Annotating Websites with Machine-processable Information in Controlled Natural Language

Rolf Schwitter and Marc Tilbrook

Centre for Language Technology
Macquarie University
Sydney, 2109 Australia

{schwitt|marct}@ics.mq.edu.au

Abstract

In this paper we present a user friendly approach to annotate websites with machine-processable information in controlled natural language. The controlled natural language serves as a high-level specification and knowledge representation language which allows human annotators to summarise individual web pages of a website and to express domain-specific ontological knowledge about that website in an unambiguous subset of English. The annotation process is backed up by an intelligent text editor which supports the writing process of the controlled natural language with the help of text- and menu-based predictive interface techniques. The text editor runs as a Java applet and is connected over the Internet to a controlled natural language processor and to a reasoning service (consisting of a theorem prover and a model builder). The controlled language processor translates the summaries of web pages and the ontological knowledge about a website into first-order predicate logic and the reasoning service combines this information into a set of micro theories for consistency and informativity checking as well as for question answering. Specification texts written in controlled natural language are both human-readable and machine-processable, and can be easily exported and distributed as web feeds.

Keywords: Knowledge Representation, Ontologies, Controlled Natural Languages, Theorem Proving, Model Building, Question Answering

1 Introduction

It has been argued that the current architecture for the Semantic Web, with its strong emphasis on RDF for syntactic and semantic compatibility, has severe problems when expressive Semantic Web (reasoning) languages are incorporated (Patel-Schneider 2005). An alternative approach is to use conventional first-order logic as the semantic underpinning for the Semantic Web. First-order logic is well understood, and well established subsets of first-order logic offer tradeoffs with respect to expressive power, complexity and computability (Horrocks and Patel-Schneider 2003). For example, the direct mapping of description logic-based ontology languages and Horn rule languages into subsets of first-order logic provides immediate semantic interoperability and builds the prerequisite for efficient reasoning (Grosof, Horrocks, Volz, and Decker 2003). Instead of relying on RDF, we suggest using a machine-oriented controlled natural language which is based on first-order logic as an interface language to the Semantic Web. To promote our approach, we will introduce a prototype application, which uses a controlled natural language to summarise web pages and to augment these summaries with domain-specific ontological knowledge. The result is a web feed which is easy to read by humans in contrast to other formal languages and as easy to process by a machine as other formal languages.

The rest of this paper is structured as follows: In Section 2, we briefly explain what controlled natural languages are, for what they can be used, and what kind of problems they can solve. In Section 3, we present PENG, a machine-oriented controlled natural language that looks seemingly informal, but can be unambiguously processed as a formal specification language. In Section 4, we introduce PENG Online, an intelligent text editor, which supports the writing of web page summaries and the specification of ontological knowledge in controlled natural language. In Section 5, we look at some details of the controlled natural language processor which is used by PENG Online to translate the controlled natural language into first-order predicate logic. In Section 6, we touch on the reasoning service which accomplishes several reasoning tasks. In Section 7, we show that a web feed specification in controlled natural language can directly be exported as an RSS feed, and finally in Section 8, we conclude and summarise the advantages of our approach.

2 Controlled Natural Languages

In general, a controlled natural language is a subset of a full natural language with explicit restrictions on the grammar, lexicon, and style. These restrictions usually have the form of writing rules and help to reduce (or even exclude) ambiguity and to cut down the complexity of full natural language. Traditionally, controlled natural languages fall into two categories: human-oriented and machine-oriented controlled natural languages. Human-oriented controlled natural languages (for example ASD Simplified Technical English (ASD 2005)) aim at improving text comprehension for human readers while machine-oriented controlled natural languages (for example Common Logic Controlled English (Sowa 2004)) focus on improving text processability for machines. An important difference between human-oriented and machine-

oriented controlled natural languages is that the writing rules for machine-oriented controlled natural languages must be precise and computationally tractable (Huijsen 1998). However, as a rule of thumb, simplification works in both ways: human-oriented controlled natural languages are also easier to process by machines and machine-oriented controlled natural languages are also easier to understand by humans compared to full natural language.

3 PENG (Processable ENGlish)

PENG is a machine-oriented controlled natural language designed for writing unambiguous and precise specification texts for knowledge representation (Schwitter 2002, Schwitter 2004, Schwitter 2005). PENG covers a strict subset of standard English and is precisely defined by a controlled grammar and a controlled lexicon. Specification texts written in PENG are incrementally parsed using a unification-based phrase structure grammar and then translated into first-order predicate logic via discourse representations structures (Kamp and Reyle 1993, Schwitter and Tilbrook 2004). In the general case, the result is a logic theory which can be checked for consistency and informativity as well as be used for question answering. In contrast to other machine-oriented controlled natural languages (Pullman 1996, Fuchs, Schwertel and Schwitter 1999, Holt, Klein and Grover 1999, Sowa 2004), the author of a PENG text ***does not*** need to know the grammatical restrictions of the language explicitly. The text editor of the PENG system dynamically enforces these restrictions while the text is written and displays the interpretation of a sentence in the form of a paraphrase in controlled natural language.

3.1 The Philosophy of PENG

The language PENG can be used as a high-level specification and knowledge representation language. Specification texts written in PENG look seemingly informal on the surface level, similar to full English, but in contrast to full English the language is designed to bring about the same precision and formality as a formal specification language. All sentences in PENG are correct English but only an unambiguous subset of English sentence structures and verb form-types are allowed in PENG. For example, PENG restricts the use of verb form-types in contrast to full natural language. In PENG verbs can only be used in their active voice, in their indicative mood, and in their simple present tense. Furthermore, modal verbs (such as *can*, *must*, *should*, etc.) and intensional verbs (such as *believe*, *seek*, *want*, etc.) are not allowed, since the underlying formal language does not immediately support modalities or intensional contexts. All seemingly ambiguous constructions in PENG are interpreted in a principled way and the interpretation is reflected in an unambiguous paraphrase. In summary: PENG has been carefully designed to be easy for humans to read and to write and easy for machines to process.

3.2 The Grammar of PENG

The grammar of PENG defines how words and their constituents combine to form simple sentences, complex sentences and questions. In our scenario simple and complex sentences are used to summarise web pages and to specify ontological information about a website. Questions are then used to interrogate various aspects of the resulting micro theories, for example to query the existence of a situation or to find specific entities which are part of a situation.

3.2.1 Simple Sentences

Simple sentences have a hierarchical structure consisting of words and constituents whereas each word is itself a constituent. Several constituents can be joined together in a controlled way to form simple PENG sentences. Constituents can be distinguished according to their function and their form. In the subsequent sentence:

1. *Bill Smith reboots the webserver on Monday.*

the constituent *Bill Smith* functions as the subject of the sentence and its form is a noun phrase. The constituent *reboots the web server on Monday* functions as the predicate of the sentence with the verb *reboots* as predicator while the constituent's form is a verb phrase. The functional dependents of the predicator within the verb phrase are of two kinds: complements and adjuncts. The constituent *the webserver* functions as a necessary complement of the predicator and its form is again a noun phrase. Finally, the constituent *on Monday* functions as an optional adjunct of the predicator (since the sentence is syntactically well-formed without this constituent) and its form is a prepositional phrase.

At the highest level, simple PENG sentences are composed of the following functional units:

subject + predicator + complements + adjuncts

Instantiations of this functional pattern are, for example, sentences such as:

2. *Bill Smith works.*
3. *Bill Smith maintains Apache.*
4. *Bill Smith works at Macquarie University.*
5. *Bill Smith is a diligent research programmer.*
6. *Bill Smith who is a diligent research programmer works at Macquarie University.*
7. *The supervisor of Bill owns a BMW.*
8. *The research programmer owns a Sony laptop.*

Sentence (2) shows the simplest possible structure of a PENG sentence consisting of a noun phrase (*Bill Smith*) in subject position and an intransitive verb (*works*) in predicator position. In sentence (3), a transitive verb (*maintains*) subcategorizes for a noun phrase (*Apache*) which occurs in complement position. In sentence (4), a prepositional phrase (*at Macquarie University*) occurs in adjunct position and modifies the verb (*works*) or better the underlying verbal event. In sentence (5), an adjective (*diligent*) occurs as a pre-nominal modifier of a complex noun (*research programmer*). In sentence (6), a relative sentence (*who is a diligent research programmer*) occurs as a post-nominal modifier of a proper noun (*Bill Smith*)

and constitute together a noun phrase. In sentence (7), the *of*-construction marks the noun (*supervisor*) as a relational noun with two arguments. In sentence (8), the noun phrase (*the research programmer*) in subject position is definite and the noun phrase (*a Sony laptop*) in complement position is indefinite. Definite noun phrases can be used to refer to previously introduced objects and indefinite noun phrases introduce new objects into the universe of discourse.

3.2.2 Complex Sentences

In PENG, complex sentences are built from simpler constituents and sentences with the help of a small number of constructors (coordinators, subordinators, quantifiers and negation markers). The subsequent sentences are examples of complex sentences:

9. *Bill Smith works at Macquarie University and maintains a webserver.*
10. *Bill Smith owns a Sony laptop or an Apple iPod.*
11. *Bill Smith is not a staff member.*
12. *No research programmer is a staff member.*
13. *Every research programmer owns a laptop.*
14. *If X is a research programmer then X is a programmer.*
15. *If X is a research programmer then X is not a staff member.*
16. *Every research programmer is a programmer.*
17. *If X buys Y then X acquires Y.*

In sentence (9), two verb phrases are coordinated by means of the conjunctive coordinator *and*. In sentence (10), two verb phrases are coordinated by means of the disjunctive coordinator *or*. In sentence (11), the negation marker *not* negates the entire verb phrase in complement position and in (12) the negation marker *no* negates the entire noun phrase in subject position. In sentence (13), the universal quantifier *every* is used to speak about all objects which belong to a specific class. In sentence (14), (15) and (17), the subordinator *if* introduces the antecedent of a conditional statement. Note that sentence (14) and sentence (16) are logical equivalent. The only difference between these two sentences is that the universally quantified variable (*X*) is made explicit in sentence (14) on the surface of the controlled natural language. This is a powerful mechanism to generate class hierarchies (see for example (14)) and property hierarchies (see for example (17)) in controlled natural language. As we will see later, this mechanism allows us to also – among other things – specify domain and range restrictions of properties.

3.2.3 Questions

In PENG, questions can be used to query the content of a specification text. Questions are systematically derived from simple and complex sentences to extract information from the constituents of these sentences and to guarantee wide coverage for question answering. Formally, *yes/no*-questions are built via subject-operator inversion and *wh*-questions are built by moving the interrogative word (e.g. *where, when, how*) to the initial position in the sentence, and where needed, by inserting the dummy *do* operator after the interrogative word. The following are examples of questions which can be used to interrogate a specification text written in PENG:

18. *Does Bill Smith work?*
19. *Who maintains a web server?*
20. *Where does Bill Smith work?*
21. *When does Bill reboot the webserver?*
22. *Is Bill Smith a programmer?*
23. *Is every research programmer a programmer?*
24. *Who maintains a web server and owns a laptop?*

Yes/no-questions such as (18), (22) and (23) allow us to check whether a specific situation is true or not and *wh*-questions such as (19), (20), (21) and (24) allow us to interrogate a specific aspect of a situation (for example finding a person who is involved in an event, a specific location or a point in time).

3.3 The Lexicon of PENG

The controlled lexicon of PENG consists of a base lexicon and a user lexicon. The base lexicon contains the most frequent content words of English (*proper nouns*, *common nouns*, *verbs*, *adjectives*, and *adverbs*) and predefined function words (*determiners*, *prepositions*, *coordinators*, *subordinators*, *negation* and *disambiguation markers*) which build the syntactic scaffolding of the controlled natural language. The base lexicon also contains illegal words (which cannot be processed by the PENG system). The user lexicon can be extended with domain-specific content words by the annotator while a text is written in controlled natural language.

4 PENG Online

PENG Online implements the web-based version of the PENG editor. The editor features built-in browser functionality for viewing web pages. It also provides a layout for expressing ontological knowledge about a website and for summarising the content of individual web pages which belong to that website. The editor can be used to create and update machine-processable descriptions of websites and to export them as web feeds in RSS format.

4.1 Architecture

PENG Online is based on a client-server architecture which consists of three main components: an intelligent text editor, a controlled natural language processor, and a reasoning service.

The web-based editor is implemented as a Java applet which runs in a web browser and communicates with a Prolog server via a socket interface. The Prolog server implements the controlled natural language processor and the reasoning service.

The controlled language processor incrementally translates specification texts into first-order predicate logic via discourse representation structures and generates predictive look-ahead information for the text editor as well as paraphrases for the input text.

The reasoning service makes use of SRI's Open Agent Architecture (OAA) where a facilitator coordinates a number of agents (Martin, Cheyer, and Moran, 1999, Cheyer and Martin, 2001). In our case, the reasoning interface agent fuses the summaries of web pages and the ontological knowledge about the website into a set of micro theories. These micro theories are sent to the facilitator which utilises a model builder agent and a theorem prover agent. These two reasoning agents complement each other and can check the micro theory for either consistency or informativity. These can also be utilised as a starting point for question answering.

The ontological knowledge about a website and the textual summaries of the individual web pages can be exported as a web feed in RSS format. Since the information is available in controlled natural language and fully human readable, any RSS feed aggregator can subscribe to such a web feed. However, the full benefit of having a machine-processable controlled natural language can only be brought into effect by a PENG-compliant tool which can reprocess these web feeds.

4.2 The PENG Editor

The PENG editor provides a standard mode and a web feed mode. The standard mode can be used to write normal specification texts in controlled natural language. The web feed mode is specially designed to annotate websites in controlled natural language. When the annotator selects the web feed mode, the text editor asks if the current user lexicon should be used for the new task or if a new user lexicon should be created. Once selected, the editor displays the interface of the feed mode as shown in Figure 1:

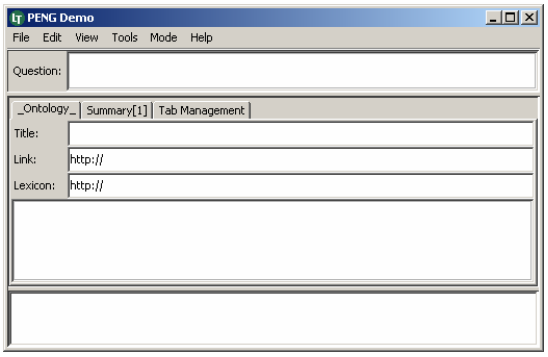

Figure 1: The PENG Editor in Web Feed Mode

This interface has a tabbed pane containing an ontology pane for the specification of the ontological knowledge about a website and one or more summary panes for the description of those individual web pages which are part of the website. Below the tabbed pane there is a message field for the system feedback and above the tabbed pane is a question field for asking questions about various aspects of a feed specification. The annotator can view a website using the built-in browser of the text editor which is available form the tools menu in the menu bar.

4.2.1 The Ontology Pane

In Figure 1, the ontology pane is active but still empty. This pane contains a title field for the name of the web feed, a link field for a URL to the HTML web page that corresponds to the channel, a lexicon field for a URL that points to the (exported) user lexicon of the controlled natural language, and a description field for the specification of the domain-specific ontological knowledge about a website. For example, the following complex PENG sentences express ontological knowledge about a website:

25. *If X is a research programmer then X is a programmer.*
26. *If X is a research programmer then X is not a staff member.*
27. *If X is a staff member then X is not a research programmer.*
28. *If X maintains Y then X is a programmer and Y is a webserver.*
29. *If X maintains Y then X looks after Y.*

Sentence (25) specifies a hierarchical class relationship between the subclass *research programmer* and the superclass *programmer*. The two sentences (26) and (27) specify that the two classes *research programmer* and *staff member* are disjoint. In Sentence (28), the verb (= property) is restricted in its domain by the class *programmer* and in its range by the class *webserver*. That means that only individuals that belong to the class *programmer* can occur in the subject position and only individuals that belong to the class *webserver* can occur in the complement position. Finally, in sentence (29), a hierarchical property relationship between the transitive verb *maintains* and the prepositional verb *looks after* is specified. Please note that all these sentences fall under the description logic subset of the controlled natural language (for details see Schwitter and Tilbrook 2006).

4.2.2 The Summary Pane

In Figure 2, the summary pane is active. This pane contains a title field for the name of a web page, a link field for the URL which points to the original web page and a description field for the summary of a web page in controlled natural language:

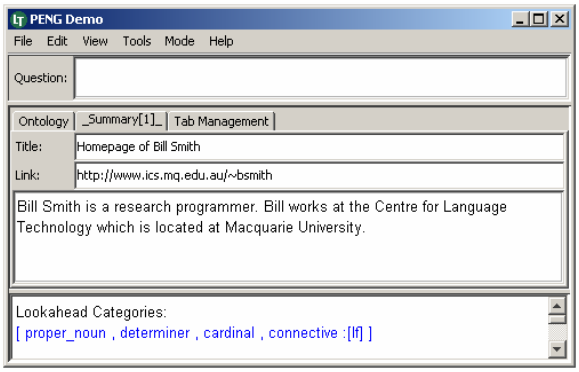

Figure 2: The Summary Pane

As this example illustrates, the annotator already added the title of the web page to the title field and the URL of the original web page occurs in the link field. The description field so far contains the following two sentences:

30. *Bill Smith is a research programmer.*
31. *Bill works at the Center for Language Technology which is located at Macquarie University.*

These sentences describe parts of the original web page (see Figure 3) that is currently open for annotating in the browser. Note that the message field in Figure 2 contains look-ahead information that informs the annotator about how the current specification text can be continued (for details see Section 4.2.3).

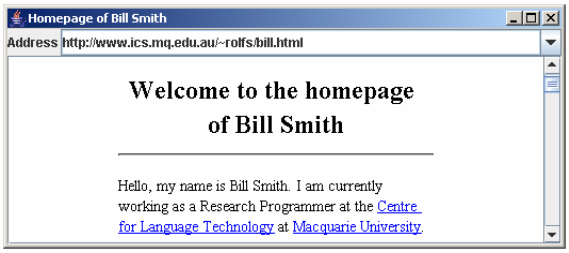

Figure 3: Excerpt of the Original Web Page

Note that not all of the information in the original web page can be represented in controlled natural language. The idea is to produce a machine-processable summary of a web page that can be easily read by humans and efficiently processed by a machine. This requires a careful tradeoff between expressiveness and processability of the controlled natural language.

4.2.3 Writing in PENG

The form of the input to the description field of both the ontology pane and the summary pane is restricted by the language processor of PENG. The language processor generates look-ahead information for each word form that the annotator enters while the specification text is written. This look-ahead information consists of syntactic categories which predict what kind of input can follow the current word form. The look-ahead categories are implemented as hypertext links. By clicking on a look-ahead category the author is able to access help information. The author composes a sentence either by typing the word forms which fall under the look-ahead categories or by selecting word forms from a cascade of menus (Schwitter, Ljungberg and Hood 2003, Thompson, Pazandak and Tennant 2005).

Please note that the look-ahead categories are generated on the fly and use linguistic information produced by the incremental chart parser of the controlled language processor. The processing of these look-ahead categories does not slow down the author significantly while typing the text and happens in near real-time (ca. 140 milliseconds on average per word form).

The look-ahead categories in Figure 2 indicate that the author can continue the specification text, for example, using a proper noun as in (32), a determiner as in (33), a cardinal number as in (34), or a specific subordinator as in (35):

32. … *at Macquarie University.* ***Bill*** …
33. … *at Macquarie University.* ***The*** …
34. … *at Macquarie University.* ***Two*** …
35. … *at Macquarie University.* ***If*** …

Instead of typing an approved word form into the description field of the editor, the author can alternatively select a word form from the currently active look-ahead categories via the context menu as Figure 4 illustrates:

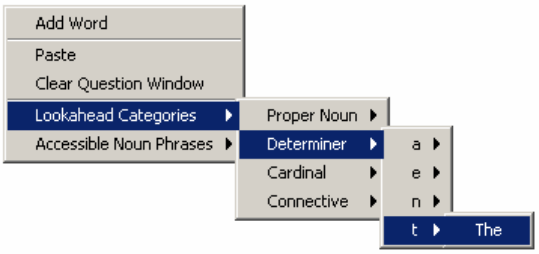

Figure 4: Active Look-ahead Categories

Once such a word form has been selected, it will be immediately inserted into the text at the current cursor position and the processing of the text is automatically resumed. Not only can approved word forms be inserted in this way, but also all noun phrases which are accessible in the specification text. Accessible noun phrases occur in the context menu and can be selected from there. Figure 5 shows that after the processing of sentence (30) and (31) the following three noun phrases are available in the context menu:

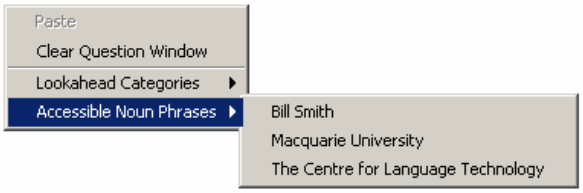

Figure 5: Accessible Noun Phrases

Please note that the noun phrase *a research programmer* is not accessible here, since it forms a property together with the copulative verb *be* and cannot be referred to by a definite noun phrase.

4.2.4 The Message Field

The message field displays a paraphrase for each sentence and clarifies the interpretation of the input – if this option is selected. The paraphrase indicates, for example, if synonyms or anaphoric expressions have been used in the text. Let us assume that the author added the following two sentences to the description field:

36. *Bill has a homepage.*
37. *The page contains a picture of Bill.*

And let us further assume that the noun *page* has previously been defined as a synonym of its main form *homepage* in the user lexicon. After processing this informa-

tion, the paraphrase in the message field will indicate – as Figure 6 illustrates – that the noun phrase *the homepage* and the proper noun *Bill* are two anaphoric expressions which have been previously introduced in the text and that the synonym *page* has been replaced, respectively normalized, by its main form *homepage*.

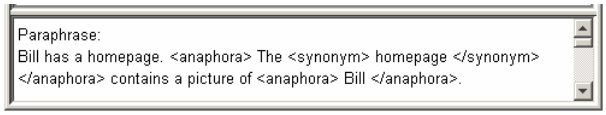

Figure 6: Paraphrase in Controlled Natural Language

Additionally, the message field displays the syntax tree for the last input sentence, the actual discourse representation structure for the entire text and its representation in first-order predicate logic. Furthermore, the message field shows the output of the reasoning engine (the proof or the model) and the specific result (for example the answer to a question). Not all of this information is relevant for the annotator and parts of this information can therefore be selectively removed.

4.2.5 The Question Field

The purpose of the question field is to interrogate a web feed in controlled natural language. As Figure 7 illustrates, the question field uses the same kind of look-ahead mechanism to guide the writing process as the description fields of the ontology and the summary pane.

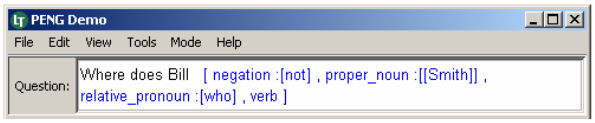

Figure 7: Question Field with Look-ahead Categories

Once a question is completely formulated, it is translated into first-order predicate logic via discourse representation structures (similar to simple and complex sentences), combined with the micro theories, and answered with the help of PENG's reasoning service.

4.2.6 The Lexical Editor

Part of the text editor is a lexical editor for adding user-specific content words. If the author enters a content word (i.e. *proper noun*, *common noun*, *verb*, *adjective* or *adverb*) into the text editor which is not yet available in the lexicon and is not in the list of illegal words, then this content word needs to be added to the user lexicon of the PENG system. The interface to the lexical editor has been designed in such a way that only minimal linguistic knowledge is required by the author to add a new content word to the lexicon. As soon as a new content word is available in the lexicon, the parsing process is resumed. User-defined content words can also be deleted from the user lexicon, but the author cannot delete words in the base lexicon of the PENG system which contains the most frequent 3000 words of English as well as all predefined function words. Note that the existing user lexicon (or a new user lexicon) is exported once the web feed is complete.

5 The Controlled Language Processor

When the author types a word form into the text editor, this word form is immediately sent to the incremental chart parser of the controlled language processor. The chart parser uses a unification-based phrase structure grammar as syntactic scaffolding (Schwitter 2003, Schwitter and Tilbrook 2004). As Figure 8 shows, the phrase structure rules of the grammar are highly parameterised.

```
s([ coord:no, drs:D, para:P1-P4, tree:[s,T1,T2],
    gap:G, styp:decl, snum:N, sana:M1-M3 ])
    -->
    n3([ coord:_, arg:I, spec:Q, ana:A, drs:D,
         sco:S, para:P1-P2, tree:T1, gap:n3:[]-[],
         styp:decl, snum:N, sana:M1-M2 ]),
    v3([ coord:_, vform:fin, arg:I, drs:S, para:P2-P3,
         tree:T2, gap:G, styp:decl, snum:N, sana:M2-M3 ]),
    pm([ cat:pm, para:P3-P4, snum:N ]).
```

Figure 8: A Phrase Structure Rule

The beauty of this approach is that it allows us to deal with syntactic, semantic and pragmatic information concurrently and in the same logic-based framework. The grammar currently consists of about 150 such phrase structure rules. During parsing the incremental chart parser generates a chart which can be used to harvest the look-ahead information for the text editor. The other important information in the chart is the discourse representation structure which represents the meaning of the text. For example, the two sentences (36) and (37) repeated here as (38) and (39)

38. *Bill Smith has a homepage.*
39. *The page contains a picture of Bill Smith.*

result in the following (simplified) discourse representation structure:

```
[A,B,C,D,E]
obj([bill,smith],A),
pred(C,[have],A,B),
obj([homepage],B),
pred(E,[contain],A,D),
obj([picture],D,A).
```

Figure 9: Simplified Discourse Representation Structure

whereas the variables `A`, `B`, `C`, and `D` represent discourse referents and the predicates conditions which hold for these discourse referents. Discourse representation theory (Kamp and Reyle 1993) allows us to deal in an elegant way with anaphoric references between sentences. Such discourse representation structures can be translated in linear time into a set of first-order logic formulas. These first-order logic formulas can then be further processed by the reasoning service of the PENG system as we will describe in the next section.

6 The Reasoning Service

One possible setting of the reasoning service is to use the theorem prover Otter (McCune 2003a) in combination with the model builder Mace4 (McCune 2003b) for consistency and informativity checking as well as for question answering (Bos 2003, Blackburn and Bos 2003, Blackburn and Bos 2005). Another interesting option we

are currently exploring but that we will not further discuss here is to use Satchmo instead of Otter and Mace4 as reasoning service (see Manthey and Bry 1988 and in particular Fuchs and Schwertel 2003 for a discussion).

The idea of using a theorem prover and a model builder in combination has been explored for other (natural) language processing tasks, for example for solving logical puzzles (Schwitter 2002, Lev, MacCartney, Manning, and Levy, 2004) and as a spoken language interface to a robot and in an automated home environment (Bos 2006).

6.1 Otter and Mace

Otter is an automated theorem prover for first-order logic with equality that searches for a refutation of a set of formulas and is designed to detect inconsistency (or unsatisfiability) of a theory. Mace4 is a model builder that searches for finite models of first-order formulas for a given domain size and its task is to check for satisfiability of a theory.

Otter and Mace4 can work on the same problem at the same time and complement each other. If Otter can find a proof for the negation of a set of formulas, then Mace4 has to do an exhaustive search that does potentially not terminate. In this case, Otter can inform Mace4 to stop searching for a model as soon as it found a proof. In a similar way, if Mace4 can build a finite model for the formulas, then Otter has to do an exhaustive search that does potentially not terminate. In this case, Mace4 can inform Otter to stop searching for a proof. Of course, the problem of detecting whether a set of first-order formulas is valid is not decidable and therefore we have to assign a time limit on the search for both Otter and Mace4.

The input to Otter and Mace4 can be specified with first-order formulas or first-order clauses or a combination of both. If the input consists of non-clausal first-order formulas, then the input is immediately further translated into first-order clauses involving negation normal form conversion, skolemisation, quantifier operations, and conjunctive normal form conversion. In contrast to Mace4, Otter has an interactive and an autonomous mode for selecting search strategies and provides more options to control the processing, but both accept similar input files.

6.2 Otter and Mace in PENG

The PENG system translates the discourse representation structures (which have been derived from the ontological knowledge), the summaries, and the questions, into first-order formulas. These formulas in turn are combined in various ways depending on the reasoning task. In PENG, we distinguish three reasoning tasks: consistency checking, informativity checking and question answering. Each task requires a specific preparation of the formulas which results in a micro theory to be processed by Otter and Mace4.

In our case, Otter runs in the autonomous mode and takes a micro theory as input, translates the input into clauses, scans the clauses and automatically decides on inference rules and a search strategy. Mace4 takes the micro theory, translates the input first into clauses and then into an equivalent propositional problem which is then given to a satisfiability procedure.

Before we discuss the various reasoning tasks in more detail, let us assume that Φ is a set of first-order formulas derived from the text in controlled natural language which summarises a web page; X is a set of first-order formulas derived from the text in controlled natural language which describes the ontological knowledge about a website; Ψ is a first-order formula derived from a new sentence; and δ is a first-order formula derived from a question stated in controlled natural language, and finally A is an answer literal. Answer literals record instantiations of variables during Otter's search for a refutation proof and can be used to answer *wh*-questions.

6.2.1 Consistency Checking

A micro theory $(X \wedge \Phi)$ is consistent if and only if all formulas can be satisfied together in some model with the same variable assignment. In the case of Otter we need to find out if $\neg(X \wedge \Phi)$ is valid and in the case of Mace4 we need to find out if $(X \wedge \Phi)$ is satisfiable.

If we give the negation of the micro theory $\neg(X \wedge \Phi)$ to Otter (thus we give it $\neg\neg(X \wedge \Phi)$) and it finds a proof for this input, then we know that $(X \wedge \Phi)$ is not consistent. If a micro theory is not consistent, then a theorem prover like Otter will always succeed in finding a proof.

If we give the micro theory $(X \wedge \Phi)$ to Mace4 and it successfully builds a finite model for this input, then we know that $(X \wedge \Phi)$ must be satisfiable (= consistent). If a micro theory is consistent and satisfiable on a finite model, then a model builder like Mace4 will always succeed in building a model.

6.2.2 Informativity Checking

A new formula Ψ is informative with respect to a context $(X \wedge \Phi)$ if and only if it is not a logical consequence of this context (or not satisfiable in all models).

If we give the negation of the micro theory $(X \wedge \Phi \rightarrow \Psi)$ to Otter and it finds a proof for this input, then we know that Ψ is not informative. If a new formula is not informative, then a theorem prover like Otter will always succeed in finding a proof.

If we give the micro theory $(X \wedge \Phi \wedge \neg\Psi)$ to Mace4 and it builds a finite model, then we know that Ψ is informative. If a new formula is informative, then a model builder like Mace4 will always succeed in building a model.

6.2.3 Question Answering

The simplest type of questions are *yes/no*-questions which do not contain free variables. However, *wh*-questions contain free variables which need to be bound to specific values during a proof. In order to accomplish this, the translation of interrogative words in the case of Otter results in answer literals which can be used to record instantiations of variables during a search for refutation. Mace4 does not provide such a mechanism, since

Mace4 is in fact a model builder and not a model checker which could tell us wether the model satisfies a query or not. However, Mace4 builds minimal models which are not redundant and answers to questions can be looked up immediately in the model once such a model exists.

If we give the micro theory $\neg\delta \wedge (X \wedge \Phi)$ to Otter and it finds a proof for this input, then we know that δ results in a positive answer to a *yes/no*-question.

If we give the micro theory $\neg(\delta \wedge A) \wedge (X \wedge \Phi)$ to Otter and it finds a proof for this input, then we know that the variable bindings in the answer literal A are results for a *wh*-question.

If we give the micro theory $(X \wedge \Phi)$ to Mace4 and it builds a finite model, then we can start searching for answers to the question δ in this model. However, this process requires a simple transformation of the model which Mace4 generates into a model which corresponds to the formal signature of the question.

6.2.4 An Example

We will now illustrate the reasoning abilities of the PENG system by a few examples. Let us assume that Φ_1 represents the two sentences:

- *Bill Smith is a research programmer.*
- *Bill works at Macquarie University.*

Φ_2 represents the two sentences:

- *Bill Smith is a research programmer.*
- *Bill Smith is not a programmer.*

Ψ_1 represents the new sentence:

- *Bill Smith is a programmer.*

X_1 represents the ontological background information:

- *If X is a research programmer then X is a programmer.*
- *If X is a research programmer then X is not a staff member.*
- *If X is a staff member then X is not a research programmer.*

Furthermore, let us assume that δ_1, δ_2, δ_3, δ_4 and δ_5 represent the five subsequent questions:

- *Does Bill Smith work at Macquarie University?*
- *Is Bill Smith a programmer?*
- *Is Bill Smith a staff member?*
- *Who works at Macquarie University?*
- *Where does Bill Smith work?*

If we want to check the micro theory $(X_1 \wedge \Phi_1)$ for *consistency* and feed the negation of $\neg(X_1 \wedge \Phi_1)$ to Otter and $(X_1 \wedge \Phi_1)$ to Mace4, then Mace4 will find a satisfiable model and we can stop Otter searching for a proof. That means we know that the micro theory is consistent.

If we want to check the micro theory $(X_1 \wedge \Phi_2)$ for *consistency* and feed the negation of $\neg(X_1 \wedge \Phi_2)$ to Otter and $(X_1 \wedge \Phi_2)$ to Mace4, then Otter will find a proof and we can stop Mace4 looking for a finite satisfiable model. That means we know that the micro theory is not consistent.

If we want to check the formula Ψ_1 for *informativity* with respect to the context $(X_1 \wedge \Phi_1)$ and feed the negation of $(X_1 \wedge \Phi_1 \rightarrow \Psi_1)$ to Otter and $(X_1 \wedge \Phi_1 \wedge \neg\Psi_1)$ to Mace4, then Otter will find a proof and we can stop Mace4 looking for a model. That means we know that the theory is not informative.

If we want to answer questions such as δ_1-δ_5, then we have to negate the formulas derived from the questions, before we combine them in a micro theory and feed them to Otter, since Otter conducts a resolution proof. This is not necessary for Mace4, since we can extract answers to questions from the model in a separate step. Figure 10 shows the input to Otter for the question (theorem)

- *Where does Bill Smith work?*

given the information (axiom)

- *Bill Smith works at Macquarie University.*

without any additional background knowledge.

```
set(auto).
assign(max_seconds,5).
assign(max_proofs,-1).
set(prolog_style_variables).

formula_list(sos).

((exists A (exists B
 (exists C (exists D
 (prop(A,[at],B,C) &
 (role(A,location) &
 (pred(B,[work],D) &
 (evtl(B,event) &
 (obj([macquarie,university],C) &
 (struc(C,atomic) &
 (obj([bill,smith],D) &
  struc(D,atomic))))))))))) &
 -((exists E (exists F (exists G
   (exists H (exists I (exists J (exists K
   (prop(E,F,G,H) &
   (role(E,location) &
   (obj(I,H) &
   (struc(H,J) &
   (pred(G,[work],K) &
   (evtl(G,event) &
   (obj([bill,smith],K) &
    struc(K,atomic))))))) &
    -$answer([[where],F,I,E,H])))))))))).

end_of_list.
```

Figure 10: Input to Otter with Answer Literal

As the input to Otter shows the question has been negated and an answer literal has been added. The answer literal

- `-$answer([[where],F,I,E,H]).`

retains the interrogative word and records the variable bindings during the proof for the subsequent answer generation.

7 RSS Export

A web feed written in controlled natural language can be exported as an RSS feed. RSS is a family of XML-based web feed formats designed for sharing and aggregating web content (RSS 2002). RSS feeds provide summaries of web content together with links to the full versions of the content. In our case, the specification texts written in the web feed mode can be exported as an RSS feed. Basically, an RSS feed is an XML document consisting of an <rss> element with a single <channel> element, which contains meta information about the channel and its content, and any number of <item> elements, which store the summaries of individual web pages. Let us have a closer look at the general structure of an RSS feed that is generated by PENG Online system:

```
<?xml version="1.0"?>
<rss version="2.0">

  <channel>
    <languge> x-peng </language>
    <generator> PENG Online </generator>
    <title> ChannelTitle </title>
    <link> ChannelLink </link>
    <description> Ontology </description>
    <category domain=URILexicon> </category>

    <item>
      <title> WebPageTitle </title>
      <link> WebPageURI </link>
      <description> WebPageSummary
      </description>
    </item>

  </channel>
</rss>
```

Figure 11: Structure of RSS Feed

In our case the <channel> element uses six different subelements for storing the meta information and one or more <item> elements for storing information about individual web pages. The first subelement of the <channel> element is the <language> element which stores the information about the language the channel is written in. In our case, the value x-peng denotes an experimental language tag for the controlled natural language PENG. The second subelement is the <generator> element which indicates that the program used to generate the channel is PENG Online. The third subelement is the <title> element and specifies the title of the channel. The fourth subelement is the <link> element which contains the channel's URL. The fifth subelement is the <description> element which stores the ontological knowledge about the web feed. The sixth subelement is the <category> element which is empty in our case but uses an attribute with a URL as value. The URL points to the exported user lexicon which needs to be accessed when the RSS feed is reloaded by the PENG system. In our case, a <channel> element may contain one or more <item> elements - one for each summary of a web page which is part of the web site. The <item> element has a <title> element as subelement which stores the title of the web page and a <link> element which points to the full version of the web page. Finally, the <description> element of the <item> element stores the summary of the web page in controlled natural language.

8 Conclusions

In this paper, we presented a new approach that allows non-specialists to annotate individual web pages of a website with machine-processable information in controlled natural language and to augment these descriptions with domain-specific ontological information in controlled natural language. The writing process of these specification texts is supported by a text editor which uses predictive interface techniques. The text editor is implemented as a Java applet and communicates over the Internet with a language processor and a reasoning service. The language processor provides look-ahead information for the text editor and translates a specification text into first-order predicate logic via discourse representation structures. The resulting first-order formulas can be combined for various reasoning tasks into micro theories. These micro theories are processed by a reasoning service which combines a theorem prover together with a model builder. The theorem prover provides a negative check on consistency, informativity and questions, and the model builder provides a positive check for the same inference tasks. It is important to note that Web feeds written in PENG are both human-readable and machine-processable and can be maintained by non-specialists with the help of the PENG editor. Any RSS aggregator can subscribe to such a "seemingly informal" web feed, but the full processing power is only available via PENG Online or another PENG-compliant tool.

Acknowledgments

The research reported here is supported by the Australian Research Council, Discovery Project No. DP0449928. We would also like to thank to three anonymous reviewers for their valuable comments.

References

ASD (2005): ASD Simplified Technical English, Specification ASD-STE100, *A Guide for the Preparation of Aircraft Maintenance Documentation in the International Aerospace Maintenance Language*, Issue 3, January.

Blackburn, P. and Bos, J. (2003): Computational Semantics, in: *Theoria* 18(1), pp. 27-45.

Blackburn, P. and Bos, J. (2005): *Representation and Inference for Natural Language*, A First Course in Computational Semantics, CSLI Publications.

Bos, J. (2003): Exploring Model Building for Natural Language Understanding, in: *Proceedings of ICoS-4*.

Bos, J. (2006): Three Stories on Automated Reasoning for Natural Language Understanding, in: *Proceedings of ESCoR (IJCAR Workshop)*: Empirically Successful Computerized Reasoning, pp. 81-91.

Cheyer, A. and Martin, D. (2001): The Open Agent Architecture, in: *Journal of Autonomous Agents and Multi-Agent Systems*, vol. 4, no. 1, March, pp. 143-148.

Fuchs, N. E. and Schwitter, R. (1996): Attempto Controlled English (ACE), in: *Proceedings of CLAW 96*,

First International Workshop on Controlled Language Applications, University of Leuven, Belgium, March 1996, pp. 124-136.

Fuchs, N. E., Schwertel, U. and Schwitter, R. (1999): Attempto Controlled English – Not Just Another Logic Specification Language, *Lecture Notes in Computer Science* 1559, Springer.

Fuchs, N. E. and Schwertel, U. (2003): Reasoning in Attempto Controlled English, in: F. Bry, N. Henze and J. Maluszynski (eds.): *Principles and Practice of Semantic Web Reasoning, International Workshop PPSWR 2003*, Mumbai, India, pp. 174-188.

Grosof, B., Horrocks, I., Volz, R. and Decker, S. (2003): Description Logic Programs: Combining Logic Programs with Description Logic, in: *Proceedings of WWW 2003*, Hungary, ACM, pp. 48-57.

Holt, A., Klein, K. and Grover, C. (1999): Natural language for hardware verification, in: *Proceedings of ICoS-1 workshop: Inference in Computational Semantics*, Institute for Logic, Language and Computation (ILLC), Amsterdam, August, pp. 133-137.

Horrocks, I. and Patel-Schneider, P. F. (2003): Three theses of representation in the semantic web, in: *Proceedings of WWW 2003*, Hungary, ACM, pp. 39-47.

Huijsen, W. O. (1998): Controlled Language – An Introduction, in: *Proceedings of CLAW 1998*, Pittsburgh, pp. 1-15.

Kamp, H. and Reyle, U., (1993): *From Discourse to Logic*, Kluwer Academic Publisher.

Lev, I., MacCartney, B., Manning, C. D. and Levy, R. (2004): Solving logic puzzles: from robust processing to precise semantics, in: *Proceedings of the ACL-04 Workshop on Text Meaning and Interpretation*, pp. 9-16.

Manthey, R. and Bry, F. (1988): SATCHMO: A Theorem Prover Implemented in Prolog, in: P*roceedings CADE 88*, pp. 415-434.

Martin, D., Cheyer, A. and Moran, D. (1999): The Open Agent Architecture: A Framework for Building Distributed Software Systems, in: Applied Artificial Intelligence, vol. 13, no. 1-2, January-March, pp. 91-128.

McCune, W. (2003a): Otter 3.3 Reference Manual. Tech. Memo ANL/MCS-TM-263, Mathematics and Computer Science Division, Argonne National Laboratory, Argonne, IL, August 2003.

McCune, W. (2003b): Mace4 Reference Manual and Guide. Tech. Memo ANL/MCS-TM-264, Mathematics and Computer Science Division, Argonne National Laboratory, Argonne, IL, August 2003.

Patel-Schneider, P. F. (2005): A Revised Architecture for the Semantic Web Reasoning, in: *Proceedings of PPSWR'05*, Dagstuhl, Germany, LNCS 3703, pp. 32-36.

Pulman, S. G. (1996): Controlled Language for Knowledge Representation, in: *Proceedings of CLAW 96*, First International Workshop on Controlled Language Applications, University of Leuven, Belgium, March 1996, pp. 233-242.

RSS 2.0 Specification (2002): Technology at Harvard Law, Internet technology hosted by Berkman Center, available at: http://blogs.law.harvard.edu/tech/rss.

Schwitter, R. (2002): English as a Formal Specification Language, in: *Proceedings of the Thirteenth International Workshop on Database and Expert Systems Applications (DEXA 2002)*, W04: Third International Workshop on Natural Language and Information Systems - NLIS, 2-6 September 2002, Aix-en-Provence, France, pp. 228-232.

Schwitter, R. (2003): Incremental Chart Parsing with Predictive Hints, in: *Proceedings of the Australasian Language Technology Workshop 2003*, December 10, University of Melbourne, Australia, pp. 1-8.

Schwitter, R., Ljungberg, A. and Hood, D. (2003): ECOLE - A Look-ahead Editor for a Controlled Language, in: *Proceedings of EAMT-CLAW03*, Controlled Translation, Joint Conference combining the 8th International Workshop of the European Association for Machine Translation and the 4th Controlled Language Application Workshop, May 15-17, Dublin City University, Ireland, pp. 141-150.

Schwitter, R. (2004): Dynamic Semantics for a Controlled Natural Language, in: *Proceedings of the Fifteenth International Workshop on Database and Expert Systems Applications (DEXA 2004)*, NLIS'04: 4th International Workshop on Natural Language and Information Systems, 30 August - 3 September 2004, Zaragoza, Spain, pp. 43-47.

Schwitter, R. and Tilbrook, M., (2004): Dynamic Semantics at Work, in: *Proceedings of LENLS2004* (in conjunction with the 18th Annual Conference of the Japanese Society for Artificial Intelligence, 2004), in Kanazawa (Japan), May 31, pp. 49-60.

Schwitter, R. (2005): A Layered Controlled Natural Language for Knowledge Representation, in: S. Cardey, P. Greenfield and S. Vienney (eds.), *Machine Translation, Controlled Languages and Specialised Languages*: Special Issue of Linguisticae Investigationes, Vol. 28, No. 1, pp. 85-106.

Schwitter, R. and Tilbrook, M. (2006): Let's Talk in Description Logic via Controlled Natural Language, in: Proceedings of the Third International Workshop on Logic and Engineering of Natural Language Semantics (LENLS2006) in Conjunction with the 20th Annual Conference of the Japanese Society for Artificial Intelligence, Tokyo, Japan, June 5-6, pp. 193-207.

Sowa, J. F. (2004): Common Logic Controlled English, *Draft*, 24 February 2004, available at: http://www.jfsowa.com/lce/specs.htm.

Thompson, C. W., Pazandak, P. and Tennant, H. R. (2005): Talk to Your Semantic Web, in: *IEEE Internet Computing*, Vol. 9, No. 6, pp. 75-79.