An application for translation of Spanish sentences into First Order Logic implemented in Prolog

Emilio Rodríguez Vázquez de Aldana (Departamento Informática y Automática. Universidad de Salamanca)
E-mail: aldana@gugu.usal.es

1. Introduction

This work presents an application of the Natural Language Processing (NLP) designed to accept a Spanish text as input and produce its normalisation into First Order Logic (FOL) as output. It was written in Prolog[1], one of the most used programming languages in this kind of applications.

The program was developed for the doctoral course "Extensiones a la lógica de primer orden" (*Extensions of First Order Logic*), at the "Informatica y Automatica" Department of the "Universidad de Salamanca" (Spain), imparted by Dra. María Manzano, in 1999. At present, it is part of a group of applications developed in Prolog that students use in the practical lessons of the "Sistemas de Representación y Procesamiento Automático del Conocimiento" subject, of the "Documentación" degree, at the "Universidad de Salamanca".

2. Basic modules of the application

A whole NLP application uses different knowledge types corresponding to the language analysis levels (phonetic and phonological, morphological, syntactical, semantic and pragmatic) which are formalised and used efficiently in the computer (ALLEN 1994).

The present application only comprises the syntactic and semantic levels. For this purpose, it includes two basic modules corresponding to both analysis levels.

2.1. Syntactic module

It determines if the input string is a grammatical or ungrammatical sentence. If the input phrase is recognised as grammatical, it will produce the tree of syntactic structure as output.

The linguist knowledge necessary to solve this question is divided into lexicon or dictionary and grammar. The NLP applications store all the words of the language (terminal symbols) together with information about, at least[2], the syntactic category (e.g. man: noun; tall: adjective) in the lexicon. The rules with all the allowed combinations of the grammatical categories, that is to say rules of the nonterminal symbols, are registered in the grammar.

The grammar of the present application was written as Context-Free Grammar (CFG) -type 2 of the Chomsky Hierarchy- which is one of the most frequently used types in NLP. The CFG allows to describe an important part of the natural language structures. Furthermore, it is possible to write efficient syntactic analysis algorithms for the CFG (ALLEN 1994). The generative capacity of the CFG was augmented to solve the linguistic problem of the agreement, which is a feature dependent of the context, and thus avoiding the excessive generation of rules in the grammar of this application. The agreement feature is more complex in Spanish than in English, due to the greater inflectional richness of the words in Spanish (SANDOVAL 1998). For example, in the non augmented CFG, the rule *np* ® *det, n* (it means: the noun phrase

[1] The interpreter SWI-Prolog of the Amsterdam University was used.

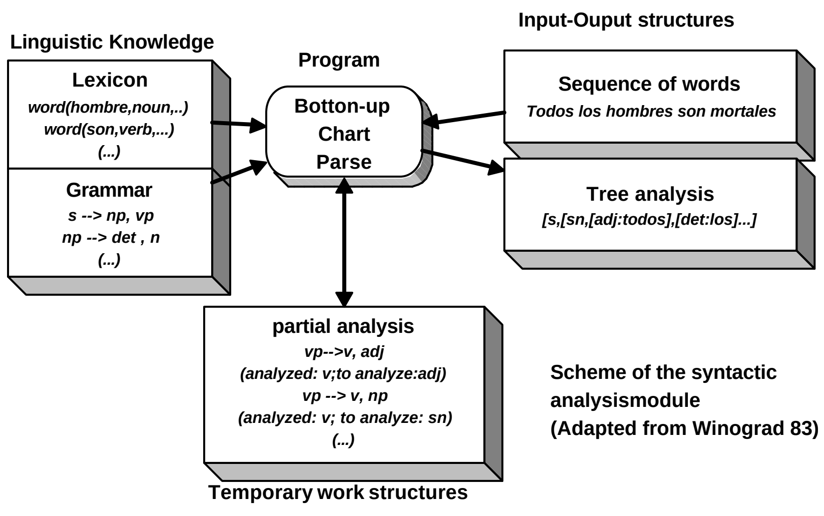
[2] We say "at least" because the actual tendency in Computational Linguistic is to make very complex Lexicons with a great knowledge variety.

comprises a noun after a determiner) would generate four Spanish rules to solve the agreement problem between the *determiner* and the *noun* in gender and number.

np_m_s → det_m_s ,n_m_s	Where: _m (masculine)
np_f_s→ det_f_s, n_f_s	_f (femenine)
np_m_p→ det_m_p, n_m_p	_s (singular)
np_f_p→ det_f_p, n_f_p	_p (plural)

The rules of the Spanish grammar have not been written in the Definite Clause Grammar (DCG) form available in Prolog, because an algorithm of syntactic analysis -called parser[3]- has been implemented, instead of the Top-Down technique that Prolog incorporates. The developed parser was a Bottom-Up Chart that obtains in parallel way all the possible tree analysis. A Chart parser reduces the number of checking when applies the rules of the grammar, by storing temporarily all the partial analysis that are resolved. It is more efficient than the Top-Down parser and it has been described widely in the literature (GAZDAR 1989, ALLEN 1995).

The lexicon of this application, which comprises a very small number of inputs, stores each word together with information of the syntactic category, gender, number, and semantic description, for a further translation. Given that we did not developed a module for the morphological processing, an input for each inflected form of the lexeme was made. For example, for the adjective "bueno" (*good*) we have to make the following inputs: "buena" (feminine, singular), "bueno" (masculine, singular), "buenas" (feminine, plural), "buenos" (masculine, plural). Since the Spanish language has a wide inventory of inflected forms for each lexeme (e.g. more than 50 different forms for any verb; up to 4 for nouns and adjectives...), a more ambitious program for the Spanish language will require a morphologic processing to prevent the excessive growth of the computational lexicon[4].



Scheme of the syntactic analysismodule (Adapted from Winograd 83)

The typology of the sentences recognised by the present application are described using the square brackets [ ] for those symbols that can appear optionally, thus simplifying the reading

[3] The parser is the other element necessary in the module of syntactic analysis of any PLN application.

[4] Obviously, the Spanish research groups in Computational Linguistic have developed computational models to process the specificity of the Spanish language morphology (MORENO, A. and GOÑI, J.M. 1995)

and reducing the number of rules. However, note that this presentation does not says the processing of the agreement (e.g. between determinant and noun, verb and subject, noun and adjective...) neither expresses certain restrictions of unification between the nonterminal symbol, here referred as *quantifier* -which includes the words: “todos” (*all*), “ningún” (*none*), “algún” (*some*)- and articles and nouns.

> *sentence → noun_phrase, verb_phrase*
> *noun_phrase → [quantifier], article, noun, [conjunction, noun_phrase]*
> *noun_phrase→ quantifier, noun, [conjunction, noun_phrase]*
> *verb_phrase → verb_to_be, adjective, [conjunction, adjective]*
> *verb_phrase → transitive_verb, direct_object*
> *direct_object→ [preposition], [quantifier], determinant, noun*

2.1. Semantic Module

This module receives the output of the former phase as input and generate the translation into the formal language of the FOL as a result.

This process comprises two steps. First, the syntactic structure equivalent for the target language is generated starting from the syntactic tree structure of the Spanish sentence. For such a purpose, we determine the functions of the different outstanding constituents of the Spanish sentence, starting from the application of a set of rules. Thus, we make the syntactic tree in FOL. Example:

Input: syntactic structure of the Spanish sentence “Todos los hombres admiran a algún filósofo” (*Every man admire some philosopher*)

```
[SENTENCE
        [NOUN_PHRASE
                [QUANTIFIER: todos]
                [DETERMINER: los]
                [NOUN: hombres]]
        [VERB_PHRASE
                [VERB: admiran]
                [DIRECT OBJECT
                        [PREPOSITION: a]
                        [QUANTIFIER: algún]
                        [NOUN: filósofo]]]
```

Output: syntactic structure of the equivalent sentence in FOL

```
[CONNECTIVE:
        [SN
                [QUANTIFIER: (SUBJECT)]
                [NOUN: hombre(SUBJECT)]]
        [SV
                [VERB: admira(SUBJECT,OBJECT)]
                [DIRECT OBJECT
                        [QUANTIFIER: (OBJECT)]
                        [NOUN:filósofo(OBJECT)]]]
```

The second step is to obtain the string of characters from the syntactic tree in FOL. An algorithm to explore the tree in depth and left to right obtains firstly, all the quantifiers to show them in the beginning. The same algorithm obtains the rest of elements and puts them in the output string.

3. Convenant of translation

The universal quantifier symbol ∀ is represented with the predicate todo(_). The existential quantifier symbol ∃ corresponds with the predicate existe(_).

4. Examples of translation

Some examples of translation are shown. The inputs made by the user are presented as underlined, the English translation between square brackets, and the outputs of the application in italic:

<u>todos los humanos son mortales</u> [all humans are mortal]: *todo(x) : (humano(x) ---> mortal(x))*

<u>algún hombre es bueno y responsable</u> [some man is good and responsible]:

existe(x) : (hombre(x) /\ (bueno(x) /\ responsable(x)))

<u>todos las frutas y las verduras son sanas y nutritivos</u> [all fruits and vegetables are healthy and nutritious]*: Se esperaba un adjetivo femenino despues de: "todos las frutas y las verduras son sanas y ..."* [An femenine adjective was expected after: all fruits and vegetables are healthy and...]

<u>todos las frutas y las verduras son sanas y nutritivas</u> [all fruits and vegetables are healthy and nutritious]: *todo(x) : ((fruta(x) \/ verdura(x)) ---> (sana(x) /\ nutritiva(x)))*

<u>ningún político es honrado y responsable</u> [none politician is honest and responsible]*:*

(~ existe(x)) : (politico(x) /\ (honrado(x) /\ responsable(x)))

<u>todos los hombres y las mujeres admiran la honradez</u> [all men and women admire honesty]:

todo(x) : ((hombre(x) \/ mujer(x)) ---> admira_honradez(x))

<u>algún hombre admira a todos los filósofos</u> [some man admire all the philosopher]:

(existe(x) : todo(y)) : (hombre(x) /\ (admira(x, y) /\ filosofo(y)))

<u>todos los hombres y las mujeres admiran a algún filosófo</u> [all men and women admire some philosopher]: *(todo(x) : existe(y)) : ((hombre(x) \/ mujer(x)) ---> (admira(x, y) /\ filosofo(y)))*

5. Future extensions

We pretend to make an application able to translate a greater set of sentences. For this purpose, not only the grammar will be extended, but also we will include all the potentiality of the Unification-Based Grammar (UBG) formalisms. Nowadays, the UBG is the most powerful tool to make computational grammars (MORENO 1998).

6. References

ALLEN, J. (1995): *Natural Language Understanding*. Redwood, Benjamin/Cummings.

COVINGTON, M.: (1994). *Natural Language Processing for Prolog Programmers.* Englewood Cliffs, Prentice-Hall.

GAZDAR, G. and C. MELLISH (1989): Natural Language Processing in PROLOG. An Introduction to computational linguistics.Reading MA, Addison-Wesley.

MORENO, A and GOÑI, J.M. (1995): “GRAMPAL: A Morphological Processor for Spanish implemented in Prolog” en *Proceedings of the Joint Conference in Declarative Programming*, Salerno, Italy

MORENO SANDOVAL, A. (1998): *Lingüística computacional. Introducción a los modelos simbólicos, estadísticos y biológicos*. Madrid, Síntesis

WINOGRAD, T (1983).: *Language as a Cognitive Process*. Vol. 1: *Sintax*, Reading MA, Addison-Wesley.