UDC 519.178, 519.711

Pavlov D.
Artificial Intelligence Department,
Kharkiv National University of Radioelectronics,
Lenin Av., 14, Kharkiv, 61166, Ukraine,
E-mail: dmitry_pavlov@ukr.net

DISTRIBUTED ONTOLOGIES STRUCTURAL-TIME ANALYSIS SYSTEM

There is emphasized actuality of developing ontological structures research. A complex distributed developed ontologies structural-time analysis system is suggested. The system greatly increases facilities of an expert, who supports ontological structure's functioning, and allows evaluating importance of changes inside the structure more effective and accurate. There is described a general functional scheme of the system.

Keywords – ontology, ontology evaluation, modular ontology, ontology development, version control systems.

1. Introduction

As ontologies and their groups, so called ontological structures (OS), are complex objects, their life cycle contains lots of improvement stages. Because of same reason tracing and evaluating of changes happening during their development is really complex task. It stays on the edge of abilities of highly qualified specialists in knowledge engineering and object domain. All this shows that it is necessary to create a system, which could simplify such problems solving. Such system is actual and demanded because of significant growth of ontology usage fields' number.

There exist approaches, which are used with developed ontologies [1-5] and modular OS [2,6]. Despite their considerable problem analysis, they do not give a whole Fig. of OS change process, which is vital for an adequate decisions making.

So it is necessary to suggest a decision support system for developed object domain OS analysis. Taking into account current tendencies in ontological knowledge representation [7] it is needed to support operating with fuzzy ontologies. The system must give a whole set of automatic OS analysis capabilities and also a number of basic mechanisms of its transformation.

2. Typification of structural-time analysis

Structural-time analysis subsumes evaluation of development both in time and in space, or in other words intensive and extensive.

Analysis is effectively categorized by two features into four types (Table I): analysis of single ontologies or OS wholly, and analysis in case of constant or variable time.

Basic processing of single ontologies (A) is a prior stage, which is needed to lower the number of unnecessary computations. There are exist a number of effective methods, which solve the task [1,3]. Analysis of structure integrity (B) has a lot in common with A, but there exist some features peculiar to modular ontologies evaluation [2]. There is suggested to process a single developed ontology (C) using theory stated in [8]. This processing is realized by computation of inadequacy criteria (nAd), in particular incompleteness (Npl) and superfluity (Iz). A complex analysis of a developed OS (D) has features peculiar to both B and C.

Let's note that influence of single structure changes upon structure's constant parts is a considerable element of OS model adequacy evaluation. For example, consider an OS Struct`. Let Struct` = Struct/Ont, where Ont is an ontology, Struct is an OS. We'll take up two cases: Struct`(t) = Struct`(t+1) (D.1) and Ont(t) = Ont(t+1) (D.2), where t is time of creation of the corresponding object. Each of the cases is a subtype of analysis D for Struct, but they have a number of features that lower computational complexity.

TABLE 1

ANALYSIS TYPIFICATION

	Single ontologies	OS wholly
Constant time	A	B
Variable time	C	D (D.1, D.2)

3. System structure

There is suggested a system. Its functional scheme is depicted on the Fig. 1.

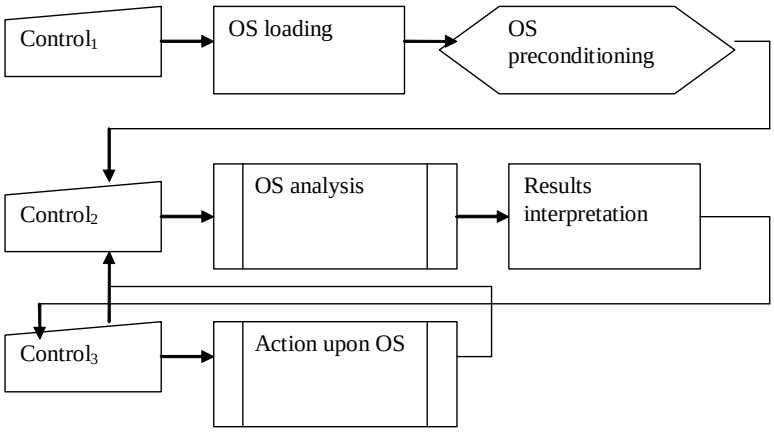

Fig. 1 General functional scheme of the system

Let's consider each system element separately.

Control1: on this stage a user could choose an OS and specify a number of settings (for example, to substitute physical ontology names for virtual).

Loading: there is made a hybrid extensive-intensive OS development model during the loading, Struct = <MO, MRI> and {Dj}, where MO is a set of ontologies, contained by the OS, MRI is a set of import relations, Dj is a development model of an ontology Ontj, where Ontj∈MO, $D = \left\{ \left\langle Ont_i, t_i \right\rangle_i \middle| t_i < t_{i+1}, i = \overline{0,I} \right\}$, t is a time of ontology creation.

Preconditioning includes capabilities to simplify the base set of import relations MRI, and to shift out certainly inadequate stages of development Dij.

Control2: this stage is the main part of system control. It gives whole selection between all the analysis methods.

OS analysis: this system function is called correspondingly to the selection made on the previous stage. It could be:

- checking of any OS element separately;
- checking of OS integrity at moment t, Ad(Struct(t));
- computation of adequacy criteria values for current ontology development, Ad(Ont(t), Ont(t+1));
- computation of adequacy criteria values for ontology development, when a connected to it ontology is developed, Ad(Ont(t), Ont(t+1)), where imp(Ont, Onta) = true, Onta(t) ≠ Onta(t+1), where imp is an importing function;
- computation of current ontology version actuality in case of whole OS change, Ad(Ont(t), Ont(t+1))| Struct(t) ≠ Struct(t+1);
- localization of inadequacy sources, nAd(Sub)→max, Sub⊆Ont, nAd(Ont) ≠ 0.

Results interpretation: this function subsumes automatic evaluation of the results obtained on the previous stage. The function returns evaluation of inadequacy degree results, results of estimation of the OS usage possibility, or a set of suggestions how to act upon the structure to lower its inadequacy, nAd(Struct)→min.

Control3: this stage is optional and completely based on the results obtained on the previous step. Here a user chooses an option how to act upon the analyzed OS. The type of acting could be chosen from a list generated automatically during the interpretation.

Action upon OS: at this level of system development we have following mechanisms:

- a search of the top adequacy bound;
- OS parts changing cancellation;
- OS elements addition.

4. Analysis results interpretation and action upon OS

Let's consider variety of results interpretation correspondingly to the typification shown on the Table I.

There are suggested to analyze following things for type A: correspondence to the language, formal logic rules, additional rules. Errors are interpreted into instructions for an expert. A positive result of the analysis allows working with more complex functions.

There is checked for type B existence of elements, which are used to connect ontologies. Having a number of them absent, we assume that there exists a previous or a following version of the imported ontology, which should've contained them. We also check observation of the formal logic rules inside whole OS. Having some of

them violated there is suggested to find the least inadequate substructure Submin or the largest adequate substructure Submax, Submin⊆Struct, Submax⊆Struct [3]. These substructures could help an expert during a nonautomatic errors checking.

Considerable values of incompleteness criteria (Npl) in analysis C are interpreted as an object domain specification reduction or its narrowing, Npl(Ont) > a. There is suggested to an expert checking of the ontology structure, which is marked with degree of each element changing [8]. If the made assumption is wrong, then it is supposed that the missing information is moved to ontology Ont1, which is another part of the OS. Consequently Ont1 should have increased value of superfluity criteria, Iz(Ont1) > b. We suggest to compute the criteria Npl and Iz for a union of Ont and Ont1, Npl(Ont∩Ont1) = x, Iz(Ont∩Ont1) = y. If x < Npl(Ont) and y < Iz(Ont1), then the conclusion is assumed to be true and is shown to the expert. In other case we suggest him considering focused incomplete substructures. Processing of superfluity Iz is made similarly to Npl.

Considerable values of Npl and Iz in case D.1 show that the unchanged ontology with high certainty degree became out of date and needs a close revision. The same features in case D.2 show that someone should undo most contradictive changes in the developed ontology or he should examine the OS under the condition of a new object domain tasks. At the present development level of the system we suggest in case of common analysis type D to convert the OS and solve the task as D.1 or D.2.

OS actions are made on two conceptually different levels, of OS elements, and ontology elements. Actions upon OS elements are choice of a specific ontology version, transformation of imports MRI, extraction of a subset of analyzed ontologies in MO. Actions upon ontology elements are addition and deletion of change assertions (ontology nodes N and relations E), insertion of additional assertions (which are alternate to the changes made), changing values of membership functions.

5. Conclusion

There is suggested at first time a distributed ontologies structural-time analysis system. The system's analysis is suggested to be categorized into four types and gives means to part a complex nontrivial task into a set of simple. The system greatly increases facilities of an expert, who supports ontological structure's functioning, and allows evaluating importance of changes inside the structure more effective and accurate. There is described a general functional scheme of the system and each its function's acting methods and tasks solved.

There are developed complex ontology evaluation methods, which give facilities to automate some aspects of inadequacy elimination. There is suggested an "expert" - "result interpretation functions" dialog mechanism, which potentially could be improved up to an effective decision support system.

Further research lays in integration of the system with existing ontology development systems. There are peculiarities of automatic and nonautomatic ontology generation methods, which also need a close attention.

References

[1] N. F. Noy, M. A. Musen, "PROMPT: Algorithm and Tool for Automated Ontology Merging and Alignment," Seventeenth National Conference on Artificial Intelligence (AAAI-2000), Austin, TX, 2000. http://dit.unitn.it/~p2p/RelatedWork/Matching/SMI-2000-0831.pdf (12.04.2006)

[2] H. Stuckenschmidt, M. Klein, "Integrity and change in modular ontologies," In Proceedings of the International Joint Conference on Artificial Intelligence - IJCAI'03, Acapulco, Mexico, Morgan Kaufmann, pp. 900-908, 2003.

[3] P. Haase, L. Stojanovic, "Consistent evolution of OWL ontologies," In Proceedings of the Second European Semantic Web Conference, Heraklion, Greece, 2005. http://www.aifb.uni-karlsruhe.de/WBS/pha/publications/owlevolution05eswc.pdf (12.04.2006)

[4] J. Heflin, Z. Pan, "A Model Theoretic Semantics for Ontology Versioning," ISWC 2004, LNCS 3298, pp. 62–76, 2004.

[5] M. Klein, "Change Management for Distributed Ontologies," PhD thesis, Vrije Universiteit, 206 p., 2004.

[6] L. Serafini, A. Borgida, A. Tamilin, "Aspects of Distributed and Modular Ontology Reasoning," In Proc. of IJCAI-2005, 2005. http://sra.itc.it/people/serafini/distribution/ddl-ijcai-05-techrep.pdf (09.06.2006)

[7] Ye. I. Kucherenko, D.A. Pavlov, "Some aspects of fuzzy ontologies development analysis," Artificial Intelligence Vol. 3, Donetsk, pp.162-169, Sep. 2005.

[8] Ye. I. Kucherenko, D.A. Pavlov, "About problems of information incompleteness and superfluity display in an ontological space," Applied radioelectronics, Kharkiv, Vol.4, N2, pp. 175-179, 2005.