Linguist Philos (2006) 29:419–454
DOI 10.1007/s10988-006-0008-x

RESEARCH ARTICLE

A grammar systems approach to natural language grammar

M. Dolores Jiménez López

Received: 25 March 2003/Accepted: 16 March 2006 /
Published online: 17 October 2006

Abstract Taking as its starting point significant similarities between a formal language model—Grammar Systems—and a grammatical theory—Autolexical Syntax—in this paper we suggest the application of the former to the topic of the latter. To show the applicability of Grammar Systems Theory to grammatical description, we introduce a formal-language-theoretic framework for the architecture of natural language grammar: *Linguistic Grammar Systems*. We prove the adequacy of this model by highlighting its features (modularity, parallelism, interaction) and by showing the similarity between this framework and accepted and well-known grammatical models (e.g. Autolexical Syntax).

Keywords Formal language theory · Grammar systems · Grammar · Autolexical syntax

1 Introduction

It has been pointed out by several authors (cf. Sadock 1991; Jackendoff 1997) that much of the linguistic research of the second half of the 20th century was carried out following goals, assumptions and methodological tools introduced in the late 1950s which have subsequently revealed themselves to be neither well-founded nor necessary. Some examples of those ideas that have guided a large part of linguistic studies are:

– Syntactocentrism: ‘*The fundamental generative component of the computational system is the syntactic component; the phonological and the semantic components are ‘interpretative*’’ (Jackendoff 1997, p. 15);

M. D. Jiménez López (✉)
Department de Filologies Romàniques
Research Group on Mathematical Linguistics
Rovira i Virgili University
Pl. Imperial Tàrraco, 1
43005 Tarragona, Spain.
e-mail: mariadolores.jimenez@urv.net

- Hierarchicality: *'The organizational dimensions of language are 'levels' obtainable from one another in a certain fixed order, depriving them of any genuine autonomy'* (Sadock 1991, p. 5);
- Derivationalism: *'The computational system takes representations of a given form and modifies them. That is, the computational system performs derivations, rather than, for example, imposing multiple simultaneous constraints'* (Jackendoff 1997, p. 12). *'Grammars make use of rules that apply to full representations of expressions and produce from them distinct representations'* (J.M. Sadock, unpublished);
- Substitution as the only operation: *'The fundamental operation of the computational system is the substitution of one string or structural complex for another in a phrase marker'* (Jackendoff 1997, p. 13);
- Non-redundancy: *'A working hypothesis in generative grammar has been that the language faculty is nonredundant, in that particular phenomena are not 'overdetermined' by principles of language'* (Jackendoff 1997, p. 14).

Many of the above assumptions have been abandoned by grammatical theories not directly linked to the generativist tradition such as GPSG (Gazdar, Klein, Pullum & Sag 1985), Montague grammar (Partee 1976), categorial grammar (Buszkowski, Marciszewski & van Benthen 1988), HPSG (Pollard and Sag 1994), LFG (Bresnan 2001), word grammar (Hudson 1984), role and reference grammar (Van Valin 1993), autolexical syntax (Sadock 1991), and Jackendoff's model (Jackendoff 1997). In this paper, we use autolexical syntax to show how grammar systems are suitable for natural language description.

By comparing autolexical syntax with grammar systems theory and by pointing out the great deal of similarity between these two theories, we suggest that grammar systems may offer a formal framework for natural language grammar. We introduce *Linguistic Grammar Systems* as a formal-language-theoretic grammatical model. We show that both autolexical syntax and grammar systems give good results in their respective disciplines thanks to the same concrete features, namely:

- Modularity. According to Harnish and Farmer (1984, p. 257), a system can be modular in at least two ways:

  1. A system is *externally modular* when it operates only on a specific domain of information, and has principles of operation that do not reach outside that system, even though useful information must be available there;
  2. a system is *internally modular* when it is analyzable into distinct, but interacting subsystems.

According to Chomsky (1984, p. 17), there is *internal modularity* whenever there are sub- systems with their own quite specific properties that interact in highly determined ways. Here, by modularity we mean the coexistence in a system of autonomous components. A system is thus considered to be modular if it is made up of several independent (but interacting) components each of which has its own alphabet, rules, etc.

- Distribution and cooperation. The terms distribution and cooperation will be used whenever a complex task is distributed among a set of 'modules or processors' that work together in a well defined way. According to Csuhaj-Varjú, Dassow, Kelemen and Păun (1994), in the early 1980s the study of *distributed systems of* *cooperating agents* in artificial intelligence was characterized by Davis (1980) as *'... concerned with those problems for which a single problem solver, single machine, or single locus of computation seems inappropriate'*. The preferred methodology was formulated as a *'turn to the use of multiple, distinct problem-solvers each embodied in its own system'*. An overall view of the present-day concepts of such types of systems is presented in Werner (1989). From his definition of *social goals* which are achievable by distributed and cooperating systems of agents, but are not achievable by any single agent, it follows that distributed and cooperative systems can be considered as closely related to the *complex systems* characterized by Simon (1982) as *'made up of a large number of parts that interact in a nonsimple way. In such systems the whole is more than the sum of the parts, not in an ultimate, metaphysical sense but in the important pragmatic sense that, given the properties of the parts and the laws of their interaction, it is not a trivial matter to infer the properties of the whole'*.

– Parallelism and interaction. In a system of autonomous components, there is parallelism if every component of the system works simultaneously, in parallel. We understand parallel models as opposed to serial, sequential or hierarchical models where different representations are seen as levels obtainable from one another in a certain fixed order. Representations of a high-level component are thus passed to a lower-level component that modifies them and passes the resulting representation on to a lower component, or what amounts to the same, the task of one component of the system must be over before the task of another component begins. In contrast, in a parallel and interactive model, components can perform their tasks without being constrained by a serial and hierarchical structure.

As will be shown in this paper, it is possible to establish an analogy between elements and function in autolexical syntax and in grammar systems. Taking this into account, the main thesis of this paper is: Autolexical syntax is considered to be a good grammatical theory; grammar systems and autolexical syntax present similar traits and a coincident way of functioning; thus, grammar systems may offer a formal-languages approach to linguistic issues. Note that our main goal is to show the applicability of grammar systems—a theory widely investigated from the formal point of view but whose possible applications have been scarcely considered—rather than going into the concrete linguistic content of the modules that make up the model proposed.

Formal language theory was conceived in 1950s as a tool for modelling and investigating the syntax of natural languages. After 1964 it rapidly became a theory for studying formal systems irrespectively of possible applications. Our approach is directly the opposite: we start with a new branch of formal language theory, namely grammar systems which was conceived as a purely theoretical model, and then show how it could be applied.

The paper is organized into seven sections. Section 2 offers some formal language prerequisites. Sections 3 and 4 introduce grammar systems and autolexical syntax, respectively. In Section 5, a comparison is made between those two theories. Section 6 presents Linguistic Grammar Systems as a formal-language-theoretic model for grammatical theory. Our conclusions are outlined in Section 7.

2 Formal language prerequisites

Throughout the paper, we assume that the reader is familiar with the basics of Formal Language Theory. In this section, we give some of the formal prerequisites that are needed in order to understand the formalization presented in this work. For further information on the theory of formal languages see Salomaa (1973) and Rozenberg and Salomaa (1997).

A *set* is a collection of elements taken from some pre-specified universe. A set is *finite* if it contains a finite number of elements, and otherwise it is *infinite*. A *sequence* of elements, from some universe, is a list of elements possibly, but not necessarily, with repetitions. Because the list is ordered by position, the elements are referred to as the first, second and ith.

A finite, nonempty set V of symbols or letters is called an *alphabet*. A *word* or a string over an alphabet V is a finite sequence of symbols from V. The *empty word* is denoted by λ and is the empty sequence of symbols. For an alphabet V, we denote by V^* the free monoid generated by V under the operation of concatenation, i.e. the set of all words over V. V^+ denotes the set of all nonempty strings over V. The *length* of a string $x \in V^*$ (the number of symbol occurrences in x) is denoted by $|x|$. The number of occurrences in $x \in V^*$ of symbols $U \subseteq V$ is denoted by $|x|_U$.

Given two words x and y over V, their *catenation* is xy. $\lambda x = x\lambda = x$, for all x in V^*.

For two words x and y over V, x is a *prefix* of y if $y = xz$, for some z in V^*. $x = y$ if x is a prefix of y and $|x| = |y|$. x is a *proper prefix* of y if it is a prefix, $x \neq y$, and $y \neq \lambda$. A *suffix* and *proper suffix* are defined similarly. x is said to be a *subword* of y if $y = wxz$, for some w and z in V^*. x is a *proper subword* if $x \neq y$ and $x \neq \lambda$.

Let Σ and Δ be alphabets. Then a mapping $h : \Sigma^* \rightarrow \Delta^*$ is said to be a *morphism* if $h(\lambda) = \lambda$ and $h(a_1 \ldots a_n) = h(a_1)h(a_2)\ldots h(a_n)$, for all $a_1 \ldots a_n$ in Σ. A morphism is also known as a *homomorphism*.

A natural generalization of the replacement of symbols by words as found in morphisms is the replacement of symbols by sets of words, i.e. *substitution*. Let $\mathcal{L}$ be a family of languages and Σ and Δ be two alphabets. Then a mapping $\sigma \colon \Sigma^* \rightarrow 2^{\Delta^*}$ is said to be a *substitution* if $\sigma(\lambda) = \{\lambda\}$ and $\sigma(a_1 \ldots a_n) = \sigma(a_1)\ldots\sigma(a_n)$, for all $a_1 \ldots a_n$ in Σ. If additionally, for all a in Σ, $\sigma(a)$ is in $\mathcal{L}$, then we say that σ is an $\mathcal{L}$ *substitution*.

Given an alphabet V, a *language* L over V is a subset of V^*. Since languages are sets, the Boolean operations of *union*, *intersection* and *complement* are applicable and defined in the usual way. Given two languages L_1 and L_2, possibly over different alphabets, the *catenation* of L_1 and L_2 is denoted L_1L_2 and equals the set $\{x_1x_2 \mid x_1 \text{ in } L_1 \text{ and } x_2 \text{ in } L_2\}$.

A *Chomsky grammar* is a quadruple $G = (N, T, P, S)$ where N is the nonterminal alphabet, T is the terminal alphabet, $S \in N$ is the axiom (start symbol) and P is the set of rewriting rules (or productions), written as $x \rightarrow y$. A grammar is *context-sensitive* when its rules are of the form $x_1Ax_2 \rightarrow x_1wx_2, x_1, x_2$, where w is strings over $V_G, A \in N, w \neq \lambda$. A grammar is *context-free* when all its rules are of the form $A \rightarrow x$, where $A \in N, x \in V_G^*$. A grammar is called *regular* when it contains only rules of the form $A \rightarrow a$, $A \rightarrow aB$, $a \in T$, $A, B \in N$.

A direct *derivation* in G is denoted by $\Longrightarrow$ (by $\Longrightarrow_G$ when we need this information). The transitive (reflexive) closure of the relation $\Longrightarrow$ is denoted by $\Longrightarrow^+$ ($\Longrightarrow^*$).

The language generated by G, denoted by $L(G)$, is $L(G) = \{w \mid S \Longrightarrow_G^* w, w \in T^*\}$.

We denote by RE, CS, CF, LIN, REG the classes of unrestricted, context-sensitive, context-free, linear and regular grammars, respectively.

A rewriting system $G = (V, w, P)$ where V is an alphabet, $w \in V^+$ is an axiom and P is a finite set of rewriting rules $A \to v$, with $A \in V$, $v \in V^*$ is called a *pure context-free grammar*. Rules of G are applied in derivations as follows: for two strings $x, y \in V^*$, we say that x directly derives y in G, written as $x \Longrightarrow y$ iff $x = x_1 a x_2$, $y = x_1 v x_2$, $x_1, x_2 \in V^*$, and $a \to v \in P$. The generated language is defined by $L(G) = \{x \in V^* \mid w \Longrightarrow^* x\}$.

3 Grammar systems

Grammar systems theory is a branch of the field of formal languages that provides syntactic models for describing multi-agent systems at a symbolic level using tools from formal grammars and languages. The theory was launched in 1988 (Csuhaj-Varjú and Dassow 1990), when Cooperating Distributed Grammar Systems were proposed as a syntactic model of the blackboard architecture of problem solving (cf. Nii 1989). One year later, Parallel Communicating Grammar Systems—very much inspired by the 'classroom model' of problem solving—were introduced as a grammatical model of parallelism (Păun and Sântean 1989). Since 1988, the theory has developed in several directions related to several scientific areas. Distributed and decentralized artificial intelligence (Durfee, Lesser & Corkill 1989; Bond and Gasser 1988; Demazeau & Muller 1990), artificial life (Langton 1989; Păun 1995a), molecular computing (Păun 1998), robotics, natural language processing, ecology, sociology, etc. have suggested some modifications to the basic model and have given rise to different variants and subfields of the theory (cf. Dassow, Păun & Rozenberg 1997).

Easy generation of noncontext-free structures using context-free rules, modularity, parallelism, interaction, distribution, and cooperation are just some of the advantages that grammar systems have over classical models, and mean that this theory could be applied to several fields.

But, what is a grammar system? Roughly speaking, a grammar system is a set of grammars working together, according to a specified protocol, to generate a language. Note that while in classical formal language theory *one* grammar (or automaton) works individually to generate (or recognize) *one* language, here we have *several* grammars working together in order to produce *one* language.

There are two basic classes of grammar systems:

1. *Cooperating Distributed Grammar Systems* (CDGS) which work in a sequential way.
2. *Parallel Communicating Grammar Systems* (PCGS) which function in parallel.

A CDGS consists of a finite set of generative grammars that cooperate in the derivation of a common language. Component grammars generate the string in turns (thus, sequentially), under some cooperation protocol. This system works as follows. Initially, the axiom is the common sentential form. At each moment in time, one grammar (and only one) is active, that is, it rewrites the common string, while the rest of the grammars in the system are inactive. Conditions under which a component can start/stop its activity on the common sentential form are specified by a cooperation protocol. Terminal strings generated in this way form the language of the system. The basic model of CDGS presents sequentiality in its work and homogeneity in the cooperation protocol; however, variants have been introduced that have some parallelism in their function (teams) and that change the initial homogeneity into heterogeneity of modes of cooperation (hybrid systems). The basic model has been extended, also, by the addition of 'extra' control mechanisms. For information about all these variants see Csuhaj-Varjú, Dassow, Kelemen & Păun (1994); Dassow, Păun & Rozenberg (1997). Formally:

Definition 1 A CDGS of degree n, $n \geq 1$, is a construct:

$$\Gamma = (N, T, S, G_1, \ldots, G_n),$$

where:

- N, T are disjoint alphabets;
- $S \in N$ is the axiom;
- $G_i = (N, T, P_i)$, $1 \leq i \leq n$, the so-called *components* of the system Γ, are usual Chomsky grammars without axiom where:
  - N is the nonterminal alphabet;
  - T is the terminal alphabet;
  - P_i is a finite set of rewriting rules over $N \cup T$.

Definition 2 Let Γ be a CDGS as defined above. Let $x, y \in (N \cup T)^*$. Then, we write $x \Longrightarrow_{G_i}^{k} y$, for $1 \leq i \leq n$, iff there are words $x_1, x_2, \ldots, x_{k+1}$ such that:

(i) $x = x_1$, $y = x_{k+1}$;
(ii) $x_j \Longrightarrow_{G_i} x_{j+1}$, i.e. $x_j = x'_j A_j x''_j$, $x_{j+1} = x'_j w_j x''_j$, $A_j \rightarrow w_j \in P_i$, $1 \leq j \leq k$.

Moreover, we write:

$x \Longrightarrow_{G_i}^{\leq k} y$ iff $x \Longrightarrow_{G_i}^{k'} y$, for some $k' \leq k$;
$x \Longrightarrow_{G_i}^{\geq k} y$ iff $x \Longrightarrow_{G_i}^{k'} y$, for some $k' \geq k$;
$x \Longrightarrow_{G_i}^{*} y$ iff $x \Longrightarrow_{G_i}^{k} y$, for some k;
$x \Longrightarrow_{G_i}^{t} y$ iff $x \Longrightarrow_{G_i}^{*} y$, and there is no $z \neq y$ with $y \Longrightarrow_{G_i}^{*} z$.

Any derivation $x \Longrightarrow_{G_i}^{k} y$ corresponds to k direct derivation steps in succession by grammar G_i. $\leq k$-derivation mode corresponds to a time limitation, since the agent can perform *at most* k derivation steps. $\geq k$-derivation mode represents competence, since it requires the agent to perform *at least* k derivation steps. $*$-mode denotes an arbitrary derivation: the agent *can* work on the sentential string as long as it wants to. And finally, t-mode stands for a terminal derivation, where the agent *must* perform derivation steps for as long as it can.

Definition 3 Let Γ be a CDGS, and denote $D = \{*, t\} \cup \{k, \leq k, \geq k \mid k \geq 1\}$. The language generated by the system Γ in the derivation mode $f \in D$ is:

$$L_f(\Gamma) = \{w \in T^* \mid S \Longrightarrow_{P_{i_1}}^{f} w_1 \Longrightarrow_{P_{i_2}}^{f} \ldots \Longrightarrow_{P_{i_m}}^{f} w_m = w, m \geq 1, 1 \leq i_j \leq n, 1 \leq j \leq m\}.$$

For formal results on this type of grammar systems, see Csuhaj-Varjú, Dassow, Kelemen & Păun (1994, pp. 36–72); Dassow, Păun & Rozenberg (1997, p. 162).

A PCGS consists of several usual grammars, each with its own sentential form. In each time unit (a common clock divides the time into units in a uniform way for all the components) each component uses a rule which rewrites the associated sentential form. Cooperation among agents takes place thanks to the so-called query symbols that permit communication among components. When a component introduces a query symbol for another component, the latter sends its current sentential form to the former, which replaces the query symbol with the string received. One component is identified as the *master*, and the language generated by it (with or without communication) is the language of the system. Formally:

Definition 4 A PCGS of degree n, $n \geq 1$, is a construct:

$$\Gamma = (N, K, T, G_1, \ldots, G_n),$$

where:

- N, T, K are mutually disjoint alphabets;
- $K = \{Q_1, Q_2, \ldots, Q_n\}$ are called query symbols and they are associated in a one-to-one manner to components $G_1, \ldots, G_n$;
- $V_\Gamma = N \cup K \cup T$;
- $G_i = (N \cup K, T, P_i, S_i)$, $1 \leq i \leq n$, the so-called *components* of the system, are usual Chomsky grammars where:
  - N is the nonterminal alphabet;
  - $K = \{Q_1, Q_2, \ldots, Q_n\}$ is the set of query symbols;
  - T is the terminal alphabet;
  - P_i is a finite set of rewriting rules over $N \cup K \cup T$, for $1 \leq i \leq n$;
  - $S_i \in N$ is the axiom, for $1 \leq i \leq n$.

Definition 5 Given a PCGS $\Gamma = (N, K, T, G_1, \ldots, G_n)$ as above, for two n-tuples $(x_1, x_2, \ldots, x_n)$, $(y_1, y_2, \ldots, y_n)$, $x_i, y_i \in V_\Gamma^*$, $1 \leq i \leq n$, we write $(x_1, \ldots, x_n) \Longrightarrow (y_1, \ldots, y_n)$ if one of the next two cases holds:

(i) $|x_i|_K = 0$, $1 \leq i \leq n$, and for each i, $1 \leq i \leq n$, we have $x_i \Longrightarrow y_i$ in grammar G_i, or $x_i \in T^*$ and $x_i = y_i$;

(ii) there is i, $1 \leq i \leq n$, such that $|x_i|_K > 0$; then, for each such i, we write $x_i = z_1 Q_{i_1} z_2 Q_{i_2} \ldots z_t Q_{i_t} z_{t+1}$, $t \geq 1$, for $z_j \in V_\Gamma^*$, $|z_j|_K = 0$, $1 \leq j \leq t+1$; if $|x_{i_j}|_K = 0$, $1 \leq j \leq t$, then $y_i = z_1 x_{i_1} z_2 x_{i_2} \ldots z_t x_{i_t} z_{t+1}$ [and $y_{i_j} = S_{i_j}$, $1 \leq j \leq t$]; when, for some j, $1 \leq j \leq t$, $|x_{i_j}|_K \neq 0$, then $y_i = x_i$; for all i, $1 \leq i \leq n$, for which y_i is not specified above, we have $y_i = x_i$.

Note that rules $Q_i \rightarrow x$, $1 \leq i \leq n$, are never used, hence we shall assume that such rules do not appear.

We have denoted in the same way, by $\Longrightarrow$, both the component-wise derivation steps and the communication steps. As usual, by $\Longrightarrow^*$ we denote the reflexive transitive closure of this relation, that is, the sequences of intercalated derivations and communication steps.

The work of a PCGS is blocked when:

1. a component x_i of the current n-tuple $(x_1, \ldots, x_n)$ is not terminal with respect to G_i, but no rule of G_i can be applied to x_i (this can happen both due to rules in P_i and to communication);

2. a *circular query* appears: if G_{i_1} introduces Q_{i_2}, G_{i_2} introduces Q_{i_3} and so on, until $G_{i_{k-1}}$, which introduces Q_{i_k}, and G_{i_k}, which introduces Q_{i_1}, then no derivation is possible (communication has priority), but no communication (in this cycle) is possible (only strings without query symbols occurrences are communicated).

Definition 6 The language generated by a PCGS Γ as above is:

$$L(\Gamma) = \{x \in T^* \mid (S_1, S_2, \ldots, S_n) \Longrightarrow^* (x, \alpha_2, \ldots, \alpha_n), \alpha_i \in V_\Gamma^*, 2 \leq i \leq n\}.$$

So, we start from the n-tuple of axioms $(S_1, \ldots, S_n)$ and proceed by repeated rewriting and communication steps, until component G_1 produces a terminal string. Note that in $L(\Gamma)$ we retain the strings generated in this way on the first component, without caring about strings generated by $G_2, \ldots, G_n$. Component G_1 is called the *master* of the system.

The PCGSs we have presented in the above definitions are systems without any restrictions on the use of query symbols, that is, each component G_i is allowed to introduce a symbol Q_j (obviously, when G_i introduces the symbol Q_i, the derivation is blocked by query circularity). Now,

Definition 7 Let $\Gamma = (N, K, T, G_1, \ldots, G_n)$ be a PCGS. If only G_1 is allowed to introduce query symbols (formally, $P_i \subseteq (N \cup T)^* \times (N \cup T)^*$, for $2 \leq i \leq n$), then we say that Γ is a *centralized* PCGS; in contrast, the unrestricted case is called *non-centralized*.

Definition 8 A PCGS $\Gamma = (N, K, T, G_1, \ldots, G_n)$ is said to be *returning* (to axiom) if, after communicating, each component whose string has been sent to another component returns to axiom. A PCGS is *non-returning* if in point (ii) of Definition 5, the words in brackets, $[y_{i_j} = S_{i_j}, 1 \leq j \leq t]$, are erased. That is, after communicating, the grammar G_{i_j} does not return to S_{i_j}, its axiom, but continues processing the current string.

In PCGS, we also find variants of the basic model. Modifications in the type of components or in the way strings are communicated produce several new types of PCGS (cf. Dassow, Păun & Rozenberg 1997). Since they will be used in the definition of Linguistic Grammar Systems, we refer here to two variants of PCGS: the so-called *PCGS with communication by command* and *PCGS with renaming*. For formal results on PCGS see Csuhaj-Varjú, Dassow, Kelemen & Păun (1994, pp. 158–176); Dassow, Păun & Rozenberg (1997, pp. 180–186).

PCGSs with communication by command (CCPC) were introduced in Csuhaj-Varjú, Kelemen & Păun (1996). This type of PCGS was inspired by the data flow of WAVE-like architectures of highly parallel processors, by connection machines, by Boltzmann machines and by other parallel devices. In contrast to the PCGS presented above (where communication is done by request) in this new variant of PCGS communication is done by command. This means that when a string derived in a component of the system matches the pattern associated with another component, it is sent to this latter component. So, we do not need a query symbol in the sentential form of a component to get a string of another component. In this case, each component sends its sentential form whenever its string matches the language selector of another component of the system. A CCPC works in very much the same way as a usual PCGS functions. We start with an initial configuration where each component has its own start symbol, and proceed with a sequence of alternating rewriting and communication steps. As already mentioned, a sentential form can be communicated only if it matches the language selector associated with the addressee component. After communication, the component can either return to its axiom or continue with its current string. As with the usual PCGS, communication has priority over rewriting. And, as expected, the language of the system is the language generated by the component designated as the *master*. Formally:

Definition 9 A PCGS with communication by command (CCPC) is a construct:

$$\Gamma = (N, T, (S_1, P_1, R_1), \ldots, (S_n, P_n, R_n)), \quad n \geq 1,$$

where:

- N and T are disjoint sets;
- N is the nonterminal alphabet;
- T is the terminal alphabet;
- (S_i, P_i, R_i), $1 \leq i \leq n$, are the components of the system, where:
  - $S_i \in N$ is the axiom;
  - P_i is the set of rewriting rules over $N \cup T$;
  - $R_i \subseteq (N \cup T)^*$ is the selector language of the component i (R_i will be either specified as a regular set or by a pattern[1]).

Let us now look at how a CCPC functions. We start from an initial configuration $(S_1, \ldots, S_n)$. At any step, the state of the process will be described by an n-tuple $(x_1, \ldots, x_n)$ of strings over $N \cup T$. The system will modify the current configuration either:

1. By **rewriting** steps. A rewriting step will be performed component-wise. There are two main possibilities:
   (a) Each component has to use a rule, rewriting its current sentential form, except those components whose string is terminal which are allowed to do nothing. This can be considered as a *minimal* derivation strategy: one rewrites exactly until a target language is matched, or
   (b) Each component has to perform a *maximal* derivation step, i.e. the rewriting step is accomplished when no further rewriting is possible on the sentential form of that component.

In both cases, at the end of the step, one checks whether or not a communication step can be performed. That is, one checks if currently generated strings match

[1] Having an alphabet A of *constants* and an alphabet V of *variables*, a **pattern** is a string over $A \cup V$. For a pattern π over A and V, the *language* associated with π, denoted by $L_A(\pi)$, consists of all the strings obtained from π by consistently replacing the variable occurrences with nonempty strings over A ('consistently' means that all the occurrences of the same variable are replaced by the same string).

For example, if $A = \{a, b\}$ and $V = \{X_1, X_2\}$, then $\pi = aaX_1X_1bX_2bb$ is a pattern. It defines the set of all the strings over A starting with two occurrences of a, continuing with a replication of any string over A, one occurrence of b, any string over A, and ending with a double occurrence of b. Therefore, $x = aa(abb)(abb)b(bab)bb$ has the form specified by this pattern, while $y = aaabbababbabbb$ does not. So, we have $L_A(\pi) = \{aawwbzbb \mid w, z \in \{a, b\}^+\}$.).

selector languages R_i. In the second case, a communication step *must* be performed, otherwise the system cannot perform another derivation step. In the first case, if no communication is to be performed, another rewriting step can be done.

In between those two cases we may also consider different derivation modes as k, $\leq k$, $\geq k$ steps, like in CDGS.

2. Or by **communication** steps. In order to define a communication step we have to specify rules for:

   – Defining the string to be communicated. This problem has at least two solutions:

     (a) Communication *without* splitting. In this case only complete current sentential forms are considered as messages. If $x_i \in R_j$, then x_i as a whole is transmitted to the component j;
     (b) Communication *with* splitting. In this case, for every decomposition $x_i = x_{i,1}x_{i,2}\ldots x_{i,k}$, $k \geq 1$, such that $x_{i,j} \in R_{s_j}$, for some $1 \leq s_j \leq n$, $1 \leq j \leq k$, component i will produce the messages $x_{i,1}, x_{i,2}, \ldots, x_{i,k}$ which will be transmitted to the matching components $s_1, s_2, \ldots, s_k$.

   – Solving the conflicts at target components. This problem can have several solutions. Basically, the received messages can be:

     (a) *Adjoined* to the current string of the addressee, or
     (b) They can *replace* the string of the addressee.

     In both cases we can consider that:

     (a) *Only one* of the received messages is taken into account. Once again we have two possibilities:
        (i) Either we choose *nondeterministically* one message, or
        (ii) We consider a *priority* relation, for instance, defined by the natural ordering of the component sending messages.
     (b) Or *all* the received messages concatenated in a specified order.

   – Defining the next string for components that have transmitted messages. As in the case of normal PCGS, a component that has submitted its string to another component, without receiving new strings, can:
     (a) Either *return* to its axiom, or
     (b) *Continue* to process its current string.

Definition 10 Let $\Gamma = (N, T, (S_1, P_1, R_1), \ldots, (S_n, P_n, R_n)), n \geq 1$ be a CCPC working with *maximal* derivations as rewriting steps, communicating *without splitting* the strings, *replacing* the strings of the target component by a *concatenation* of the received messages, in the order of the system components, and *returning* to axioms after communication. For such system we define a **rewriting step** by $(x_1, \ldots, x_n) \Longrightarrow (y_1, \ldots, y_n)$ iff:

$x_i \Longrightarrow^* y_i$ in P_i and there is no $z_i \in (N \cup T)^*$ such that
$y_i \Longrightarrow z_i$ in P_i (if $x_i \in T^*$, then $y_i = x_i$, otherwise $x_i \Longrightarrow^+ y_i$).

A **communication step** denoted by $(x_1, \ldots, x_n) \vdash (y_1, \ldots, y_n)$ is defined as follows. Let:

$$\delta_i(x_i,j) = \begin{cases} \lambda, & \text{if } x_i \notin R_j \quad \text{or } i = j, \\ x_i, & \text{if } x_i \in R_j \quad \text{and } i \neq j \end{cases}$$

for $1 \leq i,j \leq n$,

$$\Delta(j) = \delta(x_1,j)\delta(x_2,j)\ldots\delta(x_n,j)$$

for $1 \leq j \leq n$ (this is the 'total message' to be received by the jth component), and:

$$\delta(i) = \delta(x_i,1)\delta(x_i,2)\ldots\delta(x_i,n)$$

for $1 \leq i \leq n$ (this is the 'total message' sent by the ith component, a power of x_i indicating to how many targets the ith component sends a message). Then for $1 \leq i \leq n$, we define:

$$y_i = \begin{cases} \Delta(i), & \text{if } \Delta_i \neq \lambda, \\ x_i, & \text{if } \Delta_i = \lambda \quad \text{and } \delta(i) = \lambda, \\ S_i & \text{if } \Delta(i) = \lambda \quad \text{and } \delta(i) \neq \lambda. \end{cases}$$

In words, y_i is either the concatenation of the received messages, if any exists, or it is the previous string, when this component is not involved in communications, or it is equal to S_i, if this component sends messages but it does not receive them. Note that a component cannot send messages to itself.

The language thus generated by the CCPC can be defined in the two following ways:

1. As the set of all the terminal strings generated by a designated component of the system, its *master* (as in usual PCGS where the first component is designated, by convention, as being the master),
2. Or as the set of all the terminal strings generated by any component of the system.

In the formalization presented here, we consider the first case, that is, when the language of the system is the one generated by the first component (the *master*).

Definition 11 Let $\Gamma = (N, T, (S_1, P_1, R_1), \ldots, (S_n, P_n, R_n))$, $n \geq 1$ be a CCPC as above; then the language generated by Γ is defined as follows:

$L(\Gamma) = \{w \in T^* \mid (S_1,\ldots,S_n) \Longrightarrow (x_1^{(1)},\ldots,x_n^{(1)}) \vdash (y_1^{(1)},\ldots,y_n^{(1)}) \Longrightarrow (x_1^{(2)},\ldots,x_n^{(2)}) \vdash (y_1^{(2)},\ldots,y_n^{(2)}) \Longrightarrow \ldots \Longrightarrow (x_1^{(s)},\ldots,x_n^{(s)}),$ for some $s \geq 1$ such that $w = x_1^{(s)}\}$

For formal results on CCPC see Dassow, Păun & Rozenberg (1997, p. 192).

PCGS with renaming appeared in 1996 and was aimed at solving some problems in the generation of specific noncontext-free structures present in natural languages (Păun 1996a). Consider the following three languages:

$$L_1 = \{xx \mid x \in \{a,b\}^*\},$$

$$L_2 = \{a^n b^n c^n \mid n \geq 1\},$$

$$L_3 = \{a^n b^m c^n d^m \mid n \geq 1\}.$$

All three can be generated using PCGS with context-free components; L_1 and L_2 can be generated even with right-linear systems; and a small number of components suffices for their generation (for formal details cf. Csuhaj-Varjú, Dassow, Kelemen & Păun 1994; Păun 1995b and Chiţu 1997).

If we now compare:

$$L_3 = \{a^n b^m c^n d^m \mid n, m \geq 1\},$$
$$L_3' = \{a^n b^m a^n b^m \mid n, m \geq 1\},$$

we will see that L_3 is difficult to generate (PCGSs that generate it are complex and have an intricate work) whereas L_3' is easy to generate (cf. Păun 1995b). Clearly, the difficulty of L_3 rests on the correlation of a^n with c^n and b^m with d^m where $a \neq c$ and $b \neq d$ while in L_3' letters that are correlated (a with a, and b with b) are the same. This fact suggested the variant of PCGS with renaming. Formally:

Definition 12 A PCGS with renaming is a construct:

$$\Gamma = (N, K, T, G_1, \ldots, G_n, h_1, \ldots, h_m),$$

where:

- N, T, K are mutually disjoint alphabets;
- $K = \{Q_1, Q_2, \ldots, Q_n\}$ are called query symbols and they are associated in a one-to-one manner to the components $G_1, \ldots, G_n$;
- $V_\Gamma = N \cup K \cup T$;
- $G_i = (N \cup K, T, S_i, P_i)$, $1 \leq i \leq n$, the so-called *components* of the system, are usual Chomsky grammars where:
  - N is the non-terminal alphabet;
  - $K = \{Q_1, Q_2, \ldots, Q_n\}$ is the set of query symbols;
  - T is the terminal alphabet;
  - $S_i \in N$ is the axiom, for $1 \leq i \leq n$;
  - P_i is a finite set of rewriting rules over $N \cup T \cup K \cup K'$, for $1 \leq i \leq n$, where $K' = \{[h_j, Q_i] \mid 1 \leq i \leq n, 1 \leq j \leq m\}$ and each $[h_j, Q_i]$ is a symbol.
- $h_j : (N \cup T)^* \rightarrow (N \cup T)^*$, $1 \leq j \leq m$, are weak codes such that:

  (i) $h_j(A) = A$, for $A \in N$;
  (ii) $h_j(a) \in T \cup \{\lambda\}$, for $a \in T$.

A PCGS with renaming works in the same way as a usual PCGS with only one difference: whenever a $[h_j, Q_i]$ symbol appears in the sentential form of any of the components of the system, it must be replaced not by w_i, but by $h_j(w_i)$. The language generated by a PCGS with renaming is, as in the case of the usual PCGS defined above, the language of the 'master'.

4 Autolexical syntax

Autolexical syntax (AS), which was conceived in 1985 in the United States within the field of linguistics, represents a radical departure from the approaches used to model grammar dominant in 1980s. It is defined as a variety of nontransformational generative grammar in which autonomous systems of rules characterize the various dimensions of the linguistic representation. In this section, we give a brief outline of this theory in order to compare it with grammar systems and thus highlight the similarities between these two frameworks that were conceived of at the same time but in two different research fields and in very distant places. For more information about autolexical syntax, see Sadock (1985), Sadock (1991) and Schiller, Steinberg & Need (1995).

Conceived as a theory of parallel grammatical representations, the crucial assumption of AS is that components are independent mini-grammars each containing information relating to a single aspect of grammar, and that they are related to one another only by an interface system. This means that fully autonomous systems of rules characterize the various dimensions of linguistic representation. Each component is a self-contained system, with its own independent set of rules, principles and basic vocabulary. The number of components of such a construct is a matter of debate, but, as shown in Fig. 1, most studies have considered at least three modules:

– *Syntax*, which specifies the phrasal constituent structures allowed by the language;
– *Semantics*, which provides representations of natural language expressions in terms of logical relations;
– *Morphology*, which makes explicit the structure of all and only the grammatical word-forms in the language.

Unlike what is assumed in other linguistic theories, modules are not hierarchically related to one another. A module does not have to wait for the output of another to carry out its task, but has the power to generate an infinite set of representations quite independently of what is going on in any of the other components.

In order for an expression to count as fully well-formed, it must satisfy the independent requirements of each of the modules, so that we can say that each module acts as a filter on all the others. An expression that is syntactically well-formed may thus not qualify as a sentence either because it does not have a well-formed semantic parsing, or because there is no morphologically correct clustering of morphemes to it, or for both reasons. However, when an expression is well-formed with respect to each

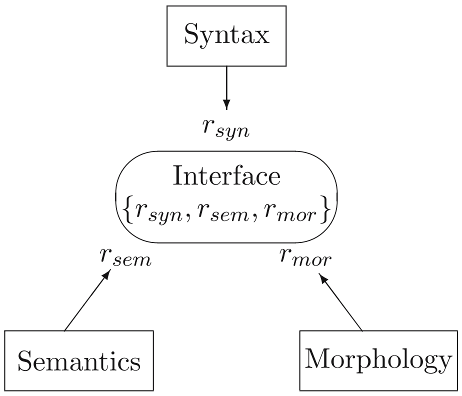


Fig. 1 Autolexical syntax

dimension, it may still not be qualified as grammatical, if it does not pass the principles relating representations contained in the fourth element shown in Fig. 1: *Interface*. It can be considered as an *overarching control center* which checks whether the members of a set of parsings from the individual components fit one another and count as parsings of the same total expression. It thus coordinates the various representations produced by the autonomous modules in order to form a grammatical expression of language. It contains three basic elements:

1. *Lexicon*, which contains the basic vocabulary for each of the modules, as well as information on the structural properties of each lexical item with respect to the various autonomous components.
2. *Paradigmatic Constraints*, which capture important correspondences operating across modules. A variety of intermodular correspondence is the existence of some interpredictability between semantics and syntax, morphology and syntax, and semantics and morphology.
3. *Syntagmatic Constraints*, which constrain mismatches between simultaneous positions of a lexeme in representations from two modules. The degree of allowable mismatch between the simultaneous positions of a lexeme in representations from two modules is by no means unlimited. There are very powerful principles that operate to constrain such mismatches (cf. Sadock 1991, p. 43).

The *output* of an autolexical grammar is a set of triples like the following: $\{r_{\mathrm{syn}}, r_{\mathrm{sem}}, r_{\mathrm{mor}}\}$. Each triple corresponds to an expression of language to which grammar ascribes syntactic, semantic, and morphological parsings. In order for a set of triples to count as the total representation of an expression of language, it must satisfy the following requirements:

1. each of the elements of the triple must be an adequate output of the corresponding module;
2. the triple must respect the paradigmatic and syntagmatic constraints of the interface; and
3. each component's representation must be fully and correctly lexicalizable by means of the same set of lexical items.

As for their *generative power*, the class of languages describable by autolexical grammars that make use of strictly context-free components is beyond the set describable in terms of individual single context-free phrase structure grammars.

Autolexical syntax has been particularly successful in description of cliticization and incorporation (cf. Sadock 1985, 1991). Both phenomena are analyzed as allowable mismatches between syntax and morphology. But, those are not the only phenomena tackled from the autolexical point of view. There is also an interesting attempt to give an autolexical explanation to some more familiar syntactic phenomena which in other linguistic theories have so far been treated in terms of movement and deletion.

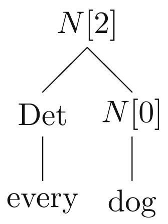
Before ending this section, let us considered some examples from Sadock (1991, pp. 29–38) in order to illustrate how AS functions.

Let us consider the English lexeme *dog* whose lexical content is the following:

– *dog*:

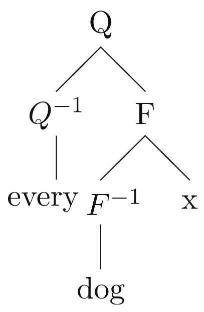
$$\begin{aligned}&\text{syntax} = N[0],\\ &\text{semantics} = F^{-1},\\ &\text{morphology} = N[-0].\end{aligned}$$

In words, *dog* is a noun in the syntax, a noun stem in the morphology, and an intransitive predicate in the semantics. The syntactic statement in the lexical entry '*dog*' indicates that it may be found in structures like:

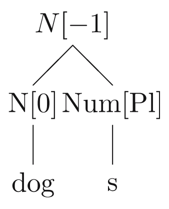


because of the existence of rule $N[2] \longrightarrow Det\ N[0]$ in English syntax.

The semantic statement of *dog*'s entry sanctions—on the basis of semantic rules such as (1) $F = F^{-1}(x)$ and (2) $Q = Q^{-1}(F)$- semantic structures as the following one:



And finally, since the English morphological module contains the rule $X^{-1} \longrightarrow X^{-0}, Y$, the morphological part of *dog's* lexical content allows it to be found in structures such as:

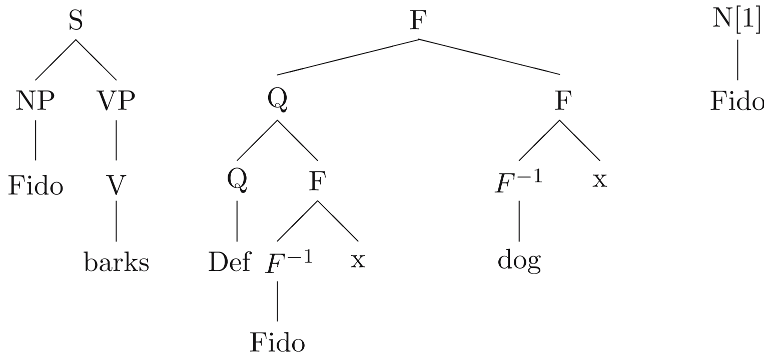


If we take now a proper noun such as '*Fido*' we would have a different representation in the lexicon:

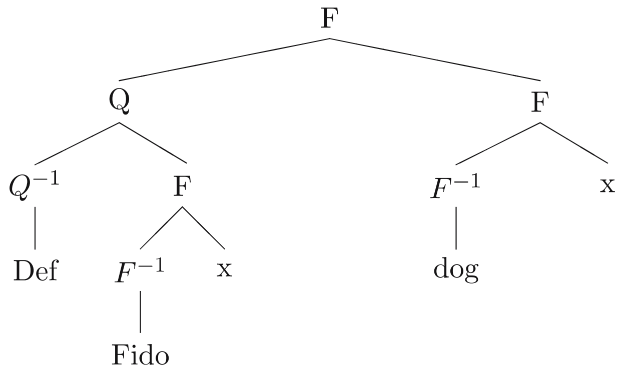
– *Fido*:

$$\begin{aligned}&\text{syntax} = N[2],\\ &\text{semantics} = [_Q[_{Q[-1]}DEF]\ [_F[_{F[-1]} - - - -]x]],\\ &\text{morphology} = N[-1].\end{aligned}$$

The above lexical entry allows *Fido* to be found in structures like the following ones (syntactic, semantic, and morphological, respectively):

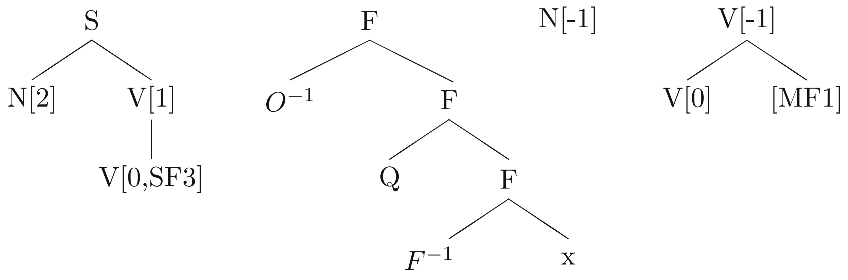


After having seen how a lexeme is represented in the lexicon of an autolexical grammar, let us now illustrate the filtering effect among modules. A structure like **Fido dog* is not an English sentence. However, from a semantic point of view, this expression presents no problem and has the following semantic representation:



The above expression predicates the dogginess of Fido and it is quite comprehensible, though ungrammatical. What is wrong with it is that English contains no syntactic rule that expands *S* as *NP* + *Pred*. In this case, therefore, it is the syntactic module of English that filters out a semantically unproblematic construction. So, as already mentioned, for an expression to be acceptable it must receive a correct analysis from each of the modules. This means that a syntactic well-formed structure will not be considered as a language expression if it does not receive an acceptable parsing in the semantic module or vice versa.

An output of AS is a set of triples. In any triple, each element must be an output of the appropriate component, but in addition the separate elements must pass the congruity requirements imposed by the interface. One such requirement is that if we want the triple to count as an expression of the language, each component's representation in a triple must be fully and correctly lexicalizable by means of the same set of lexical items. In order to illustrate this idea, let us consider the following three representations that could be considered as good outputs of the three modules that make up our autolexical grammar. Now if we consider the following three lexemes:



where (MF1): $X^{-1} \longrightarrow X^{-0}, Y$

– *Fido*:

$$\begin{aligned}&\text{syntax} = N[2]\\&\text{semantics} = [_Q[_{Q[-1]}DEF]\ [_F[_{F[-1]} - - - -]x]]\\&\text{morphology} = N[-1]\end{aligned}$$

– *bark*:

$$\begin{aligned}&\text{syntax} = V[0, SF3]\\&\text{semantics} = F^{-1}\\&\text{morphology} = V[-0]\end{aligned}$$

– *-s*:

$$\begin{aligned}&\text{syntax} = \text{nil}\\&\text{semantics} = O^{-1}\\&\text{morphology} = [_{V[-1]}V[-0] - - - -]\end{aligned}$$

it is easy to see that they can lexicalize all three trees above, thus allowing such a triple to be the representation of the English sentence *Fido barks.*

5 Autolexical syntax and grammar systems

The two theories we have presented in the above sections appeared more or less in the same years in two different research fields. *Grammar systems theory* appeared in Europe in 1988 within the field of formal language theory, and *autolexical syntax* was conceived in the United States in 1985 within the area of linguistics. Both theories have interesting resemblances that deserve to be pointed out, particularly as we are interested in showing the possible adequacy of grammar systems theory to account for linguistic issues. If we are able to show that grammar systems have clear similarities with autolexical syntax, and if we take into account the good results that the latter has reached in the field of linguistics, we could say that the former (which has similar traits) could also be suitable for describing the grammar of natural languages.

If seen as a theory of parallel grammatical representations where different modules work independently and collaborate in order to generate a well-formed linguistic structure, AS has several ideas in common with grammar systems theory. In order to see the analogies between AS and grammar systems, we compare this linguistic theory with one of the variants of grammar systems, namely *Parallel Communicating Grammar Systems*. In order to establish the analogy, we recall that in a PCGS there are several grammars working independently and in parallel, with their own axioms, alphabets and rules, and coordinated by a distinct element, *the master*. In this variant of grammar systems, the language of the master, which is generated with or without communication, can be considered the language of the system.

If we recall what was said in the two previous sections, we can easily observe the following interesting similarities between AS and PCGS:

1. *Modules* in AS clearly correspond to *component grammars* in a PCGS.
2. In AS every module has its *own primitives and rules*; in PCGS, every grammar has its *own alphabet and rules*.
3. The *interface* in AS corresponds to the *master* in PCGS. Both are coordinating elements: in AS, representations given by each module are brought together to form a grammatical expression of language through the interface; in PCGS it is the master who, with the help of the rest of the grammars, produces the language of the system.
4. In AS, every module works *independently and in parallel;* in PCGS, each grammar works, in a *parallel way*, in its own string irrespectively of what is happening in the rest of grammars.
5. In both cases, the output is provided by the *coordinating element:* in AS, the output is a set of triples in which each element must be an output of the appropriate component and has to pass some requirements imposed by the interface. The output is thus *given by the interface* which puts in correspondence the work of modules ($\{r_{\text{syn}}, r_{\text{sem}}, r_{\text{mor}}\}$); in PCGS, the language of the system is the one *generated by the master*, which has used strings generated by other grammars in order to produce the output of the system ($L(\Gamma)$).

Table 1 Autolexical syntax and grammar systems: similarities

Autolexical syntax	Grammar systems
Modules	Component grammars
Each module = own primitives	Each grammar = own alphabet
Each module = own rules	Each grammar = own rules
Interface	Master
Autonomy and parallelism	Autonomy and parallelism
Output by the interface	Output by the master
+ Generative power than modules	+ Generative power than components
Modularity	Modularity
Distribution	Distribution
Coordination	Coordination

6. Both AS and PCGS *exceed the generative power of their component grammars.*
7. *Parallelism, modularity, distribution, autonomy, and coordination* are important notions in both theories.

These similarities are summarized in Table 1.

Taking into account similarities between elements and function in autolexical syntax and grammar systems, and without forgetting the good results the former has reached in explaining some linguistic phenomena that other grammatical theories cannot easily handle, we suggest that the grammar systems theory can be applied without much effort to linguistic issues and that, due to its features, it may offer useful tools to account for arrangement and interaction of the various dimensions of natural language grammar. This suitability is demonstrated by the following definition of *Linguistic Grammar Systems* (LGS).

6 Linguistic grammar systems

Since the basic definition of grammar systems would be too simple to fully account for the intricate arrangement of components in grammar, we define an alternative model which, by combining several properties of different variants of grammar systems, may provide an adequate description of grammar architecture: Linguistic Grammar Systems (LGS).

The LGS offers a highly modular general device in which properties such as modularity, parallelism, interaction, coordination, and distribution can be captured. We want to keep our LGS as framework-free as possible in order to show which traits are indispensable in any model that seeks to account for natural language grammar issues, and in order to isolate the features that make the most sense no matter what specific machinery one chooses for writing grammars. We will focus on the interaction among modules and on the general design of the system, rather than giving a detailed account of every module.

In order to define LGS we have selected what we consider to be the most suitable tools provided by grammar systems theory in order to obtain a model that meets the requirements of the generation of grammatical representations. Since in this paper we want to establish an analogy between our model and AS, the postulates of this linguistic theory will guide decisions in the selection of grammar systems tools. Consider the following ideas from Sadock (1991, p. 19):

– we take a grammar to be a set of subgrammars called modules;
– each of these modules is a grammar of an independent level of linguistic representation;
– these modules are not hierarchically related to one another;
– a module need not wait for the output of another to do its work, but has the power to generate (analyze) an infinite set of representations quite independently of what is going on in any of the other components;
– each component is a self-contained system, with its own independent set of rules, principles and basic vocabulary;
– the lexicon plays a special, transmodular role in the theory.

In order to capture every feature of the above list, we have to define LGS as a *PCGS* (cf. Definitions 4–6). Such a variant offers the possibility to (1) have different modules that represent independent levels of linguistic representation; (2) have modules working in parallel and thus not hierarchically related; (3) account for an independent set of rules, principles and basic vocabulary for each module (the presence of different basic vocabulary led us to choose PCGS with separate alphabets as defined in Mihalache (1996); (4) have a special, transmodular module: the master.

The modular approach to grammar has been shown to have important consequences for the study of language and has been used not only by AS but by other grammatical theories. These range from Chomsky's generative grammar to Jackendoff's (1997) model, and to Harnish and Farmer's (1984) modular theory or Farmer (1984), where a theory of language is built as a system of rules and representations factorable into independent but interacting subsystems. One of the main advantages of modular grammars is that they can reduce the delays imposed by nonmodular grammars. This reduction of delays is due to the fact that in modular grammars subsystems of grammatical principles can be applied independently of each other and in parallel (cf. Frazier 1988). The same idea of modularity has been proposed by many authors in natural language processing (cf. Allen 1987; Altman 1987; Clifton and Ferreira 1987; Frazier 1988; Sabah 1997).

Now, we know that a PCGS can have any type of mechanism as its component, so we have to establish which type of device we will have as modules of the PCGS we have considered. Taking into account the advantages of modularity, we may think of grammar as a modular system consisting of several parallel interacting components (syntax, semantics…) which are internally modular (divided into several different modules, as well). In fact, several authors have defended internal modularity in the various dimensions of grammar. In Crocker (1991), for example, a highly modular organization of *syntax* is suggested where modules are determined by the representations they recover. The author proposes four modules in the syntactic processor, each related with a 'representational' or 'informational' aspect of grammar: (1) Phrase Structure (X-Bar Theory, Move α); (2) Chains (Bounding Theory, Case Filter); (3) Thematic Structure (θ Theory); (4) Coindexation (Binding and Control Theory). Also in Weinberg (1987) a modularization of the syntactic module is proposed. The author argues that the syntactic processor first creates a basic syntactic tree using phrase-structure, selectional, and subcategorization features together with information retrieved using bounded amounts of prior context. Then, from the first-stage representation it constructs another structure which it uses to establish binding relationships between categories. Another example of internal modularity, this time in *semantic and phonological* modules, is presented in Jackendoff (1990) where the author suggests that:

> … meaning, like phonological structure, is organized into independent but interacting tiers, each of which contributes a different class of conceptual distinctions to meaning as a whole (Jackendoff 1990, p. 2).

Studies on modularity in *morphology* can be found in Everaert et al (1988). In Wilson and Sperber (1991) there are some approaches to the internal modularity of *pragmatics*, and also in Kasher (1991) where pragmatics is divided into three independent parts, and again in Horn (1988) where it is argued that:

> Conceptually distinct subcomponents of pragmatic analysis may be simultaneously called upon within a single explanatory account of a given phenomenon, just as autonomous but interacting grammatical systems may interact to

yield the simplest, most general, and most comprehensive treatment of some linguistic phenomenon. (p. 115)

So, from the above examples it follows that not only can grammar as a whole be regarded as a modular system consisting of several parallel interacting components (syntax, semantics, phonology, etc.), but that also every component that builds up grammar can be viewed as *internally modular*, since it is divided into several different modules as well. Thus, we need to account for two levels of modularity. The first one, i.e. the one that states that LGS are made up of several modules, namely syntax, semantics, pragmatics, etc., is already accounted for by postulating a PCGS as the definition for our framework. One way to account for the second one, i.e. for internal modularity of components that form LGS, is to consider that each module of the framework is a CDGS, where different components work sequentially, cooperating with each other in order to generate the language of the system, i.e. the corresponding syntactic, semantic, phonological or any other structure. LGS could thus be considered as a doubly-modular framework: it is modular first in that it is made up of several modules that work in parallel; and second because every module in the system is internally modular.

We have defined independent modules with different alphabets and different rules. But, we also have to defend interaction among modules. The idea of interaction among different dimensions of grammar is widely accepted in the linguistic literature. Some examples from Smith (1991) will help us to understand this necessity. Consider a parser that has to parse the following sentences:

(1) The wealthy can eat soup.
(2) I saw the elderly man and child.
(3) As he unlocked the door with his key the fire engine arrived.

In (1) the parser will not be able to disambiguate *can* until it has seen the entire sentence. In (2) a parser cannot decide whether *elderly* modifies only *man* or *man and child*. Nor can it decide whether the prepositional phrase in (3) modifies *unlocked* or *arrived*. So, from those examples it follows that many sentences have multiple parses, and syntax is not able to choose between them without the help of semantics. The fact that none of those ambiguities impede communication suggests that semantics comes into play early in the process of sentence comprehension. Something similar happens if we look at semantic ambiguities. Consider the following sentence:

(4) Marie flew John from New York to Boston.

If we use only the semantic module to analyze this sentence, we will have four different interpretations: either *Marie* or *John* may be agent; either *New York* or *Boston* may be the origin and the destination. To resolve these ambiguities we could incorporate some information about word order and preposition, and thus introduce syntax.

In spite of our assertion that syntactic structures are different from semantic ones (and so consequently they may be generated by different modules), all the above suggests that there should be interaction between syntax and semantics, both modules should cooperate somehow with each other. The same can be said for other dimensions of grammar.

Bearing in mind the idea of interaction, let us consider the *correspondence rules* outlined in Jackendoff (1997):

It appears that phonological structure and syntactic structure are independent formal systems, each of which provides an exhaustive analysis of sentences in its own terms. [...] Each of these requires a generative grammar, a source of discrete infinity, neither of which can be reduced to the other. Given the coexistence of these two independent analyses of sentences, it is a conceptual necessity that the grammar contain a set of correspondence rules that mediate between syntactic and phonological units. Such rules must be of the general form: 'Syntactic structure X must/may/preferably does correspond to phonological structure Y', pairing a structural description in syntactic terms with one in phonological terms. [...] This prescription can be generalized to the structure of any interface between two distinct forms of representation. [...] The correspondence rules do not perform derivations in the standard sense of mapping a structure within a given format into another structure within the same format: they map between one format of representation and another [...] (p. 28).

Taking into account the above and having defined independent modules with different alphabets and different rules, it appears that we have to find a way in GS of establishing correspondences among modules. The mechanism we have chosen is the one present in PCGS with renaming, namely, *weak codes* (cf. Section 3). As can be noted, the idea of weak codes in PCGS with renaming is very close to the 'correspondence rules' defended in a parallel model of grammar as is Jackendoff's.

Continuing now with our analogy with AS, consider the following idea from Sadock (1991):

There is an overarching control center, an interface protocol that checks to see whether the members of a set of parsings from the individual components fit one another and count as parsings of the same total expression. (p. 20).

The overarching control of AS may be represented in LGS by the *master*. In fact, we consider the master as being a meta-module whose task consists of coordinating strings generated by other modules in order to provide the language of the system. In accordance to this idea, we have defined the master as a grammar system without an axiom, that is, not able to start its work until the moment it receives strings from the rest of the components. Note that the master doesn't have an *output* filter, this is because it does not send any string to any other component. Moreover, the master's input filter is defined over the union of *terminal alphabets* of components $\gamma_2, \ldots, \gamma_n$, since only terminal strings from the point of view of ith component, with $2 \leq i \leq n$, are accepted by this coordinating element. The master's task is to rewrite (lexicalize) what the modules of the system have already generated as acceptable terminal strings (always from a module point of view).

The above set of decisions is formally captured in the following definition:

Definition 13 A LGS of degree $n + m$, with $n, m \geq 1$ is an $(n + m + 1)$-tuple:

$$\Gamma = (K, (\gamma_1, I_1), (\gamma_2, I_2, O_2), \ldots, (\gamma_n, I_n, O_n), h_1, \ldots, h_m),$$

where:

- $K = \{Q_1, \ldots, Q_n, q_1, \ldots, q_n\}$ are query symbols, their indices $1, \ldots, n$ pointing to $\gamma_1, \ldots, \gamma_n$ components, respectively. $Q_1, \ldots, Q_n$ refer to the *whole* string of the ith component, while $q_1, \ldots, q_n$ make reference to a *substring* of the ith component.
- $(\gamma_1, I_1,), (\gamma_2, I_2, O_2), \ldots, (\gamma_n, I_n, O_n)$ are the components of the system:
  - $\gamma_1 = (N_1, T_1, G_1, \ldots, G_k, f_1)$, is the 'master' of the system, where:
    - N_1 is the nonterminal alphabet;
    - T_1 is the terminal alphabet;
    - There is no axiom;
    - $G_r = (N_1, T_1, P_r)$, for $1 \leq r \leq k$, is a usual Chomsky grammar, where:
      - N_1 is the nonterminal alphabet;
      - T_1 is the terminal alphabet;
      - P_r is a finite set of rewriting rules over $N_1 \cup T_1 \cup K \cup K'$, where $K' = \{[h_j, Q_i] \mid 1 \leq i \leq n, 1 \leq j \leq m\}$, and every $[h_j, Q_i]$ is a symbol.
    - $f_1 \in \{*, t, \leq k, \geq k, = k\}$ is the derivation mode of γ_1.
  - $\gamma_i = (N_i, T_i, S_i, G_1, \ldots, G_s, f_i)$, for $2 \leq i \leq n$, is a CDGS where:
    - N_i is the nonterminal alphabet;
    - T_i is the terminal alphabet;
    - S_i is the axiom;
    - $G_r = (N_i, T_i, P_r)$, for $1 \leq r \leq s$, is a usual Chomsky grammar, where:
      - N_i is the nonterminal alphabet;
      - T_i is the terminal alphabet;
      - P_r is a finite set of rewriting rules over $N_i \cup T_i \cup K \cup K'$, where $K' = \{[h_j, Q_i] \mid 1 \leq i \leq n, 1 \leq j \leq m\}$, and every $[h_j, Q_i]$ is a symbol.
    - $f_i \in \{*, t, \leq k, \geq k, = k\}$ is the derivation mode of γ_i.
  - $I_i \subseteq \bigcup_{j=2}^{n} T_j^*$, $i = 1$ is the input filter of the master.
  - $I_i \subseteq \bigcup_{j=2}^{n} (N_j \cup T_j)^*$, $2 \leq i \leq n$, is the input filter of the ith component.
  - $O_i \subseteq \bigcup_{j=1}^{n} (N_j \cup T_j)^*$, $2 \leq i \leq n$, is the output filter of the ith component.
- $h_j : \bigcup_{i=1}^{n} (N_i \cup T_i)^* \to \bigcup_{i=1}^{n} (N_i \cup T_i)^*$, $1 \leq j \leq m$, are weak codes such that:
  - $h_j(A) = A$, for $A \in N$;
  - $h_j(a) \in T \cup \{\lambda\}$, for $a \in T$.

We write $V_i = N_i \cup T_i \cup K \cup K'$ and $V_\Gamma = \bigcup_{i=1}^{n} (N_i \cup T_i) \cup K \cup K'$.
The sets N_i, T_i, K, K' are mutually disjoint for any i, $1 \leq i \leq n$.
We do not require $N_i \cap T_j = \emptyset$, for $1 \leq i, j \leq n$, $i \neq j$.

The above definition gives us a static view of LGS. A dynamic characterization of those systems requires the definition of the derivation process. But, in order to define the derivation process, we need to specify how the *state* of an LGS is to be understood at any moment in time:

Definition 14 Given a LGS $\Gamma = (K, (\gamma_1, I_1), (\gamma_2, I_2, O_2), \ldots, (\gamma_n, I_n, O_n), h_1, \ldots, h_m)$, its state is described at any moment by an n-tuple $(x_1, \ldots, x_n)$, where each $x_i \subseteq V_i^*$, $1 \leq i \leq n$, represents the string that is available at node i at that moment.

Let us continue with our remarks on the features of AS. Consider the following assertions:

- 'A module has the power to generate (or equivalently, analyze) an infinite set of representations *quite independently* of what is going on in any of the other components.' (Sadock 1991, p. 20).

– 'AS radically separates syntax and semantics (as well as morphology and other subsystems) into maximally simple modules that *interact* in potentially complex ways.' (Kathman 1995, p. 104).

It thus seems that we have to account for two different kinds of steps in the derivation process:

1. *rewriting steps*. that will account for the generation of the representation associated with each component of the system. This means that every module in the system *rewrites* its own string according to its specific rules and its derivation mode, without taking much account of what is happening in the other modules.
2. and *communicating steps*. From the above assertions it would seem essential to account for the interaction among modules (see the words in italics). To allow interaction among modules, in grammar systems theory we have to use communication steps.

We believe that it might be interesting to have a combination of two types of communication in LGS, namely communication *by request* and communication *by command* (cf. Section 3). By postulating the two types of communication, components could ask both for information from other modules if they need it, and send information to other modules if they consider that their strings match the requirements of those modules. Suppose, for example, that the syntactic module needs some semantic information in order to solve a possible ambiguity that appears in its syntactic derivation. In such a situation, syntax would introduce a *query symbol* asking the semantic component for help: i.e. communication by request. Now, if the semantic module has already finished its derivation, it can send its string to the master: i.e. communication by command. The following extract from Jackendoff (1997) suggests the convenience of speaking of the two types of communication proposed here:

> Suppose the auditory-to-phonological processor, in response to a speech signal, dumps a string of phonetic information onto the phonology blackboard. The immediate response of the phonology module is to try to break this string into words in order to make it phonologically well formed. It sends a call [`this is a query symbol in our terms`] to the lexicon: "Do you know of any words that sound like this?". The lexicon [...] just sends back all candidates that have phonological structure compatible with the sample structure sent it. But it doesn't just write these candidates on the phonology blackboard. In fact, the syntactic and conceptual parts of words can't be written on the phonological blackboard. Rather the lexicon simultaneously sends [`this is a communication by command in our terms`] the syntactic and conceptual parts of the candidate items to the syntactic and conceptual blackboards respectively, stimulating those modules to get busy (p. 104).

Another decision we have to make regards which modules can use query symbols. In order to stress the idea of cooperation among modules, we have chosen to define a *noncentralized* system (cf. Definition 7) in which *every* component is allowed to introduce query symbols whenever it considers it necessary.

Regarding what can be interchanged in such communication steps, we defined two possibilities in order to capture what might be needed in a framework that accounts for grammatical issues: a module can send either its *whole current string* (whenever it is asked for it) or a *subword*[2] of its current string. Note that by having communicating subwords of a module's string we are accounting for the fact that not all the information present in one module is interesting for another one. For example, suppose that phonology needs syntactic information in order to perform its derivation. The information it needs is not every piece of syntactic information in the syntactic module, but one specific piece. It might not need information regarding c-commands relations but only information about word order. This decision is supported in Jackendoff (1997, p. 103) as well as the necessity, already pointed out, of having weak codes: '*any single level of representation typically may receive fragmentary input from a number of distinct sources* [...] *the best way to view the individual generative grammars is as providing a repertoire for integrating and filling out fragmentary information coming from other representations via correspondence rules*'.

Taking into account that we have stated that every module produces different structures (syntactic, semantic...), we do not want modules to re-start their derivation every time they perform a communication step. This is why we have defined a *nonreturning* PCGS instead of a returning one (cf. Definition 8).

We formally define derivation steps in the following terms:

Definition 15 Given a LGS $\Gamma = (K, (\gamma_1, I_1), (\gamma_2, I_2, O_2), \ldots, (\gamma_n, I_n, O_n), h_1, \ldots, h_m)$, for two n-tuples $(x_1, x_2, \ldots, x_n)$, $(y_1, y_2, \ldots, y_n)$, $x_i, y_i \in V_i^*$, $1 \leq i \leq n$, we write $(x_1, \ldots, x_n) \Longrightarrow (y_1, \ldots, y_n)$ if one of the following cases holds:

1. $|x_i|_K = 0$, $|x_i|_{K'} = 0$, $x_i \notin O_i$, and for all i, $1 \leq i \leq n$, we have $x_i \Longrightarrow y_i$ in the CDGS γ_i, or $x_i \in T_i^*$ and $x_i = y_i$.
   For each $\gamma_i = (N_i, T_i, S_i, G_1, \ldots, G_s, f_i)$, for $1 \leq i \leq n$, with $x_i, y_i \in V_i^*$ we write $x_i \Longrightarrow_{G_r}^{k} y_i$, for $1 \leq r \leq s$, iff there exists $x_1, x_2, \ldots, x_{k+1}$ such that:

   – $x_i = x_1$, $y_i = x_{k+1}$;
   – $x_j \Longrightarrow_{G_r} x_{j+1}$, i.e. $x_j = x'_j A_j x''_j$, $x_{j+1} = x'_j w_j x''_j$, $A_j \rightarrow w_j \in P_r$, $1 \leq j \leq k$.
2. There is an i, $1 \leq i \leq n$, such that $|x_i|_K > 0$, then for each such i we write $x_i = z_1 Q_{i_1} z_2 Q_{i_2} \ldots z_t Q_{i_t} z_{t+1}$ [or $x_i = z_1 q_{i_1} z_2 q_{i_2} \ldots z_t q_{i_t} z_{t+1}$ for subword communication], $t \geq 1$, for $z_j \in V_\Gamma^*$, $|z_j|_K = 0$, $1 \leq j \leq t+1$; if $|x_{i_j}|_{(K \cup K')} = 0$, $1 \leq j \leq t$, then $y_i = z_1 x_{i_1} z_2 x_{i_2} \ldots z_t x_{i_t} z_{t+1}$ providing that $y_i \in V_i^*$; when for some j, $1 \leq j \leq t$, $|x_{i_j}|_{(K \cup K')} \neq 0$, then $y_i = x_i$; for all i, $1 \leq i \leq n$, for which y_i is not specified above we have $y_i = x_i$.
3. There is an i, $1 \leq i \leq n$, such that $|x_i|_{K'} > 0$, then for each such i we write $x_i = z_1[h_j, Q_{i_1}]z_2[h_j, Q_{i_2}] \ldots z_t[h_j, Q_{i_t}]z_{t+1}$, [or $x_i = z_1[h_j, q_{i_1}]z_2[h_j, q_{i_2}] \ldots z_t[h_j, q_{i_t}]z_{t+1}$ for subword communication], $t \geq 1$, for $z_j \in V_\Gamma^*$, $|z_j|_{K'} = 0$, $1 \leq j \leq t+1$; if $|x_{i_j}|_{(K \cup K')} = 0$, $1 \leq j \leq t$, then $y_i = z_1 h_j(x_{i_1}) z_2 h_j(x_{i_2}) \ldots z_t h_j(x_{i_t}) z_{t+1}$ providing that $y_i \in V_i^*$; when for some j, $1 \leq j \leq t$, $|x_{i_j}|_{(K \cup K')} \neq 0$, then $y_i = x_i$; for all i, $1 \leq i \leq n$, for which y_i is not specified above we have $y_i = x_i$.
4. $(x_1, \ldots, x_n) \vdash (y_1, \ldots, y_n)$ iff $y_i = x_i \cdot (I_i \cap (\bigcup_{j=1, i \neq j}^{n} O_j \cap x_j))$, for $i = 1, \ldots, n$.

Point 1 defines a *rewriting step*. In 2, we define a *communication step by request without renaming*. In 3, we define a *communication step by request with renaming*. And finally, in 4, we define *a communication step by command.*

[2] This idea of communicating a *substring* of the current string of a specific module is already present in Păun (1996b) where a new class of PCGS in which prefixes of the current sentential forms can be communicated is introduced.

The derivation process is carried out by alternating rewriting and communication steps. After every rewriting step, we check whether a communication step (by request or command) is possible. If so, we perform it, if not, we continue derivation with a new rewriting step. The process is finished when, after reaching a terminal string, every CDGS in the LGS has sent by command its result to the master. As soon as the master has all the strings generated by the various components of the system, it applies its rules in order to unify or coordinate those strings, and thus generates the language of the LGS.

Definition 16 The language generated by a LGS as above is:

$$L(\Gamma) = \{x \in T_1^* \mid (\lambda, S_2, \ldots, S_3) \Longrightarrow (\lambda, \alpha_2^{(1)}, \ldots, \alpha_n^{(1)}) \vdash (\lambda, y_2^{(1)}, \ldots, y_n^{(1)}) \Longrightarrow (\lambda, \alpha_2^{(2)}, \ldots, \alpha_n^{(2)}) \vdash (\lambda, y_2^{(2)}, \ldots, y_n^{(2)}) \Longrightarrow \ldots \Longrightarrow (\lambda, \alpha_2^{(s)}, \ldots, \alpha_n^{(s)}) \vdash (\alpha_2^{(s)} \ldots \alpha_n^{(s)}, \alpha_2^{(s)}, \ldots, \alpha_n^{(s)}) \Longrightarrow \ldots \Longrightarrow (x, \alpha_2^{(s)}, \ldots, \alpha_n^{(s)}), s \geq 1, \alpha_i^{(s)} \in T_i^*, 2 \leq i \leq n\}.$$

Note that in contrast with what happens in AS where:

> 'an output of such a grammar is a set of triples $\{r_{\text{syn}}, r_{\text{sem}}, r_{\text{mor}}\}$ where each such triple corresponds to a fully-fledged expression of the language to which the grammar ascribes syntactic, semantic and morphological parsings' (Sadock 1991, p. 20).

in LGS what we obtain is a *single* language. This language would be the result of putting in correspondence, *via the master*, every structure generated by every component of the framework.

The result of the above set of decisions is a non-centralized, non-returning macro-PCGS with renaming, with separate alphabets, with two types of communication (command or request), with the possibility of subword communication, and whose

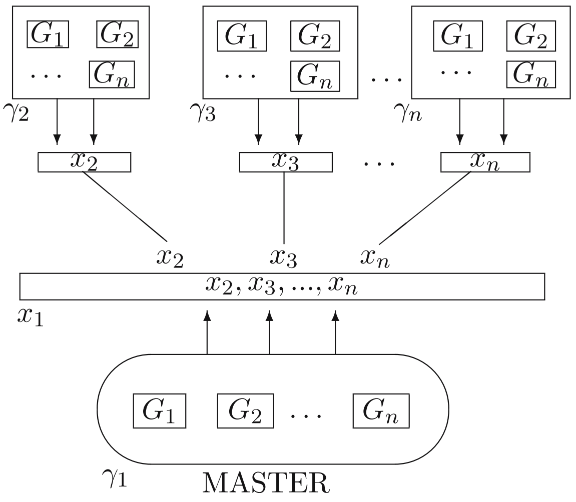


Fig. 2 Linguistic grammar system

components are CDGS built up by several grammars and containing input and output filters in order to allow for communication by command (see Fig. 2).

The following example illustrates how the LGS functions. For the sake of simplicity, we only indicate the necessary components of the system.

Example 1 Consider the following LGS:

$$\Gamma = (K, (\gamma_1, I_1), (\gamma_2, I_2, O_2), (\gamma_3, I_3, O_3), (\gamma_4, I_4, O_4), h_1),$$

where:

- $\gamma_1 = (N_1, T_1, G_{1_1}, G_{1_2}, t)$ is the master: $N_1 = \{A, B, C, D\}$; $T_1 = \{a, b, c, d\}$; $G_{1_1} = \{A \to a, B \to b\}$; $G_{1_2} = \{C \to c, D \to d\}$; $I_1 = \{A, B, C, D\}^*$.
- $\gamma_2 = (N_2, T_2, S_2, G_{2_1}, G_{2_2}, *)$: $N_2 = \{S_2\}$; $T_2 = \{A\}$; $G_{2_1} = \{S_2 \to AS_2\}$; $G_{2_2} = \{S_2 \to A\}$; $I_2 = \{A\}^*$; $O_2 = \{A\}^*$.
- $\gamma_3 = (N_3, T_3, S_3, G_{3_1}, G_{3_2}, *)$: $N_3 = \{S_3\}$; $T_3 = \{B\}$; $G_{3_1} = \{S_3 \to BS_3\}$; $G_{3_2} = \{S_3 \to B\}$; $I_3 = \{B\}^*$; $O_3 = \{B\}^*$.
- $\gamma_4 = (N_4, T_4, S_4, G_{4_1}, G_{4_2}, *)$: $N_4 = \{S_4\}$; $T_4 = \{C, D\}$; $G_{4_1} = \{S_4 \to S_4\}$; $G_{4_2} = \{S_4 \to [h_1Q_2][h_1Q_3]\}$; $I_4 = \{B\}^*$; $O_4 = \{B\}^*$.
- $h_1(A) = C$, $h_1(B) = D$, $h_1(C) = A$, $h_1(D) = B$.

We start the derivation with an empty string in the first component and with the axioms of γ_2, γ_3, γ_4. We apply the first component of every CDGS:

$$(\lambda, S_2, S_3, S_4) \Longrightarrow (\lambda, AS_2, BS_3, S_4).$$

We can apply reiteratively the first component of every CDGS γ_i because the derivation mode under which they are working is $*$:

$$\begin{aligned}(\lambda, AS_2, BS_3, S_4) &\Longrightarrow (\lambda, AAS_2, BBS_3, S_4) \Longrightarrow (\lambda, AAAS_2, BBBS_3, S_4)\\ &\Longrightarrow \cdots \Longrightarrow (\lambda, A^nS_2, B^mS_3, S_4).\end{aligned}$$

We can apply, now, the second component of every CDGS γ_i:

$$(\lambda, A^nS_2, B^mS_3, S_4) \Longrightarrow (\lambda, A^{n+1}, B^{m+1}, [h_1Q_2][h_1Q_3]).$$

Since within the string of the fourth component two query symbols have appeared, we cannot continue with rewriting steps, but we must perform two communication steps. Note that the query symbols present in γ_4 refer to strings of the second and third component, respectively, but require a translation via the weak codes present in the system. We thus have to perform a communication step with renaming:

$$\begin{aligned}(\lambda, A^{n+1}, B^{m+1}, [h_1Q_2][h_1Q_3]) &\Longrightarrow (\lambda, A^{n+1}, B^{m+1}, C^{n+1}[h_1Q_3])\\ &\Longrightarrow (\lambda, A^{n+1}, B^{m+1}, C^{n+1}D^{m+1}).\end{aligned}$$

Note that, since we have defined LGS as *nonreturning*, components γ_2 and γ_3 send copies of their strings, in such a way that they need not re-start from the axiom again, but rather keep their string.

No further rewriting step can now be performed on components γ_2, γ_3 and γ_4 because they have reached a terminal string. Furthermore, the strings of each component match its respective output filter and master's input filter. This means that a communication step by command can be performed, in which every component sends its string to the master:

$$(\lambda, A^{n+1}, B^{m+1}, C^{n+1}D^{m+1}) \vdash (A^{n+1}B^{m+1}C^{n+1}D^{m+1}, A^{n+1}, B^{m+1}, C^{n+1}D^{m+1}).$$

Note that strings sent by components to the master have been concatenated by order of system components. Moreover, γ_2, γ_3, and γ_4 have sent copies of their strings, thus keeping their strings and without needing to start again from the axiom.

No further rewriting step can be performed on γ_2, γ_3, and γ_4. Now, it is the turn of the master. It has received strings from every component and must rewrite them until it reaches a terminal string. We apply, first, component G_{1_1}. Since γ_1 has to work in t mode, the first component should perform as many rewriting steps as it can:

$$\begin{aligned}
&(A^{n+1}B^{m+1}C^{n+1}D^{m+1}, A^{n+1}, B^{m+1}, C^{n+1}D^{m+1}) \Longrightarrow \\
&(aA^{n}B^{m+1}C^{n+1}D^{m+1}, A^{n+1}, B^{m+1}, C^{n+1}D^{m+1}) \Longrightarrow \\
&(aA^{n}bB^{m}C^{n+1}D^{m+1}, A^{n+1}, B^{m+1}, C^{n+1}D^{m+1}) \Longrightarrow \\
&\cdots \Longrightarrow (a^{n}b^{m}C^{n+1}D^{m+1}, A^{n+1}, B^{m+1}, C^{n+1}D^{m+1}).
\end{aligned}$$

Component G_{1_1} cannot be further applied, so we must turn to G_{1_2}:

$$\begin{aligned}
&(a^{n}b^{m}C^{n+1}D^{m+1}, A^{n+1}, B^{m+1}, C^{n+1}D^{m+1}) \Longrightarrow \\
&(a^{n}b^{m}cC^{n}D^{m+1}, A^{n+1}, B^{m+1}, C^{n+1}D^{m+1}) \Longrightarrow \\
&(a^{n}b^{m}cC^{n}dD^{m}, A^{n+1}, B^{m+1}, C^{n+1}D^{m+1}) \Longrightarrow \cdots \\
&\Longrightarrow (a^{n}b^{m}c^{n}d^{m}, A^{n+1}, B^{m+1}, C^{n+1}D^{m+1}).
\end{aligned}$$

γ_1 has reached a terminal string, so we can consider that derivation in Γ as finished. According to our definition, the language generated by the system is the language of the master, so we are not interested in strings of γ_2, γ_3, and γ_4, and we consider that language of Γ to be:

$$L(\Gamma) = \{a^{n}b^{m}c^{n}d^{m} \mid n, m \geq 1\}.$$

6.1 LGS in natural language grammar: an example

According to the framework we have proposed, a natural language expression could be considered as the final language of a PCGS whose modules represent each of the simultaneously different informational dimensions along which natural language expression is organized. The functioning of the system in order to generate an acceptable language structure can be outlined as follows.

Let us consider that our LGS consists of the following components, each containing different terminal and non-terminal alphabets as well as different rules:

1. Syntactical CDGS: units such as N, V, A, P, NP or VP and syntactic rules.
2. Semantic CDGS: units such as physical objects, events, properties, times, quantities, intentions, etc. and semantic rules.

3. Phonological CDGS: units such as phonological distinctive features, notions of syllable, word, phonological and intonational phase, stress, tone, intonation contour, etc. and principles of phonological combination.
4. Master (the Lexicon): words and rules for coordinating the structures generated by the three modules.

LGS starts functioning as soon as every module starts its derivation process:

– Within the *syntactic module*, several grammars, each responsible for a different syntactic level (e.g. phrase structure, dependencies, agreement, case marking), cooperate distributively, and sequentially, in order to produce a well-formed syntactic structure.
– Within the *semantic module*, several grammars (e.g. one responsible for function-argument relations, another for variable-binder relations, etc.) cooperate sequentially in order to produce a semantically well-formed structure.
– Within the *phonological module*, where a module can be responsible for syllable structure, another for stress assignment, etc. the generation of a well-formed phonological structure takes place sequentially.
– In the meantime, while those three modules are working independently, nothing happens in the *master* since this module does not contain any information: it must wait for strings produced by the other three modules.

Note that derivation processes in LGS take place in parallel and independently. That is, every module in the system will generate its structure according to its own rules and primitives, without waiting for the outputs of any other module.

According to the above scheme we have three parallel modules working independently with the master waiting. However it is not realistic to think that there is no interaction at all among those three modules. In fact, from time to time, they need information from one another in order to solve possible difficulties and ambiguities in their derivations:

– Let us suppose that the phonological module needs information about word-order for it to continue with its derivation. What happens at that point? The phonological module introduces a query symbol referring to the syntactic module and asking for such subword of the syntactic string that accounts for word-order. Once the phonological module has introduced such a query, rewriting processes stop, and communication is performed: the syntactic module sends to the phonological module the string that the latter has asked for. Once the communication has been performed, modules re-start their function from where they left off.
– Rewriting processes continue and now it is the syntactic module that requires some information from the semantic module. But this time, syntax introduces neither a query symbol Q_i nor a q_i, but a $[h_j, Q_i]$, that is, a query symbol with renaming. Suppose that at this stage of the derivation, the semantic module contains the string [Agent Theme Locative] and suppose also that the LGS contains the following weak codes: (1) $h_j(Agent) = NP$, (2) $h_j(Theme) = NP$, (3) $h_j(Locative) = PP$. Provided that the syntactic module has asked for the current semantic string, but that it has required a translation before communication takes place, we would consider that the string that the syntactic module will get from that communication step is something like [NP NP PP], that is, it will receive the semantic string translated by means of the weak codes defined in the system.

Since we have shown that LGS works in a nonreturning way, we will consider that modules send copies of their strings in such a way that they do not need to re-start from their axiom every time a communication step takes place.

Derivation processes in each of those three modules continue by alternating rewriting and communication until they reach three terminal strings. Each of those strings will be considered as terminal from the respective module's point of view. That is, the first module will reach a syntactic string; the second one, a semantic string; and the third one, a phonological string. Since derivations in each module cannot go on, because all of them have already reached a terminal string that cannot be further rewritten, they will perform a communication step by command. Each module will send its own terminal string to the master. So, the master will get a syntactic, a semantic and a phonological string, respectively.

Once the master has received strings produced by every module of the system, it starts its work. According to its rules, it tries to combine and put in correspondence, the three structures generated by the syntactic, semantic and phonological modules, respectively. The master's task consists of lexicalizing the structures it has received; that is, its task is to rewrite the syntactic, semantic, and phonological structures by introducing words. If it is possible to lexicalize those structures, if they are compatible and they can be lexicalized by means of the same words, then we could say that the master has successfully reached a terminal string and that, therefore, a well-formed natural language expression has been generated by the LGS.

Let us see now how can we generate an example of the so-called cross-serial dependencies of Dutch by using a very simple LGS.

Consider the following LGS with three components:

$$\Gamma = (K, (\gamma_1, I_1), (\gamma_2, I_2, O_2), (\gamma_3, I_3, O_3), h_1),$$

where:

- $\gamma_1 = (N_1, T_1, G_{1_1}, G_{1_2}, G_{1_3}, t)$ is the master of the LGS:
  - $N_1 = \{A, B\}$;
  - $T_1 = \{Jan, Piet, Marie, laat, help, zie\}$;
  - $G_{1_1} = \{A \rightarrow Jan, A \rightarrow Piet, A \rightarrow Marie, A' \rightarrow Jan', A' \rightarrow Piet', A' \rightarrow Marie'\}$;
  - $G_{1_2} = \{B \rightarrow laat, B \rightarrow help, B \rightarrow zie, B' \rightarrow laat', B' \rightarrow help', B' \rightarrow zie'\}$;
  - $G_{1_3} = \{Jan' \rightarrow \lambda, Piet' \rightarrow \lambda, Marie' \rightarrow \lambda, laat' \rightarrow \lambda, help' \rightarrow \lambda, zie' \rightarrow \lambda\}$;
  - $I_1 = \{A, B\}^*$.
- $\gamma_2 = (N_2, T_2, S_2, G_{2_1}, G_{2_2}, *)$ is the syntactic component:
  - $N_2 = \{S_2, X\}$;
  - $T_2 = \{A, B\}$;
  - $G_{2_1} = \{S_2 \rightarrow AX\}$;
  - $G_{2_2} = \{X \rightarrow S_2B, X \rightarrow B\}$;
  - $I_2 = \{A, B\}^*$;
  - $O_2 = \{A, B\}^*$.
- $\gamma_3 = (N_3, T_3, S_3, G_{3_1}, G_{3_2}, *)$ is the semantic component:
  - $N_3 = \{S_3, X\}$;
  - $T_3 = \{A', B'\}$;
  - $G_{3_1} = \{S_3 \rightarrow A'X\}$;

- $G_{3_2} = \{X \rightarrow B'S_3, X \rightarrow B'\}$;
- $I_3 = \{A', B'\}^*$;
- $O_3 = \{A', B'\}^*$.

– $h_1(A') = A, h_1(B') = B, h_1(A) = A', h_1(B) = B'$.

Derivation proceeds as follows. We use alternatively the first and the second component of γ_2 and γ_3 in the following way:

$$(\lambda, S_2, S_3) \Longrightarrow (\lambda, AX, A'X) \Longrightarrow (\lambda, AS_3B, A'B'S_2) \Longrightarrow \\ (\lambda, AAXB, A'B'A'X) \Longrightarrow (\lambda, AAS_3BB, A'B'A'B'S_2) \Longrightarrow \\ (\lambda, AAAXBB, A'B'A'B'A'X) \Longrightarrow (\lambda, AAABBB, A'B'A'B'A'B').$$

Now since both syntax and semantics have reached a terminal string from their points of view, a communication by command takes place, and the syntactic and semantic strings are sent to the master:

$$(\lambda, AAABBB, A'B'A'B'A'B') \vdash (AAABBBA'B'A'B'A'B', AAABBB, A'B'A'B'A'B').$$

Now the master starts its work by lexicalizing the semantic and syntactic structures it has got. First, the first component of γ_1 is used in the terminal mode:

(AAABBBA'B'A'B'A'B', AAABBB, A'B'A'B'A'B') $\Longrightarrow^*$
(Jan Piet Marie BBB Jan'B'Piet'B'Marie'B', AAABBB, A'B'A'B'A'B').

Then, component G_{1_2} of γ_1 must be used in the terminal mode:

(Jan Piet Marie BBB Jan' B' Piet' B' Marie' B', AAABBB,
A'B'A'B'A'B') $\Longrightarrow^*$ (Jan Piet Marie laat help zie Jan' laat' Piet'
help' Marie' zie', AAABBB, A'B'A'B'A'B).

Now the derivation ends by applying component G_{1_3} of the master:

(Jan Piet Marie laat help zie Jan' laat' Piet'help' Marie' zie',
AAABBB, A'B'A'B'A'B') $\Longrightarrow^*$ (Jan Piet Marie laat help zie $\lambda\lambda\lambda\lambda\lambda\lambda$,.
AABBB, A'B'A'B'A'B')

Since we just consider the string of the master as the output of the system, we can see that in this case what we have obtained is the following sentence:

'Jan Piet Marie laat helpen zien'

However, this output has been obtained thanks to the collaboration of three modules. In fact, note that the string of γ_2 (i.e. *AAABBB*) can be considered a representation of the syntax of the sentence, while the string of component γ_3 (i.e. $A'B'A'B'A'B'$) offers a pretty fair representation of the meaning (semantics) of the Dutch subordinate sentence.

Summing up, with this example we have shown how a simple system, as the LGS used, with only two autonomous components coordinated by the master can characterize the properties of the above Dutch subordinate sentence. The cross serial dependencies are problematic when attributed to a single level of structure, but not when they reside at the interface between two. For problems related to cross-serial dependencies see Bresnan et al (1987).

Note that in the example we have presented:

- modules work independently and in a parallel and autonomous fashion;
- they have different rules, different alphabets, and produce different structures;
- they can interact via communication steps, interchanging in this way information that can facilitate their work;
- once every module has reached a terminal string, it sends this string to the master, with a communication step by command;
- the master puts in correspondence syntactic and semantic strings and generates the language of the system.

As can be seen, we do not get an *n*-tuple of structures as the language of the system, but a single language, namely the natural language expression which, of course, combines in it the several informational dimensions along which natural language expressions are supposed to be organized. Every such dimension is present in the final result, because we have obtained such a final language thanks to the cooperation of two different modules and the coordinating labor of the master.

6.2 Some notes on the linguistic motivation of LGS

The relationship between formal and natural languages is not something new. In fact, formal language theory was born in the middle of the 20th century as a tool for modelling and investigating syntax of natural languages. After 1964, it developed as a separate branch with specific problems, techniques and results. Now, taking into account this fact, and considering the emergence of grammar systems as a new research area in the field of formal languages, a somehow natural question is: Can natural languages be modelled by means of grammar systems? Can this new framework of formal languages become a new tool for modelling natural languages? In order to answer this question we have introduced LGS as a model that aims to:

1. *Reconstruct grammar of natural languages* by means of a formal-language model.
2. *Formulate* a *model* capable of generating and/or recognizing natural-language structures.
3. *Define a formalization* of language that because of its computational suitability can be implemented.
4. *Offer a method of linguistic manipulation* that can be useful for natural language processing.

Natural languages can be imagined as a number of modules that interact in a nonsimple way. Understanding and generation needs cooperative phonological, morphological, lexical, syntactic, pragmatic, semantic... modules. If we adopt a modular strategy to write our grammar model we will obtain advantages such as the following:

1. what is undoubtedly a complex task is divided into convenient and manageable subtasks;

2. we deal with one structure, or set of structures, at one time;
3. each module can be written and tested in isolation;
4. minor changes and/or additions can be introduced in one module if necessary, without having to modify the whole grammar.

In general, formal and computational approaches to natural language demand nonhierarchical, parallel, distributed models in order to explain the complexity of linguistic structures as the result of the interaction of a number of independent but cooperative modules. Linguistic grammar systems offer a modular theory where the various dimensions of linguistic representation are arranged in a parallel distributed framework and where the language of the system is the result of the interaction of those independent cooperative modules that form the LGS. Therefore, linguistic grammar systems provide a suitable framework according to the requirements of most of the current formal/computational approaches to natural language. Moreover, LGS offer a highly formalized model that offers several advantages with respect to classical formal languages and that can be easily implemented, two important features for the computational treatment of natural language.

According to Nijholt (1988, p. 239), a formal language can be considered as a model of a natural language. Similarly, a formal grammar can be considered as a model of a natural grammar and, therefore, it is a theory of this particular natural language. In the choice of the model we are restricted to models which give an account of linguistic intuitions and, at least, do not contradict human mental properties. We think that linguistic grammar systems do not contradict those properties, and might provide a realistic and useful theory of natural language where linguistic structures are generated in a modular, parallel and distributed fashion.

7 Final remarks

Grammar systems theory has been widely investigated and is now a well-developed formal theory that has several advantages over classical models. However, since it is a branch of formal languages, researchers in the field of grammar systems have concentrated mainly on theoretical aspects. Although some attempts at application have been carried out, research on grammar systems is still mainly theoretical. Taking into account results obtained from the formal study of the theory, this paper has tried to show that grammar systems theory is not only a good theory from a formal point of view, but that it may also be a useful tool from the point of view of its possible applications. We have tried to apply grammar systems to the description and analysis of natural language. To do this, we started by comparing grammar systems with AS. Both theories have reached good results in their respective disciplines thanks to some concrete features: *modularity*, *parallelism*, *cooperation*, and *interaction*. But, the similarities between AS and grammar systems go beyond this sharing of the same traits. As we have shown, in grammar systems there is a correspondence for each element that is part of an autolexical grammar.

The analogy we have shown between grammar systems and autolexical syntax has led us to suggest the possible applicability of grammar systems theory to the study of natural languages. The LGS have been introduced as a grammar systems model that may account for the arrangement and functioning of the various grammatical dimensions that work in order to generate natural language expressions. In this paper, we have focused on presenting the formal definitions of the model and justifying the grammar systems tools selected in order to define the new variant, i.e. LGS. The number of modules needed to describe natural language and the exact content of each of these modules are complex issues that should be determined. In fact, they have not been specified in this paper because we are essentially interested in generality and what we have presented here is just a rough approach to the topic. However, taking into account the known flexibility of grammar systems, we can consider as many modules as we deem necessary and the content of each of them can be whatever formalism, not necessarily string rewriting. By introducing LGS we simply want to offer a highly modular general device in which properties like modularity, parallelism, interaction, coordination, and distribution can be captured. We want to keep our LGS as framework-free as possible in order to show which traits are indispensable in any model that seeks to account for natural language grammar issues, and in order to isolate the features that make the most sense no matter what specific machinery one chooses for writing grammars. Since our first aim was to show the applicability of grammar systems, we have focused on interaction among modules and on the general design of the system, rather than giving a detailed account of every module.

The model we have presented is quite general. Despite this, we believe that it has the same features as in accepted grammatical theories (e.g. AS), along with a similar arrangement of components and quite a close way of functioning. Such grammatical theories have already been proved to be adequate natural language models and would seem to support the modest thesis of this paper, that is, that *LGS may be seen as a grammar-systems approach to natural language grammar thus showing the applicability of this theory*.

Acknowledgement
This research has been supported by a Marie Curie Fellowship of the European Community programme *Human Potential (IHP)* under contract number HPMF-CT-2001-01169.

References

Allen, J.F. (1987). *Natural language understanding*. CA: The Benjamin/Cummings Publishing Company.

Altman, G. (1987). Modularity and interaction in sentence processing. In Garfield, J.L. (ed.), *Modularity in knowledge representation and natural-language understanding* (pp. 249–257). Cambridge: MIT Press.

Bond, A.H. & Gasser, L. (eds.) (1988). *Readings in distributed artificial intelligence*. San Mateo: Morgan Kaufmann.

Bresnan, J. (2001). *Lexical functional syntax*. Oxford: Blackwell.

Bresnan, J., Kaplan, R.M. Peters, S. & Zaenen, A. (1987). Cross-serial dependencies in Dutch. In Savitch, W.J., Bach, E., Marsh, W. & Safran-Naveh, G. (eds.), *The Formal complexity of natural language* (pp. 286–319). Dordrecht: Kluwer.

Buszkowski, W., Marciszewski, W. & van Benthen, J. (eds.) (1988). *Categorial grammar*. Amsterdam: John Benjamins.

Chitų, A. (1997). PC grammar systems versus non-context-free constructions from natural and artificial languages. In Păun, Gh. & Salomaa, A. (eds.), *New trends in formal languages. Control, cooperation, and combinatorics* (pp. 278–287). LNCS, 1218. Berlin: Springer.

Chomsky, N. (1984). *Modular approaches to the study of mind*. San Diego: State University Press.

Clifton, C. Jr., & Ferreira, F. (1987). Modularity in sentence comprehension'. In Garfield, J.L. (ed.), *Modularity in knowledge representation and natural-language understanding* (pp. 277–290). Cambridge: MIT Press.

Crocker, M.W. (1991). Multiple meta-interpreters in a logical model of sentence processing. In Brown, C.G. & Koch, G. (eds.), *Natural language understanding and logic programming, III* (pp. 127–145). North-Holland: Elsevier.

Csuhaj-Varjú, E. & Dassow, J. (1990). On cooperating/distributed grammar systems. *Journal of Information Processing and Cybernetics (EIK), 26*, 49–63.

Csuhaj-Varjú, E., Dassow, J., Kelemen, J. & Păun, Gh. (1994). *Grammar systems: a grammatical approach to distribution and cooperation*. London: Gordon and Breach.

Csuhaj-Varjú, E., Jiménez-López, M.D. & Martín-Vide, C. (1999). Pragmatic eco-rewriting systems: pragmatics and eco-rewriting systems. In Păun, Gh. & Salomaa, A. (eds.), *Grammatical models of multi-agent systems* (pp. 262–283). London: Gordon and Breach.

Csuhaj-Varjú, E., Kelemen, J. & Păun, Gh. (1996). Grammar systems with WAVE-like communication. *Computers and AI, 15* (5), 419–436.

Dassow, J., Păun, Gh. & Rozenberg, G. (1997). Grammar systems. In Rozenberg, G. & Salomaa, A. (eds.), *Handbook of formal languages* (Vol. 2, pp. 155–213). Berlin: Springer.

Davis, R. (1980). Report on the workshop on distributed AI. *SIGART New-Letter 73*, 42–52.

Demazeau, Y. & Muller, J.P. (1990). *Decentralized artificial intelligence*. North-Holland: Elsevier.

Durfee, E.H., Lesser, V.R. & Corkill, D.D. (1989). Cooperative distributed problem solving. In Barr, A., Cohen, P.R. & Feigenbaum, E.A. (eds.), *The handbook of artificial intelligence* (Vol. IV, pp. 85–148). USA: Addison-Wesley.

Everaert, M., Evers, A., Huybregts, R. & Trommelen, M. (eds.) (1988). *Morphology and modularity: in honour of Henk Schultink*. USA: Foris.

Farmer, A.K. (1984). *Modularity in syntax: a study of Japanese and English*. Cambridge: MIT Press.

Frazier, L. (1988). Grammar and Language Processing. In F.J. Newmeyer (ed.), *Linguistic theory: extensions and implications* (Vol. II, pp. 15–34). Cambridge University Press.

Gazdar, G., Klein, E., Pullum, G. & Sag, I. (1985). *Generalized phrase structure grammar*. Oxford: Blackwell.

Harnish, R.M. & Farmer, A.K. (1984). Pragmatics and the modularity of the Linguistic System, *Lingua 63*. 255–277.

Horn, L.R. (1988). Pragmatic theory. In Newmeyer F.J. (ed.), *Linguistic theory: foundations* (Vol. I, pp. 113–145). Linguistics: The Cambridge Survey. Cambridge: Cambridge University Press.

Hudson, R. (1984). *Word grammar*. Oxford: Blackwell.

Jackendoff, R. (2002). *Foundations of language. Brain, meaning, grammar, evolution*. Oxford: Oxford University Press.

Jackendoff, R. (1990). *Semantic structures*. Cambridge: MIT Press.

Jackendoff, R. (1997). *The architecture of language faculty*. Cambridge: MIT Press.

Jiménez-López, M.D. (2001). A new approach to natural language with formal languages. In Kelemen, J. & Kelemenová, A. (eds.), *Grammar systems and related fields*. Opava: Silesian University.

Jiménez-López, M.D. (2004). Formal languages for conversation analysis. In García Español, A. (ed.), *Estudios Hispánicos y Románicos*, Universitat Rovira i Virgili, Tarragona.

Jiménez-López, M.D. (2000). *Grammar systems: a formal-language-theoretic framework for linguistics and cultural evolution*. PhD Dissertation, Rovira i Virgili University, Tarragona.

Jiménez-López, M.D. (2003). Linguistic grammar systems: a grammar systems approach to natural languages. In Martín-Vide, C. & Mitrana, V. (eds.), *Grammars and Automata for String Processing: from mathematics and computer science to biology, and back* (pp. 55–65). London: Taylor and Francis.

Jiménez-López, M.D. (2002). *Using grammar systems*, GRLMC Report, Rovira i Virgili University, Tarragona.

Jiménez-López, M.D. & Martín-Vide, C. (2000). Grammar systems and autolexical syntax: two theories, one single idea. In Freund, R. & Kelemenová, A. (eds.), *Grammar systems 2000* (pp. 283–296). Opava: Silesian University.

Jiménez-López, M.D. & Martín-Vide, C. (1997). Grammar systems for the description of certain natural language facts. In Păun, Gh. & Salomaa, A. (eds.), *New trends in formal languages. control, cooperation, and combinatorics* (pp. 288–298). LNCS, 1218. Berlin: Springer.

Kasher, A. (1991). Pragmatics and the modularity of mind. In Davis, S. (ed.), *Pragmatics. A reader* (pp. 567–582). New York: Oxford University Press.

Kathman, D. (1995). Control in autolexical syntax. In Schiller, E., Steinberg, E. & Need, B. (eds.), *Autolexical theory. Ideas and methods* (pp. 103–129). Berlin: Mouton de Gruyter.

Langton, C.G. (ed.) (1989). *Artificial life*. CA: Addison-Wesley.

Mihalache, V. (1996). PC grammar systems with separated alphabets. *Acta Cybernetica, 12* (4), 397–409.

Nii, H.P. (1989). Blackboard systems. In Barr, A., Cohen, P.R. & Feigenbaum, E.A. (eds.), *The handbook of artificial intelligence* (Vol. IV. pp. 3–82); USA: Addison-Wesley.

Nijholt, A. (1988). *Computers and languages. Theory and practice*. North Holland: Elsevier Science Publishers.

Partee, B. (ed.) (1976). *Montague grammar*. New York: Academic Press.

Păun, Gh. (ed.) (1995). *Artificial life: grammatical models*. Bucharest: Black Sea University Press.

Păun, Gh. (ed.) (1998). *Computing with bio-molecules*. Singapore: Springer.

Păun, Gh. (1995). Parallel communicating grammar systems. A survey. In Martín-Vide, C. (ed.), *Lenguajes naturales y lenguajes formales XI* (pp. 257–283). Barcelona: PPU.

Păun, Gh. (1996b). PC grammar systems: recent results, open problems, *Acta Cybernetica, 12,* (4), 381–395.

Păun, Gh. & Sântean, L. (1989). Parallel communicating grammar systems: the regular case, *Annals of the University of Bucharest, Mathematics-Informatics Series, 38*, 55–63.

Pollard, C., & Sag, I. (1994). *Head-driven phrase structure grammar*. Chicago: Chicago University Press.

Rozenberg, G. & Salomaa, A. (eds.) (1997). *Handbook of formal languages*. Berlin: Springer.

Sabah, G. (1997). The fundamental role of pragmatics in natural language understanding and its implications for modular, cognitively motivated architectures. In Black, B. and Bunt, H. (eds.), *Studies in computational pragmatics: abduction, belief, and context*. London: University College Press.

Sadock, J. M. (1985). Autolexical syntax. A proposal for the treatment of noun incorporation and similar phenomena. *Natural language and linguistic theory 3*, 379–439.

Sadock, J. M. (1991). *Autolexical syntax. A theory of parallel grammatical representations*. Chicago: University of Chicago Press.

Salomaa, A. (1973). *Formal languages*. New York: Academic Press.

Schiller, E., Steinberg, E. & Need, B. (eds.) (1995). *Autolexical theory. Ideas and methods*. Berlin: Mouton de Gruyter.

Simon, H.A. (1982). *The sciences of the artificial*. Cambridge: MIT Press.

Smith, G.W. (1991). *Computers and human languages*. New York: Oxford University Press.

Van Valin, R.D. (ed.) (1993). *Advances in role and reference grammar*. Amsterdam: John Benjamins.

Weinberg, A. (1987). Modularity in the syntactic parser. In Garfield, J.L. (ed.), *Modularity in knowledge representation and natural-language understanding* (pp. 259–276). Cambridge: MIT Press.

Werner, E. (1989). Cooperating agents: a unified theory of communication and social structure. In Gasser, L. & Huhns, N. (eds.), *Distributed artificial intelligence* (Vol. 2). London: Pitman.

Wilson, D. & Sperber, D. (1991). Pragmatics and Modularity. In Davis, S. (ed.), *Pragmatics. A reader* (pp. 583–594). Oxford: Oxford University Press.