A computational methodology for linguistic rules

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 format and functionality of the principles and the transformations of a computational methodology for expressing general linguistic rules on the X-bar trees.

Keywords: methodology, linguistics, system, grammars.

1. Introduction

The methodology that I have developed allows:

- the declaration of a set of principles and transformations
- the declaration of a set of theories in terms of principles and transformations
- the selective application of the above on the X-bar structures and obtaining the desirable results

The linguistic knowledge of this methodology has the following structure.

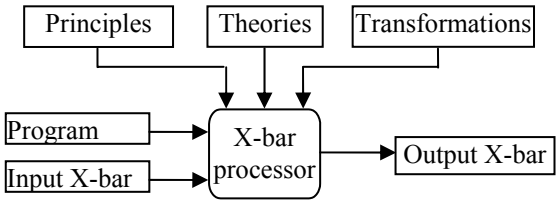


Fig.1 Linguistic universe

It represents the linguistic knowledge universe that is analyzed as follows:

- Input X-bar structures

It contains the X-bar tree structures of the phrases. Their format is given below and is according to the X-bar theory[1][2][3].

- 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 its parts. Also, they 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 trees and produce one or more new with different structure, nodes, features of nodes, anaphors or terminals. Their format is given 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 X-bar trees that the next rule is going to be applied on can be changed according to the produced x-bar structures by the rules.

- Linguistic program

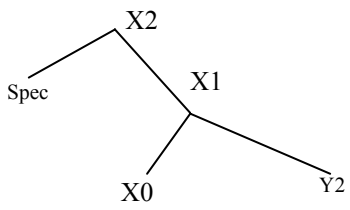
It is the actual part of the linguistic universe which declares the rules of the universe (theories, principles, transformations) that are applied on the X-bar trees and in what order.

- Output X-bar structures

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

The general form of the X-bar trees that are manipulated by the principles and transformations is the following:

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



The *Y2* is a structure of the form *X2*. The X represents linguistic categories (verb, noun, etc). In the following, the words that are in bold are operators.

2. The principles and transformations

They are described according to the methodology by using the following format:

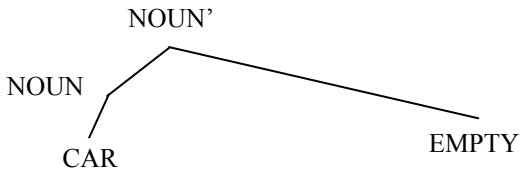
- **principle** / **transformation** *the name of the rule*.
- **variables** (we declare the variables which correspond to parts of an X-bar structure)
- **structuredescription** (we describe a subtree structure of an x-bar tree on which we want to apply the principle rule)
- **structurecommands** (we describe the checks, the variables values changes, the new declarations of variables and the transformations if the rule is of transformation type)

The **structuredescription** is used for the designation of the subtree on which the specific rule will be applied either this rule belongs to the principles category or to the transformations category. In order to apply the principle or the transformation on an X-bar tree, the subtree that we describe in the **structuredescription** field must be part of the tree or even the whole of this X-bar tree.

An example of the subtree that we can enter in the **structuredescription** field:

(**node** noun **bari**, (**node** noun **bar**, **terminal** car), **empty**)

The subtree's scheme is the following:



The above subtree, apart from the specific structure, has also specific names for the nodes and the terminal elements. This subtree is of the X1 category. The variable X has the value NOUN and the terminal element is the word "car". The specific structure and elements of the above tree limit the application of the specific rule in only one subtree. Therefore in order to apply the rule, it is necessary to find an X-bar tree with exactly the same subtree. However, this constraint does not enable us to define principles and transformations that will cover general cases for a set of trees that will have a certain common structure and characteristics on which the specific principle or transformation can be applied.

The linguistic theory [1][2][3][4][5] regarding the form and the characteristics of the trees demands general rules that should cover a lot of cases. The use of presented methodology at a corresponding software system in order to define linguistic rules requires flexibility in the way that these rules are stated. Due to the above, it was found necessary to develop a group of appropriate operators, as well

as the use of different types of variables in the **variables**, **structuredescription** and **structurecommands** fields of principles and transformations. The next sections describe the variables in these fields. Except the local variables that are valid inside a principle or transformation, also there are the grammar variables that can be used by more than one rule (principle or transformation) and are defined in the theory input. This ability facilitates the manipulation and the exchange of information between more than one X-bar trees that are used by the principles and transformations.

3. The variables type of the field “variables”

In the field **variables** of principles and transformations we can define variables of the type: node of tree, terminal element, anaphor, features of node or subtree and (see below in the examples the variables node1 and n in the first example and noun and v in the second one) in the values of every new variable we can use variables that we have already declared. They can be used in the fields **structuredescription** and **structurecommands** of the principles and the transformations.

The general format for variables declaration is:

type operator variable name **set** *variable value* **or** *variable value* **….**

The values of the node variables have the format:

name of node type of node : **features** features of node

the *type of the node* can be **barii**, **bari**, **bar** corresponding to X2, X1, X0 of the X-bar scheme. The *features of a node* is a list of : + *Name of the feature*, - *Name of the feature* or *Name of the feature*.

The anaphors connect terminals and/or subtrees.

4. The “structuredescription” field

4.1 The two different kinds of variables

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 tree that the rule is applied on, in a specific set of values (see below in the example "rule of dominance" the variables node1 and n). Also, we can use new variables of the **variables** type. They are defined automatically the first time 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 below the example "rule of dominance" the variables node2, t1 and t2). The main importance of these variables is that they provide an easy way to check if two or more elements of the **structuredescription** structure are of the same type and have the same values.

The second kind of variables can be of type node of tree, 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 name of this variable is the name that follows the **transformationvariable** operator. 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 below the example "rule of dominance" and the variables sd1 and sd2).

4.2 The tree operators

a) There are operators that declare constraints between two trees:

1) a *subtree1* must be or not (left or right) subtree of the *subtree2*
2) a *subtree1* must be or not subtree of a tree with a specific head node

b) Also, there are two operators that are applied to only one subtree of the **structuredescription** structure. The first one (**not**) declares that a *subtree* must not be subtree of an x-bar tree at a specific position and the second (**atree**) declares that the *subtree* must be subtree of another subtree of an x-bar tree at this position (see the example "rule of dominance" below)

c) Finally there are the operators **and**, **or** and **anytree** that determine:

1) *subtree1* **and** *subtree2* **and …** (all the *subtrees* are subtrees at this position of an x-bar tree)
2) *subtree1* **or** *subtree2* **or …..** (at least one of the *subtrees* is a subtree at this position of an x-bar tree)
3) **anytree** (any tree could be at this position of an x-bar tree)

5. The field “structurecommands” of principles and transformations

In the **structurecommands** field of the rules we can make checks, declare variables, change values of variables and determine the transformations of an x-bar tree.

5.1 The variables and their use

In the **structurecommands** field we can define new variables :

1) in the same way as the **variables** field
2) of features type that take values from tree nodes
3) of anaphor type that take values from terminals subtrees
4) of subtree type that take values an input x-bar structure

Additionally, we can change the values of variables by:

1) adding or removing their values
2) setting new values at terminals variables
3) calculating all the values of a variable according to the current values of the possible variables that are used in its values
4) seting new values at tree nodes, their features, their name or their type
5) seting new values at subtree variables
6) adding or removing a specific anaphor at tree, terminal or anaphor variables
7) adding or removing a specific feature at tree node variables

Finally, it is possible to check the existence of a variable or if it has been declared as grammar variable.

5.2 The Transformations in the “structurecommands” field

We can also transform an x-bar tree with a transformation rule.

transformations *transformation1* **also** *transformation2* **also ….**

We can perform more than one transformation in a transformation rule by using more than one such command and every transforamtion can change more than one parts of a tree. Every *transformation* in the above form is defined as following:

&*name of variable of type transformationvariable* **transform** *new value*

The *new value* can be a variable or a tree, a node or a terminal that may contain variables. The **&***name of variable of type transformationvariable* must be of **transformationvariable** type and has the same the type with the *new value* (see below the example "attachment of noun phrase" and the transformation of the transformation variable sd2).

5.3 The Grammar variables in the "structurecommands" field

All types of variables can be declared as grammar variables. These variables can be used by more than one **principle** or **transformation**. This means that a variable that has been declared in a rule can be used and manipulated by the next rule or rules in every field of the three fields of a principle or transformation (see below the first example and the variable sd1 in its **structurecommands** field). There is a operator that defines a variable as grammar variable and a second one that deletes a grammar variable. They can be used in the **structurecommands** field. Their main usage is in the theory input (see introduction).

5.4 Checks in the "structurecommands" field

In the **structurecommands** field we can also use structures of **if - then - else** type. In the **if - condition** part it is possible to apply checks at every type of elements (anaphors, terminals, features of nodes, nodes of trees, subtrees).

The operators about anaphors check if the anaphors of their right and left arguments are or not the same or if a specific anaphor exists. Their left and right arguments can be a sequence of anaphors, a terminal or a subtree.

The operators about terminals check if the terminals are equal or not, with or without checking their anaphors.

The operators about the features of the nodes check if their operands have or not the same features, if a specific feature exists, if a list of features is subset or have at least one common with another list of features. The left and the right arguments can be either nodes or features of node.

The operators about the nodes of the trees it is possible to check if two nodes are equal or not. Additionally, it is possible to check if they have or not the same name or type.

The operators about the trees check if are equal or not.

The above operators take arguments that can be variables or they can contain variables.

6. Examples

The following are examples of linguistics rules that have been expressed according to the presented methodology. Also, a grammar variable is declared in the field **structure commands** of the **principle** 'The rule of c-command'.

The rule of c-command [6]

An X element commands structurally (c-commands) an Y element, if and only if the first bifurcated node that dominates X, dominates also Y and neither X dominates Y nor Y dominates X.

principle 'The rule of c-command'.

variables **node** node1 **set** 'Verb' **bar or** 'Preposition' **bar**

also node n **set** 'N' **bar or** 'Noun' **bar**.

structuredescription

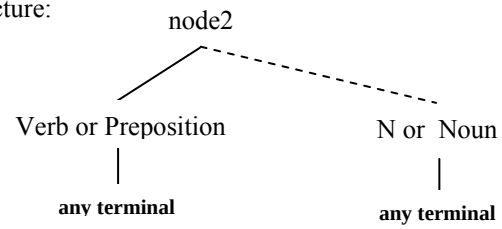
(**node &**node2, (**node &**node1, **terminal &**t1)**: transformationvariable** sd1,

atree (**node &**n,**terminal &**t2)**:transformationvariable** sd2)

structurecommands

comment &sd1**:**' c-commands '**:&**sd2, **addGrammarVariable** sd1.

The above principle acts upon an X-bar structure that has a sub tree of the following structure:



The discontinuous line means that the right sub tree can be at any depth as the operator **atree** describes.

The rule of noun phrase attachment [6]

This transformation describes the movement of a noun phrase.

transformation 'Attachment of noun phrase'.

variables **node** 'Noun' **set** 'N' **barii or** 'Noun' **barii**

also node 'V' **set** 'V' **bari or** 'Verb' **bari.**

structuredescription

(node &'V'**: transformationvariable** sd3**,**

subtree &sb1**, (node &**'Noun'**,anytree,anytree):transformationvariable** sd1

): transformationvariable sd2**.**

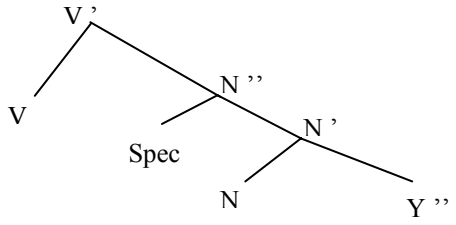
structurecommands

(&sd1 **addanaphor** i1, **%** addition of anaphor reference

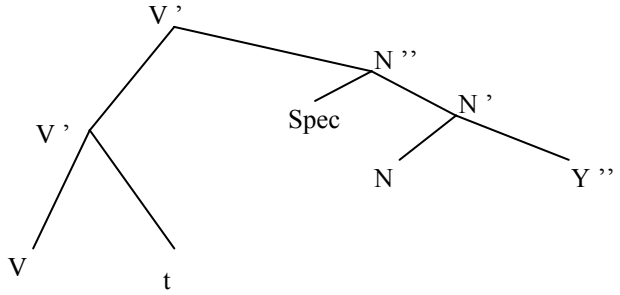
transformations &sd2 **transform** **%** declaration of transformations

(node &sd3**, (node &**sd3**, subtree &**sb1**, t:anaphor** i1**), subtree &**sd1**)).**

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



The produced X-bar structure is the following:



Since, the **sd1** variable of the first rule was declared as a grammar variable, if we execute these two rules we have two variables with the same name in the transformation rule. We must always use a different notation for the names of the variables that we intend to use as grammar variables.

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 and they are the rules that are described in the first section for the x-bar scheme.

The main characteristics of the presented methodology:

- It examines x-bar structures and their elements and rejects invalid structures.
- It transforms x-bar structures and their elements and produces new ones.
- It manipulates the semantic and syntactic information of the x-bar structures.
- It is necessary for the lexicon to have the syntactic and semantic information as a list of features:

  \+ *Name of the feature*, - *Name of the feature*, *Name of the feature*
- The features of the nodes of x-bar structures can be changed dynamically by using transformations. The syntactic and semantic information has simple structure, simpler than the HPSG[7]. The relation between the elements is determined and by the structure of the x-bar scheme.
- It is possible to define general rules that are applicable in many different x-bar structures since they are produced from the same general scheme and it is possible for them to adapt in different cases with the variables and the if-then-else checks.
- The simplicity and generality facilitates the implementation, the maintenance and extension of the corresponding applications.
- It is better for complicated embedded applications since the defined and produced structures are simpler and smaller and it is not necessary to have large memory size and strong processor.

References

[1] N. Chomsky: 'Remarks on nominalization'. In Jacobs & Rosenbaum (eds) Readings in English Transformational Grammar. Massachusetts: Xerox College, 184-221, 1970.

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

[3] N. Chomsky: Some concepts and consequences of the theory of Government and Binding. Cambridge: MIT Press, 1982.

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

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

[6] L. Haegeman: Introduction to Government and Binding 2nd Edition. Oxford: Blackwell, 1994.

[7] D. Tatar: Inteligenta arificiala. Cluj-Napoca: Editura Albastra, 2001.