Re-engineering OntoSem Ontology Towards OWL DL Compliance

Guntis BARZDINS, Normunds GRUZITIS and Renars KUDINS
Institute of Mathematics and Computer Science, University of Latvia
Raina bulv. 29, Riga, LV-1459, Latvia
guntis@latnet.lv, normundsg@ailab.lv, renars.kudins@gmail.com

Abstract. Re-engineering of successful pre-OWL ontologies or other formal ER or UML system models towards OWL DL compliance opens new possibilities in ontology debugging, enabled by the formal semantics and automated reasoners developed for OWL DL, such as RacerPro and others. Meanwhile the transformation of pre-OWL ontologies to OWL DL is a challenging and interesting task, which we illustrate in this paper on the example of the OntoSem lexical common-sense ontology and its application framework for deep semantic analysis of natural language texts.

Keywords. Common sense ontology, OWL DL, reasoning, natural language processing, text-meaning representation.

Introduction

Reported research is based on the SemTi-Kamols project [1] at the Institute of Mathematics and Computer Science, University of Latvia, which aims at developing resources and methodologies for efficient integration of Latvian language and the latest semantic web technologies. If successful, this approach promises to open new possibilities, enabled by automatic meaning extraction from texts or generating semantically correct translations. This project is part of a larger "Semantic Latvia" initiative [1], aiming at coordinated development of semantically rich ontology-driven applications in Latvia.

For ontology-driven application development, the key advantage of conforming to W3C Semantic Web standards and particularly to OWL (Web Ontology Language) [2] is the eventual opportunity to integrate multiple ontologies into the "global" semantic web, as well as the possibility to apply an ever-growing arsenal of powerful tools being developed for OWL. The focus of this paper is on OWL DL, a sublanguage of OWL with strictly defined semantics rooted in description logic. OWL DL formal semantics enables ontology debugging [12] and application development based on formal logic and automated reasoners, such as RacerPro [4], FaCT++ [6], or Pellet [11]. These advantages are a good reason for attempting to re-engineer the successful pre-OWL ontologies (or other formal models, such as ER-database models or UML system models) to OWL DL compliance. In our view, this shall be a true "gold rush" waiting

1 Project is funded by the National Research Program and partially supported by European Social Fund. Anno 2005. Project website: http://www.semti-kamols.lv

to happen, because the successful conceptualization of some application domain for the purpose of building a formal model or ontology is a notoriously difficult task and any reuse possibility there is highly rewarding.

Re-engineering of a pre-OWL ontology towards OWL DL compliance consists of two stages: transformation of the original ontology into OWL DL notation (Section 2) and semantic debugging of the transformed ontology by means of formal OWL DL reasoning techniques (Section 3). It is important to keep in mind that in most cases the ultimate goal of the whole ontology transformation exercise is not the OWL DL compliance itself, but rather the possibility to create debugged and highly consistent ontologies — a goal hardly achievable in the pre-OWL era.

1. The OntoSem ontology

In this paper we are discussing transformation towards OWL DL compliance[2] of a pre-OWL common sense lexical ontology OntoSem [10]. Besides that, in Section 4 of this paper we briefly discuss also the re-engineering of the related ontology-driven lexical application for deep semantic analysis of natural language texts [7, 8, 9] and how it can contribute to further ontology testing.

The original OntoSem ontology was designed in the proprietary frame-based LISP notation. The semantics of this ontology is based on the common sense of its authors and to some extent encoded also into the text analysis application driven by this ontology. The upper levels of the original OntoSem ontology are depicted in Figure 1.

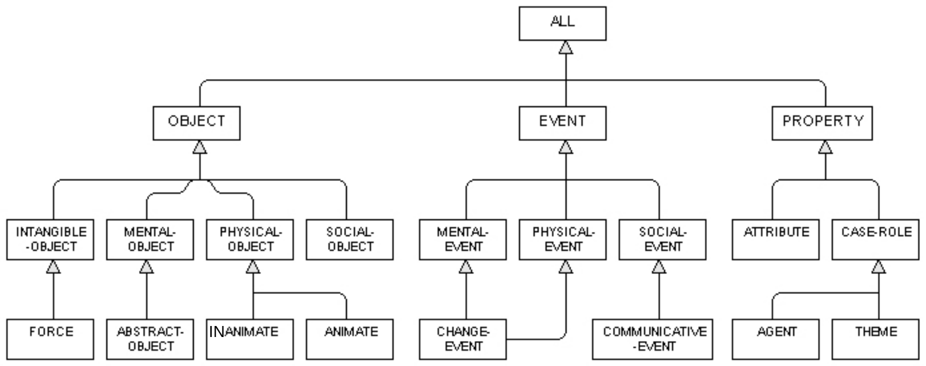


Figure 1. Fragment of the upper levels of the original OntoSem ontology, which tries to capture the most universal object, event and property (relation) concepts referred to by the natural language texts.

The reason for choosing specifically OntoSem framework for the further discussion, is that to our knowledge currently it is the most successful attempt to integrate lexical knowledge of natural language with the full-featured ontological "world model" for semantic disambiguation and formal text meaning representation (TMR, see Section 4). Unlike many other lexico-ontological approaches, which primarily concentrate on building large lexical taxonomies (like WordNet [3]) without a clear application in mind, the OntoSem framework incorporates not only a detailed high-level ontology of

[2] It shall be noted that OntoSem authors have converted OntoSem ontology into OWL format [7], but this conversion has not resulted in the OWL DL compliant ontology.

~8000 concepts and ~350 properties, but also an ontology driven application for automatic creation of coherent TMR repository from natural language sentences. Thus OntoSem ontology is practically validated by its successful application. Besides that, OntoSem framework has been already successfully used for the number of non-English languages [8], what gave assurance that the ontology and application framework is sufficiently language independent to be used also for Latvian.

2. Transformation of OntoSem ontology into OWL DL syntax

This re-engineering step is specific to the pre-OWL ontology considered. Here we will only briefly illustrate the typical steps and problems encountered during such transformation. The used syntactic conversion rules are similar to the ones already described in [7], therefore here, by means of examples, we will only illustrate the problematic issues specific to OWL DL.

Besides syntactic differences, Figure 2 shows also more essential changes imposed by formal semantics of OWL DL. For example, we are forced to explicitly state that "location" of "soccer" event is a "unionOf" concepts "playing-field" and "sports-arena" (and not, for example, "intersectionOf" or "complementOf" of these concepts) — note, that this information is not explicitly present in the original ontology, although this appears to be the intended meaning.

"soccer" in the original OntoSem ontology	**"soccer" in the OWL DL ontology**
(make-frame soccer	<owl:Class rdf:about="#soccer">
(agent (inv (striker)))	<rdfs:subClassOf rdf:resource="#sports-discipline"/>
(is-a (value (sports-discipline)))	<rdfs:subClassOf>
(instrument (inv (soccer-ball)))	<owl:Restriction>
(location (sem (playing-field sports-arena))))	<owl:onProperty rdf:resource="#location"/>
	<owl:allValuesFrom>
	<owl:Class>
	<owl:unionOf rdf:parseType="Collection">
	<owl:Class rdf:about="#playing-field"/>
	<owl:Class rdf:about="#sports-arena"/>
	</owl:unionOf>
	</owl:Class>
	</owl:allValuesFrom>
	</owl:Restriction>
	</rdfs:subClassOf>
	</owl:Class>

Figure 2. Definition of the concept "soccer" in the original LISP-like syntax of OntoSem ontology and the corresponding transformation into OWL DL.

Another issue is that defining an OWL DL "allValuesFrom" restriction on property "location" of "soccer" event being "unionOf" "playing-field" and "sports-arena" might be not sufficient. This restriction means "only in case soccer event has a location, then it is either playing-field or sports-arena". In reality in this case we might want to state also "any soccer event has a location, which is either playing-field or sports-arena". For the second assertion we would need to add also a “someValuesFrom” restriction on “location” of “soccer” being the same “unionOf” “playing-field” and “sports-arena”. In Figure 2 this second restriction is not shown for the sake of simplicity (and because for some properties it indeed is optional); this issue will be discussed also in Section 3.

Inverse relations highlight one more problem. In Figure 2, relation “(agent (inv (striker)))” has been added automatically in the original OntoSem ontology based on the explicit definition of the inverse relation “agent-of” under the “striker” concept:

```
(make-frame striker
    (is-a (value (athlete)))
    (agent-of (sem (soccer))))
```

Semantics of such automatically added inverse relations might be ambiguous. Does it imply a restriction on the “agent” of the “soccer” to be only a “striker”? Or can “soccer” still have other “agent” types, e.g. “goalkeeper” and “defenseman”, which are subclasses of “athlete” — a more general “agent” type inherited from the “soccer’s” super-class “sports-discipline”? Second interpretation is assumed in the OWL DL translation shown in Figure 2 (restriction for “agent” is not explicitly present in the “soccer” definition, because in OWL DL it is automatically inherited from the super-class “sports-discipline”). Meanwhile, the first interpretation might have been more appropriate for single-player sports disciplines (e.g. “large hill ski jumping”), where restriction to the specific skills (e.g. “ski jumper”) is appropriate instead of permitting the generic “athlete” inherited from the “sports-discipline”.

The most difficult step in the transformation towards OWL DL is the mandatory requirement to correctly specify which ontology concepts are disjoint (mutually exclusive) and which are not. For example, OntoSem concepts representing social roles “bride” and “groom” are disjoint, while social roles “singer” and “dancer” are not. Unfortunately, the disjointness information is not present in the original OntoSem ontology at all, which means that it has to be added manually during the transformation to OWL DL. As we will show in Section 3, adding correct “disjointWith” information to the ontology is essential both for lexical disambiguation, as well as for reasoning in (by default, classes in OWL DL are allowed to overlap, unless they are specifically defined to be “disjointWith” each other.) Without adding “disjointWith” statements where appropriate, reasoning is limited to only simple taxonomy checks. Of course, it is virtually impossible to automate adding of the “disjointWith” statements, as this is one of the key steps on the way from informal to formal ontology semantics. Practical experience shows, that it is rational to follow top-down approach here, as stating disjointness between ontology upper level concepts will make immediate impact on subclasses as well. Obviously, the process should be performed with support of a prearranged methodology to gain consistent and objective usage of “disjointWith” statements.

These examples are by no means a complete list of problems encountered during transformation, but we tried to illustrate the most essential ones specific to achieving OWL DL compliance; most of other more conventional OntoSem to OWL conversion steps are already well described in [7].

3. Debugging of the transformed OWL DL ontology

Once the ontology is finally converted into the OWL DL syntax, we can start reaping the fruits of this exercise — by applying an automated OWL DL reasoner, like RacerPro, we can automatically obtain the list of logical errors, warnings and redundancies found in our ontology. Automated reasoning in OWL DL allows finding various types of common ontology errors, like inheritance anomalies (classification), classes not permitted to have any individuals due to erroneous restrictions (satisfiability/consistency), and some other types of errors related to class individuals. Needless to say, in the large ontology like OntoSem (with thousands of classes and properties), finding all such bugs manually would be a virtually impossible task.

The number of errors we discovered in OntoSem ontology by automated reasoning ranges in hundreds, depending on the error types we are interested in. We will show only few errors found, to illustrate common ontology error types, which can be discovered automatically thanks to OWL DL formal semantics.

The most basic reasoning test for any OWL DL ontology is consistency check — are there any classes, which are not allowed to have any individuals? This test relies on the “disjointWith” information added to the ontology in the previous section. Assuming that “object” and “event” classes have been defined to be “disjointWith” each other in the previous step, the RacerPro reasoner finds about 30 (out of 8000) classes to be inconsistent. For example, the class “recreation” is found to be inconsistent — if we trace this class back to the original OntoSem ontology, we see that the inconsistency was present already there (see Figure 3).

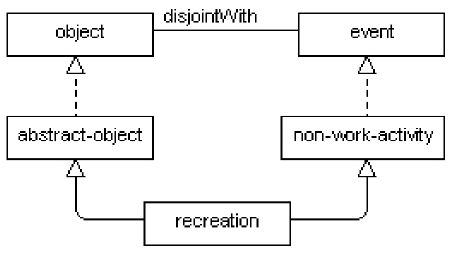
Excerpt form the OntoSem[3]	**UML representation**
(make-frame recreation (is-a (value (abstract-object non-work-activity)))	

Figure 3. Unsatisfiability error found by automated reasoning over the improved ontology.

Though, this example raises also a discussion whether our assumption that all the events are disjoint from all the objects is true? Indeed, the specified axiom seems to be too restrictive (attached too high in the ontology) if we consider other concepts with similar dual nature as well, e.g. “chart” in OntoSem is defined both as “physical-object” and as “mental-object”. This entails difficulties in decision making while applying “disjointWith” statements and/or resolving inconsistencies found by reasoner.

The next basic reasoning test is subclass hierarchy check — are there any subclass classification bugs? Figure 4 illustrates one of the subclass classification redundancies found by RacerPro reasoner.

---

[3] Although debugging is being performed exploiting the OWL DL form of the ontology, we provide excerpts in the LISP syntax for readability as far it concerns only syntactical differences.

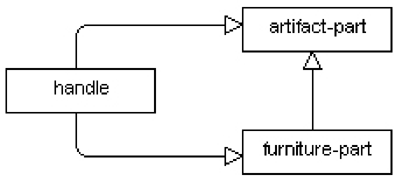
Excerpt form the OntoSem	UML representation
(make-frame handle (is-a (value (artifact-part furniture-part)))) (make-frame furniture-part (is-a (value (artifact-part))))	

Figure 4. Classification error found by automated reasoning over the ontology.

A more difficult to find error type is illustrated in the Figure 5. In this example the unsatisfiability error is hidden behind the inheritance — here the relation “theme” of class “eat-meal” has direct “someValuesFrom” restriction towards “ingestible” and inherited “allValuesFrom” restriction towards “abstract-object”. Since “ingestible” and “abstract-object” are subclasses of “physical-object” and “mental-object” (which we have defined to be “disjointWith” each other), this leads to the unsatisfiability of the class “eat-meal” (it can’t be instantiated). Also this error is found by the RacerPro reasoner, provided that “someValuesFrom” and “allValuesFrom” restrictions are correctly added to the ontology (see Section 2). Note that Figure 5 shows only those restrictions involved in the inconsistency — in reality ontology contains both “someValuesFrom” and “allValuesFrom” restrictions for the relation “theme” of class “eat-meal”.

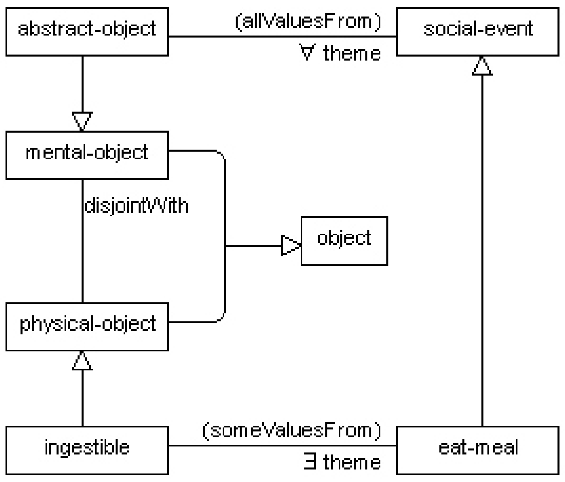
Excerpt form the OntoSem	UML representation of the transformed version
(make-frame eat-meal (is-a (value (social-event))) (theme (sem (ingestible)))) (make-frame social-event (is-a (value (event))) (theme (sem (abstract-object)))) (make-frame ingestible (is-a (value (inanimate)))) (make-frame inanimate (is-a (value (physical-object)))) (make-frame abstract-object (is-a (value (mental-object))))	 abstract-object (allValuesFrom) ∀ theme social-event mental-object disjointWith object physical-object ingestible (someValuesFrom) ∃ theme eat-meal

Figure 5. Unsatisfiability error found by automated reasoning over the ontology.

The illustrated error is typical for lexical ontologies, and originates from the polysemy of the concept “eat-meal” and its property “theme”, which in different contexts can describe either a social event or a nutrition event. The solution to such kind of problems is to either split the “eat-meal” concept into two different concepts (e.g. “eat-ingest” and “eat-socialize”), or to split “theme” relation into two different relations (e.g. “ingestible-theme” and “social-theme”).

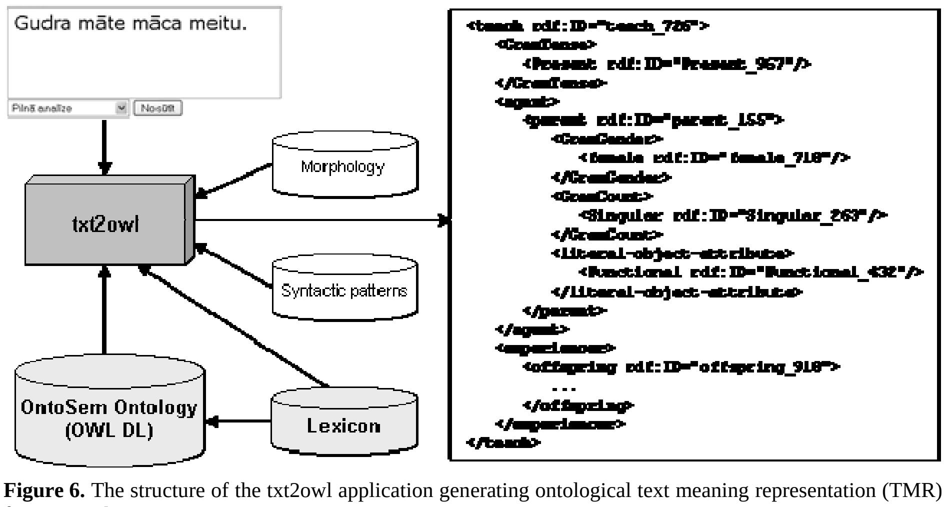
Finally, ontology might contain strange constructs, which are not necessarily formal errors, but nevertheless deserve to be identified as warnings, because they point to a likely error, or at least an unnecessary redundancy in the ontology. Such constructs cannot be found by a formal reasoner like RacerPro, but are easily identified by OWL DL ontology tests built into OWL-plugin of the Protégé ontology editor [5]. For example, OntoSem property “length” is found to contain redundant domain “literary-style”, not permitted by the super-property “size”:

```
(make-frame length
        (is-a (value (size)))
        (domain  (sem (physical-object literary-style))))

(make-frame size
        (is-a (value (scalar-physical-object-attribute)))
        (domain (sem (physical-object)))
```

4. Adaptation of the OntoSem lexical application framework

The original OntoSem ontology has been designed for use by the related ontology-driven lexical application for deep semantic analysis of natural language texts. By embarking on re-engineering the OntoSem ontology to OWL DL semantics, we also have re-implemented the related application, which we will call here txt2owl. Figure 6 shows the overall structure of the txt2owl application.



```
<teach rdf:ID="teach_726">
   <GramTense>
      <Present rdf:ID="Present_967"/>
   </GramTense>
   <agent>
      <parent rdf:ID="parent_155">
         <GramGender>
            <female rdf:ID="female_718"/>
         </GramGender>
         <GramCount>
            <Singular rdf:ID="Singular_263"/>
         </GramCount>
         <literal-object-attribute>
            <Functional rdf:ID="Functional_432"/>
         </literal-object-attribute>
      </parent>
   </agent>
   <experiencer>
      <offspring rdf:ID="offspring_918">
         ...
      </offspring>
   </experiencer>
</teach>
```

Figure 6. The structure of the txt2owl application generating ontological text meaning representation (TMR) for Latvian language sentences.

The main product of txt2owl application is conversion of the natural language sentences into their text meaning representation (TMR), which effectively is a list of OntoSem ontology individuals related by the corresponding ontology properties. Our txt2owl implementation accepts sentences in restricted Latvian, maps them through the Latvian morphology, syntax and lexicon databases into OntoSem ontology individuals and properties, and finally applies a specialized reasoner to validate the result according to the restrictions defined in OntoSem ontology. Txt2owl application is implemented in SWI-Prolog (which includes convenient libraries for OWL/RDF manipulation) and involves a lot of backtracking through the lexicon mappings in the OWL DL validation process. Although parsing of a single sentence into TMR takes just few seconds, for larger corpus processing the application is being ported also to the grid environment. The example in Figure 7 shows OWL syntax TMR created automatically for the sentence *"gudra māte māca meitu"* (*"clever mother teaches daughter"*).

```
<teach rdf:ID="teach_341">
    <GramTense>
          <Present rdf:ID="Present_373"/>
    </GramTense>
    <agent>
          <parent rdf:ID="parent_155">
                <GramGender>
                      <female rdf:ID="female_222"/>
                </GramGender>
                <GramCount>
                      <Singular rdf:ID="Singular_594"/>
                </GramCount>
                <literal-object-attribute>
                      <Clever rdf:ID="Clever_558"/>
                </literal-object-attribute>
          </parent>
    </agent>
    <experiencer>
          <offspring rdf:ID="offspring_964">
                <GramGender>
                      <female rdf:ID="female_691"/>
                </GramGender>
                <GramCount>
                      <Singular rdf:ID="Singular_562"/>
                </GramCount>
          </offspring>
    </experiencer>
</teach>
```

Figure 7. Text meaning representation (TMR) produced by txt2owl application according to OWL DL/OntoSem ontology from the Latvian sentence *"gudra māte māca meitas"*. Since the generated TMR conveys full meaning of the original sentence, it can be translated into English sentence *"clever mother teaches daughter"* with no knowledge of Latvian.

The described txt2owl application (slightly modifying it) can be exploited also for ontology testing. Apart from preventing logical bugs in the ontology, it is important to detect real-world inconsistencies as well. This can be achieved by checking ontology mapping of the test suites containing real-world examples. In case of a lexical ontology, the most appropriate source for test-case acquisition is the language itself — common assertions expressed in the form of natural language sentences.

Manual validation of large ontology is time-consuming task and might not ensure wide coverage. This is where the txt2owl implementation of a language-specific lexicon and word sense alignment to the ontology concepts[4], makes it possible to automate the test-case generation process illustrated below. Moreover, such approach could add more objectivity to the test data in contrast to merely manual approach that involves human subjectivity factor.

Candidates that suite well as comprehensive and reliable knowledge sources include, but are not limited to: balanced text corpora (provides also frequencies of utterance), common encyclopedias, or the dictionaries, which contain definitional phrases and selected typical examples assigned to word senses, usually given in a rather canonical form. Test-cases can be seen as graph patterns, which we are trying to map to the ontology. For instance, typical patterns that can be found in example phrases given for verb senses are:

EVENT—$^{\text{AGENT}}$—OBJECT (e.g. *“a boy runs”*)
EVENT—$^{\text{THEME}}$—OBJECT (e.g. *“to prepare food”*)
OBJECT—$^{\text{AGENT}}$—EVENT—$^{\text{THEME}}$—OBJECT (e.g. *“a student reads a book”*)

We will illustrate the steps required in test-case acquisition by a simplified definition of word *honey*: *“a sweet sticky fluid made by bees”*. For this example txt2owl application by means of morphological analysis, syntactic parsing and mapping of syntactic roles into semantic roles (properties), could produce a test-case (extended TMR including combinations of concept and property mappings in case of a syntactically/semantically ambiguous sentence) shown in Figure 8.

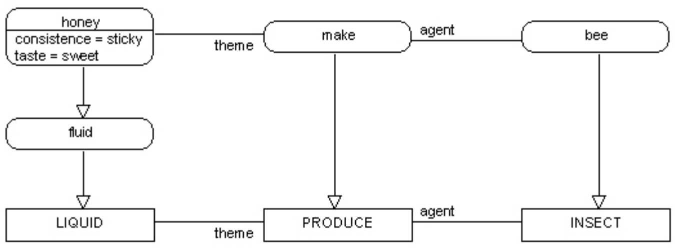


Figure 8. Test-case pattern for “produce” event, derived from the definition of *honey* in a dictionary. Ontological concepts (in upper case) are linked with the running words through the txt2owl lexicon.

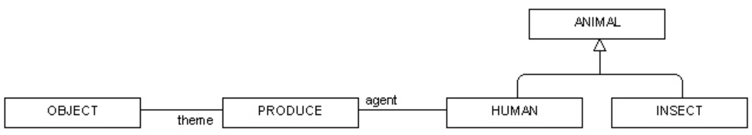


Figure 9. OntoSem ontology fragment relevant to test-case in Figure 8. Test fails — too restrictive range for property “agent” discovered.

[4] We have skipped description of the model of a language-specific lexicon as it is out of the scope of this paper. See [7, 8, 9] for more details.

It is easy to see that the original OntoSem ontology (relevant fragment shown in Figure 9) does not permit the test-case depicted in Figure 8: the range of “agent” property is restricted too narrowly to include only “human”. To correct this error and get this test-case accepted, ontology needs to be corrected by relaxing restriction on “agent” towards “animal”.

Conclusions

This paper describes the work in progress, as several aspects of the original OntoSem framework are not yet ported to txt2owl application, such as fact repository, co-reference resolution and procedural semantic routines. Nevertheless, the core functionality of TMR creation is completed and demonstrates the viability of the approach.

References

[1] Barzdins J., Barzdins G., Balodis R., Cerans K., Kalnins A., Opmanis M., Podnieks K. *Towards Semantic Latvia* // In: Seventh International Baltic Conference on Data Bases and Information Systems, DB&IS'2006 (submitted).

[2] Dean D., Schreiber G., Bechhofer S., van Harmelen F., Hendler J., Horrocks I., McGuiness D., Patel-Schneider P., Stein L. A. *OWL Web Ontology Language Reference*. W3C Recommendation, 2004. Available at URL: http://www.w3.org/TR/owl-ref [date of citation: 2006-03-20].

[3] Fellbaum C. (Ed.) *WordNet: An Electronic Lexical Database*. Cambridge: The MIT Press, 1998.

[4] Haarslev V., Möller R. *RACER System Description*. // In: Goré R., Leitsch A., Nipkow T. (Eds.) International Joint Conference on Automated Reasoning (IJCAR 2001), Siena. Springer-Verlag, 2001, pp. 701–705.

[5] Horridge M., Knublauch H., Rector A., Stevens R., Wroe C. *A Practical Guide to Building OWL Ontologies Using the Protégé -OWL Plugin and CO-ODE Tools*. The University of Manchester, 2004. Available at URL: http://www.co-ode.org/resources/tutorials/ProtegeOWLTutorial.pdf [date of citation: 2006-03-20].

[6] Horrocks I. FaCT++ web site. Available at URL: http://owl.man.ac.uk/factplusplus [date of citation: 2006-03-20].

[7] Java A., Finin T., Nirenburg S. *Integrating Language Understanding Agents Into the Semantic Web* // AAAI Press, 2005. Available at URL: http://ebiquity.umbc.edu/get/a/publication/221.pdf [date of citation: 2006-03-20].

[8] Nirenburg S., McShane M., Beale S. *Increasing Understanding: Interpreting Events of Change* // In: Proceedings of the OntoLex 2005 Workshop, 2005, pp 43-52.

[9] Nirenburg S., Raskin V. *Ontological Semantics*. Cambridge: The MIT Press, 2004.

[10] Nirenburg S., Raskin V. Version of OntoSem ontology. Available at URL: http://thoth.ilit.umbc.edu/CMSC-771/mikro-ontology.lisp [date of citation: 2006-03-20].

[11] Parsia B., Sirin E.. *Pellet: An OWL DL reasoner* // In: Möller R., Haarslev V. (Eds.) Proceedings of the International Workshop on Description Logics, 2004.

[12] Wang H., Horridge M., Rector A., Drummond N., Seidenberg J. *Debugging OWL-DL Ontologies: A Heuristic Approach*. // In: Gil Y. et al. (Eds.) International Semantic Web Conference (ISWC 2005), LNCS 3729, 2005, pp. 745–757.