Long-Distance Dependencies and Applicative Universal Grammar

Sebastian Shaumyan
Yale University, U.S.A., e-mail: shaumyan@minerva.cis.yale.edu
Frédérique Segond
Rank Xerox Research Centre, France, e-mail: frederique.segond@xerox.fr

Abstract

To deal with long-distance dependencies, Applicative Universal Grammar (AUG) proposes a new type of categorial rules, called *superposition rules*. We compare the AUG rules with the alternative rules of Steedman's Combinatory Categorial Grammar (CCG) (Steedman, 1987, 1988, 1990; Szabolcsi, 1987; Ades and Steedman, 1982). In contrast to Steedman's rules, the AUG rules are free from inconsistencies in their semantic interpretation, free from spurious ambiguity. The superposition rules are based on the *Theory of Type Superposition*, established independently of the problem of long-distance dependencies and having a broad unifying power.

1. Characterization of Applicative Universal Grammar

Applicative Universal Grammar (AUG) is a linguistic theory that uses the formalism of categorial grammar as a means for representing the structure of language. AUG has two levels: 1) the study of the grammatical structure in itself ***(genotype grammar)***, and 2) the study of the linear representation of the grammatical structure ***(phenotype grammar)***. AUG includes a system of combinators (Curry and Feys, 1958) and formulates semiotic concepts, principles, and laws that determine the functioning of natural languages as sign systems (for a complete description of AUG, see Shaumyan, 1974, 1977, 1987; Desclés, 1990; Segond, 1990a; some applications of AUG are discussed in Shaumyan 1989, 1991).

AUG is based on the relation *operator-operand*, which corresponds to the relation *function-argument* in categorial grammar. We prefer the terms *operator-operand* for reasons similar to those given by Hindley and Seldin (1986, pp. 44-45). In AUG categories are generated recursively by the type-forming operator *O*, and are called *O-types*. AUG recognizes two primitive types—terms (nouns and noun-phrases) and sentences, denoted by *t* and *s*, respectively. The rule for generating *O*-types is:

1) The primitive types *t* and *s* are *O*-types.
2) If *x* and *y* are *O*-types, then *Oxy* is an *O*-type. (1)

For the sake of brevity, we use the term *type* in the sense of the *O*-type. Taking *t* and *s* as primitives, we generate the inductive class of types: *t, s, Ott, Oss, Ots, Ost, OtOts, OOtsOts*, and so on.

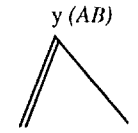
In representing the types we use the parentheses-free Polish notation, which is more convenient than Curry's notation with internal parentheses.

The basic rule of combination of phrases is the ***Rule of Phrase Application***, which is defined as follows:

Phrase *A* of type *Oxy*, called an *operator*, combines with phrase *B* of type *x*, called its *operand*, to form phrase *AB* of type *y*, called its *resultant:*

$$\frac{\text{Oxy } A \qquad \text{x } B}{\text{y } (AB)} \tag{2}$$

The applicative tree of (2) has the form:



(3)

The concept of ***immediate constituents*** is defined as:

If phrase *A* is an operator and phrase *B* is its operand, then they are immediate constituents of the resultant *(AB)*. (4)

The concept of ***closeness*** is defined as:

Given phrases *A* and *B* that are immediate constituents of phrase *(AB)*, if *A* is a complex phrase comprising immediate constituents *C* and *D*, then the syntactic and semantic connection between *C* and *D* is closer than the syntactic and semantic connection between *A* and *B*. (5)

Under definition (5) various degrees of relative closeness of syntactic and semantic connection between immediate constituents are distinguished depending on the complexity of a phrase.

In phenotype grammar the application operation is constrained by two principles: the Principle of Adjacency of Operators and Their Operands and the Principle of Uniqueness of Immediate Constituents.

Principle of Adjacency of Operators and Their Operands:

An operator and its operand must be adjacent elements of a sequence, so that the operator either directly precedes or directly follows its operand. (6)

Under the Adjacency Principle we have two new rules — the notational variants of operator application: one for forward combination and one for backward combination:

$$\frac{\text{Oxy } A \qquad \text{x } B}{\text{y } (AB)} \tag{7}$$

$$\frac{x\ A \qquad Oxy\ B}{y\ (AB)} \tag{8}$$

These rules are called the ***Linear Precedence Rules***. An alternative notation for these rules splits the type-forming operator O into indexed type-forming operators O_r and O_l which generate types of the form O_rxy and O_lxy. The operator of type O_rxy has its operand on its right, and the operator of type O_lxy has its operand on its left. So the ***Linear Precedence Rules*** may be presented as follows:

$$\frac{O_rxy\ A \qquad x\ B}{y\ (AB)} \tag{9}$$

$$\frac{x\ A \qquad O_lxy\ B}{y\ (AB)} \tag{10}$$

Here is an example of applying this notation:

$$\frac{\dfrac{O_rtO_lts\ \textit{bought} \qquad t\ \textit{newspapers}}{t\ \textit{John} \qquad O_lts\ \textit{bought newspapers}}}{s\ \textit{John bought newspapers}} \tag{11}$$

Given the Rule of Phrase Application and Linear Precedence Rules, we can combine the two rule formats into one system, as is done with the corresponding rule formats in Generalized Phrase Structure Grammar (Gazdar et al., 1985: 44-50).

Principle of Uniqueness of Immediate Constituents:

> If phrase *A* and phrase *B* are immediate constituents of phrase *C*, then neither *A* nor *B* can be an immediate constituent of another phrase *D*. (12)

To illustrate, consider the sentence: *John loves vodka.* Here *loves* and *vodka* are the immediate constituents of *(loves vodka)*, and *John*.and *(loves vodka)* are the immediate constituents of *(John (loves vodka))*. Under the above constraint, this analysis precludes analyzing this sentence as: *((John loves) vodka)*.

In terms of algebra, the Principle of Uniqueness of Immediate Constituents corresponds to ***non-associativity:*** AUG is a non-associative system.

To make the AUG notation compact, we introduce recursively defined ***adjoined symbols*** (Shaumyan 1987: 199):

> A type symbol is called *adjoined* if it is introduced into the type system by a definition of the form:
> $$z = Oxy$$
> where z denotes an adjoined type and Oxy denotes a type where x and y are either other adjoined type symbols, or t, or s. (13)

This type of definition is called ***definitional reduction***. By this process all adjoined type symbols are defined in terms of the ultimate definientia t and s. We can introduce as many adjoined type symbols as we need. Here are examples of the definitional reduction for adjoined type symbols that will be used below:

$$\begin{aligned} p_1 &= Ots \\ p_2 &= Otp_1 = OtOts \\ p_3 &= Otp_2 = OtOtOts \\ d_1 &= Op_1p_1 = OOtsOts \\ d_2 &= Op_2\,p_2 = OOtp_1Ot\ p_1 = OOtOtsOtOts \end{aligned} \tag{14}$$

AUG claims that a typology of word order must be based on a comparison of specific word orders in individual languages with a canonical word order as defined in genotype grammar. The canonical word order requires that an operator precedes its adjacent operand. For example, the canonical form of the sentence *My older brother bought an interesting book yesterday* is: *(yesterday ((bought (an (interesting book))) (my (older brother))))*.

2. Long-Distance Dependencies in CCG

Consider, for example, the phrase *Apples which Harry eats*. This phrase contains three sets of binary dependences: 1) *apples-eats*, 2) *which-eats*, and 3) *Harry-eats*. The first two sets consist of discontinuous constituents. This is an instance of the phenomenon called *intersecting dependencies*. Intersecting dependencies arise when one set of discontinuous constituents is intercalated by another set of discontinuous constituents in the surface expression. To find an adequate formal characterization of discontinuous constituents and intersecting dependencies is one of the central problems for categorial grammar, as for any linguistic theory that is concerned with linear representation of expressions. This problem induced some linguists to introduce new rules extending the formalism of categorial grammar. Steedman's Combinatory Categorial Grammar (CCG) proposes the following analysis of the phrase *Apples which Harry eats* (1987: 415; presented here in the AUG notation):

(apples)	*which*	*Harry*	*eats*	
	OOtsOtt	OOtss	OtOts	
		———	———	compose backward
		Ots		
	———	———		apply forward
	Ott			(15)

In (15), *subject type raising* (assigning *OOtss* to *Harry)* in conjunction with *composition* is used to resolve the difficulty caused by gapping involved in the extraction of the direct object of the finite verb *eats*.

Forward and ***backward composition*** are defined as follows (in terms of AUG):

> Under the rule of *"compose forward"*, *A* of type *Oxy* and *B* of type *Oyz* combine to yield the result *(AB)* of type *Oxz*. Under the rule of *"compose backward"*, *A* of type *Oyz* and *B* of type *Oxy* combine to yield the result *(AB)* of type *Oxz*. (16)

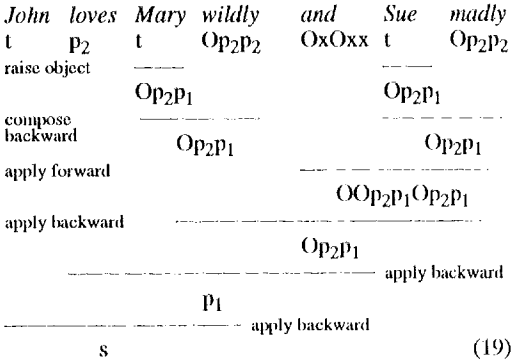
Type raising is defined as an operation whereby an operand acquires a new type that turns it into an operator over its operator. The general rule of type raising in the AUG notation is:

$$x \rightarrow OOxyy \tag{17}$$

For example, ***subject type raising*** is defined in terms of AUG as follows:

> *Subject type raising* is a process by which a subject of type *t* acquires the type *OOtss*, which turns it into an operator over the predicate of type *Ots*. (18)

As another example of the analysis that uses type raising, let us consider the sentence *John loves Mary wildly and Sue madly* (Bouma, 1989: 25). Using type raising and composition, the analysis of this sentence can be presented as follows in the AUG notation:



(19)

In (19), to resolve the difficulty caused by gapping involved in the coordination operation, object type raising is used (assigning Op_2p_1 to *Mary* and *Sue)* along with composition.

Does the CCG machinery produce adequate syntactic and semantic representations of the structure of a sentence? What is the semantic interpretation of type raising?

It is claimed that the nominative case morphology in languages like Latin determines a noun-phrase argument like *Balbus* to be something that must combine with a predicate (Steedman, 1990: 221). But case endings are not reliable criteria for determining facts of syntax and semantics. In Russian and many other languages the nominative has no endings. Semantically, *predicate + subject* is an attributive connection just as *adjectival modifier + subject.* Predicate and adjectival modifier are ***determining members,*** and subject is their ***determined member.*** Accordingly, we get the proportion:

> *predicate : subject = adjectival modifier : subject* (20)

This means that if the syntactic categorial system is to conform to the semantic categorial system, predicates must be operators over subjects just as adjectives. Type raising transforming subjects into operators over predicates conflicts with the semantic categorial system.

Furthermore, if in (19) we have a correct analysis of coordination, we should be able to deduce the two interpretations of the sentence *John loves Mary wildly and Sue madly*: "John loves Mary wildly, and John loves Sue madly" and "John loves Mary wildly, and Sue loves Mary madly." This is a classic case of ambiguity with coordination (we do not know if the second conjunct is subject or object). Unfortunately CCG fails to distinguish between the two interpretations.

The other well-known problem with type raising is ***spurious ambiguity***. Spurious ambiguity is multiple analyses of one sentence, all of them related to the same semantic interpretation. For instance, just by using subject and object type raising one obtains four different analyses for a simple sentence:

John	*loves*	*Mary*		
t	Otp_1	t	→	s
Op_1s	Otp_1	t	→	s
Op_1s	Otp_1	Op_2p_1	→	s
t	Otp_1	Op_2p_1	→	s

(21)

These four analyses are associated with just one meaning: *((loves Mary) John).*

There are other difficulties with type raising. We see that in (19) *(Mary wildly)* and *(Sue madly)* are assigned type Op_2p_1, which is associated with the accusative function. It is very difficult to accept that *(Mary wildly)* or *(Sue madly)* are direct objects of *love,* or that they are at all compatible. The correct analysis is: the adverbs *wildly* and *madly* are modifiers of the verb *love,* and the nouns *Mary* and *Sue* are direct objects of the verb *love. (Mary wildly)* and *(Sue madly)* are phantom constituents that do not correspond to any syntactic or semantic reality.

Type raising corresponds to the combinator $\mathbf{C}_*$ and composition correspond to combinator **B**. Both are powerful tools when properly used. One of the conditions of their proper use is respect for constituency. AUG uses these combinators widely when their use is justified.

The main sin of CCG is that it fails to recognize that syntactic and semantic connections are non-associative. CCG bans from linguistics the normal non-associative constituency analysis based on the explicit or implicit recognition of the hierarchy of relative syntactic and semantic closeness of connections between immediate constituents of a sentence.

3. The AUG Theory of Type Superposition

An alternative method of parsing gapping constructions is based on the Theory of Type Superposition. To explain our new method, we need to outline this theory briefly.

Given a syntactic unit, a secondary syntactic type may be superposed on its inherent, primary syntactic type so as to form a new bistratal, syncretic type. For example, when the suffix *-ing* is used to change the finite form of the verb *to instruct* into a verbal noun—the so-called gerund—*instructing,* we have a case of the superposition of type *t* on type *OtOts.* The verbal noun retains the syntactic function of the verb to *instruct:* it can take an object in the accusative *(on instructing him)* and an adverb *(He suggested our immediately instructing them).* The same is true of the English or French infinitives: they behave both like verbs and nouns. For example, in the French sentence *Lire des livres est divertissant* or in the English sentence *To read books is fun* the infinitives take direct objects like finite verbs and simultaneously are subjects like nouns. The suffix *-ing* (or any other similar device) we call a

superposer, and the finite form of the verb *to instruct* with respect to the suffix *-ing* we call the ***superponend*** of *-ing*. The suffix *-ing* superposes the syntactic type *t* on the syntactic type *OtOts* of the verb *to instruct* so as to combine them into a new syncretic syntactic type.

We can formalize the notion of ***superposition*** as follows:

> Let *E* be an expression of type *x*, and let *E* take on type *y* on type *x*. Then *E* shall be said to belong to type *z* such that *z* is stratified into *y superposed* onto *x*. Type *z* is represented by the formula:
>
> ‹*y*:*x*› (22)

where the colon (:) indicates the stratification of type *z* into y superposed on *x* enclosed into angle brackets. The right part of the formula indicates the primary type of *E*, and its left part indicates the secondary type of *E*.

Definition of ***superposer:***

> An operator *R* of type *Ox*<*y*:*x*> shall be called a *superposer.* (23)

Rule of Superposition:

$$\frac{\text{Ox<y:x>}\ A \qquad \text{x}\ B}{\text{<y:x>}\ (AB)} \tag{24}$$

Type superposition has important consequences both for linguistic theory and computational linguistics, the discussion of which is beyond the scope of the present paper. We will only focus on the topic of our paper—long-distance dependences. For the lack of space we must confine ourselves to some examples of our approach that concern topicalization, relative clauses, and gapping (a detailed presentation of the theory of type superposition is given in Shaumyan and Segond, 1993).

4. Long-Distance Dependencies in AUG

We propose a new approach to parsing gapping sentences that allows us to dispense with the concept of type raising. AUG claims that gapping superposes secondary types on primary types of the adjacent syntactic units of a sentence, thereby establishing new relations between them on top of the old ones preserved in superposition.

Here is the AUG alternative analysis of the phrase in (15):

(apples)	*which*	*Harry*	*eats*	
	OxOtt	t	OtOts	
			——— superpose Ots	
			‹Ots:OtOts›	
		——— apply backward		
		s		
——— apply forward				
Ott				(25)

Under the characterization of superposition in the foregoing section, the obligatory absence of the adjacent direct object in *apples which Harry eats* is a contextual operator superposing type *Ots* on type *OtOts* of *eats*. The superposition yields the same result as the hypothetical application of *eats* to its absent direct object. That is, the secondary type of *eats* is equivalent to the type of the hypothetical combination *eats + direct object*. Then, the application of *eats* to *Harry* results in *Harry eats* of type *s*. Following Benveniste's analysis of relative pronouns (1966: 208-224), we consider them operators having variable operands; hence, type *OxOtt* is assigned to *which*.

Type superposition is a strictly formal concept reflecting observable formal processes of language. There are observable strictly formal criteria for defining superposition. A derived syntactic unit with a syncretic type is always more complex than the initial one; it consists of two parts: initial syntactic unit + superposer. So *read-ing*, where *-ing* is a superposer, is more complex than *read*.

But where are formal markers of superposition in (25)? The answer is that superposers, as all other language items, are signs, and a sign is not necessarily a sequence of sounds. It may be a change of stress, an alternation, a change of word order, a change of grammatical context, etc. (Shaumyan, 1987: 3-5). In (25) the syntactic configuration of the phrase *apples which Harry eats* contains observable contextual signs of superposition. To do justice to this fact we have to use an adequate formalism.

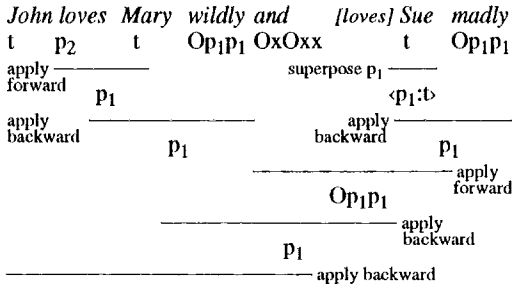
AUG includes two principles to describe superposition: the Principle of Elimination of Empty Constituents and the Principle of Syntactic Assimilation.

Principle of Elimination of Empty Constituents:

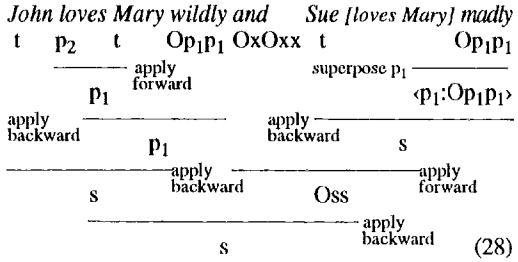
> Given a syntactic group of an operator *A* of type *Oxy* and its operand *B* of type *x*, either *A* or *B* may be empty: 1) if *B* is empty, empty *B* serves as a contextual sign superposing type *y* on type *Oxy* of *A*, so that *A* is assigned the syncretic type ‹*y*:*Oxy*›; and 2) if *A* is empty, empty *A* serves as a contextual sign superposing type *y* on type *x* of *B*, so that *B* is assigned the syncretic type ‹*y*:*x*›. (26)

The Principle of the Elimination of Empty Constituents characterizes natural syntactic connectivity. When in the group *operator:operand* the empty operand is eliminated, the operator represents the whole group and is assigned the type of the whole group. Conversely, when in the group *operator:operand* the empty operator is eliminated, the operand represents the whole group and is assigned the type of the whole group.

Let us turn to the sentence *John loves Mary wildly and Sue madly* in (19). As was said above, this sentence is ambiguous: *Sue* may be a subject or an object. A correct syntactic analysis of this sentence must reflect this semantic ambiguity. Depending on two possible interpretations of this sentence, we discover two different gappings here: *"John loves Mary wildly, and* [loves] *Sue madly"* and *"John loves Mary wildly, and Sue* [loves Mary] *madly"*. In the light of the Principle of Elimination of Empty Constituents, AUG proposes the following two analyses of the sentence to reflect two different gappings:



(27)

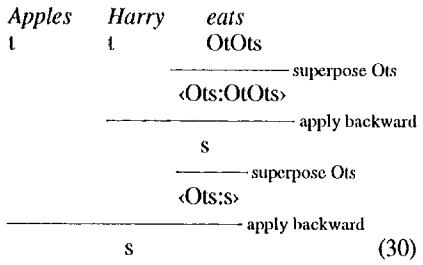


(28)

Principle of Syntactic Assimilation:

> Given two phrases *A* and *B* belonging to types incompatible under the Rule of Phrase Application, one of these phrases can change its type by superposition so that the types of the two phrases become compatible, if the relation *A:B* is analogous to some relation *X:Y* between phrases of compatible types. (29)

Consider the sentence *Apples Harry eats.* This sentence is an example of long-distance dependency because the subject intervenes between the direct object and the predicate. Here is the AUG analysis:



(30)

We observe that in (30) *Apples* is the topic and *Harry eats* is the comment. Since the proportion *topic : comment = subject : predicate* holds, type *Ots* is superposed on type *s* of *Harry eats*.

The Principle of Syntactic Assimilation is completely general: it concerns both long-distance and immediate dependencies. Consider the phrase *gold watch.* Both words have type *t*. Therefore, they belong to incompatible types. But since the proportion *gold : watch = golden : watch* holds, type *Ott* is superposed on type *t* of *gold.*

The phenomenon of superposition must not be confused with polymorphism. Polymorphism is a situation when a word is assigned several types, having equal syntactic weight. For example, an English adverb can be assigned at least three types having equal syntactic weight: Op_1p_1, Op_2p_2 or Op_3p_3, depending whether it modifies an intransitive, transitive or ditransitive verb. Here we have an equality between the types. But in the above example the noun *gold* remains a noun even though it modifies another noun. To describe polymorphism in a compact way, Frédérique Segond has introduced the concept of *type variable.* Thus, the above and other types that can be assigned to an adverb are coded by the formula *Oxx* (Segond, 1990a: 131-132). Other cases of polymorphism are exhibited by the conjunction *and,* which can combine two sentences, two nouns or any units belonging to identical types; and by the relative pronoun *which* of type *OxOtt,* mentioned in (25). Depending on different cases of polymorphism, we introduce various type variables.

5. Conclusion

We have compared two alternative methods of computation of long-distance dependencies: the CCG and AUG methods. Both methods are consistent with respect to their mathematical machinery.

The essential difference between the two methods is that while AUG with its theory of superposition expands its formalism to reflect the linguistic reality, CCG, by abandoning the normal constituency analysis, gets caught up in its formalism to lapse into linguistic unreality.

CCG analysis produces phantoms, as:

> *(He must) leave.*
> *((He must) love) her.* (31)

This startling analysis does not permit us to correctly describe agreement, government and clitization. These artificial constituent structures are completely divorced from the syntactic and semantic reality.

The CCG's use of type raising in conjunction with type composition changes the initial natural types assigned to words into artificial types and produces artificial constituents for the convenience of computation. By contrast, superposition, in conjunction with the Principle of Elimination of Empty Constituents and the Principle of Syntactic Assimilation, changes natural types into natural types and produces syntactically and semantically appropriate constituents without any sacrifice in the consistency of the mathematical formalism or in the convenience of computation.

In support of their departure from the accepted analyses of syntactic constituents the proponents of the CCG refer to psychological studies on speech recognition claiming that human "recognizer" works "from left to right".(Ades and Steedman, 1982: 517-518).

Two problems arise here. First, although human speech is linear and the words of a sentence are produced from left to right, so to say, that does not mean that the listener analyzes speech word by word. It is reasonable to assume that the listener performs the analysis of a sentence first by syntactic blocks and then globally. There is no conclusive psychological evidence that the hearer's recognition of the sentence structure corresponds to the CCG method that disposes with the normal constituency analysis.

Second, psychological phenomena are irrelevant to confirmation or refutation of linguistic theories, because lin-

guistics is completely independent of psychology. True, linguistic processes involve the psychological processes in the human mind. But logical and mathematical reasoning also involve psychological processes. However, nobody tries to base logic or mathematics on psychology. Linguistics is part of semiotics—the theory of sign systems. Sign systems, as well as mathematical systems, are in the human mind. But the laws of semiotics and mathematics are different from the laws of psychology.

One may argue that computational linguistics is different from ordinary linguistics and therefore any parser will do for computational linguistics as long as it "works". We believe that good computational linguistics must be good linguistics as well. Both ordinary and computational linguistics must share common theoretical principles characterizing the nature of human language. Computational linguistics is not second-rate linguistics where anything goes.The real difference between the two types of linguistics is that computational linguistics expands ordinary linguistics by rules characterizing its interaction with computers rather than distorts it. Computational linguistics is at the cutting edge of the study of human language: it must enrich our understanding of all its aspects, rather than fudge the linguistic concepts for the sake of the ease of the implementation.

The irreparable defect of the CCG method is that it produces phantom constituents and phantom structures that preclude a correct analysis of linguistic processes.

The CCG method is interesting and important as an experiment in the application of combinators in linguistics. The negative results of this experiment are important in that they reveal the hazards involved in the use of combinators (for use of combinators in AUG, see Shaumyan, 1987; Desclés, 1990; Desclés et al. 1985, 1986).

As an instrument of cognition mathematics has a specific function—to be a tool of deduction. But deduction is neutral to the value of ideas. It is like a mill: if you put grain into it, you will get flour; and if you put in chaff, you will get processed chaff. Mathematical consistency does not guarantee a correct description of reality. "Side by side with mathematization of knowledge, mathematization of nonsense also goes on (Nalimov, 1981: 149)." The use of mathematics as a tool of deduction makes sense only when the initial ideas from which we deduce their consequences have value (on use and abuse of mathematical formalism, see Shaumyan 1987: 28-29, 318-321).

In conclusion, we would like to say a few words about the computer implementation of AUG. Frédérique Segond has implemented AUG and its theory of superposition to deal with infinitive clauses and gerunds in French (for a complete description of the parser, see Segond, 1990a). This parser has been implemented in PLNLP (Programming Language for Natural Language Processing, described in Heidorn, 1972) at the IBM Research Center in Paris. The parser uses a machine dictionary of 50,000 entries and was tested on more than one hundred different types of sentences, including constructions such as relative clauses, simple cases of coordination, infinitive clauses, and gerunds, among others. Currently Sebastian Shaumyan is working on implementing AUG in functional programming languages (Miranda, Haskell).

References

Ades, Anthony and Steedman, Mark. 1982. "On the Order of Words". *Linguistics and Philosophy, 4,* pp. 515-578.

Benveniste, Émile. 1966. *Problèmes de linguistique générale.* Éditions Gallimard.

Bouma, Gosse. 1989. "Efficient Processing of Flexible Categorial Grammar". In *Proceedings of ACL (European Chapter),* Manchester, pp. 12-26.

Curry, Haskell B. and Feys, Robert. 1958. *Combinatory Logic.* Vol. I. Amsterdam: North-Holland Publishing Company.

Desclés, Jean-Pierre; Guentchéva, Zlatka; Shaumyan, Sebastian. 1985. *Theoretical Aspects of Passivization in the Framework of Applicative Grammar.* Amsterdam & Philadelphia: John Benjamins Publishing Company.

Desclés, Jean-Pierre; Guentchéva, Zlatka; Shaumyan, Sebastian. 1986. "Reflexive Constructions: Towards a Universal Definition in the Framework of Applicative Grammar." *Linguisticae Investigationes, 2.*

Desclés, Jean-Pierre. 1990. *Languages applicatifs, langues naturelles et cognition.* Paris: Hermes.

Gazdar, Gerald; Klein, Ewan; Pullum, Geoffrey; Sag, Ivan. 1985. *Generalized Phrase Structure Grammar.* Cambridge, Massachusetts: Harvard University Press.

Heidorn G. E. 1972. *Natural Language Inputs to a Simulation Programming System.* Technical report from the Naval Postgraduate School.

Hindley, J. R. and Seldin, J. P. 1986. *Introduction to Combinators and λ-Calculus.* Cambridge: Cambridge University Press.

Nalimov, V. V. 1981. *In the Labyrinth of Language: A Mathematician's Journey.* Philadelphia: ISI Press.

Segond, Frédérique, 1989. "Grammaire catégorielle enrichie: une implementation". *Proceedings of the 7th Congress AFCE RFIA,* Paris, pp. 599-613.

Segond, Frédérique. 1990a. *Grammaire catégorielle du français. Etude théorique et implantation. Le système GraCE (Grammaire Catégorielle Etandu).* Paris: IBM FRANCE.

Segond, Frédérique, 1990b. "Approches des grammaires catégorielles". *Mathématique, Informatique et Sciences Humaines, 110,* pp. 47-60.

Shaumyan, Sebastian. 1974. *Applikativnaja grammatika kak semanticeskaja teorija jazyka.* Moskva: Nauka.

Shaumyan, Sebastian. 1977. *Applicative Grammar as a Semantic Theory of Natural Language* (translation of Shaumyan, 1974). Chicago: University of Chicago Press.

Shaumyan, Sebastian. 1987. *A Semiotic Theory of Language.* Bloomington & Indianapolis: Indiana University Press.

Shaumyan, Sebastian. 1989. "A Semiotic Theory of Knowledge Representation and Symbolic Computing." *Proceedings of the Fourth International Conference on Symbolic and Logical Computing.* Madison, SD.

Shaumyan, Sebastian. 1991. "Applicative Universal Grammar and Translation". *Proceedings of the Fifth International Conference on Symbolic and Logical Computing.* Madison, SD.

Shaumyan, Sebastian and Segond, Frédérique, 1993. "The Theory of Superposition of Applicative Universal Grammar". *Colloque «Informatique & Langues Naturelles» (I.L.N.) '93.* Nantes: I.R.I.N.

Steedman, Mark. 1987. "Combinatory Grammars and Parasitic Gaps". *Natural Language and Linguistic Theory, 5,* pp. 403-439.

Steedman, Mark. 1988. "Combinators and Grammars". In Oehrle R. T., Bach E., Wheeler D. (eds.), 1988, *Categorial Grammars and Natural Language Structures,* Dordrecht: D. Reidel Publishing Company, pp. 207-263.

Steedman, Mark. 1990. "Gapping as Constituent Coordination". *Linguistics and Philosophy, 13,* pp. 207-263.

Szabolcsi, Anna. 1987. "On Combinatory Categorial Grammar". In *Proceedings of the Symposium on Logic and Language, Debrecen,* Budapest: Akademiai Kiado, pp.151-162.