

QUANTIFICATION IN A THREE-VALUED LOGIC FOR NATURAL LANGUAGE QUESTION-ANSWERING SYSTEMS

Veronica Dahl
Professor
Departamento de Matemática
Facultad de Ciencias Exactas
1428 Buenos Aires, Argentina

This paper examines quantification in relation to a typed three-valued logical system which was developed to serve as the internal query language for virtual data bases accepting natural language (NL) consultation. Such a system is capable, among other things, of reflecting certain NL presuppositions, handling relations among sets and coping with certain NL ambiguities, within a simple though natural and fairly vast NL subset.

In our approach, quantification is dealt with through a single mechanism by which all NL quantifiers introduce the formula "those(x,p)", denoting the set of all those x's in x's associated domain which satisfy statement p. The meaning of each particular NL quantifier-- including any presuppositions it may induce-- is rendered through particular constraints upon the set (e.g., cardinality constraints).

1. INTRODUCTION

The idea of using Logic as a conceptual framework in question-answering systems is not a new one. The fact that it can formally deal with the notion of logical consequence makes it particularly attractive to represent meaning.

Nevertheless, efforts in this direction had been more or less abandoned in Artificial Intelligence research, mainly because standard predicate calculus (PC) has been shown to be unsuitable for an accurate representation of NL. Its main inadequacies seem to relate to a general inability to incorporate relevant semantic features into the logical representation of a sentence.

Recent linguistic research ([1],[2]) has arrived at interesting results concerning the extension of standard PC in order to provide a better formal model of language.

It is our thesis that research in this direction, together with the fact that programming in logic ([3],[4]) has become possible since the development of the PROLOG programming language [5], is likely to yield some very good results concerning NL processing.

On the other hand, the evolution in data base technology has been drifting more and more towards the use of logic, both for data descrip-

tion and for queries [6]. Consequently, it is tempting to explore the possibilities of using logic throughout a NL data base system, in the hope of reducing interfaces to a strict minimum.

These considerations have led us to develop a rigorous logical system with a clearly defined semantics, which has been successfully used as the internal query language into which a computer translates NL input in order to consult different data bases ([7],[8],[9],[10]).

In our approach, a data base is actually an interpretation according to which a well-formed formula in our logical system is evaluated. Its evaluation directly produces the answer, since it yields either a truth value, which corresponds to a yes-no question, or the extensional representation of a set, which corresponds to a wh-question.

This paper focusses on our treatment of quantification within this logical system. The most important theoretically related work is [1],[2] and [11]. Related computer systems are mentioned throughout the paper.

Section 2 discusses the need of typing variables in connection with meaningfulness and ambiguity solving. Section 3 motivates our approach to quantification and the need of three logical values in connection with existential and number presuppositions, and then describes

the process of translating NL quantifiers into logical ones. Section 4 discusses the results obtained in concrete applications.

For space limitations, we are forced to omit a formal and complete definition of our logical system. Those of its features that are relevant to our subject, however, are intuitively described in Section 3.

Spanish is used throughout the paper as the concrete point of reference with which to exemplify our theoretical considerations.

2. MEANINGFULNESS, AMBIGUITY AND SEMANTIC TYPES

A NL processing system must have a means for checking semantic as well as syntactic accord, in order to reject anomalous sentences such as "Which dogs speak Latin?".

A widely spread solution to this problem consists in first generating a "deep structure" of the sentence, taking only syntax into account, and then performing all the necessary semantic operations and checkups on it.

As has already been observed [12], this often implies a tradeoff between syntactic and semantic complexity, when it is overall simplicity and efficiency that are important.

Moreover, linguists themselves are not unanimous as to whether the semantic component should be separate or intermingled with the syntactic one [13].

Where logic is concerned, there is a simple and elegant way of dealing with meaningfulness: by using types. Types, by the way, are also a useful means for associating the universe of PC to the relations in a particular data base. If we associate a type to each data base domain, and model the relations in a data base through predicates whose arguments are restricted to values in specific domains, the analyser can check that the arguments in each predicate generated by an input sentence are of the expected type.

Incidentally, types are also useful to improve efficiency: a) by narrowing the search space, since only those values in a variable's associated domain need to be considered, and b) by avoiding sterile data base consultations, since absurd queries can be rejected by the analyser on the grounds of domain incompatibility.

Types have been proposed by other authors (e.g. [14],[15],[16],[17]) in connection with efficiency and meaningfulness, but they have still another interest: they allow to automatically solve certain NL ambiguities. Take for instance the query:

¿Cuál es el salario del empleado que vive en Lomas?--Which is the salary of the employee who lives in Lomas?

A priori, a machine has no way to decide whether the antecedent of the relative clause is "the salary of the employee" or "the employee". But in a type-checking system in which the first argument of the relation "live" is associated to the HUMAN domain, and in which employees--and not salaries-- are known to belong to this same domain, the first alternative is not even possible.

Ambiguities concerning different aceptions of a word can generally also be solved through domain checking. This is particularly true in our implementation, in which variables are associated to a type contextually, during the analysis of an input sentence (cf. Section 3.1).

3. PRESUPPOSITIONS, QUANTIFIERS AND A THREE-VALUED LOGIC

Typed calculus in itself is not enough to make all sentences meaningful. We actually need more than two logical values, since in NL there are two ways in which a statement may fail to be true: either because its negation holds, or because something presupposed by the statement fails to be satisfied. In the latter case, the statement is felt to be pointless rather than false.

There is another reason why it must not be considered false. Take for instance the statement:

El sombrero loco odia a Alicia.
The mad hatter hates Alice.

In a context in which no hatter is mad, it is obviously not true. However, we can not either consider it false, since then the statement

El sombrero loco no odia a Alicia.
The mad hatter does not hate Alice.

would have to be considered true.

The inexistence of a referent for the definite noun phrase makes the whole sentence meaningless. The existence of more than one referent

would also make it meaningless, since we could not tell which of the referents the sentence is talking about. This is because the Spanish singular definite article induces a presupposition of existence and unicity upon the noun phrase's referent.

Our treatment of quantification has been devised to account for those presuppositions induced by NL quantifiers. We prefer to call them "determiners", since they include all articles, cardinal numbers and words such as "some", "many", etc.

If a sentence contains a determiner, a quantification of the form "those(x,p)" is introduced, where x is a typed variable and p is a logical formula in our system. Its evaluation yields the set of all x's in x's associated domain which satisfy p.

According to the determiner's meaning, number and existence presuppositions on such a set are represented within the output formula. For instance, "Three blind mice run" is represented as

```
card(those(x, and( and(mice(x), blind(x)), run(x))), 3)
```

(the cardinality of the set of those blind mice who run is 3)

Definite articles introduce the formula "if(f₁, f₂)", whose value is "undefined" whenever f₁ fails to be satisfied, and has the same value as f₂ if f₁ is true. Here is an example, using the easier-to-picture tree representation:

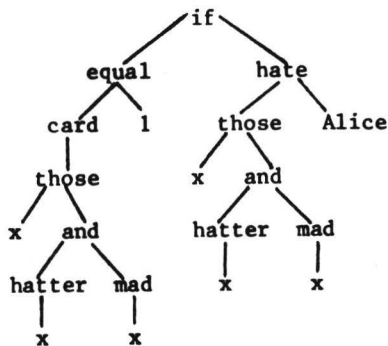


Fig. 1 Representation of the sentence "The mad hatter hates Alice"

The formula represented in Fig. 1 will evaluate to "undefined" if the set of mad hatters does not contain exactly one element.

3.1 Spanish determiners and their translations

We can now examine the general process by which a determiner introduces a "those" formula. Let us consider a sentence consisting of a noun phrase followed by a verb phrase, in which the noun phrase contains a noun introduced by a determiner. We can first represent it through a three-branched quantification of the form:

$$q(x, f_1, f_2)$$

where q is a quantifier into which the determiner translates, x is a typed variable, f₁ is the noun phrase's translation, and f₂ is the verb phrase's translation. Intuitively, f₁ specifies the domain of quantification, and q states what portion of the domain f₂ holds for.

Our previous example, for instance, can first be represented:

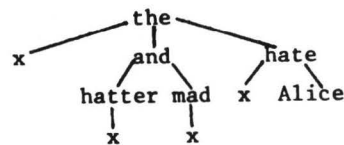


Fig. 2 A first representation of "The mad hatter hates Alice"

In our implementation, a variable's associated domain depends both on f₁ and f₂ in the following manner: each predicate known to the data base system has a domain associated to each of its arguments. For instance, "mad" and "hatter" could require its arguments to belong to the HUMAN domain, while "hate" could require its first argument to belong to the ANIMAL domain.

When a three-branched quantification is generated, the variable it creates is typed by the intersection of all those domains it has been associated with by the predicates appearing in either f₁ or f₂. In our example, x's type would be HUMAN (the intersection between the HUMAN and the ANIMAL domains).

Instead of generating a different quantifier for each determiner, it is useful to represent all quantifications through a single one of

the form

```
para(x,p,c)
for
```

whose intuitive meaning is: "c holds for the set of all x's in x's domain which satisfy p".

Each quantification is thus assigned an equivalent "for" expression, in which the determiner's meaning is represented. Here are the representations of the Spanish determiners "un" (a), "unos" (some), "el" (singular the), "los" (plural the), "ningún" (no) and "i" (any cardinal number). The rest are considered in Section 3.1.2.

```
un(x,f1,f2)=unos(x,f1,f2)=for(x,and(f1,f2),
greater-than(card(x),0))
el(x,f1,f2)=for(x,f1,if(equal(card(x),1),f2))
los(x,f1,f2)=for(x,f1,if(greater-than(card(x),
1),f2))
ningún(x,f1,f2)=for(x,and(f1,f2),equal(card(x),
0))
i(x,f1,f2)= for(x,and(f1,f2),equal(card(x),i))
```

Note that we have chosen to identify "un" (a) and "unos" (some). This is because "un" is frequently used to denote "at least one". In order to avoid ambiguity, "1" should be used to mean "exactly one". This convention is particularly useful when negation is involved. For instance, "No tengo un centavo" (I have not a cent) would have a wrong representation with the "exactly one" interpretation: it would state that the cardinality of the set of cents I possess is not 1, which means it can either be 0,2,3,etc.

Finally, any formula of the form "para(x,p,c)" can be replaced by just the formula c, in which all occurrences of x have been replaced by the formula:

```
aquellos(x,p)
those
```

representing the subset of x's domain whose elements satisfy p.

The reader can now verify that the representations shown in Figs. 1 and 2 are equivalent.

3.1.1 Quantifier hierarchy

Quantifier hierarchy obeys three perhaps too simplistic rules, but which have proved to be useful within our NL subset *.

- Rule 1: A determiner in a verb's subject introduces a quantification which dominates all quantifications introduced by the verb's complement(s). For instance, "Toda rosa tiene (algunas) espinas" -- Every rose has (some) thorns-- is represented:

```
toda(x,rosa(x),algunas(y,espinas(y),tiene(x,
every rose some thorns has
y)))
```

Note that the representation

```
some(y,thorn(y),every(x,rose(x),has(x,y)))
```

would be incorrect, since it rather means: "There exists a certain set of thorns which every rose has".

- Rule 2: Whenever a noun has a complement, the quantification introduced by the complement's determiner dominates the one introduced by the noun's determiner. For instance, "Sábado autografía el libro de cada visitante" -- Sábado autographs the book of each visitor-- is represented:

```
cada(x2,visitante(x2),el(x1,libro-de(x2,x1),
each visitor the book-of
autografía(Sábado,x1)))
autographs
```

- Rule 3: When a referential word (that is, a verb, a noun or an adjective) has more than one complement, quantification takes place from right to left: the rightmost complement generates a quantification which dominates the quan-

* As a formal characterization of our NL subset is beyond the scope of this paper, we shall informally state that it has a fixed vocabulary--determiners, prepositions, the conjunction "y" (and), relative and interrogative pronouns and the word "no"; and a variable vocabulary--nouns, proper names, adjectives and verbs in the third person of a simple tense. Sentences are either declarative or interrogative, in the active voice, and restricted relative clauses are allowed and can be nested with no other limit than the computer's capacity.

tification(s) introduced by the leftmost complement(s). For instance, "Raúl regala un espejo a cada niño"-- Raúl gives a mirror to each child-- is represented:

```
cada(x2, niño(x2), un(x1, espejo(x1), regala(Raúl,
each child a mirror gives
x1, x2)))
```

3.1.2 Determiners with a negative implication

As a general rule, the negation introduced by "no" in a sentence is translated by placing the operator "no" (not) right after the quantification introduced by the subject. For instance, "Los chicos no conocen al padre de Martín" (The children do not know Martín's father) is represented:

```
los(x1, chicos(x1), no(el(x2, padre-de(Martín, x2),
the children not the father-of
conoce(x1, x2)))
know
```

But negation is not always explicit. The Spanish determiner "ningún" (no) can be regarded as an implicit negation, since it expresses that no portion of the domain of quantification satisfies the statement involved.

In a non-inverted subject position (e.g., "Ningún elefante vuela"-- No elephant flies--), it generates a special quantifier called "ningún", whose representation takes this fact into account (see Section 3.1).

There are two other cases, however, in which "ningún" coexists with an explicit negation. These cases require a different quantifier, since otherwise the negation would be represented twice. These cases are:

a) In a subject position, plus subject-verb inversion: the "ningún" determiner is assimilated to the "every" quantifier. For instance, "No vino ningún hombre"-- No man came-- is represented:

```
todo(x, hombre(x), no(vino(x)))
every man not came
```

b) In a position other than the subject: the "ningún" determiner is assimilated to the indefinite article's quantifier. For instance, "Carlos no tiene ningún hijo"-- Carlos has not any child-- is represented:

```
no(un(x, hijo(x), tiene(Carlos, x)))
not a child has
```

Another special case is the negation preceding the "todo" (every) determiner (e.g. "No todo pájaro canta"-- Not every bird sings--). The analyser considers "no todo" as a single determiner generating its own associated quantifier:

```
no-todo(x, f1, f2) = for(x, and(f1, not(f2)), greater-
than(card(x), 0))
```

4 SOME PRACTICAL CONCLUSIONS

The ideas previously discussed have been confronted with several successful PROLOG mini-computer implementations of NL consultable data base systems, in which NL analysis is performed through a metamorphose grammar [18], a powerful PROLOG tool for defining context-sensitive rewriting rules.

The first one concerned just one particular application: the representation of the hardware and software catalogues for the SOLAR 16 series of computers, in order to generate, from a user's question in French, the characteristics of a configuration suiting his particular needs ([7], [8]).

Our first conclusions were reinforced and improved by further research concerning a more general data base system implemented both for Spanish and French consultation ([9], [10]). The Spanish version of this system has been adapted to Portuguese by H. Coelho and L. Pereira (personal communication, 1978). Their work confirms our expectations regarding the generality of our ideas-- and of our implementation--, since remarkably few modifications of the program proved necessary, and they were strictly related to lexical differences, whereas rules concerning structure remained unchanged.

Similar ideas to the ones discussed above concerning quantification have inspired other NL data base systems, namely LSNLIS [19] and PHLIQAI ([17], [20]). But in spite of the points in common, our general approach is markedly different. We have tried to incorporate all relevant semantic as well as syntactic NL features into a single formalism, in order to do without intermediate sublanguages and have a single process perform the analysis of an input sentence. LSNLIS, on the contrary, first generates deep structures and then maps them into a semantic representation. PHLIQAI has several successive levels of semantic analysis, each requiring a special formal language. Some of them are meant to deal with ambiguity, which in our approach, as we have seen, is dealt with through the contextual typing of variables during the quantification process.

Another important point in our approach, which for reasons of space we must be content with just stating, is a high degree of formalization through logic: because of the use of PROLOG, important theoretical aspects such as a rigorous characterization of our NL subset and of the syntax and semantics of our internal query language need not be dissociated, as is generally the case, from those practical aspects concerning the implementation.

From the performance point of view, no frustrating delays in answering have been encountered so far in the systems we have implemented, and the conciseness obtained is rather great: the complete listing of our Spanish analyser, including the vocabulary associated to a small sample data base, holds in just four pages.

Among our limitations, let us mention the fact that we have not attempted to use NL for data base creation or modification, and that interrogation is limited to the context of one isolated sentence which can not refer to other previous questions or answers.

ACKNOWLEDGEMENTS

The research partially reported here was conducted at the Aix-Marseille II University, under the supervision of A. Colmerauer.

REFERENCES

- [1] Keenan, E. L. "On semantically based grammars". Linguistic Inquiry, 1972.
- [2] Hausser, R. "Quantification in an extended Montague grammar". Dissertation, University of Texas at Austin, 1974.
- [3] Kowalski, R. A. "Logic for problem solving". DCL Memo 75, Department of Artificial Intelligence, Edinburgh, March, 1974.
- [4] Van Emden, M. H. "Programming with resolution logic". Report CS-75-30, Dept. of Computer Sciences, Univ. of Waterloo, November, 1975.
- [5] Roussel, Ph. "Prolog: manuel de référence et d'utilisation". GIA, Univ. d'Aix-Marseille II, September, 1975.
- [6] Gallaire H. and Minker J (eds.) Logic and data bases. Plenum Publishing Corporation, N.Y., 1978.
- [7] Dahl, V. and Sambuc, R. "Un système de banques de données en logique du premier ordre, en vue de sa consultation en langue naturelle". DEA report, GIA, Université d'Aix-Marseille II, October, 1976.
- [8] Colmerauer A., Dahl V. and Sambuc, R. "Consultation en français d'un catalogue in-

dustriel: configuration de la série d'ordinateurs SOLAR 16". Final report of contract BNIST N°291767, GIA, Univ. d'Aix-Marseille II. To appear.

- [9] Dahl, V. "Un système déductif d'interrogation de banques de données en espagnol". These de Doctorat de Spécialité en Intelligence Artificielle, GIA, Univ. d'Aix-Marseille II, November, 1977.
- [10] Dahl, V. "Some experiences on natural language question-answering systems". Preprints of the international workshop "Logic and data bases", CERT, Toulouse, November 1977, XXVIII, pp. 1-9.
- [11] Colmerauer, A. "An interesting natural language subset". Preprints of the international workshop "Logic and data bases", CERT, Toulouse, November, 1977, XXIX, pp. 1-32.
- [12] Petrick, S. R. "On natural language based computer systems". IBM Journal of Research and Development, July, 1976.
- [13] Lakoff, G. and Ross, J.R. "¿Es necesaria la estructura profunda?" In Semántica y sintaxis en la lingüística transformatoria. Alianza Editorial de Madrid, 1974, pp. 226-231.
- [14] Mc. Skimin, J. and Minker, J. "A predicate calculus based semantic network for question-answering systems". Preprints of the workshop "Logic and data bases", CERT, Toulouse, November, 1977, VI, pp. 1-37.
- [15] Pirotte, A. and Wodon, P. "A comprehensive formal query language for a relational data base: FQL". R.A.I.R.O. Computer Sciences, 2:2, 1977.
- [16] Reiter, R. "Deductive question-answering on relational data bases". In: [6].
- [17] Scha, R. "Semantic types in PHLIQA1". In Proc. 6th International Conf. on Computational Linguistics, Ottawa, 1976.
- [18] Colmerauer, A. "Grammaires de métamorphose". To appear in: Natural language communication with computers. Lecture Notes in Computer Science, Springer Verlag.
- [19] Woods, W.A., Kaplan, R.M. and Nash-Webber, B. "The Lunar Sciences Natural Language Information System: Final report", BBN Report 2378, Bolt Beranek and Newman, Inc, Cambridge, Mass, 1972.
- [20] Landsbergen, S.P.J. and Scha, R.J.H. "Formal languages for semantic representation", M.S. 10.259, Philips Research Laboratories, Eindhoven, Holland.