# Controlled Spanish as a Data Definition and Manipulation Language

Dulce Aguilar-Solis and Veronica Dahl

Logic and Functional Programming Group
Department of Computing Science
Simon Fraser University
Burnaby, B.C., Canada
dma@cs.sfu.ca

**Abstract.** Controlled natural language is the perfect midpoint between full natural languages and formal languages. A controlled natural language could be efficiently processed by a computer and, at the same time, easy to use by untrained users. We argue that the state of the art in language processing, coupled with that in computational logic, is presently ripe enough to allow interesting uses of human language itself as a "computer" language. We present a novel system as proof of concept, which allows a user of controlled Spanish to define databases or knowledge bases directly in a simplified but natural subset of Spanish. The resulting knowledge base can be queried or incremented at any time.

*Keywords: natural language processing, logic grammars, knowledge bases, controlled natural language as a programming language.*

*Paper Track Paper.*

*Conference Topics: Natural Language Processing, Knowledge Representation.*

## 1 Introduction

The dream of programming at ever higher levels has, in the recent past, suffered serious setbacks. After the initial enthusiasm generated by AI languages, and the expectation that their very high-level traits would percolate into the real world of programming, the advent of the Internet forced us to revert to programming in the often lower-level terms required by languages such as Java. Recently, amalgamations of higher level tools, in particular logic programming with web processing, have given us hope of maintaining the convenience of interconnectivity while relinquishing little of the expressive power offered by languages such as Prolog.

Natural language applications of logic programming mostly span either language to language translation (e.g. English to Spanish) with some meaning representation formalism mediating between the source language and the target language, or language-to-query translations (human language as database front ends). While written text has long been used for consulting knowledge bases (back to [1]), its use for representing knowledge itself lags behind. Database updates through language have been studied (e.g. [2]), but neither database creation through natural language nor the more daring idea of creating programs through language have been extensively explored.

Everyday there are more people using computers. However, just a few of them have the ability (or training) to create and consult a knowledge base. The use of natural language as a programming language helps the ocasssional or untrained user to communicate with the computer. Because natural language is the fundamental mean of communication, there is no special learning effort. The use of human language to create and consult a knowledge base reduces the human/database confusion and mistmatch becuase the users do not need to translate from their own language to a more restrictive one. Natural language is also flexible, powerful, and easy to use and to understand.

On the other hand, the use of uncontrolled natural language involves a high risk because it could be vague, imprecise, and in many cases ambiguous; leading to wrong interpretations and eventually to inconsistent knowledge bases. To avoid wrong interpretations, most data/knowldge bases use a formal language. Formal languages are unambiguous and precise because they use predefined syntax and semantics. However, formal languages are not natural, they use an unfamiliar notation which has to be learnt beforehand. Consequently, formal languages can be difficult to understand and hard to use.

Given the advantages and disadvantages of uncontrolled natural language vs. formal languages, it is natural to think in using the midpoint as the more natural but at some point restricted programming language. Controlled natural language combines the advantages of natural and formal languages and avoid most of their disadvantages. It is a restricted but expresive language which is processed efficiently by a computer.

Controlled natural language, in specific English, has been used for technical documentation [3], machine translation [4], requirements specification [5, 6], concept-based retrieval [7], and as database query language (front end) [8].

In this article, we discuss how to endow computers with the ability to translate a controlled human language, Spanish, into a knowledge base which can later be consulted also in the same controlled natural language.

## 2 From Spanish Sentences to Horn Clauses

To recover Horn clauses from sentences we need to distinguish the sentence's main verb in order to form the clauses's head. The rest of the sentence (subject, subject's complements and verb's complements) form the body, which is a conjunction of Horn clauses. For instance, the grammar shown in figure 1 could be used to obtain the Horn clause equivalent to the sentence *"mulder felicito a la nueva maestra / mulder congratulated the new teacher"*. The representation obtained by this toy parser is:

```
M = felicito(mulder,_x74902-[det,sing,fem]),
    nueva(_x74902),
    maestra(_x74902)
```

```
parse(Input,M):-#<Input,sent(M),#>[].

sent((Head,Body)):- noun_phrase(X,NP), verb_phrase(X,Head,VP),
                    append(NP,VP,Body).

noun_phrase(X,[]):-  name(X).
noun_phrase(X-Det,NP):- quant(Det), adj(X,A), noun(X,N), append(A,[N],NP).

verb_phrase(X,Head,VP):- verb(X,Y,Head), noun_phrase(Y,VP).

adj(X,[A]) :- adjective(X,A).
adj(_,[]).

quant([det,sing,fem]):- #la.
quant([und,plur,masc]):- #unos.

name(mulder):- #mulder.
name(carter):- #carter.

noun(X,maestra(X)):- #maestra.
noun(X,estudiante(X)):- #estudiante.

verb(X,Y,felicito(X,Y)):- #felicito, #a.
verb(X,Y,gano(X,Y)):- #gano.

adjective(X,nueva(X)):-#nueva.
```

**Fig. 1.** Toy Grammar: Translating NL Sentences into Horn Clauses

### 2.1 Relative Clauses: an Assumptive Approach

A relative clause is a subordinate clause which modifies what they are attached to (usually a noun or pronoun). A relative clause is usually introduced by a relative pronoun. The definition of the noun phrase should be changed to allow the use of relative clauses:

```
noun_phrase(X-Det,NP):- quant(Det),
                        adj(X,A),noun(X,N),
                        relative_pronoun,
                        relative(X,R),
                        appendN([A,[N],R],NP).
```

There are two types of relative clauses: restrictive and non-restrictive relative clauses. Restrictive relative clauses defines more closely what the noun modified by the clause is referring to. The sentence would be incomplete without the relative clause. In restrictive relative clauses, the subject is always missing but could be easily identified with the relative's antecedent (i.e. *"mulder felicito a la maestra que gano el premio / mulder congratulated the teacher who won the price").* Consequently, the relative clause is reduced to a verb phrase:

```
noun_phrase(X-Det,NP):- quant(Det),
                        adj(X,A),noun(X,N),
                        relative_pronoun,
                        verb_phrase(X,H,VP),
                        appendN([A,[N],[H],VP],NP).
```

```
M = felicito(mulder,_x56834-[det,sing,fem]),
    maestra(_x56834),
    gano(_x56834-[det,sing,fem],_x76503-[det,sing,masc]),
    premio(_x76503).
```

Non-restrictive relative clauses merely give additional information about the noun, where that information is dispensable. Sometimes another noun phrase than the subject is missing (i.e. *"la maestra que el director felicito enseña español / the teacher who the principal congratulated teaches Spanish").* To obtain a complete Horn clause, it is neccesary to pass the antecedent, which needs to be identified with the missing noun phrase, all the way to the (arbitrarily distant) place where the noun phrase is missing.

We have used the Assumption Grammar approach [9] to relate long-distant constituents. Assumptions are basically hypotheses obtained during the execution of a program. Those hypotheses are true during the entire continuation of the program or until they are proved wrong. When an assumption is no longer true, it disappears with backtracking. There are three types of assumptions: linnear, timeless, and intuitionistic. A linnear assumption can be consumed only once and it must be assumed (+X) before it could be consumed (-X). A timeless asumption can also be used at most once but it could be consumed (-=X) before is assumed (=X). On the other hand, an intuitionistic assumption is assumed (*X) just once but could be consumed (-X) many times.

Using assumptions we can define a relative clause as a sentence with a missing noun phrase preceded by a relative pronoun:

```
relative(X-Det,R):- relative_pronoun,
                +missing_np(X-Det),
                sentence((H,B)),
                append([H],B,R).
relative(_,[]).

noun_phrase(X-Det,NP):- quant(Det),
                        adj(X,A),
                        noun(X,N),
                        relative(X-Det,R),
                        appendN([A,[N],R],NP).
noun_phrase(X-Det,[]) :- -missing_np(X-Det).
```

```
      M = ensena(_x83042-[det,sing,fem],espanol),
          maestra(_x83042),
          felicito(_x93452-[det,sing,masc],_x-83042-[det,sing,fem]),
          director(_93452).
```

## 3 Controlled Spanish

Controlled natural language is defined as a subset of full natural language with a well-defined syntax and semantics. In this section we present the controlled Spanish (vocabulary and grammar) that could be used to create and query a knowledge base.

### 3.1 Vocabulary

The vocabulary is formed by two different classes: closed and open word classes. Closed word classes do not appear to be productive because new words cannot easily be coined in the class, or the class is growing extremely slowly (e.g. determiners, prepositions, conjunctions, etc.). On the other hand, in open word classes new words can be added anytime as the need arises (e.g. proper names, nouns, verbs, adjectives, adverbs, etc). Vocabulary reasonably common to all application belongs to the static part of our system. Vocabulary specific to each application is entered at definition time (e.g. *"andrea es un nombre propio femenino"* generates `name(andrea-[femenino,singular])`). Any time a new word is defined, it is also added to the corpus permanently. If a conflict arises, the last definition of the word is kept while the previous one is thrown away.

**Nouns** Nouns are divided in four different categories, according to their characteristics:

– Individualized nouns refer to active elements that performs actions and could be identified by a specific name. Those elements refer to individuals in the database. A variable is introduced to denote that element through the program (e.g. *'maestra'* will be denoted as `maestra(X)`). The program will look for a doctor in the database ( e.g. *"andrea es una maestra / andrea is a teacher"* generates `maestra(andrea-femenino)`.
– Generic nouns represent things that are usually associated to the verbs: "ser/estar/to be" or "tener/to have" (e.g. *silla/chair, pizarron/blackboard).* They do not correspond to named individuals. The program does not introduce a variable, instead the generic name is used and its characteristics are stored as arguments. To differentiate two instances of the same class, we should qualify the noun (e.g. *"la silla negra / the black chair"* is represented as `silla-[det,sing,fem]-[ser(negra)]` and *"la silla de andrea / andrea's chair"* is represented as `chair-[det,sing,fem]-[de(andrea)]`).
– Combined nouns have some characteristics of the two prevoius sets: they could be identified by a specific name but they are only associated with the verbs "tener/to have" or "ser/estar/to be". For every combined noun, a variable is introduced (e.g. *'escuela/school'* will be denoted as `escuela(X)`) but it has the same restrictions as generic nouns.
– Proper nouns are specific names given to person, animals, places or things. They could be associated to any verb but they do not need a variable to represent them because they are represented by constants.

**Verbs** Spanish is a very rich language that contains more than 15 tenses. The difficulty of obtaining accurate answers increases with the inclusion of each tense. In our system, we only allow three tenses: present, past or future in the active voice. Sentences must refer to a third person either singular or plural.

**Adjectives** An adjective is a word that describes or classifies a noun. In Spanish, most descriptive adjectives can be placed either before or after the noun. Demostrative and possessive adjectives are not allowed. When a new adjective is added, the user has to specify if it is regular or irregular and include all the available forms of the irregular case. The user also must specify if the adjective can be used before and after the noun or just in one position.

### 3.2 Grammar

Grammar defines the construction rules recongnized by the system. The use of a limited number of construction rules avoids imprecision and ambiguity. Three tyeps of construction rules are recognized by our prototype:

– Simple sentences have the general form: *subject + verb + complements.*
– Composite sentences introduced by relative pronouns.
– Query sentences: either yes/no question or wh-questions.

Synonyms are handled by associating the same meaning representation to alternative words in the lexicon. This technique, by the way, saves us from the need for expensive equality axioms to check whether two differently-named constants correspond to the same individual. Various parts of the sentences can be arrange in different ways and still produce the same meaning. This phenomenon is very common in Spanish, especially in questions. The following questions have the same representation: *¿Andrea resolvio la tarea con Karla?, ¿Resolvio la tarea con Karla Andrea?*

```
resolver(pasado,andrea,_x1555-[det,sing,fem],[con(karla)]):-tarea(_x1555).
```

Appendix I shows the Spanish grammar recognized by the parser.

### 3.3 Meaning Representation

In this section we present the logic system that the human language is translated into.

Every sentence is translated into a Horn clause term with two parts: the head and the body. The body is a collection of predicates generated by the noun phrase and/or verb phrase's complements. The head is represented by a tree based in its main verb, which has four branches with the following characteristics:

- The first argument is the tense.
- The second argument is the variable or generic noun introduced by the noun phrase.
- The third argument is the direct object if the verb is transitive or the value "none" if it is intransitive
- The last argument is a list that contains the representation of the prepositional phrase(s).

We already explained how nouns are represented. The proper determiner is attached to every noun (excluding proper nouns), with the following format: `[type,number,gender]`.

Adjectives, relative clauses and prepositional phrases are noun complements. Noun complements are mapped to new facts if the nouns are not generic. Otherwise, they are added to the noun as a list. For example, *"la vieja escuela / the old school"* is represented as `school-[det,sing,fem]-[ser(0-none,viejo)]` where the argument (`"0-none"`) represents the degree of the adjective. On the other hand, *"el nuevo director/ the new principal"* is represented as `nuevo(0-none,X-masculino)` where `"masculino"` represents the gender. Prepositional phrases as noun complements follow the same rules: *"el hijo de la maestra... / the teacher's son..."* generates `verb(Tense,Subject,Object,Complements):- hijo_de(Subject-masculino,Y-femenino), maestro(Y-femenino)`, but *"el escritorio de madera... / the wooden desk..."* is represented by `verb(Tense,escritorio-[de(madera)],Object,Complements)`.

Verb complements such as direct object and prepositional phrases represent the third and fourth arguments of the head of the Horn clause generated by the sentence. Examples: *"andrea ama a antonio / andrea loves antonio"* generates `amar(presente,andrea,antonio,[])`, *"andrea baila con el director / andrea* *dances with the principal"* is represented as `bailar(presente,andrea,none,[con(X-[det,sing,masc])]) :- director(X-masculino)`, *"andrea baila salsa con el director/ andrea dances salsa with the principal"* is translated as `bailar(presente,andrea,salsa,[con(X-[det,sing,masc])]) :- director(X-masculino)`.

Finally, the copula generates two clauses (e.g. *"andrea es una maestra / andrea is a teacher"* is represented as `maestro(andrea-femenino)` and `ser(presente,andrea, maestro,[])`). The second clause is used to answer questions like: *¿Que es Andrea?*.

## 4 Conclusion and Future Research

We have developed a system that allows the creation and consultation of knowledge bases using controlled Spanish. However, typing in the human language sentences necessary to create a knowledge base is time consuming and error-prone. With a good speech analyzer, dictating natural language sentences that represents a given corpus of knowledge becomes a much more attractive task. The next step is to integrate a speech recognition module.

We are the first generation with exponents that have spent twenty or thirty years working in front of a computer terminal, and the ill effects are all too visible around us: tendonitis; eye, neck and back strain; Carpal Tunnel syndrome. It is high time to develop alternative computer work modes than the present typing/screen based model. Speech-driven knowledge base creation and consultation could bring relief from such problems.

In this article we have proposed some emerging approaches towards such ends, and hope to have shown that logic programming holds special promise in this respect. More work needs to be done to create a complete and robust system. Currenlty we are working in the addition of more construction rules that includes anaphora, coordination, and ellipsis. The inclusion of more tenses and persons is being evaluated.

## References

1. Dahl, Veronica. Logical Design of Deductive, Natural Language Consultable Data Bases. In Proc. Fifth International Conference on Very Large Databases, Rio de Janeiro, Brazil (1979) 24–31
2. Davison, James. A Natural Language Interface for Performing Database Updates. In First International Conference on Data Engineering (ICDE), Los Angeles, California, USA (1984) 69–76
3. Wojcick, R. H., Hoard, J. E., and Holzhauser, K.C. The Boeing Simplified English Checker. In Proc. International Conference of Human Machine Interation and Artificial Intelligence in Aeronautics and Space, Toulouse, France (1990) 43–57
4. Mitamura, T. and Nyberg, E. Controlled English for Knowledge-based MT: Experience with the Kant System. In Proc. of the Sixth International Conference on Theoretical and Methodological Issues in Machine Translation, Leuven, Belgium (1995) 158–172
5. Fuchs, N.E., Schwertel, U., and Schwitter, R. Attempto Controlled English - Not Just Another Logic Specication Language. In Lecture Notes in Computer Science, Vol. 1559. Springer Verlag, Berlin, Heidelberg and New York (1999) 1-20

6. Macias, B. and Pulman, S. G. A Method for Controlling the Production of Specifications in Natural Language. In the Computer Journal 38:4 (1995) 310–318
7. Zaiane, O. R., Fall, A., Rochefort, S., Dahl, V., and Tarau, P. Concept-based Retrieval using Controlled Natural Language. In Proc. Workshop on Applications of Natural Language to Information Systems (NLDB97), Vancouver, Canada (1997)
8. Androutsopoulos, I., Ritchie, G. D., and Thanisch, P. Natural Language Interfaces to Databases – An Introduction. In Journal of Natural Language Engineering 1:1 (1995) 29–81
9. Dahl, V., Tarau, P., and Li, R. Assumption Grammars for Processing Natural Language. In Lee Naish, editor, Proc. of the Fourteenth International Conference on Logic Programming, MIT press (1997) 256–270
10. Aguilar-Solis, Dulce and Dahl, Veronica. Spanish as a Data Definition Language. In Proc. Second International Workshop on Spanish Language Processing and Language Technologies (SLPLT01), Jaen, Spain (2001) 247–253
11. Dahl, V., Tarau, P., Rochefort, S., and Scurtescu, M. A Spanish Interface to LogiMoo- Towards Multilingual Virtual Worlds. In Informatica, volume 2 (1999)

## Appendix I: Spanish Grammar

```
Sentence      ::= Noun_Phrase Verb_Phrase |
                  Open_¿ Noun_Phrase Verb_Phrase Close_? |
                  Open_¿ Verb_Phrase Noun_Phrase Close_?

Noun_Phrase ::=   Quant [Adj_Phrase] Noun [Adj_Phrase]
                  [Prep_Phrase] [Rel_Clause] |
                  Name [Rel_Clause] |
                  Question_Word

Verb_Phrase ::=   Intrasitive_Verb [Adv_Phrase] Prep_Phrase |
                  Transitive_Verb Noun_Phrase [Adv_Phrase] Prep_Phrase |
                  Transitive_Verb Noun [Adv_Phrase] Prep_Phrase |
                  Copula Noun_Phrase |
                  Copula Adjec_Phrase |
                  Copula Prep_Phrase

Prep_Phrase ::=   Preposition Noun_Phrase |
                  Prep [Adj_Phrase] Noun [Adj_Phrase] [Prep_Phrase]

Adv_Phrase ::=    [Degree] Adverb | Question_Word

Adj_Phrase ::=    [Degree] Adjective

Rel_Clause ::=    rel_pron Noun_Phrase Verb_Phrase |
                  rel_pron Verb_Phrase

Copula ::=        ser | estar
```

## Appendix II: Sample Creation and Consultation Session

```
              B I E N V E N I D O
> andrea es una doctora
  Agrego a la base de datos:
doctor(andrea-fem). ser(presente,andrea,doctor,[]).
Escribe adios para salir
> karla es una enfermera
  Agrego a la base de datos:
enfermera(karla-fem). ser(presente,karla,enfermera,[]).
Escribe adios para salir
>javier es un paciente
  Agrego a la base de datos:
paciente(javier-masc). ser(presente,juan,paciente,[]).
Escribe adios para salir
> andrea es la doctora de javier
  Agrego a la base de datos:
doctor_de(andrea-fem,javier-masc). ser(presente,andrea,doctor,[de(javier)]).
Escribe adios para salir
> betania es un hospital muy caro
  Agrego a la base de datos:
hospital(betania-masc). caro(2-muy,betania-masculino).
ser(presente,betania,hospital,[como(muy,caro)]).
Escribe adios para salir
> andrea trabaja en un hospital muy caro
  Agrego a la base de datos:
trabajar(presente,andrea,none,[en(_x43323-[ind,sing,masc])]):-
hospital(_x43323-masc), caro(2-muy,_x43323-masc).
Escribe adios para salir
>javier es el padre de karla
  Agrego a la base de datos:
padre_de(javier-masc,karla-fem). ser(presente,javier,padre,[de(karla)]).
Escribe adios para salir
> el padre de karla nacio en puebla
  Agrego a la base de datos:
nacer(pasado,_x38901-[det,sing,masc],none,[en(puebla)]):-
padre_de(_x38901-masc,karla-fem).
Escribe adios para salir
> karla nacio en vancouver
  Agrego a la base de datos:
nacer(pasado,karla,none,[en(vancouver)]).
Escribe adios para salir
> andrea opero a el padre de karla
  Agrego a la base de datos:
operar(pasado,andrea,_x63456-[det,sing,masc],[]) :-
padre_de(_x63456-masc,karla-fem).
```

```
Escribe adios para salir
> la doctora que opero a javier esta embarazada
  Agrego a la base de datos:
embarazada(_x23098-fem) :- doctor(_x23098-fem),
operar(pasado,_x23098-[det,sing,fem],javier,[]).
estar(presente,_x23098,embarazada,[que(operar(pasado,javier))]).
Escribe adios para salir
> el paciente que nacio en puebla murio de un infarto
  Agrego a la base de datos:
morir(pasado,_x49111-[det,sing,masc],none,[de(infarto)]):- paciente(_x49111-masc),
nacer(pasado,_x49111-[det,sing,masc],none,[en(puebla)]).
Escribe adios para salir
> la doctora que opero a el paciente que murio de un infarto nacio en puebla
  Agrego a la base de datos:
nacer(pasado,_x29001-[det,sing,fem],none,[en(puebla)]):- doctor(_x29001-fem),
paciente(_x30885-masc), morir(pasado,_x30885[det,sing,masc],none,[de(infarto)]),
operar(pasado,_x29001-[det,sing,fem],_x30885-[det,sing,masc],[]).
Escribe adios para salir
> ¿la doctora de javier trabaja en donde?
  Representacion:
trabajar(presente,_x56565-[det,sing,fem],none,[en(_x612121)]):-
doctor_de(_x56565-fem,juan-masc).
  Respuesta(s): betania
Escribe adios para salir
> ¿quien nacio en puebla?
  Representacion:
nacer(pasado,_x56565-_x93452,none,[en(puebla)]).
  Respuesta(s): javier | andrea
Escribe adios para salir
> ¿el paciente que murio de un infarto es el padre de quien?
  Representacion:
paciente(_x32895-masc), morir(pasado,_x32895-[det,sing,masc],none,
[de(infarto)]), padre_de(_x32895-masc,_x34902-_x30103).
  Respuesta(s): karla
Escribe adios para salir > ¿que es javier?
  Representacion:
ser(presente,juan,_x64789,_x69123).
  Respuesta(s): paciente, [] | padre, [de(karla)]
Escribe adios para salir
> ¿quien esta embarazada?
  Representacion:
embarazada(_x84234-_x88912).
  Respuesta(s): andrea
> adios
             H A S T A   L U E G O
```