8th International Conference on DEVELOPMENT AND APPLICATION SYSTEMS
Suceava, Romania, May 25 – 27, 2006

A COMPUTATIONAL METHODOLOGY AS AN ARTIFICIAL LANGUAGE ABOUT NATURAL LANGUAGE RULES

Konstantinos FOUSKAKIS
"Politehnica" University of Timisoara,
Faculty of Automation and Computers,
Bd. V. Parvan, No. 2, 300223 Timisoara, Romania
costasfous@yahoo.com

Abstract. *The scope of this paper is the presentation of the main linguistic and computational linguistic approaches and the main characteristics of the computational methodology as an artificial language that permits the declaration and application of linguistic rules on x-bar trees for their manipulation and construction.*
Keywords: *linguistics, computational linguistics, x-bar theory, x-bar trees, artificial language.*

Introduction

Natural language processing is a field that has concerned both artificial intelligence and the computational linguistics especially during the 60's and after. The modern theoretical linguistics is concerned with the scientific study of language as a natural communication system that is used by man. The linguistics is not limited only to one particular language but studies the general universal characteristics and has the following branches:

- *Phonetics* and *phonology*
  Phonetics is concerned on how the words of a language are pronounced. For the phonetic description, we use special symbols that are called *phonetic symbols*. The study of the phonemes' function within a particular linguistic system is called phonology.
- *Morphology*
  The word constitutes the basic unit both in the syntactic and the lexical level, while the words are not the minimum units of this level.
- *Syntax*
  The syntax is concerned with the rules under which the words are combined in bigger structures.
- *Semantics*
  Semantics is concerned with the meaning of the words and the sentences.
- *Pragmatics*
  Pragmatics is concerned with issues regarding the meaning of the sentences, as these are interpreted in a specific place, time and application field.

The standard linguistic theory that has influenced linguistics since 1957 up today, is the one that mainly Chomsky has developed in 1957 and still does up today. This theory is known as *generative transformational grammar.* The following can be considered as the basic stages in the evolution of his theory:

1) the *Syntactic Structures* in 1957.
2) the *Standard Theory* in 1965.
3) the *Extended Standard Theory*[1], the evolution of the *generative transformational grammar* in 70's.
4) The *Government and Binding*[2][3] in 1981, under the terms of the *Extended Standard Theory*.
5) Since the beginning of the 90's, a new tendency to modify the standard [4] has as starting point the role of the *principles of economy* and *minimalism*.

According to Chomsky's conclusions:

"The universal grammar is a theory of the linguistic ability initial state, before any linguistic experience"

Since the language is a basic element in a series of human manifestations, linguistics co-operate with other sciences:

- *Psycholinguistics* studies the relation between the language behavior and the psychological mechanisms.
- *Sociolinguistics* studies the ways in which language is affected by the social differences among the members of a linguistics society.
- *Stylistic linguistics* examines the selections of a litterateur in certain texts.
- *Mathematic linguistics* examines the mathematical properties of the language.
- *Computational linguistics* studies the language with the use of computers aiming to confront a series of subjects such as the automatic translation, the information retrieval, man-machine interface or the general development of the artificial intelligence.
- *Clinical linguistics* uses the linguistic theory mainly in order to study problems in pronunciation or writing.

Computational Linguistics

A. The HPSG grammars

This approach is not a transformational approach, like chomsky's theories, but it is based on the main mechanisms of the unifications grammars and supports features structures. It does not support rewriting rules in the general sence and there is no notion of deriving one structure from the other. It supports the sign structure with very detailed information from the lexicon. The representations are subparts of a single larger structure related by declarative constraints. It is said to be surface oriented because it provides a direct characterization of the surface order of elements in a sentence. The HPSG[5][10] puts a lot of emphasis on the precise mathematical modelling of linguistic entities. Because of the focus on precision, a lot of linguistic computer implementations are based on it. But, this approach has problems in translation systems because the sign of the source and destination language are not possible to be determined directly and it necessary for another semantic mechanism to be used.

B. The PATR grammar

The PART[10] has its initials from the words parse and translate. It is one of the oldest unification based approach (after the FUG) and it supports grammar (CFG) rules that consists of a mother category and zero or more daughter categories with a list of feature equations. A category is a set of feature-value pairs. A feature is an atom and a value can be an atom, a variable or a category. The feature equations on a rule set constraints on their values. The lexical items are viewed as rules without daughter categories. The PATR formalism is reasonably expressive. But, it doesn't have some desirable properties, like disjunction and negation of a set or list of value features. It is declarative, monotonic and reversible. Also, it is turing equivalent and if a PATR contains only atom-valued features it is as CFG of $Q(n^3)$.

C. The FUG grammar

The FUG has the initials of the words functional unification grammars. It was presented before the PATR grammars and in many ways is similar. The context free part in the PATR grammars is replaced by two features the cset and the pattern. They declare which items are the daughters of a category and at which order they appear. Multiple CFG 'rules' about one category are declared by disjunction. Finally, there is the feature value any that declares the requirement that it is obligatory of a feature to exist. This possibility adds the non-monotonic in the unification based approach.

D. The TAG grammar

The Tree Adjoining Grammar (TAG)[9] is defined as a tree rewriting system. In the

definition given traditionally, TAG is defined by a finite set of trees and an operation called adjoining to compose trees. It represents a extension of the basic rule rewriting scheme that underlies other modern grammatical formalisms. Unlike these string rewriting formalisms that write recursion into the rules that generate the phrase structure, a TAG captures recursion and dependencies (agreement, subcategorization filler-gap connections) into a finite set of elementary trees. Finally, they have extended with features structures that substitute the simple nodes.

E. The phrase-structure grammars

They were presented mainly by Chomsky in 1957. They[7][10] have the general form of x -> y, where x, y can be any combination of terminal and no-terminal elements. The different categories 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 grammars is that they have a very simple general format.

F. Transition networks

They are represented as finite states automatons[7]. 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.
- The check, the maintenance 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.

The Computational Methodology As Artificial Language

In my thesis there was an attempt[6][7][8] to develop a new systematic methodology as computer artificial language (a new approach in the domain). This language gives us the ability to define typically the rules of the generative transformational grammar in general (x-bar theory) and more general other linguistic rules on x-bar trees. It tries to integrate the Chomsky's ideas in a more general and abstract approach (different than parsers that do natural language processing and implements his theories) by combining the ideas from other theories and by presented better and more flexible mechanisms. The evolution of the linguistic theories results to a new basic scheme and from that scheme derives all the trees of any natural language. The basic scheme is the x-bar scheme. The above show the great value of a systematic methodology for the definition of the linguists' rules that could be applied on natural language trees that derive from the x-bar scheme. It permits the definition of fewer and

more general rules that are applicable in many trees since they are derivations of the same basic tree. The methodology that was developed is open to the changes in the linguists' theory and enables us to set the rules that are necessary each time. These rules are set in a simple way, while they are also more descriptive, flexible and powerful[8]. Also, we are enabled to deal in a general and uniform way the several linguistic phenomena. The linguistic knowledge of this methodology has a structure which is presented in the Fig. 1.

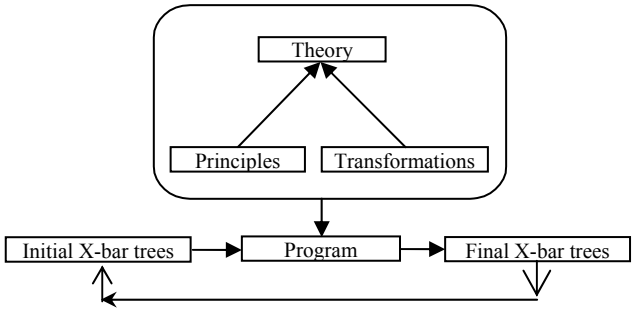

Figure 1. The organization of the linguistic knowledge.

Let 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 of Chomsky (see Fig 2). Their general format is according to the following rules:
X2 → Spec X2, X2 → Spec X1, X → X1 Y2, X1 → X0 Y2
Spec → X0, Spec → X2, X0 → terminal

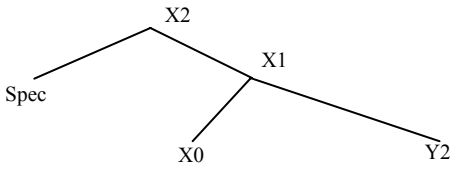

Figure 2. The x-bar scheme.

They support multiply anaphoric connections between terminals and/or trees that are in the same or in a different tree structure. The features that support in their nodes are lists that can contain the following elements: +feature, -feature, feature, featureX=featureY or [feature1, feature2, …., featureX]=featureY.

- Principles

It contains all the principles that have defined so far. The principles check an x-bar structure 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 all the transformations that have defined so far. 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.

- 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 Linguistic Program

It is the part of the linguistic system which declares the rules of the theory, principles,

transformations sets that are applied on the initial x-bar structure and their order.

- The final x-bar trees

It contains the generated x-bar structures according to the linguistic program.

A. The principles and Transformations

We can express the linguistic knowledge that is of our interest by using these rules. Thus, we can process the natural language trees accordingly. The complexity and the number of the rules depend on our requirements.
They are stated using the following general pattern:

- **principle** / **transformation** The name of the rule
- **variables** (The variables are declared which correspond to parts of an x-bar structure. They can have more than one values and be defined hierarchically by reusing already defined variables)
- **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 **transformation** variables that can be defined only in the **structureDescription** field and are used for the description of the transformations in the **structureCommands** field of the **transformation** rules.
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. Also, we can use new variables of the **variables** kind 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. The second kind of variables can be of type node of structure, terminal or subtree. They can be used in combination with the other kind of variables. 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 the rule is applied.
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. 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 principles and transformations input and output structures are according the Fig. 3. It is the possibility to check the accepted rate between the input and output structures by using the command **acceptedRate**(*Rate*).

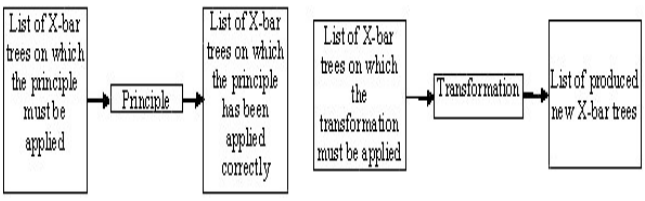

Figure 3. The general functionality of the Principles and Transformations.

B. 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 set of x-bar structures 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 set. 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. 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[6]. Additionally, it is possible to combine this with another software system that produces the x-bar trees. That system can use a set of very simple rewriting rules (even only lexical rules that produce all the X0 -> Terminal subtrees of the words of the phrases). The rules can be based only on general phrase structure information.

The conclusions:

It is an artificial computer language (a new approach in the domain) with variables, operators, if-then-else structures and repetitions-recursions dedicated in the natural language processing.

It provides a mechanism that examines and transforms x-bar structures by permitting multiply simultaneous transformations.

It manipulates the syntactic, semantic and pragmatic information of the x-bar structures. Additionally, it supports the checking of the accepted rate at a rule application and permits the evolutionary changing of the x-bar structures.

The syntactic and semantic information has simpler structure than the HPSG. The relation between the elements is determined and by the structure of the x-bar scheme. The variables have much stronger functionality with hierarchical way of declaration which is better than in the unification grammars like the HSPG (the most interesting computational linguistic approach). The features of the nodes of x-bar structures can be changed dynamically by using transformations.

It is more flexible than the TAGs[8] in the operations related with the trees construction and features manipulation by using its flexible system of rules, variables and operators.
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 have the same structure and the same way of linguistic treatment.

It is according to the Chomsky ideas of the universal grammar theory, it combines his ideas in more general and abstract new approach and it is unique in this sense. (operators like aFirstTree or leftMost intergrades his ideas of government theory).

It is a different approach than the classical parsers that implement a version of the Chomsky's theory (GB-government and binding or Minimalistic Program).

It is an artificial language that permits the declaration of natural language rules and its main characteristics are simplicity and generality.

It supports anaphoric connections inside or outside of an x-bar structure.

A better and simpler covering of the ambiguity problems of the phrases of natural languages by supporting more that one structures in the structureDescription field of the principles and transformations connected by the and, or operators and by using the variables. This fact permits the examination of different linguistic cases in one rule and the execution of different commands in a man machine interface system.

It integrates ideas from different theories.

The simplicity, flexibility and generality facilitate the implementation, the maintenance and extension of the corresponding applications.

It is better for embedded applications since the defined and produced structures are simpler and smaller (restricted only in a subset of binary trees) and it is not necessary to have large memory size and strong processor.

It facilitates the man-machine communication for the execution of commands and the retrieving of the required information that is expressed by natural language phrases. Possible applications can be in the domain of railway, airway or tourist information software systems. Also, it is possible to be used in the automotive domain to facilitate the communication with the today complicate information systems.

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. (1995) *The Minimalist Program*, Massachusetts: MIT Press, 1995.

[5]Copestake, A. (2002) *Implementing Typed Feature Structure Grammars*, Standford, CSLIS.

[6]Fouskakis, K. (2000) *An open system for linguistic rules on the x-bar trees*, Ukrainian Journal of Computational Linguistics 4, Ukraine.

[7]Fouskakis, K (2004) *An overview of a computational approach for linguistic rules on the x-bar trees*, 7th International Conference on Development and Application Systems DAS2004: 353--360, Suceava, Romania.

[8]Fouskakis, K. (2005) *The Basic Notions of the Tree Adjoining Grammars and a Methodology as Artificial Language about Linguistic Rules*, Intelligent Linguistic Technologies Conference – World Academy of Science, Las Vegas Nevada USA.

[9]Millett, R. and Lonsdale, D., (2004) *Expanding tree adjoining grammar to create junction grammars trees*, 7th International Workshop on Tree Adjoining Grammar and Related Formalisms: 163--170, Vancouver, Canada.

[10]Tatar, D. (2003) *Inteligenta Artificiala – Aplicatii in Prelucrarea Limbajului Natural*, Cluj-Napoca, Editura Albastra.