Ontology-based Controlled Natural Language Editor Using CFG with Lexical Dependency

Hyun Namgoong, Hong-Gee Kim

Biological Knowledge Engineering Lab, Seoul National University,
28-22 Yeongeon Dong, Jongno Gu, Seoul 110-749, Korea
{ngh, hgkim}@snu.ac.kr

Abstract. In recent years, CNL (Controlled Natural Language) has received much attention with regard to ontology-based knowledge acquisition systems. CNLs, as subsets of natural languages, can be useful for both humans and computers by eliminating ambiguity of natural languages. Our previous work, OntoPath [10], proposed to edit natural language-like narratives that are structured in RDF (Resource Description Framework) triples, using a domain-specific ontology as their language constituents. However, our previous work and other systems employing CFG for grammar definition have difficulties in enlarging the expression capacity. A newly developed editor, which we propose in this paper, permits grammar definitions through CFG-LD (Context-Free Grammar with Lexical Dependency) that includes sequential and semantic structures of the grammars. With CFG describing the sequential structure of grammar, lexical dependencies between sentence elements can be designated in the definition system. Through the defined grammars, the implemented editor guides users' narratives in more familiar expressions with a domain-specific ontology and translates the content into RDF triples.

Keywords: Controlled Natural Language, Context-Free Grammar, Lexical Dependency, Ontology, OntoPath, Look-Ahead Editor.

1 Introduction

CNLs, as subsets of natural languages, have recently received much attention with regard to ontology-based knowledge acquisition systems, for its ability to eliminate ambiguity of expressions in natural languages. Several studies were devoted to the use of CNL in ontology-related data processing such as ontology construction, query generation, and data annotation [3][4]. A CNL-based guided look-ahead editor might help users select proper words that meet his intended but vague notions without proper knowledge on the sentence structures. The statements controlled by predefined grammars, usually defined in CFG which is a computational notation of natural language structures, can be translated into ontology-referenced data and queries with precision [2][5].

Our previous work, OntoPath, assists editing in such an intelligent way that it recognizes the resource type of a description and offers users context-sensitive actions

to perform on that description. A domain-specific ontology plays a role in collecting language constituents, such as nouns and verbs, to be translated into RDF triples [9][10]. A lightweight look-ahead editor helps users, specifically medical experts, with guidance on choosing next words, using the approved grammars and semantic relations of entities from the ontology. Because most medical sentences have general recommended structures to ensure precise knowledge expression, CNL and a look-ahead guiding system can assume an important role in such an application.

However, our previous work and other systems have difficulties in enlarging the expression capacity, expanding the grammar, specifying patternized sentences, and adapting informal expressions such as Korean sentences with English words. These difficulties are attributed to the fact that the grammar definition system like CFG does not include semantic structures, but sequential structures of a sentence. These limitations need to be solved to deal with various sentences so that users can exploit more familiar expressions, and are enforced into using the patternized sentences.

A newly developed editor, which we propose in this paper, permits grammar definitions through CFG-LD that includes both sequential and semantic views on sentence structures. Using this grammar definition system, we can define grammars and the semantic structures of sentences to be used in our editor. The Grammar definitions include the structural descriptions of grammatical states to mention sequences of POS (Part-Of-Speech) with CFG. Designations of lexical dependency between sentence elements are also included. Using defined grammars, the implemented CNL editor enables us to get structure data from writer's narratives with 1) more sophisticated expressions, 2) patternized expressions, and 3) informal expressions consisting of multi language constituents.

We begin this paper with the description of related works on a CNL. An explanation of the representation of narratives using RDF triples is provided in section 2. The CFG-LD and its definition rules are discussed in section 3. In sections 4 and 5, we explain the architecture and implementation of the developed editor. Finally, we provide conclusions of this work in section 6.

2 Controlled Natural Language to Semantic Web Data

2.1 Controlled Natural Language

CNL was restricted subsets of natural languages on grammars and dictionaries to eliminate ambiguity and complexity of pure natural languages. Originally, the main purpose of controlled languages was to improve readability for human readers, particularly non-native speakers. An example is AECMA Simplified English that was created as a manual description language for aircraft maintenance guideline. Another advantage of CNL is to improve text processing capability of computers with removed complexity.

Many studies have been done to develop systems that transform written sentences into formal logical expressions. Some well-known examples are as follows: ACE (Attempto Controlled English) [6], CLCE (Common Logic Controlled English) [7],

and PENG-D [2]. For an automatic translation from a discourse representation structure into a variant of first-order logic, ACE is defined as a controlled natural language in Attempto project. The ACE based sentences are translated into the Semantic Web querying language PQL [8]. Other example of a controlled natural language is CLCE that has been developed by Sowa [9]. CLCE, as a formal language with an English-like syntax, is supplied with more expression than ACE in the sense that it supports ontology for sets, sequences, and integers, and also allows in-line declarations of words linked to relational databases. It supports the automated translation of written narratives to conceptual graph or other logical expressions [8]. PENG-D also proposed a computer-processable CNL to be translated into formal logical sentences decidable with an OWL (Web Ontology Language) language.

2.2 Representation of Narratives using RDF Triples in OnthPath

In this subsection, we overview a translation from a sentence to RDF triples with a gross description narratives example of pathologic examination. The description language supported by OntoPath is compatible with a restricted form of RDF and RDF Schema. The system is designed to annotate the semantic metadata in RDF with the vocabularies that are already constrained by a given ontology in RDF Schema. The ontology then plays a role in guiding the generation of medical narratives as RDF documents. The narratives are validated with the syntactic and semantic rules of RDF Schema and are transformed into RDF documents.

An RDF triple statement consists of a specific resource, which is an individual primitive semantic element with a named property and value for that resource. The basic RDF model represents the named properties and property values. A property is a rule that provides the meaning of the expressions, which is specifying the way the thing should be constituted. A built ontology such as a schema, which is a vocabulary description language, provides mechanisms for describing groups of related resources and the relationships between these resources. Instead of defining a class in terms of the properties its instances may have, the ontology describes properties in terms of the resource classes to which they apply. This is the role of the *domain* and *range* mechanisms.

A specimen received contains a cyst which measures 2 × 1cm.

This example sentence can be translated as shown in the figure 1 when it is typed in the form of the predefined grammar through the guidance of the editor [9][10]. The instances about a real patient, ‘a specimen received’ and ‘a cyst’ are conceptualized as the instances of classes **Tissue** and **Cyst**, respectively. The properties, *contains* and *measures,* are also specified with their object values in the sentence.

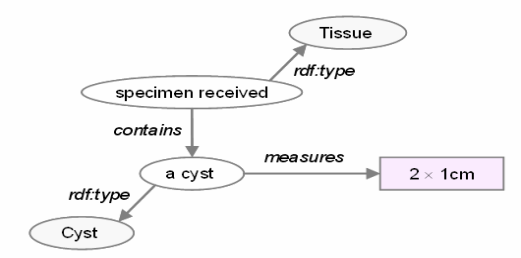

Fig. 1. RDF triples generated from an example sentence.

This translation can be definitely performed on the example sentence written in a predefined grammar. However, if the user describes the sentence with another manner, it can not be successfully translated, because the translation system will assume restricted grammars and translation processes. To expand the grammars for enlarging expression capacity, we can add more grammars using CFG, but it is still not enough for this translation work, since the translation can be different from the composed structure and semantic dependency among the sentence elements.

3 Grammar Expression through CFG-LD

In this chapter, we introduce CFG-LD, which is a grammar definition system for describing grammars with lexical dependences. As we have shown in the previous chapter, the translation between a simple English sentence and a RDF triple is possible through quite simple translation rule on the grammar. However, it is hard to deal with those sentences with different structures, and an annexed expression such as idioms (e.g., “there is something”) or patternized phrases appearing in the sentences. Other grammatical expressions following a different sequence of POS such as ‘subject-object-verb’ are also hardly handled through the original approach. Sequential and semantic structures of those sentences should be declared to enlarge the translation capacities.

Resolving the various structures of sentences can be possible through the previously developed CNL systems listed in the previous chapter. Their built-in sentence resolutions mainly relied on English are restricted in the informal expressions consisting of multi-language constituents, and it is also hard to gather well-defined CNL grammars written in every desired language. Therefore, in our CNL based editor, we employ slightly modified grammar expressions named CFG-LD. It notifies a lexical parser for both grammars and lexical dependencies, to let the parser or system know sequential and semantic structures of the grammars where the ontology provides language constituents and domain and range relations of them.

3.1 Grammar Expression with Context-Free Grammar

CFG is a famous computational notation used to express natural language structure, and to make development of applications that parse natural language sentences easily.

Chomsky proposed the notion of CFG as a model for describing natural languages with following four quantities: Terminals, Non-terminals, Productions, and Start symbol [11].

The grammars described below in CFG express simple grammars to parse an example sentence, 'Nam is a student supervised by a professor named Kim' with a set of lexicons enabling aware of terminals' tokens.

$$\begin{aligned} &S \rightarrow NP\ VP\ OP \\ &NP \rightarrow L\ Be\ Det\ N \\ &VP \rightarrow Verb \\ &OP \rightarrow Det\ Verb\quad L \end{aligned} \tag{1}$$

The states marked with the italic characters means terminals whereas others characters denotes non-terminal states. *L* also denotes unfound literals in the dictionaries, or lexicon set. Other words, such as 'is', 'student' and 'supervised by' were known as each terminal. Now, we can parse the example sentence to make a parse tree. These CFG based definitions, however, only contains the sequential structure, but hardly provide dependency structures that show whether a literal is semantically attached to the verb 'supervised by' as a subject or an object. Then, the derived parse tree from the example sentence using the CFG grammar will only contains the structural sequence.

Grasping the dependency structures from a sentence is possible with both structural and the semantic point of view. The semantic view on the sentence can be shown as lexical dependencies which can be generally derived from stochastic analysis to show word-to-word dependency relationships [13][14]. The figure 2 describes the lexical dependency found in the sentence.

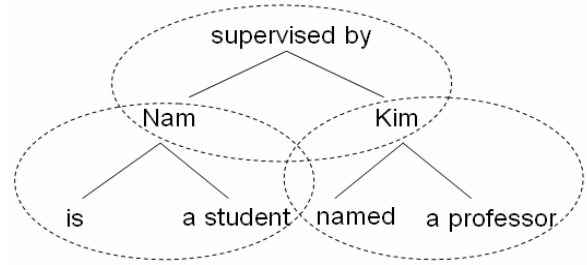

Fig. 2. Dependency set of the example sentence, 'Nam is a student supervised by a professor named Kim.'

From the figure 2 we can easily capture the semantic dependences between words. As the dotted circles show, there are three meaningful structures that exist in the tree. The domain ontology for representing a triple expression make us generate RDF triples from those dependencies like Nam is *instanceOf* a class **student**, Kim is *instanceOf* a class **professor**, and, Nam *supervisedBy* Kim. Here, it is easy to grasp the difference between the dependency and CFG parse tree even through we do not show the parse tree at here.

Then we can add CFG states to the dependency graph as the figure 3 below. From the integrated graph, the dependencies between CFG states are also easily informed [12]. As the graph shows the VP dominates literals from NP and OP, and the *L* in LP node also depends on *Be*, and *Det N*.

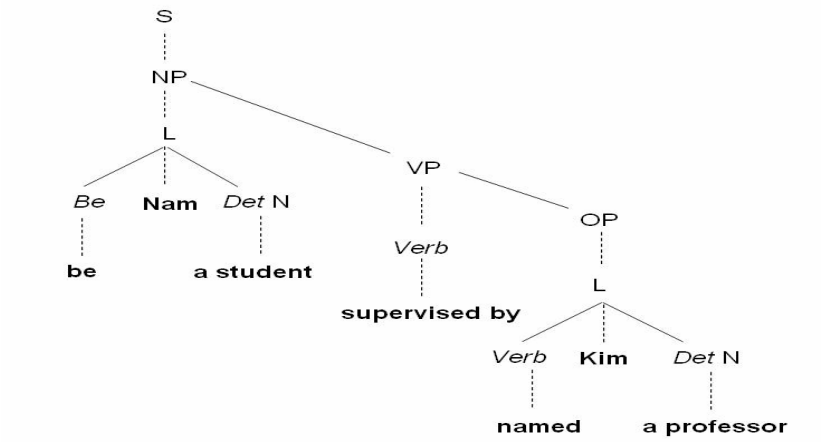

Fig. 3. Integrated parse tree with dependency structures.

The aim of the CFG-LD is to announce possible dependency structures of a sentence written in the defined grammars. With settled dependencies, the dependency existing between states that appear from an incoming sentence can be caught and can be translated into RDF triples precisely.

3.2 CFG with Lexical Dependency

Here, we will introduce CFG-LD with an example of a basic grammar definition. As we mentioned, the entities of the domain ontology are a source of lexicons of terminal states; a class in the ontology is presented as a *noun* state whereas a predicate is presented as a *verb*. Supplying additional well-known functional words as lexicons for other terminal states, the words appearing in a sentence can be tokenized and located in its category using the set of the lexicons.

```
S → NP(S) VP (NP<-V) OP(VP<-O)

NP → L (NP<-S)(S) Be(L<- V) Det Noun (L<-O)
        | L (NP<-S)

VP → Verb (NP<-V)

OP → Det Noun (O) Verb (Noun<-V)
                       L (Verb<-S)(OP<-O)
```

(2)

The above notational rule is an example grammar definition of CFG-LD to handle the sentence ‘Nam is a student supervised by a professor named Kim.’ It shows the rule for a simple sentence composed of non-terminals; ‘NP’, ‘VP’, and ‘OP’. This basic grammar definition is similar to conventional CFG, save the brackets next to each state. The characters in the brackets designate the dependency among the states. As in the first line, we can present the main dependency conducted by NP, VP, and OP. The single character ‘S’ within the brackets, ‘(’ and ‘)’, tells that the state NP has a dependency set, and it has a dependency element ‘Subject.’ The real value of the element will be filled from its nested state because NP is a non-terminal. The symbols with arrow, ‘<-’, within the brackets mean that the state has depending elements such as ‘Verb’ or ‘Object’ on the left side of the arrow, where ‘S’ denotes ‘Subject’, ‘V’ denotes ‘Verb’, and ‘O’ means ‘Object’. In here, the left state should be selected from the states shown see in the same line. In the second line, a terminal *L* has an actual value for Subject element of the dependency set owed by NP and has its own dependency set starting from Subject with its own value. The new creation of the dependency set constructed with *L* and following *Be* and *noun* is determined by the structure of a sentence varied by an OR operator, ‘|.’

These notations with a set of brackets allow us to describe the sentence structure with lexical dependencies existing in the sentence. Relatively less meaningful words like ‘a’, which is denoted as *Def* state, also can be presented in the sentence structure to elevate the familiarity of expressions, even when we are not intended to deal with the meanings of such words. In the next section, we explain how the rules can be used to parse a sentence using a CFG-LD parser.

3.3 Resolving Sentences Using CFG-LD

Here, we describe how a sentence is resolved in the CFG-LD parser using CFG-LD applied to our previous example. First, as we use CFG rules to deal with sentences, the CFG-LD parser dynamically makes a parse tree from an incoming sentence with a supplied lexicon set. The parse tree can be derived from the example sentence - ‘Nam is a student supervised by a professor named Kim’- or from its front part. From a fundamental description, the lexicon set has ‘is’ as a *Be* state, ‘a’ as a *Def*. The domain ontology provides lexicon sets for nouns and verbs such as ‘student’ and ‘professor’ as *Noun*s, and ‘*supervised by*’ as a *Verb*. Then, *L* is an unfound token, just a literal.

As figure 4 shows, the parser constructs dependency sets with designated lexical dependencies from the sentence. A set of rectangle boxes means a dependency set created during parsing process. If the parser meets dependency creation instructions such as ‘(S)’, it creates a new dependency set. The dotted lines show which state puts the dependency element values. If a nested terminal state is employed by the incoming sentence, the parser puts the type of state with its value as a dependency element. Especially, *Noun* and *Verb* are states come from the ontology resources, and their actual values become the URIs of the resources.

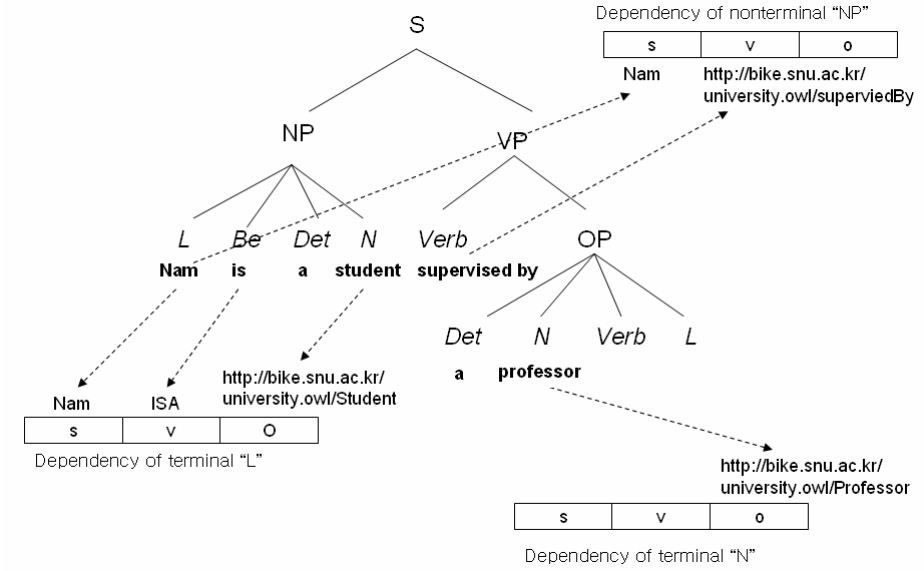

Fig. 4. Dependency sets from the parse tree from the example sentence, 'Nam is a student supervised by a professor'.

The produced dependencies sets are used for two functionalities in the CNL editor; triple generation and context-based word recommendation. From the dependency sets in the figure 4, triples can be easily generated from them. Word recommendation also relies on dependency sets. Basically, the editor can know available next states from the parse tree, but it also can narrow down the scope of the proper words using semantic relations among the entities from the domain ontology. In the above example, the editor will recommend next verb-type words by looking ahead the parse tree, and narrow down those verbs by using those *domain* and *range* relations in the ontology. Therefore, the editor will show users those verbs that can take 'professor' or the parent classes in the ontology as their range value

To avoid ambiguities, when the dependency sets are captured from the parsed sentences and triples are generated from the sets, the CFG-LD should express following agreements.

1) Every state in CFG is divided into two kinds; 'dependency involved', and 'dependency not involved.' The 'dependency involved' state should provide at least one value for an element to a dependency set when it is fully expanded. The 'dependency not involved' state is never involved with any dependency sets. Therefore, the 'dependency not involved' state is declared for the definition of additional terms and patternized phrases
2) Every state can create a dependency set with elements, e.g., subject, verb and object.
3) A state and its nested states can set numerous values for a single element. If an element has multiple values, they are respectively used during triple generations. For example, if a dependency set has two subject values ('I' and

'You'), a verb value ('like'), and an object value ('cake'), then two triples, 'I-like-cake' and 'You-like -cake', are generated.

4) A state can be related to multiple dependencies. A state can make a new dependency set and also give values to other dependencies.

With such grammar definitions following the agreements, we can also define grammars specific to languages of interest that exhibit different sequential appearances from sentence constituents, e.g., Korean. The next rule shows the grammar definitions for Korean sentences. We assume that the lexicons of the terminals are provided through pre-declared lexicons and Korean terms for entities of the ontology. For example, '초등학생' is a Korean word for a class 'ElementarySchoolStudent', '중학생' is a word for a class 'MiddleSchoolStudent', '인' is a postpositional word expressing a 'is-a' relation, and '와' is a postpositional word expressing 'and'. '영희', '철수', and '영수' are literals, and '는 좋아한다' is a verb corresponding to a property '*likes*.'

```
S → OP (O) SP (OP<-S,V)

OP → Class (OP<-O)(O) is (Class<-V) literal
                              (Class<-S) ObjectPost

SP → Class(SP<-S)(O) is (class<-S) literal
  (Class<-V) AndPost literal(Class<-V) Verb (SP<-V)
```

(3)

Through the rule 3, we can notify the structures and dependency of a sentence to the CFG-LD parser- for example, the literal in OP has a relationship with the literals and the verb in SP, so that the editor can make triples from Korean sentences such as '초등학생 인 영희 를 중학생 인 철수 와 영수 는 좋아한다'. The triples from the sentence consists of as follows: '영희' is *instanceOf* '**ElementarySchoolStudent**'; '철수' is *instanceOf* 'MiddleSchoolStudent'; '영수' is *instanceOf* '**MiddleSchoolStudent**'; '철수' *likes* '영희'; and '영수' *likes* '영희'.

3.4 Internal APIs

The CFG-LD definitions can be used to express grammars with their lexical dependency; however, it is sometimes hard to deal with narratives when a user use anaphoric terms like pronouns and quantifiers. Such anaphoric terms are an inevitable feature of languages because human writers much rely on those terms. They cab be replaced with those words that they reference in the document he currently edits.

Therefore, CFG-LD provides a space for specifying dependency values using internal APIs which usable in grammar definitions. By adding the braces, '{', '}', we can also specify the values to be replaced when the dependency set is terminated. In that space, the way for specialized handling of particular anaphoric terms can be presented with several internal APIs or some predefined terms like ISARELATION. The table 1 lists some internal APIs for handling anaphoric terms.

Table 1. Internal API set for anaphoric terms.

Function Name	Functionality description
nearInstanceOf(Class URI)	Returns the nearest instance of the Class
nearInstancesOf(ClassURI)	Returns the near instances of the Class
recentlyCreatedInstance()	Returns instance recently created
recentlyCreatedInstances()	Returns instances recently created
domainInstanceFitDependency()	Returns an instance of domain class of depended predicate
domainInstancesFitDependency()	Returns a set of instances of domain class of depended predicate
rangeInstanceFitDependency()	Returns an instance of range class of depended predicate
rangeInstancesFitDependency()	Returns a set of instances of range class of depended predicate
rangeClassFitDependency()	Returns a class of depended predicate
rangeClassesFitDependency()	Returns a set of classes of depended predicate

As the functionality descriptions show, the internal functions are mainly related to the referenced instances created in the current document. The first two functions, *nearInstanceOf* and *nearInstancesOf,* return an instance or instance of the designated class already made in the document. The *RecentlyCreatedInstance* and *RecentlyCreatedInstances* functions return any instance recently created. The other functions return a single or multiple instances, or a class suitable to the state's connected dependency sets.

The rule below shows CFG-LD grammar descriptions using these internal functions. The expressions deal with a pronoun, 'it', using the function *RecentlyCreatedInstances().* The *Noun* with the definite article 'the' are handled with the *nearInstanceOf(CLASS)* function where a *Noun* means a class from the domain ontology.

```
S  →  it (S) verb (it<-V) object (it<-O)

it  →  IT (it<-S){recentlyCreatedInstance()}

verb  →  Verb (verb<-V)                                    (5)

object  →  Def Noun (object<-O){nearInstanceOf(Noun)}
```

4 Architecture of CNL based Editor

In this chapter, we introduce the architecture of the CNL-based editor. It consists of five main components: Look-Ahead Interface, CFG-LD Parser, Lexicon Pool, Triple Generator, and Predictor, as we can see in figure 5.

1) Look-Ahead Interface: It provides a narrative description interface for a writer. Observing user's input, it recommends proper next words to let the writer describe sentences with precise structures and semantics.

2) CFG-LD Parser: It parses an incoming sentence and makes dependency sets using CFG-LD definitions. First, it tokenizes an incoming sentence with the support of Lexicon Pool, and then dynamically generates a parse tree and dependency sets from a set of the tokenized words.

3) Predictor: It predicts the next terminal states from the parse tree. It also examines the semantic relations in the domain ontology, e.g., domain and range relations, to provide semantically suitable recommendation

4) Lexicon Pool: It holds lexicon sets from the domain ontology and functional words with their terminal types in the grammar definitions. It provides interfaces for searching for words that have a terminal state and contain a certain substring.

5) Triple Generator: It generates RDF triples from the dependency sets when the sentence is completely terminated. A literal which became an identifier and recursively referenced by other triples in the same sentence are replaced with its URIs in this component.

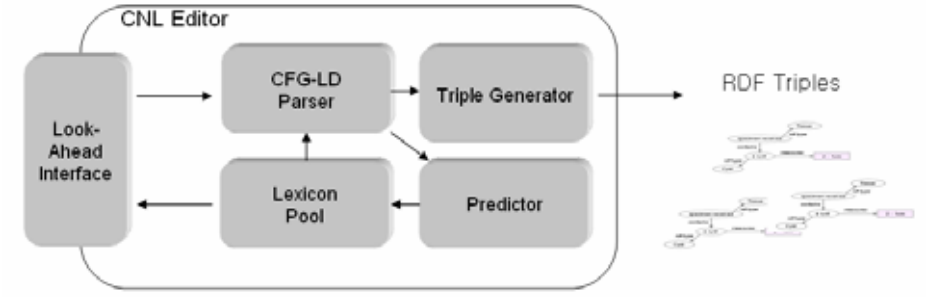

Fig. 5. Architecture of CNL based Editor with CFG-LD.

When a writer starts to type a sentence with Look-Ahead Interface, the CFG-LD Parser builds a parse tree using the terminals identified by the Lexicon Pool at every modification. By looking ahead the defined grammars and the produced dependency sets, the Predictor foretells the suitable terminals and words. Then, the Lexicon Pool delivers candidate next words to Look-Ahead Interface for guiding a user. At the end of a sentence, when dependency sets are completely filled up by the CFG-LD Parser, the Triple Generator generates RDF triples from the dependency sets.

5 Implementation

We implemented a prototype of the Ontology based CNL editor with the explained architecture. In the prototype, AJAX (Asynchronous JavaScript and XML) was employed for implementation of the Look-Ahead Interface. The other components are hided in the screenshot but, they were supplied a pathology ontology written in OWL and some functional words for medical pathology examination descriptions. User-displayable labels written in the multi language words of the ontology entities are provided, for example, a property, '*fixed_in*', is offered as the multiple labels, such as, 'fixed in' and '고정된' which a Korean word. Several grammars are defined and informed to the CFG-LD parser.

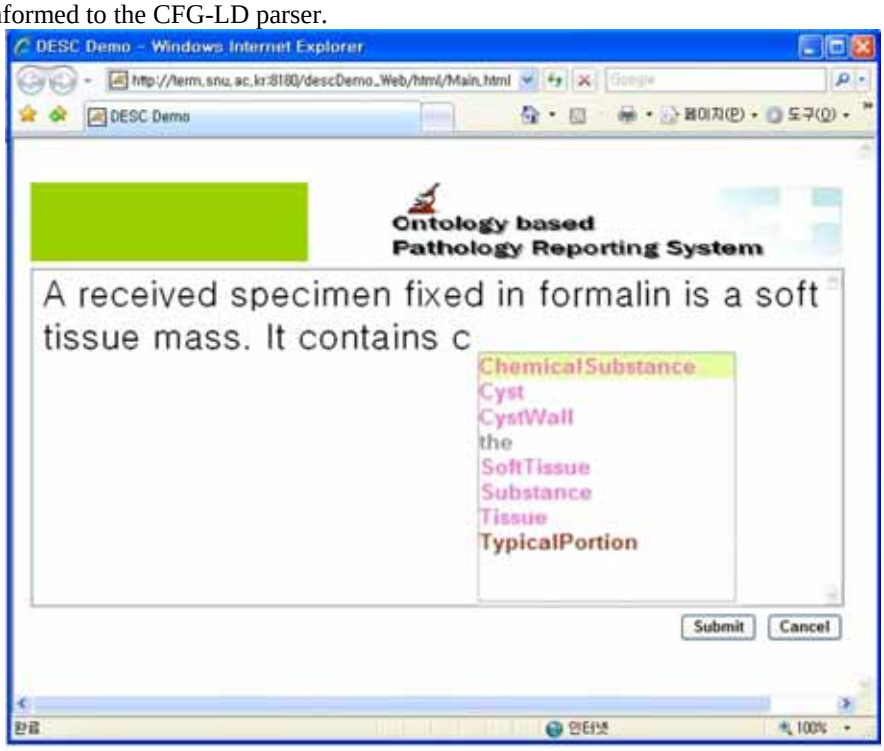

Fig. 6. Screenshot of Ontology based Pathology Reporting System exploiting CFG-LD in English narratives editing.

The figure 6 shows the English narratives editing. If a writer starts to enter characters for the description, the editor displays proper terms to be selected by his/her for precise sentence expressions. The first sentence is realized through a patternized grammar rule which starts with a phrase, 'A received specimen.' The pink-colored terms in the tool-tip box mean words from the ontology. If a user sends the document by clicking the 'Submit' button, the editor stores the narratives as RDF triples.

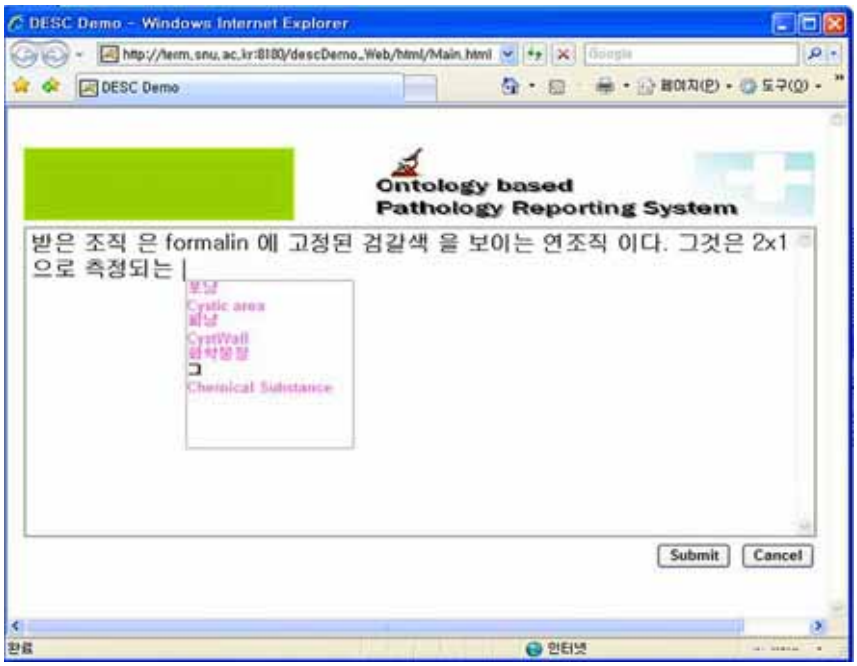

Fig. 7. Screenshot of Ontology based Pathology Reporting System exploiting CFG-LD in Korean narratives editing.

Figure 7 also shows Korean narratives editing. The first sentence presented in the editor is attained through a patternized grammar rule which starts with a Korean phrase, '받은 조직' which means 'a received specimen.' The sentence has the same meaning with 'Received Specimen fixed in formalin is a dark-black soft tissue mass,' but with different sequential structures. Because the editor is equipped with the Korean grammars, the tool-tip box shows some Korean words.

6 Conclusion

We proposed a slightly revised grammar definition system named CFG-LD (CFG with Lexical Dependency) in this paper. It permits grammar expressions with lexical dependency designation among sentence constituents. To meet our research purpose for the enablement of domain-specific descriptions, we arranged grammars usable in the ontology-based CNL editor through the definition system. The editor, we implemented with a CFG-LD parser, provides guidance on proper choice of words and translates the end results into RDF triples.

We expect that such grammar definitions can facilitate the definition of the approved and recommended sentence structures for domain-specific narratives. With the implemented CFC-LD Parser, it enables the development of a lightweight CNL editor with domain-specific ontology. The CNL editors will give us a chance to get structured data from users' narratives, along with 1) more sophisticated expressions,

2) patternized expressions, and 3) informal expressions consisting of multi language constituents. However, the dependency structure appearing in a sentence can vary due to types and semantics of language constituents. Though it might be possible to describe states in a more detailed way according to their characteristics, it is by no means an easy task to complete. We are planning to improve our system and CFG-LD to unburden those efforts for the definitions of CFG-LD grammars. In this paper, validation and resolution of conflicts between the grammar rules were not considered when a sentence is parsed by the CFG-LD parser. We will conduct further improvements of the parser considering occurrence of conflicts in our future works.

References

1. Bernstein, A., Kaufmann, E., Fuchs, N., von Bonin, J.: Talking to the Semantic Web – A Controlled English Query Interface for Ontologies. In proceeding of 14th Workshop on Information Technology and Systems (2004)
2. Schwitter, R., Tilbrook, M.: Controlled natural language meets the Semantic Web. In Proceedings of the Australasian Language Technology Workshop 2004 (2004) 55-62.
3. Schwritter, R.: Controlled Natural Language as Interface Language to the Semantic Web. In proceeding of 2nd Indian International Conference on Artificial Intelligence (2005) 1699-1718
4. Fuchs, N.E., Schwertel, U., Schwitter, R.: Attempto Controlled English – Not just another logic specification language. Logic-Based Program Synthesis and Transformation, Lecture Notes in Computer Science 1559, Springer-Verlag, Berlin (1999) 1-20
5. Bernstein, A., Kaufmann, A.: GINO - A Guided Input Natural Language Ontology Editor. In proceeding of International Conference on Semantic Web 2006, Lecture Notes in Computer Science 4273, Springer Verlag (2006) 144- 157
6. Fuchs, N.E., Schwertel, U.: Reasoning in Attempto Controlled English. Principles and Practice of Semantic Web Reasoning, Lecture Notes in Computer Science 2901, Springer Verlag (2003)
7. Sowa, J.: Graphics and languages for the Flexible Modular Framework. Conceptual Structures at Work, Lecture Notes in Artificial Intelligence 3127, Springer-Verlag, Berlin (2004) 31-51.
8. Sowa, J.: Common Logic Controlled English. http://www.jfsowa.com/clce/specs.htm (2004)
9. Kim, H.G., Ha, B.H., Lee, J.I., Kim, M.K.: Narrative Information Processing based on Controlled Natural Language in EMR Systems, International Journal of Medical Informatics, 2007 (in Press)
10. Kim, H.G., Ha, B.H., Lee, J.I., Kim, M.K.: A multi-layered application for the gross description using semantic web technology. International Journal of Medical Informatics, Vol. 74 (2005) 399-407,
11. Chomsky, N.: Three models for the description of language, IRE Trans. Info. Theory 2 (3) (1956) 113-124
12. Barbero, C., Lesmo, L., Lombardo, V., Merlo, P.: Integration of syntactic and lexical information in a hierarchical dependency grammar. Proceedings of the Workshop on Processing of Dependency-Based Grammars (ACL-COLING) (1998) 58–67
13. Carroll, G., Charniak, E.: Two experiments on learning probabilistic dependency grammars from corpora. Technical Report TR-92, Department of Computer Science, Brown University. (1992)
14. Hellwig, P.: Dependency unification grammar. Dependency and Valency, Walter de Gruyter (2003) 593–635