A Computational Methodology as the Artificial Language for Natural Language Rules and the Unification Based Approach

Konstantinos Fouskakis

Department of Computer Science, "Politehnica" University of Timisoara, Faculty of Automation and Computers, Bd. V. Parvan, No. 2, 1900 Timisoara, Romania, E-Mail: costasfous@yahoo.com

Abstract: The purpose of this work is to describe the main features of a computational methodology of expressing linguistic rules on X-bar structures and the main ideas of the unification based approach.

Keywords: methodology, linguistics, language, unification.

1 Introduction

The X-bar theory is a linguistic framework proposed and improved by Noam Chomsky [1][2][3] mainly in the context of the syntactic analysis of natural language phrases. The X-bar theory has been elaborated by several workers both in the past and more recently not only in the context of syntax but also in the context of morphology. One of the basic points of the X-bar theory is that it advocates the existence of a general linguistic structural scheme expressed by a restricted set of abstract grammatical rules, which, according to the linguistic area of concern and to the specific case within this area, are constrained and mapped according, to specific linguistic categories. This general structure is the main built-in assumption of this methodology, in all other respects the methodology is open to subtheories, principles and transformations as long as these are complying the basic X-bar scheme. Under these assumptions, the methodology allows:

- the development of a set of principles and transformations
- the development of a set of X-bar theories in terms of principles and transformations
- the selective application of the above on the X-bar structures in order to obtain the desirable results

We must emphasize that the methodology does not impose any restrictions but the basic one (which is the most general one) and hence, it is believed to be open to future developments of the X-bar theory. It can describe general linguistic rules on the X-bar trees in a formal way similar to the X-bar theory and under the assumptions of the X-bar theory. The X-bar structure that the methodology manipulates may be used for syntactic, semantic and pragmatic information. In this way, it can be used as a representation for a machine translation system or a man-machine interface system by using a set of required transformation rules. Additionally, it supports the checking of the accepted rate at a rule application and permits the evolutionary changing of the manipulated X-bar structures. It imposes a new methodology of expressing linguistic rules and it is implemented in prolog. In the unification features based approaches like the HPSG[6][7], which is based on the sing representation, the sing of the source and the destination language are not possible to be determined directly. That is why it is still necessary for another semantic representation at this kind of systems. Also, the checking possibilities of the developed methodology allow a more flexible and strong controlling manner of the different cases of the linguistic knowledge.

2 The Unification Based Approach

2.1 The Context Free Grammars

An example of a CFG [4][6][7] can be the following:

S-> NP VP
VP -> V NP VP -> V
NP -> D N NP -> PRON NP -> PROPER_NOUN
D -> the | a | every N -> car | bicycle | boat | bus
V -> drives | repairs | drive | repair | rides | ride
RPON -> I | you | he | she | they | us | them
PROPER_NOUN -> ANN | GEORGE | NICK

This grammar produces a set of grammatically and semantically correct and incorrect sentences. Some examples of sentences that are produced and are not grammatically or semantically correct are the following:

them repairs bicycle bicycle drives car

Ann drive George the bus repair Nick

The phrase structure is the only syntactic relationship. The terminal and non-terminal symbols are atomic with out any properties. The information that is encoded in the grammar is based only on production rules and any attempt to encode semantic information requires an additional mechanism. The CFG

mechanism must be stronger in order to be able to fulfill the linguistic requirements (e.g. features structures, generalized phrase structures, unification grammars).

2.2 The Feature Structures And Unification Based Grammars

The CFG can be extended by associated features structures with the terminal and no terminal symbols of a CFG. The features structures are known and as AVM (attribute value matrixes). The words in the lexicon can be enhanced with additional information by using the features. Two examples:

Word: Car [NUM: singular, PER: third] Word: I [NUM: singular, PER: first]

Except the simple atomic values of the features NUM and PERSON in these examples, it is possible to have as value of the features other features structures. An example of a verb and its feature AGR:

Word: Runs [AGR: [NUM: singular, PER: third]]

Also, it is possible to use variables[7] with name e.g. X or with number e.g. [1] as following:

[AGR:X [NUM: singular, PER: third]] [AGR:[1] [NUM: singular, PER: third]]

The variables are used in order to determine that two elements of an AVM [7] have the same values. The general format of an AVM is the following:

$$A = [i_0] \begin{bmatrix} F1: & [i_1] & A1 \\ & & \\ Fn: & [i_n] & An \end{bmatrix} \quad \begin{array}{l} dom(A) \\ F_i \neq F_j \\ val(A,F_i) = A_i \end{array}$$

According to this, the previous example has:

dom(a) = {ARG} val(A,ARG) = [NUM: singular, PER: third]

Also, there is the notion of path π. At the same example the value of the path:

val(A,<AGR, NUM>) = singular val(A,<AGR, PER>) = third

but the val(A,<PER, AGR>) = undefined

Between two different features structures we can define the relation of subsumption [7].

If A and B are two AVMs, the A subsumes B (A≤B) when:

- A is an atomic AVM and B is an atomic AVM with the same atom
- For every F that belongs in dom(A) then and F belongs in dom(B) and val(A,F) subsumes val(B,F).

If two paths are re-entrant in A they are also re-entrant in B.

An example is:

[NUM: singular] ≤ [NUM: singular
PER: third]

An operation between two features structures A and B is the unification[6][7]. An example:

A= [NUM: singular] B= [PER: third]

and after the unification we have the: [NUM: singular
PER: third]

If variables exist in the A and B features structures:

A = [AGR:[1] [NUM: singular]] B = [AGR:[2] [PER: third]]

After the unification [AGR: [1][2] [NUM: singular
PER: third]]

We can add features in the rules. An example is:

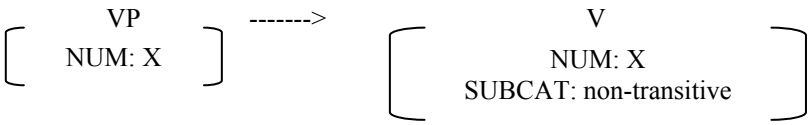
[NP
NUM: X] ------> [D
NUM: X] [N
NUM: X]

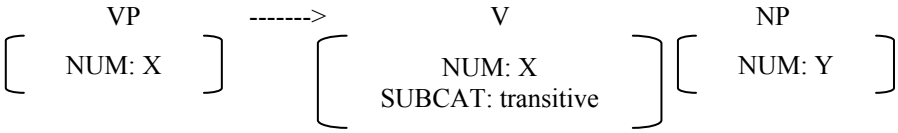
In this example the scope of the variable X is inside the rule and it means that the noun phrase (NP), determiner (D) and noun (N) have the same number. Also, if we want to control the case we can add a second feature the CASE:

NP -------> D N

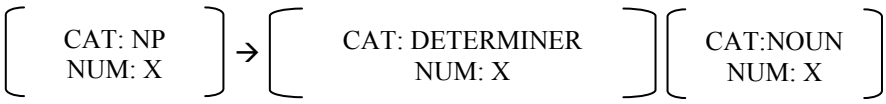
[NUM: X
CASE: Y] [NUM: X] [NUM: X
CASE: Y]

The rule for the verb phrase (VP) depends from the type of the verb. There are transitive and non-transitive verbs.

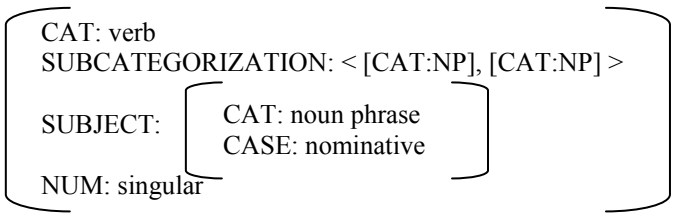




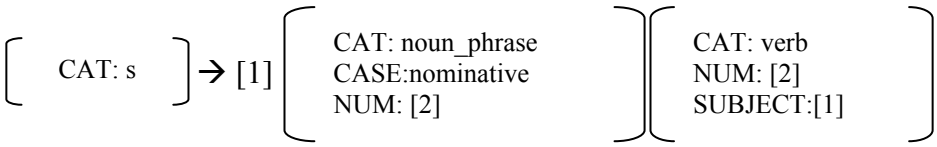
In the above examples it was used the CFG rules associated by the features structures. It is possible to include the non-terminals as values of a CAT feature.



In order to have complete sub categorization information we can enter in the lexicon the complete list of complements and the subject. It is possible to add additional features like the CASE that is determined for the subject of the verb taken in the following example (it is named sign in HPSG):



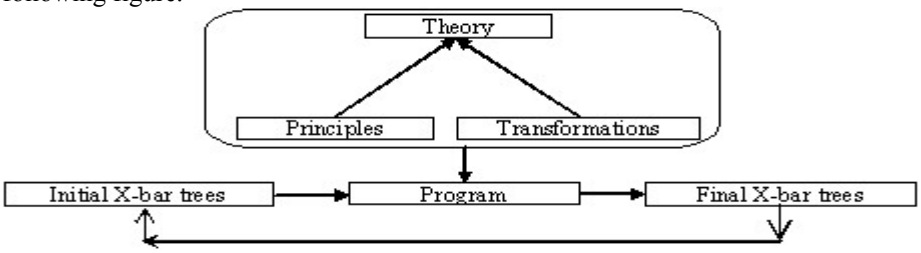
According to the above if we want to express the initial rule of the CFG: S - > NP VP with the use of features structures it will be as follows:



All the above examples and different cases describe the main notions and mechanisms of the unification based grammars.

3 Structure of Linguistic Knowledge

The linguistic knowledge for the presented methodology is demostrated in the following figure.



Let's define:

- LS: the system of the linguistic knowledge
- PR: the set of rules in the Principles
- TR: the set of rules in the Transformations
- GR: the set of rules in the Theory
- SR: the linguistic program
    - SR is subset of the concatenation of the sets GR, PR and TR
- IT: the set of initial X-bar trees
- OT: the set of final X-bar trees
- LS=(PR,TR,GR,SR,IT,OT)
- **The Initial X-bar Trees**

It contains trees that derive from the X-bar scheme in order to apply on them the defined rules.

- **Principles**

It contains the principles that have been defined. The principles check an X-bar tree if it accomplishes certain structural requirements as a whole or at its parts. Also, they can check if nodes, features of nodes, anaphors, even terminals are according to certain linguistic requirements.

- **Transformations**

It contains the transformations that have been defined. The transformations additionally, transform the X-bar trees and produce one or more new X-bar trees with different structure, nodes, features of nodes, anaphors or terminals.

- **The Linguistic Theory**

It contains rules that express the linguistic theory that one wishes to develop. These rules are expressed as sequences of principles and transformations. We can also have a conditional application of the rules by using expressions if-then-else and change the X-bar trees that are used by the next rules. The abilities that these rules have will be described in detail in the next sections.

– **The Linguistic Program**

It is the part of the linguistic system which declares the rules of the theory, principles, transformations that are applied on the initial X-bar tree and their order.

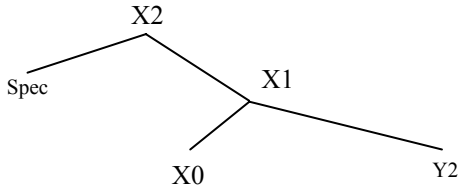
– **The Final X-bar Trees**

It contains the generated X-bar trees according to the linguistic program.

4 X-Bar Structures

The X-bar structures[1], that the system manipulates, are derived from the following rules:

X2 → Spec X2 *X2 → Spec X1*
X1 → X1 Y2 *X1 → X0 Y2*
Spec → X0 *Spec → X2*
X0 → terminal



In the above rules the Y2 is a structure of the form X2. These rules can derive structures of form X'' or XP of the X-bar theory[2][3]. The above X-bar structures are represented in the system with the use of parentheses and they have the following form:

(X2 (Spec) (X1 (X0) Y2)) (X2 (Spec) (X1 (X1(….) Y2)) (X2 (Spec) (X2(Spec)…

Every phrase, sentence or utterance can be represented in the initial X-bar trees by more than one X-bar tree.

4.1 Nodes and their Features

A node of an X-bar structure is defined by its name followed by its category. So the node X2 is declared as x **barii**, the X1 as x **bari** and the X0 as x **bar**. The features of a node give grammatical, syntactic and semantic information of a node or subtree. A feature is notated as following:

- + *Name of the feature, - Name of the feature, Name of the feature*
- *Name of the featureY= Name of the featureX*
- [*Name of the feature1,….,Name of the featureN*]=*Name of the featureX*

Their semantics depend from our interpretation. In the last two cases the order of the features is important and these cases are not supported by the X-bar theory of Chomsky. They permit better well expressed additional descriptions. Examples of the previous cases are the following:

- +male, -human, singular
- phrase_time=t2, focus=v1, [+live_being,+thing]=complements
- **node** article **bar: features [**+singular**,** +nominative**]**

4.2 Terminals

The terminal elements of the X-bar structures are represented:

terminal *terminal element*

Examples of terminals: **terminal** man, **terminal** woman

4.3 Anaphors

The anaphor declaration is between the following elements (they can be in different X-bar trees):

- terminal elements or traces of terminal elements
- subtrees or traces of subtrees

The general format for anaphoric connections: **anaphor** *name of anaphor*

An example of a terminal element with its anaphor: **terminal** the**:anaphor** i1

5 Principles and Transformations

The principles and the transformations are rules that we define according to the presented methodology[5]. These rules are stated to be applied on the X-bar trees that were described in the previous section. The principles are used to control the correctness of an X-bar tree according to the requirements that we state. The transformations are stated in the same way and have the same abilities with the principles, but they can also change the structure and the elements of the tree on which they are applied on, leading to one or more trees. The principles and transformations are the main part of the methodology and are declared in the presented linguistic knowledge system. We can enter in the system a large set of such rules and use only these rules that we wish to apply each time on the X-bar trees. With these rules we express the main linguistic knowledge that is of our interest and thus we can process the natural language trees accordingly. The

complexity and the number of the rules depend on our requirements. Both the principles and the transformations are stated using the same general pattern.

- **principle** / **transformation** The name of the rule
- **variables** (The variables are declared which correspond to parts of an X-bar structure and can have more than one values)
- **structureDescription** (An X-bar subtree structure is described on which the rule is applied. Also, variables and operators are used)
- **structureCommands** (The different elements checks, the variables values changes, the new declarations of variables and the transformations, if the rule is of transformation type, and other possible commands are used)

In order to define general rules, a group of operators that describe the relations between different subtrees, as well as variables in the fields of principles and transformations have been developed.

These are the variables of the kind of **variables** field and the variables that can be defined only in the **structuredescription** field and are used for the description of the transformations in the **structurecommands** field.

The variables of the first kind can be either variables that have already been defined in the field **variables** or new variables. If a variable has already been defined then it must be of the same type with the corresponding element of the **structuredescription** structure that it substitutes. This variable constraints the corresponding element of an X-bar structure that the rule is applied on, in a specific set of values (see in the example below the variables Noun and Verb). Also, we can use new variables of the **variables** type that are defined automatically the first time as they appear in the **structuredescription** structure by taking their values from the corresponding element of the X-bar structure where this rule is applied on (see in the example below the variable sbLeft).

The other kind of variables can be of type node of structure, terminal element or subtree. They can be used in combination with the other kind of variables and they belong to the **transformationvariable** kind. The result of this definition is the declaration of a new variable. The type of this variable is the type of the corresponding element of the **structuredescription** structure. The initial value of this variable is the value that has the corresponding element of the X-bar structure on which we apply the rule (see in the example below the variables sdRoot, sdRight and sdWhole).

Also, there are different kinds of operators in the **structuredescription** field. There are operators with two arguments that declare that a subtree must be or not (left or right) subtree of another subtree and a subtree must be or not subtree of a another tree with a specific head node. Except the above, there are operators with one argument that declare a subtree must not be subtree of an X-bar structure at a

specific position (**not**), a subtree should exist as a subtree in any depth in respective place of the X-bar structure (**aTree**), a subtree is the first subtree in any depth if the tree is scanning top-down left to right starting from the respective place of the X-bar structure (**aFirstTree**), a subtree is the left most subtree in any depth (**leftMost**). Also, there are the operators (**and**, **or**) with two or more operands. The first operator declares that all the operands are subtrees at this position of an X-bar structure. The second operator declares that at least one of the operands is subtree at this position of an X-bar structure.

Finally, an assumption is stated:

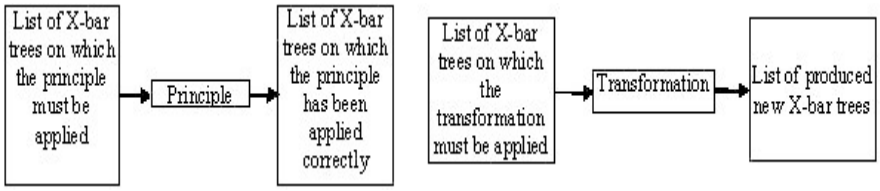
> If the tree of the **structureDescription** field or a subtree of this tree contains less anaphors or features of nodes than the X-bar tree in its corresponding position then the rule is applied on this tree.

This assumption is based on the general principle:

> If the required information for the application of a rule exists in an X-bar tree then it is possible for this rule to be applied on this X-bar tree. The tree examination is top-down left to right.

In the **structurecommands** field we can define new variables in the same way as the **variables** field, of features type that take values from tree nodes, of anaphor type that take values from terminals subtrees and of subtree type that take values from an input X-bar structure. Additionally, the values of variables can be changed by adding or removing their values, setting new values, calculating all the values of a variable according to the current values of the possible used variables and other possibilities. The transformations are declared by using the corresponding operators (see the operators **transformations** and **transform** in the following example). Also, there are many operators that check features, anaphors, trees, nodes terminals and their parts. It is possible to use these operators in combination with the **if – then – else** command in order to execute various commands.

The principle has as output structures the subset of its input structures on which has correctly been applied on. The transformation has as output structures the new set of structures that have been produced by it. Also, there is the possibility to check the accepted rate between the input trees and their subset of succeeded trees by using the command **acceptedRate**(*Rate*).



An example of transformation rule is the movement of a noun phrase.

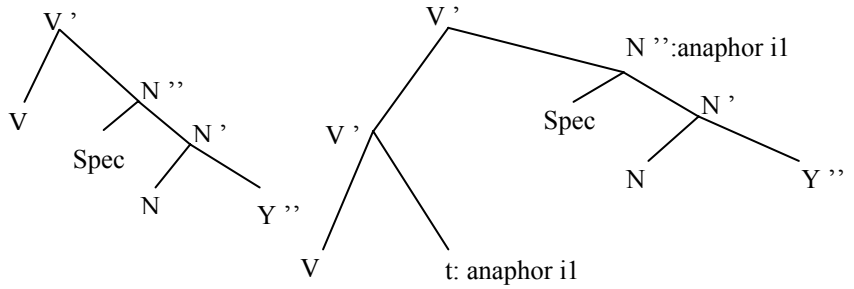
```
transformation   'Attachment of noun phrase'.

variables node 'Noun'  set  'N' barii        or       'Noun' barii
   also     node 'Verb'  set  'V' bari        or       'Verb' bari.

structuredescription
  atree  (node &'Verb': transformationvariable sdRoot,
                  subtree &sbLeft,
                  (node
          &'Noun',anytree,anytree):transformationvariable sdRight
          ): transformationvariable sdWhole.

structurecommands
(&sdRight addanaphor i1,          % addition of anaphor reference
 transformations                  % declaration of transformations
      &sdWhole transform
           (node &sdRoot,
                  (node &sdRoot, subtree &sbLeft, t:anaphor i1),
                  subtree &sdRight)).
```

The above transformation acts upon an X-bar structure that has a sub tree (operator **atree**) as the left structure and produces a new X-bar structure with a subtree as the right structure:



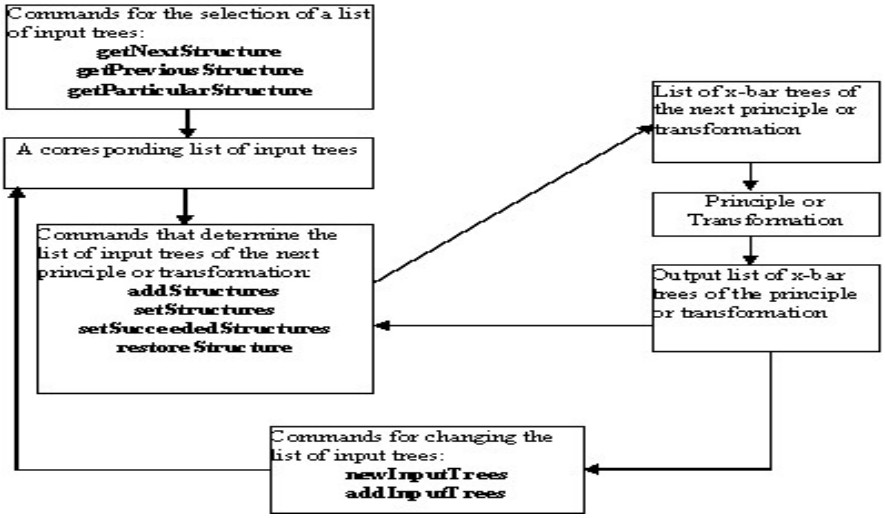
6 Linguistic Theory

We can describe a set of rules by using principles and transformations. This set of the rules constitutes our theory. Their general pattern is:

- **grammar** *name of grammar*
- *the main part of the grammar*

In *the main part of the grammar*, we use **principles** and **transformations**, as well as other **grammars** that have already been defined. Also, it is possible to have conditional application of the rules in a grammar, depending on the result from the application of some other rules by using **if - then – else.** We can perform a repeated application of the grammar if in the main part of the grammar we use the command **grammar** *name of the same grammar*.

The transformational rules are able to produce one or more new X-bar structures, also the principles returns the set of succeded X-bar structures. These structures can be used by the next rule (principle, a transformation or a grammar) for further processing. The operator that adds the new set of structures is the **addStructures**, that sets the new set of structures is **setStructures**. The operator **setSucceededStructures** sets as X-bar structures for the next rule the structures that the last rule has been successfully applied on. The operator **restoreStructure** resets the X-bar structures for the next rule to the last X-bar structures list that has gotten from the initial X-bar structures. Also, it is possible to select another X-bar structure from the set of structures by using the operators **getNextStructure**, **getPreviousStructure** and **getParticularStructure**(Id). Also, there is the operator **getInputTreeId**(Id) that returns the id of an input X-bar structure. Except the above operator there are the operators **newInputTrees**(Id) and **addInputTrees**(Id). They change the input structures according to the output structures of the last principle or transformation.



In order to exchange information between the different rules that are used by the grammars, there are the grammar variables. They can be used by more than one principle or transformation and permit smaller rules by using known information. Their operators are **addGrammarVariable, removeGrammarVariable**. The commands about variables of the **structureCommands** field of principles (new variables declaration, change and checks of the variables values) can be used.

Conclusions

A computational system that implements the presented methodology is possible to be used as a tool by researchers. They can define rules and they can apply them on a set of X-bar structures. Moreover, it is possible to combine this with another system that produces these X-bar structures based only on general phrase structure information. They can produce a set of X-bar structures and then the second

software system (that implements the presented methodology) will examine and transform these structures and will produce new ones or will reject invalid structures. The software system of the presented methodology can manipulate the semantic, syntactic and pragmatic information of the X-bar structures. The main advantage of this approach is the possibility to define more general and simple rules that can be close related with the X-bar theory. The structures are all derivations of a specific binary tree, the X-bar scheme. The above facilitates the implementation, maintainance and extension of the corresponding applications. This particular two levels implementation is better for embedded applications since the defined and produced structures are simpler and it is not necessary to have large memory size and strong processor. The most important is that it can be used for the description of general linguistic rules on the X-bar trees in a computational way as computer language implemented in prolog that has specific syntax and semantics with operators and variables (different than approaches of natural language proccessing like rewriting rules or the no-transformational unificational LFG, HPSG). It describes a powerful and flexible way of staying and manipulating the linguistic knowledge with step-by-step production and checking the X-bar trees and imposes a new methodology of expressing linguistic rules. It manipulates the semantic and syntactic information of the x-bar structures and according to the acceptance rate of a rule and permits the evolutionary changing of the manipulated X-bar structures.

References

[1] N. Chomsky: Lectures in government and binding, Dordrecht: Foris, 1981

[2] N. Chomsky: Barriers, Massachusetts: MIT Press, 1986

[3] N. Chomsky: The minimalist program, Massachusetts: MIT Press, 1995

[4] D. Cristea: Formalisme si instrumente de descriere si prelucrare ale limbajului natural, Iasi: Editura universitatii “Alexandru Ioan Cuza”:2002

[5] K. Fouskakis: ‘A computational methodology for linguistic rules’, SACI Romanian-Hungarian Joint Symposium on Applied Computational Intelligence (IEEE Romanian section), Timisoara, Romania:2004

[6] D. Tatar: Inteligenta arificiala - demostrarea automata a teoremelor prelucrarea limbajului natural. Cluj-Napoca: Editura Albastra:2001

[7] D. Tatar: Inteligenta arificiala – aplicatii in prelucrarea limbajului natural. Cluj-Napoca: Editura Albastra:2003