Proceedings of the Australasian Language Technology Workshop 2004
Macquarie University, Sydney, December 8th, 2004. Copyright, Australian Speech Science & Technology Association Inc.

Accepted after full paper review

Controlled Natural Language meets the Semantic Web

Rolf Schwitter and Marc Tilbrook
Centre for Language Technology
Macquarie University
Sydney, NSW 2109, Australia
{schwitt|marct}@ics.mq.edu.au

Abstract

In this paper we present PENG-D, a proposal for a controlled natural language that can be used for expressing knowledge about resources in the Semantic Web and for specifying ontologies in a human-readable way. After a brief overview of the main Semantic Web enabling technologies (and their deficiencies), we will show how statements and rules written in PENG-D are related to (a subset of) RDFS and OWL and how this knowledge can be translated into an expressive fragment of first-order logic. The resulting information can then be further processed by third-party reasoning services and queried in PENG-D.

1 Introduction

"The Semantic Web is an extension of the current web in which information is given well-defined meaning, better enabling computers and people to work in cooperation." (Berners-Lee et al., 2001)

This claim falls short, since the languages – based on RDF – that have been designed for making statements about Web resources and for specifying ontologies are not built for human consumption. These languages have been developed with machine processability and automatic information exchange in mind (Manola and Miller, 2004; Smith et al., 2004). They are painful to read, write and understand for non-specialists.

To overcome the disadvantages of machine-processable formal languages based on RDF and full natural languages that are far too expressive for the task at hand, we will introduce PENG-D, a controlled natural language that reconciles rigid formality and natural familiarity.

In a nutshell, a controlled natural language is a subset of a natural language that has been restricted with respect to its grammar and its lexicon. Grammatical restrictions result in less complex and less ambiguous sentences, while lexical restrictions reduce the size of the lexicon and the meaning of the lexical entries for a particular domain (Huijsen, 1998). Using a controlled language for knowledge representation, specification texts become easier to read and understand for humans, and easier to process for machines (Schwitter, 2004).

Traditionally, controlled natural languages have been classified into two major categories: human-oriented and machine-oriented controlled natural languages. Human-oriented controlled natural languages have been designed to improve the readability and understandability of technical documents, particularly for non-native speakers. An example are aircraft maintenance documents in the aerospace industry (AECMA, 2004). Machine-oriented controlled natural languages have been developed to ameliorate the quality of machine translation for technical documents (Mitamura, 1999) and for writing specification texts that can be easily translated into a formal language (Fuchs et al., 1999; Schwitter, 2002; Sowa, 2004).

Most of these machine-oriented controlled natural languages are defined by their translation into first-order logic that automatically restricts their expressivity to a small subset of constructions compared to full English.

Since it is very likely that the emerging Semantic Web will finally rely on a variant of description logic as knowledge representation formalism and since description logic is a decidable fragment of first-order logic (Grosof et al., 2003), a machine-oriented controlled natural language is required that is a compromise between expressive power, complexity and computability.

PENG-D is a proposal for such a machine-oriented controlled natural language that fulfils these requirements.

2 Semantic Web Enabling Technologies

The Semantic Web aims at making Web resources better accessible to automated agents by adding information (meta-data) that describes Web content in a machine-readable way. It is based on RDF, which relies on eXtensible Markup Language (XML) for syntax, Uniform Resource Identifiers (URIs) for naming, and on the RDF Vocabulary Description Language: RDF Schema (RDFS) for describing meaning and relationship of terms (Manola and Miller, 2004). RDF and RDFS form the lowest layers in the functional architecture of the envisioned Semantic Web. Web ontology languages and rule languages are expected to build the next two layers on top of RDFS to supply richer modelling primitives and reasoning support. Unfortunately, the relationships between RDFS and Web ontology languages are not clearly specified (Horrocks and Patel-Schneider, 2003).

2.1 RDF

RDF is a datamodel for representing meta-data about Web resources. The basic RDF model contains just the concept of a statement, and the concept of reification - making a statement about a statement. RDF is based on the idea of using URI references to identify the resources referred to in a RDF statement. RDF uses a particular terminology for talking about the various parts of a statement. The part that identifies what the statement is about is called the *subject*. The part that identifies the property of the subject that the statement specifies is called the *predicate* and the part that identifies the value of that property is called the *object*. In RDF, the English sentence

Nic is a human.

could be represented by a statement having three URI references:

```
http://www.example.org/about-nic#nic
http://www.w3.org/1999/02/22-rdf-syntax-ns#type
http://www.example.org/biology#Human
```

RDF models statements as nodes and arcs in a graph. In this notation a statement is represented as a node for the subject, a node for the object, and an arc for the predicate directed from the subject node to the object node. RDF/XML is a graph serialisation syntax and provides a machine-processable way to record and exchange graph-based RDF statements (Beckett, 2004). Here is an excerpt of an RDF/XML document:

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<rdf:RDF
  xmlns:rdf=
   "http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns="http://www.example.org/about-nic#"
  xmlns:bio="http://www.example.org/biology#"
  xmlns:con="http://www.example.org/contact#">

 <bio:Human rdf:about=#nic>
   <con:title>Dr.</con:title>
   <con:name>Nic Miller</con:name>
   <con:name>Nicolas Miller</con:name>
   <bio:pet>
     <bio:Labrador>
       <con:name>Rex</con:name>
     </bio:Labrador>
   </bio:pet>
 </bio:Human>

</rdf:RDF>
```

The *rdf:RDF* element is the root of the RDF document and defines the XML document to be an RDF document containing a reference to the *xmlns:rdf* namespace and to three other namespaces. Qualified names such as *bio:Human* or *con:title* are shorthands for full URI references whereas an URI reference with an empty prefix stands for the current document.

RDF/XML is the standard interchange format for RDF on the Semantic Web, but there exists a non-XML serialisation alternative called Notation3 (N3) that is generally accepted to be easier to use and is convertible to RDF/XML and vice versa (Berners-Lee, 2001). N3 was designed with human-readability in mind and was created as an experiment in optimising the "expression of data and logic in the same language" (Palmer, 2002). This optimisation can be seen as a first step towards the design of a controlled natural language. Below is the automatic translation of our RDF/XML excerpt into N3 using CWM, a forward chaining reasoner that supports data conversion (Berners-Lee, 2003):

```
@prefix : <http://www.example.org/biology#>.
@prefix con: <http://www.example.org/contact#>.

<#nic> a :Human;
  :pet [a :Labrador; con:name "Rex"];
  con:name "Nic Miller", "Nicolas Miller";
  con:title "Dr.".
```

Without doubt this representation is easier to read, to create and to understand by humans. However, there are still a couple of notational particularities which non-specialists have to master.

It would be much easier if a human could express this information in a well-defined subset of natural language, for example as follows:

Nic is a human who has Dr. as title.
Nic's name is Nic Miller and Nicolas Miller.
Nic's pet is a labrador that has the name Rex.

Obviously the writing process would need to be supported by an intelligent writing assistant that tells the user which constructions are admissible.

2.2 RDF Schema

So far, we have only addressed the syntax of RDF and have not considered the meaning of terms used in XML/RDF and N3. RDF itself provides no means for defining application-specific classes and properties. Instead, such classes and properties need to be described via RDF Schema (RDFS), an RDF-based vocabulary description language (Brickley and Guha, 2004). RDFS does not provide an application-specific vocabulary for classes and properties but it offers the facilities that are needed to describe such classes and properties. Basically, RDFS provides the means for constructing a type hierarchy for RDF.

2.2.1 Describing classes

A class in RDFS corresponds to the generic concept of a type or category. In N3 we can declare a class such as *Dog* in the following way:

```
:Dog a rdfs:Class.
```

This states that *Dog* is a type of *rdfs:Class*. If there is a hierarchical relationship between two classes, then this can be stated, for example, as follows:

```
:Labrador a rdfs:Class; rdfs:SubclassOf :Dog.
```

This says that *Labrador* is a *rdfs:Class* which is a subclass (*rdfs:SubclassOf*) of the class *Dog*.

2.2.2 Describing properties

In RDFS, properties are used to declare a relationship between two things and are described as instances of class *rdf:Property*:

```
:pet a rdfs:Property.
```

This states that *pet* is a type of *rdfs:Property*. RDFS also provides vocabulary for describing how properties and classes are intended to be used together. For example in

```
:pet rdfs:domain :Human; rdfs:range :Animal.
```

the *rdfs:domain* property is used to indicate that the values of the subject of the property *pet* are instances of the class *Human* and the values of the object of the same property are instances of the class *Animal*.

In some respect, RDFS is a very limited "knowledge representation" language with few modelling primitives and more expressive power is necessary to describe Web resources in sufficient detail. Apart from this expressive restriction, RDFS has a non-standard and non-fixed layer meta-modelling architecture, which makes some elements in the model have dual roles in the RDFS specification. This makes it extremely difficult to layer more expressive ontology and rule languages on top of RDFS (Pan and Horrocks, 2003).

2.3 OWL

The recognition of RDFS's limitations led to the development of OWL. The OWL language provides three increasingly expressive sublanguages (OWL Lite, OWL DL and OWL Full) that have been developed as a trade-off between expressivity and efficiency. OWL Lite supports users who need a classification hierarchy and simple constraint features combined with desirable computational complexity. OWL DL is closely related to description logic (DL) and supports users who want the maximum expressiveness without losing computational completeness and decidability. OWL Full is designed for users who require the meta-modelling facilities of RDFS with no computational guarantees (Smith et al., 2004).

OWL uses the same syntax as RDF (and RDFS) to represent ontologies and uses RDFS resources directly whenever the required functionality already exists in RDFS. For example, OWL uses *rdfs:subClassOf* to assert subclass relationships and specific classes and properties to extend RDFS functionality, e.g., the *equivalentClass* axiom is used to link a class description to another class description:

```
<owl:Class rdf:ID="Man">
  <rdfs:subClassOf rdf:resource="#Human" />
  <owl:equivalentClass rdf:resource=
    "#MaleAdult" />
</owl:Class>
```

While syntactic layering of the these languages satisfies the design requirements of the Semantic Web, the semantic layering is more problematic, since OWL is largely based on

DL which is a decidable fragment of first-order logic. The semantics of a DL is normally given by a standard first-order model theory, while the semantics of RDFS is given by a non-standard model theory, because of its meta-modelling facilities (Hayes, 2004). This incompatibility leads to serious problems when trying to layer first-order based languages on top of RDFS, since it is not clear how applications would be able to reason with these languages.

3 Description Logic Programs

Instead of relying on RDFS, conventional first-order logic (FOL) has been proposed as the semantic underpinning of the Semantic Web. From a theoretical point of view, it is well known that reasoning in FOL is undecidable. However, there are many decidable subsets of FOL that have been extensively studied and many reasoning systems have been developed for FOL and its sublanguages. Although this FOL-based approach is not immediately compatible with RDFS, it is compatible with a simplified version of RDFS – "the FOL subset of RDFS"(Horrocks and Patel-Schneider, 2003).

Recently, the fusion of DL ontologies and rule languages (Horn logic) has been studied and this effort resulted in the Description Logic Programs (DLP) paradigm (Grosof et al., 2003). DLP is defined as the expressive intersection of OWL Lite and Horn logic (without negation and without function symbols) and captures a significant part of OWL Lite. The interoperability between OWL Lite and Horn logic is expressed by a meaning preserving bidirectional translation of premises and inferences from the DLP fragment of OWL Lite to Horn logic, and vice versa from the DLP fragment of Horn logic to OWL Lite. In this paradigm, a mapping function translates for example the DL axiom

DL: $A \sqcap \exists R.C \sqsubseteq B \sqcap \forall P.D$

into Horn logic (HL) rules

HL: $\mathrm{b(X)} \leftarrow \mathrm{a(X), r(X,Y), c(X).}$
$\mathrm{d(Z)} \leftarrow \mathrm{a(X), r(X,Y), c(X), p(X,Z).}$

and vice versa (Grosof et al., 2003). Please note that the Horn rules here correspond to definite Horn clauses (with exactly one positive literal). All variables are universally quantified at the outer level and have scope over the entire clause and only fact-form conclusions (as in logic programs) are available.

4 PENG-D

PENG is a machine-oriented controlled natural language that has been developed to write specifications for knowledge representation (Schwitter, 2002; Schwitter et al., 2003; Schwitter, 2004). While PENG was designed for writing specifications that are first-order equivalent, the proposed language PENG-D has formal properties that are equivalent to DLP. Although PENG-D is weaker than PENG, it provides a clear computational pathway to layer more expressive constructions on top of it.

4.1 Architecture of PENG-D

The planned architecture of the PENG-D system looks similar to the PENG system but offers support for ontology construction. The PENG-D system consists of four main components: a look-ahead text editor, a controlled language (CL) processor, an ontology component and an inference engine.

4.2 The text editor

The text editor can be used either to construct ontologies (TBox mode) or to assert knowledge about a specific domain (ABox mode). The user does not need to learn the rules of the controlled language explicitly, since the writing process of the user is guided by the look-ahead text editor. The first thing that the user sees, for example, after opening the text editor in the ABox mode, are look-ahead categories:

[Proper Noun, Determiner, [There is a]]

After entering, for example, the name *Nic*, the look-ahead editor displays further look-ahead categories:

Nic *[Verb, Relative Pronoun, ['s]]*

The user can now type the word that belongs to one of these look-ahead categories directly into the editor or select it from a context menu. This context menu contains entries that have been derived from the ontology used. Namespace definitions are handled via the editor.

4.3 The grammar of PENG-D

The grammar of PENG-D describes how simple sentences are built and provides constructors to join simple sentences into complex sentences. Anaphoric relations between nominal constituents in sentences can be expressed via definite noun phrases, proper nouns, and variable names.

4.3.1 Simple sentences

Simple sentences are used for making statements in the ABox. Note that the result of these statements are always ground terms containing no variables. In a first approximation, we can describe the structure of simple sentences in the following way:

sentence	→	subject + predicate
subject	→	nominal head
subject	→	specifier {+ pre-nominal modifier} + nominal head {+ post-nominal modifier}
predicate	→	verbal head + complement

These rules need to be carefully restricted to be useful for our purpose:

Specifier. There are only two approved determiners that are available in the specifier position: the definite determiner *the* and alternatively the indefinite determiner *a* (with a "specific reading"):

***The** dog Rex ...*
***A** dog Rex ...*

Apart from that, a possessive construction such as *Nic's* can be used in the specifier position, for example:

***Nic's** dog Rex ...*

Once a specific instance has been introduced, we can refer to it:

*The dog Rex ... **The dog** ... **Rex** ...*

Pre-nominal modifier. A pre-nominal modifier can only consist of one single adjective in the positive form. Adjectives can be used to give additional information about a resource (and are interpreted intersectively):

*The **friendly** dog Rex ...*

Nominal head. The nominal head must be realised by a proper noun, a common noun or a relational noun. Common nouns and relational nouns (in subject position) always need a determiner. The structure of nouns can be simple or compound like in full English:

***Nicolas Miller** ...*
*The **hunting dog** Rex ...*
*The **dog** Rex of ...*

Post-nominal modifier. A post-nominal modifier can be realised in form of an *of*-construction, a finite relative clause or a named variable that starts with one of the four capital letter *X, Y, Z* or *W* (and is interpreted as a proper noun):

*The dog Rex **of Nic** ...*
*The dog Rex **which is a labrador** ...*
*The dog **X** ...*

Verbal head. Only transitive verbs, the construction *has ... as ...* and the copula *be* are available in PENG-D. Furthermore, verbs can only be used in the simple present tense, the active voice, the indicative mood, and the third person singular:

*Nic **likes** Rex.*
*Nic **has** Rex as friend.*
*Nic **is** ...*

Verbs are used for property assertions, apart from the copula *be* that can be used in constructions for class assertions and property assertions. Note that the possessive construction *Nic's dog Rex* and the *of*-construction *The dog Rex of Nic* result in the same translation as the sentence *Nic has Rex as dog.* The editor will displays a paraphrase for the possessive construction and for the *of*-construction.

Complement. The complement position can be realised by most of the syntactic structures that are approved for the subject position. Additionally, it allows for the prepositional construction *has ...* **as** *...* and for coordinated structures:

*Nic is married **to Sue**.*
*Nic has **Rex as dog**.*
*Nic has **Rex as dog and Tweety as bird**.*

There-sentences. A special case are "skolemized" *there*-sentences that have the following form:

***There** is a dog Rex that is happy.*

4.3.2 Compound sentences

Constructors join simple sentences into complex sentences. The main constructors that are used on this level are: *if*, *iff*, *and* and *or*, for example:

***If** X is a dog then X is an animal.*
*If X is a man then X is a male **and** X is an adult.*

Conditional (and biconditional) sentences are only available in the TBox for constructing ontologies. The nominal heads of such sentences can be realised by variables.

4.4 From PENG-D to HL and DL

We will now show how sentences of PENG-D are related to HL and DL statements, explain the function of these statements and explain which constraints apply. We do this by first looking at the RDFS statements that belong to the FOL subset of RDFS and then by considering those OWL Lite statements that extend the expressivity of the FOL subset of RDFS but are still contained within DLP, that is, the intersection of HL and DL. Thereafter, we will discuss how constructors in PENG-D sentences are reflected in HL and DL and which consequences the use of these constructors has.

4.4.1 RDFS statements

There are only two types of ABox statements that belong to the FOL subset of RFDS: *class assertions* and *property assertions*. The TBox statements that belong to this subset are: *subclass*, *subproperty*, *range* and *domain* statements.

Class assertions. Classes let us express membership information about individuals:

CL: *Nic is a human and Rex is a dog.*
HL: human(nic). dog(rex).
DL: nic : Human $\sqcap$ rex : Dog

Property assertions. Properties let us express specific facts about individuals:

CL: *Nic has the title Dr.*
HL: has_title(nic,doctor).
DL: <nic,doctor> : hasTitle

Subclass. Subclasses let us organise classes into a hierarchical taxonomy. Whenever the individual being an instance *X* of one class, this individual will necessarily be an instance of some other class:

CL: *If X is a labrador then X is a dog.*
HL: dog(X) $\leftarrow$ labrador(X).
DL: Labrador $\sqsubseteq$ Dog

Subproperty. Properties like classes can be arranged in an hierarchy. We can declare a property as a subproperty (specialisation) of an existing property:

CL: *If X has Y as dog then X has Y as animal.*
HL: has_animal(X,Y) $\leftarrow$ has_dog(X,Y).
DL: hasDog $\sqsubseteq$ hasAnimal

Domain of a property. We can restrict the domain of a property. In our example, the property *has as dog* has a domain of *human*. It relates instances of the class *human* to instances of *Y*:

CL: *If X has Y as dog then X is a human.*
HL: human(X) $\leftarrow$ has_dog(X,Y).
DL: $\top \sqsubseteq \forall$hasDog$^-$.Human

Range of a property. Similarly, we can restrict the range of a property. In our example, the property *has as dog* has a range of *animal* and ties instances of class *animal* to instances of *X*:

CL: *If X has Y as dog then Y is an animal.*
HL: animal(Y) $\leftarrow$ has_dog(X,Y).
DL: $\top \sqsubseteq \forall$hasDog.Animal

To restrict the domain and the range at the same time, we can write:

CL: *If X has Y as dog then X is a human and Y is an animal.*

4.4.2 OWL Lite statements

OWL Lite extends RDFS with additional TBox axioms. It adds explicit statements about class and property equivalence as well as the inverse of a property and transitivity.

Class equivalence. It is sometimes useful to indicate that a particular class in an ontology is equivalent to (i.e. has the same extension as) another class. In PENG-D, equivalence can be expressed via two conditional sentences but – as we will see below – this can be simplified using a biconditional operator:

CL: *If X is a man then X is a male adult.*
If X is a male adult then X is a man.
HL: male(X) $\leftarrow$ man(X).
adult(X) $\leftarrow$ man(X).
man(X) $\leftarrow$ male(X), adult(X).
DL: Man $\equiv$ Male $\sqcap$ Adult

The biconditional operator *Iff* is a shorthand to express in one sentence that two (possible complex) class descriptions have precisely the same instances, for example:

CL: *Iff X is a man then X is a male adult.*

Property equivalence. Similar to class equivalence, we can express that a particular property is equivalent to another property:

CL: *Iff X has Y as price then X costs Y.*
HL: cost(X,Y) $\leftarrow$ has_price(X,Y).
has_price(X,Y) $\leftarrow$ cost(X,Y).
DL: cost $\equiv$ hasPrice

Inverse of a property. If a property is the inverse of another property, then the variables in the first property switch their argument position in the second property, for example:

CL: *Iff X has Y as child then Y has X as parent.*
HL: has_parent(Y,X) $\leftarrow$ has_child(X,Y).
has_child(X,Y) $\leftarrow$ has_parent(Y,X).
DL: hasChild $\equiv$ hasParent$^-$

Transitivity of a property. Transitivity is a property of a binary relation such that if *X* and *Y* are related, and *Y* and *Z* are related, then it follows that *X* and *Z* are also related, for all *X*,

Y, and Z for which the relation may apply. For example the property *ancestor of* is transitive:

CL: *If X is an ancestor of Y and Y is an ancestor of Z then X is an ancestor of Z.*
HL: ancestor(X,Z) $\leftarrow$ ancestor(X,Y), ancestor(Y,Z).
DL: ancestor $^{+} \sqsubseteq$ ancestor

4.4.3 Constructors

The use of constructors is restricted in PENG-D, because they can not be expressed in HL or because they only be used in a restricted form in DL.

Conjunction. A conjunction of classes in the antecedent of a conditional sentence can be directly expressed in the body of a HL rule and creates no problem for DL:

CL: *If X is married and X is a woman then X is a wife.*
HL: wife(X) $\leftarrow$ married(X), woman(X).
DL: Married $\sqcap$ Woman $\sqsubseteq$ Wife

Conjunction in the consequent of a conditional sentence becomes a conjunction in the head of the corresponding HL rule, however this can be transformed into a pair of HL rules:

CL: *If X is a man then X is male and X is a person.*
HL: male(X) $\leftarrow$ man(X).
person(X) $\leftarrow$ man(X).
DL: Man $\sqsubseteq$ Male $\sqcap$ Person

Disjunction. Disjunction of classes in the antecedent of a conditional sentence becomes a disjunction in the body of the corresponding HL rule. This again can be transformed into a pair of HL rules:

CL: *If X is a woman or X is a man then X is a human.*
HL: human(X) $\leftarrow$ woman(X).
human(X) $\leftarrow$ man(X).
DL: Woman $\sqcup$ Man $\sqsubseteq$ Human

When a disjunction of classes occurs in the consequent of a conditional sentence, then it becomes a disjunction in the head of the HL rule, but this cannot be expressed within HL.

Universal restriction. In DL, the universal quantifier can only be used in restrictions of the form $\forall P.C$. Therefore, universal restriction can only be expressed in the following form in PENG-D:

CL: *If X is a women and X is married to Y then Y is a husband.*
HL: husband(X) $\leftarrow$ woman(X), married_to(X,Y).
DL: Woman $\sqsubseteq$ $\forall$marriedTo.Husband

Expressing universal restriction of the form in the consequent of a conditional sentence would require negation in the rule body of HL.

Existential restriction. In DL, the existential quantifier can only be used in existential restrictions of the form $\exists P.C$. When an existential restriction occurs in the antecedent of a conditional sentence, it becomes a conjunction in the body of HL:

CL: *If X is married to Y and Y is a husband then X is a wife.*
HL: wife(X) $\leftarrow$ married_to(X,Y), husband(Y).
DL: $\exists$marriedTo.Husband $\sqsubseteq$ Wife

However, if the existential restriction occurs in the consequent of a conditional sentence, then it becomes a conjunction in the head of corresponding HL rule with a variable that is existentially quantified. This cannot be handled in HL and would require transformation and skolemization in a logic program.

4.4.4 Beyond DLP

PENG-D is potentially a good starting point for language layering. More expressive language constructs could allow, for example, for expressing full existential quantification, instance equivalence, enumerating members of a class and cardinality constraints. While such descriptions cannot be directly expressed in DLP, many of them can be implemented in logic programming environments. Note that recursive HL rules such as transitivity need to be rewritten anyway for practical applications.

4.5 The inference engine

We are currently experimenting with various DL (and FOL) inference engines for question answering in PENG-D. Although not optimal, available FOL provers could provide reasoning services for more expressive DLs.

5 Conclusions

In this paper we referred to a number of deficiencies of RDFS as a "knowledge representation" language for the envisioned Semantic Web. Layering more complex ontology and rule languages on top of RDFS is not straightforward, because of its non-standard and non-fixed layer meta-modelling architecture. The relatively new DLP paradigm offers a promising first-order based alternative that enables ontological definitions to be combined with rules. To make such machine-processable information easily accessible for non-specialists, we proposed the use of PENG-D, a machine-oriented controlled natural language that has the same expressivity as DLP. We expect that PENG-D is easy to write for non-specialists with the help of a look-ahead text editor, easy to read in contrast to RDF-based notations, and easy to translated into a corresponding machine-processable format. In brief: PENG-D has the potential for complementing these more machine-oriented notations.

Acknowledgments

The research reported here is supported by the Australian Research Council (Discovery Project DP0449928). The authors would also like to thank two anonymous reviewers for the valuable comments on a previous version of this paper.

References

AECMA. 2004. The European Association of Aerospace Industries. *AECMA Simplified English*, AECMA Document PSC-85-16598. A Guide for the Preparation of Aircraft Maintenance Documentation in the International Aerospace Maintenance Language. Issue 2, January 15.

D. Beckett (ed). 2004. RDF/XML Syntax Specification (Revised). W3C Recommendation 10 February 2004. <http://www.w3.org/TR/rdf-syntax-grammar/>.

T. Berners-Lee, J. Hendler, Ora Lassila. 2001. The Semantic Web. In: Scientific American. May 17.

T. Berners-Lee. 2001. Notation 3. An RDF language for the Semantic Web. <http://www.w3.org/DesignIssues/Notation3.html>.

T. Berners-Lee. 2003. Processing your data using N3 and CWM. <http://www.w3.org/2000/10/swap/doc/Processing>.

D. Brickley, R. V. Guha. 2004. RDF Vocabulary Description Language 1.0: RDF Schema. W3C Recommendation 10 February 2004. <http://www.w3.org/TR/rdf-schema/>.

N. E. Fuchs, U. Schwertel, and R. Schwitter. 1999. Attempto Controlled English - Not Just Another Logic Specification Language. Lecture Notes in Computer Science 1559, Springer, pp. 1-20.

B. N. Grosof, I. Horrocks, R. Volz, S. Decker. 2003. Description logic programs: Combining logic programs with description logic. In: Proceedings of the Twelfth International World Wide Web Conference (WWW 2003), pp. 48-57.

P. Hayes. 2004. RDF Semantics. W3C Recommendation 10 February 2004. <http://www.w3.org/TR/2004/REC-rdf-mt-20040210/>.

I. Horrocks, P. F. Patel-Schneider. 2003. Three theses of representation in the semantic web. In: Proceedings of the Twelfth International World Wide Web Conference (WWW 2003), pp. 39-47.

W. O. Huijsen. 1998. Controlled Language - An Introduction. In: Proceedings of CLAW 1998. Pittsburgh, pp. 1-15.

F. Manola, E. Miller. 2004. RDF Primer. W3C Recommendation 10 February 2004. <http://www.w3.org/TR/rdf-primer/>.

T. Mitamura. 1999. Controlled Language for Multilingual Machine Translation. Invited paper. In: Proceedings of Machine Translation Summit VII. Singapore, September 13-17. <http://www.lti.cs.cmu.edu/Research/Kant/>.

S. Palmer. 2002. A Rough Guide to Notation3. <http://infomesh.net/2002/notation3/>.

J. Z. Pan, I. Horrocks. 2003. RDFS(FA): A DL-ised Sub-language of RDFS. : In Proceedings of the 2003 Description Logic Workshop (DL 2003), volume 81 of CEUR, pp. 95-102.

R. Schwitter. 2002. English as a Formal Specification Language. In: Proceedings of the Thirdteenth International Workshop on Database and Expert Systems Applications (DEXA 2002). Aix-en-Provence, pp. 228–232.

R. Schwitter, A. Ljungberg, and D. Hood. 2003. ECOLE: A Look-ahead Editor for a Controlled Language. In: Proceedings of EAMT-CLAW03, Controlled Language Translation. May 15-17, Dublin City University, pp. 141–150.

R. Schwitter. 2004. Representing Knowledge in Controlled Natrual Language: A Case Study. In: M. G. Negoita, R. J. Howlett, L. C. Jain (eds.), Proceedings KES2004, Part I, Springer LNAI 3213, pp. 711-717.

M. K. Smith, C. Welty, D. L. Mc Guinness. 2004. OWL Web Ontology Language. Guide. W3C Recommendation 10 February 2004. <http://www.w3.org/TR/2004/REC-owl-guide-20040210/>.

J. F. Sowa. 2004. Common Logic Controlled English Draft, 24 February 2004. <http://www.jfsowa.com/clce/specs.htm>.