A Knowledge Representation Semantic Network for a Natural Language Syntactic Analyzer Based on the UML

Alberto Tavares da Silva[1], Luis Alfredo V. Carvalho[2]

1 Escuela Politecnica del Ejército del Ecuador
atsfc@espe.edu.ec
WWW home page:
http://www.espe.edu.ec/espe_portal/portal/main.do?sectionCode=91
2 Universidade Federal do Rio de Janeiro,
COPPE - Instituto Alberto Luiz Coimbra
de Pós-graduação e Pesquisa de Engenharia - Brasil
LuisAlfredo@ufrj.br
WWW home page: http://www.cos.ufrj.br/~alfredo

Abstract. The need for improving software processes approximated the software engineering and artificial intelligence areas. Artificial intelligence techniques have been used as a support to software development processes, particularly through intelligent assistants that offer a knowledge-based support to software process' activities. The context of the present work is a project for an intelligent assistant that implements a linguistic technique with the purpose of extracting object-oriented elements from requirement specifications in natural language through two main functionalities: the syntactic and semantic analyses. The syntactic analysis has the purpose of extracting the syntactic constituents from a sentence; and the semantic analysis has the goal of extracting the meaning from a set of sentences, i.e., a text. This paper focuses on the syntactic analysis functionality and applies the UML to its core as a semantic network for knowledge representation, based on the premise that the UML is *de facto* a standard general modeling language for software development.

1 Introduction

In the software engineering area, object-oriented technology use has increased to the point of becoming a currently dominant technology in software development [1]. In spite of the advantages that object-oriented technology can provide in the software development community, the fundamental problems associated with the

identification tasks of the object-oriented elements, i.e., classes, attributes, relationship and multiplicities, remain; these tasks are easily handled manually and guided by heuristics that the analyst acquires through experience, whose results are posteriorly transferred to a CASE tool characterizing an automation gap between natural language requirement specifications and the respective conceptual modeling [2]. The automatic support to requirement analysis processes can better reflect the *problem solve* behavior of experienced analysts [3].

The need for improving software processes approximated the software engineering and artificial intelligence areas. A growing number of researches have used artificial intelligence techniques as a support to software development processes, particularly through intelligent assistants that offer a knowledge-based support to software process' activities [4].

The context of the present work is a project for an intelligent assistant that implements a linguistic technique with the purpose of extracting object-oriented elements from requirement specifications in natural language, enabling the generation of a conceptual model based on the UML class diagram notation. The referred approach includes three main linguistic requirements: a grammar, a knowledge representation structure and a knowledge representation language. The linguistic technique for the proposed intelligent assistant adopts, from computational linguistics (which automatically analyses natural language in terms of software programs called parsers), two main functionalities: the syntactic and the semantic analyses [5]. The syntactic analysis has the purpose of extracting the syntactic constituents that include the lexicon syntactic structures, like verb phrases; and the grammatical categories, like nouns. The semantic analysis has the goal of extracting the meaning from a text. Fig. 1 illustrates a general schema of the problem solution for the referred assistant.

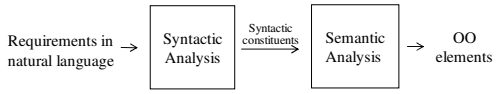

Fig. 1. The problem solution in a pipeline style.

This paper focuses on the syntactic analysis functionality and applies the UML to its core as a semantic network for knowledge representation based on the premise that the UML is *de facto* a standard general modeling language for software development [1]. Based on the referred structure, a knowledge base (KB) can be generated, enabling the syntactic analysis. The proposed semantic network realizes two more logical representations for the intelligent assistant: the static structure and the database conceptual model.

The present article is structured in the following way: the second section presents the three linguistic requirements for the proposed syntactic analyzer; the third section presents the proposed UML semantic network with a case study; and the fourth section presents the conclusions.

2 Linguistic Requirements

2.1 Grammar

There are three basic approaches to a grammar: the traditional, the phrase structure and the transformational. The *traditional grammar* denominates as subject and predicate the essential parts of any construction whose core is the verb. The *phrase structure grammar* includes the syntactic description based on the identification of all kinds of syntactic constituents and the formulation of rules that order the words inside a sentence. The *transformational grammar* has the transformational rules as its basis, making it possible to convert the deep structures, identified in the constituent grammar analysis, into surface structures that correspond to the real form of the enunciation, i.e., the kernel sentence [6-8].

This work adopts the *phrase structure grammar* because it allows the representation of the knowledge to be modeled by the proposed UML semantic network as well as the extraction of the syntactic constituents from the sentences. The cited grammar permits to specify a language with an infinite number of sentences as the natural language, being well-founded on a formalism based on production with four components [5, 9]:

T – terminal vocabulary: language words and symbols being defined;
N – non-terminal vocabulary: symbols used to specify the grammar;
P – set of production;
S – start symbol.

2.2 Knowledge Representation Language

The language enables the formulation of knowledge through symbolic representations that will capacitate a system to reason [10]. First-order logic (FOL) is the most widely used, studied and implemented version of logic [11]. It is important to note, from [10], that whatever other features a knowledge representation language may have ought to comprise a well-defined notion of entailment because the so-called *job reasoning* here means to compute the entailments of a KB.

Many modern logicians limit the expressive power of FOL to a more easily computable subset, like Horn Clauses [10, 11]. In Horn clause representation, a KB can often be separated in two types of clauses: facts and rules. The facts are used to determine the basic truth from a domain, whereas the rules are used to understand the vocabulary and express new relationships. The propositions considered as true arguments are denominated *hypotheses or axioms*, and the propositions that search the logical consequences from the reported axioms are denominated *theorems*. Based on the abovementioned concepts, there appears the activity denominated *a theorem proof*, whose objective is to derive the logical consequences from the given propositions [12, 13]. The logical reasoning, or logical inference, involves the logical consequence concept. Logic is the inference science which is based on two basic hypotheses: in a correct inference the premises must be true and the inferred conclusion must have a logical relation with the premises in a way that guarantees

the transference of the truth contained in these premises to the conclusion; the relation between the premises and conclusion, which guarantees the transference of the truth, is a formal relation denominated logical consequence or logical entailment, which can be analyzed as a relation between logical forms [14]. The Resolution Procedure permits to automate the deductive reasoning in a FOL knowledge base, in a complete and consistent way, with the objective of determining whether a sentence α, or a formula**,** is true or not in a *KB*, i.e., if $KB \models \alpha$ (whether α is a logical consequence of KB). The resolution procedure is more manageable when applied to a Horn clause KB [10], being called a *SLD* resolution (Selected literals, Linear pattern, over Definite clause).

There are two languages that allow a high level symbol manipulation in NLP: Lisp and Prolog. This work emphasizes Prolog, which syntax is in FOL logic with clauses written in Horn clause [10-12, 15]. Prolog clauses include facts and rules that are accepted as a set of axioms and the user's question as the presumed theorem. The Prolog inference mechanism tries to prove the theorem, as it is a theorem prover based on the SLD resolution procedure [12, 13]. Prolog is also a suitable language to implement the phrase structure grammar [12, 13, 16]. A large number of Prolog implementations provide a notational extension called *Definite Clause Grammar (DCG)*, that facilitates the formal grammars' implementation in Prolog and it is directly executed as a syntactic analyzer, which enables sentence decomposition in its constituents. DCG allows implementing the *context dependence*, where a sentence depends on the context where it happens, like a concordant number [13].

Based on the abovementioned considerations, this work adopts the Horn clause in DCG notation as the knowledge representation language, based on the premise that DCG was developed as a linguistic modeling tool that permits the depiction of any sentential structure that can be represented by a phrase structure grammar and, also, that has a well-defined notion of entailment based on an SLD resolution procedure.

2.3 Knowledge Representation Structure

Semantic network is a structure for representing knowledge as a pattern of interconnected nodes and arcs, in a way that the nodes represent concepts of entities, attributes, events and states; and the arcs represent the connections among the concepts [17]. Stuart Shapiro, who implemented the first semantic network with integral support for FOL, believes that a network structure can actually support important types of "subconscious" reasoning that are not directly representable in a linear logical form [17]. Shastri affirms that, in a general way, it is possible to translate a semantic network into a non-graphic language and vice versa [18]. Russel and Norvig [15] consider the semantic network a system specially projected to organize and to reason about/upon categories, offering graphic help to visualize a KB, being also a logical form. To Sowa [19], the semantic network is a declarative graphic depiction which can be used to represent knowledge as an automated reasoning support for it.

Sowa [19] classifies UML diagrams as a semantic network and justifies it by saying that central to the UML exists a network definition of object types and another one like a relational graph that permits the representation of metalevel

information. The UML semantic understanding demands to comprehend the UML specification through a four-layer metamodel hierarchy [1]. The meta-metamodeling layer, called M3 in Fig. 2, defines the metamodeling specification language. The metamodeling layer, called M2, defines the model specification language as the UML. The model layer, called M1, has the primary responsibility of describing semantic domains, i.e., it allows the generation of user models as UML metamodel instances. The lowest layer, called M0, includes run-time instances of the model elements. The UML semantic refers to the run-time interpretation from the generated models [20].

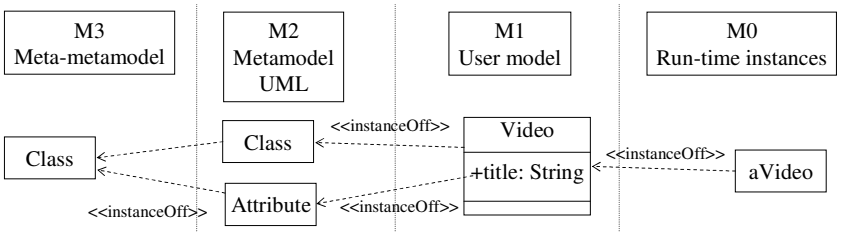

Fig. 2. The UML layer hierarchy

The present work adopts the UML class diagram notation, from the metamodel layer in Fig. 2, as the knowledge representation structure, i.e., the semantic network. The justification considers that a UML class diagram model, in the user model layer of Fig. 2, is a static structure composed by nodes and arcs interconnected, where the nodes represent classes and the arcs, associations [17]. The referred model is a system projected to organize and to reason with object instances, e.g., from UML run-time instance layers, objects representing axioms can be instantiated and asserted in a KB in accordance to the adopted knowledge representation language syntax, i.e., a Horn clause in DCG notation. The Prolog inference engine enables the reasoning through a logical consequence among the referred axioms.

3 Semantic Network for the Syntactic Analyzer

3.1 Knowledge Representation Structure

The linguistic requirements for the proposed syntactic analyzer include: the phrase structure grammar as the grammar, the Horn clause in DCG notation as the knowledge representation language and the UML class diagram as the knowledge representation structure, being this last requirement the core of the present work.

Two domains are defined based on a general problem-solution vision: real world aspects and symbolic representations, as illustrated in Fig. 3. Both domains were defined based on three representation concepts. The first defines representation as a relationship between two domains, where the first is meant to stand for or take the place of the second [10]. The second defines representation as the correspondence

between the real world and the symbolic representation [15]. In the third one, the symbols serve as surrogates for external things [11].

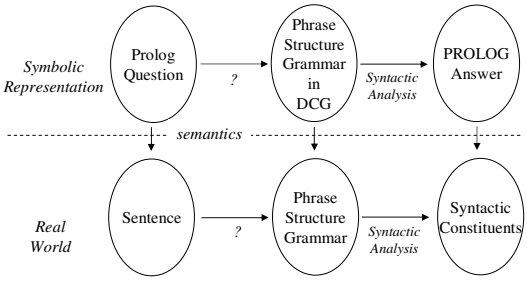

Fig. 3. Space solution domains

Fig. 3 permits a syntactic analysis process correspondence in both domains. In the real word, a natural language sentence is grammatically evaluated by a *phrase structured grammar* that permits to associate the grammar with its constituents' analysis; the results are the syntactic constituents. In the symbolic representation world, the sentence corresponds to a question in Prolog notation, or a theorem to be proved by the Prolog theorem prover; the sentence is processed by the KB inference engine, which axioms implement the *phrase structure grammar* from the respective real world in DCG notation; the result is a Prolog clause as the answer to the goal. There is a semantic relationship between both domains as illustrated in Fig. 3.

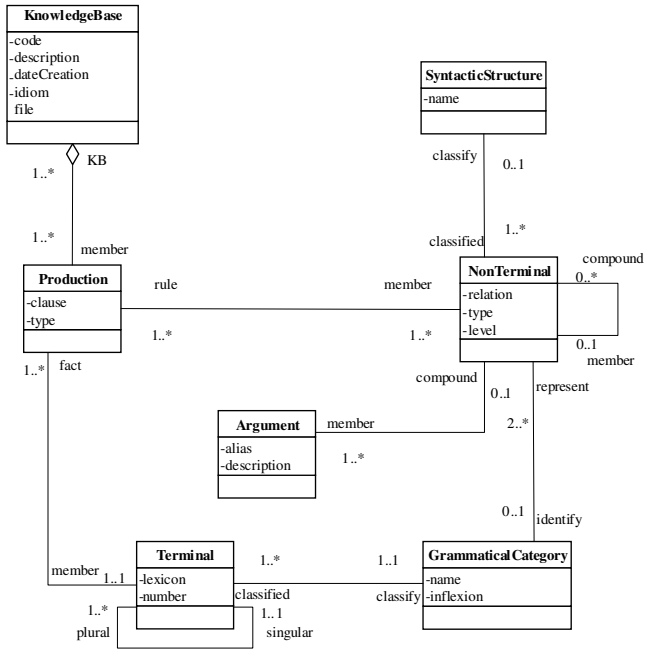

Fig. 4. Semantic network for the syntactic analyzer

Based on the premise that the knowledge kernel of the syntactic analyzer is the grammar, Fig. 4 illustrates a knowledge representation structure of a *phrase structure grammar.* The proposed semantic network works like a bridge between the other two linguistic requirements, i.e., the phrase structure grammar as the grammar and the Horn clause in DCG notation as the knowledge representation language. The knowledge represented by the referred network is the phrase structure grammar and the knowledge generated is a set of axioms from the cited grammar represented symbolically through the Horn clause. This confirms Shastri's affirmation that it is possible to translate a semantic network into a non-graphic language [18].

Based on the premise that the *job of reasoning* is to compute the entailments of a KB [10], the Horn Clause knowledge representation language adopted has a well defined notion of logical entailment. The reasoning occurs through the theorem prover inference mechanism, i.e., Prolog. The reasoning in the proposed syntactic analyzer includes examining whether a sentence has the grammatical sequence in accordance to the phrase structure grammar implemented and, based on the correct analysis, generating the syntactic constituents.

3.2 Case Study

This case study adapts a short example implemented in [13] and illustrates a production instantiation from the proposed UML semantic network, i.e., it shows the composition of an axiom enabling its own assertion in a KB. The following grammar is based on the phrase structure grammar, the knowledge language is Horn clause in DCG notation, implementing the context dependence based on a concordant number [13]:

```
T:  { the, hates, hate, cat, mouse, cats, scares }.
N: { noun_phrase, verb_phrase, determiner, noun, verb,NP, VP, Verb, Det,
     Noun, Number, Number1}.
P: {  (sentence, (Number,sentence(NP, VP)) --> noun_phrase(Number, NP),
      verb_phrase(Number, VP).,
      verb_phrase(Number, verb_phrase(Verb, NP)) --> verb(Number1,Verb),
      noun_phrase(Number, NP).,
      noun_phrase(Number, noun_phrase(Det, Noun)) --> determiner(Det),
      noun(Number,Noun).,
      determiner(determiner(the)) --> [the].,
      verb(singular,verb(hates)) --> [hates].,
      verb(plural,verb(hate)) --> [hate].,
      noun(singular,noun(cat)) --> [cat].,
      noun(singular,noun(mouse)) --> [mouse].,
      noun(plural,noun(cats)) --> [cats].,
      verb --> [scares].,
      verb --> [hates]. }.
 S: { sentence }
```

The non-terminal symbols (N) represent the grammar and include the grammatical category symbols, the syntactic structure symbols and the argument symbols. The NonTerminal class enables the generation of the relations based on the

referred symbols; these relations are, incidentally, the ones that compose the productions. As an example, the following relations compose the above second production:

- *verb_phrase(Number, verb_phrase(Verb, NP)).*
- *verb_phrase(Verb, NP).*
- *verb(Number1,Verb).*
- *noun_phrase(Number, NP).*

Fig. 5 illustrates a run-time interpretation, i.e., layer M0 in Fig.2, which corresponds to an instantiation of the object oProduction from the Production class, which *clause* attribute has the following state: "verb_phrase(Number, verb_phrase(Verb, NP)) --> verb(Number1, Verb), noun_phrase(Number, NP)". The object oProduction is composed by three instantiations from the NonTerminal class.

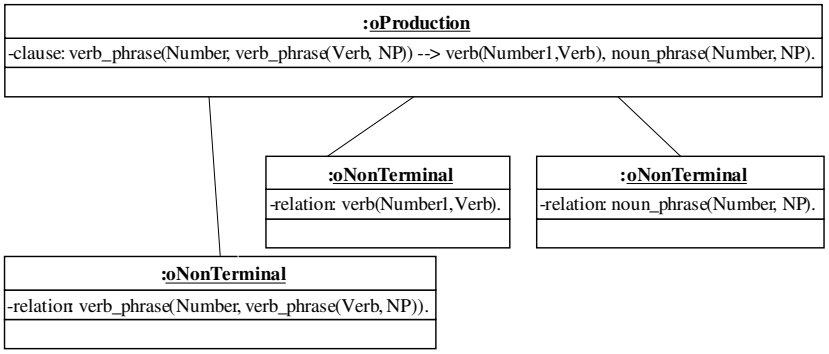

Fig. 5. A run-time interpretation for a production

Fig. 6 shows the instantiation of the oNonTerminal object, also in layer M0 in Fig. 2, whose *relation* attribute has the following state: "noun_phrase(Number, NP)". The referred object composition depends on other three object instantiations, i.e., one object from the SyntaticStructure class and two objects from the Argument class.

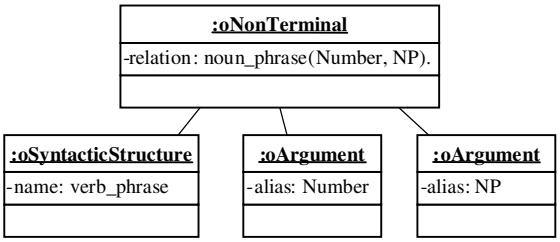

Fig. 6. A run-time interpretation for a relation

As abovementioned, the purpose of the Syntactic Analyzer is extracting the syntactic constituents from a sentence. The prototype, illustrated in Fig. 7, instantiates all the productions of the present case study from the proposed UML semantic network, permitting to extract the syntactic constituents, e.g., from the

sentence “the mouse hates the cat”, the word “mouse” belongs to the grammatical category “noun” and is inside the syntactic structure “noun phrase”.

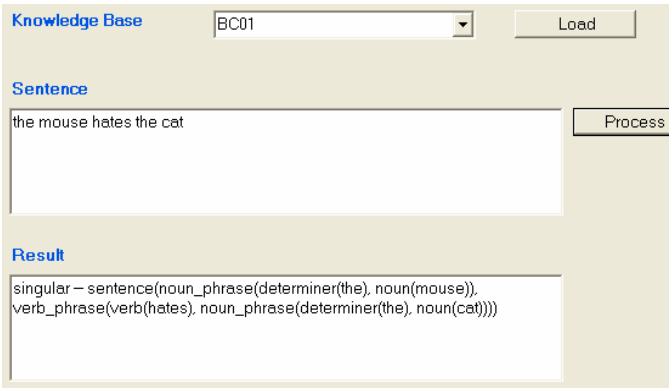

Fig. 7. Syntactic analyzer output

4 Conclusion

The linguistic requirements for the proposed syntactic analyzer include: the phrase structure grammar as the grammar, the Horn clause in DCG notation as the knowledge representation language and the UML class diagram as the knowledge representation structure, i.e., the semantic network, being this last requirement the core of the present work. The referred semantic network works like a bridge between the other two linguistic requirements, i.e., the knowledge represented by the semantic network is the phrase structure grammar and the knowledge generated by the same network is a set of axioms from the cited grammar, represented symbolically through the Horn clause. This confirms Shastri’s affirmation that it is possible to translate a semantic network into a non-graphic language [18].

The semantic network based on the UML class diagram complies with some important concepts. Firstly, its structure is equivalent to a model made by nodes and arcs interconnected [17]. Secondly, it is a system projected to organize and to reason with categories that offer graphical help to visualize the KB, being a kind of logic [15]. Moreover, it is a surrogate for external things, i.e., the phrase structure grammar [11]. In addition to that, it gives support to automation reasoning by means of the theorem prover inference mechanism proposed, which includes examining whether a sentence has the grammatical sequence in accordance to the phrase structure grammar implemented and, based on the correct analysis, generating the syntactic constituents. Finally, inside an object-oriented view from the UML model, the semantics occurs through the object run-time interpretations, as illustrated in Fig. 5.

Briefly, the meaning of the semantic network proposed is a KB whose axioms are able of reasoning under a question, i.e., a theorem to be proved.

References

1. OMG 2004. Object Management Group: UML 2.0 Infrastructure. OMG document ptc/04-10-14, http://www.omg.org/cgi-bin/doc?ptc/2004-10-14.
2. Overmyer, S. P.; Lavoie, B.; and Rambow O. 2001. Conceptual Modeling through Linguistic Analysis Using LIDA. In *Proceedings of the 23rd International Conference on Software Engineering*, 0401. Washington, DC: IEEE Computer Society.
3. Rolland, C., and Proix, C. 1992. A Natural Language Approach for Requirements Engineering. In *Proceeding on Conference Advanced Information Systems Engineering* 257-277. Manchester: Springer-Verlag.
4. R. A. Falbo, Integração de Conhecimento em um Ambiente de Desenvolvimento de Software. Ph.D. Dissertation, COPPE, UFRJ, 1998.
5. R. Hausser, *Foundations of Computational Linguistic* (Springer-Verlag, Berlin, 2001).
6. N. Chomsky, *Syntactic Structures* (Mouton de Gruyter, Berlin, 2002).
7. J. C. Azevedo, *Iniciação à Sintaxe do Português* (Jorge Zahar Editor, Rio de Janeiro, 2003).
8. M. C. P. Souza e Silva and I. V. Koch, *Lingüística Aplicada ao Português: Sintaxe* (Cortez Editora, São Paulo, 2004).
9. R. Grishman, *Computational Linguistics: An Introduction* (Cambridge University Press, 1999).
10. R. Brachman, and H. J. Levesque, *Knowledge Representation and Reasoning* (Morgan Kaufmann Publishers, San Francisco, 2004).
11. J. F. Sowa, *Knowledge Representation. Logical, Philosophical and Computational Foundations* (Brooks/Cole, California, 2000).
12. W. F. Clocksin and C. S. Mellish, *Programming in Logic* (Springer-Verlag Berlin Heidelberg, 2003).
13. I. Bratko, *PROLOG – Programming for Artificial Intelligence* (Addison-Wesley Publishers, 2001).
14. H. Kamp, and U. Reyle, *From Discourse to Logic. Introduction to Modeltheoretic Semantics of Natural Language, Formal Logic and Discourse Representation Theory* (Kluwer Academic Publishers, Netherlands, 1993).
15. S. Russel, Stuart and P. Norvig, *Inteligência Artificial* (Editora Campus, Rio de Janeiro, 2004).
16. A. Gal, G. Lapalme, P. Saint-Dizier and H. Somers, *Prolog for Natural Language Processing* (John Wiley & Sons, 1991).
17. J. F. Sowa, *et al.*, *Principles of Semantic Networks. Explorations in the Representation of Knowledge* (Morgan Kaufmann Publishers, 1991), pp 1-3.
18. L. Shastri, Why Semantic Networks? In: *Principles of Semantic Networks. Explorations in the Representation of Knowledge* (Morgan Kaufmann Publishers, 1991), pp. 109-136.
19. J. F. Sowa, Semantic Networks (August, 12, 2002); http://www.jfsowa.com/pubs/semnet.htm.
20. B. V. Selic, On the Semantic Foundations of Standard UML 2.0, *LNCS 3185*, (Springer-Verlag Berlin Heidelberg, 2004), pp. 181-199.