AN IMPLEMENTATION OF FORMAL SEMANTICS IN THE FORMALISM OF RELATIONAL DATABASES.

Claire VANDERHOEFT
Département de Linguistique Générale, CP175
Université Libre de Bruxelles
50, avenue Roosevelt
1050 Bruxelles
Belgique

E-mail: X00106@BBRBFU01.bitnet

ABSTRACT

This paper presents an implementation of **formal semantics** as described in Keenan and Faltz's *Boolean Semantics for Natural Language* [4]. The main characteristic of this implementation is that it avoids the intermediate step of translating NL into a formal language, such as an extended version of predicate calculus. My choice of not using any intermediate language, which Montague already suggested in *Universal Grammar* [5], makes my implementation free of the problems related to the syntax of such a language like binding the variables and resolving scope ambiguities. On the other hand, not translating NL into an intermediate language requires **every denotation** (i.e. semantic value) to be explicitly and accurately represented in a database.

0. INTRODUCTION.

In extensional semantics, each denotation corresponds to an object of the world. The world is the set of all the denotations. In the implementation that I shall present in this paper, the world will be represented by means of a database, more precisely a **relational database**.

The structure of the database is designed in such a way that it makes explicit the **semantic type** of each denotation. Although I will not always stick to the standard version of formal semantics when assigning semantic types to syntactic categories, I aim at accounting for the same range of phenomena that formal semantics deals with.

The paper will be divided into five parts. First, I shall trace back research results to which my contribution can be related. Next, I describe the database. Then, I explain how the principles used to design the database meet the requirements of formal semantics. The fourth part is concerned with entailment while the last part mainly shows how one proceeds to interpret sentences.

1. BACKGROUND.

The topics which this work is concerned with have mainly been studied from three points of view.

A first class of studies covers the problems encountered in trying to translate NL into a formal language. On the one hand, there is theoretical research aiming at such a translation, like PATR [7]. On the other hand, various kinds of inaccuracies of NL translations into logical form in view of accessing databases have been discussed, see [6] for example.

A second field of research that need be mentioned is concerned with NL interfaces. Famous systems are described in [9] and [1]. There are important differences between these systems and my work since I am not aiming at accessing a knowledge base at all. The database that I use encodes NL meanings and it does so according to linguistic constraints. Traditionally, the database rather encodes a certain knowledge independent of the language used to talk about it. Problems specific to NL interfaces can be found in [3] and [8].

From another point of view, there are works which are concerned with the question of the organisation of the knowledge base constituted by NL meanings, see [2]. The difference between my approach and ones like [2], is that I am sticking to the theory of formal semantics. Consequently, I do not (yet) address questions about the structure of the

lexicon nor do I treat pragmatic phenomena like common sense inferences.

2. THE DATABASE.

The structure of the database is dependent on the semantic properties of the denotations. More specifically, the structure of the database is dependent on the fact that denotations are classified into different **types** and specifically recognized as the denotations of such and such syntactic categories.

Each denotation of each constituent is a value in the database. Some of the denotations result from the composition of other denotations. Which denotations can be composed with which other ones are properties of their type. These properties are not encoded as such. The overall structure of the database shows how the semantic types combine with each other. Consequently, complex denotations (denotations of complex expressions) are represented by **atomic** values, but the fact that they are complex is deduced from the structure of the database. Consider the case of noun phrase denotations. The denotation of a determiner combines with the denotation of a common noun. This combination yields the denotation of a noun phrase, i.e., an atomic value in the database. The representation of this denotation is **connected** (in the sense of relational databases) to the representations of the denotations of the noun and of the determiner. Therefore, it can be recognized as a complex denotation.

The design of the database is dependent on the fact that we need an explicit means to recognize the type of each denotation represented in it. Within the formalism of relational databases, defining types of denotations amounts to defining a **relation** for each such type.

A **relation** is formally defined as an n-tuple of formal attributes. By **formal attribute** is meant a way to identify the attribute (a position in the relation or a name) and the definition of the set of its possible values. The **extension** of a relation is the set of all well-formed n-tuples of attribute values for the corresponding formal attributes. (An ill-formed n-tuple has at least one non-possible value for a formal attribute.)

Relations each represent a type. Each of them has (at least) one attribute whose domain is the set of denotations of the corresponding type. For example, the relation which corresponds to the type of noun phrases has an attribute whose values are noun phrase denotations. Each denotation is an **atomic** value of the attribute of the relation. Furthermore, each such value actually belongs to an n-tuple belonging to the extension of the relation. (This is due to the fact that the set-theoretical model of the world, i.e. the database, contains all the denotations built on an ontology constituted by a set of entities, U, and the truth values.)

The structure of the database captures the (degree of) complexity of a denotation by connecting the relation which represents the semantic type assigned to the corresponding syntactic category with the relations which represent the semantic types assigned to the constituents of a complex expression of the same syntactic category. For example, a proper name has a complex denotation because it belongs to the syntactic category of noun phrases. Therefore, its denotation belongs to the extension of the relation representing the type of noun phrases. Since there are noun phrases constituted by a determiner and a common noun, the relation representing the type of noun phrases actually connects to the relations representing respectively the types of determiners and of common nouns. Hence, the structure of the relation associated to the type of noun phrases and, in particular, of proper names, shows that they are complex expressions.

To the extent that we need to define the connections which show the respective complexity of each type of denotation, a relation is actually defined for each semantic type. For example, we shall define relations like Tn, Tdet, Tnp, Tvp standing, respectively, for the type of denotations of common nouns, determiners, noun phrases, verb phrases.

Still, there is a problem in defining connections. The problem is that, in formal semantics, expressions of different syntactic categories can have the same semantic type. For example, common nouns and intransitive verbs share the same type. Now, the complexity of the denotation of a noun phrase is encoded in the fact that it is connected to the denotation of a common noun. On the contrary, the denotation of a verb phrase must be connected to the denotation of a simple verb and to the denotations of complements. In general, we not only need to define relations as counterparts of semantic types, but, where expressions of different syntactic categories collapse into the same type, their types must nevertheless correspond to different relations.

With respect to the example, the relation Tv (the type of simple verbs) cannot be the same as the relation Tn, because Tvp connects to Tv while Tnp connects to Tn.

Notice that it is true of all the syntactic categories that they have one and only one relation as semantic counterpart. Sometimes however, the relation could be defined as the sum of several relations. For example the complex relation Tvp has several mutually exclusive sets of connections. It connects to the relation Tv and the relation Tnp, or to the relation Tv and the relation Tpp or the relation Tvp and the relation Tpp, etc.

Let me illustrate these principles by showing the definition of two relations. The Tnp relation is defined as a triple of formal attributes:

Tnp = <**[np]**,Tn,Tdet>

where it is understood that the possible values of the first attribute are noun phrase denotations, the possible values of the second attribute are pointers to noun denotations, and the possible values of the third attribute are pointers to determiner denotations. Notice that proper names have dummy values for Tn and Tdet.

Relations which encode simple types, i.e. types of lexical categories, cannot be encoded the same way as relations corresponding to complex expressions: they have no connections since they do not have any constituents. Instead, they are generally defined by pairs of attributes, the first one instantiates to a denotation of the type in question and the second one to the symbolic expression which is the item. For example, Tn, which represents the type of simple common nouns, will be defined by the pair:

Tn = <**[n]**,"n">

where it is understood that the possible values of the first attribute are common noun denotations while the possible values of the second attribute are the nouns themselves considered as symbolic expressions. As expected, the role of the second attribute in a relation such as Tn is to anchor the denotation of simple expressions into the lexicon.

In summary , the design of the database meets the two following principles:

- i) relations that correspond to types of lexical categories have two attributes: the first one has as domain the set of denotations of all the lexical items which belong to the lexical category in question, and the second one has as domain those items themselves regarded as symbolic expressions.
- ii) relations that correspond to types of non-lexical categories have one attribute whose domain is the set consisting of all the denotations of all these expressions. Moreover, they have other attributes, one for each of their constituents. These attributes have as domain the extension of the relations corresponding to the types of these constituents.

These principles ensure that denotations are submitted to the principle of **compositionality** which states that the denotation of a complex expression is a compound of the denotations of its constituents. (It is important to understand that we want to be able to check that denotations are submitted to compositionality.) How compositionality constrains the definitions of the relations will now be illustrated on Tn and Tnp.

In the extension of a relation like Tn, all the pairs of values have the property that the first value is the denotation of the second one. We augment the schema of the database with the constraint on Tn that: [["n"]] = **[n]**. In the extension of relations like Tnp, all the n-tuples are required to satisfy the constraint that the value of the first attribute, i.e. the np denotation, is the denotation of an np whose constituents, i.e. the determiner and the noun, have as respective denotations the ones connected to by the remaining attributes of the n-tuple. For Tnp, we augment the schema of the database with the constraint that: $\mathfrak{C}$(Tn,Tdet) = **[np]**, where $\mathfrak{C}$ operates the composition of its arguments.

3. THEORETICAL PRINCIPLES MET BY THE DATABASE.

Let us summarize how principles of formal semantics are taken into account in designing the database:

i) we restrict ourselves to extensional semantics. Therefore, every denotation must be represented and must correspond to one object of the (unique) world.

ii) the word is the smallest unit that receives a denotation, the sentence is the biggest one.

iii) all the expressions that are well-formed syntactic constituents have a denotation.

iv) all the denotations are encoded in the extension of a specific relation, that is, all the denotations have a type defined in the database.

v) denotations of complex expressions are connected to the denotations of the constituents that are contained in those expressions.

The theory of databases states that relational databases are logically equivalent to a first order language whose predicates are the relations. In this first order language, the extensions of the predicates are the sets of tuples of argument values on which the "relation-predicates" evaluate to true. Therefore, the fact that the database is interpreted as a first order language ensures that all the denotations have a type and their type is explicitly attached to them.

4. ENTAILMENT.

What kinds of things are the denotations is indispensable to know in order to define what it means for an expression to entail another expression (of the same category).

Let us distinguish between attributes that point to other relations, attributes that are instantiated to symbolic expressions and attributes that take as values the denotations of the type represented by the relation they are attributes of. Only the last kind of attributes are concerned with entailment.

According to formal semantics, attribute values that represent denotations are sets. (Do not forget that they are atomic from the point of view of the structure of the database.) Some (primitive) **entities** are implicitly defined. Then, all the denotations (except for the denotations of sentences) are sets of entities, or sets of sets of entities, or functions whose domain and range are such kinds of sets. Let me use the meta-variable X which ranges over the sets of entities, while Y ranges over sets of sets of entities. The set structure of the world is the following:

Tn	X
Tnp (np's in subject position)	Y
Tnpconnection(complement np's)	X -> X
Tv	X
Tvp	X
Tdet	X -> Y
Tprep	Y -> (X -> X)
Tpp	X -> X
Tadv	X -> X

What is entailment, in the implementation? First, for expressions which denote functions, the fact that a certain expression entails another one is given by the fact that the respective expressions in which each of them appears (with other constituents) entail each other. For example, we will not say that "most" entails "some", but rather that "most Xs" entails "some Xs". This being so, functions can be represented by symbols, either names (the lexical items) or connections. Now, representing functions by symbols rather than by the sets of pairs argument-result implies that entailment **cannot** be defined on the relations representing functional types.

For relations not corresponding to functional types, i.e. Tn, Tnp, Tv and Tvp, entailment is defined by means of set inclusion. Let t1 and t2 be two tuples of Tnp, t1 entails t2 if the attribute value which is the np denotation in t1 is a subset of the attribute value which is the np denotation in t2. Take another example. Assume that Tvp has four attributes: the first one is the denotation of the vp, e.g. *eat an apple*, the third one is the denotation of the verb without the complement whose denotation is the value of the fourth attribute, i.e. *eat* :

Tvp(**[eat an apple]**,W,**[eat]**, **[an apple]**)

Now, anything that has necessarily the property of eating an apple has the property of eating. So, for Tvp=(y1,W,y2,Z), we set the general constraint that $y2 \supseteq y1$. We will say that $y2 \supseteq y1$ is an axiom that belongs to the definition of the Tvp relation. Crucially, objects that would not meet the axioms characterizing the extension of the relation to which they belong cannot correspond to objects of the world. In the example, if *eating an apple* does not entail *eating*, then the two expressions fail to have acceptable denotations.

5. THE SKETCH OF A SEMANTIC INTERPRETOR AND SENTENCE DENOTATIONS.

The denotations of sentences are truth values. I have not insisted on the way sentence denotations are encoded but one might expect that there is a relation Ts = <**[s]**,Tnp,Tvp>. Although this agrees with the principles of the database, it is not the solution that I have adopted.

Assume that there is no Ts relation. We must nevertheless ensure that the interpretor will provide sentences with the truth values they denote. Furthermore, we would like to show how these truth values depend on the denotations of the constituents of the sentence because we want to represent how compositionality is respected.

The basic operation for interpreting an expression, that is for assigning it its denotation, is the selection of values in the database. How is the interpretor meant to select denotations actually encoded in the database?

The interpretor proceeds in parallel with a syntactic parser which yields (at least) the constituent structure of the expressions. Imagine that the information that the parser sends to the interpretor is the context-free rule used by the parser in parsing such expression to be interpreted. For example, suppose that the rule:

np -> det , n

parses the given expression. And suppose that the interpretor knows that the category np is the category of an expression of the type of noun phrases, hence of the relation Tnp. Likewise, n corresponds to Tn and det to Tdet.

Knowing that the above context-free rule applies, the interpretor can perform the selection of a tuple in the Tnp relation. The schema of the selection to perform will be written: $\Sigma_{Tn,Tdet}(Tnp)$. The specific selection to perform in order to interpret a specific noun phrase requires the interpretor to instantiate the parameters of the selection, Tn and Tdet, to the denotations of the noun and the determiner, respectively. The output of the instantiated selection:

$\Sigma_{[men],[most]}(Tnp)$

is:

$Tnp([\textbf{most men}],[men],[most]) \wedge$
[most]([men]) = [most men]

Notice that I use a logical notation to note the output of the interpretor. This output is, itself, **a relational database** containing one relation, the extension of which is constituted by one tuple which satisfies the second conjunct of the formula. In the example, the tuples consists of attribute values such that **[most]([men]) = [most men]** is true.

The way the interpretor selects the denotation of a noun phrase can be easily generalized to the other types of expressions. I immediately turn to the case of sentences.

Let us assume that there is only one rule to parse sentences, namely :

S -> np,vp

Since there is no Ts type, there is no selection. Nevertheless, the pseudo-schema of selection corresponding to this rule is defined by

$\Sigma_{Tvp,Tnp}()$

(to use a notation coherent with the one used for the other selections). The denotation that such a "selection" will yield is a formula interpretable as the truth value denoted by the sentence. I shall represent this formula by $\mathfrak{E}$.

The way in which $\mathfrak{E}$ is assumed to yield either truth value conforms to the account of standard formal semantics: it consists in checking whether the property, i.e. the set, denoted by the verb phrase is a member of the set of properties denoted by the noun phrase.

The outputs of $\Sigma_{Tvp,Tnp}()$. are more than just $\mathfrak{E}$. Indeed, in order to show that compositionality is respected, we must show explicitly what are the denotations of constituents which combine to yield the truth value. The latter denotations involve denotations of their own constituents. Therefore, the denotation of a sentence will be logically represented by an **assertion**. This assertion is the logical conjunction of the denotations of all the constituents of the sentence. For example, the sentence: *Most men eat an apple* denotes:

Tdet("most") $\wedge$ Tn(y1,"men") $\wedge$
Tnp(y2,y1,"most") $\wedge$ **[most]**(y1) = (y2) $\wedge$
Tv(y3,"eat",1) $\wedge$
Tdet("a") $\wedge$ Tn(y4,"apple") $\wedge$
Tnp(y5,y4,"a") $\wedge$ **[a]**(y4) = (y5) $\wedge$
Tvp(y6,"eat",y3,y5) $\wedge$ y3 $\supseteq$ y6 $\wedge$
y6 ε y2

where $\mathfrak{E}$ = y6 ε y2.

It is easy to predict that sentences having the same constituent structure as *Most men eat an apple* will each be interpreted by an assertion of the same form as this one.

The computational counterpart of such an assertion is a database contained in the original database. (We call such a database a **view** in the computational terminology.) In summary, all the sentences that share the same syntactic structure denote assertions equivalent to databases having the same structure (but different extensions, of course). Thus, the denotation of a sentence has iconic properties and its structure is of the same kind as that of the representation of the world. We shall say that it is a **possible fact**, where "fact" means that the denotation of the sentence is a part of the world, while "possible" means that its structure conforms to that of the world.

Since $\mathfrak{E}$ has been defined independently from any relation of the database, false sentences can have the same kind of denotation as do true sentences. By this, I want to emphasize the fact that, when $\mathfrak{E}$ does not yield the value true, it does not follow that the assertion is ill-formed. On the contrary, the fact that a false sentence fails to denote the

actual state of the world does not prevent it from denoting a possible fact as long as its denotation is a well-formed assertion.

6. CONCLUSION.

I have presented the main principles of my implementation of formal semantics. If I had more space to do so, I could now develop two crucial issues: I could show how the process of interpreting an expression parallels its syntactic parsing and prove that the structure of the database allows to cover the same range of phenomena as formal semantics does.

Other important issues that must be dealed with include further phenomena related to coordination, negation, passive and other constructions in which quantifiers appear to have a non trivial behavior. I am currently pursuing these developments on the basis of empirical linguistic data. My hypothesis is that they can be accounted for without changing the design of the system presented so far, and even without augmenting it much.

Let me finally stress that it is useful to know that the database is logically equivalent to a first-order language. Indeed, this fact gives a synthetic view of the behavior of the system and allows us to envisage further developments in the direction of non-monotonic logic.

ACKNOWLEDGMENTS.

The author is thankful to Prof. M. Dominicy for providing the opportunity to conduct this research. This text presents research results which were supported by the Belgian National incentive-program for Fundamental research in artificial intelligence initiated by the Belgian State, Prime Minister's Office, Science Policy Programming The scientific responsability is assumed by the author.

REFERENCES.

1- Grosz, B.J., Appelt, D.E., Martin, P.A. and Pereira, F.C.N. (1987). TEAM: An experiment in the Design of Transportable Natural-Language Interfaces. *Artificial Intelligence*. Vol. 32, No2, May 1987. 173-244.

2- Hobbs, J.R. (1984). Building a Large Knowledge Base for a Natural Language System. *Proceedings of Coling84*. 283-286.

3- Jones, K.S. (1984). Natural Language and Databases, Again. *Proceedings of Coling84*. 182-183.

4- Keenan, E. and Leonard M. Faltz. (1984). Boolean Semantics for Natural Language. D. Reidel Piblishing Company, Vol 23.

5- Montague, Richard. (1974b). Universal Grammar. In R. Thomason (ed.) *Formal Philosophy. Selected Papers of Richard Montague.* Yale University Press. New Haven and London. 222-246.

6- Moore, R.C. (1982). Natural-Language Access to Databases-- Theoretical/Technical Issues. *Proceedings of the 20th Annual Meeting of the Association for Computational Linguistics*. 44-45.

7- Rosenschein, S.J.and Schieber, S.M. (1982). Translating English into Logical Form. *Proceedings of t he 20th Annual Meeting of the Association for Computational Linguistics*. 1-8.

8- Templeton, M., & Burger, J. (1983) Problems in Natural-Language Interface to DBMS with examples from EUFID. Proceedings of the Conference on Applied Natural Language Processing. 3-16.

9- Woods, W.A. (1978). Semantics and Quantification In Natural Language Question Answering. In M Yovits (ed) *Advances in Computers*. Vol. 17, New York. Academic Press. 2-64.