AN OVERVIEW OF A COMPUTATIONAL APPROACH FOR LINGUISTIC RULES ON THE X-BAR TREES

Konstantinos FOUSKAKIS
"Politehnica" University of Timisoara, Faculty of Automation and Computers
Bd. V. Parvan, No. 2, 1900 Timisoara, Romania
costasfous@yahoo.com

Abstract. *The purpose of this work is to describe the main features of a methodology, which allows the manipulation of X-bar based structures and the definition of linguistic rules on them in a variety of ways. This methodology addresses the needs of those linguists who are conducting research on generative transformational grammars using some variant of the X-bar theory. Furthermore, this methodology can be used into a computational system which produces or manipulates X-bar structures for additional processing. The methodology is open in the sense that it may be used both for various X-bar theory versions, but also for various phases or aspects of natural language analysis, such as syntax and morphology.*
Keywords: *methodology, linguistics, system, principles, transformations, grammars.*

Introduction

The X-bar theory is a linguistic framework proposed and improved by Noam Chomsky [1] [2] [3] [4] [5] mainly in the context of the syntactic analysis of natural language phrases. Also, other researchers have studied the language in a structural way [6]. The X-bar theory has been elaborated by several workers both in the past [9] and more recently [8] not only in the context of syntax [12] but also in the context of morphology [11]. 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.

The exact general structure, which is the main built-in assumption of this methodology, will be given in a next section. In all other respects, the methodology is consistent with the spirit of the X-bar theory and open to subtheories, principles and transformations as long as these are obeying the basic X-bar scheme.

Under these assumptions, my 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 and obtaining 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. The most important is that it can describe general linguistic rules on the X-bar trees in a formal way similar to the way that the X-bar theory does and under the assumptions of the X-bar theory. It imposes a new methodology of expressing linguistic rules. The methodology is also implemented as a computer system. This system [7] is a special tool for linguists and additionally, it can be integrated into a computational system (even a machine translation system) which produces X-bar structures for additional processing.

Structure of Linguistic Knowledge

The linguistic knowledge for the presented methodology has a structure which is presented in the following figure.

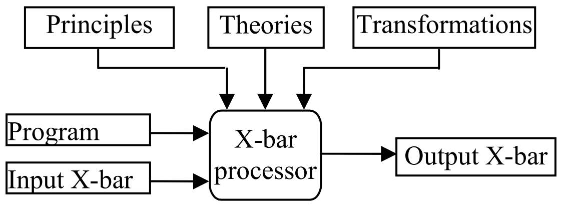


Fig.1 Linguistic universe

This structure represents the linguistic knowledge universe. It has the above schema that is analyzed as follows:

- Input X-bar structures

It contains the X-bar tree structures of the phrases (we assume that the linguist himself or another system has produced these initial X-bar structures). Its format is given in a section below and is according to the X-bar theory.

- Principles and transformations

It contains all principles and transformations that have been defined so far. The principles check an X-bar structure if it accomplishes certain structural requirements as a whole or in parts of it. Also, the principles can check even if nodes, features of nodes, anaphors, terminals or even subtrees are according to certain linguistic requirements. On the other hand the transformations additionally transform the X-bar structures and produce one or more new X-bar trees with different structure, nodes, features of nodes, anaphors or even terminals. Their format is given in the corresponding section below.

- Linguistic theories

It is actually the various theory versions as expressed in the presented methodology. Each version of the theory is defined in terms of principles and/or transformations which may be conditionally applied via if-then-else expressions. The format of a linguistic theory is described in the corresponding section below.

- Linguistic program

It is the actual part of the linguistic universe which declares the rules of the universe (theories, principles, transformations) that are applied on the X-bar structures and in what order. The format of a linguistic program is described in the corresponding section below.

- X-bar processor

It is the heart of the linguistic knowledge universe and controls its action.

- Output X-bar structures

This is the output with the generated X-bar structures and the corresponding information of the application of the linguistic program.

The linguistic knowledge universe is a system with rules, in the following sections the capabilities that the presented methodology implies will be described in more details.

The Existing Computational Methodologies

A. *The phrase-structure grammars*

They were presented mainly by Chomsky in 1957. They have the general form of x -> y, where x, y can be any combination of terminal and no-terminal elements.
The different categories of the phrase-structure grammars are the following:

- regular grammars:
  - left-linear grammars
  - right-linear grammars
- context-free grammars
- context sensitive grammars
- unrestriced grammars

These grammars are used in computational systems with different kinds of enhancements in order to produce or recognize natural language phrases. They are not restricted to specific tree structures and it is difficult to maintain and extend an application that uses this type of grammars. However the advantage of these grammar is that they have a very simple general format.

B. *Transition networks*

They are represented as finite states automatons [10]. They are directed graphs with arcs noted by terminal elements. One node of the graph is denoted as starting point and another one as ending point. A sentence is accepted by the system if there is a path from the starting point to the ending point and its arcs contain the words of this sentence. There are different kinds of transition networks :

- (STN) simple transition networks
- (RTN) recursive transition networks that are the same with the STNs but they additionally permit at their arcs phrasal categories except the lexical categories and recursions.
- (ATN) augmented transition networks that are RTNs with a set of registers for each network.

The disadvantages of these networks are:

- The networks are very complicated.
- It is not possible to describe general rules for the different phrase categories in one network. Usually, they are spread in many different networks.
- There are problems at phrases with ambiguities.
- The check, the maintainance and the extension of these networks is very difficult.

The main advantage is that they have a simple general formalism that is possible to be implemented easily.

C. *Lexical functional grammar*

The basic characheristic of this grammar type is that the lexical records are declared as predicate structures with arguments. These structures are independent from the phrase structures and they are a form of functional comments for the lexical records. Also, there is the functional information of the phrase structures. This information is combined with the functional information of the lexical records and the final functional structure of a phrase is produced. The disadvantage of this theory is the only two functional equations between the functional information of the phrase structures and the lexical records. This sets restrictions on the declaration of rules.

D. *Generalized Phrase Structure Grammar*

This grammar type emphasizes on the information that the syntactic categories have. The internal structure of the syntactic categories is recognized. The corresponding theory suggests the separation of the rules of syntactic structures in two categories:

- Rules of immediate dominace
- Rules of linear precedence

The first type refers to the hierarchical relation between different categories. The second type refers to the position that the different categories have in a sentence. This type of grammar is better for free order languages. It does not support a specific tree structure and it is more difficult to extent an application or to declare reusable and general rules.

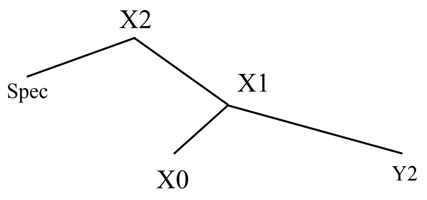
E. *Head-driven Phrase Structure Grammar*

This grammar type requires the existence of detailed morphological, syntactical and semantical information for every word. It requires more detailed information than the lexical functional grammars. This grammar is not a syntactical grammar but it combines both syntax and semantics. It organizes the linguistic knowledge as features structures. These features are sorted according to the specialization of them. Also, there is the possibility for paths that define the relation between them. The biggest difference between this theory and the previous ones is the way for the manipulation of the lexical records. Every representation requires very complicated information and there are very big problems for the maintenance of this huge information. Additionally, there is not any specific tree format and it is possible to have arbitrary different structures.

X-Bar Structures

The X-bar structures, 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 [8]. 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)…

In the following sections words or symbols in bold are operators.

A. 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 node X1 is declared as x **bari** and the node X0 is declared as x **bar**.
Also every node of the tree may have a set of features. The features give grammatical, syntactic and semantic information of a node or subtree. In order to declare the features of a node we use the operator **features** followed by the features of the node. The way of declaring the features of nodes is described below. The features of a node are surrounded by the **[** and **]** and separated by commas, their order is not important.
A feature is notated as following:

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

Their semantics depend from our interpretation. Examples of the previous cases are the following:

- +male
- -human
- singular
- [+live_being,+thing]=complements

An example of a complete node is the following:
node article **bar: features [**+singular**,** +nominative**]**

B. Terminals

The terminal elements of the X-bar structures are declared by the operator **terminal** followed by the terminal element:

terminal *terminal element*

Examples of terminals of the X-bar structures are the following:

- **terminal** man
- **terminal** woman

C. Anaphors

The system also supports the anaphor declaration between elements of an X-bar structure. The elements that we can connect to anaphor are the following:

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

In order to declare the anaphor between elements of the above types we use the following format:

anaphor *name of anaphor*

The anaphor always follows the terminal element. An example of the terminal element *the* with its anaphor i1 is the following:
terminal the**:anaphor** i1

An example of anaphor between two different elements is the existence of the terminal element *man* and the terminal element *that* that is connected to the anaphor i1. The way of declaration of this anaphor is the following:

- **terminal** man**:anaphor** i1
- **terminal** that**:anaphor** i1

Besides an anaphor between terminal elements, there are the following possibilities of anaphors:

1) between the terminal and its trace, for example the word man and its trace **t**:
   terminal man**:anaphor** i1
   terminal t:anaphor i1
2) between the subtree and its trace, for example the subtree of the noun phrase (OP (article the) (O'(O house) e)) and its trace **t** :
   (**node** o **barii**,
    (**node** article **bar**, **terminal** the),
    (**node** o **bari**,
      (**node** o **bar**, **terminal** house),
      empty))**: anaphor** i1
   t:anaphor i1

The operator **empty** is used to denote an empty branch of a tree.

Principles and Transformations

The principles are described according to the methodology by using the following format:

- **principle** *the name of the principle.*
- **variables** (in this field we declare the variables which correspond to parts of an X-bar structure)
- **structuredescription** (in this field we describe the structure of the subtree of an X-bar structure on which we want to apply the principle rule)
- **structurecommands** (in this field we describe the conditions and the commands of this rule)

The above rule can define principles in order to check whether the X-bar structures match specific requirements. An example of principle of the X-bar theory that can be defined with the **principle** rule is the case filter (see examples at a section below).

The transformations are described according to the methodology by using the following format:

- **transformation** *the name of the transformation.*
- **variables** (in this field we can declare the variables which correspond to parts of an X-bar structure)
- **structuredescription** (in this field we describe the structure of the subtree of an X-bar structure on which we want to apply the transformation rule)
- **structurecommands** (in this field we describe the conditions, the commands and the required transformations of this rule)

The above rule can define transformations in order to transform the X-bar structures. An example of transformations of the X-bar theory that one can define with the **transformation** rule is the movement of a noun phrase (see examples at a section below).

The principles and the transformations rules can express the linguistic rules in an abstract way. Of prime importance for the above is to declare variables at the field **variables**. These variables can determine the possible values of parts (trees, nodes, features of nodes, terminals, anaphors) of the X-bar structures. Also, the principles and transformations use a set of operators and define or use variables at the fields **structuredescription** and **structurecommands** in a way close to the English language. These operators and the variables provide the necessary abstract way of expressing the linguistic rules [7]. Finally, it is important to be mentioned that the rules return boolean values denoting the success or failure of the application of the principle or transformation on an X-bar structure. When a principle or a transformation is applied on an X-bar structure, it succeeds if and only if both **structuredescription** and **structurecommands** succeed, otherwise it fails, thus affecting the course of action of the linguistic program.

Linguistic Theory

In order to describe a theory we use transformations, principles and other theories that we have already defined.
The general scheme for the declaration of the theory is the following:

- **theory** *name of the theory.*
- *The main part of the theory.*

In *the main part of the theory* we can use a sequence of principles, transformations and other theories as follows:

- **principle** *name of the principle*
- **transformation** *name of the transformation*
- **theory** *name of the theory*

or with the use of external user intervention:

- **askprinciple** *name of the principle*
- **asktransformation** *name of the transformation*
- **asktheory** *name of the theory*

Besides the above unconditional way for the application of rules, there is the ability for conditional application of them. When we say rules we mean the principles, the transformations and the theories.
The structure for the selective application of the rules is the following:
if *condition* **then** *action 1* **else** *action 2*
On the conditional part of this structure we can check if a rule or any logical combination of rules is true or false in order to proceed to the application of the first (**then**) or the second *action* (**else**).

Also there is the ability for recursive application of a theory by using in the main body of the theory the following formula:
theory *the name of the same theory*

The above can be used within an **if - then - else** structure in order to have conditionally recursive application of the theory.

Finally, we can use in the main body of the theory, the following four operators for the modification of the X-bar structures that are to be used by the next principle, transformation or theory. The operators are the following:

- **addstructures** : It adds the structures that have been produced by the last transformation or theory on the existing X-bar structures.
- **setstructures** : It sets as X-bar structures the structures that have been produced by the last transformation or theory.
- **restorestructure** : It resets the X-bar structures to the structure that have been read from the X-bar trees input.
- **getstructure** : This operator gets the next input X-bar structure in order to apply the following rule.

All the above elements can be used in the main body of a theory and are separated by commas and finish with a full stop. A sequence of the above elements composes the main body of a theory. These elements are applied on the X-bar structures according to the order that have been declared in the theory.

Linguistic Program

It contains the principles, the transformations and the theories that we want to apply on the X-bar structures. These are applied in the order that they are in this input.

The way of the selectively calling of the principles, the transformations and the theories is the following:

- **theory** *the name of the theory*
- **transformation** *the name of the transformation*
- **principle** *the name of the principle*

Also, it is possible to have external user intervention in the following way:

- **askprinciple** *name of the principle*
- **asktransformation** *name of the transformation*
- **asktheory** *name of the theory*

Examples

The following two examples are rules that are well known in the X-bar linguistic theory.

The principle of case filter [8]

The linguistic rule is the following:

Case filter

No nominal phrase can stand in a structure unless it bears a case. In other words, the structure that contains (NP [-case]) is rejected.

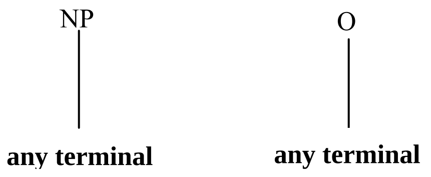
principle 'Case Filter'.

variables
node noun **set** 'NP' **bar or** 'O' **bar**.

structuredescription
(**node &**noun: **transformationvariable** sd1,
terminal &t)**:transformationvariable** sd2.

structurecommands
(**features** case **set** [+ptosi] **or** [+case],
ifthenelse(**&**sd1 **acommon &**case,
comment
"The principle of case filter is valid at : "**:&**sd2,
comment
"The principle of case filter is not valid at : "**:&**sd2)).

The above principle acts upon X-bar structures which have one of the following two sub trees:



Then at the field **structurecommands** checks if the node NP or O has the feature +case or the feature +ptosi and sends the corresponding message at the output.

The rule for the movement of a noun phrase [8]

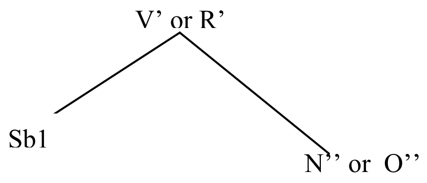
This transformation moves a noun phrase from one position of the X-bar tree to another.

transformation 'Movement of a noun phrase'.
variables
node 'Noun' **set** 'N' **barii or** 'O' **barii**
also node 'V'**set** 'V' **bari or** 'R' **bari**.

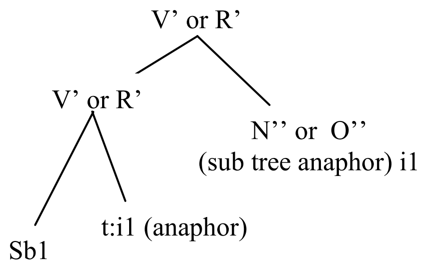
structuredescription
(**node &**'V'**:transformationvariable** sd3,
subtree &sb1,
(**node &**'Noun', **anytree**, **anytree**)**:**
transformationvariable sd1
)**: transformationvariable** sd2.

structurecommands
(**&**sd1 **addanaphor** i1,
transformations
&sd2 **transform**
(**node &**sd3,
(**node &**sd3,
subtree &sb1,t**:anaphor** i1),
subtree &sd1)
).

The above transformation acts upon an X-bar structure that has a subtree of the following structure and produces a new X-bar structure:



And the produced X-bar structure is as follows:



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. Additionally, it is possible to combine this with another system that produces these X-bar structures. That system can use a set of very simple rewriting rules for the production of the X-bar structures.

These rules can be 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 and syntactic information of the X-bar structures. For this reason it is necessary for the lexicon to have the syntactic and semantic information as a form of node features.

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. According to the linguistic researchers this scheme is strong enough to be used for the representation of the natural language sentences. The above facilitates the implementation, the maintainance and extension of the corresponding applications. This two level 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.

References

[1] Chomsky, N. (1970) *Remarks on nominalization*, In Jacobs & Rosenbaum (eds) Readings in English Transformational Grammar, Massachusetts: Xerox College, 184-221.
[2] Chomsky, N. (1981) *Lectures in government and binding*, Dordrecht: Foris.
[3] Chomsky, N. (1982) *Some concepts and consequences of the theory of Government and Binding*, Cambridge: MIT Press.
[4] Chomsky, N. (1986) *Barriers*, Massachusetts: MIT Press.
[5] Chomsky, N. (1995) *The minimalist program*, Massachusetts: MIT Press.
[6] Fodor, J.A & Katz,. J.J (1964) *The structure of language - Readings in the philosophy of language*, Englewood: Prentice Hall.
[7] Fouskakis, K. E. (2000) *An open system for linguistic rules on the X-bar trees*, Ukrainian Journal of Computational Linguistics, Lviv, Ukraine.
[8] Haegeman, L. (1994) *Introduction to Government and Binding* 2nd Edition, Oxford: Blackwell.
[9] Jackendoff, R. (1977) *The X-bar syntax – A study of phrase structure*, Massachusetts: MIT Press.
[10] Noble, H. M. (1988) *Natural Language Processing*, Oxford: Blackwell Scientific Publications.
[11] Spencer, A. (1991) *Morphological theory an introduction to word structure in generative grammar*, Oxford: Blackwell.
[12] Theofanopoulou, D. (1994) *Transformation syntax from the theory to application II*, Athens: University of Athens.