What is a Natural Language and How to Describe It?

Meaning-Text Approaches in contrast with Generative Approaches

Sylvain Kahane

CNRS & Lattice-Talana, Université Paris 7, UFRL, case 7003,
2, place Jussieu, 75251 Paris Cedex 05, France
sk@ccr.jussieu.fr
http://www.linguist.jussieu.fr/∼skahane

Abstract. The paper expounds the general conceptions of the Meaning-Text theory about what a natural language is and how it must be described. In a second part, a formalization of these conceptions—the transductive grammars—is proposed and compared with generative approaches.[1]

1 Introduction

The Meaning-Text theory (MTT) was put forward in Moscow, thirty-five years ago, by Žolkovski and Mel'čuk ([29], [30]), in the framework of research in machine translation. Presentations of MTT can be found in [20], [21], [25].

MTT considers that a natural language is a correspondence between meanings and texts. Although this conception of language is a more or less accepted by everybody, it appears that most contemporary linguistic theories do not model natural languages in the same ways as MTT. The postulates of MTT will be explained, commented and compared with other conceptions of language in Section 2. In Section 3, I propose a formal definition of what a grammar is in the spirit of MTT, that is, a grammar which defines a correspondence between meanings and texts or, more generally, between any two sets of structures. My definition will by exemplified by a very simple grammar which ensures the correspondence between syntactic and morphological representations. Various definitions of this grammar will be proposed, which allows me to make various comparisons with other formal modelings of natural languages (Sect. 4).

1. I want to thank Kim Gerdes, Alain Polguère and Pascal Amsili for many valuable comments and corrections. I want also to thank Alexander Gelbukh for his suggestions about the topic of this paper.

Invited talk, *Computational Linguistics*, A. Gelbukh (ed.), 2001, Springer, 1-17.

2 What is a Natural Language?

The answer of MTT to the central question—What is a natural language?—is based on the three following postulates.

Postulate 1
Natural language is (considered as) a many-to-many correspondence between meanings and texts.[2]

Postulate 2
The Meaning-Text correspondence is described by a formal device which simulates the linguistic activity of a native speaker.

Postulate 3
Given the complexity of the Meaning-text correspondence, intermediate levels of (utterance) representation have to be distinguished; more precisely, a syntactic and a morphological level.

1) The first postulate of MTT means that the description of a natural language $\mathcal{L}$ consists of the description of the correspondence between the set of meanings of $\mathcal{L}$ and the set of texts of $\mathcal{L}$. This point of view must be compared with the one of Chomsky 1957 ([6]), which has had an enormous influence on linguistics and formal language theory: the description of a natural language $\mathcal{L}$ consists of a formal device deriving the set of all (acceptable) sentences of $\mathcal{L}$. For a long time, his outlook has had a rather restrictive interpretation, a sentence being understood as a string of characters[3]—that is, a text in the MTT terminology—or, in the best case, a sentence being understood as a phrase structure tree. Nevertheless, Chomsky's postulate is formally equivalent to MTT's first postulate, provided a sentence is considered in its Saussurian sense, that is, a linguistic sign with a *signifié* (meaning) and a *signifiant* (text). From a mathematical point of view, it is indeed equivalent to define a correspondence between the set of meanings and the set of texts and to define the set of couples consisting of a meaning and its corresponding text, we can call a sentence.[4]

2) The second postulate stresses on the fact that a natural language must be described as a correspondence. A speaker speaks. A Meaning-Text model must model the speaker activity, that is, model how a speaker transforms what he wants to say (a meaning) into what he says (a text). It is certainly the main specificity of MTT to say that a natural language must be described as a (Meaning-Text) correspondence and moreover that the direction from meaning

2. *Text* refers to any fragment of speech, of whatever length, and *sound* could be a better term.

3. Perhaps, the best example of the restrictive interpretation of Chomsky's works is the definition of the term *formal language*, as a set of string of characters. In this sense, a formal language can never model the essence of a natural language.

4. We forget the fact that the description of a natural language cannot be reduced to the description of isolated sentences.

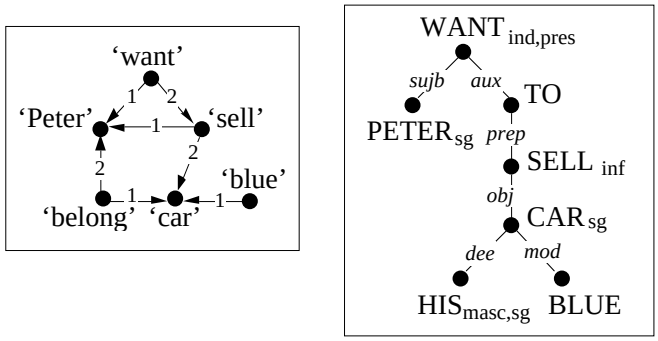
to text must be privileged. This point will be looked at in more details in Section 3.

3) The third postulate asks for several comments. Most of linguistic theories consider a morphological and a syntactic level of representations. What is important here is that these levels are intermediate between the semantic and phonological levels (= meanings and texts). This means that the correspondence from meanings to texts will be completely modular: a correspondence between the semantic and the syntactic level, a correspondence between the syntactic and the morphological level and a correspondence between the morphological and the phonological level (in fact, MTT consider more than two intermediate levels of representations, but this does not change anything to our discussion).

The result is that the syntactic module, which ensures the correspondence between the syntactic representations and the morphological representations, only associates syntactic representations with morphological representations. It does not, as a generative grammar would do, give a complete characterization of the representations it handles. In the synthesis direction, a syntactic module handles syntactic representations which have been synthetized by deeper modules from well-formed semantic representations which represent real meanings. Consequently, a well-formed syntactic representation is characterized by all the modules, by the fact that it is a possible intermediary between a well-formed semantic representation and a corresponding phonological representation. It is not the aim of MTT to give an explicit characterization of well-formed syntactic representations.

I want to insist on the fact that MTT clearly separates the different levels of representation. Representations of different levels have different structural organizations: semantic representations are graphs (of predicate-argument relations), syntactic representations are (non ordered) dependency trees and morphological representations are strings. In the MTT approach, everything that can be differentiated is differentiated. And objects with different organizations must be represented by different means. Moreover, MTT carefully pays attention to the geometry of the representation: a morphological representation is one-dimensional (a string), a syntactic representation is two-dimensional (a tree) and a semantic representation is multi-dimensional (a graph).

One other point should be underlined. MTT uses dependency trees as syntactic representations contrary to most of other linguistic theories, which use phrase structure trees. In fact, ever since the X-bar theory ([13]), the constituents of a phrase structure are considered as projections of lexical heads, and dependency trees and phrase structure trees contain more or less the same information (see [14] for a formal comparison of the two means of representation). Nevertheless, there is a fundamental distinction: a phrase structure contains the linear order of the words of the sentence. In other words, a phrase structure tree does not separate the syntactic structure from the morphological structure. Contemporary theories, such as HPSG ([28]), even mix the semantic representation with the phrase structure representation and use a single formalism—feature structures—to represent all these objects.



PETER$_{sg}$ WANT$_{ind,pres,3,sg}$ TO SELL$_{inf}$ HIS$_{masc,sg}$ BLUE CAR$_{sg}$

Fig. 1. Semantic[5], syntactic and morphological representations of *Peter wants to sell his blue car*

Moreover, the primacy is given to the syntactic structure, that is, the structure of the whole representation—mixing semantic, syntactic, morphological and phonological information—is a phrase structure tree and the geometry of the other structures does not appear explicitly.

I think that, now, thirty-five years after their first description (!), the MTT postulates, even if they are given different formulations, are more or less accepted by the whole linguistic community. For instance, I can quote the first sentences of Brody's Minimalist Program presentation ([5]): "It is a truism that grammar relates sound and meaning. Theories that account for this relationship with reasonable success postulate representational levels corresponding to sound and meaning and assume that the relationship is mediated through complex representations that are composed of smaller units." The main point that is not clearly taken into account by most of the contemporary formal description of natural language is the fact that a natural language must be described as a correspondence. This point will be emphasized now.

3 How to Describe a Language?

To introduce our discussion, let us recall what Mel'čuk says: "The MTM [= Meaning-Text model] is by no means a generative or, for that matter, transformational system; it is a purely EQUATIVE (or TRANSLATIVE) device. The rules of the MTM do not generate (i.e., enumerate, specify) the set of all and only grammatically correct or meaningful texts. They simply match any given SemR

5. The semantic representation of Fig. 1 is far to be complete. In fact, a semantic graph, which indicates the predicate-argument relations between the meanings of the full words of the sentence, cannot be interpreted as a semantic representation without a communicative organization, such as a theme-rheme partition with communicative dominant nodes ([27]).

[= Semantic Representation] with all PhonRs [= Phonological representations] which, in accordance with native speakers' linguistic intuition, can convey the corresponding meaning; inversely, they match any given PhonR with all SemRs that can be expressed by the corresponding text." ([20]:45). He adds: "Un MST [= Modèle Sens-Texte] est purement ÉQUATIF ou TRADUCTIF; à la différence de beaucoup de ses contemporains, ce n'est pas un modèle génératif. [...] Il fait correspondre à chaque SemR toutes les PhonR qui peuvent l'exprimer dans une langue donnée; c'est pourquoi il est qualifié d'"équatif". [...] Un MST essaie de se comporter comme un locuteur, qui ne passe son temps ni à générer des ensembles de phrases grammaticalement correctes ou à distinguer entre phrases correctes et incorrectes, ni à transformer des structures abstraites; un locuteur parle, c'est-à-dire qu'il exprime, au moyen de textes, les sens qu'il veut communiquer. Un MST doit faire la même chose: "traduire" un sens donné en un texte qui l'exprime (voilà pourquoi ce modèle est "traductif")." ([21]:16).

Although sizeable fragments of natural languages have been described in the MTT framework (see [23], [20], [22]), an MTT formalism has never been achieved. Many rules have been written but no directions for use have been proposed explicitly. Mel'čuk justifies this, saying: "The transition mechanism, i.e., the dynamic device, or procedure, for moving from actual complex SemRs to actual complex PhonRs and vice-versa is not considered [by an MTM]. I believe that such a dynamic device, while necessary to put the above static mapping to work, lies outside the field of linguistics, at least as yet. The MTM can be compared to a bilingual dictionary, which presupposes, but does not include, rules looking up the words it contains; then the dynamic device driving the MTM correspondence compares to the psychological ability of a human to use these rules in order to actually look up any given word. It stands to reason that such an ability is not part of the dictionary and should not concern the lexicographer too much." ([20]:45). Indeed, a Meaning-Text model is a grammar of a particular language and the directions for use of such a grammar must be separated from the grammar itself, as it is done in other formalisms. But, the problem with MTT is that this information is nowhere or only implicit.

The goal of this part will be to propose a formalization of the concept of Meaning-Text grammar and to compare this concept with the framework of reference, the generative grammars, and the canonical example of such grammars, context-free grammars.

3.1 Transductive Grammars and Supercorrespondence

In this section, I will propose a very general formal definition of what a grammar is in the spirit of MTT. Such a grammar will be called a transductive grammar, by analogy with transducer (see, for instance, [1]) (although, as far as I know, the transducer theory is limited to the correspondence between strings).

Let $\mathcal{S}$ and $\mathcal{S}'$ be two sets of structures (graphs, trees, orders ...). A *transductive grammar* G between $\mathcal{S}$ and $\mathcal{S}'$ is a formal grammar which associates elements of $\mathcal{S}$ with elements of $\mathcal{S}'$. As a formal grammar, G contains a finite set of rules, which are called the *correspondence rules*. A correspondence rule

associates a piece of structure from elements of $\mathcal{S}$ with a piece of structure from elements of $\mathcal{S}'$. Consequently, a transductive grammar G defines *more* than a correspondence between the sets of structures $\mathcal{S}$ and $\mathcal{S}'$. Indeed, for each couple (S, S') that are associated by G, G also defines partitions of the structures S and S' and a one-to-one mapping $\varphi_{(S,S')}$ between the pieces of these two partitions. This will be called a *supercorrespondence* between the sets $\mathcal{S}$ and $\mathcal{S}'$. The supercorrespondence defined by a transductive grammar G between two sets of structures $\mathcal{S}$ and $\mathcal{S}'$ is mathematically equivalent to a family of product structures $(S, S', \varphi_{(S,S')})$, with $S \in \mathcal{S}$, $S' \in \mathcal{S}'$ and $\varphi_{(S,S')}$ a correspondence between the pieces of partitions of S and S'.[6]

We see now that the first postulate of MTT has not been well enounced. A natural language is more than a correspondence between meanings and texts, that is, a set of couples meaning-text. A natural language is a supercorrespondence between meanings and texts, that is, a set of product structures meaning-text. And similarly, a sentence is not a couple meaning-text or *signifié-signifiant*, but a product structure, each piece of the meaning being related to a piece of the text. I am just giving a new expression of the well-known notion of *compositionality*: a sequence is a sign that can be decomposed into smaller signs.

3.2 Example of Syntactic Transductive Grammar

We will now focus our discussion on a particular module of a Meaning-Text model. We have chosen the syntactic module, because it is the module which receives the biggest attention in most of the natural language models.

The MTT syntactic module ensures the correspondence between syntactic and morphological representations. A *syntactic representation* is a *non ordered* dependency tree (assorted with other pieces of information, such as the theme-rheme partition ...). The nodes of a syntactic tree are labeled by lexical units[7] and the branches are labeled by *syntactic relations* (*subj(ect)*, *obj(ect)*, *mod(ifier)* ...). A *morphological representation* is a linearly ordered string of lexical units, that is, a *linear order* on a set of lexical units (assorted with other pieces of information such as prosody ...). Each lexical unit points to a dictionary entry. In order to simplify, only the part of speech will be considered and it will be added on the node labeling.

All our notions will be exemplified with the following trivial example:

(1) *Peter eats red beans.*
Peter$_{(N)sg}$ eat$_{(V)ind,pres,3,sg}$ red$_{(A)}$ bean$_{(N)pl}$

6. In mathematics, a *product structure* is a structure obtained by combining two structures on a same set. For instance, if S is a tree and S' is a string and if $\varphi_{(S,S')}$ is one-to-one mapping between the nodes of S and the elements of S', then $(S, S', \varphi_{(S,S')})$ is equivalent to a linearly ordered tree, that is, to the product of tree structure and and a linear order structure on a same set of nodes.

7. In fact, each lexical unit is accompanied by grammemes [= inflections], but our presentation is oversimplified in order to focus only on our topic, the comparison between transductive and generative approaches.

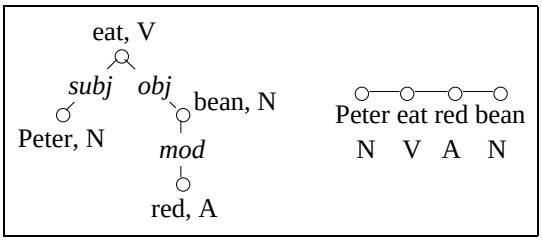


Fig. 2. (Simplified) syntactic tree and morphological string of (1)

We will now define a family of syntactic transductive grammars, which we will call atomic dependency grammars. These grammars are atomic because they associate only atoms of structures, that is, nodes and edges. Two kinds of rules are considered: *sagittal rules* (Lat. *sagitta* 'arrow'), which associate a dependency between two nodes with an order relation between two nodes, and *nodal rules*, which associate a node with a node. The nodal rules are trivial here and do not appear in the formal definition. (They will be introduced in the generative definition, cf. Section 4.)

An *atomic dependency grammar* is a 5-tuple $G = (\Sigma, \mathcal{C}, \mathcal{R}, \mathcal{O}, \Delta)$, where Σ is the set of lexical units, $\mathcal{C}$ is the set of (grammatical) categories, $\mathcal{R}$ is the set of syntactic relations, $\mathcal{O}$ is the set of linear positions and Δ is the set of sagittal rules, that is, a subset of $\mathcal{R} \times \mathcal{O} \times \mathcal{C} \times \mathcal{C}$.

Let X^* be the set of strings on X and $\mathcal{T}(X, Y)$ be the set of trees whose nodes are labeled in X and whose branches are labeled in Y. The grammar G defines a supercorrespondence between $\mathcal{T}(\Sigma \times \mathcal{C}, \mathcal{R})$ and $(\Sigma \times \mathcal{C})$, as it will be seen in the following sections. Before that, I will exemplify my definition by a grammar which ensures the correspondence between the tree and the string of Fig. 2.

Let us consider $G_0 = (\Sigma_0, \mathcal{C}_0, \mathcal{R}_0, \mathcal{O}_0, \Delta_0)$ with:
- $\Sigma_0 = \{\text{Peter}, \text{bean}, \text{eat}, \text{red}\}$;
- $\mathcal{C}_0 = \{V, N, A\}$;[8]
- $\mathcal{R}_0 = \{subj, obj, mod\}$;
- $\mathcal{O}_0 = \{<, >\}$;[9]

8. The set of categories considered here is very simple (and limited to the part of speech). A real size grammar for natural language must consider richer categories. Such categories can be expressed with feature structures. In this case, as it is done in most of contemporary formalisms, categories must be combined by unification rather than identity.

9. An atomic dependency grammar with $\mathcal{O} = \{>, <\}$ only permits to specify the position of a node towards its governor—before ($<$) or after ($>$)—and not the relative position of codependents of a node. This problem can be solved by indicating the relative distance of the different dependents of a node towards it (see solutions with $\mathcal{O} \subset \mathbf{Z}$, the set of integers, in [19], [7], [15]).

- $\Delta_0 = \{(subj, <, V, N), (obj, >, V, N), (mod, <, N, A)\}$.

The sagittal rule $(subj, <, V, N)$ says that, for all lexical units X and Y such that X is a V(erb) and Y is a N(oun), the syntactic dependency $X - subj \rightarrow Y$ *corresponds* to the linear order $Y < X$. In the synthesis direction, this means that for all lexical units X and Y such that X is a V and Y is an N, IF $X - subj \rightarrow Y$, THEN it is POSSIBLE to have $Y < X$. And in the analysis direction, it means that for all lexical units X and Y such that X is a V and Y is an N, IF $Y < X$, THEN it is POSSIBLE to have $X - subj \rightarrow Y$. See Fig. 3 for the conventional representation of such a rule in MTT. The symbol $\Leftrightarrow$ indicates the correspondence between two pieces of structures; $\alpha \Leftrightarrow \beta$ is interpreted by $\alpha \Rightarrow \beta$ and $\beta \Rightarrow \alpha$, where $\alpha \Rightarrow \beta$ means IF α, THEN β IS POSSIBLE.

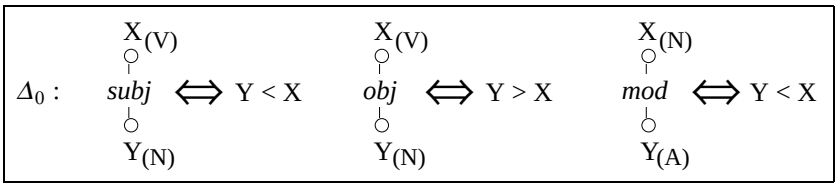


Fig. 3. The rules of Δ_0 in the MTT style

We will see now how a supercorrespondence can be defined. Different strategies are possible. We will study several solutions, in this and the following section.

But first, a remark: the correspondence rules, which only associate pieces of structures, are generally not sufficient to encode all the properties of the product structure, and some *global rules* must be stated. This is the case here, for the syntactic-morphology correspondence, where it is necessary to ensure a property such as projectivity for the product structures.[10] The projectivity is a property of compatibility between a tree and a linear order, that is a property of a linearly ordered tree. A linearly ordered tree is said to be *projective* if no branch crosses another branch and no branch covers the root ([17], [11]).

3.3 Transductive Presentation in the Synthesis Direction

As we have said, an atomic dependency grammar $G = (\Sigma, \mathcal{C}, \mathcal{R}, \mathcal{O}, \Delta)$ defines a supercorrespondence between the trees of $\mathcal{T}(\Sigma \times \mathcal{C}, \mathcal{R})$ and the strings of $(\Sigma \times \mathcal{C})^*$. In the synthesis direction, an atomic dependency grammar ensures the linearization of trees. The synthesis starts with a given tree $T \in \mathcal{T}(\Sigma \times \mathcal{C}, \mathcal{R})$.

10. It is well-known that many linguistic constructions are not projective (unbounded extractions, German scrambling, Dutch cross serial dependencies, Czech clitics ...). Nevertheless, even in these cases, the word order is far to be free and it must be controlled by some global property, weaker than the projectivity. Note that the projectivity is also considered in the phrase structure frameworks where it corresponds to the continuity of the constituents.

A derivation[11] processes is as follows. For each branch of T labeled r whose governor is of category C_1 and whose dependent is of category C_2, a sagittal rule $(r, \omega, C_1, C_2) \in \Delta$ is triggered off and the label $\omega \in \{>, <\}$ is attached to the branch. A string $s = X_1 \dots X_n \in \Sigma^*$ corresponds to T if T has exactly n nodes labeled $X_1, \dots, X_n$ and if for each dependency $X - r \rightarrow Y$ of T the label ω attached by the sagittal rule is compatible with the order of X and Y in s, that is, $Y < X$ if $\omega = <$ and $Y > X$ if $\omega = >$. Moreover, the ordered tree $T \times s$ must be projective. See an example of derivation Fig. 4.

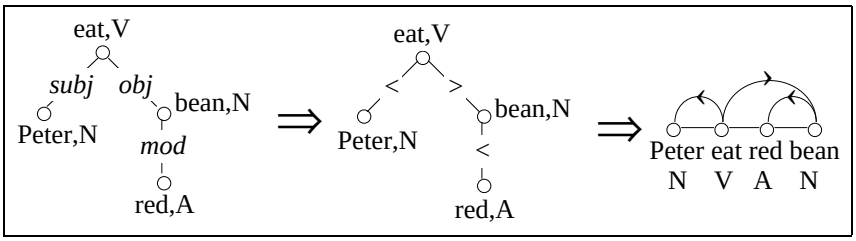


Fig. 4. G_0 used as transductive grammar in the synthesis direction

The process fails if no rules can apply to an element of T. Several orders can be obtained with one derivation, in particular if some codependents are positioned on the same side of their governor, their respective order being free in this case. To obtain all the strings corresponding to a tree T, all the combinations of rules must be tried.

The process proposed here is quite declarative, no order being proposed in the application of the different rules. In fact, the order is free and the derivation could be top-down, in the fashion of the context-free rewriting grammar, or incrementally, following the linear order of the nodes (cf. Section 4).

3.4 A Transductive Grammar in the Analysis Direction

The analysis starts with a given string $s = X_1 \dots X_n \in \Sigma^*$. The derivation processes as follows. For each couple of nodes (X, X') of categories (C, C') with $X < X'$, a sagittal rule $(r, >, C, C')$ or $(r, <, C', C) \in \Delta$ is triggered off and a branch labeled r from X to X' or from X' to X is introduced. A tree T corresponds to s if T has exactly n nodes labeled $X_1, \dots, X_n$ corresponding to the nodes of s and if each branch of T corresponds to one of the branches added to s of same extremities and same label. Moreover, the ordered tree $T \times s$ must be projective. See an example of derivation Fig. 5.

Note that the derivation does not need that a sagittal rule can be triggered off for each couple of nodes of s. Conversely, even if more than $n-1$ sagittal rules apply, it is not sure that a tree can be extracted from the graph obtained by the

11. Although the process described here is not generative, I prefer using the term *derivation* of the generative framework rather introducing a new term.

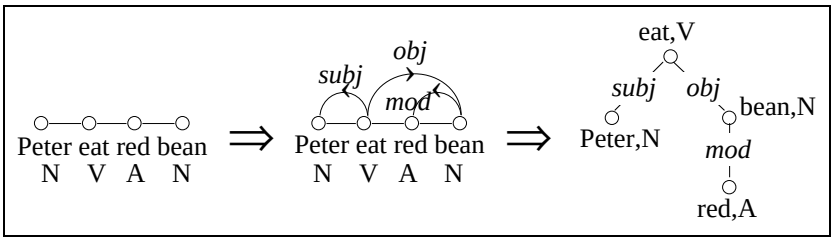


Fig. 5. G_0 used as transductive grammar
in the analysis direction

application of the rules. Different algorithms permit to extract a projective tree from such a graph; see, for instance, [2], [3].[12]

The process proposed here is still quite declarative. It is possible to constrain the derivation further. For instance, it is possible to produce the tree by reading the string from the left to the right. In this case, the grammar acts as a push-down automaton. We will see that more precisely later. An atomic dependency grammar $G = (\Sigma, \mathcal{C}, \mathcal{F}, \mathcal{O}, \Delta)$ can be associated to a push-down automaton $M = (\Sigma, \mathcal{Z}, \Lambda)$ with a stack alphabet $\mathcal{Z} = \mathcal{C} \times \{+, -\} \times \mathbf{N}$ and a set of transitions $\Lambda \subset \mathcal{U} \times \mathcal{Z}^* \times \mathcal{Z}^* \times \mathcal{W}$, where $\mathcal{U} = \{\varepsilon\} \cup (\Sigma \times \mathcal{C})$ and $\mathcal{W} = (\Sigma \times \mathcal{C} \times \mathbf{N}) \cup (\mathbf{N} \times \mathbf{N} \times \mathcal{R}))$. A tree in $\mathcal{T}(\Sigma \times \mathcal{C}, \mathcal{R})$ can be encoded as a subset of $\mathcal{W}$: a triple $(X, C, i) \in \Sigma \times \mathcal{C} \times \mathbf{N}$ means that the i-th node of the tree is labeled (X, C) and a triple $(i, j, r) \in \mathbf{N} \times \mathbf{N} \times \mathcal{F}$ means that the i-th node governs the j-th node one by a dependency labeled r.

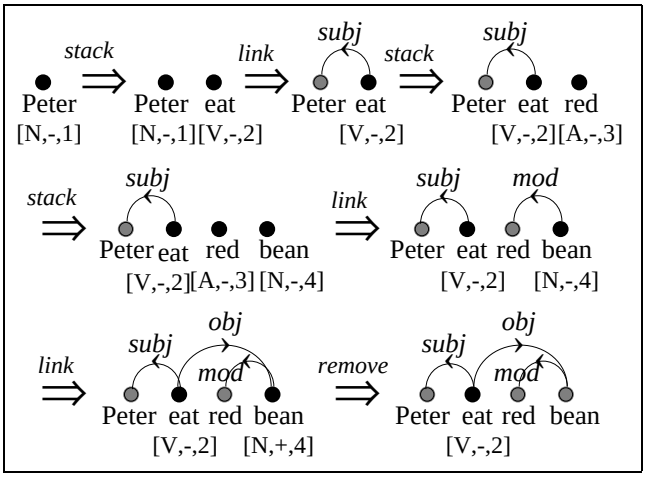


Fig. 6. G_0 used as a push-down automaton

The idea is to stack the nodes which are produced and to remove from the stack the nodes that can no longer have links with other nodes (see [18] for a

12. The grammar considered by Blache in [3] is a context-free phrase structure grammar, but this grammar is in fact used exactly in the present way.

similar idea). The signs + and − of stack symbols mean that the corresponding node is or is not yet governed. A transition $\lambda = (x, \alpha, \beta, y) \in \Lambda$ is interpreted as follows: the element x of s is read (if nothing is read, $x = \varepsilon$), the string α of stack symbols is removed from the stack, the string β of stack symbols is stacked and the tree description y is produced. Λ is build as follows:

- each couple $(X, C) \in \Sigma \times \mathcal{C}$ yields a *stacking transition* $((X, C), \varepsilon, [C, -, i], (X, C, i))$ reading the i-th node (X, C) of the string, stacking $[C, -, i]$ and producing the node of the tree (X, C, i);

- each sagittal rule $(r, <, C_1, C_2) \in \Delta$ yields a *left dependent linking transition* $(\varepsilon, [C_2, -, i][C_1, -, j], [C_1, -, j], (j, i, r))$ producing the dependency (j, i, r) and removing $[C_2, -, i]$ from the stack, because, due to the projectivity, (X_2, C_2, i) cannot be linked to a node at the right of its governor (X_1, C_1, j);

- each sagittal rule $(r, >, C_1, C_2) \in \Delta$ yields a *left governor linking transition* $(\varepsilon, [C_1, \pm, i][C_2, -, j], [C_1, \pm, i][C_2, +, j], (j, i, r))$ producing the dependency (j, i, r) and replacing $[C_2, -, j]$ by $[C_2, +, j]$ in the stack, because the node (X_2, C_2, j) is now governed;

- at last, each $C \in \mathcal{C}$ yields a *removing transition* $(\varepsilon, [C, +, i], \varepsilon, \varepsilon)$ removing $[C, +, i]$ from the stack, which is possible, because the i-th node is governed.

The stack is empty at the beginning. The process can only stop when the stack content is reduced to a slot $[C, -, i]$. This ensures that there is only one non governed node, that is, that we have produced a tree, whose i-th node is the root. Moreover, our process ensures that no links can cross each other and that the root is not covered by a link, i.e., the tree is projective. Fig. 6 presents the parsing of (1) by the automaton associated to the grammar G_0. A black node represents a node in the stack and a grey node, one that has been removed from the stack. Note that the indexes i in the stack symbols are only useful for the production of the tree corresponding to the read string.

4 Transductive Grammars and Generative Grammars

In the narrow sense, a generative grammar is a formal device which allows to generate a formal language, i.e., a set of linearly ordered string. I will call *generative grammar* every formal device which allows to generate a set of structures, strings being a particular case. In this wide sense of the notion of generative grammars, a transductive grammar can be seen as a generative grammar, as we will see.

4.1 Transductive Grammars as Generative Grammars

Remember that the supercorrespondence defined by a transductive grammar G between two sets of structures $\mathcal{S}$ and $\mathcal{S}'$ is mathematically equivalent to a set of product structures $(S, S', \varphi_{(S,S')})$, with $S \in \mathcal{S}$, $S' \in \mathcal{S}'$ and $\varphi_{(S,S')}$ a correspondence between the pieces of partitions of S and S'. Therefore, a transductive grammar can be seen as a generative grammar generating a supercorrespondence. For instance, an atomic dependency grammar G defines a set of linearly ordered trees, that is, products of a tree and a linear order on a

same set of nodes. In a generative interpretation of a transductive grammar, the correspondence rules are seen as pieces of product structures which are assembled to generate the product structures.

Let us see more precisely how an atomic dependency grammar can be used as a generative grammar. Each rule is seen as a piece of a linearly ordered tree. The sagittal rules introduce ordered dependencies. The nodes will be introduced by the nodal rules, which were not considered until now, because they trivially associate a node with a node of same label. We use the conventions of representation of [26]: the nodes, which are really introduced by the rules are in black and the requirements are in white. If we use the terminology of rewriting systems, we can say that black elements are terminal and white elements are non terminal. The rules are combined by unification of nodes: two black elements cannot unify, a black and a white node yields a black node and two white nodes yields a white node. The final structure must be entirely black. A derivation simply consists generating a set of rules and combining them, provided that the resulting structure is a projective linearly ordered tree.

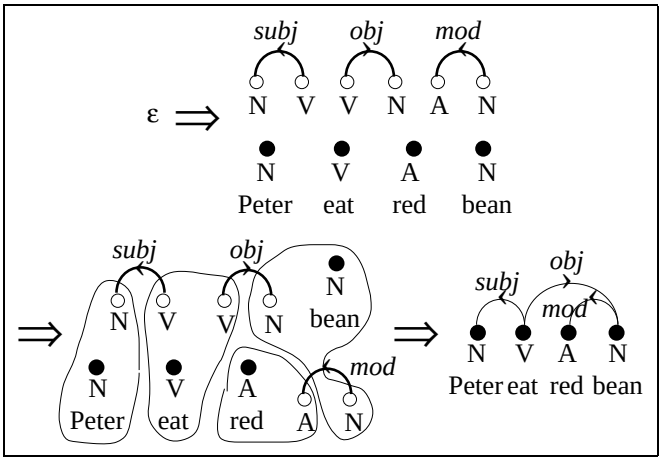


Fig. 7. G_0 used as generative grammar

Although it is not necessary, it is possible to specify an order on the operations of derivation. For instance, we can follow the tree structure and process top-down: a first nodal rule is triggered off generating the root of the tree; after that, a sagittal rule whose governor can unify with the root is triggered off, then a nodal rule that can unify with the requirement of the previous rule and so on (Fig. 8). Cf. [8] for dependency grammars of this kind.

An atomic dependency grammar used as a tree-driven generative grammar is very similar to a context-free rewriting system. This similarity can be brought to the fore: the grammar G_0 can be simulated by the rewriting system $G'_0 = (\Sigma_T, \Sigma_{NT}, \bar{V}, \mathcal{R})$, with the terminal alphabet $\Sigma_T = \Sigma_0 \cup \mathcal{R}_0 \cup \{(,),:\}$, the non terminal alphabet $\Sigma_{NT} = \bar{\mathcal{C}}_0 = \{\bar{V}, \bar{N}, \bar{A}\}$ and the set of rewriting rules $\mathcal{R}$:

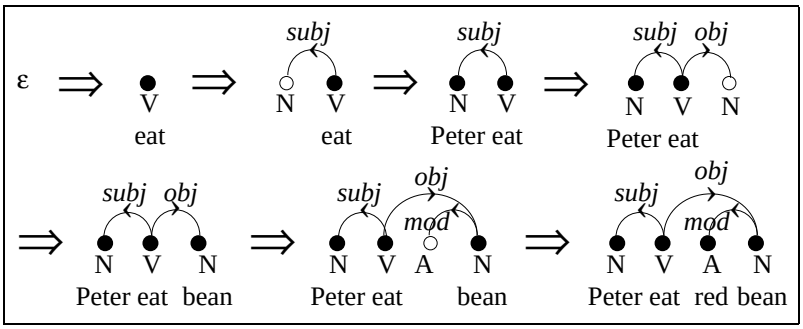


Fig. 8. G_0 used as tree-driven generative grammar

$$\bar{V} \rightarrow (\bar{N} : subj)\bar{V} \qquad \bar{V} \rightarrow \bar{V}(\bar{N} : obj) \qquad \bar{N} \rightarrow (\bar{A} : mod)\bar{N}$$
$$\bar{V} \rightarrow (\text{eat}, V) \quad \bar{N} \rightarrow (\text{Peter}, N) \quad \bar{N} \rightarrow (\text{bean}, N) \quad \bar{A} \rightarrow (\text{red}, A)$$

We can derive, beginning with $\bar{V}$, a sequence which encodes the product of the structures of Fig. 2: $((\text{Peter}, N) : subj)(\text{eat}, V)(((\text{red}, A) : mod)(\text{bean}, N) : obj)$. This derivation is very similar to the derivation of Fig. 8. In particular, a non terminal symbol $\bar{X}$ is similar to the label of a white node, while a terminal symbol X is similar to the label of a black node.

4.2 Generative Grammars as Transductive Grammars

Contrary to the fact that a transductive grammar can be seen as a generative grammar, I think that most of the generative grammars used for modeling natural languages can be seen as transductive grammars.

Let us consider the basic case of context-free grammars. Such grammars generate a set of sentences, but associate to each derivation of a sentence a derivation tree, which is interpreted as the syntactic structure of the sentence. In other words, a context-free grammar can be seen as a transductive grammar that defines a correspondence between sentences and constituency trees. Nevertheless, this is not a correspondence in the MTT sense, because the constituency tree contains both the syntactic structure of the sentence and the sentence itself (with its linear order).[13] However, a context-free grammar can also be seen as a transductive grammar that defines a correspondence between sentences and non ordered constituency trees. In this case, the context-free grammar is not far from an atomic dependency grammar (see [9] for conditions on context-free grammars to be strongly equivalent to a dependency grammar). It is the case of the context-free grammar G'_0 which simulates the atomic dependency grammar

13. Many linguists think in the analysis direction and they do not find any problem to associate a string of words to a richer structure containing the string of words with more information. But such a view on language is absurd as soon as you are thinking in the synthesis direction. When you want to express a meaning, you cannot suppose that you already known the word order. If it were the case, there would be nothing to compute!

G_0 (Section 4.1). But as we have seen, G'_0 corresponds to a particular procedural implementation of the transductive grammar G_0, where the generation is tree-driven generation. This can be generalized: context-free grammars can be seen as transductive grammars including a tree-driven procedure of generation (which can be unuseful and constraining in some cases, for instance when we want to use the grammar for analysis).

More recent models of language also define product structures, like LFG (morphological string + *c*-structure + *f*-structure), TAG (morphological string + derived tree + derivation tree), HPSG (morphological string + syntactic-semantic feature structure) ... It is not the case of whatever generative grammar; for instance, it seems difficult to interpret a Turing machine as a transductive grammar, because it is not clear which structure associates with a string generated by the machine.

I nevertheless must insist on the fact that even the generative grammars that can be interpreted as transductive grammars do not clearly consider two levels of representations that are in correspondence and the fact that they define a (super)correspondence is in fact a side effect. For instance, for the rewriting grammars, it is the process of derivation itself which is interpreted as a representation and not the result of the derivation (although this vision has evolved with tree-rewriting system such as TAG).

I will finish this discussion on transductive and generative grammars with a point which seems very important to me, even if I miss elements to support it. I think that the notion of *strong generative capacity* is related to the fact that the grammars devoted to the description of natural language are in fact hidden transductive grammars. It can be considered that two grammars are *strongly equivalent* if they are equivalent as transductive grammars, that is, if they generate the same supercorrespondence.

4.3 Equative Grammars

To conclude my presentation I present a third way of defining a supercorrespondence with a transductive grammar, in a complementary way to transductive and generative ways. As we have seen, the main difference between a transductive and a generative presentation of a transductive grammar G is that the former presupposes that one of the two sets of corresponding structures is available (G producing the other one), while the latter considers that G produces directly both ones.

The third way of defining the supercorrespondence, the *equative* approach, presupposes that both sets of structures are available: the process consists to filter all the couples of structures which correspond to each other. The correspondence rules are used as a set of equations or constraints to be met (as does, for instance, HPSG). From a purely mathematical viewpoint, this is the simplest definition of the correspondence and it is also the least procedural one: a product structure is given and each piece of the structure must be validated by a correspondence rule. For instance, we can verify that the two structures of Fig. 2 are associated with our atomic dependency grammar G_0; for that, we

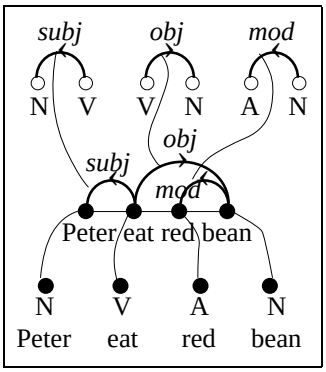


Fig. 9. G_0 as an equative grammar

must verify that each element (node or branch) of the product structure (the linearly ordered tree) is validated by a correspondence rule (Fig. 9.

In other words, the equative presentation is the *declarative* presentation of a transductive grammar, i.e., a grammar relating two set of structures. Nevertheless, such a presentation is not very useful for computational applications, because, in such cases, meaning and text are never available together. And, in fact, another device is needed which generates couples of structures, the equative grammar being only used to filter the couples of structures that really correspond to each other.

5 Conclusion

There is now a consensus about the fact that grammars describing natural languages must relate meaning and text. However, few formalisms describe languages as correspondences between meanings and texts explicitly. For that, I have proposed a general mathematical notion of grammars defining a (super)correspondence between two sets of structures.

I have focused my discussion on an elementary family of transductive grammars, the atomic dependency grammars, defining a supercorrespondence between dependency trees and linear strings. I think that these grammars are the simplest grammars to define such a supercorrespondence, because they handle only atoms of structure: nodes and edges. Clearly, this formalism does not permit to describe all the linguistic phenomena and various extensions are possible (cf., for instance, [15]). This simple formalism has been only introduced in order to show that, even with a very simple grammar defining a very little fragment of language, there is many ways to define the language from the grammar. (It appears that atomic dependency grammars may also be a formalism of reference for dependency grammars, just as context-free grammars are a formalism of reference for phrase structure grammars.)

If you accept that a language is a correspondence between meanings and texts, only three directions are possible to describe a language with a grammar:

1. to produce directly couples of corresponding structures (generative approaches);
2. to produce structures corresponding to a given structure (transductive approaches);
3. to filter couples of corresponding structures (equative or declarative approaches).

From a computational point of view, as well as, from a cognitive point of view, we need transductive approaches, i.e., we need to get a meaning from a text (analysis or reading) or to get a text from a meaning (synthesis or speaking).

References

1. Aho Alfred & Ullman Jeffrey, 1972, *The Theory of Parsing, Translation and Compiling, Vol. I: Parsing*, London: Prentice-Hall.
2. Arnola Harri, 1998, "On parsing binary dependency structures deterministically in linear time", in Kahane & Polguère (eds), *Workshop on dependency-based grammars, COLING-ACL'98*, Montreal, 68-77.
3. Blache Philippe, 1998, "Parsing ambiguous structures using controlled disjunctions and unary quasi-trees", *Proc. COLING-ACL'98*, Montreal, 124-130.
4. Boyer Michel & Lapalme Guy, 1985, "Generating paraphrases from Meaning-Text semantic networks", *Comput. Intell.*, *1*, 103-117.
5. Brody Michael, 1997, *Lexico-Logical Form: A Radically Minimalist Theory*, Cambridge: MIT Press.
6. Chomsky Noam, 1957, *Syntactic Structure*, Cambridge: MIT Press.
7. Courtin Jacques & Genthial Damien, 1998, "Parsing with Dependency Relations and Robust Parsing", in Kahane & Polguère (eds), *Workshop on dependency-based grammars, COLING-ACL'98*, Montreal, 95-101.
8. Dikovsky Alexander & Modina Larissa, 2000, "Dependencies on the other side of the Curtain", in S. Kahane (ed.), *Grammaires de dépendance*, *T.A.L.*, *41:1.*
9. Gaifman Haïm, 1965, "Dependency systems and phrase-structure systems", *Information and Control* ***8***, 304-337; Rand Corporation Techn. Report RM-2315, 1961.
10. Gladkij Aleksej & Mel'čuk Igor, 1975, "Tree grammars: A Formalism for Syntactic Transformations in Natural Languages", *Linguistics, 150*, 47-82.
11. Iordanskaja Lidija, 1963, "O nekotoryx svojstvax pravil'noj sintaksičeskoj struktury (na materiale russkogo jazyka)" [On some properties of the correct syntactic structure (on the basis of Russian)], *Voprosy jazykoznanija*, *4*, 102-12.
12. Iordanskaja Lidija & Polguère Alain, 1988, "Semantic processing for text generation", in *Proc. First International Computer Science Conf. - 88*, Hong Kong, 310-18.
13. Jackendoff Ray S., 1977, *X Syntax : A Study of Phrase Structure*, Linguistic Inquiry Monograph, MIT Press.
14. Kahane Sylvain, 1997, "Bubble trees and syntactic representations", in Becker & Krieger (eds), *Proc. 5th Meeting of the Mathematics of Language (MOL5)*, Saarbrücken: DFKI.
15. Kahane Sylvain, 2001, "A fully lexicalized grammar for French based on the Meaning-Text theory", this issue.

16. Kahane Sylvain & Mel'čuk Igor, 1999, "Synthèse des phrases à extraction en français contemporain (Du graphe sémantique à l'arbre de dépendance)", *T.A.L.*, *40:2*, 25-85.
17. Lecerf Yves, 1961, "Une représentation algébrique de la structure des phrases dans diverses langues naturelles", *C. R. Acad. Sc. Paris, 252*, 232-234.
18. Kornaï Andreás & Tuza Zsolt, 1992, "Narrowness, pathwidth, and their application in natural language processing", *Disc. Appl. Math, 36*, 87-92.
19. Mel'čuk Igor, 1967, "Ordre des mots en synthèse automatique des textes russes", *T.A. Informations*, *8:2*, 65-84.
20. Mel'čuk Igor, 1988, *Dependency Syntax: Theory and Practice*, Albany, NY: State Univ. of New York Press.
21. Mel'čuk Igor, 1997, *Vers une Linguistique Sens-Texte*, Leçon inaugurale au Collège de France, Paris: Collège de France.
22. Mel'čuk Igor, 1993-2000, *Cours de morphologie générale*, *Vol. 1, 2, 3, 4 & 5*, Montreal: Presses de l'Univ. Montreal / Paris: CNRS.
23. Mel'čuk Igor & Pertsov Nikolaj, 1987, *Surface Syntax of English. A Formal Model within the Meaning-Text Framework*, Amsterdam: Benjamins.
24. Mel'čuk Igor *et al.*, 1984, 1988, 1992, 1999, *Dictionnnaire explicatif et combinatoire du français contemporain*, *Vol. 1, 2, 3, 4*, Montreal: Presses de l'Univ. Montreal.
25. Milićević Jasmina, 2001, "A short guide to the Meaning-Text linguistic theory", in Alexander Gelbukh (ed.), *Computational Linguistics and Intelligent Text Processing*, *Colección en Ciencias de Computación*, Fondo de Cultura Económica - IPN - UNAM, Mexico.
26. Nasr Alexis, 1995, "A formalism and a parser for lexicalised dependency grammars", *4th Int. Workshop on Parsing Technologies*, State Univ. of NY Press.
27. Polguère Alain, 1997, "Meaning-Text Semantic Networks as a Formal Language", in L. Wanner (ed.), *Recent Trends in Meaning-Text Theory*, Amsterdam/Philadelphia: Benjamins.
28. Pollard Carl & Sag Ivan A., 1994, *Head-Driven Phrase Structure Grammar*, CSLI series, Chicago : Univ. of Chicago Press.
29. Žolkovskij Aleksandr & Mel'čuk Igor, 1965, "O vozmožnom metode i instrumentax semantičeskogo sinteza" [On a possible method an instruments for semantic synthesis (of texts)], *Naučno-texničeskaja informacija* [Scientific and Technological Information], 6, 23-28.
30. Žolkovskij Aleksandr & Mel'čuk Igor, 1967, "O semantičeskom sinteze" [On semantic synthesis (of texts)], *Problemy kybernetiki* [Problems of Cybernetics], 19, 177-238. [Fr. transl. : 1970, *T.A. Information, 2*, 1-85.]