STUDIA INFORMATICA 2011
Volume 32 Number 2A (96)

Paweł KAPŁAŃSKI
Gdansk University of Technology,
Faculty of Electronics, Telecommunications and Informatics

CONTROLLED ENGLISH INTERFACE FOR KNOWLEDGE BASES

Summary. This paper describes the approach to interface for OWL Knowledge Bases based on Controlled English that is transformed directly to OWL 2 expressions and back. Two possible modes enable usage of the interface to specify Terminology, World Description as well as Integrity Constrains. Knowledge Base then can be validated against the Integrity Constrains to enable its continuous validation.

Keywords: OWL, Controlled Natural Language, Description Logic, Knowledge Base

KONTROLOWANY JĘZYK ANGIELSKI, JAKO INTERFEJS BAZ WIEDZY

Streszczenie. W artykule przedstawiono interfejs dla baz wiedzy, stworzony na podstawie kontrolowanego języka angielskiego, który to jest przekształcany bezpośrednio do wyrażeń OWL 2 i z powrotem. Wskazano dwa tryby pracy, opierające się na tym interfejsie, umożliwiające zapisywanie terminologii bazy wiedzy oraz opisu świata, jak również na zapisie ograniczeń wewnętrznych bazy wiedzy. Taka baza wiedzy może być następnie automatycznie kontrolowana pod kątem ww. ograniczeń.

Słowa kluczowe: OWL, kontrolowany język naturalny, logika opisowa, baza wiedzy

1. Introduction

Knowledge Bases[1] (KB) naturally appear in almost every area of endeavor where computers are used intensively and knowledge management is required. Nowadays we observe the evolution process that brings KB from the human-readable form (where KB acts as an archive of searchable information) into the more-and-more computer-readable form that allows for automated deductive reasoning – semantic KB. To understand the knowledge model[2] stored in semantic KB one is required to have a background in the field of an artificial intelligence, knowledge representations and knowledge modeling. There is also required to know the supporting tools that are mostly organized around the graphical knowledge modeling languages. While without the support of formal methods it is almost impossible to trace and understand the knowledge model if KB is complex, it is still very hard to trace a formal knowledge model for an authority that is not familiar with a particular graphical knowledge representation language. In consequence, strategic decisions that are made by the authorities not familiar with semantic tools might reveal a lack of crucial information.

To overcome those limitations we propose to use Controlled English (CE) as a knowledge modeling language. Proposed CE is supported via Predictive Editor that will prohibit one to enter any sentence that is not grammatically or morphologically correct and will actively help the user during sentence writing.

2. Natural language as an interface for OWL Knowledge Bases

CE is a subset of Standard English with restricted grammar and vocabulary in order to reduce the ambiguity and complexity inherent in full English. In the last years Controlled English established itself in various application fields as powerful knowledge representation language that is readable by humans and processable by computers. CE texts can automatically be translated into and from description logic, specifically *SROIQ*, the basis of the semantic web language OWL2. Although OWL2 [6] is associated with the Semantic-Web [3], new applications appeared recently proving its usability also in other fields of interest.

The research for CE described in this paper was inspired by ACE_{OWL} [8], however grammar of invented language, was implemented using LARL(1) top-down parser generator [11] and equipped with additional features that are not accessible within ACE_{OWL}. Therefore,

[1] Knowledge Base is an integral component of knowledge management systems. It is a special kind of database used to optimize information collection, organization, and retrieval for an organization, or for the general public.

[2] Knowledge models are expressed in knowledge representation languages or data structures that enable the knowledge to be interpreted by software and to be stored in a database or data exchange file.

there exist sentences of invented language (called $LARL_{(1)}CE_{DL}$) that are not a valid expressions in ACE (and even in English), nevertheless the EBNF grammar (see fig. 2) of the $LARL_{(1)}CE_{DL}$ is designed to be as close as possible to Natural English and can be translated to OWL2 and back easily.

$LARL_{(1)}CE_{DL}$ language deals with concepts, roles and instances that can be represented by symbols in form of buzz-words[3]. Concept and role identifiers should start with small letter in opposite to instances and cannot be any of keywords. The morphology is applied if needed by separated dictionary based module. Additionally it is required that each sentence starts with upper-case letter and ends with 'full stop' sign. There are four general groups of sentences that are allowed:

1. Concept (possibly pseudo-modal) subsumption – represents all cases where there is a need to specify the fact (or constrain) about specific concept or instance (or expressions that evaluate to concept or instance) in form of subsumption e.g.: "Every cat is a mammal.", "Pawel has two legs." or "One cat (that is a brown-one) has red eyes"
2. Concept equivalence[4] – represents facts about concept or instance equivalences. "Every man and every male-human means-the-same."
3. Role (possibly complex) inclusion – specify the properties and relationships between roles in terms of expressiveness of *SROIQ* e.g.: "If X loves something that covers Y then X loves-cover-of Y."
4. Role equivalence – useful e.g.: in definition of inverse roles e.g.: "X is-type-of Y and Y has-type-that-is X means-the-same."

ALGORITHM Validation(terminology, world-description, integrity-constrains)

```
 1. Tell the Reasoner about terminology
 2. Tell the Reasoner about world-description

 3. IF(knowledge base is not consistent)
 4. THEN
 5.     show inconsistency
 6.     STOP
 7. ENDIF
 8. FOR(ic∈integrity-constrains)
 9.     FOR(instance∈LeftHandSide(ic))
10.       IF(instance∉RightHandSide(ic))
11.          show error or warning depending on modality flag
12.          STOP
13.       END
14. END
```

Fig. 1. The validation algorithm
Rys. 1. Algorytm walidujący

[3] Buzz words are a group of words connected with '-' sign.
[4] "A and B means-the-same." construction correspond to $A \equiv B$ expression.

```
<paragraph> ::= {<sentence>}
<sentence> ::= <subject>[<modalWord>]<objectRoleExpression>
             | 'if' 'X' <roleChain> 'Y' 'then' 'X' <role> 'Y' '.'
             | 'if' 'X' <roleChain> 'Y' 'then' 'Y' <role> 'X' '.'
             | <subject> 'and' <subject> 'means-the-same' '.'
             | 'X' <role> 'Y' 'and' 'X' <role> 'Y' 'means-the-same' '.'
             | 'X' <role> 'Y' 'and' 'Y' <role> 'X' 'means-the-same' '.'
<subject> ::= ('every'|'no') <single> | 'everything' [<that>] | <instance>| 'nothing'
<modalWord> ::= ['must'|'should']
<objectRoleExpression> ::= [{'not'}] (('is'|'be'|'are') <object>| <role> <objectRestriction>)
<roleChain> ::= <role> [{'something' 'that' <role>}]
<role> ::= <name> | ('is'|'be'|'are') <name> 'by'
<instance> ::= <bigName>
<object> ::= ['a'|'an'] <single> | 'something' [<that>] | <instance> | 'nothing'
<objectRestriction> ::= <object>
             | ('nothing-but'|<comparer><count>) (<single>|<instance>|'something' <that>)
             | 'none'
             | 'itself'
<single> ::= <name> [<that>] | 'thing' | 'things'
<that> ::= 'that' <objectRoleExpressionIntersectionUnion>
       | '(' 'that' <objectRoleExpressionIntersectionUnion> ')'
       | '(' 'that-is-one-of:' <instance> {',' <instance>} ')'
<objectRoleExpressionIntersection> ::= <objectRoleExpression> [{'and' <objectRoleExpression>}]
<objectRoleExpressionIntersectionUnion>::=<objectRoleExpressionIntersection>
             [{'or' <objectRoleExpressionIntersection>}]
<comparer> ::= ['at-most'|'at-least'|'less-than'|'more-than'|'different-than']
<count> ::= ('no'| 'single' |'two' |'three' |...|'ten') | {<digit>}
<name> ::= <smallLetter>{<digit>|<smallLetter>|<bigLetter>|'-'|'.'|'/'}
<bigName> ::= <bigLetter>{<digit>|<smallLetter>|<bigLetter>|'-'|'.'|'/'}
<digit> ::= '0'|'1'|'2'|'3'|'4'|'5'|'6'|'7'|'8'|'9'
<smallLetter> ::= 'a'|'b'|'c'|'d'|'e'|'f'|'g'|'h'|'i'|'j'|'k'|'l'|'m'|'n'|'o'|'p'|'q'|'r'|'s'|'t'|'u'|'v'|'w'|'x'|'y'|'z'
```

Fig. 2. EBNF Grammar of $LARL_{(1)}CE_{DL}$
Rys. 2. Gramatyka $LARL_{(1)}CE_{DL}$ w postaci EBNF

To enable expression of complex *SROIQ* expressions $LARL_{(1)}CE_{DL}$ grammar allows to use parentheses that can be nested if needed in form of (that <expression>)[5] e.g.: "Every human is something (that is a man or is a woman or is a hermaphrodite).". The pseudo-modal subsumption expressions use the "must" or "should" keywords e.g.: "Every child should have parents." Such pseudo-modal expressions are in-fact translated to the OWL expressions that are simply annotated with the modal-word. Grammar in this form was required to provide one common language for:

1. Terminology – conception of terms in the world (the global axioms and core taxonomy). This category includes generalized T-Box (set of expressions, role specifications and other global assumptions – possibly including nominal that in this case become a part of such metaontology)

[5] "A (that is B or is C)" expression corresponds to A⋘ (B⋙ C).

2. World Description – particular manifestations of Terminology. This category includes generalized A-Box (set of expressions about instances that are related to particular entity of analyzed problem).
3. Integrity Constrains (pseudo-modal expressions that can be validated by Reasoner)

While OWL2 lack the expressivity to describe modal logics there is a need to annotate the pseudo-modal expressions with additional tag: "must" or "should". It is then possible to build the KB validator by continuously applying Integrity Constrains on both World Description and Terminology using Reasoner[6] (see fig. 1).

3. Motivating example

Object-oriented program built over object-oriented design ontology[7] forms a World Description of particular object-oriented program[8]. The object oriented design ontology is a Terminology as it consists from general rules e.g.: polymorphism and encapsulation. Incorporating $LARL_{(1)}CE_{DL}$ as a language that is able to model object-oriented structures enables us to use one and the same system for storing Integrity Constrains, as well as the project and system architecture, which in turn ensures the logical cohesion of artifacts created in different stages of software development (see fig. 3).

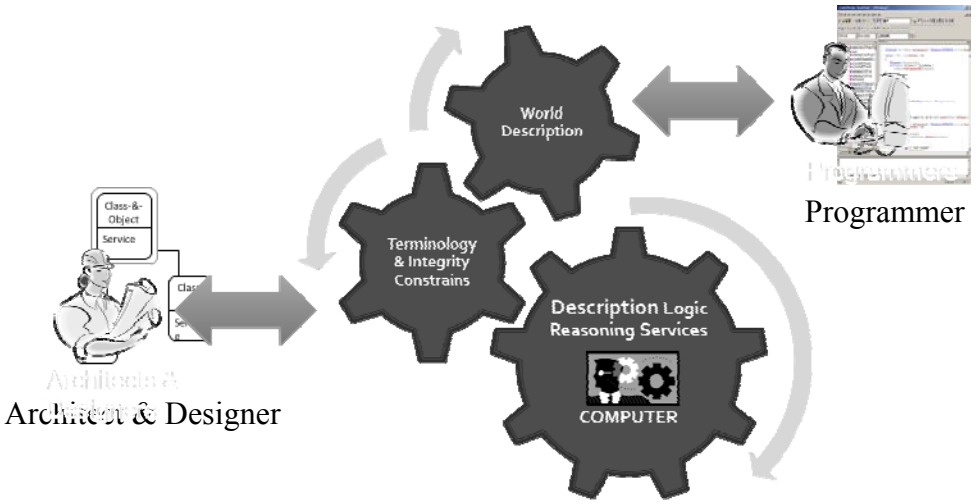

Fig. 3. Software Engineering with semantic KB
Rys. 3. Inżyniera Oprogramowania z wykorzystaniem semantycznej Bazy Wiedzy

[6] The proposed algorithm is insufficient but proves the idea. The real-life solution require Reasoner to have embedded integrity-constrains checking as a reasoning-task (e.g.: Pellet Integrity Constrain Validator http://clarkparsia.com/pellet/icv/).

[7] A hierarchical structure of design constructs.

[8] Program entities form a structure of design constructs (derived from object-oriented design ontology) using various relations that may exist amongst these constructs. This structure can be seen as a model of knowledge about those entities and therefore it forms ontology.

4. The toolchain and methodology

To prove the concept, the toolchain was implemented as a pair of complementary tools. To deal with KB in form of OWL2 files that include each other using <import> OWL statement, we developed Predictive Editor for $LARL_{(1)}CE_{DL}$ called "FluentEditor for OWL". It is fully functional editor for OWL2 with natural language interface (see fig. 4). The second main component is Integrity Constrains Validator that runs validation algorithm (see fig. 1) on stored KB. The main validation tasks are performed by HermiT [12] Reasoner. Both: Predictive Editor and Integrity Constrain Validator are implemented on top of OWLAPI [2].

Among the toolchain the work methodology was invented. By default we distinguish between two groups of users: Knowledge Engineers that specify the Terminology and map requirements to Integrity Constrains and Knowledge Providers that produce World Description; it can be also produced from already existing data sources. Both participants do complementary work using Predictive Editor and are continuously synchronized by Integrity Constrains Validator.

The toolchain and methodology were especially designed to be a part of novel software engineering methodology called OASE (Ontology Aided Software Engineering) designed for requirement engineering a part of that methodology among others e.g.: software architecture, design and program-code. While requirement engineers use Predictive Editor as a standalone application, other participants use Predictive Editor as a tool build-in IDE[9].

5. Evaluation

5.1. Advantages

Proposed methodology has a number of advantages for all involved groups of contributors. Input and output of KB is based on the same natural-looking language and the logic is hidden behind. Users do not need any training in formal logic formalisms nor the ICT support due to the Predictive Editor. Knowledge Providers are continuously checked by Validator if the entered knowledge fulfills the Integrity Constrains. Knowledge Engineers are provided with ability to explore the software using formal tools and they can continuously modify the Integrity Constrains while the KB implementation progress.

[9] Integrated Development Environment (e.g.: Visual Studio, Eclipse, Code Lite)

5.2. Limitations

First general limitation of CE is the need for Predictive Editor. It is very hard to create a correct CE sentence without such support. Second general limitations come from selection of Description Logic as underlying formalism. This limitation allows specifying and verifying only statements that are in the expressivity power of *SROIQ*. Future research need to be made to allow for expressivity in e.g.: Adjectives, Full Modal Logic, Temporal Logic etc. – however it is not obvious if then it will be possible to use LARL(1) or any other context-free grammar anymore.

6. Related Work

Very expressive CE languages like ACE [5] are already used to verbalize OWL. ACE can be translated into a non-decidable subset of first-order logic equipped with modal extensions. It includes subset called ACE_{OWL} [8] that can be translated into OWL 2. On the other hand most of OWL 2 can be translated into a subset ACE_{OWL}. Moreover, it was recently shown [9] that ACE_{OWL} is more natural for people than formal-looking CE_{DL} syntaxes (like Manchester [7] /Sydney [4] OWL Syntax).

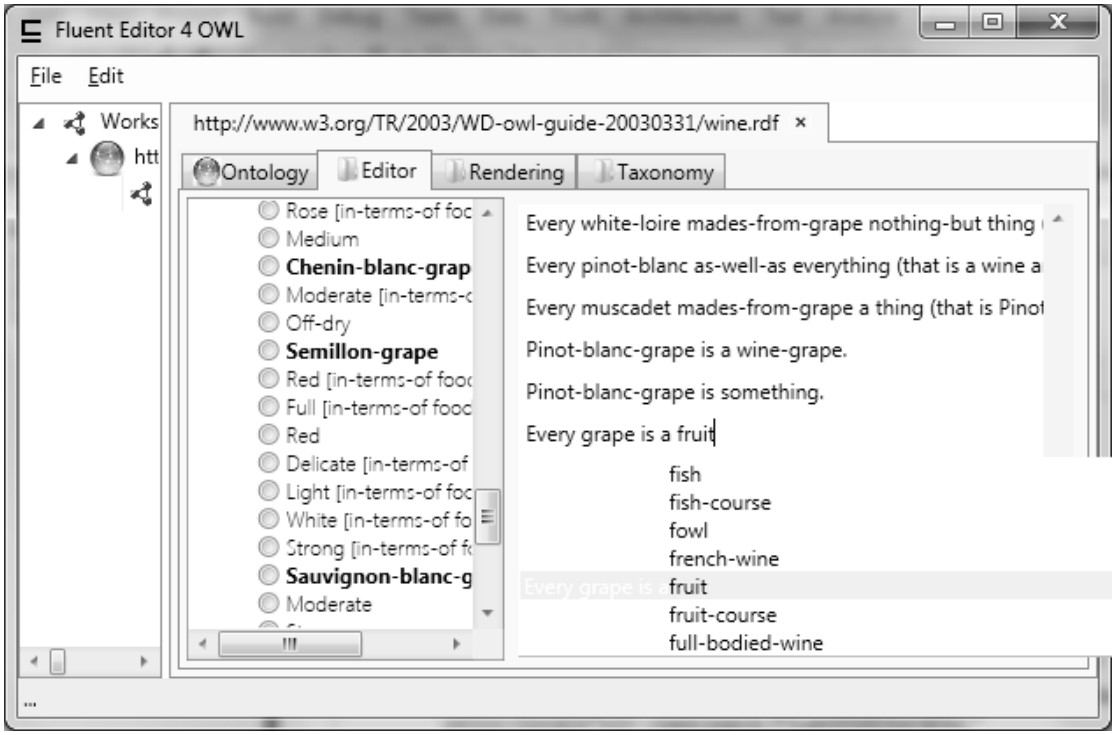

Fig. 4. Fluent Editor 4 OWL – the main window
Rys. 4. Fluent Editor 4 OWL – główne okno programu

There are dozens known ontology-editors and the number of them is growing from day to day. The most famous is Protégé [10] Most of them are difficult to understand and use by a typical domain user (they require knowledge about ontology engineering). Protégé allows editing ontology and inspect the inferred knowledge.

7. Future Work

Existing implementation is based on a file-system approach however, in the next step the approach with database instead of file-system is planned to be performed. It should be easy task as the solutions for it already exists [1].

Moving further, one can consider modern software-intensive systems as made of three kinds of participants: software, hardware and bioware[10]. While communication between software and hardware is realized by the computer-code, a programming language bridges software components with bioware. In our opinion it is worth to explore the abilities of $LARL_{(1)}CE_{DL}$ if and how it allows bioware to access (and to be accessed by[11]) the software. We expect then to know how to provide the smooth communication between computers and human beings that lack any prior knowledge of CE language.

8. Conclusion

This paper presents the results of research on application of Description Logic verbalized by Controlled English implemented with LARL(1) grammar. It was shown that invented language can be used as interface for OWL Knowledge Bases as it has the same expressivity. The invented methodology is a part of novel Ontology Aided Software Engineering methodology however successful implementation of predictive editor based on this approach opens a spectrum of potential applications (e.g.: in Crowd Sourcing).

[10] Bioware is a neologism for the human that can be seen as a part of computer system through the special API (see e.g.: Crowd Sourcing).

[11] Computer System can use the bioware as a system service (or just like any other system component). CE defines here a communication protocol for interaction with such a bio-service.

BIBLIOGRAPHY

1. Auer S., Ives Z. G.: Integrating ontologies and relational data. Technical Report MS-CIS-07-24, University of Pennsylvania Department of Computer and Information Science Technical, 11, 2007.
2. Bechhofer S., Volz R., Lord P. W.: Cooking the semantic web with the OWL API. In The Semantic Web – ISWC 2003: Second International Semantic Web Conference, Sanibel Island, FL, USA 2003, p. 659÷675.
3. Berners-Lee T., Hendler J., Lassila O.: The Semantic Web. Scientific American, 2001, 284(5), p. 34÷43.
4. Cregan A., Schwitter R., Meyer T.: Sydney owl syntax – towards a controlled natural language syntax for owl 1.1. In OWLED, 2007.
5. Fuchs N. E., Schwertel U., Schwitter R.: Attempto controlled english – not just another logic specification language. In LOPSTR '98: Proceedings of the 8th International Workshop on Logic Programming Synthesis and Transformation, London, UK 1990, Springer-Verlag, p. 1÷20.
6. Hitzler P., Krötzsch M., Parsia B., Patel-Schneider P. F., Rudolph S.: OWL 2 Web Ontology Language Primer. W3C Recommendation, World Wide Web Consortium, October 2009.
7. Horridge M., Drummond N., Goodwin J., Rector A. L., Stevens R., Wang H.: The manchester owl syntax. In OWLED, 2006.
8. Kaljurand K.: Attempto Controlled English as a Semantic Web Language. PhD thesis, Faculty of Mathematics and Computer Science, University of Tartu, 2007.
9. Kuhn T.: How to evaluate controlled natural languages. CoRR, abs/0907.1251, 2009.
10. Stanford School of Medicine. Protégé [http://protege.stanford.edu], 2010.
11. Rosenkrantz D. J., Stearns R. E.: Properties of deterministic top down grammars, [in:] STOC'69: Proceedings of the first annual ACM symposium on Theory of computing, New York, NY, USA 1969, p. 165÷180.
12. Shearer R., Motik B., Horrocks I.: HermiT: A Highly-Efficient OWL Reasoner, [in:] Ruttenberg A., Sattler U., and Dolbear C. (eds.): Proc. of the 5th Int. Workshop on OWL: Experiences and Directions (OWLED 2008 EU), Karlsruhe, Germany, October 26–27 2008.

Recenzent: Dr inż. Michał Świderski

Wpłynęło do Redakcji 31 stycznia 2011 r.

Omówienie

W artykule przedstawiono podejście do interfejsów baz wiedzy. Jest ono oparte na kontrolowanym języku angielskim, a gramatyka proponowanego języka jest gramatyką typu LARL. Zdania języka mogą być bezpośrednio transformowane do wyrażeń OWL 2, a także z powrotem. Dwa tryby pracy umożliwiają wkorzystanie interfejsu do opisu wiedzy zarówno o obszarze zainteresowań (terminologii), do zapisania samych danych (opisu świata), jak i do zapisu ograniczeń zapewniających integralność bazy wiedzy (ograniczeń wewnętrznych). Pozwalają one na ciągłe monitorowanie bazy wiedzy pod kątem spełniania przez nią narzuconych ograniczeń.

Metodologia jest częścią metody wytwarzania oprogramowania na podstawie ontologii (OASE – Ontology Aided Software Engeenering). Udana implementacja edytora predyktywnego (4), powstałego, jako jedno z narzędzi wspierających tę metodę, otwiera jednak szerokie spektrum potencjalnych zastosowań także w innych dziedzinach zainteresowań (np. w crowdsourcingu).

Address

Paweł KAPŁAŃSKI: Politechnika Gdańska, Wydział Elektroniki, Telekomunikacji i Informatyki, ul. Gabriela Narutowicza 11/12, 80-233 Gdańsk Wrzeszcz. pawel@kaplanski.pl.