A&QT-R 2002 (THETA 13)
International Conference on Automation, Quality and Testing, Robotics
May 23-25, 2002, Cluj-Napoca, Romania

SEMANTIC REPRESENTATION OF PROPOSITIONS USING FEATURE STRUCTURES WITH APPLICATION TO ROMANIAN LANGUAGE

Dana Avram

"Babeş-Bolyai" University of Cluj-Napoca, Department of Informatics

Abstract. This paper presents a way of representing semantic and syntactic information by using feature structure. We used 3BQ trees for semantic representation of a sentence. Then we traduced this representation into a typed feature structure representation and enriched it with the syntactic information. We presented an example by using the computing language Prolog for representing feature structure. We showed that the type hierarchy on which the typed feature is based on can encode the semantic representation of 3BQ trees. We presented a set of 4 rules that can be applied to determiners (in particular to articles) so that to transform an original true sentence in a new true one.

Keywords: semantic representation, syntactic representation, 3BQ trees, AVM's, attribute structures for Romanian language, Horn clauses, Prolog, determinants.

1. Introduction

Over the time was tried the defining of some structures and methodologies for semantic representations, but many of them isolate them the semantic information from the syntactic information and was ignored general semantic information (such as the membership of an object to a category), and, in opposition, a speaker more or less conscious include them in him propositions [1]. The similarity between this fact situation and the significance of semantic representation is notable remark.

The paper presents one of most intuitive method of semantic representation for sentences that is used frequently in ambiguity eliminating, the 3BQ trees. Starting with a representation of this type, there is presented a modality of representation with feature structures (and there representation by AVM's), that respect the model induced by 3BQ trees semantic representation and complete them with syntactic and general semantic information.

More, are presented few rules with can be used to obtain other feature structure that represent also the semantic information of a truly affirmation from an feature structure that represent the semantic information of a truly affirmation, different from first, in a similar way with deduction processes from natural language.

2. The semantic representation using 3BQ trees

The presented semantic representation formalism in this section is named 3BQ tree representation (tree branch quantifier trees) [2]. In this type of representation the words that are member of major grammar classes (nouns, adjectives, and verbs) are translated in predicates with arguments. The determinants introduce a metha-predicate that have as arguments predicates that correspond to referred words. All structure is represented as a tree and most of the nods have 3 successors.

The nouns are grammar categories which are represented the first and joining them an essentially role is playing by the determinants (noted Det) [3].

The nods of a Det constituent will be represented through following tree:

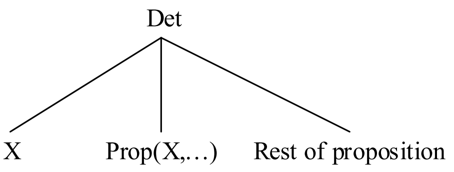

Fig. 1 Formal decomposition with 3BQ trees

where the X variable it refer an element from set of elements with Prop property.

Note that the X variable can or cannot appear in Rest of proposition. The order of proposition representation is with respect to the following algorithm:

- translate the subject;
- translate the complements, in apparition order;
- translate the predicate.

As an example, let's consider the proposition

Orice student foloseşte un calculator. (Every student uses a computer.)

and apply to it the decomposition described before. The result is depicted in Fig. 2:

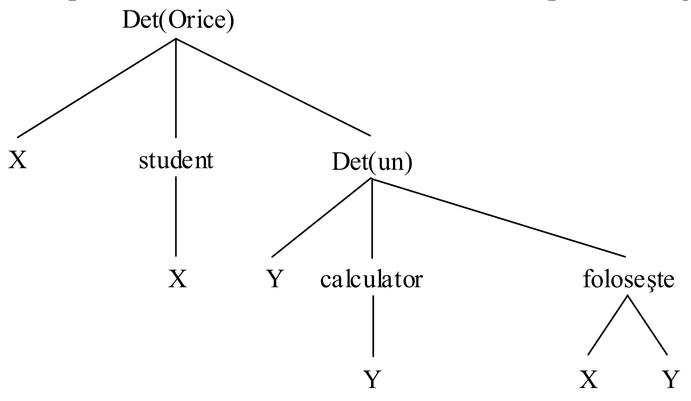

Fig. 2 Example of proposition decomposition using 3BQ trees

In this example (Fig. 2) the verb *foloseşte* is transitive and have (as any transitive verbs) two successors: first, the subject, and second, the complement. An intransitive verb has (usually) only one successor and this is the subject. An impersonal verb (*ploua*, to rain) doesn't have any successor.

The previous tree corresponds to following logical formula:

$$(\forall X)(\text{student}(X) \wedge (\exists Y)(\text{calculator}(Y) \wedge \text{foloseşte}(X,Y)))$$

The information contained in proposition is formed by a semantic part, and also by a syntactic part.[1,4] Supplementary syntactic information can later use for semantic analysis in context. Another problem is how we can operate with tree nodes directly, for getting specific information or a more general information, as a function of known data for the specific subject. That it correspond to charging of tree with supplementary information or extending him. Such mechanism already exists and is defined for another form of information representation named feature structure. Later, we will show that the feature structures can be defined such that it contains the semantic information from 3BQ trees and also it contains supplementary wished information.

3. Feature structures

Let consider a finite set named *Feats* with attributes (or features) and a set gifted with a *Type* inheritance hierarchy.

A typed feature structure over a set *Type* and set Feats is an n-order relation:[2,5]

$$F = (Q,q,\theta,\delta),$$

where:

Q is a finite set of nodes; $q \in Q$ is root node;

$\theta : Q \rightarrow \mathit{Type}$ is function of typing;

$\delta : \mathit{Feat} \times Q \rightarrow Q$ a partial function of attributes values.

In present paper we will refer only to typed feature structures, but we will simple called him as feature structures.

A feature structure can be graphic represented as attributes and values matrix (AVM). Let us note the attributes with f, g, …, types with α, β, …, and feature structures with A, B, …; then the representation of attributes structure as a AVM is in form:

$$A : \begin{bmatrix} \alpha & \\ f & : A \\ \cdots & \cdots \\ f & : A_n \end{bmatrix}$$

For a proposition, the semantic representation using feature structures can be build with respect to following rules:

- the words is types; language parts are types;
- their role in proposition will be associated with a name: AGENT for subject, PRED for predicate and CD for direct complement, etc.; more, for complements at this name will be attached a number, the number of complement type apparition; for a single apparition, this number can be omitted; the numbering is necessary to avoid apparition of two attributes with same name in structure of attributes;
- the name of AVM that represent the proposition will be chose as a unique identifier;

Let consider following hierarchy of type inheritance hierarchy, based on example from Fig. 2:

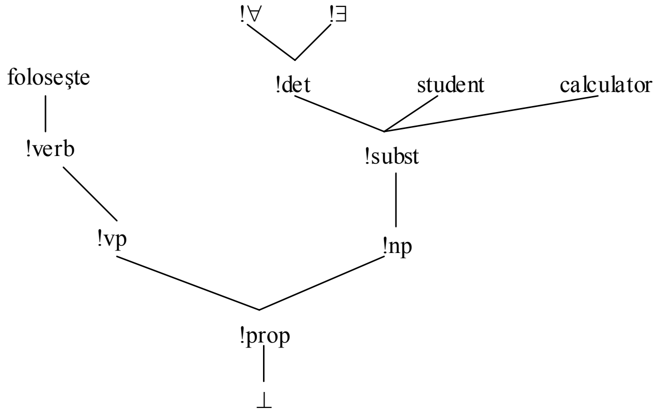

Fig. 3. Type inheritance hierarchy for example from Fig. 2

The proposition from Fig. 3 a AVM representation can be shown as in Fig. 4.

$$\begin{bmatrix} !\mathit{prop} & \\ \mathit{PRED}: & \begin{bmatrix} !\mathit{vp} & \\ \mathit{VERB}: & \begin{bmatrix} !\mathit{verb} & \\ \mathit{SYNSEM} & \mathit{foloseste} \\ \mathit{AGENT} & [1] \\ \mathit{CD} & [2] \end{bmatrix} \end{bmatrix} \\ \mathit{AGENT}: & \begin{bmatrix} !\mathit{np} & \\ \mathit{SUBST}: & \begin{bmatrix} !\mathit{subst} & & \\ \mathit{DET}: & & !\forall \\ \mathit{SYNSEM}: & [1] & \mathit{student} \end{bmatrix} \end{bmatrix} \\ \mathit{CD}: & \begin{bmatrix} !\mathit{np} & \\ \mathit{SUBST}: & \begin{bmatrix} !\mathit{subst} & & \\ \mathit{DET} & & !\exists \\ \mathit{SYNSEM} & [2] & \mathit{calculator} \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

Fig. 4. An AVM structure for example from Fig. 2

In classic way, the proposition are decomposed *by hand* in every proposition parts and, after that, there are identified the speak parts from them [6] The succession of the operation is obviously in previous representation. Note that there are words from inheritance hierarchy that have only an organizing role, and cannot appear in propositions as words. To make difference, these words was prefixed with the symbol *!*.

For easiest identification, we use the convention of noting the types with small letters and attributes with capital letters.

This manner of representation will not lose the sense imposed of words order in proposition and sense of determinants. It can be making a one to one function between elements of proposition set from a language and a restricted subset of AVM's [7]

Now, it is easy to observe that the determinants are operators over the set of AVM's constructed such in Fig. 5.

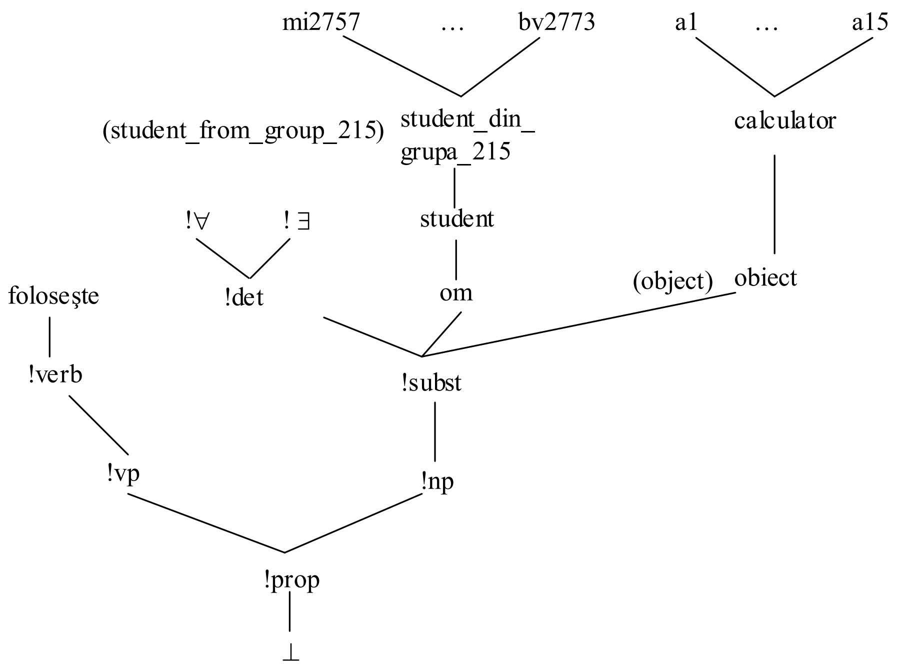

Fig. 5. A completed AVM structure for example from Fig. 2

Few natural rules induced by presence of the determinants are following described. Let's try to extend the hierarchy of types through adding words (types). If we consider that all computers that we have are named (with unique identifiers) a1, a2, …, a11. The students that are using these computers in labs time (in number of 9) assume that have the registration numbers mi2757, au2798, mf2812, ba2730, cr2710, dd2759, dv2708, ku2798, bv2773 (also unique identifiers in knowledge universe at wish to report the proposition). Completed types structure can be like in Fig. 5.

The feature structures use a relation named subsumption.

Intuitively, a feature structure F_1 *subsumption* another feature structure F_2 if F_2 it contain more information than F_1 or, more preciously, if it contain all information from F_1 and, eventually, more supplementary information [2,5].

A <u>definition</u> is necessary:

Let be $F_1 = (Q_1, q_1, \theta_1, \delta_1)$ and $F_2 = (Q_2, q_2, \theta_2, \delta_2)$ two feature structures.

Then can say that F_1 subsuming F_2 iif $\exists\ h : Q \rightarrow Q_2$ a total function, so that:

1. $h(q_1) = q_2$;
2. if $\delta(q, f)$ is defined, then $h(\delta_1(q, f)) = \delta_2(h(q),f)$, $\forall\ q \in Q$, $f \in$ *Feats*;
3. $\theta_1(q) = \theta_2(h(q))$, $\forall\ q \in Q_1$.

The function h is usually called subsumption morphism. An remark can be made: the subsuming relation defined for feature structures are assimilated in the logic of natural language with case of referring a noun joined by the determinant *oricare* (everything, ∀).

4. Horn clauses and Prolog

The final scope of preoccupation in semantics is to find a modality of computer representation for natural language propositions. Most approaching of this desiderate is makes by logic languages. One of them is Prolog language, which uses Horn clauses [8].

Looking for a implementation for:

Orice student foloseşte un calculator.

we will find:

foloseşte(X,Y):-student(X),calculator(Y).

The word order in proposition is not arbitrary. In this Prolog representation, the word order is lost. The sense of Prolog affirmation is:

dacă ∀ un student X, şi ∀ un calculator Y, atunci studentul X foloseşte calculatorul Y (if ∀ a student X, and ∀ a computer Y, then the student X use the computer Y).

A Prolog program with a given list of students and given list of computers will do Cartesian product of sets when are questioned with scope:

foloseşte(X,Y).

The impossibility to make difference between ∀ and ∃ representation makes Prolog to enumerate all possible pairs (student, computer). If the proposition has an ambiguity then the interpretation conducts to a list of possibilities, such in considered case.

Thus, in this logic, it is possible that all students to use the computer a1, and also it is possible that every student to use a different computer. Total number of possibilities for choused case is 11 [8]. This leads to too many data information.

However, the human mind if are confronted with a situation of this type, for such as proposition retain the *idea* (the proposition in a internal form) and do not try to find a concrete answer by browsing the possibilities.

In same order of ideas, if someone asks us: “Who and what computer uses?” the normal answer can be: “I don’t know!”. Probably, our brain has a representation that drive out the cases with very low probability.

It is clear that the representation with Horn clauses, although rigorous, can’t transpose the human way of words semantic representation. The problem appears from impossibility of exact reproduction of determinants. The Prolog allow us to use the *cut* predicate, that lead to number of solution limitation to only one solution, that is more closely to natural speaking. However, that is not enough.

5. The ∃ and ∀ in determinants and link between them

Most used determinants in natural languages are the articles. These have a semantic that contains the sense of ∃ or ∀ of joined noun.

The indefinitely singular article say us that ∃ the material object that it correspond to the given noun. This has a decisive role in context; we referred him and will refer in follows.

The definitely plural article “*toţi*” (all) has the sense of ∀, and joining noun can be referred more or less in context.

The indefinitely plural article indicate the existing (∃) of one or more than one objects of respectively type. Few rules are necessary.

<u>Rule no. 1.</u> From semantically point of view it can be easy observed that if in a truly proposition a word (specifically noun) is joined by ∀ determinant, then by replacing in the proposition the word with a word that correspond to a more specific type (from types hierarchy) than given word, then we also get a truly proposition. As example, starting from the proposition:

Orice student foloseşte un calculator.

(Any student use a computer.) we can say (see Fig. 5):

Orice student_din_grupa_215 foloseşte un calculator.

(Any student_from_group_215 use a computer.)

Feature structures that correspond to the first proposition subsume the feature structure of second proposition. That is not the case of ∃ determinant. For given proposition:

Un student_din_grupa_215 foloseşte un calculator.

(A student_from_group_215 use a computer) we cannot say that:

(Studentul) mi2757 foloseşte un calculator. (The student mi2757 use a computer)

In such a situation we can only estimate.

E posibil ca (studentul) mi2757 foloseste un calculator.

(It is possible that the student mi2757 uses a computer.)

<u>Rule no. 2.</u> Now, is easy to observe that we have the relation: ∀ implies ∃, and not vice versa. We can say that:

Daca orice student foloseste un calculator, atunci un student foloseste un calculator.

(If any student use a computer, then a student use a computer.) If we descend in types hierarchy (to the general) from a given expression, there are satisfied following rules about the ∀ and ∃ determinants:

<u>Rule no. 3.</u> ∃ remain ∃. As example:

Daca mi2757 foloseste un calculator, atunci un student foloseste un calculator.

(If mi2757 uses a computer, then a student uses a computer.) As well true, but less used in current language is the implication:

Dacă orice student foloseşte un calculator, atunci orice student foloseşte un obiect.

(If any student uses a computer, then any student uses an object.)

<u>Rule no. 4.</u> ∀ becomes ∃. For example:

Daca orice student_din_grupa_215 foloseste un calculator, atunci un student foloseste un calculator.

(If any student_of_group_215 uses a computer, then a student uses a computer.)

6. Conclusions and remarks

For semantic representation of a proposition the 3BQ trees are used. This representation was made by preloading of information from 3BQ trees to the feature structures. Additionally, joining this information in feature structures, there are stored supplementary information, like type of proposition parts and type of speaking parts. These are syntactic information (subject, predicate, …) and morphologic information (noun, verb, …).

Type's hierarchy that is base for feature structure includes the information that belongs to the semantic type (student – man). Are used this belonging for characterization of semantic information that are carried up by the determinants (such that are showed in case of articles).

A set of transformation rules that keep the initial truth value in newly resulted proposition are defined.

The modality of proposition representation from Prolog perspective through Horn clauses is exposed and was showed why this perspective is not sufficiently.

References

[1] Allen, J. (1995), Natural Language Understanding, The Benjamin Cummings Publishing Company, New York.

[2] Carpenter, B. (1992), The Logic of Typed Feature Structures, Cambridge Tracts in Theoretical Computer Science, no. 32, Cambridge University Press, New York.

[3] Gal, A., Lapalme, G., Saint-Dizier, P., Somers, H. (1991), Prolog for Natural Language Analysis, John Wiley, London.

[4] Jurafsky, D., Martin, J.M. (2000), Speech and Language Processing, Prentice Hall, Inc., University of Colorado, New Jersy, USA.

[5] Tatar, D., Avram D. (2000), Phrase Generation in Lexical Functional Grammars and Unification Grammars, *Studia Universitatis „Babes-Bolyai"*, vol XLV, pag. 69-78.

[6] Polard, C., Sag, I.A. (1994), Head-driven Phrase Structure Grammar, University of Chicago Press and Stanford CSLI Publications.

[7] Francez, N., Wintner, S. (1998), Feature structure based linguistic formalisms, draft, http.

[8] Tatar, D. (1994), Logical Grammars as a tool for studying Logic Programming, *Studia Universitatis „Babes-Bolyai"*.