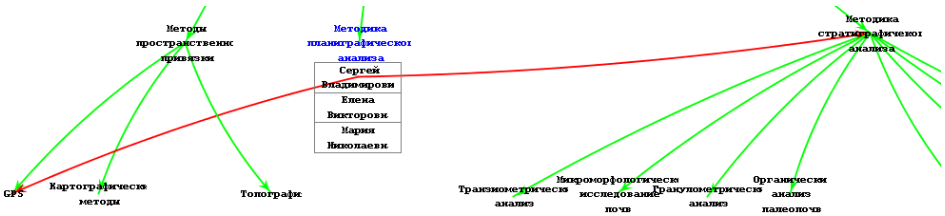


Рисунок 10. Совместное изображение ассоциативных отношений и отношения вложенности на уровне изображения объектов

Визуализация объектов и связей между ними оказалась весьма удобным инструментом, позволяющим быстро идентифицировать ошибки, возникающие при ручном вводе информационного наполнения портала. Часто признаком ошибочного ввода данных является наличие циклов в графе связей объектов, так и наличие петель, то есть ребер, соединяющих вершину с самой собой. Такие ошибки обнаруживаются при визуализации отношений между объектами одного и того же класса (петля на графе ассоциативных отношений между классами). Например, при визуализации отношения «Исторически следует» для объектов класса «Период» был обнаружен цикл, связывающий объекты «Нижний палеолит», «Средний палеолит» и «Верхний палеолит» (Рисунок 11).

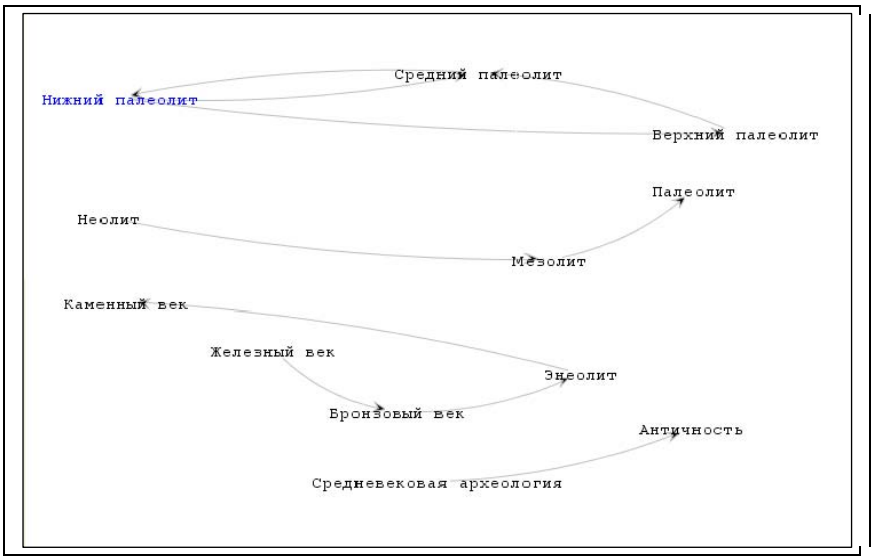


Рисунок 11. Наличие циклов для объектов типа Период, связанных отношением “Исторически следует»

Временные границы, определенные для этих периодов имеют следующий вид:

Название периода	Нижний палеолит
Начало периода	1500000 до н.э.
Конец периода	100000 до н.э.

Название периода	Средний палеолит
Начало периода	100000 до н.э.
Конец периода	40000 до н.э.

Название периода	Верхний палеолит
Начало периода	40000 до н.э.
Конец периода	14000 до н.э.

Согласно временным границам, определенным для этих периодов, они должны бы быть упорядочены линейно: «Нижний палеолит» –> «Средний палеолит» –> «Верхний палеолит», и никаких циклов в получаемом графе быть не должно. Проверка показала, что данные действительно содержали ошибку. Более того, граф отношения «Исторически следует» оказался несвязным, что тоже может указывать на потенциальные недоработки.

В настоящий момент единственным способом обнаружения подобных ошибок является просмотр всех имеющихся классов и отношений, что может оказаться весьма длительной и трудоемкой проблемой. Поэтому желательно проанализировать заранее семантику имеющихся в онтологии отношений и выделить те средства анализа, которые позволяли бы автоматически определять корректность ввода данных. Имеющиеся на данный момент методы визуализации позволяют, по крайней мере, сформулировать критерии для такой автоматической проверки.

Заключение

Эксперименты с визуализацией информационного наполнения портала знаний показали необходимость дальнейшего развития подсистемы визуализации в нескольких направлениях. Во-первых, для эффективной визуализации классов, содержащих большое количество объектов и отношений, необходимо дополнить существующие алгоритмы методами кластеризации и размещения кластеров. Во-вторых, необходимо исследовать типы отношений, имеющихся в онтологии и определить наиболее эффективные методы визуализации как для определенных типов отношений, так и для различных типов подграфов, выделяемых при анализе информационного наполнения портала знаний. И, в третьих, для эффективной работы с информационным наполнением портала необходима разработка специализированных методов анализа и верификации этого наполнения. При разработке первой версии подсистемы визуализации использовалась свободно распространяемая библиотека классов Java, называющаяся JUNG[8].

Благодарности. Авторы выражают благодарность Ю.А. Загорулько и С.В. Булгакову за предоставленные тестовые данные в xml-формате.

Список литературы

1. Апанович З.В. Методы навигации при визуализации графов // Вестник НГУ. – 2008. – Т. 6, вып. 3. – С. 35–47 .
2. Апанович З.В. Методы интерактивной визуализации информации // Проблемы управления и моделирования в сложных системах: Труды X Международной конференции (Самара, 23–25 июля 2008 г.). – 2008. – С. 478–489.
3. Апанович З. В. Средства для работы с графами большого объема: построение и оптимизация компоновочных планов // Системная информа-

тика. Вып. 10: Методы и модели современного программирования. – Новосибирск: Изд-во СО РАН, 2006. – С. 7–58.
4. Загорулько Ю.А., Боровикова О.И., Холюшкин Ю.П. Построение предметной онтологии для археологического портала научных знаний // Информационные технологии в гуманитарных исследованиях. Вып 10. – Новосибирск, 2006.
5. Холюшкин Ю.П., Гражданников Е.Д. Системная классификация археологической науки (элементарное введение в науковедение) Новосибирск. – Новосибирск. –2000: 58С.
6. Di Battista G., Eades P., Tamassia R., Tollis I.G. Algorithms for Drawing Graphs: an Annotated Bibliography // Computational Geometry, Theory and Applications. 1994. N 4. P. 235-282.
7. Fruchterman T. M. J., Reingold E. M.: Graph Drawing by Force-Directed Placement Software // Practice and Experience. – 1991. – Vol. 21, N 11. – P. 1129–1164.
8. JUNG 2.0 Tutorial http://jung.sourceforge.net.
9. Kamada T., Kawai S. An algorithm for drawing general undirected graphs // Information Processing Letters. – 1989. – Vol. 31. – P. 7–15.
10. Katifori A., Halatsis C., Lepouras G., Vassilakis C., Giannopoulou E. Ontology visualization methods–a survey // ACM Comput. Surv. – 2007. – Vol. 39, N. 4. – P. 10.

EZOP – Web-сервис коллективного построения онтологий с открытым языком представлений и запросов
EZOP – Web 2.0 service of ontologies' construction

Е.М. Бениаминов, В.А. Лапшин, Д.В. Перов
Evgeny Beniaminov, Vladimir Lapshin, Dmitry Perov

Российский Государственный Гуманитарный Университет
Москва, Миусская площадь д.6 Корпуса 1-7
Russian State University for the Humanities,
Miusskaya sq, 6, Moscow, GSP-3, 125993
ebeniamin@yandex.ru, mefrill@yandex.ru, dperov2005@gmail.com

Аннотация Описан проект EZOP, представляющий собой сервис Web, предназначенный для коллективного построения библиотек онтологий с открытым языком представлений и запросов. Сервисы такого вида могут быть также использованы в качестве инструмента формирования социальных сетей в Semantic Web, позволяющих пользователям формировать не только тексты страниц данной социальной сети, но и программные модели их содержимого. В данном проекте предполагается создание среды в Web, поддерживающей вместе с процессом коллективного формирования библиотек онтологий в социальной сети коллективный процесс формирования языка, на котором эти онтологии формируются и используются. Работа на проектом EZOP ведется по гранту РФФИ **09-07-00079-а**.

The report describes our work on EZOP project. The work is performed on the RFBR's grant 09-07-00079-a. EZOP is Web 2.0 service to construct ontologies in some areas of knowledge. EZOP also provides the service to change the language of the knowledge representation by users. So, the language of ontologies construction in EZOP system is open one and allows extension of the syntax and lexis by new forms.
EZOP service provides the tools to collectively build the ontologies by members of some community in Web 2.0 style. The result of such the process is so named active page. The active Web page is the one, which may talk something to a user, not only provides the text to read. The dialog of the page to a user is performed via special query language. A user can specify query language according to the content of the page's ontology. Thus, the dialog may perform in the language, which expresses the knowledge of the page in the best way.
Users may collectively build the ontology of the active page. And, both language of page's ontology construction and language of queries to the page can be specified according to the content of the ontology. EZOP

service provides the base language to build ontologies, which can be extended by new syntax and lexical constructions.

A user usually selects one or more of existing active pages to form his active page. This likes to Wikipedia templates in some way. But, the new page also inherits the language and semantics of the old one. A user may have access to vocabularies and texts of other active pages in the ontology. The active page contains the text (which is written in the formal language, which can be extended by users) and contains the internal representation of the page's semantic as well. The internal representation is formed in the special EZOP system's language, which does not designed to read humans, but for machines only. The internal representation can be determinately built from the page text.

The semantic of an active page in EZOP system is built as the finite approximation of so named ideal semantic of the page. Ideal semantic is the set of correctly constructed expressions (terms and formulas) which is built from the terms of the page's ontology by using the equivalence relation defined by the ontology's equalities. The approximation contains only equivalence classes that are needed to represent the page semantic.

The language of page's ontology representation and queries to the page is constructed by using language templates. The language template allows combining the syntax and the semantic in the single unit. The syntax of the template is defined by the string of symbols, some of them are variable, and other ones represent plain text. The variable name is started from symbol "@". An each variable has type and each template has a result type. So, language template's syntax is just the production of context-free Chomsky's grammar, where nonterminal symbols are variable types and the template result type is the nonterminal symbol in the left part of the production.

An each template has equalities, calculation condition and action. The equalities specify relations between variables. The calculation condition is some kind of restriction on the template's action. And the action is the specification of the calculation of the result. The action semantic can be specified in EZOP system internal terms as rewriting rules and as the external action module.

Thus, the ontologies of active pages are constructed as follows:

1. Ontologies are built in Wiki style collectively.
2. EZOP system supports the environment to extend the ontology language by new constructions.
3. The active page contains in addition to its text the internal representation of the ontology written in the language understated by the program system.

1 Введение и мотивация

В течении последних нескольких лет пространство Web документов приобрело принципиально новое содержание. Для подчеркивания этого факта даже используется новый термин: Web 2.0. Основным отличием Web 2.0 от

предыдущего поколения Web является наличие групп пользователей, объединенных общими интересами в так называемые *социальные сети*. Компании предоставляют сервис для того, чтобы пользователи имели возможность в режиме онлайн общаться друг с другом в рамках своего сообщества. В процессе этого общения формируется новое содержание Web документов. В этом состоит принципиальное отличие Web 2.0 от Web прошлого десятилетия: в Web 2.0 документы формируются пользователями Сети, а не хозяевами сервера.

Естественным развитием этого подхода видится предоставление сообществу пользователей социальной сети возможности формирования *активных документов*. Активная страница – это Web документ, который предназначен не только для чтения его содержимого, но и способен поддерживать диалог с пользователем: отвечать на вопросы, выполнять команды и т.п.

В информатике уже есть примеры такого использования специфических текстов. Примерами являются тексты, формируемые в системах математических вычислений Mathematica, MathLab и т.д. и в некоторых системах программирования. В этих системах можно спросить, какие значения имеют переменные или сложные выражения, составленные из объектов страницы и доступные из текущего текста функций.

Дополнительной проблемой для реализации активных документов в среде Web 2.0 является то, что такие документы должны будут формироваться самими пользователями. Для этого необходимо предложить пользователям соответствующий сервис. Такой сервис должен будет программно поддерживать модель содержания текста. Подобные модели обычно называют *онтологиями*. Итак, пользователи социальной сети будут формировать онтологии содержимого Web документов точно также, как они сейчас формируют тексты.

Формирование онтологий представляет собой сложный процесс, специфическую деятельность, требующую от участников больших творческих усилий. Поэтому, надеяться на то, что пользователи будут специально формировать онтологии текста документа на каком-то специфическом формальном языке, довольно наивно. Единственной возможностью сделать такую деятельность естественной и нетрудоемкой для пользователей, представляется использование языка, синтаксически адаптированного для выражения содержания Web страницы.

Такой язык должен легко расширяться с тем, чтобы адаптироваться под потребности того или иного содержания. Содержание страниц социальных сетей обычно сходно, ведь социальные сети формируются на базе общих интересов, а значит, и общих представлений. Представляется, что сервис данной социальной сети формирует ее базовый язык, на основе которого пользователи могут формировать содержимое ее Web документов. Но пользователи также должны иметь возможность дополнить базовый язык своим синтаксисом, адаптированным для выражения содержания страниц той или иной социальной группы в рамках данной социальной сети.

Таким образом, пользователи социальных сетей в Web будут иметь возможность коллективно формировать онтологии и снабжать их специфическим языком, позволяющим вести с содержимым страницы (онтологией данной страницы) прямой диалог. Задачу реализации сервиса, предоставляющего такие возможности, можно переформулировать следующим образом: разработать Web-сервис построения онтологий с открытым языком представлений и запросов.

Проблемы, связанные с реализацией такого сервиса, а также их решения, достаточно подробно рассмотрены ниже.

Формирование содержания Web документов представляет собой главный предмет изучения в Web-технологии, получившего название «Semantic Web» [5]. В Semantic Web релевантный поиск по представлению семантического содержимого (онтологии) текста страницы рассматривается как основная функция, побуждающая вводить семантическую информацию на страницах. Нам представляется, что улучшение качества поиска страниц не побудит массового пользователя делать большие усилия для формализации «мира страницы» и делать это на программистских языках онтологий. Некоторое развитие предложений «Semantic Web» было сделано в направлении Semantic Wiki [6].

Далее, мы рассмотрим проект EZOP ([4]) – реализацию Web сервиса коллективного формирования онтологий с открытым языком представления и онтологий.

2 О проекте EZOP

EZOP представляет собой Web-сервис, который предоставляет среду для коллективного формирования пользователями библиотек формализованных знаний (онтологий) для различных предметных областей и создания (фиксации) формализованных языков этих областей. По организации этот сервер близок к Web-сервису «Wikipedia». Главное отличие от сервиса «Wikipedia» состоит в том, что страницы сервера EZOP содержат части, которые пишутся пользователями на формальном открытом языке. На основании этих частей система строит семантическое представление страницы и язык запросов к ней.

При формировании новой страницы пользователь обычно использует какую-либо из существующих страниц в системе в качестве среды новой страницы (при этом новой страницей наследуется язык и семантика страницы-среды) и может использовать другие страницы системы для формирования семантики текущей (формируемой) страницы без отображения текстов используемых страниц на формируемой странице. В процессе формирования страницы пользователю обеспечивается доступ к словарям и текстам, хранящимся в системе, для использования в формируемой странице.

Сформированную страницу можно сохранить в системе с сохранением прежнего имени (но в новой версии) или под новым именем. Любую страницу, хранящуюся в системе, можно сделать текущей для редактирования

или для обращения к странице с вопросами. Язык запросов и формирования страницы, определяется языком страницы-среды, содержанием текущей страницы и видимыми из текущей страницы языковыми средствами, введенными на других страницах.

EZOP предоставляет пользователям возможность задавать вопросы к редактируемой странице, т.е. обеспечивать диалог данной страницы с пользователем. Ответы на вопросы также формируются на языке, который коллективно создан участниками данного сообщества. Кроме того, сервис поддерживает расширяемость пользователями языка представления онтологий, т.е. формального языка текстов, на которых формируются онтологии.

С каждым формальным текстом онтологии связано его внутреннее представление в базе данных системы. Каким образом представляется онтология внутри системы, не специфицируется пользователем, но определяется вводимым текстом. Сервис должен предоставлять возможность перевода внутреннего представления онтологии страницы на популярные языки представления онтологий, такие как OWL [3] или KIF [2] для межмашинного обмена онтологиями. Также, обеспечивается возможность использования внутреннего представления онтологий страниц, которые использует пользователь при редактировании данной страницы, без обращение к тексту этих страниц.

3 Особенности реализации

EZOP использует алгебраический и категорный подходы для представления онтологий (см. [1]). Обычно, онтология представляется как **теория**, в которой задаются (определяются пользователем) **операции** и **отношения**, включая отношение равенства, которые связываются **аксиомами** (формулами). Типы сложности онтологий определяются типом сложностей аксиом. (Например, аксиомы только в виде равенств с кванторами всеобщности. Более сложные, в виде хорновских формул.) Стандартным образом определяются формулы, которые являются следствиями аксиом.

Сигнатурой (словарем) онтологии называется множество всех терминов, используемых в онтологии с указанием типов этих терминов.

Множество всех правильно построенных выражений (**термов** и формул), построенных из терминов онтологии, профакторизованное по отношению эквивалентности (равенства) на них, следующего из аксиом онтологии, называется **идеальной семантикой** онтологии.

Для аксиом общего вида, задача определения, является ли формула следствием из аксиом, как известно, алгоритмически неразрешима, и идеальная семантика онтологии, в этом случае, является недоступным для приложений объектом. Поэтому, во многих приложениях стараются ограничиться типом аксиом, для которых эта задача разрешима. Но так получаются только достаточно простые (по теореме Гёделя) онтологии, которые не могут покрыть многие, интересные для приложений области.

С другой стороны, так как для систем представления знаний неполнота представленных знаний является естественным свойством, то мы не будем требовать от компьютерной системы, чтобы она умела находить все следствия. Но, если некоторое следствие уже найдено, и оно полезно, то в системе должны быть средства сделать это следствие доступным. В связи с этим вводится понятие конечной аппроксимации семантики онтологии, которое строится аналогично аппроксимациям в виде машинным чисел для бесконечных алгебр целых и вещественных чисел.

Конечной аппроксимацией идеальной семантики онтологии называется пара, состоящая из некоторого конечного множества термов данной онтологии и некоторого подмножества отношения эквивалентности идеальной семантики на них, которые вычисляются эффективными алгоритмами. Одна аппроксимация считается более точной, чем другая, если первая содержит вторую.

В EZOP хранится не только текст онтологии, но и некоторая аппроксимация ее семантики, на основании которой проводятся необходимые быстрые вычисления (например, в алгоритме грамматического анализа). Пользователь, может расширять аппроксимацию онтологии, вводя в онтологию формулы-следствия аксиом, либо с помощью вопросов и систем логического вывода.

Онтологии делятся на более общие и более конкретные. Онтология, которая получена из некоторой другой онтологии добавлением к ней аксиом, по определению считается более конкретной, чем исходная.

Онтологии связываются отношением использования и интерпретации (реализации). Говорят, что первая онтология используется во второй, если все термины и аксиомы первой онтологии входят во вторую. Интерпретацией первой онтологии во второй называется некоторая третья онтология, содержащая первую и вторую, в которой каждый термин первой онтологии выражается некоторой формулой во второй с сохранением типов терминов и так, что аксиомы первой онтологии с подставленными в них вместо терминов их выражений из второй являются следствиями аксиом второй онтологии.

Некоторые общие онтологии создаются в системе как заготовки (модули) для использования в других онтологиях. Среди онтологий могут быть онтологии, описывающие типы данных, схемы баз данных, схемы запросов к базам данных и онтологии, описывающие конкретные задачи.

В системе должна поддерживаться функция построения ответов на некоторые вопросы к конкретной онтологии. Вопросы используются при опробовании и отладке формируемых онтологий и при построении онтологий для задач моделирования. В системе поддерживаются общие типы вопросов, относящиеся к ядру системы, и вопросы, язык которых определяется в конкретных онтологиях.

К общим вопросам, поддерживаемым ядром системы, относятся вопросы:

– Какие объекты данного класса введены или используются в онтологии?

– Какие подклассы имеются у данного класса в онтологии?
– Является ли выражение объектом (подклассом) данного класса, которое тоже может задаваться выражением?
– Равны ли два выражения в текущей аппроксимации онтологии?
– Нельзя ли вывести равенство двух выражений с использованием существующих средств расширения аппроксимаций онтологий?
– Чему равно выражение (в аппроксимации или с расширением аппроксимации)?

Языковые выражения (на формальном открытом языке системы) используются для задания вопросов к конкретной онтологии или используется для расширения онтологии или ее аппроксимации.

С каждой онтологией связывается ее идентификатор (который может быть скрыт от пользователя), название, онтология-среда, текст онтологии, конечная аппроксимация ее семантики, номер версии, время создания, автор. Кроме того, с каждой онтологией связываются языковые конструкции (шаблоны языка), введенные в данной онтологии (включая термины), которые могут быть видны (или не видны) из других онтологий, а также языковые конструкции, введенные в других онтологиях, но видимые из текущей онтологии.

Используя видимые из данной (текущей) онтологии языковые конструкции, пользователь может строить из них сложные выражения вопросов к текущей онтологии или выражения, меняющие онтологию или ее язык. Система должна определять синтаксическую и семантическую правильность построенного пользователем выражения, исходя из текущей конечной аппроксимации онтологии. С каждой конструкцией языка связывается действие, которое определяется либо ядром системы (если это конструкция из ядра), либо пользователем в онтологии при определении этой конструкции.

4 Примеры построения онтологий

Рассмотрим два примера построения онтологий в системе EZOP.

4.1 Булева алгебра

Первый пример представляет собой онтологию, описывающую алгебру булевых операций. В ядре системы EZOP определен класс «область», который используется для объявления имен классов. Сначала, вводится тип bool следующим образом:

```
bool - область.
```

Высказывания в языке системы EZOP формируются в виде заканчивающихся точкой предложений. В данном случае, предложение говорит о том, что имя bool обозначает область. Далее, задаем элементы области bool:

```
f,t - элементы области bool.
```

Область bool содержит два элемента: t (истина) и f (ложь). Объявим три логические операции и задаем их действия:

```
AND: bool x bool -> bool.
OR : bool x bool -> bool.
NOT : bool -> bool.

NOT(f)=t. NOT(t)=f.
AND(f;f)=f. AND(f;t)=f. AND(t;f)=f. AND(t;t)=t.
OR(f;f)=f. OR(f;t)=t. OR(t;f)=t. OR(t;t)=t.
```

Хотя операции введены, пользоваться ими неудобно, хотелось бы задавать операции операции в более привычной инфиксной нотации. Для этого необходимо ввести шаблоны, задающие инфиксные операции на булевыми типами. В системе EZOP шаблоны вводятся посредством специального шаблона «Введем шаблон», в который передается строка, задающая синтаксис шаблона, а также описываются условия применения шаблона и его действие. Имена переменных в тексте шаблона начинаются со знака «@».

```
Введем шаблон "@u & @v"
с переменными:
"u : bool; v : bool"
и переменной результата " w:bool " ;
Пояснения: [операция конъюнкции]
Условие применения шаблона:
[]
Действие шаблона:
[w=AND(u;v)]
Тип доступа шаблона: [внешний].
```

В данном шаблоне определяется инфиксная операция конъюнкции. В шаблоне объявляются две переменные типа bool и переменная результата, также типа bool. Условие применения шаблона в данном случае пусто, т.е. не ставится никаких дополнительных условий. Действие шаблона задается определенной ранее операцией AND. Данны шаблон определен как «внешний», что позволяет использовать его в других онтологиях. Аналогичным образом вводятся инфиксные шаблоны и для других операций.

Операциями, введенными в булевой алгебре, можно пользоваться в других онтологиях. Можно, например, задавать вопросы к онтологии с помощью встроенного шаблона «Чему равно @Выражение ?». Например:

```
чему равно (f&t) V t?
```

Система выдаст ответ:

```
Ответ: t
```

4.2 Равномерное движение

Второй пример представляет собой онтологию, описывающую понятия связанные с равномерным движением. Равномерное движение описывается аналитической формулой, в которой скорость на участке пути определяется, как отношение пройденного пути ко времени, затраченному на преодоление этого участка. Поэтому, сначала определяются классы «путь», «время» и «скорость», как подтипы класса real (встроенного в ядро ситемы класса действительных чисел).

```
Путь, скорость, время: real.
```

После этого, вводятся связывающие данные классы соотношения:

```
Путь = скорость*время.
Время = путь/скорость.
Скорость = путь/время.
```

Следует отметить, что в ядро системы EZOP встроена онтология основных арифметических операций, а также определены инфиксные шаблоны для этих операций. Здесь символ «*» обозначает операцию умножения, а символ «/» – операцию деления.

На данный момент, основные понятия онтологии равномерного движения введены. Также введены и соотношения между этими понятиями. Но пользоваться такой онтологией неудобно, необходимо ввести шаблоны, позволяющие удобно для человека описывать дополнительные понятия, связанные с данной онтологией. Таких шаблонов можно придумать много, рассмотрим, в качестве примера, один из них.

```
Введем шаблон "@Тело равномерно движется со скоростью @V"
с переменными: "Тело: new; V:  real_выражение"
и переменной результата "  x: команда " ;
Пояснения:
[Вводится объект @Тело, равномерно движущийся со скоростью @V ]
Условие применения шаблона:
[]
Действие шаблона:
[x = пустая команда;
тело - объект понятия "равномерное движение";
тело's скорость = V.  ]
Тип доступа шаблона:[внешний].
```

В данном шаблоне вводится возможность удобно для человека задавать скорость движения некоторого тела. Заметим, что переменная «@V», задающая скорость движения, имеет тип «real_выражение», т.е. вместо этой переменной возможно подставлять сложные выражения, имеющие в качестве типа результата real. Тело определяется как новая константа, для чего используется встроенный в ядро тип «new». В качестве результата применения шаблона возвращает пустая команда. В действии шаблона переданное значение скорости связывается с константой, передаваемой в переменной «тело».

Теперь можно в удобном для человека виде описывать различные задачи, связанные с данной онтологией. Например:

```
Пешеход равномерно движется со скоростью 5. Пешеход's время = 2.
Велосипедист равномерно движется со скоростью 6*пешеход's
скорость.
Велосипедист's время = 3*пешеход's время.
```

Системе можно задавать вопросы:

```
Чему равно велосипедист's путь?
Вычисление >> Результат:  180.
```

5 Выводы

- Программные системы формирования онтологий создаются и развиваются уже более 10 лет. Успех и значимость этого направления очевидны.
- Однако, темп внедрения технологий, основанных на онтологическом подходе (онтотехнологий), особенно, в среде Web, все еще невелик. Пока практические успехи получены при финансовой поддержке государственных органов, либо внутри больших корпораций.
- Для широкого внедрения онтотехнологий предлагается строить системы управления онтологиями с использованием следующих трех принципов.

Три принципа построения новых баз онтологий:

1. Онтологии строятся в стиле Wiki с поддержкой модульности, коллективной работы, версий и системы согласований.
2. В системе поддерживается среда открытого языка работы с онтологиями, который формируется самими пользователями, по мере пополнения базы онтологий.
3. Вместе с текстом онтологии в системе формируется внутреннее представление онтологии, которое используется при семантическом анализе выражений языка, при формировании ответов на запросы к онтологии и ее отладке, при межмашинном обмене онтологиями в некотором стандарте и при использовании онтологий в приложениях.

Список литературы

1. Бениаминов Е.М. Основания категорного подхода к представлению знаний. Категорные средства.//Изв. АН СССР Техн. кибернетика, №2, 21-33, (1988).
2. Сайт KIF (http://ksl.stanford.edu/knowledge-sharing/kif).
3. Сайт OWL (http://www.w3.org/TR/owl-ref).
4. Страница проекта EZOP (http://ezop-project.wiki.sourceforge.net/).
5. Страница проекта Semantic Web (http://en.wikipedia.org/wiki/Semantic_Web).
6. Страница проекта Semantic Wiki (http://en.wikipedia.org/wiki/Semantic_wiki).

Методы машинного обучения, основанные на индукции правил
Machine Learning Techniques Based on Rule Induction

Воронцов К. В.
Konstantin Vorontsov

Москва, Вычислительный центр им. А. А. Дородницына РАН
Moscow, Dorodnicyn Computing Centre of RAS
http://www.ccas.ru/voron, vokov@forecsys.ru

Аннотация В статье представлен обзор логических алгоритмов классификации. Рассматриваются различные критерии информативности, алгоритмы поиска информативных закономерностей, построения решающих списков и бустинга закономерностей.

This paper gives a short introduction to rule induction classifiers.

In introduction some examples of application domains are given in order to show the usefulness of the rule induction approach to classification. These domains are: medical diagnostics, credit scoring and spam detection. The notion of *rule* is introduced informally from the point of view of two basic requirements: the informativeness and the interpretability.

In section 1 several definitions of informativeness are introduced. First is a heuristical ε, δ-criterion which is really a pair of criteria: the accuracy and the coverage of the rule. The aggregation of both criteria to a scalar criterion is not so easy task as might seem. Examples are given. Then, the Fisher Exact Test (FET) is described, which is a statistically valid criterion. FET is asymptotically equivalent to the popular information gain entropy criterion, but it is more strict on small samples. The calculation of FET can be very effective if the table of logarithms of factorials is stored in advance. FET is almost always less restrictive than the heuristical criterion. Nevertheless, both are useful in rule induction algorithms: the statistical criterion is preferable during an iterative rule enhancement, whereas heuristical one is more convenient for a final selection of best rules.

In section 2 a local algorithms of the conjunction rules search are described. A common algorithm is given, which can be easily specialized to several well known algorithms: the greedy conjunction growing, the stochastic local search, the stabilization and the pruning procedures, which help to improve a rule. The upgrade of the local search algorithm to the evolutionary (genetic) search engine is discussed in brief.

In section 3 another forms of rules are introduced: balls and hyperplains. To be well interpretable they must depend from a small number of features. In further sections two classifiers are considered that can use any form of rules.

In section 4 the decision list learning algorithm is considered. The tradeoff between the complexity and the generalization ability of the classifier is discussed. Several variants of decision lists known in the literature are mentioned.
In section 5 the weighted voting learning algorithm is considered based on the famous AdaBoost algorithm. The outliers elimination is discussed. AdaBoost can serve is an universal envelope for any kind of base classifiers, particularly, for any forms of rules: conjunctions, balls of hyperplanes. As an example the SLIPPER (simple learner with iterative pruning to produce error reduction) algorithm is considered, which is a combination of boosting with greedy local search and pruning of conjunction rules.

Рассматривается задача классификации в стандартной постановке: имеется пространство *объектов* X и конечное множество *классов* $Y = \{1, \ldots, M\}$. Объекты описываются набором n *признаков* $f_j \colon X \to D_j$, $j = 1, \ldots, n$, где D_j — множество допустимых значений j-го признака. *Целевой признак* $y^* \colon X \to Y$ известен только на объектах *обучающей выборки* $X^\ell = (x_i, y_i)_{i=1}^\ell$, где $x_i \in X$, $y_i = y^*(x_i)$. Требуется построить *функцию классификации*, называемую также классификатором, $a \colon X \to Y$, которая по признаковому описанию произвольного объекта x предсказывала бы его класс $y^*(x)$, то есть приближала бы неизвестную функцию y^* на всём множестве X.

Задачи классификации возникают в различных предметных областях.

В задачах *распознавания спама* необходимо разделить поток входящих текстовых сообщений на два класса — желательные и нежелательные (спам). Обучающими объектами являются сообщения, классифицированные данным пользователем. Признаками являются: наличие определённых символов или ключевых слов в тексте письма, язык сообщения, домен отправителя, число адресатов в рассылке, и т. д.

В задачах *кредитного скоринга* признаки характеризуют социально-экономическое положение заёмщика (физического лица). Обучающими объектами являются заёмщики с известной кредитной историей. Признаки — это ответы на вопросы анкеты. В простейшем случае принятие решений сводится к классификации на два класса: выдать кредит или отказать.

В задачах *медицинской диагностики* объекты — это пациенты. Признаки характеризуют результаты обследований, симптомы заболевания и применявшиеся методы лечения. Целевым признаком может быть диагноз, исход заболевания или рекомендуемый способ лечения. Обучающая выборка представляет собой формализованные истории болезни тех пациентов, для которых целевой признак уже известен.

Особенностью медицинских задач является наличие *синдромов* — таких сочетаний признаков, при которых диагноз ставится достаточно надёжно. Каждый синдром образуется небольшим числом признаков, имеет понятную эксперту содержательную интерпретацию, но классифицирует лишь определённую часть объектов. При этом знания совокупности синдромов может оказаться достаточно для классификации практически любых объектов.

Существование синдромов характерно для многих предметных областей. В машинном обучении синдромы принято называть *правилами* (rule) [14] или *закономерностями* [3,8], а задачу выявления синдромов по обучающей выборке — *индукцией правил* (rule induction, rule learning) или *индуктивным выводом закономерностей*.

Пусть $\varphi\colon X \to \{0,1\}$ — некоторый предикат, определённый на множестве объектов X. Говорят, что предикат φ *выделяет* объект x, если $\varphi(x) = 1$.

Предикат называют *закономерностью*, если он является информативным, то есть выделяет достаточно много объектов какого-то одного класса c, и практически не выделяет объекты других классов. Предикат называется *правилом*, если он описывается простой, содержательно интерпретируемой формулой, зависящей от небольшого числа признаков.

Пример закономерности из области медицины: если возраст пациента выше 60 лет и ранее он перенёс инфаркт, то операцию делать не стоит. Пример из задачи кредитного скоринга: если заёмщик указал в анкете свой домашний телефон, и его зарплата превышает \$2000 в месяц, и сумма кредита не превышает \$5 000, то риск дефолта мал. Пример из задачи распознавания спама: если в письме присутствует слово «бесплатно», телефонный номер и домен отправителя находится в Китае, то это спам.

Во всех перечисленных случаях правило представляет собой конъюнкцию небольшого числа бинарных признаков. Информативные правила такого вида называют *логическими закономерностями*. На практике этот вид закономерностей используется чаще всего, поскольку он позволяет представлять функцию классификации в виде набора правил «если–то».

Композиции логических закономерностей образуют обширный класс *логических классификаторов*, включающий решающие списки и деревья, простое и взвешенное голосование конъюнкций, и другие конструкции.

Расширенная версия данного обзора находится на сайте [10].

1 Понятия информативности и закономерности

Эвристическое определение информативности. Предикат $\varphi(x)$ тем более информативен, чем больше он выделяет объектов «своего» класса $c \in Y$ по сравнению с объектами всех остальных «чужих» классов. Свои объекты называют также *позитивными* (positive), а чужие — *негативными* (negative). Введём следующие обозначения:

P_c — число объектов класса c в выборке X^ℓ;

$p_c(\varphi)$ — из них число объектов, для которых $\varphi(x) = 1$;

N_c — число объектов всех остальных классов $Y \setminus \{c\}$ в выборке X^ℓ;

$n_c(\varphi)$ — из них число объектов, для которых $\varphi(x) = 1$.

Для краткости индекс c и аргумент (φ) будем иногда опускать. Предполагается, что $P \geqslant 1$, $N \geqslant 1$ и $P + N = \ell$.

Предикат $\varphi(x)$ будем называть ε, δ*-закономерностью* для класса $c \in Y$, если доля негативных среди всех выделяемых объектов достаточно мала:

$$E_c(\varphi, X^\ell) = n_c(\varphi) \;/\; (p_c(\varphi) + n_c(\varphi)) \leqslant \varepsilon,$$

Таблица 1. Критерии информативности при $P = 200$, $N = 100$. Пять колонок слева соответствуют простым эвристическим критериям. Три правые колонки соответствуют более адекватным критериям, которые рассматриваются ниже.

p	n	$p-n$	$p-5n$	$\frac{p}{P}-\frac{n}{N}$	$\frac{p}{n+1}$	$\frac{p}{n+p}$	I_c	IGain_c	$\sqrt{p}-\sqrt{n}$
50	0	**50**	50	0.25	50	1	22.65	23.70	7.07
100	50	**50**	−150	0	1.96	0.67	2.33	1.98	2.93
50	9	41	**5**	0.16	**5**	**0.85**	7.87	7.94	4.07
5	0	5	**5**	0.03	**5**	**1**	2.04	3.04	2.24
100	0	**100**	100	**0.5**	100	1	52.18	53.32	10.0
140	20	**120**	40	**0.5**	6.67	0.88	37.09	37.03	7.36

а доля выделяемых позитивных объектов достаточно велика:

$$D_c(\varphi, X^\ell) = p_c(\varphi)/\ell \geqslant \delta.$$

Если $n_c(\varphi) = 0$, то закономерность φ называется *непротиворечивой*. В большинстве практических задач данные являются неполными и неточными; ошибочные классификации неизбежны, поэтому нет смысла ограничиваться поиском только непротиворечивых закономерностей.

Задача построения информативного предиката φ сводится к оптимизации одновременно по двум критериям: $p_c(\varphi) \to \max$ и $n_c(\varphi) \to \min$. При этом возникает дилемма: что лучше — уменьшение n или увеличение p? Для поиска закономерностей удобно иметь один скалярный критерий информативности. Однако изобрести адекватную свёртку критериев n и p не так просто. В обзорной статье [14] приведено около двух десятков различных критериев. На первый взгляд, закономерности можно сравнивать по разности $p-n$ или отношению p/n. В таблице 1 собраны контрпримеры, показывающие неадекватность простых критериев информативности. Жирным шрифтом выделены пары примеров, в которых верхняя закономерность явно информативнее нижней, тем не менее, критерий принимает на них равные значения, или даже большее значение на нижней закономерности.

Статистическое определение информативности. Адекватную скалярную характеристику информативности даёт техника проверки статистических гипотез. Пусть X — вероятностное пространство, выборка X^ℓ случайная и независимая, $y^*(x)$ и $\varphi(x)$ — случайные величины. Если верна гипотеза о независимости событий $\{x\colon y^*(x) = c\}$ и $\{x\colon \varphi(x) = 1\}$, то вероятность реализации пары (p, n) подчиняется гипергеометрическому распределению [12]:

$$h_{P,N}(p,n) = \frac{C_P^p C_N^n}{C_{P+N}^{p+n}}, \quad 0 \leqslant p \leqslant P, \quad 0 \leqslant n \leqslant N, \tag{1}$$

где $C_m^k = \frac{m!}{k!(m-k)!}$ — биномиальные коэффициенты, $0 \leqslant k \leqslant m$.

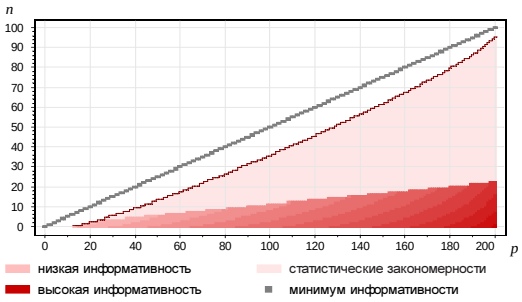


Рис. 1. Области ε, δ-закономерностей (при $\varepsilon = 0.1$) и статистических закономерностей (при $I_0 = 5$) в координатах (p, n) при $P = 200$, $N = 100$

Если вероятность (1) мала, и тем не менее пара (p, n) реализовалась, то гипотеза о независимости должна быть отвергнута. Чем меньше значение вероятности, тем более значимой является связь между y^* и φ.

Информативность предиката $\varphi(x)$ относительно класса $c \in Y$ по выборке X^ℓ определим как $I_c(\varphi, X^\ell) = -\ln h_{P_c,N_c}(p_c(\varphi), n_c(\varphi))$.

Предикат $\varphi(x)$ будем называть *статистической закономерностью* для класса c, если $I_c(\varphi, X^\ell) \geqslant I_0$ при заданном достаточно большом I_0.

Порог информативности I_0 выбирается так, чтобы ему соответствовало достаточно малое значение вероятности. Описанный критерий применяется в статистике для проверки гипотезы о независимости событий и называется *точным тестом Фишера* (Fisher's exact test, FET).

Известны и другие критерии информативности, в частности, энтропийный критерий прироста информации IGain [18]. Он является асимптотическим приближением FET и при малых n или p несколько завышает информативность, см. таблицу 1. Точный критерий может давать в этих ситуациях ощутимые преимущества [16]. Однако на практике чаще используется энтропийный критерий, что трудно объяснить чем-либо, кроме сложившейся традиции. Соображения эффективности вычисления также говорят в пользу FET. Вычисление логарифма $h_{P,N}(p, n)$ сводится к сложению 9 чисел, если заранее сформировать таблицу логарифмов факториалов $L_k = \ln k!$, использовав рекуррентную формулу $L_1 = 0;\ L_k = L_{k-1} + \ln k$.

При разумных сочетаниях параметров ε и I_0 эвристический критерий практически всегда оказывается строже статистического, см. Рис. 1. Имеется обширная область статистических закономерностей, которые допускают слишком много ошибок и не являются ε, δ-закономерностями. Отсюда следует философский вывод: *неслучайность ещё не означает закономерность*; закономерность — это гораздо более сильное свойство, чем неслучайность. При построении классификаторов полезны оба критерия. Статистический чаще применяется для оценки перспективности правил в процессе их оптимизации, логический — для финального отбора закономерностей.

2 Поиск закономерностей в форме конъюнкций

Пусть задано конечное множество *элементарных предикатов* $\mathscr{B}$. Это могут быть исходные бинарные признаки $f_j\colon X \to \{0,1\}$ или *бинаризованные* признаки других типов. Например, для числовых признаков f_j часто используются *пороговые предикаты* (data stamps) вида $\beta(x) = [f_j(x) \leqslant d]$ или $\beta(x) = [d \leqslant f_j(x) \leqslant d']$. Здесь и далее квадратные скобки означают преобразование логического значения в числовое: [ложь] $= 0$, [истина] $= 1$. Для выбора порогов d, d' диапазон значений признака f_j предварительно разбивается на информативные интервалы [10].

Логические закономерности часто ищут в форме конъюнкций элементарных предикатов $\varphi(x) = \beta_1(x) \wedge \cdots \wedge \beta_k(x)$ невысокого ранга, $k \leqslant K$. Они имеют вид привычных для человека логических высказываний и легко поддаются содержательной интерпретации. Максимальный ранг K обычно устанавливают от 2 до 7, в зависимости от прикладной задачи и представлений эксперта-прикладника об «интерпретируемости» правил.

Поиск наиболее информативных конъюнкций в общем случае требует полного перебора. Представим вполне реалистичную ситуацию: имеется 100 числовых признаков; диапазон значений каждого признака разбит на 10 интервалов. Тогда число конъюнкций ранга K равно $C_{100}^K 10^K$. Уже при $K = 5$ эта величина имеет порядок 10^{13}, что исключает возможность полного перебора на современных компьютерах. Поэтому на практике используются различные эвристические алгоритмы сокращённого перебора.

Алгоритмы локального поиска. Начиная с заданной конъюнкции φ_0 (например, пустой), строится последовательность конъюнкций $\varphi_0, \ldots, \varphi_t, \ldots$, в которой каждая следующая конъюнкция φ_t выбирается из окрестности предыдущей $V(\varphi_{t-1})$. *Окрестностью* конъюнкции φ называется множество конъюнкций $V(\varphi)$, получаемых из φ путём добавления, изъятия или модификации одного из термов. Конъюнкции с максимальной информативностью являются наиболее перспективными с точки зрения их дальнейшего улучшения. Тем не менее, они могут допускать слишком много ошибок. Поэтому на каждом шаге t выделяется «наилучшая» конъюнкция, удовлетворяющая дополнительному условию $E_c(\varphi_t^*) < \varepsilon$, и в общем случае не обязательно совпадающая с наиболее перспективной φ_t.

Алгоритм 1 принимает на входе обучающую выборку X^ℓ, начальное приближение φ_0 и класс, для которого строится конъюнкция, $c \in Y$. Параметры алгоритма: максимальное число итераций $t_{\max}$, критерий остановки d, порог доли ошибок для отбора наилучших конъюнкций ε. На выходе формируется одна конъюнкция φ. Критерий информативности I_c и генератор окрестностей V, вообще говоря, также являются параметрами алгоритма. Варьируя параметры Алгоритма 1, можно получать различные процедуры поиска или улучшения информативных конъюнкций.

Жадный алгоритм синтеза конъюнкции использует только операцию добавления термов. Начальным приближением является пустая конъюнкция (не содержащая термов). Недостаток жадной стратегии в том, что она

Алгоритм 1. Локальный поиск информативной конъюнкции.

$I^* := I_c(\varphi_0, X^\ell)$;
для всех $t = 1, \ldots, t_{\max}$
  текущее множество перебираемых конъюнкций: $V_t := V(\varphi_{t-1})$;
  наиболее перспективная конъюнкция: $\varphi_t := \arg\max_{\varphi \in V_t} I_c(\varphi, X^\ell)$;
  наилучшая конъюнкция на t-й итерации: $\varphi_t^* := \arg\max_{\varphi \in V_t:\, E_c(\varphi) < \varepsilon} I_c(\varphi, X^\ell)$;
  если $I_c(\varphi_t^*) > I^*$ **то**
    запомнить, на какой итерации была получена наилучшая конъюнкция:
    $t^* := t$; $\varphi^* := \varphi_t^*$; $I^* := I_c(\varphi^*)$
  если $t - t^* > d$ (конъюнкция не улучшилась за последние d шагов) **то**
    выход;
вернуть φ^*;

может уводить в сторону от глобального максимума информативности. Терм, найденный на k-м шаге, перестаёт быть оптимальным после добавления последующих термов. Тем не менее, в простых задачах классификации эта эвристика позволяет находить неплохие закономерности.

Стохастический локальный поиск (stochastic local search, SLS) также начинает с пустой конъюнкции, но использует полный набор возможных модификаций, включая удаления и замены неоптимальных термов. С другой стороны, мощность окрестности $|V(\varphi)|$ может оказаться настолько большой, что перебор всех допустимых модификаций станет нерентабельным. Поэтому в SLS строится не вся окрестность, а только некоторое её случайное подмножество. Максимальная допустимая мощность этого подмножества задаётся как дополнительный параметр алгоритма $V_{\max}$.

Для улучшения построенной конъюнкции φ к ней применяют методы «финальной шлифовки» — стабилизацию и редукцию.

Процедура стабилизации локально улучшает конъюнкцию φ, удаляя или заменяя по одному терму. В отличие от SLS, перебираются все возможные удаления и замены. Модификации производятся до тех пор, пока возрастает информативность конъюнкции $I_c(\varphi, X^\ell)$. Стабилизация повышает устойчивость закономерностей относительно малых вариаций обучающей выборки или других условий обучения (например, генератора псевдослучайных чисел в SLS). Если алгоритм SLS использовался многократно для построения большого числа правил, то в результате стабилизации число различных правил может сократиться во много раз. Обычно это положительно сказывается на интерпретируемости. Эксперт больше доверяет правилу, когда видит, что попытки скорректировать его «вручную» приводят только к его ухудшению.

Процедура редукции отличается тем, что термы только удаляются, а информативность вычисляется по *контрольной выборке* X^k, составленной из объектов, не участвовавших в построении конъюнкции φ. Контрольную выборку формируют до начала обучения, выделяя из массива исходных данных около 30% объектов, как правило, случайным образом. Объекты разных

классов должны быть представлены в одинаковых пропорциях на обучении (этот принцип отбора называется *стратификацией* выборки). Смысл редукции в том, чтобы проверить, не является ли найденная конъюнкция избыточно сложной, и либо упростить её, либо вовсе признать неудачной. Упрощение повышает общность логического правила, поскольку множество выделяемых им объектов расширяется. Недостаток редукции в том, что она требует оставить значительную долю данных для контроля, уменьшив представительность обучающей выборки. Однако при разумном выборе соотношения $\ell : k$ поиск правил по X^ℓ с последующей редукцией по X^k может давать лучшие результаты, чем поиск по $X^\ell \cup X^k$ без редукции.

Генетический алгоритм синтеза конъюнкций можно рассматривать как дальнейшее усовершенствование SLS на основе идей дарвиновской эволюции. Главное отличие от SLS в том, что на каждом шаге отбирается не одна наилучшая конъюнкция, а целое множество лучших конъюнкций, называемое *популяцией*. Из них порождается большое количество конъюнкций-*потомков* с помощью двух *генетических операций* — скрещивания и мутации. *Скрещивание* (crossingover) — это образование новой конъюнкции путём обмена термами между двумя членами популяции. В роли *мутаций* выступают те же операции добавления, замещения и удаления термов. Вероятность мутации p_m является параметром алгоритма. Полученное множество из N конъюнкций-потомков проходит *естественный отбор*, в результате которого в следующее поколение переходят не более n наиболее информативных конъюнкций, где n и N — ещё два параметра алгоритма. Генетические алгоритмы отличаются большим разнообразием всевозможных эвристик, заимствованных из живой природы. Часто используется *искусственный отбор* — вероятность стать родителем увеличивалась с ростом информативности. Поскольку функционал информативности является многоэкстремальным, имеет смысл организовывать несколько параллельно развивающихся популяций (островная модель эволюции), чтобы увеличить разнообразие конъюнкций. Эти и другие эвристики описаны в обширной литературе по генетическим алгоритмам, см. например [22,9,1,2].

3 Другие виды закономерностей

Конъюнкции, составленные из термов вида $\beta(x) = [d \leqslant f(x) \leqslant d']$ описывают области пространства X, имеющие форму гиперпараллелепипедов. На практике применяются и другие формы закономерностей например, шары или гиперплоскости. Выбор формы определяется особенностями конкретной задачи. В общем случае к форме предъявляются два требования — интерпретируемости и эффективности решения оптимизационной задачи $I(\varphi, X^\ell) \to \max_\varphi$. Интерпретируемость, в частности означает, что правило $\varphi(x)$ должно зависеть от небольшого подмножества признаков $\omega \subset \{1, \ldots, n\}$.

Параметрическое семейство шаров:

$$\varphi_c(x) = \left[\sum_{j\in\omega} \alpha_j |f_j(x) - f_j(x_0)|^p \leqslant r_0^p\right], \tag{2}$$

где центр шара x_0, радиус шара r_0 и веса признаков α_j являются параметрами правила. Выделение объекта x шарообразной закономерностью $\varphi(x)$ легко интерпретируется в терминах сходства: «объект x относится к классу c потому, что он близок к эталонному объекту x_0 класса c по совокупности признаков ω». Такого рода объяснения хорошо воспринимаются в тех предметных областях, где распространён прецедентный стиль мышления — в медицине, геологии, социологии, юриспруденции, и др. На закономерностях данного типа основаны *алгоритмы вычисления оценок* [5].

Параметрическое семейство полуплоскостей:

$$\varphi_c(x) = \left[\sum_{j\in\omega} \alpha_j f_j(x) \leqslant \alpha_0\right], \tag{3}$$

где параметрами являются веса признаков α_j и смещение α_0 разделяющей гиперплоскости. Максимизация информативности сводится к подбору таких α и α_0, при которых по одну сторону гиперплоскости лежат объекты преимущественно одного класса. Если признаки обладают свойством монотонности — «чем выше значение признака $f_j(x)$, тем скорее объект x относится к классу c», то коэффициенты α_j должны оказаться неотрицательными, в таком случае они интерпретируются как степени важности признаков.

4 Решающие списки

Решающий список — это классификатор $a\colon X \to Y$, который задаётся набором закономерностей $\varphi_1(x), \ldots, \varphi_T(x)$, приписанных, соответственно, к классам $c_1, \ldots, c_T \in Y$. Объект x относится к классу c_t, если φ_t — первая в списке закономерность, выделившая данный объект: $\varphi_1(x) = \cdots = \varphi_{t-1}(x) = 0$ и $\varphi_t(x) = 1$. Если ни одна из закономерностей не выделяет объект x, то список отказывается от классификации. Тогда объект x можно отнести к классу с минимальной ценой ошибки. Например, в задаче выдачи кредитов отказ приведёт к более осторожному решению «не выдавать». В задаче распознавания спама более осторожным будет решение «не спам».

Жадный алгоритм построения решающего списка. Алгоритм 2 на каждой итерации строит ровно одно правило φ_t, выделяющее максимальное число объектов некоторого класса c_t и минимальное число объектов всех остальных классов. Для выбора класса c_t на шаге 3 можно придерживаться различных стратегий, например, выбирать тот класс, за который удаётся построить наиболее информативное правило. Логика решающего списка будет более понятна эксперту, если сначала построить все правила одного

Алгоритм 2. Жадный алгоритм построения решающего списка.

1: $U := X^\ell$;
2: **для всех** $t := 1, \ldots, T_{\max}$
3: $c := c_t$ — выбрать класс из Y, для которого будет строиться правило;
4: найти наиболее информативное правило при ограничении на долю ошибок:
$\varphi_t := \arg\max_{\varphi \in \Phi'} I_c(\varphi, U)$, где $\Phi' = \{\varphi \in \Phi \colon E_c(\varphi, U) \leqslant E_{\max}\}$;
5: **если** $I_c(\varphi_t, U) < I_{\min}$ **то выход**;
6: исключить из выборки объекты, выделенные правилом φ_t:
$U := \{x \in U : \varphi_t(x) = 0\}$;
7: **если** $|U| \leqslant \ell_0$ **то выход**;

класса, затем все правила второго класса, и т. д. На шаге 4 производится поиск наиболее информативного правила $\varphi_t \in \Phi$, допускающего относительно мало ошибок, то есть используется сразу два критерия отбора правил I_c и E_c. Семейство правил Φ может быть каким угодно, лишь бы для него существовала эффективная процедура поиска закономерностей. В частности, если Φ — конъюнкции, то можно применить Алгоритм 1. После построения правила φ_t выделенные им объекты изымаются из выборки и алгоритм переходит к поиску следующего правила φ_{t+1} по оставшимся объектам.

Алгоритм 2 принимает на входе обучающую выборку X^ℓ и параметры: $T_{\max}$ — максимальное число правил в списке, $I_{\min}$ — минимальная информативность правил в списке, $E_{\max}$ — максимальная доля ошибок на обучающей выборке, ℓ_0 — максимальное допустимое число отказов.

Параметр $E_{\max}$ позволяет найти компромисс между точностью классификации обучающей выборки и сложностью списка. Уменьшение $E_{\max}$ снижает число ошибок на обучении, но поскольку отбор правил становится жёстче, уменьшается число объектов, выделяемых каждым отдельным правилом, и увеличивается длина списка T. Правила, выделяющие слишком мало объектов, статистически не надёжны и могут допускать много ошибок на независимых контрольных данных. Увеличение длины списка при одновременном «измельчении» правил может приводить к переобучению. На практике параметр $E_{\max}$ подбирается по скользящему контролю.

Жадный алгоритм построения решающего списка переизобретался огромное число раз и известен под разными названиями, в частности, как комитет с логикой старшинства (seniority voting) [17] и машина покрывающих множеств (set covering machine, SCM) [15]. Многочисленные варианты отличаются семейством предикатов Φ, критерием информативности и методом поиска информативных предикатов. Алгоритм BuildSCM [15] строит покрытие обучающей выборки шарами (data dependent balls). Алгоритм дробящихся эталонов ДРЭТ [6] также основан на покрытии выборки шарами, но отличается тем, что список строится от конца к началу. В алгоритме Маршанда [20] перебираются всевозможные гиперплоскости, разделяющие какие-нибудь три точки (data dependent half-spaces), и из них выбирается полуплоскость с максимальной информативностью. Алгоритм Белецко-

го [7] строит полуплоскости, отделяющие как можно больше объектов одного класса. Для этого применяются методы линейного программирования. В отличие от жадного Алгоритма 2, после построения каждой полуплоскости оптимизируется положение предыдущих полуплоскостей.

5 Взвешенное голосование (бустинг) закономерностей

Пусть для каждого класса $c \in Y$ построен набор закономерностей $R_c = \{\varphi_c^t(x),\ t = 1, \ldots, T_c\}$. *Простое голосование* (simple voting) — это классификатор, относящий объект x к тому классу, для которого доля закономерностей, выделяющих данный объект, максимальна:

$$a(x) = \arg\max_{c \in Y} \Gamma_c(x), \quad \Gamma_c(x) = \frac{1}{T_c} \sum_{t=1}^{T_c} \varphi_c^t(x).$$

Если максимум достигается одновременно на нескольких классах, выбирается тот, для которого цена ошибки меньше. Нормирующий множитель $1/T_c$ вводится для того, чтобы наборы с бóльшим числом правил не перетягивали объекты в свой класс.

Взвешенное голосование (weighted voting) действует более тонко, учитывая, что правила могут иметь различную ценность. Каждому правилу φ_c^t приписывается вес $\alpha_c^t \geqslant 0$, и вычисляются взвешенные суммы голосов:

$$\Gamma_c(x) = \sum_{t=1}^{T_c} \alpha_c^t \varphi_c^t(x), \quad \alpha_c^t \geqslant 0. \tag{4}$$

Простой общий подход к настройке весов α_c^t заключается в том, чтобы использовать правила $\varphi_c^t(x)$ вместо признаков и построить в этом новом признаковом пространстве линейную разделяющую поверхность (кусочно-линейную, если $|Y| > 2$). Для этого можно использовать логистическую регрессию или метод опорных векторов [21].

На первый взгляд, вес правила должен определяться его информативностью. Однако, важно ещё, насколько данное правило уникально. Если имеется 10 хороших, но одинаковых (или почти одинаковых) правил, то их суммарный вес должен быть сравним с весом столь же хорошего правила, не похожего на все остальные. Таким образом, веса должны учитывать не только ценность правил, но и их различность.

Существует простое теоретико-вероятностное обоснование *принципа диверсификации* (повышения различности) правил [4]. Пусть X — вероятностное пространство, множество ответов Y конечно. Введём случайную величину $M(x)$, равную перевесу голосов в пользу правильного класса; её называют также *отступом* (margin) объекта x от границы классов:

$$M(x) = \Gamma_y(x) - \max_{c \in Y \setminus \{y\}} \Gamma_c(x), \quad y = y^*(x).$$

Если отступ положителен, $M(x) > 0$, то объект x классифицируется правильно. Предположим, что в среднем наш классификатор работает хотя бы немного лучше, чем наугад: $\mathsf{E}M > 0$. Тогда можно оценить вероятность ошибки по неравенству Чебышёва:

$$\mathsf{P}\{M < 0\} \leqslant \mathsf{P}\big\{|\mathsf{E}M - M| > \mathsf{E}M\big\} \leqslant \frac{\mathsf{D}M}{(\mathsf{E}M)^2}.$$

Таким образом, для уменьшения вероятности ошибки необходимо максимизировать ожидание перевеса голосов $\mathsf{E}M$ и минимизировать его дисперсию $\mathsf{D}M$. Для выполнения этих условий объекты должны набрать примерно одинаковый суммарный вес. На практике ни одно из правил не выделяет класс целиком (иначе задача решалась бы слишком просто), поэтому правила должны быть существенно различными.

Алгоритм бустинга (boosting) решает одновременно обе проблемы — в процессе построения закономерностей увеличивает их различность и одновременно определяет их веса. Наиболее известен алгоритм AdaBoost, изначально предложенный для построения линейных комбинаций произвольных классификаторов [13]. Мы рассмотрим его вариант, приспособленный для взвешенного голосования закономерностей [19,11].

Рассмотрим задачу классификации с двумя классами, $Y = \{-1, +1\}$ и взвешенное голосование $T = T_{-1} + T_{+1}$ закономерностей:

$$a_T(x) = \operatorname{sign}\Bigg(\underbrace{\sum\nolimits_{t=1}^{T_{+1}} \alpha_{+1}^t \varphi_{+1}^t(x)}_{\Gamma_{+1}(x)} - \underbrace{\sum\nolimits_{t=1}^{T_{-1}} \alpha_{-1}^t \varphi_{-1}^t(x)}_{\Gamma_{-1}(x)}\Bigg), \qquad \alpha_c^t > 0,\ c \in Y.$$

Пусть уже построено T закономерностей, вместе составляющих классификатор $a_T(x)$. При добавлении ещё одной закономерности $\varphi_c(x)$ в список R_c взвешенная сумма голосов за класс c примет вид $\Gamma_c(x) + \alpha\varphi_c(x)$. Задача состоит в том, чтобы найти закономерность φ_c и её вес α, при которых классификатор $a_{T+1}(x)$ допускает минимальное число ошибок на обучающей выборке X^ℓ. Перед добавлением число ошибок есть

$$Q_T = \sum_{i=1}^{\ell} \left[\Gamma_{y_i}(x_i) - \Gamma_{-y_i}(x_i) < 0\right].$$

Число ошибок $a_{T+1}(x)$ после добавления закономерности φ_c:

$$Q_{T+1}(\varphi_c, \alpha) = \sum_{i=1}^{\ell} [y_i = c] \left[\Gamma_{y_i}(x_i) - \Gamma_{-y_i}(x_i) + \alpha\varphi_c(x_i) < 0\right] + \\ + \sum_{i=1}^{\ell} [y_i \neq c] \left[\Gamma_{y_i}(x_i) - \Gamma_{-y_i}(x_i) - \alpha\varphi_c(x_i) < 0\right].$$

Выписанный функционал содержит параметр α внутри пороговой функции вида $[z(\alpha) < 0]$, следовательно, является разрывной функцией от α.

Минимизация такого функционала является нетривиальной задачей комбинаторной оптимизации. Для приближённого её решения заменим пороговую функцию непрерывно дифференцируемой оценкой сверху. Выбор конкретной оценочной функции является эвристикой. В алгоритме AdaBoost используется оценка $[z < 0] \leqslant e^{-z}$.

Запишем верхнюю оценку $\tilde{Q}_T$ функционала Q_T:

$$Q_T \leqslant \tilde{Q}_T \equiv \sum_{i=1}^{\ell} \underbrace{\exp\bigl(\Gamma_{-y_i}(x_i) - \Gamma_{y_i}(x_i)\bigr)}_{w_i} = \sum_{i=1}^{\ell} w_i.$$

Если классификатор $a_T(x)$ ошибается на объекте x_i, то $w_i > 1$, иначе $w_i < 1$. Чем больше перевес голосов в пользу ошибочного класса, тем больше вес w_i. Таким образом, больший вес получают наиболее «трудные» объекты.

Введём вектор $w = (\tilde{w}_i)_{i=1}^{\ell}$ с компонентами $\tilde{w}_i = w_i \ell / \tilde{Q}_T$. Тогда будет выполнено условие нормировки $\sum_{i=1}^{\ell} \tilde{w}_i = \ell$.

Теорема 1 ([19]). *Минимум функционала $\tilde{Q}_{T+1}(\varphi_c, \alpha)$ достигается при*

$$\varphi_c = \arg\max_{\varphi} J_c(\varphi, X^\ell), \qquad J_c(\varphi, X^\ell) = \sqrt{p_c(\varphi)} - \sqrt{n_c(\varphi)};$$

$$\alpha = \frac{1}{2} \ln \frac{p_c(\varphi_c)}{n_c(\varphi_c)}, \text{ при } n_c(\varphi_c) \neq 0, \tag{5}$$

$$p_c(\varphi) = \sum_{i=1}^{\ell} \tilde{w}_i [y_i = c][\varphi(x_i) = 1]; \tag{6}$$

$$n_c(\varphi) = \sum_{i=1}^{\ell} \tilde{w}_i [y_i \neq c][\varphi(x_i) = 1]; \tag{7}$$

Заметим, что определение числа позитивных и негативных объектов $p_c(\varphi)$ и $n_c(\varphi)$, выделяемых предикатом φ, отличается от того, что было дано в первом разделе. Здесь объекты взвешиваются с весами $\tilde{w}_i$.

Теорема 1 не позволяет определить вес непротиворечивой закономерности φ_c, так как знаменатель (5) обращается в нуль. Непротиворечивые закономерности бесконечно выгодны с точки зрения минимизации функционала $\tilde{Q}_{T+1}$. Чтобы предотвратить этот нежелательный эффект, в знаменатель вводят дополнительное слагаемое $\lambda \in (0, 1)$. Уменьшая параметр λ, можно ориентировать алгоритм на поиск непротиворечивых закономерностей.

Алгоритм 3 принимает на входе обучающую выборку X^ℓ и два параметра: число закономерностей всех классов T и коэффициент поощрения непротиворечивых закономерностей λ. На выходе получаются списки закономерностей и их весов $\bigl\{\varphi_c^t(x), \alpha_c^t \bigm| t = 1, \ldots, T_c\bigr\}$ для всех классов $c \in Y$.

В процессе работы алгоритма имеет смысл проанализировать распределение весов объектов. Объекты с наибольшими весами w_i являются наиболее «трудными» для всех построенных закономерностей. Возможно, это «шумовые выбросы» — объекты, в описании которых допущены грубые ошибки. Исключение таких объектов, называемое *цензурированием выборки*, как

Алгоритм 3. Бустинг закономерностей при классификации на два класса.

1: инициализировать веса: $w_i := 1$ для всех $i = 1, \ldots, \ell$;
2: **для всех** $t = 1, \ldots, T$
3: $c := c_t$ — выбрать класс, для которого будет строиться закономерность;
4: $\varphi_c^t := \arg\max_{\varphi \in \Phi} \sqrt{p_c(\varphi)} - \sqrt{n_c(\varphi)}$;
5: $\alpha_c^t := \frac{1}{2} \ln \frac{p_c(\varphi_c^t)}{\max\{n_c(\varphi_c^t), \lambda\}}$;
6: **для всех** $i = 1, \ldots, \ell$ пересчитать вес w_i:
7: $$w_i := \begin{cases} w_i, & \varphi_c^t(x_i) = 0; \\ w_i \exp(-\alpha_c^t), & \varphi_c^t(x_i) = 1 \text{ и } y_i = c; \\ w_i \exp(\alpha_c^t), & \varphi_c^t(x_i) = 1 \text{ и } y_i \neq c; \end{cases}$$
8: нормировать веса:
$Z := \frac{1}{\ell} \sum_{i=1}^{\ell} w_i; \quad w_i := w_i / Z$ для всех $i = 1, \ldots, \ell$;

правило, повышает качество классификации. После исключения выбросов построение закономерностей лучше начать заново, поскольку выбросы мешали адекватно оценивать информативность закономерностей.

Фактически, теорема 1 вводит ещё один функционал информативности предикатов $J_c(\varphi) = \sqrt{p_c(\varphi)} - \sqrt{n_c(\varphi)}$. Как видно из таблицы 1, он достаточно адекватно оценивает качество закономерностей, а вычисляется существенно проще, чем I_c или IGain_c. Его можно применять не только в алгоритме бустинга, но и отдельно, поскольку теорема 1 остаётся верна для функционала $\tilde{Q}_1$, если положить все веса w_i равными единице.

Следствием теоремы 1 является теорема о сходимости. Чтобы гарантировать построение классификатора, не допускающего ошибок на обучающей выборке, достаточно потребовать, чтобы на шаге 4 всякий раз удавалось найти закономерность φ_c^t со строго положительной информативностью $J_c(\varphi_c^t)$. Иными словами, закономерности должны классифицировать объекты хоть немного лучше, чем наугад.

Для максимизации информативности $J_c(\varphi)$ на шаге 4 можно воспользоваться локальным поиском закономерностей, заменив в Алгоритме 1 критерий информативности I_c на J_c. Комбинация жадного добавления термов с процедурой редукции по контрольной выборке лежит в основе алгоритма SLIPPER (simple learner with iterative pruning to produce error reduction), который показал себя одним из лучших логических алгоритмов в обширном эксперименте на 32 реальных задачах классификации [11].

Список литературы

1. *Емельянов, В. В., Курейчик, В. В., Курейчик, В. М.* Теория и практика эволюционного моделирования. — М.: Физматлит, 2003. — С. 432.
2. *Гладков Л. А., Курейчик В. В., Курейчик В. М.* Генетические алгоритмы. — М.: Физматлит, 2006. — С. 320.

3. *Журавлёв, Ю. И., Рязанов, В. В., Сенько, О. В* «Распознавание». Математические методы. Программная система. Практические применения. — М.: Фазис, 2006.
4. *Полякова, М. П., Вайнцвайг, М. Н.* Об использовании метода «голосования» признаков в алгоритмах распознавания // Моделирование обучения и поведения. — М., 1975. — С. 25—28.
5. *Журавлёв Ю. И.* Об алгебраическом подходе к решению задач распознавания или классификации // *Проблемы кибернетики.* — 1978. — Т. 33. — С. 5–68. .
6. *Лбов Г. С.* Методы обработки разнотипных экспериментальных данных. — Новосибирск: Наука, 1981.
7. *Белецкий Н. Г.* Применение комитетов для многоклассовой классификации // Численный анализ решения задач линейного и выпуклого программирования. — Свердловск, 1983. — С. 156–162.
8. *Загоруйко Н. Г.* Прикладные методы анализа данных и знаний. — Новосибирск: ИМ СО РАН, 1999.
9. *Редько В. Г.* Эволюционная кибернетика. — М.: Наука, 2003. — С. 155.
10. *Воронцов К. В.* Лекции по логическим алгоритмам классификации. — www.ccas.ru/voron/teaching.html. — 2007.
11. *Cohen W. W., Singer Y.* A simple, fast and effective rule learner // Proc. of the 16 National Conference on Artificial Intelligence. — 1999. — Pp. 335–342. .
12. *Dubner P. N.* Statistical tests for feature selection in KORA recognition algorithms // *Pattern Recognition and Image Analysis.* — 1994. — Vol. 4, no. 4. — P. 396.
13. *Freund Y., Schapire R. E.* A decision-theoretic generalization of on-line learning and an application to boosting // European Conference on Computational Learning Theory. — 1995. — Pp. 23–37.
14. *Fürnkranz J., Flach P. A.* Roc 'n' rule learning-towards a better understanding of covering algorithms // *Machine Learning.* — 2005. — Vol. 58, no. 1. — Pp. 39–77 .
15. *Marchand M., Shawe-Taylor J.* Learning with the set covering machine // Proc. 18th International Conf. on Machine Learning. — Morgan Kaufmann, San Francisco, CA, 2001. — Pp. 345–352.
16. *Martin J. K.* An exact probability metric for decision tree splitting and stopping // *Machine Learning.* — 1997. — Vol. 28, no. 2-3. — Pp. 257–291. .
17. *Osborne M. L.* The seniority logic: A logic for a committee machine // *IEEE Trans. on Comp.* — 1977. — Vol. C-26, no. 12. — Pp. 1302–1306.
18. *Quinlan J.* Induction of decision trees // *Machine Learning.* — 1986. — Vol. 1, no. 1. — Pp. 81–106.
19. *Schapire R. E., Singer Y.* Improved boosting using confidence-rated predictions // *Machine Learning.* — 1999. — Vol. 37, no. 3. — Pp. 297–336.
20. The set covering machine with data-dependent half-spaces / M. Marchand, M. Shah, J. Shawe-Taylor, M. Sokolova // Proc. 20th International Conf. on Machine Learning. — Morgan Kaufmann, 2003. — Pp. 520–527.
21. *Smola A., Schoelkopf B.* A tutorial on support vector regression: Tech. Rep. NeuroCOLT2 NC2-TR-1998-030: 1998.
22. *Whitley D.* An overview of evolutionary algorithms: practical issues and common pitfalls // *Information and Software Technology.* — 2001. — Vol. 43, no. 14. — Pp. 817–831.

Как роботам решить задачу о назначениях?* Can Robots Solve an Assignment Problem?

Н. О. Гаранина
Natalia Garanina

Институт систем информатики им. А.П. Ершова СО РАН,
Новосибирск, Россия
garanina@iis.nsk.su

Аннотация В данной работе представлено несколько вариантов протоколов поведения агентов в мультиагентной системе, представляющей модель частного случая задачи о назначениях, проведен анализ временной и эпистемической сложности представленных протоколов.

1 Введение

Современные исследования мультиагентных систем редко осуществляют анализ алгоритмов поведения агентов как в отношении временной сложности алгоритмов, так и знаний (в смысле [4]), приобретаемых агентами в процессе их деятельности. Данная работа является началом исследований различных общих мультиагентных алгоритмов на предмет их временной сложности и приобретаемых знаний агентов.

В данной работе рассматривается т.н. задача о Роботах на Марсе, в которой роботы должны определить себе путь к укрытию на поверхности так, что он не пересекается с путями других роботов. Более общо эту задачу можно рассматривать как задачу о коллективном распределении ресурсов с некоторым дополнительным условием отсутствия конфликтов. Коллективное распределение ресурсов отличается от централизованного, которое применяется в обычных подходах к этой задаче, тем, что субъект распределения сам себе назначает ресурс, либо сам определяет того, кто может его назначить. То есть коллективное распределение, в отличие от централизованного, соответствует мультиагентному подходу при составлении алгоритмов решения данной задачи.

Распределение ресурсов в задаче о роботах в сравнении с классической задачей о пироге, в которой необходимо разделить целый имеющийся ресурс между агентами, можно считать более простым в том смысле, что пирог уже поделён и участникам осталось только выбрать куски, но и более сложным в том смысле, что предпочтение в выборе куска определяется не столько

* Это исследование поддержано в проекте СО РАН 2/12 “Формальные языки и методы спецификации, анализа и синтеза информационных систем”.

желанием агента(робот может хотеть, чтобы расстояние до укрытия было как можно меньше), сколько зависит от выбора остальных участников дележа (поскольку агент-робот хочет избежать конфликтов, выраженных пересечением путей). То есть мало просто выбрать себе кусок, нужно ещё сделать это так, чтобы этот выбор не задевал ничьих интересов. Отметим, что для агента один и тот же кусок может быть как допустимым, так и нет, в зависимости от выбора остальных агентов.

2 Задача о роботах

2.1 Формулировка

Цель представленной работы – разработать мультиагентные алгоритмы и обсудить их эффективность для решения задачи, которая в классической «одноагентной» постановке может быть сведена к комбинаторной задаче о назначениях в двудольном графе (наибольшем парасочетании минимального веса во взвешенном двудольном графе).

Начнем с неформальной постановки задачи.

> На участке поверхности Марса находятся n роботов. Они могут посылать и принимать сообщения друг от друга (но только в режиме один-один, широковещание использовать невозможно). Роботы могут перемещаться по поверхности Марса. Известно, что на поверхности Марса расположено n укрытий и роботы знают их координаты, как и собственные. Задача каждого робота состоит в том, чтобы добраться до укрытия по пути, который не пересекается с путями других роботов. Предполагается, что в начальный момент времени никакие три “объекта” (объект – это робот или укрытие) не лежат на одной прямой.

В классическом одноагентном случае эта задача может быть решена следующим образом.

Во-первых, переформулируем задачу в виде задачи комбинаторной геометрии на плоскости.

> На плоскости n чёрных точек и n белых точек в общем положении[1]. Необходимо соединить отрезками точки разного цвета так, чтобы эти отрезки не пересекались.

Эту задачу мы будем называть задачей Дейкстры [5,1].

Во-вторых, назовем соединением набор из n отрезков, соединяющих попарно черные и белые точки взаимнооднозначным образом, и докажем, что верна простая

Лемма 1 *Если сумма длин отрезков, участвующих в соединении, минимально возможная, соединение не содержит пересекающихся отрезков.*

[1] т.е. никакие три точки не лежат на одной прямой

Доказательство.
Предположим противное: пусть заданное соединение имеет минимальную сумму длин отрезков, но содержит пересекающиеся отрезки. Пусть эти отрезки $[B_1, W_1]$ и $[B_2, W_2]$ (Рис.2.1 a)). Тогда, очевидно, новое соединение, полученное "переключением" этих отрезков и содержащие новые отрезки $[B_1, W_2]$ и $[B_2, W_1]$ (Рис.2.1 b)), будет иметь меньшую сумму длин отрезков, что противоречит минимальности предыдущего соединения.□

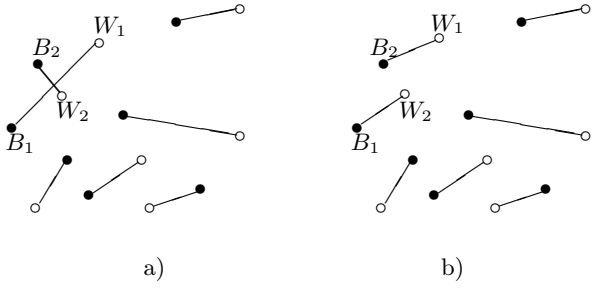


Рис. 1. Переключение отрезков

В-третьих, напомним формулировку задачи о назначениях [2].

> Дан взвешенный двудольный (бихроматический) граф, состоящий из белых и черных вершин и ребер, их соединяющих, каждому из которых приписан некоторый положительный вес[2]. Требуется найти наибольшее паросочетание минимального веса.

Поэтому задача о роботах и укрытиях сводится к следующему частному случаю задачи о назначениях.

> На плоскости n чёрных точек и n белых точек в общем положении. Построим полный двудольный взвешенный граф, состоящий из n чёрных вершин и n белых вершин, в котором вес ребра $[B_i, W_j]$ равен (Евклидову) расстоянию между точками B_i и W_j. Согласно только что доказанной лемме, если решить задачу о назначении для построенного графа, то соответствующий набор отрезков образует соединение без пересечений.

[2] Число черных и белых вершин может быть различным и необязательно, что все черные и белые вершины соединены ребрами.

Как известно, задача о назначениях может быть решена за время $O(n^3)$ например знаменитым венгерским алгоритмом [3].

Однако, у сформулированной выше комбинаторной геометрической задачи Дейкстры есть и другие варианты решения. В частности, мы примем за основу мультиагентных алгоритмов решения задачи о роботах и укрытиях следующий метод, предложенный самим Дейкстрой.

> Сначала черные и белые точки соединяются произвольным образом. Затем пока в соединении есть пересекающиеся отрезки (см. рис. 2.1 a)) происходит переключение так, что пересечение элиминируется (см. рис. 2.1 b)). Переключения происходят до тех пор, пока пересекающихся отрезков не останется.

Этот метод завершается, так как при переключении сумма длин всех отрезков соединения убывает (см. рис 2.1).

Толчком к формулировке мультиагентного варианта задачи о роботах и укрытиях и исследованию мультиагентных алгоритмов ее решения для нас послужила интерпретация “переключения” из метода Дейкстры как локального разрешения конфликта между двумя роботами. В дальнейшем под мультиагентным вариантом задачи о роботах и укрытиях мы будем понимать следующую формулировку.

> На Эвклидовой плоскости расположены n агентов-роботов и n укрытий. Как и ранее предполагается, что в начальный момент времени никакие три “объекта” не лежат на одной прямой. Роботы могут посылать и принимать сообщения друг от друга (но только в режиме один-один, широковещание использовать невозможно). Задача состоит в разработке индивидуального протокола переговоров (правил обмена сообщениями между роботами) для каждого из роботов такого, что после его выполнения робот знает, к какому из укрытий он может двигаться по прямой, не опасаясь столкновений с другими роботами и конкуренции за укрытие.

2.2 Варианты мультиагентных решений

Мы рассмотрим несколько подходов к организации индивидуальных протоколов агентов-роботов.

Первый (“анархический”) подход состоит в том, что все роботы независимы и равноправны. Каждый робот самостоятельно выясняет, не пересечется ли его путь с маршрутами остальных, а если с кем-то пути пересекаются, то роботы участники конфликта разрешают его между собой методом переключения. Когда все роботы (каждый для себя для себя) узнают, что они ни с кем не конфликтуют, тогда каждый робот может самостоятельно двигаться к своему укрытию.

Второй подход (“коллективный”) состоит в самоорганизации роботов, когда в “коллективе” роботов определяется “лидер”, который решает задачу

Дейкстры каким-либо способом для всех роботов, а коллектив роботов принимает его решение.

Возможен также ("командный") подход, который отличается от "коллективного" варианта тем, что роботы изначально разбиты на две "команды", члены которых окрашены в два разных цвета, которые они способны различать. В этом случае в командах определяются "лидеры" своего цвета, которые потом ведут переговоры с "лидером" другого цвета о назначениях укрытий членам своих команд. При переговорах они могут руководствоваться различными соображениями полезности: могут заботиться только о выгоде своих подопечных (например, чтобы сумма их путей до укрытий была как можно меньше), или принимать во внимание общее благо (минимизация общей суммы путей), может также иметь значение время, затрачиваемое на переговоры. Всё это делает особенности "командного" протокола заслуживающими отдельного рассмотрения в дальнейших работах.

Если рассматривать роботов как BDI-агентов, то при разных вариантах решений у них будут различные наборы представлений о мире (Beliefs), целей (Desires) и намерений (Intensions), и, понятное дело, действий. В следующем разделе мы рассмотрим конкретные протоколы, реализующие "анархический" и "коллективный" варианты, и опишем соответствующих агентов. Это просто протоколы автономных агентов, которые пока никак явно не используют свои BDI-возможности для принятия решений о действиях, но взаимодействуют друг с другом посредством обмена сообщениями. Разработка полноценных агентов, принимающих решения, в планах на ближайшее будущее, а пока сделан акцент на анализе временной сложности протоколов и знаний, получаемых в процессе их исполнения.

3 Алгоритмы

3.1 Анархический алгоритм

Все агенты независимы и равноправны, то есть они имеют одинаковые возможности и никакой агент не может заставить другого агента делать что-либо.

Определение. Конфликт – ситуация, в которой пути роботов к укрытиям пересекаются. Конкуренция – это ситуация, когда два (или более) выбрали одно и то же укрытие.

Пусть на роботах установлен строгий линейный порядок $\prec$, заранее известный роботам. Согласно этому порядку они могут распознавать два множества: "меньше себя" и "больше себя". Например, это может быть естественный лексикографический порядок на координатах.

Неформальное описание алгоритма: на первом этапе агенты-роботы выбирают ближайшее к себе укрытие и информируют о выборе всех остальных. Если получается, что несколько роботов выбрали одно и то же укрытие, то это укрытие остаётся за наименьшим (по отношению к принятому порядку) роботом, а остальные выбирают ближайшие к ним незанятые

укрытия. Так продолжается до тех пор, пока есть конкуренция. После этого начинаются раунды переговоров между конфликтующими роботами для разрешения конфликтов. В каждом раунде каждый конфликтующий робот выбирает среди конфликтующих с ним партнёра для переключений (обмена облюбованными укрытиями). Выбрав партнёра, роботы "меняются" укрытиями, после чего могут возникнуть новые конфликты, которые решаются на следующем раунде переговоров. Это продолжается до тех пор, пока есть конфликты. В сущности, агенты вслепую решают задачу Дейкстры, поэтому в некоторый момент времени конфликтов не остается[3]. Вопрос только в том, как каждому индивидуальному роботу узнать, когда конфликтов не осталось.

Выделим явно BDI-особенности в роботах этого алгоритма, а именно: список фактов, которые могут быть в базе знаний агентов, локальные намерения агентов, их цели и возможные действия. Заметим, что записи в базе знаний агента необязательно соответствуют действительности.

Факты: собственные координаты; координаты всех укрытий; координаты всех роботов; укрытия, выбранные роботами в данный момент (точнее – на данном раунде); статус всех роботов (КОНФЛИКТ, МИР)[4]; статус переговоров (переключиться: ДА, НЕТ). Агент хранит *общий* список всех агентов, в котором указаны их координаты, координаты выбранных ими укрытий и их статус. Кроме того, у агента также есть список роботов, конфликтующих с ним в данный момент, в котором указан статус переговоров с ними. Списки периодически обновляются.

Намерения: выбрать укрытие; разрешить хотя бы один конфликт.

Цели: отсутствие конфликтов; занять укрытие.

Действия:

послать сообщение: свои координаты, свой выбор укрытия, свой статус, приглашение переключиться, свой статус переговоров, разрешение двигаться к укрытию;

принять сообщение: координаты робота, выбор укрытия робота, статус робота, приглашение переключиться, статус переговоров робота, разрешение двигаться к укрытию;

вычислить непосредственно конфликтующих роботов;

выбрать укрытие[5];

двигаться к укрытию.

При описании алгоритма примем соглашение, что процессы, помеченные одинаковыми буквами, происходят параллельно и асинхронно, а помеченные разными буквами – в алфавитном порядке. Приём или передача сообщений занимают один такт времени. Отметим, что если агент должен на

[3] О временной сложности пойдет речь позже.

[4] КОНФЛИКТ означает, что путь робота к выбранному укрытию пересекается с путями каких-нибудь роботов, МИР – это отсутствие конфликта

[5] Заметим, что роботы "обмениваются" укрытиями посредством действия "выбрать укрытие".

каком-либо шаге взаимодействовать со *всеми* агентами, он не может перейти к следующему шагу, пока хотя бы один агент остаётся не охваченным. Сразу отметим, что в базу знаний агента с самого начала входит знание собственных координат, координат всех укрытий, и, следовательно, количество как самих укрытий, так и роботов на поверхности Марса.

Анархический Алгоритм

1. a1. Агент выбирает ближайшее укрытие.
a2. Агент сообщает свои координаты всем остальным агентам последовательно в произвольном порядке.
a3. Агент принимает сообщения от всех агентов про их координаты.

После исполнения этого этапа факты агента дополняются знанием координат всех остальных агентов.

2. a1. Агент сообщает о своём выборе всем остальным агентам последовательно в произвольном порядке.
a2. Агент принимает сообщения от всех остальных агентов об их (новом) выборе укрытия.
b1. Если агент знает, что его укрытие выбрал другой агент,
то если он больше того агента,
то выбирает другое свободное укрытие; GOTO 2a.
b2. Если агент знает, что есть укрытие, выбранное двумя роботами,
то GOTO 2a.

После исполнения этого этапа факты агента дополняются знанием выбора укрытия всех остальных агентов на данном раунде.

3. a. Агент вычисляет, с кем его пути пересекаются, составляет список конфликтующих с ним роботов.
b1. Если список конфликтов пуст,
то робот принимает статус МИР;
сообщает об этом всем другим агентам в произвольном порядке.
Иначе робот принимает статус КОНФЛИКТ;
сообщает об этом всем другим агентам в произвольном порядке;
делит список конфликтов на две части:
ОТЦЫ: те от кого можно получать приглашения переключиться[6]
ДЕТИ: те, кому можно послать приглашение переключиться[7].
b2. Если в общем списке агента все роботы имеют статус МИР,
то агент сообщает всем агентам, что можно двигаться к укрытию;
GOTO 6.
b3. Если агент получает сообщение, что можно двигаться к укрытию,
то GOTO 6.

После исполнения этого этапа факты агента дополняются знанием статуса конфликтности всех остальных агентов на данном раунде, а также, возможно, списком конфликтующих с ним роботов. Заметим, что если статус кон-

[6] бóльшие роботы
[7] меньшие роботы

фликтности у всех агентов в его списке МИР, то робот знает, *что всем можно двигаться к укрытиям.*

4. a. Если список конфликтов пуст, то GOTO 2.
Если список ДЕТИ не пуст, посылает приглашение первому в списке.
b. 1. Если приглашение послано,
Пока не получит ответ, принимает приглашения.
2. Вычёркивает первого из списка ДЕТИ, если он есть.
3. Если ответ ДА,
то посылает ответ НЕТ всем приглашающим; GOTO 5.
4. Если ответ НЕТ и приглашения получены,
то первому в списке отправивших приглашение отвечает ДА;
остальным отвечает НЕТ; GOTO 5.
5. Если приглашений нет, **то** GOTO 3a.
c. 1. Счётчик ожидания $CO := |PARENTS|$.
2. Если список ДЕТИ пуст,
то агент ждёт приглашения пока оно не поступит или $(n-2)^2$ тактов.
3. Если приглашение поступило,
то отвечает ДА; GOTO 5.
4. Если приглашение не поступило и $CO > 0$,
то $CO := CO - 1$; GOTO 4c2.
5. Если $CO = 0$, **то** GOTO 2.

После исполнения этого этапа факты агента никак не меняются, но в процессе исполнения очевидным образом меняется статус переговоров с конфликтующими роботами.

5. Агент и приглашённый робот обмениваются укрытиями; GOTO 2.

После исполнения этого этапа факты агента изменяются в части выбора укрытия им самим.

6. Агент двигается к выбранному в последний раз укрытию.

На каждом раунде вышеописанного алгоритма, за исключением последнего, все роботы с непустым списком конфликтов знают, что двигаться к укрытиям ещё нельзя, роботы же у которых собственный список конфликтов пуст, могут получить ту же информацию на основании своих знаний о выбранных остальными укрытиях, либо из сообщений о статусе на этапе 3.

Оценим сложность раунда выбора партнёра для переключения. Первый пункт первоначального выбора укрытия и обмена сведениями о координатах исполняется только один раз и имеет временную сложность $O(n)$.

Второй пункт первичного согласования выбора укрытий, в котором роботы должны удостовериться, что нет конкуренции, занимает время $O(n^2)$, поскольку агент вообще говоря, может перебрать $n-1$ укрытие (причём о каждом выборе он должен проинформировать $n-1$ агента), прежде чем окажется, что никто больше на его укрытие не претендует.

Третий пункт, в котором агенты вычисляют и упорядочивают список конфликтов, требует $O(n)$ времени. Заметим, что если конфликтов нет ни

у кого (3.c), то это означает, что можно двигаться к укрытиям, не рискуя в кого-нибудь врезаться.

В четвёртом пункте агент пробует договориться о переключении с кем-нибудь из своего списка конфликтов: либо с “детьми” (4.b), либо с “отцами” (4.b.4, 4.c). Линейный порядок на роботах и разделение списка конфликтов введены, чтобы не возникало циклических предложений переключиться. Сложность по времени пункта 4 равна $O(n^3)$, поскольку время ожидания ответа на предложение может быть $O(n^2)$ и в п.4.c, возможно, придётся ждать приглашения $n-1$ раз.

В пунктах пять и шесть переключение и движение происходит за константное время.

Итак, таких раундов потребуется столько, сколько нужно осуществить переключений для решения задачи Дейкстры. Понятно, что задачу Дейкстры можно решить за время $O(n!)$, просто перебирая варианты соединений, по экспериментальным же данным [5,1] сложность обычно не выше $O(n^2)$. Хотя обнаружена последовательность переключений длины $O(n^3)$, неизвестно, может ли в общем случае алгоритм Дейкстры завершиться быстрее, чем за время $O(n!)$.

Пусть задачу Дейкстры можно решить за время $T(n)$. Тогда временная сложность анархического алгоритма будет $O(T(n) \times n^3)$.

3.2 Коллективный алгоритм

Сразу отметим, что, в отличие от анархического варианта, теперь не предполагается, что все роботы изначально линейно упорядочены и что им известен этот порядок. Идея решения задачи с “выбором” лидера состоит в следующем: все агенты-роботы пересчитываются (т.е. встраиваются в цепочку) и последний назначается лидером, после чего он каким-либо образом решает задачу Дейкстры и назначает всем агентам укрытия. Самоорганизация (пересчитывание) агентов происходит таким образом: агенты соединяются в отдельные цепочки (единственного агента тоже можно считать цепочкой) и постепенно все эти цепочки соединяются между собой в одну в результате переговоров последних агентов в цепочках. В первый момент времени все агенты равноправны, каждый из них может стать началом цепочки, но, поскольку действуют они параллельно и асинхронно, у самых быстрых агентов шанс стать началом выше. Цепочки объединяются в одну присоединением конца одной цепочки к началу другой (но из-за конечной скорости передачи сообщений по цепочке надо следить, чтобы не образовалось циклов). Для этого введём такое понятие как уникальный *цвет цепочки*. В его качестве в начальный момент можно, например, взять координаты первого в цепочке робота. Агентам-роботам необязательно знать координаты всех роботов, достаточно только данных о предшественниках и непосредственном последователе. Данные о предшественниках нужны, чтобы последний в получившемся списке (т.е. лидер) знал координаты всех, чтобы решить, кому и в какое укрытие идти.

Разумеется, можно было бы осуществить выбор лидера каким-либо другим способом. Например, можно было бы всем обменяться координатами и назначить главным самого первого в лексикографическом порядке на координатах, но тогда каждому роботу пришлось бы взаимодействовать с каждым, что, например, в условиях плохой связи может быть очень затратно по времени. В этом отношении предлагаемый способ самоорганизации цепочками лучше, поскольку предполагает меньшее число контактов между агентами. В принципе можно было бы рассмотреть выборы лидера голосованием, но это увело бы нас в сторону от темы исследования.

В нашем коллективном алгоритме за один такт можно принять или послать только одно сообщение. Далее цвет – это цвет цепочки. Заметим, что принадлежащие какой-либо цепочке агенты имеют цвет.

Как и для предыдущего протокола перечислим явно BDI-особенности для роботов: список фактов, возможно, неверных, которые могут быть в базе знаний агентов, локальные намерения агентов, их цели и возможные действия.

Факты: собственные координаты; координаты всех укрытий[8]; цвет цепочки; свой номер; координаты и номера роботов-предшественников; координаты и номер непосредственного робота-последователя; цвет подключаемой цепочки (кратковременно); список предложений.

Намерения: определить лидера, подключиться к цепочке, удлинить цепочку.

Цели: отсутствие конфликтов; занять укрытие.

Действия :

послать-принять сообщение: предложение подключиться, ответ на предложение (ДА, НЕТ), цвет цепочки, список предшественников с координатами, номер в цепочке, вопрос о цвете и координатах, назначение номера укрытия;
изменить: цвет, номер, список предшественников, данные последователя;
двигаться к укрытию;
решить задачу Дейкстры (только для лидера.

Обозначим агента в начале цепочки как $C.1$, агента в середине цепочки как $C.x$ $(1 < x < e)$, агента в конце цепочки как $C.e$.

Коллективный Алгоритм

1. Агент не имеет цвета
  a.1. Посылает (“в пространство”) предложение (тому кто “подберет”) присоединиться к цепочке; ждёт ответа 1 такт.
  2. Если получает ответ ДА от бесцветного агента,
  то запоминает координаты ответившего;
    передаёт свои координаты, которые становятся цветом;

[8] однако использует их только лидер

если в сообщениях есть предложения подключиться,
то отвечает всем НЕТ;
GOTO 4.

3. Если получает ответ ДА от цветного агента из середины цепочки,
то получает от него координаты и цвет С1 начала цепочки С1.1;
принимает цвет С1, становится первым номером в цепочке С1,
передаёт свои координаты (как список предшественников),
новый номер С1.1;
если в сообщениях есть предложения подключиться,
то отвечает всем НЕТ;
GOTO 4.

b. Если
(1) в полученных сообщениях есть предложения подключиться
или ответы на ранее посланные предложения подключиться, **Или**
(2) на запрос 1.a получает ДА от цветного агента конца цепочки,
то если (1), **то** отвечает первому ДА, остальным – НЕТ;
агенту присваивается номер;
он получает координаты предшественников и цвет;
GOTO 2.

c. Иначе GOTO 1.a.

После исполнения этого этапа агент становится в цепочке либо первым, либо последним, при этом в фактах агента изменяется цвет, номер, список предшественников (если номер не 1), последователь (если номер 1).

2. Агент $C1.e$ (последний в цепочке С1)

a. Если номер агента равен n,
то $C1.e$ – лидер; GOTO 5.

b.1. Посылает цвет и предложение подключиться агентам не из списка;
ждёт ответа 1 такт.

2. Если получает ответ ДА от бесцветного агента,
то запоминает координаты ответившего;
передаёт список предшественников и номер $e+1$;
если в сообщениях есть предложения подключиться,
то отвечает всем НЕТ;
GOTO 3.

3. Если получает ответ ДА от агента $C2.x$,
то получает от него координаты и цвет начала цепочки $C2$.
Если цвет начала равен $C1$, **то** GOTO 2b.
Если цвет начала равен $C3$,
то изменить свой цвет на $C3$;
послать сообщение "НОВЫЙ ЦВЕТ $C3$" агентам,
записанным как $C1.1$, $C1.e-1$ и $C1.\frac{1+e}{2}$;
послать агенту $C2.1$ новый номер $C1.(e+1)$ и
список предшественников;
GOTO 3.

c. Если в полученных сообщениях есть предложения подключиться,

то отвечает ДА первому агенту $C2.e$, не совпадающему по цвету,
остальным – НЕТ;
если агент бесцветный,
то запоминает координаты ответившего;
передаёт список предшественников и номер $e+1$;
иначе узнаёт у агента, записанного как $C1.1$,
цвет и координаты начала цепочки;
передаёт $C2.e$ цвет и координаты начала цепочки;
GOTO 3.

d. Если получает сообщение "НОВЫЙ НОМЕР y", предшественники P,
то заменить свой номер на y и список на P;

e. GOTO 2.a.

На первом шаге этого этапа агент может обнаружить, что он последний в цепочке, а значит является "лидером". После исполнения этого этапа агент становится в цепочке либо серединным и тогда в фактах меняется его цвет и последователь, либо предпоследним, при этом в фактах агента изменяется только последователь. Агент может также получить новый номер и список предшественников.

3. Агент $C.x$ (посередине цепочки)

a. Ждет какого-либо сообщения.

b. Если получает сообщение "НОВЫЙ ЦВЕТ C",
то поменять свой цвет на C;
послать сообщение "НОВЫЙ ЦВЕТ C" агентам,
записанным как $C.(x-1)$ и $C.(\frac{x}{2})$.

c. Если получает сообщение "НОВЫЙ НОМЕР y", предшественники P,
то заменить свой номер на y и список на P;
послать сообщение "НОВЫЙ НОМЕР $y+1$" и список предшеств.,
пополненный им самим, агенту $C.(x+1)$.

d. Если получает сообщение "КАКОЙ ЦВЕТ И КООРДИНАТЫ?",
то пересылает это сообщение и координаты отправителя агенту $C.1$.

e. Если получает предложение от агента $C2.e$ или бесцветного,
то если $C2.e$ не цвета $C.x$ или бесцветный,
то отвечает ДА;
узнаёт у агента, записанного как $C1.1$,
цвет и координаты начала цепочки;
передаёт $C2.e$ цвет и координаты начала цепочки.
иначе отвечает НЕТ.

f. Если получает сообщение "Укрытие x",
то GOTO 6.

g. GOTO 3.a.

В процессе исполнения этой части протокола в фактах агента может меняться цвет, номер и список предшественников. Также агент может получить назначение укрытия.

4. Агент $C.1$ (первый в цепочке)
- **a.** Ждет какого-либо сообщения.
- **b. Если** получает сообщение "НОВЫЙ ЦВЕТ C",
  то поменять свой цвет на C.
- **c. Если** получает сообщение "НОВЫЙ НОМЕР y", предшественники P,
  то заменить свой номер на y и список на P;
  послать сообщение "НОВЫЙ НОМЕР $y+1$" и список предшеств., пополненный им самим, агенту $C.2$;
  GOTO 3.
- **d. Если** получает сообщение "КАКОЙ ЦВЕТ И КООРДИНАТЫ?" и адрес отправителя,
  то пересылает свой цвет и координаты отправителю.
- **e. Если** получает предложение от агента $C2.e$ или бесцветного,
  то если $C2.e$ не цвета $C.1$ или бесцветный,
  то отвечает ДА, остальным отвечает – НЕТ;
  передаёт $C2.e$ свой цвет и координаты.
  иначе отвечает НЕТ.
- **f. Если** получает сообщение "Укрытие x",
  то GOTO 6.
- **g.** GOTO 4.a.

В процессе исполнения этой части протокола в фактах агента может меняться цвет, номер и список предшественников. Также агент может получить назначение укрытия.

5. Назначить укрытия роботам и сообщить им назначения; GOTO 6.
6. Двигаться к назначенному укрытию.

На каждом этапе алгоритма, начиная со второго, роботы имеют в базе знаний цвет, номер и список предшественников с их номерами и координатами. Однако, поскольку цепочки постоянно удлиняются и меняют цвет, эти данные часто не соответствуют действительности, особенно у агентов в середине цепочки. Но нам важно, чтобы концевые агенты имели точные знания о цвете цепочки, чтобы в процессе построения общего упорядоченного списка агентов не возникало цикла, и это требование выполняется. После того как лидер сообщил агенту координаты укрытия, к которому он должен двигаться, считается, что агент знает, что конфликтов с другими агентами у него на пути к укрытию не случится.

Сложность алгоритма зависит от скорости слияния цепочек и от того, как быстро удаётся договориться о слиянии (точнее, дождаться ответа на предложение подключиться). Само слияние, очевидно, осуществляется за время линейное относительно размера цепочки и состоит из (1) присоединения конца одной цепочки к началу другой (сложность $O(n)$, поскольку агент, которого последний в цепочке считает началом, может первым уже и

не быть, но тогда он знает адрес нового начала, передаёт информацию ему и т.д.); (2) окрашивания нового начала цепочки в цвет конца (сложность $O(\lg n)$ за счет дихотомии на шаге 3.b); (3) изменение порядковых номеров и списка предшественников (сложность $O(n)$). Выкрасить конец в цвет начала, что кажется, на первый взгляд, более естественным, на самом деле более затратно, поскольку агенты не знают координат всех последователей, а знают только предшественников и ближайшего соседа. В таком случае, они вынуждены линейно передавать цвет по цепочке дальше, причём пока они передают цвет, меняют порядковые номера, цвет может уже снова измениться, получается, что будет проделано много лишней работы. В случае же окрашивания начала, цвет передаётся от одного агента сразу двум агентам цепочки, они в свою очередь передают его ещё двум и так далее и цепочка очень быстро окрашивается. Кроме того, цвет на этом участке будет меняться только один раз, в отличие от нумерации агентов и списка их предшественников. Отметим, что поскольку окрашивание происходит значительно быстрее, чем изменение номеров предшественников, и с конца участка цепочки, то ситуация, когда цвет агента окажется не изменён, невозможна.

Пусть в данный момент времени есть m цепочек агентов. Скорость слияния цепочек в лучшем случае, когда каждые $O(n)$ шагов сливаются $\frac{m}{2}$ цепочек, логарифмическая, и в худшем случае – линейная, когда каждые $O(n)$ шагов сливаются только две цепочки. Время ожидания ответа для бесцветного робота зависит от того как скоро он обнаружит кого-то готового подключить(-ся) или обнаружат его.

Таким образом, наихудшая временная сложность алгоритма оказывается равной $O(n^2)$. Алгоритм завершается, если только на самом первом шаге все агенты одновременно циклически не пошлют заявку образовать цепочку и быть первым в ней.

4 Заключение

В данной работе проанализированы два подхода к решению частной задачи мультиагентного распределения ресурсов, а именно индивидуальные протоколы агентов, решающих задачу о роботах на Марсе. Первый протокол – “анархический”, в котором агенты самостоятельно попарно разрешают конфликты, возникающие между ними, тогда как во втором, “коллективном”, протоколе агенты выбирают лидера, который решает все конфликты за них. Первый протокол проще для описания и понимания, но второй оказывается более эффективным относительно времени исполнения и памяти, необходимой агентам в процессе взаимодействия.

Первый протокол интересен также тем, что показывает важность исследования задачи Дейкстры для протоколов попарного взаимодействия агентов. К сожалению, точную временную сложность алгоритма Дейкстры пока вычислить не удалось, но это будет сделано в ближайшем будущем. Также предполагается экспериментально исследовать предложенные протоколы,

используя подходящую платформу спецификации мультиагентных систем. Кроме этого протоколы будут модифицированы с учётом BDI-особенностей агентов, и этот вариант мы будем исследовать экспериментально.

Благодарности. Хочу выразить особенную благодарность моим коллегам доктору Николаю Вячеславовичу Шилову, а также Евгению Бодину за помощь в исследованиях и анализе протоколов задачи о роботах. Так же благодарю рецензента доктора Наталью Алёхину за ценные замечания.

Список литературы

1. Andreeva T.V., Bodin E.V., Gorodnya L.V., and Shilov N.V. *Etude on Theme of Dijkstra: Computer Scientists at home of Geometers.* Potential (in Russian), n.9, 2006.
2. Burkard, Rainer; M. Dell'Amico, S. Martello. *Assignment Problems.* SIAM, 2009
3. Jack Edmonds and Richard M. Karp. *Theoretical improvements in algorithmic efficiency for network flow problems.* Journal of the ACM 19 (2): 248–264.
4. Fagin R., Halpern J.Y., Moses Y., Vardi M.Y. *Reasoning about Knowledge.* MIT Press, 1995.
5. Shilov N.V., Shilova S.O. *Etude on theme of Dijkstra.* ACM SIGACT News 35(3), 2004.

OLAP-моделирование в задаче автоматизированной поддержки муниципального заказа
OLAP-Modelling of Municipal Procurement Automation Support Problem

Анна Коробко, Татьяна Пенькова
Anna Korobko, Tatyana Penkova

[1] Институт Вычислительного Моделирования СО РАН, Академгородок, 50, стр.44, 660036, Красноярск, Россия
[1] Institute of Computational Modeling of Siberian Branch of the Russian Academy of Sciences, Akademgorodok, 50/44, 660036, Krasnoyarsk, Russia
{Lynx, Penkova_t}@icm.krasn.ru

Аннотация. В работе предложен подход к построению OLAP-моделей на основе формального анализа терминов предметной области и особенностей решаемых задач. Представлены основные этапы моделирования: анализ запросов, определение множества терминов, генерация объектов анализа и сопоставление с базой данных. Рассмотрена реализация предложенного подхода для задач мониторинга и генерации документов в муниципальном управлении.

Ключевые слова: витрина данных, генерация документов, муниципальный заказ, OLAP, OLAP-моделирование.

Технология OLAP

OLAP-технология (On-line analytical processing) представляет собой современную концепцию анализа данных, описанную совокупностью требований к программным продуктам, обеспечивающим оперативную аналитическую обработку и представление данных. Использование OLAP-средств позволяет превратить работу с данными в быстрый, наглядный, эффективный процесс, расширить возможности доступа к информации и визуализации результатов анализа. Впервые принципы OLAP были сформулированы основоположником теории реляционных баз данных Е.Ф. Коддом. В своей статье в 1993 году «Обеспечение OLAP для пользователей-аналитиков» Е.Ф. Кодд [1] описал ряд требований, которые позволяют усовершенствовать работу с данными, путем представления и обработки их в многомерном виде. Одним из основных требований технологии OLAP является «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. Такое представление данных возможно лишь при тщательном предварительном анализе структуры исходных данных и терминологии исследуемого информационного поля.

Формирование OLAP-модели

В работе предложен подход к построению витрины данных для OLAP-моделей на основе формального анализа терминов предметной области и особенностей решаемых задач.

Витрина данных представляет собой схему подмножества данных, необходимых для решения поставленной задачи, семантическую надстройку, ставящую в соответствие наименованиям полей из базы данных значимые термины из предметной области, фильтры и сценарии расчета аналитических показателей. Отдельная витрина данных содержит уникальный набор входных параметров, необходимых для построения многомерного аналитического куба (OLAP-модели), моделирующего заданную часть предметной области. Основу витрины данных составляет множество специальных терминов, необходимых для решения поставленной задачи. Кроме того, формирование множества терминов позволяет очертить границы формируемой OLAP-модели.

На первом шаге проектирования витрины данных, путем интервьюирования конечного пользователя и изучения отчетных форм, предлагается определить набор данных, которые должны быть включены в разрабатываемый аналитический куб для решения поставленной задачи.

Анализ сформулированных запросов позволяет определить используемое в моделируемой области множество терминов, которые в дальнейшем преобразуются в объекты анализа. Следует заметить, что множество терминов неоднородно. Одни термины носят сущностный характер и, в соответствии с теорией технологии OLAP, образуют измерения проектируемого куба, а другие – являются атрибутивными и должны быть отнесены к множеству показателей. Группировка терминов может быть изменена в соответствии с поставленной задачей и концепцией проектирования. Измерения и показатели витрины данных, полученные в терминах предметной области, обеспечивают «прозрачность» проектируемого OLAP-куба для конечного пользователя.

На следующем шаге процесса формирования витрины данных, в соответствии с предлагаемым подходом, необходимо соотнести полученные объекты анализа с существующими полями таблиц базы данных, определяя тем самым физическую составляющую измерений и показателей. Объекты анализа могут быть связаны с полями таблиц напрямую или рассчитываться на основе нескольких полей по заданному алгоритму расчета с помощью OLAP-инструментария [2].

Схема базы данных автоматизируемой области может иметь довольно сложную структуру. Преимущество использования витрин данных заключается в том, что для решения поставленной задачи нет необходимости использовать всю базу данных. В соответствии с выделенными объектами анализа, в витрину

входят только те таблицы, поля которых участвуют в формировании OLAP-модели, представляя собой подмножество общей схемы. Для наполнения многомерного куба корректными данными важно сохранить между таблицами связи, присутствующие в основной схеме. Специализация проектируемого многомерного куба, возможно, потребует дополнительных ограничений и связей.

Применение OLAP-моделирования в задаче размещения муниципального заказа

Предложенный подход формирования OLAP-моделей реализован в автоматизированной системе поддержки размещения муниципального заказа АСП МЗ [3]. Функциональные возможности системы позволяют осуществлять аналитический мониторинг процессов подготовки и размещения заказа и обеспечить оперативное формирование организационно-распорядительных и отчетных документов. В систему включены инструменты построения витрин данных, формирования многомерных аналитических кубов, отображения многомерных данных, представляющих OLAP-модель автоматизируемой области.

Для мониторинга процесса проведения торгов реализованы такие витрины данных, как «Журнал закупок», «Журнал заявок», «Реестр муниципальных контрактов», «Динамика возврата обеспечения» и другие, которые позволяют оперативно отслеживать статистические и аналитические показатели в разрезе источников финансирования, главных распорядителей бюджетных средств, муниципальных заказчиков, способов размещения заказа, и т.д. Выполнение срезов и вращение построенного аналитического куба дают пользователю полное представление о фактически реализованных и планируемых муниципальных заказах и позволяют оценить эффективность освоения бюджетных средств.

Неотъемлемой частью размещения заказа является оперативное формирование различного рода документации о процессах, итогах размещения заказа и принятых решениях. Формируется огромное количество документации, предназначенной как для внутренней управленческой деятельности, так и для взаимодействия со сторонними организациями. Разнообразие, сложность форм документов и необходимость изменять их структуру требуют разработки специализированных шаблонов, обеспечивающих оперативное формирование и модификацию документов с возможностью гибкой настройки параметров заполнения [4].

На этапе разработки шаблона пользователем создается визуальный макет с помощью текстового редактора. При разработке шаблона выполняется структурирование документа на основе применения специализированной разметки. С помощью разметки определяются роли фрагментов документа в его общей структуре, их содержание и визуальное отображение.

На этапе настройки шаблона осуществляется сопоставление элементов разметки шаблона с полем таблиц базы данных. Для поддержки сложной логики формирования документов помимо исходных таблиц данных используются расчетные таблицы, являющиеся дополнительными выборками из справочников

и реестров. Необходимость построения дополнительных запросов вызвана потребностью в новом сочетании существующих данных. Например, для отдельных документов может потребоваться выборка сведений с определенной фильтрацией, создание расчетных значений или потребуется увязать между собой данные, не имеющие прямых ссылок друг на друга.

Управление выборкой данных (построение, редактирование, выполнение пользовательских запросов) выполняется с помощью средств оперативного построения запросов, которое реализуется с помощью OLAP-моделей. Формирование витрины данных выполняется с помощью визуального конструктора. В область построения витрины данных из источника данных выбираются таблицы, на основе которых необходимо построить запрос, и между ними устанавливаются связи. Затем из каждой таблицы выбираются нужные поля и задаются условия. После выполнения аналитических операций автоматически формируется SQL-запрос, результат выполнения которого представляется пользователю в виде таблицы, сохраняется в базу и может использоваться при настройке шаблона или как источник данных для других моделей.

В процессе генерации документа в диапазоны шаблона, заданные с помощью специализированной разметки, помещается результат выполнения связанных с ними запросов.

Применение OLAP-моделирования позволяет формировать документы сложной структуры на основе результатов агрегации многомерных данных, представленных в терминах предметной области.

Предложенный в работе подход к построению OLAP-моделей позволяет говорить о возможности развития теории проектирования аналитических онтологий для задач OLAP-анализа. Перспективным направлением для исследования может стать поиск способов классификации терминов предметной области и введение между ними отношений атрибутивности и подчиненности, что продиктовано существующими связями между реальными объектами и спецификой технологии аналитической обработки данных.

Список литературы

1. Codd, E.F., Codd, S. B., Salley, C.T.: Providing OLAP to User-Analysts: An IT Mandate, Arbor Software Corp. Papers (1996)
2. Горохова, А.В, Ишенин П.П., Никитина М.И.: OLAP-средства системы «Аналитик», Труды Всероссийской конференции «Информационно-аналитические системы и технологии в здравоохранении и ОМС», Красноярск: КМИАЦ, С. 220-228, (2002)
3. Щербенин, В.Ф., Лузан, Н.Ф., Ноженкова, Л.Ф., Жучков, Д.В., Исаева, О.С.: Комплексная автоматизированная поддержка подготовки, размещения и контроля муниципальных заказов /В.Ф. Щербенин, Материалы десятой Всероссийской научно-практической конференции ПИР-2007, Т.1., Красноярск, С. 3-11, (2007)
4. Пенькова, Т.Г.: Модели и методы оперативного формирования документов, Вычислительные технологии, Т.14, №2, С.98-109, (2009)

Специфика подходов к отображению онтологий
Specificity of Ontology Mapping Approaches

Н. А. Скворцов
Nikolay Skvortsov

Институт проблем информатики РАН
Institute for Informatics Problems RAS
nskv@ipi.ac.ru

Аннотация. Подходы, используемые средствами поддержки отображения онтологий, во многом основываются на методах, разработанных для отображения схем. Однако этого недостаточно, так как онтологии определяют понятия, для которых более важна понятийная семантика, нежели описание структуры. Статья рассматривает подходы, которые полезно использовать при отображении онтологий для обнаружения семантических сходств и различий понятий. Помимо упоминаемых характерных для схем оценок близости по вербальным и структурным спецификациям, раскрываются подходы, основанные на формальной проверке уточнения спецификаций, применении метаонтологий, метасвойств фундаментальных видов, проверке экстенсионалов понятий. Такие подходы позволяют обосновывать корректность отображения понятий друг в друга на семантическом уровне.

Approaches used by tools supporting ontology mapping are mostly based on schema mapping methods. But ontology has its own peculiarity. Schemas simulate entities of a domain, their conceptual structure, relationships and behavior and used for system development and implementation. Schemas often include properties of several entities or auxiliary information in a single abstract type. Ontology specifies a conceptualization, its foreground is concepts of a subject domain. Ontological information is structured not so arbitrarily. Structural specification of an ontology is not intended for storing and performing data, but reflects a place of concepts in a concept system, out of which a separate concept cannot exist.

Schema mapping tools usually include abstract type similarity evaluation by verbal and structural specifications. That is not sufficient for ontological concept semantics. The paper is dedicated to approaches that formally discover concept similarities on semantic level. Some of them specify additional concept semantics and don't depend on structural specifications.
Supposing that well specified ontologies precisely reflect concept semantics, we apply a formal criterion to be sure that semantics is saved during mapping. This criterion is specification refinement relation taken from programming theory. It means that refining specification can be transparently used in place of refined one. This relation is formally defined for abstract data types and may be proved. Subsumption relation is a case of specification refinement defined over concept extents. Today's ontological models support automatic inference of subsumption.

In addition to verbal and structural specifications of concept semantics, it's useful for mapping to have specifications considering ontologies from a common point of view.

One of such approaches is using common underlying metaontology. Every concept of ontologies (as well as every relation or property if possible) becomes an instance of metaontology class or of a class expression in terms of metaontology. So ontological concepts are sorted into classes by their semantics in terms of the metaontology. Formally, in the mapping task, subsuming (refining) concept must belong to a subclass or to the same class of the metaontology with subsumed (refined) concept. Self specifications of concepts and specifications in terms of metaontology are independent since they use different levels of classification hierarchy. Metalevel approaches take their birth in conceptual modeling and become more actual in ontological modeling.

N. Guarino's ontology gives much attention to various kinds of concept properties. There is a set of metaproperties, any concept may be evaluated by. They are essence, rigidity, identity, dependency, unity, properties of mereologyically related concepts and others. These metaproperties are formally defined, so metaproperty conflicts between mapped concepts mean incorrectness of mapping.

Metaproperty approach is partly related to the approach using top level ontology as a common top hierarchy for ontologies being mapped. Particular values of metaproperties correspond to fundamental concepts of the top level ontology.

One more approach adding semantic methods to ontology mapping is using instances of concepts as classes. It uses objects of real world, sample models, or data well classified using ontologies. Formally, in mapping task, conflict of a single instance in classes defined by related concepts means incorrectness of established relation. Such extensional approach "by example" is useful for search of relevant concepts or for verification of mapping.

Efforts of working groups on ontology development often take years. So conflicts of ontologies are conflicts of deeply thought out decisions. Ontology mapping needs tools supporting intelligent work of experts.

All the represented approaches may be implemented to be formal. But they are only helping tools for manual work of experts on ontology mapping. There are assumptions in any formal method: sufficiency of concept semantics of specifications, correctness of concept specification in terms of a metaontology and metaproperties, correctness of relating of real world object samples to ontological concepts and so on.

A protocol of experts' work on ontology mapping requires participation of experts representing every ontology (as well as experts of metaontology and fundamental metaproperties) because ontologies have been often developed by different workgroups. Experts may make decisions on mentioned assumption in fields of their competences.

Some semantic conflicts may be discover in process of expert discussions as a result of application same additional metainformation or same instances of concepts to every ontology. Every represented approach helps experts to effectively discover hidden conflicts of ontologies. They may be implemented as a multiexpert interactive tool supporting ontology mapping or as a verbal discussion protocol for experts.

Введение

Проблема отображения неоднородных онтологий является актуальной с самого начала использования онтологий при создании информационных систем. Анализ состояния исследований [8] соответствующих методов показал, что эта тема исследована до сих пор недостаточно глубоко. Разрабатываемые методы, в основном, неформальны и имеют множество открытых вопросов. Принципы и методы отображения онтологий остаются предметом дискуссий, при создании систем вопросы отображения неоднородных онтологий до сих пор предпочитают избегать. Наименее исследованы методы отображения онтологий, разработанных в неоднородных онтологических моделях.

Говоря о неоднородных онтологиях, мы подразумеваем, что две (или более) онтологии по-разному описывают одну и ту же предметную область или близкие предметные области с точки зрения разных сообществ. Онтология задаёт подразумеваемую семантику для понятий предметной области и определяет онтологический контекст, в котором работает сообщество. Разные контексты использования онтологий, созданных разными сообществами, отражаются на особенностях подходов к спецификации понятий, что становится одной из причин неоднородности. В результате, семантика понятий в контекстах, описанных разными онтологиями, может быть сходной при различных подходах к описанию их структуры: составу, ограничениям и степени детализации.

Под отображением онтологий мы понимаем процесс, при котором понятия одной онтологии выражаются через понятия другой. Часто термин отображение корректно рассматривается в двух ракурсах: как процесс отображения одной онтологии в другую или как результат [3] такого процесса, то есть множество функций отображения понятий одной онтологии в понятия другой. Отображение онтологий является неотъемлемой частью большинства задач согласования [3] онтологий, таких как слияние, выравнивание онтологий, модификация одной онтологии для достижения однородности с другой и так далее.

Модели данных, используемые сегодня в качестве онтологических, либо неформальны, либо включают достаточно простые средства спецификации для возможности использования автоматического формального вывода, довольствуясь описаниями структурных спецификаций понятий и простых ограничений над ними. Поэтому большинство методов, используемых для отображения онтологий, предварительно связывают понятия по вербальной информации (именам понятий, вербальным определениям), и затем на основе полученных связей оперируют со структурными спецификациями, оценивая их близость, обнаруживая и устраняя разного рода конфликты.

Результатом этого стало то, что для отображения онтологий применяются методы, уже наработанные в области отображения схем. Недавний обзор [18] методов отображения онтологий выявил именно такую тенденцию. Однако очевидно, что онтологическая информация специфична и не ограничивается спецификацией схемы. В данном исследовании представлены подходы к отображению онтологий, которые позволяют учитывать специфику онтологической информации.

1 Работы по теме

Сходства и различия онтологий и схем баз данных подробно обсуждаются в [15]. Схемы моделируют сущности предметной области, их концептуальную структуру, характерные соотношения между ними и поведение в предметной области и используются для проектирования и реализации систем. Онтологии отражают концептуализацию предметной области, в них на первый план выходит роль понятий предметной области.

Во введении в методы сопоставления схем и онтологий [11] также приводятся некоторые отличия между схемами и онтологиями. Во-первых, утверждается, что схемы, в отличие от онтологий, не описывают явно семантику данных. Это сомнительно, схемы специфицируют ограничения целостности данных, тем самым выражая их семантику. Более того, онтология, в общем случае, вообще не связана с данными, и такая неточность в утверждении только порождает путаницу между схемами и онтологиями. Во-вторых, сказано, что в отличие от схем, онтологические определения основаны на множестве логических аксиом, а о схемах упомянуто, что они могут не поддерживать отношение обобщения. Но это различия не онтологий и схем, а скорее, традиционно используемых для их представления моделей данных (реляционных, объектных, логических). В частности, типы в схемах выражаются с помощью логических аксиом. На этом сравнение заканчивается, и различия схем и онтологий не упоминаются в описаниях подходов и известных систем отображении и слияния схем или онтологий.

Можно привести немалый список программных систем, решающих задачу отображения, созданных изначально для работы со схемами, но применяемых равно и к онтологиям (Cupid [9], SMatch,[5]); и обратно, позиционирующих себя как созданные для отображения онтологий, но использующих те же методы, что применяются для отображения схем (OLA [2], PROMPT [10]). Состав подходов в них имеет схожие тенденции, которые раскрываются в разделе 3.

В программе прошлогоднего семинара «Онтологии и знания *ELSEWHERE*» была представлена статья о методах отображения и интеграции онтологий [16]. В ней описываются методы, принятые в большинстве систем, решающих подобные задачи над схемами и онтологиями. Оценочные методы используют вербальные и структурные спецификации для связывания классов онтологий друг с другом, а далее на основе полученных связей производятся манипуляции с классами, свойствами и отношениями для слияния онтологий. В сравнении с [16], настоящее исследование упоминает лексический и структурный подходы как неизбежно необходимые, но упор делает на другое. Основное отличие подходов, представленных в следующих разделах, заключается в предложении и обосновании методов, которые дополнительно специфицируют и проверяют семантику понятий вне зависимости от структурных спецификаций, что особенно важно для отображения онтологий.

Необходимо отметить также особенности терминологии, используемой в [16]. Со ссылкой на глоссарий деятельностей NeOn [21] в статье вводятся термины отображение и интеграция онтологий. В терминологии деятельностей над онтологиями, разработанной в проекте NeOn, которой часто стали придерживаться исследователи, есть немало неточностей. Если говорить о её части, кото-

рая относится к существу вопросов, затронутых в статье, то выравнивание (alignment) и отображение (mapping) онтологий в NeOn преподносятся как синонимы. Однако отображение всегда было направленным процессом, то есть отображением одной онтологии в другую. Поэтому в данном случае, вероятно, в статье [16] при упоминании отображения повторяется ошибка составителей глоссария деятельностей NeOn. Далее, в NeOn под слиянием (merging) онтологий понимается создание новой онтологии, содержащей понятия обеих пересекающихся онтологий, а под интеграцией (integration) онтологий – включение одной онтологии в другую, обычно в качестве необходимой части. В [16] приводится английский термин merging со ссылкой на NeOn, но по-русски этот процесс называется интеграцией на протяжении всей статьи.

2 Отображение схем

Подходы, используемые для отображения схем информационных систем или баз данных, в основном, неформальны, рекомендательны и требуют прямых действий эксперта. Большинство систем интеграции схем (и онтологий) включают связку из двух подходов: лексического и структурного.

Лексические подходы обрабатывают вербальные определения и имена элементов схем, выделяют основы слов, разбирают словосочетания, исключают стоп-слова, учитывают расстояния между словами, синонимы из тезаурусов и так далее, с целью оценить степень связи элементов схем лексически. Подобные подходы в разном составе присутствуют практически во всех без исключения проектах, связанных с отображением схем.

Структурные подходы к отображению схем обычно располагают инициирующим набором связей понятий, полученных лексическими методами. Для обнаружения структурных соответствий или структурных конфликтов схем может использоваться широкий спектр различных подходов, таких как:

- эвристические формулы и правила (PROMPT [10]);
- анализ графов, включая правила на графах, подходы к их сравнению и алгоритмы на взвешенных графах (SODA [14]);
- обучающиеся подходы, основанные на статистике (RiMOM [13]);
- формальный вывод и доказательство (S-Match [S-Match]).

Лексические и структурные методы отображения схем равно применимы и к онтологиям, так как онтологическая спецификация также является схемой как формализация модели концептуализации. Однако у онтологической информации есть и своя специфика, которая влияет на выбор методов отображения онтологий.

3 Специфика онтологической информации

Схемы баз данных и информационных систем специфицируются абстрактными типами данных и классами объектов (или приводимыми к такому представлению спецификациями). При этом на практике типы часто специфицируются таким образом, что в структуру одного типа могут попадать свойства сразу нескольких сущностей реального мира или вспомогательная информация. Объекты этих типов агрегируют произвольные данные в виде, удобном для решения задач системы. Другие критерии при составлении схем обычно играют меньшую роль.

В результате разного назначения схем и онтологий, онтологическая информация структурируется не столь произвольно. Структуры, специфицированные в онтологии, отражают не представление данных, которые необходимо хранить и обрабатывать, а собственные свойства понятий и связи их между собой. Если в схеме более важны ограничения данных о сущностях предметной области, то спецификация онтологии призвана к образованию системы понятий, вне которой каждое отдельно взятое понятие не может существовать. Другими словами, структура понятия как типа становится спецификацией понятийной семантики, определяющей его место в мире.

Неоднородность структуры онтологий может быть существенной в зависимости от концептуализации предметной области, определяемой задачами, которые необходимо решать в этой области, и предполагаемыми методами их решения. Понятия с разной структурой могут соответствовать совпадающим экстенсионалам сущностей реального мира. Методов отображения структурных спецификаций при этом может оказаться мало для отображения онтологий.

В онтологиях, как и в схемах, важным элементом спецификаций являются вербальные определения понятий и отношений. Они требуют продуманности и точности. До сих пор семантика понятий наиболее точно может быть выражена для экспертов вербально.

Исходя из вышесказанного, можно сделать выводы о методологии отображения онтологий:

1) методы связывания онтологий по вербальным определениям понятий столь же важны в задаче отображения онтологий, как и задаче отображения схем;
2) методы структурной идентификации релевантных понятий и разрешения структурных конфликтов применимы также и для онтологий;
3) однако методы отображения онтологий по структуре понятий могут быть недостаточны, и необходимы механизмы, позволяющие находить сходства и различия в понятийной семантике, несмотря на сходства и различия в спецификациях их структуры.

По большому счёту, то же верно и для отображения схем, но при работе с понятиями предметной области пристальная проверка сходств и различий их подразумеваемой семантики становится особо актуальной.

4 Подходы к отображению онтологий

Выбор предпочтительных подходов к отображению онтологий должен производиться в первую очередь не из соображений эффективного обнаружения совпадений в именах или структурных описаниях понятий, хотя это также необходимо. Основной критерий отображения понятий онтологии – близость и непротиворечивость подразумеваемой понятийной семантики.

Во-первых, если предположить, что хорошо специфицированные онтологии точно отражают семантику понятий, то важны формальные методы, доказательно сохраняющие семантику при отображении понятий друг в друга. Ниже описан универсальный критерий отношения уточнения спецификаций, который можно использовать в основе доказательства корректности отображения понятий для произвольных онтологических моделей.

Во-вторых, необходимо иметь на вооружении методы, позволяющие находить сходства и различия в понятийной семантике, вне зависимости от сходства и различия в описаниях их структуры. Эти методы должны быть также формальными, чтобы с высокой долей уверенности обосновывать связи понятий. В качестве таких подходов ниже предложено использование метаонтологий и фундаментальных метасвойств, характеризующих понятия и отношения онтологий, а также анализ объектов, являющихся экземплярами понятий как классов.

4.1 Отношение уточнения спецификаций

Подход, представленный ниже, применим равно хорошо к отображению схем и онтологий. Это формальный критерий корректности отображения спецификаций, в том числе, спецификаций абстрактных типов данных и онтологических понятий, представляемых средствами абстрактных типов данных. Таким критерием является отношение уточнения спецификаций, пришедшее из теории программирования. Установленное между спецификациями, отношение уточнения означает, что уточняющую спецификацию гарантированно можно использовать вместо уточняемой, не замечая подмены. Данное отношение определяется для абстрактных типов данных формально, поэтому утверждение об уточнении спецификаций можно доказывать. В зависимости от сложности модели данных доказательство уточнения может быть автоматическим или интерактивным [17].

Частным случаем уточнения спецификаций является отношение поглощения, устанавливаемое на экстенсионалах понятий. Оно означает, что все экземпляры класса поглощаемого понятия являются также экземплярами поглощающего. Это отношение играет важнейшую роль в сегодняшних онтологиях, и возможность его автоматического доказательства является основных критерием при разработке современных онтологических моделей [7], призванных быть понимаемыми и человеком, и машиной. Поэтому учитывая тенденции и разрешимость онтологических моделей, в рассуждениях в большинстве случаев достаточно использовать поглощение. В частности, для отображения онтологий в современных онтологических моделях отношение поглощения понятий резонно использовать в качестве основного критерия.

В целом, отношение уточнения может устанавливаться между понятиями как при разработке одной онтологии, так и при согласовании разных онтологий. Формально обоснованные отношения уточнения понятий гарантируют корректность отображения понятий одной онтологии в другую. В этом случае, предполагая, что изначально спецификации онтологических понятий достаточно отражают их понятийную семантику, можно быть уверенным, что семантика понятий при отображении сохранена.

Данный подход работает со спецификациями онтологии как со схемами. Предположение о достаточности спецификаций для отражения семантики понятий здесь существенно. Поэтому помимо формальных подходов работы с онтологиями как со схемами, необходимы подходы, выявляющие сходства и конфликты понятий на основании информации о понятиях сверх описания их структуры и ограничений.

4.2 Метаонтологии и онтологии верхнего уровня

Помимо спецификаций вербальных и структурных для описания семантики онтологических понятий согласовываемых онтологий желательно иметь спецификации, рассматривающие понятия каждой из онтологий с некоторой общей точки зрения. Реализацией такого подхода может стать применение метаонтологии, связанной с обеими согласовываемыми онтологиями.

Метаонтология может содержать описание:

- обобщённой метамодели, на основе которой можно построить большинство онтологических моделей; такая метаонтология оказывается особенно полезна при отображении онтологий, разработанных в разных онтологических моделях [19];
- более абстрактной онтологии, из понятий которой строится большинство разновидностей сущностей, встречающихся в предметной области.

Метаонтология должна стать подложкой под согласовываемые онтологии. Если онтологии изначально не описаны одной и той же метаонтологией, соотносить элементы спецификаций с понятиями метаонтологии можно специально для решения задачи отображения. При необходимости качественного отображения онтологий данный подход может оказаться затратным, но дающим хороший результат. Принцип формирования подложки следующий.

Каждое понятие из согласовываемых онтологий (а также, если возможно, каждое отношение или свойство) должно стать экземпляром некоторого понятия метаонтологии. Если семантически подходящего понятия в метаонтологии нет, то создаётся служебное понятие (оно будет являться подпонятием понятия метаонтологии), являющееся выражением, описывающим в терминах понятий метаонтологии необходимую семантику. И элемент спецификации онтологии становится экземпляром служебного понятия. Таким образом, в классах, определяемых понятиями метаонтологии или служебными понятиями, в качестве экземпляров окажутся элементы спецификаций согласовываемых онтологий,

распределённые по классам в зависимости от их семантики с точки зрения метаонтологии.

Такой принцип построения подложки на метаонтологии позволяет сделать независимыми друг от друга спецификации в терминах метаонтологии и собственно спецификации онтологий, так как эти спецификации находятся на разных уровнях иерархии классификации. К слову, по той же причине нет ограничений на одновременное использование нескольких метаонтологий, рассматривающих онтологии с разных ракурсов предметных областей.

Сформированные спецификации в терминах метаонтологии можно использовать:

- для проверки корректности отображения понятий;
- при семантическом поиске релевантных понятий для дальнейшего отображения.

В этих задачах поглощающее (уточняющее) понятие должно находиться с поглощаемым (уточняемым) в одном классе (включая его подклассы), соответствующем понятию метаонтологии или служебного понятия.

Описанный подход берёт своё начало ещё в концептуальном моделировании, где важность метауровней была осознана изначально. В онтологическом моделировании актуальность такого подхода только возрастает.

Другой подход к формированию подложки использует общую онтологию верхнего уровня, содержащую наиболее общие понятия, используемые в любых предметных областях (например, DOLCE [20]). Согласовываемые онтологии встраиваются в иерархию понятия/подпонятия онтологии верхнего уровня (при таком подходе её некорректно называть метаонтологией). Этот путь более сложен, он включает задачу интеграции каждой онтологии в онтологию верхнего уровня и может приводить к изменению изначальных онтологий для совместимости их с онтологией верхнего уровня. Однако и он позволяет избежать некорректных отображений понятий между онтологиями.

4.3 Фундаментальные метасвойства

Онтология N. Guarino представляет собой набор свойств и отношений, которые являются утверждениями об объекте. В онтологии уделяется большое внимание различным видам свойств понятий [6]. С элементами онтологии может быть связан набор фундаментальных метасвойств, с точки зрения которых можно оценить любое понятие или отношение.

- существенность – неотъемлемость свойства сущности;
- строгость – принадлежность существенного свойства сущности в любом воображаемом контексте или мире;
- идентификация – является ли утверждение об объекте идентифицирующим его свойством;
- собственная идентификация – несёт ли эту идентификацию само утверждение, либо оно наследует её из других свойств или сущностей;
- неизменность – может ли свойство меняться во времени;

– зависимость – может ли сущность существовать без других;
– постоянство – как долго сущность остаётся таковой;
– объединение – существование экземпляров понятия как целых сущностей в отношении часть/целое, и другие.

Формальное определение подобных метасвойств предполагает некоторые ограничения, которые должны выполняться при их использовании с различными сущностями. Некоторые из этих метасвойств совместимы друг с другом, другие исключают друг друга. Для свойства q, поглощающего свойство p, верны следующие ограничения:
– если q строгое для любых сущностей, то p также строгое для любых сущностей;
– если q несёт критерий идентификации сущностей, то и p также;
– если q несёт критерий объединения, то и p также;
– если q не несёт объединение, то и p также;
– всякая сущность должна быть значением наиболее общего свойства, несущего его идентификацию, и другие.

В частности, при построении таксономии поглощения между понятиями, некоторые метасвойства более специфических понятий должны наследоваться, и обнаружение конфликтов метасвойств понятий в иерархии будет означать некорректность построения таксономии. На подобных правилах основан инструмент проверки и коррекции онтологий OntoClean [6].

Те же ограничения должны выполняться и для связей, выявленных между понятиями, принадлежащими разным онтологиям. Соответственно их можно использовать для обнаружения семантических конфликтов в результатах отображения онтологий. Если метасвойства связанных понятий из двух онтологий противоречат друг другу, это означает, что отображение было составлено некорректно, и понятия имеют разную семантику.

На основе метасвойств различных видов можно проводить и другие, более сложные рассуждения, полезные в задаче отображения онтологий.

Родовые понятия онтологии могут отражать идентификацию объектов реального мира, образовывать типы. Видовые – создавать категории объектов. Ролевые – относиться к ролям объектов реального мира. Ролевые понятия могут быть подпонятиями родовых. Видовые понятия могут быть подпонятиями ролевых или родовых понятий. Если касаться метасвойств отношений часть/целое, совокупность всех частей составляет целое, это может быть использовано для предположения связи понятий. Если часть является неотъемлемой, целое может быть идентифицировано по части. Если целое является инвариантным, то части можно идентифицировать по целому. Все эти знания могут быть использованы для корректного отображения понятий между онтологиями.

Данный подход некоторым образом связан с предыдущим описанным подходом отображения онтологий, использующим общую онтологию верхнего уровня для согласовываемых онтологий. Ведь фундаментальным понятиям онтологии верхнего уровня соответствуют вполне определённые наборы значений метасвойств. И эти метасвойства также задают требования к понятиям согласовываемых онтологий. Таким образом, при совместном использовании онтоло-

гии верхнего уровня и фундаментальных метасвойств понятий появляются дополнительные возможности контроля с помощью метасвойств корректности отнесения понятий онтологий в качестве подпонятий к понятиям онтологии верхнего уровня.

4.4 Экземпляры экстенсионалов понятий

И последний подход из представленных в статье, дополняющих отображение онтологий как схем семантическими методами обнаружения сходств и конфликтов понятийной семантики, связан с экземплярами классов понятий онтологий. Такими экземплярами могут становиться:
- объекты, соответствующие сущностям реального мира;
- примеры моделей реального мира;
- хорошо классифицированные с помощью онтологий данные.

На основе принадлежности экземпляров одним и тем же понятиям разных онтологий можно заниматься поиском релевантных понятий. Обратная задача – проверка экстенсиональной составляющей связанных понятий из согласовываемых онтологий. Существование хотя бы одного примера модели, в которой сущности не принадлежат одновременно классам, соответствующим эквивалентным понятиям согласовываемых онтологий, приводит к конфликту и ставит под сомнение корректность установленной связи между понятиями. Данный экстенсиональный подход к проверке отображения онтологий «по образцу» может быть реализован формальным образом.

Близкими подходами пользуются следующие проекты. FCA-Merge [12] – метод выявления отношений эквивалентности и подкласса между онтологиями, которые имеют набор общих экземпляров или набор общих документов, аннотируемых понятиями. Авторы также предлагают использовать методы обработки естественного языка для получения аннотаций набора документов понятиями нескольких онтологий. Проект GLUE [1] представляет оригинальный подход к связыванию онтологий, использующий обучающиеся машины для предположения близости элементов онтологий по данным экземпляров понятий.

5 Регламент работы экспертов

В подтверждение применимости и эффективности приведённых выше подходов к поддержке отображения онтологий, необходимо сказать о принципах работы экспертов над задачами согласования онтологий, привлекающих представленные подходы.

Необходимость вовлечения экспертов доказывается тем, что при формальности представленных подходов в каждом из них присутствуют предположения, не доказуемые формально:
- о достаточности отражения семантики понятий спецификациями онтологий как схем;

– о корректном описании понятий в терминах метаонтологий;
– о корректной оценке значений метасвойств, связанных с понятиями;
– о корректном отнесении сущностей реального мира или информационных объектов к определённым понятиям.

Применяя существующие, даже формальные, методики, невозможно автоматически согласовывать онтологии, созданные разными рабочими группами. Поэтому первым требованием к работе экспертов по согласованию онтологий является вовлечение в работу и в дискуссию экспертов-представителей каждой из согласовываемых онтологий (а также экспертов метаонтологий и фундаментальных метасвойств).

Эксперт в области своей компетенции может принимать ответственные решения, связанные с перечисленными выше проблемами: пояснять семантику понятий, не выраженную в спецификациях, выражать понятия своей онтологии в терминах метаонтологий, декларировать фундаментальные свойства понятий, предлагать примеры моделей реального мира и решать, как они выражаются в теминах его онтологии.

Семантические различия похожих понятий могут выясняться зачастую только в процессе дискуссий, на основе применения к согласовываемым онтологиям одной и той же метаинформации о понятиях. Каждый из представленных подходов может помочь экспертам эффективно обнаруживать скрытые конфликты при отображении онтологий. Реализация этих подходов может быть не только компьютеризированной. Они могут быть полезны экспертам в качестве:

– регламента обсуждений и дискуссий в ходе совместной работы по отображению онтологий;
– автоматизированной системы поддержки совместной работы экспертов по отображению онтологий в интерактивном режиме.

Таким образом, система поддержки отображения онтологий должна обеспечивать не столько работу автоматизированных методов, результаты которых должен контролировать эксперт, сколько совместное применение различных методов верификации отображений в ходе работы нескольких экспертов – представителей конкретных онтологий.

Усилия экспертов и их групп для создания онтологий порой занимают порой годы. При согласовании пересекающихся областей сталкиваются продуманные решения, и находить консенсус между ними бывает непросто. Интенсивно занимаясь оценкой различных алгоритмов выравнивания онтологий, исследователи, тем не менее, понимают ограничения этих подходов и отмечают необходимость и перспективность разработок, позволяющих оптимально поддерживать интеллектуальную работу экспертов при согласовании онтологий [4].

Заключение

В данной статье были представлены подходы к отображению онтологий, позволяющие учитывать специфику онтологической информации, не ограничиваясь методами применяемыми для отображения схем информационных систем и баз данных. Статья декларирует и обосновывает выбранные подходы для дальнейшего применения их в системе поддержки отображения онтологий как части системы интеграции спецификации неоднородных ресурсов в спецификациях задач. Дальнейшая работа по этому направлению предполагает разработку представления данных методов в модели языка, используемого в системе для спецификации схем и онтологий, и описание принципов их реализации в составе системы интеграции.

Литература

1. Doan, A.H., J. Madhavan, P. Domingos, A. Halevy: Learning to Map between Ontologies on the Semantic Web. WWW 2002
2. J. Euzenat, D. Loup, M. Touzani, P. Valtchev. Ontology Alignment with OLA. Proc. of the 3rd EON Workshop, 3rd Intl. Semantic Web Conference, Hiroshima, 2004
3. J. Euzenat, P. Shvaiko. Ontology Matching. Springer-Verlag, New York, 2007
4. S. M. Falconer, N. F. Noy, M.-A. Storey. Towards Understanding the Needs of Cognitive Support for Ontology Mapping. International Workshop on Ontology Matching, Athens, 2006
5. F. Giunchiglia, P. Shvaiko, and M. Yatskevich:. Semantic Schema Matching. In Proc. of CoopIS'05, volume 3760 of LNCS, pages 347–360, 2005
6. Guarino, N. and Welty, C. (2004), ''An overview of OntoClean'', in Staab, S. and Studer, R. (Eds), Handbook on Ontologies, Springer, Berlin, pp. 151–72
7. I. Horrocks, U. Sattler, S. Tobies. Practical reasoning for very expressive description logics. Logic Journal of IGPL, 8(3), 2000
8. L. Kalinichenko, M. Missikoff, F. Schiappelli, N. Skvortsov. Ontological Modeling. RCDL'2003. St.-Petersburg, 2003
9. J. Madhavan, P. A. Bernstein, E. Rahm. Generic Schema Matching with Cupid. In Proc. of the 27th Conference on Very Large Databases, 2001
10. N. Noy, M. Musen. The PROMPT Suite: Interactive Tools For Ontology Merging And Mapping. Stanford Medical Informatics, Stanford University, 2003
11. P. Shvaiko & J. Euzenat. Schema and ontology matching. Tutorial. ESWC'05, 2005
12. G. Stumme, A. Medche. FCA-Merge: Bottom-up merging of ontologies. IJCAI'01, Seattle, WA, 2000
13. J. Tang, J. Li, B. Liang, X. Huang, Y. Li, and K. Wang. Using Bayesian Decision for Ontology Mapping. Journal of Web Semantics, Vol(4) 4:243–262, 2006
14. S. Zghal, S. Ben Yahia, E Mephu Nguifo, Y. Slimani. SODA: an OWL-DL based ontology matching system In Proceedings of the first French Conference on Ontology (JFO 2007), Sousse, 2007
15. М. Р. Когаловский, Л. А. Калиниченко. Концептуальное моделирование в технологиях баз данных и онтологические модели. Симпозиум «Онтологическое моделирование», Звенигород, М: ИПИ РАН, 2008
16. Кудрявцев Д. В. Практические методы отображения и интеграции онтологий. Семинар Знания и онтологии *Elsewhere*, КИИ-2008, Дубна, 2008

17. Н. А. Скворцов. Использование системы интерактивного доказательства для отображения онтологий. RCDL'2006, Суздаль. – Ярославль: Ярославский государственный университет им. П. Г. Демидова, 2006. – С. 65–69.
18. Н. А. Скворцов. Вопросы согласования онтологических моделей и онтологических контекстов. Симпозиум «Онтологическое моделирование», М: ИПИ РАН, 2008
19. Н. А. Скворцов, С. А. Ступников. Использование онтологии верхнего уровня для отображения информационных моделей. RCDL'2008, Дубна: ОИЯИ, 2008 – С. 122–127
20. DOLCE: a Descriptive Ontology for Linguistic and Cognitive Engineering. http://www.loa-cnr.it/DOLCE.html
21. NeOn Glossary of Activities. Neon Project, 2007

Подходы к моделированию потоков информационных и вещественных ресурсов посредством систем поиска решений (обзор)

Survey of Modelling of Information and Material Resources by Means of Decision-Making

Г.В. Соколов
Gennady Sokolov

Институт систем информатики им. А.П. Ершова СО РАН
genvass@iis.nsk.su

Введение

Благодаря компьютерным средствам решаются проблемы пространства (сети связи), времени (скорость принятия адекватных решений) и проблемы объёмов переработки информации. Это, так сказать, явные результаты применения ЭВМ, а неявные — это оптимизация между энерго-вещественными и соответствующими информационными потоками в рассматриваемых системах. Надо полагать, что эта неявная оптимизация потоков-ресурсов в ближайшем будущем получит развитие до теории оптимизации между этими потоками как в самой системе так и в связях этой системы с внешней средой. В природе в явном виде не существует разделения этих потоков. Их разделение возникло в результате деятельности человека. Однако, поскольку человечество создало эти потоки, постольку его же задача оптимизировать эти потоки. Какие проблемы стоят на этом пути? При анализе публикаций на эту тему представляются следующие частные проблемы:

- отсутствие критериев оптимальности;
- разделение в пространстве и во времени каналов транспортировки потоков ресурсов;
- не соотносимые и постоянно изменяющиеся технологии передачи ресурсов по каналам их транспортировки;
- несоотносимые единицы измерения ресурсов.

Далее для пояснения указанной выше проблематики будут рассмотрены следующие работы. В статье [Колесников А.А., 2002 г.] отмечается: «… Основной принцип моделирования сложных систем - это управляемое (**оптимальное**) **взаимодействие и распределение** трёх **потоков ресурсов** системы: **энергии, вещества и информации**…**».**

В автореферате (д.т.н.) [Большаков Б.Е., 2000 г.] рассмотрена работа по теории развития системы общественное производство – природная среда, которая

представляется как динамическая сеть материальных потоков двух типов: **природных и общественных ресурсов**. Автором сформулированы требования – условия (необходимые и достаточные), которым должна удовлетворять **система измеримых величин**, и вводится **понятие потока** как величины, зависимой от пространства и времени.

В работе [Дружинин В.В. и др., 1989] отмечается: «… формализация надсистемных процессов природы и общества в целом требует установления единой энергетической меры био-психических и физических явлений … определить и количественно оценить истинную сущность интересов и намерений систем во взаимодействии … ».

В статье [Моторин С.А., 2002 г.] приведена классификация функций систем как функциональных проточных объектов с видами связей: материальные и информационные. А свойство системы понимается как проявление её активности «включаться» в связи, в **обменные потоки** с другими системами в структуре надсистем.

В книге [Емельянов В.В., и др., 2003 г.] обосновывается триадный подход к моделированию сложных систем. В триаде пара элементов находится в отношении дополнительности, а третий (каждый из трёх) элемент задаёт отношение (меру) совместимости. В книге также описаны генетические алгоритмы и их применение в оптимизационных задачах (с закодированным множеством параметров).

1 Синергетические системы [Колесников А.А., 2002 г.]

На системном (философском) уровне обращает на себя внимание статья Колесникова, суть которой излагается ниже. В статье дан обзор работ по концепции управления и самоуправления в соответствии с положениями синергетики. Автор отмечает то, что пора переходить от тенденции математического формализма теории управления к поиску общих объективных законов управления, к **учёту естественных и изменяющихся во времени и в пространстве свойств объекта**. Основной принцип моделирования сложных систем - это управляемое (**оптимальное**) **взаимодействие и распределение** трёх **потоков ресурсов** системы: **энергии (E), вещества (M) и информации (Inf)**. Далее учитывается **целевой способ самоорганизации** (информационное поведение сложных физических систем). Кроме этого способа указываются стихийный и причинный способы самоорганизации.

Сложные системы, обладающие бифуркационными и хаотическими свойствами, исследовались Г. Хакеном и И. Пригожиным. **Целевая самоорганизация** – это переход от непредсказуемого поведения системы к направленному движению вдоль желаемых инвариантных многообразий – **аттракторов** (значение вектора состояния функциональной группы элементов), к которым подстраиваются все другие переменные динамической системы. В результате состояние системы будет легко переходить с одной траектории на другую при малых воздействиях или при небольших структурных изменениях в системе.

В статье приводятся формулы количества информации (Inf) при задании переменных состояния системы.

$$Inf = \ln(V/\nabla V);\ S=K \ln N;\ dS=R(dN/N);\ S=0 \text{ при } N=1, \text{ где}$$

V – это полный объём фазового пространства , а ∇V – доля фазового объёма в начальном состоянии системы. Энтропия (S – беспорядок) системы равна логарифму N возможных состояний системы, где К– постоянная Больцмана (из статистической механики). Отсюда следует, что беспорядок (dS), вносимый в макросистему, пропорционален увеличению числа её микросостояний (dN/N). В пределе, когда в системе возможно лишь одно состояние (N = 1), её энтропия (S) равна 0. При росте рукотворных потоков ресурсов соответственно увеличивается хаос на нашей планете.

2 Роль метазнаний в системах поиска решений и многозначная логика [Соколов Г.В., 1991, 2000, 2002 гг.]

В указанных работах сделан обзор литературы по теории системных исследований в части метазнаний, которые являются основой для проектирования систем поиска решений. Показана важная роль оптимума между потоками информационных и вещественных ресурсов в развитии систем на всех иерархических уровнях, как основного критерия принципа «Триединства» материального мира. Приведена таблица отражения этого принципа в разных предметных областях.

На примере разработки системы «Старт С90» продемонстрировано преимущество оптимально выверенного соотношения функционального и структурного описаний, а также преимущество многозначной логики. В системе входные и выходные сигналы могут принимать четыре значения: истина (1), ложь (0), неопределённое значение (2) и отключение (3) – отсутствие влияния на внешние связи в заданный период времени. Таким способом обеспечивается и информационная замкнутость в нужный момент времени и создание сложных иерархических устройств, систем из базового набора элементов.

В статье [Соколов Г.В., 1991г.] показана граф-схема представления противоположно направленных потоков сигналов по одному каналу связи и представления сигналов обратной связи, которая предшествует этапу создания программы имитационной модели устройства.

3 Единая энергетическая мера био-психических и физических явлений [Дружинин В.В. и др., 1989]

В данной работе рассматривается теория конфликта, который представляется как способ взаимодействия сложных систем. Теория конфликта позволяет (стр. 277):

- выявлять скрытые тенденции поведения и взаимодействия, перспективы;

– определить и количественно оценить истинную сущность интересов и намерений систем во взаимодействии.

Если описание конфликта не сводимо к единому языку с единой системой размерностей величин, то конфликт многокритериален. Такие конфликты оптимально неразрешимы, критерии неравносильны, а их ранг ситуационен и трудновыявляем. В разделе «Реальный конфликт», стр. 86, приводится таблица влияния (в баллах) личностных свойств (около 80) на принятие решения. Авторы отмечают – формализация надсистемных процессов природы и общества в целом требует установления единой энергетической меры био-психических и физических явлений.

Управление сложными системами возможно только ещё более сложными, [Ред. Волкова В.Н., 2004], Эшби У.Р. – закон необходимого разнообразия.

4 Система измеримых пространственно-временных величин [Большаков Б.Е., 2000 г.]

Это одна из тех редких работ, в которой на методологическом уровне достигается отображение и сочленение разнородных предметных областей.

Автор диссертации (д.т.н.) решает следующую проблему. В традиционной экономической теории природная среда не учитывается ни в постоянном капитале, ни в переменном. Для разрешения противоречия между обществом и природной средой необходимо соизмерять разнокачественные общественные и естественные процессы-потоки.

В данной работе сформулированы необходимые и достаточные требования–условия к системе измеримых величин для модели взаимодействия социального объекта с окружающей его природной средой. Величины должны выражать **сущность** общественно-природных систем – **процесса обмена** материальными и информационными потоками ресурсов между обществом и природной средой.

Достаточное условие – величины должны быть естественными (физическими), устойчивыми и универсальными (когда ясна их связь с пространством – L и временем – T).

Как удовлетворяющей указанным требованиям к системе измеримых величин, автором предложено использовать таблицу пространственно-временных величин Д. Максвелла (1973), Р. Бартини (1965) и Е. Лифшица (1969) с формулой размерности $[L^R T^S]$, где: L – длина (см), T – время (сек), R и S – целые числа (положительные и отрицательные). Соединение «пространственных» и «временных» величин при $R \neq 0$ и $S \neq 0$ даёт словарь исходных терминов. Сама система $[L^R T^S]$ является классификатором систем реального мира и она бесконечна. Например, скорость движения объекта представляется как $[L^1 T^{-1}]$.

Каждая величина системы $[L^R T^S]$ имеет следующие свойства:

– это качественно-количественная определённость. Качество определяется именем, размерностью и единицей измерения, а количество – численными значениями величины;

– является сущностью-инвариантом определённого класса систем. Переход от одного класса систем к другому означает переход к другой сущности, другому качеству, с другими волновыми характеристиками.
– может быть представлена как скаляр, вектор, тензор;
– Величина $[L^R T^S]$ является потоком, если S < 0, R > 0. Понятие материальный поток используется для отображения динамики ресурсов в отношениях общества с окружающей средой.

Данная работа автора является итогом многолетних теоретических, методологических и прикладных исследований по системному естественнонаучному анализу управления развитием в социальных и экономических системах, рассматриваемых во взаимодействии с окружающей средой. Автором читался курс лекций «Основы теории устойчивого развития общественно-природных систем» (в программах ООН). В диссертации дана прогнозно-предупредительная информация по критическим точкам в отношениях общество – биосфера. В 2050 г. мощность цивилизации может сравняться с мощностью биосферы, а в 2200 г. с мощностью солнечной энергии на поверхности Земли.

5 Формальное представление системы как функционального «проточного» объекта [Моторин С.А., 2002 г.]

Разработка автора посвящена знание–ориентированному развитию системного анализа и системологическому развитию объектного подхода. В статье приведена классификация функций систем, как функциональных проточных объектов с видами связей: материальные (вещественные и энергетические), информационные (данные и управляющие). Следовало бы добавить в раздел информационных и «служебные», [Соколов Г.В., 1991 г.]. Алгоритм УФО анализа (UFO – User Functional Object) позволяет формализовать метод объектно-ориентированного системологического анализа и моделирования организационных и информационных систем. Алгоритм состоит в согласованном выявлении и моделировании структурных, функциональных и ролевых (часть – целое, приток – отток) характеристик системы.

Шаги анализа, проектирования и реализации:
– выявление узлов (пересечений), поддерживающих потоки в связях структуры разрабатываемой системы;
– выявление функциональности обнаруженных узлов;
– определение объектов, реализующих функциональность узлов.

Интерпретация классов базовой иерархии (элемент, функция, явление) определяется в УФО иерархии как объект со связями, функция с отношениями, проточный узел. Проточный узел рассматривается как целостность, как структурная часть ещё более целого. **Свойство системы** понимается как проявление её активности «включаться» в связи, в **обменные потоки** с другими системами в структуре надсистем. То есть, быть «проточным узлом, элементом» в сети

замкнутых обменных потоков надобъекта, обеспечивая баланс «притока» и «оттока» по входящим и выходящим связям.

Конкретная система S с позиции структуры надсистемы-явления представляется как <O, F, (lk)L>, где:

- O – конкретный класс «Объект», экземпляром которого является система S;
- F – конкретный класс «Функция»;
- (lk)L – (часть или целое) – «явление» узел или связи lk, показателями которых служат типы L.

То есть, система S, как функциональный проточный объект, представляет собой сущность, характеризующуюся узлом (lk)L в структуре надсистемы, множеством функций, балансирующих данный узел, и множеством объектов, реализующих данные функции.

Формально УФО-элемент может быть представлен как класс языка объектного моделирования UML. УФО-анализ разрабатывался с целью повышения объективности анализа и адекватности моделирования.

6 Эволюционные методы решения оптимизационных задач и принятия решений [Емельянов В.В., Курейчик В.В., Курейчик В.М., 2003 г.]

Разработке теории, математических методов и моделированию посвящена книга известных авторов. В этой работе достаточно много материала теоретического и практического характера: и по эволюционному развитию сложных систем, и по имитационному моделированию, и по генетическим алгоритмам. В книге дано множество оригинальных формулировок основных понятий по указанным темам. В частности, в книге рассматривается подход к синтезу систем, к описанию законов и структур на основе «триад». «Триада – это комплекс трёх равноправных объектов, находящихся в заданных отношениях, например, «цель – план – стратегия», «изменчивость – наследственность – отбор», «функция – аргумент – значение». В триаде пара элементов находится в отношении дополнительности, а третий (каждый из трёх) элемент задаёт отношение (меру) совместимости. Диада – это ровно одна причина и одно следствие. Бинарная парадигма утрачивает перспективу постижения целостности», (стр.58).

Вид – это основная структурная единица в естественных системах, качественный этап их эволюции. Популяция – это многочисленная совокупность особей определённого вида, это основа эволюционных процессов. Эволюция созидательна и не имеет строго определённой внутренней цели. Таким образом, информацию мы получаем из окружающей среды и сами являемся носителями информации предыдущих поколений. Процесс эволюции есть **процесс увеличения сложности** (увеличение информации), а **сложная система требует распределённой и иерархической системы управления**, (стр.44).

В книге описаны генетические алгоритмы (ГА) и их применение в оптимизационных задачах. ГА – это модель эволюции в природе, механизм комбинаторного перебора вариантов решения оптимизационных задач. **ГА работает не с параметрами задачи, а с закодированным множеством пара-**

метров. Поэтому эволюционное моделирование ближе к природе моделируемых объектов, так как степень разложения объекта на части не так детальна, как того требуют другие способы моделирования систем.

Как правило, специалисты предметных областей используют не логический вывод, а аргументацию – обоснование некоторых выводов в системе имеющихся у них знаний. Отсюда предпочтителен переход к моделям обоснования и оправдания выводов.

Заключение

В статье дан обзор различных подходов к моделированию потоков информационных и вещественных ресурсов посредством систем поиска решений. В частности, показана возможность соизмерения разнокачественных общественных и естественных процесс-потоков с помощью системы измеримых величин (Большаков Б.Е.), обосновывается "триадный" подход к моделированию сложных систем (Емельянов В.В. и др.).

Список литературы

1. Соколов Г.В. ТРИЗ и системы поиска решений. 2000г. http://iis.nsk.su/solver\sokolov\steors
2. Загорулько Ю.А., Соколов Г.В. Функциональное моделирование логических элементов на основе системы представления знаний SEMP-TAO // Труды Международных конференций "Искусственные интеллектуальные системы" (IEEE AIS`02) и "Интеллектуальные САПР" (CAD-2002). Научное издание. – М.: Изд-во Физматлит, 2002. – 609
3. Колесников А.А. Синергетические системы: – «Программные продукты и системы», № 1, 2002.
4. Соколов Г.В. Имитационная система функционального проектирования и моделирования Старт С90 // сб. Методы теоретического и системного программирования / Под ред. В.Е. Котова. – Новосибирск, 1991.
5. Большаков Б.Е. Основы теории развития системы общественное производство – природная среда с использованием измеримых величин. Автореферат (д.т.н.). – Дубна, 2000.
6. Емельянов В.В., Курейчик В.В., Курейчик В.М. Теория и практика эволюционного моделирования. – М.: ФИЗМАТЛИТ, 2003.
7. Системный анализ и принятие решений: Словарь-справочник: Учеб. пособие для вузов / Под ред. В.Н.Волковой, В.Н. Козлова. – М.: Высш. шк., 2004 – 616 с.
8. Дружинин В.В., Конторов Д.С., Конторов М.Д. Введение в теорию конфликта. – М.: Радио и связь, 1989.
9. Маторин С.И. О новом методе системологического анализа, согласованном с процедурой объектно-ориентированного проектирования // Кибернетика и системный анализ, 2002, №1.

Author Index

Akinin, Alexander, 1
Andrichenko, A., 28
Apanovich, Zinaida, 33

Beniaminov, Evgeny, 47
Boldasov, Michael, 13

Garanina, Natalia, 72
Gavrilova, Tatiana, 22
Gorovoy, Vladimir, 22

Korobko, Anna, 87

Lapshin, Vladimir, 47

Mouromtsev, Dmitry, 22

Penkova, Tatyana, 87
Perov, Dmitry, 47
Petrashen, Elena, 22

Scherbakov, N., 28
Shilov, Nikolay, 1
Skvortsov, Nikolay, 91
Sokolov, Gennady, 105
Sokolova, Elena, 13

Vinokurov, Pavel, 33
Vorontsov, Konstantin, 57

Zubkov, Alexey, 1