When FrameNet meets a Controlled Natural Language

Guntis Bārzdiņš
University of Latvia
Riga, Latvia
`guntis.barzdins@mii.lu.lv`

Abstract

There are two approaches to the natural language processing – one is going in width to cover at shallow level (parsing, syntax) the rich linguistic variety found in the natural language, while another is going in depth (semantics, discourse structure) for a monosemous subset of natural language referred to as a controlled natural language (CNL). Today we are nowhere near to bridging the gap between the two approaches. In this presentation I argue that despite elusiveness of this goal, FrameNet might provide a sufficient insight into the deeper semantic layers of the natural language to envision a new kind of a rich CNL narrowing the gap with the true natural language. A blueprint for PAO, a procedural CNL of such new kind is discussed.

1 Introduction

Despite substantial achievements in the computational linguistics, such as rather reliable POS-tagging, syntax-tree parsing, word sense disambiguation, and statistical translation, in reality computational linguistics is still no where near to really understanding the natural language. All the mentioned techniques fail in certain situations and a human verification is always needed to achieve true accuracy - this is why accuracy measures such as precision and recall are commonly used to evaluate the computation linguistics methods. Missing background knowledge is often considered as the key reason for shortcomings of the machine-based systems.

On the other hand there are controlled natural languages (CNL) - their sole purpose is to go further in language semantics understanding than we are able for unrestricted natural language (Wiener, 2010). ACE (Fuchs, 2006), HALO project (Friedland, 2004), CYC NL subsystem (Lenat, 1995) and various OWL verbalizations (Schwitter, 2008) are among the best known CNLs. Although these CNLs are rooted in natural language, due to their narrow coverage limited by the underlying logical representation, these languages still largely resemble a programming language with strict grammar and monosemous lexicon. The main advantage of CNLs so far is that CNL text can be read and understood by an untrained person, while writing a correct CNL text is quite difficult and is similar to programming, where certain syntax and semantic constraints shall be strictly followed.

In this presentation will be discussed a possibility for constructing a more natural controlled language based on the ideas of FrameNet and situation semantics in general (Frame). The proposed approach incorporates the elements of traditional logic-based CNLs, but extends them with explicit procedural constructs derived from FrameNet. Since FrameNet itself covers a large portion of natural language constructs (Johansson, 2008), such approach bears a promise for a substantially more natural controlled language. A procedural extension of ACE-OWL (Kaljurand, 2007) controlled language (named PAO) will be used to illustrate the proposed approach (Gruzitis, 2010).

2 Defining the Background Knowledge

The key difference of PAO is that it adds support for procedural background knowledge through FrameNet like constructs besides the more traditional declarative background knowledge typically expressed through OWL ontologies. Based on the available background knowledge, PAO defines a translation from the controlled language input text into a combination of OWL and SPARQL statements.

In PAO background knowledge consists of two parts — declarative OWL ontologies (Fig.1)

and procedural templates (Fig.2). The purpose of ontologies is to define the concept hierarchies (OWL classes), their relationships (OWL properties) and restriction axioms (cardinality restrictions and others).
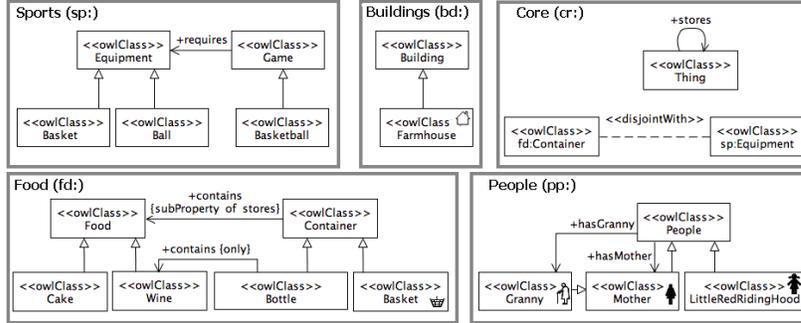

Fig 1: Declarative background knowledge ontologies

In Fig.1 OWL ontologies are visualized using UML-style OWLGrEd editor (Barzdins, 2010). Alternatively, ontologies may be defined verbally in CNL itself through ACE-OWL statements like:

Every Bottle is a Container.
Everything that contains something is a
  Container.
Everything that is contained by something is a
  Food.
If X contains Y then X stores Y.

The procedural background knowledge in Fig.2 provides a link between the action words (verbs) and their 'meaning' in SPARQL. The distinction between actions and properties is often neglected in CNLs, but in PAO they are strictly separated: in PAO action is a non-ontological SPARQL procedure, which creates/deletes OWL individuals or connects/disconnects them through the OWL properties. PAO action, unlike binary OWL properties, has no arity restriction — it can link any number of arguments as is typical for verb valencies in natural language. Syntactically a procedural template in PAO is a combination of elements inspired by FrameNet (Fillmore, 2003), Planning Domain Description Language (PDDL) (McDermott, 1998) and SPARQL. The procedural template itself corresponds to a FrameNet frame, the parameters section corresponds to FrameNet frame elements, and the lexical units section is a direct copy from FrameNet. Inclusion of precondition and effect sections in the procedural template is inspired by PDDL and has two-fold purpose: this is a compact representation of SELECT, INSERT, DELETE, MODIFY and WHERE patterns of the corresponding SPARQL statement and at the same time it preserves compatibility with PDDL for planning purposes. Elements of planning will become necessary in the final steps of PAO interpretation described later.

The ontologies and procedural templates shown in Fig.1 and Fig.2 are specifically crafted for the PAO example in the next section; for more realistic applications it would be necessary to create a much larger collection of ontologies and procedural templates covering the whole lexicon and domain-knowledge of interest.

Procedure: Residence
:parameters (?resident ?co-resident ?location)
 :precondition ()
 :effect (and(stores ?location ?resident)
     (stores ?location ?co_resident))
 :lexicalUnits (camp, inhabit, live, lodge, stay)

Procedure: Removing
 :parameters (?agent ?source ?theme)
 :precondition (stores ?source ?theme)
 :effect (and(stores ?agent ?theme)
     (not(stores ?source ?theme)))
 :lexicalUnits (confiscate, remove, take)

Procedure: Bringing
 :parameters (?agent ?goal ?theme)
 :precondition (and(stores ?agent ?theme)
     (stores ?a ?agent) (not(= ?a ?goal)))
 :effect (and(stores ?goal ?theme)
     (stores ?goal ?agent)
     (not(stores ?agent ?theme))
     (not(stores ?a ?agent)))
 :lexicalUnits (bring, carry, convey, drive)

Fig 2: Procedural templates of background knowledge

3 Example of PAO Text Processing

In PAO text has to be written in simple present tense to avoid complex event sequencing — the described events are assumed to be atomic and to occur sequentially as they are mentioned in the text. The following PAO input text will be used to illustrate the PAO processing stages:

"LittleRedRidingHood lives in a farmhouse with her mother. She takes a basket from the farmhouse and carries it to her granny."

The initial stage of PAO processing is anaphora resolution and paraphrasing of the input text into the sequence of elementary statements as shown in Fig.3.

A. Obj4 is a LittleRedRidingHood.
B. Obj4 **lives** in Obj8 with Obj11.
C. Obj8 is a farmhouse.
D. Obj4 hasMother Obj11.
E. Obj4 **takes** Obj15 from Obj8.
F. Obj15 is a food-basket.
G. Obj4 **carries** Obj15 to Obj25.
H. Obj4 hasGranny Obj25.

Fig. 3: Paraphrased PAO input text

Note that in the generated paraphrase in Fig.3 the statements A, C, D, F, and H are actually regular ACE-OWL factual statements about individuals and thus translate into regular OWL/RDF triples:

A:(<obj4> <rdf:type> <LittleRedRidingHood>)
C: (<obj8> <rdf:type> <Farmhouse>)
D: (<obj4> <hasMother> <obj11>)
    (<obj11> <rdf:type> <Mother>)
F: (<obj15> <rdf:type> <Basket>)
H: (<obj4> <hasGranny> <obj25>)
    (<obj25> <rdf:type> <Granny>)

Meanwhile the procedural statements B, E, and G do not belong to ACE-OWL and require a procedural template from the background knowledge in Fig.2 for their translation. The translation includes mapping of syntactic roles into procedural template parameters and converting the precondition and effect notation into equivalent SPARQL statements. PDDL-like planning stage is needed as well, because in the input text some obvious intermediate steps of action might often be omitted and they need to be filled-in by the planning to satisfy the procedural template preconditions — in our example for Little Red Riding Hood to be able to take a basket from the farmhouse, the basket had to be at the farmhouse in the first place.

The last analysis stage is to generate the RDF database content trace resulting from the execution of the above OWL/RDF and SPARQL translations — Fig.4 shows the resulting stepwise RDF database content trace.

Fig. 4: RDF content trace and its spatial visualization

The generated RDF database content trace is the final result of PAO text analysis — this trace is the actual discourse conveyed by the PAO input text. In the right column of Fig.4 the discourse is optionally visualized also as a sequence of graphic scenes — similarly to text-to-scene animation approach described in (Johansson, 2005). These visualizations can be generated automatically from the graphic icons provided for OWL classes in the background knowledge (Fig.1 actually includes the necessary icons); OWL properties are visualized as labeled arrows or alternatively as graphic inclusion for spatial properties like "stores". These visual scenes highlight the similarity of PAO analysis result to the dynamic scene likely imagined by a human reader incrementally reading the same input text.

4   Query Answering in PAO

The constructed RDF database trace in Fig.4 can further be used to answer queries about the input text, for example:

1. Who delivered a basket to a granny?
2. Did LittleRedRidingHood visit her granny?

3. Where initially was the basket?
4. When did the granny got the basket?

These queries can be answered through translating them into the appropriate SPARQL queries through techniques similar to those used to translate PAO paraphrase in Fig.3 earlier. The answers produced by such SPARQL queries on the RDF trace in Fig.4 would be:

1.   ?x = obj4
2.   yes
3.   ?x = obj8
4.   ?n = H

These very technical SPARQL answers can afterwards be rendered into more verbose answers:

1. LittleRedRidingHood [delivered a basket to granny].
2. Yes [, LittleRedRidingHood visited granny].
3. [Basket initially was] in the farmhouse.
4. In step H [, when LittleRedRidingHood brought the basket to granny].

Although we have not described the question answering process here in detail, these examples provide an overview of PAO potential for factual and temporal question answering over narrative input texts.

5   Conclusion

The described PAO controlled language is only a rather simple attempt to exploit the rich declarative and procedural background knowledge in a CNL to make it more natural through the inclusion of FrameNet like procedural semantics. The added expressivity allows for rich query answering about the provided input text. We are quite pleased to to been able to include ACE-OWL as a proper subset of PAO thus achieving a complementary integration of procedural and declarative approaches.

An obvious limitation of the presented PAO language is its treatment of time only as a linear sequence of events mentioned in the input text. A richer time conceptualization would be generally needed, including hypothetical, parallel and negated events to handle texts like "*Mother told LittleRedRidingHood to go directly to the granny's house and not to engage in conversations with strangers*".

The briefly mentioned optional visualization of PAO discourse is a promising area for further exploration — inversion of the mentioned visualization technique could potentially lead to a visual data acquisition in the form of CNL grounded in the same ontological and procedural background knowledge.

References

Lenat, D.1995. *Cyc: A Large-Scale Investment in Knowledge Infrastructure*. Communications of the ACM, 38:11, pp. 33--38

McDermott D. 1998. *PDDL — The Planning Domain Definition Language*. Technical report, Yale Center for Computational Vision and Control, http://www.cs.yale.edu/homes/dvm/

Fillmore, C.J., Johnson, C.R., Petruck, M.R.L. 2003. *Background to FrameNet*. International Journal of Lexicography, 16, pp. 235--250

Friedland, N., and Allen, P. 2004. *Project halo: Towards a digital aristotle*. In AI Magazine.

Johansson, R., Berglund, A., Danielsson, M., Nugues, P. 2005. *Automatic text-to-scene conversion in the traffic accident domain*. In: 19th International Joint Conference on Artificial Intelligence, pp. 1073--1078

Fuchs, N.E., Kaljurand, K., Schneider, G. 2006. *Atempto Controlled English Meets the Challenges of Knowledge Representation, Reasoning, Interoperability and User Interfaces*. In: 19th International FLAIRS Conference (2006)

Kaljurand, K., Fuchs, N.E. 2007. *Verbalizing OWL in Attempto Controlled English*. In: 3rd International OWLED Workshop

Schwitter, R., Kaljurand, K., Cregan, A., Dolbear, C., Hart, G. 2008. *A Comparison of three Controlled Natural Languages for OWL 1.1*. In: 4th International OWLED Workshop

Johansson, R., Nugues, P. 2008. *Comparing dependency and constituent syntax for frame-semantic analysis*. In: 6th International LREC Conference

Wyner A. et.al. 2010. *On Controlled Natural Languages: Properties and Prospects*. LNCS/LNAI5972, Springer, Heidelberg, 281-289

Gruzitis N. and Barzdins G. 2010: *Polysemy in Controlled Natural Language Texts*. In: CNL 2009 Workshop, LNCS/LNAI 5972, Springer, Heidelberg, 2010, pp. 102–120

Barzdins J., et.al. 2010. *OWLGrEd: a UML Style Graphical Editor for OWL* // Proceedings of ORES-2010, CEUR Workshop Proceedings, Vol-596