A Formal Lexicon in the Meaning-Text Theory (or How to Do Lexica with Words)

Igor A. Mel'čuk

Département de linguistique
Université de Montréal
C.P. 6128, Succ. "A"
Montréal, P.Q. H3C 3J7 Canada

Alain Polguère

Odyssey Research Associates
Place Outremont, 1290 Van Horne
Montréal, P.Q. H2V 1K6 Canada

The goal of this paper is to present a particular type of lexicon, elaborated within a formal theory of natural language called Meaning-Text Theory (MTT). This theory puts strong emphasis on the development of highly structured lexica. Computational linguistics does of course recognize the importance of the lexicon in language processing. However, MTT probably goes further in this direction than various well-known approaches within computational linguistics; it assigns to the lexicon a central place, so that the rest of linguistic description is supposed to pivot around the lexicon. It is in this spirit that MTT views the model of natural language: the Meaning-Text Model, or MTM. It is believed that a very rich lexicon presenting individual information about lexemes in a consistent and detailed way facilitates the general task of computational linguistics by dividing it into two more or less autonomous subtasks: a linguistic and a computational one. The MTM lexicon, embodying a vast amount of linguistic information, can be used in different computational applications.

We will present here a short outline of the lexicon in question as well as of its interaction with other components of the MTM, with special attention to computational implications of the Meaning-Text Theory.

1. Levels of utterance representation in Meaning-Text Theory and the Meaning-Text Model of natural language.

The goal of the present paper is two-fold:

1) To present a specific viewpoint on the role of lexica in "intelligent" systems designed to process texts in natural language and based on access to meaning.

2) To present a specific format for such a lexicon — so-called Explanatory Combinatorial Dictionary (ECD).

We believe that a rich enough lexicon, which could enable us to solve the major problem of computational linguistics — that of presenting all necessary information about natural language in compact form, should be anchored in a formal and comprehensive theory of language. The lexicon to be discussed, that is ECD, has been conceived and developed within the framework of a particular linguistic theory — more specifically, Meaning-Text Theory or MTT (Mel'čuk 1974, 1981, 1988:43-101). Note that this is by no means a theory of how linguistic knowledge could or should be applied in the context of any computational task. The MTT is a theory of how to describe and formally present linguistic knowledge, a theory of linguistic description; therefore, its contribution to computational linguistics is only a partial one: to take care exclusively of the linguistic part of the general endeavor.

We cannot present here the Meaning-Text Theory in detail, so we will limit ourselves to a brief characterization of the following two aspects, which are of

0362-613X/87/030261-275$03.00

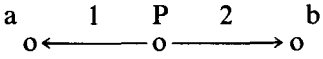
particular relevance to this paper: the system of linguistic representations the theory makes use of, and the linguistic Meaning-Text model which it presupposes.

1.1. UTTERANCE REPRESENTATIONS IN THE MTT.

We have to warn our reader that limitations of space force us to have recourse to drastic simplifications, perhaps even too drastic sometimes. Thus, an MTT utterance representation is in fact a set of formal objects called *structures*, their vocation being the characterization of separate aspects of the phenomena to be described. But in this paper we use the term *representation* to refer to the main one of the structures which compose a representation (since we disregard the other structures).

In Meaning-Text Theory, an utterance[1] is represented at seven levels:

1) The Sem(antic) R(epresentation) of utterance U is, roughly speaking, a network which depicts the linguistic meaning of U without taking into consideration the way this meaning is expressed in U (distribution of meaning between words and constructions, and the like). Thus a SemR represents in fact the meaning of the whole family of utterances synonymous with each other.[2] The nodes of a SemR network are labeled with semantic units, a semantic unit being a specific sense of a lexeme in the language in question. The arcs of the network are labeled with distinctive numbers which identify different arguments of a predicate. Thus,



is equivalent to the more familiar notation P(a , b). As the reader can see, our SemR is based on predicate-argument relations (although we do not use the linear notation of predicate calculus nor predicate calculus as such).

2) The D(eep-)Synt(actic) R(epresentation) of U is, roughly speaking, a dependency tree whose nodes are not linearly ordered (because linear order is taken to be a means of expressing syntactic structure rather than being part of it). The nodes of a DSyntR are labeled with meaningful lexemes of U, which are supplied with meaning-bearing morphological values (such as number in nouns or tense in verbs; in our example below, we indicate such values for the top node only). The branches of a DSyntR carry the names of universal DSynt-relations, which are few in number (less than ten).

3) The S(urface-)Synt(actic) R(epresentation) of U is also a dependency tree of the same formal type but, its nodes are labeled with all actual lexemic occurrences of U (including all structural words), and branches of it carry the names of a few dozen specific SSynt-relations, which correspond to the actual syntactic constructions of a particular language.

The distinction between Deep- and Surface- sublevels is related to the fact that some syntactic phenomena (basically, cooccurrence restrictions) are more linked to meaning, while other syntactic phenomena (word order, agreement, and the like) are more relevant from the viewpoint of actual text. Phenomena of the first type are captured in the DSyntR, which is geared to meaning, and those of the second type, in the SSyntR, geared to text.

In the same vein and with the same purpose, MTT introduces two sublevels — deep *vs.* surface — in morphology and phonology:

4) D(eep-)Morph(ological) R(epresentation).
5) S(urface-)Morph(ological) R(epresentation).
6) D(eep-)Phon(etic) R(epresentation),
or phonological representation.
7) S(urface-)Phon(etic) R(epresentation),
or phonetic representation proper.

We will not justify the Deep- *vs.* Surface- dichotomy or, more generally, the composition and organization of our set of representation levels. Instead of this, we will try to link our somewhat abstract statements to a specific example. Namely, we will quote a French sentence along with its representations on the first three levels. We will not consider here the morphological and phonological representations of this sentence, since they are not relevant to our goal. Note that throughout this paper we will use examples borrowed from French (since we had the corresponding information available only in that language, while to work out the English examples would require special research, which we are in no position to undertake); but to facilitate the reading of all the examples, we will supply approximate English glosses for all French lexical items.

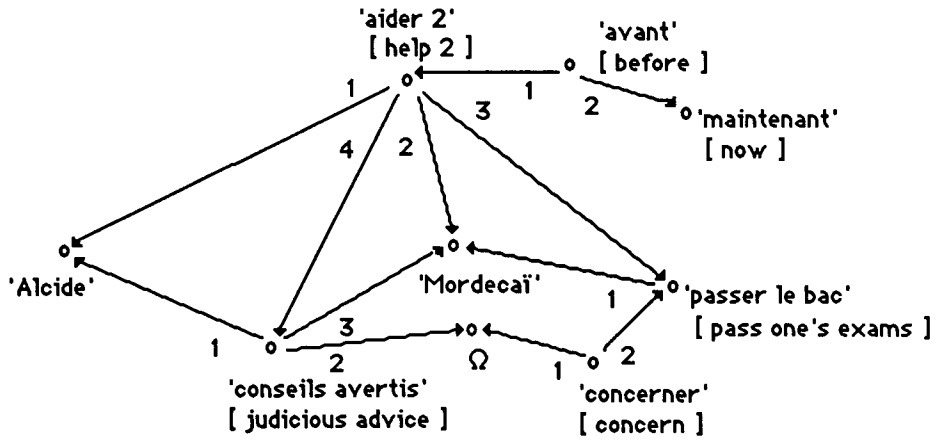
Let us consider French sentence (1):

(1) Fr. *Alcide* (X) *a aidé Mordecaï* (Y) *à passer son bac* (Z) *par ses conseils avertis* (W)
'Alcide helped Mordecaï pass his [high school leaving] exams with his judicious advice'.

At the Sem-level, sentence (1) appears as Figure 1:

[1] The vague term *utterance* is used on purpose. In the present paper we take the sentence as our basic unit, but the MTT is not restricted to sentences. In principle, it can deal with sequences of sentences, although up to now, within the MTT, the way to represent such sequences and the rules to process them have not yet been developed (in contrast with such works as, e.g., McKeown (1985)).

[2] The term *meaning* is to be construed here in the narrowest sense— as referring to strictly linguistic meaning, i.e. the meaning (of utterances) which is given to any native speaker just by the mastery of his language. We say this to avoid a misunderstanding: our *meaning* has nothing to do with "actual" meaning, which is aimed at by such questions as 'What do you mean by that?' or 'What is the meaning of this paper?' This restricted character of meaning in our interpretation will become clear when we discuss semantic representation in the MTT (see below).



SemR of Sentence (1)

Figure 1

In this figure, Ω is a dummy to indicate an unspecified meaning (it is not specified what exactly the advice from Alcide is). In principle, every lexemic sense must be identified by a number; for the sake of simplicity we do this here for one node only: AIDER 2, which will be discussed in 2.1.3. To avoid unnecessary complications, we have grouped certain semantic elements together ('passer le bac' and 'conseils avertis'). The same shortcut is used in the next two figures.

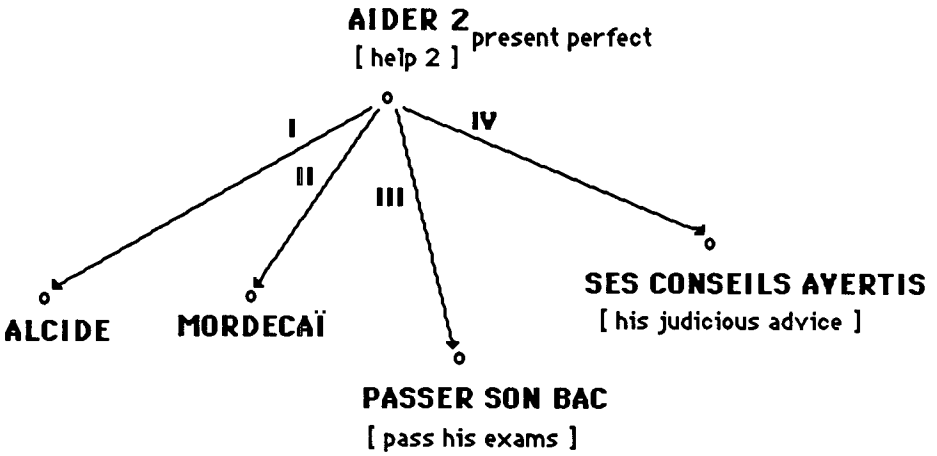
At the DSynt-level, sentence (1) appears as follows: The subscript "present perfect" on AIDER 2 corresponds, in the SemR, to the subnetwork 'before now': this is roughly the meaning of the French present perfect (called "passé composé"). During the transition to the SSynt-level, this subscript triggers a DSynt-rule that introduces the appropriate auxiliary verb (in our case, AVOIR 'have') along with the auxiliary SSynt-relation, linking it to the lexical verb (i.e., AIDER 2) in the participial form: see Fig. 3. Note that the first DSynt-dependent of the verb AIDER 2 in the DSyntR (ALCIDE) must be linked as the grammatical subject to the AVOIR node in the SSyntR, while all other dependents of AIDER 2 remain dependents of its participial form.

At the SSynt-level, we get Figure 3:

Now we will move to the characterization of the formal device, a set of rules, which ensures the transition between the representations of the above levels: the Meaning-Text Model (MTM).

1.2. THE MEANING-TEXT MODEL

The MTM is nothing else but what is currently called *grammar* (we avoid this usage because we would like to distinguish and even contrast *grammar* vs. *lexicon*, both being parts of a linguistic model). More specifically, it is



DSyntR of Sentence (1)

Figure 2

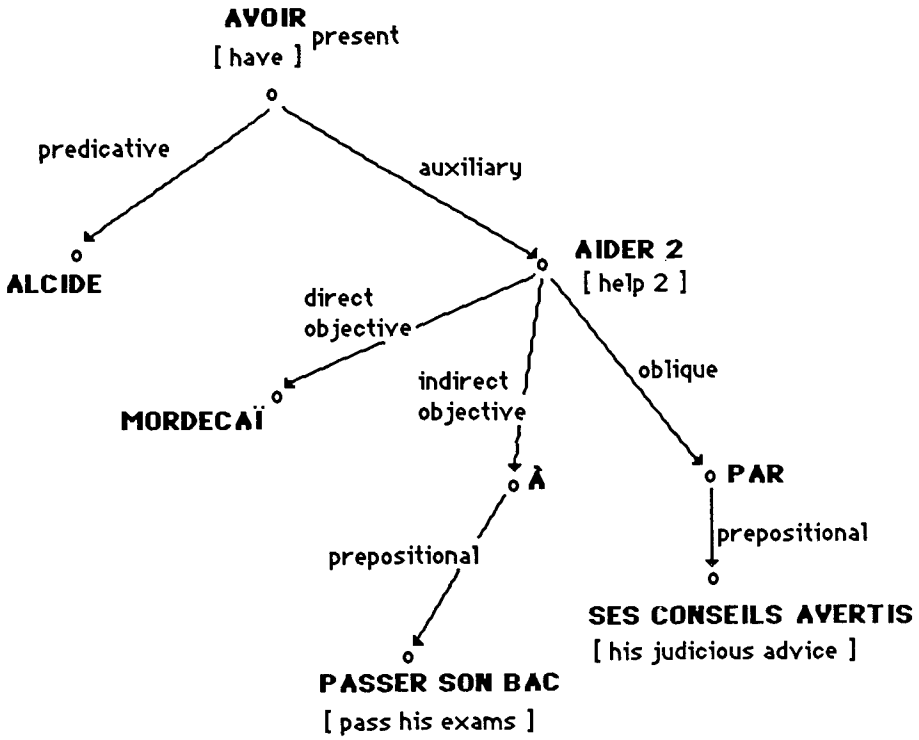


Figure 3

a set of formal rules, with complex internal organization, which, so to speak, translate the initial SemR into a final SPhonR (or into a written text) and vice versa. Of course, in doing so, the rules pass through all intermediate representations. Therefore the MTM is subdivided into components such that each one deals with the correspondence between two adjacent levels n and $n + 1$; given seven levels of representation, there are six components:

1) The Semantic Component (transition between SemR and DSyntR);
2) The Deep-Syntactic Component (transition between DSyntR and SSyntR);
3) The Surface-Syntactic Component (transition between SSyntR and DMorphR);
4) The Deep-Morphological Component (transition between DMorphR and SMorphR);
5) The Surface-Morphological Component (transition between SMorphR and DPhonR);
6) The Deep-Phonetic Component (transition between DPhonR and SPhonR).

Each component has a roughly identical internal structure; namely, it contains three types of rules:

— well-formedness rules for representations of the source level;

— well-formedness rules for representations of the target level;

— transition rules proper.

The well-formedness rules serve simultaneously both to check the correctness of the representation in question and to contol the application of the transition rules. Let it be emphasized that until now the MTM (and the MTT in general) does not contain the data necessary to effectively carry out the said control; thus the development and organization of these data constitutes, from a computational viewpoint, the main problem in the application of the MTT.

To sum up, the synthesis of a sentence appears in the Meaning-Text framework as a series of subsequent transitions, or translations, from one representation to the next one, beginning with SemR; the analysis takes of course the opposite direction, starting with the SPhonR or with the written text. This is, however, only a logical description of what happens. In a real computational implementation, it is often necessary, in order to take a decision concerning a particular level of representation, to consider several other levels simultaneously.

As the reader has probably realized, the MTM belongs to so-called *stratificational* models of language, launched about a quarter of a century ago (Lamb 1966, Sgall 1967, Mel'čuk 1974); note that at present we can see a clear tendency to introduce certain ideas of the stratificational approach into theoretical and computational linguistics (compare, e.g., the distinction of sev-

eral levels of linguistic representation in Bresnan's Lexical Functional Grammar).

Since our goal in the paper is to discuss the role of a formal lexicon in a linguistic model, and, as we will try to show, our formal lexicon is a part of the semantic component (of the MTM), we will concentrate only on this portion of the model.

2. EXPLANATORY COMBINATORIAL DICTIONARY.

A lexicographic unit in the ECD[3], i.e. a dictionary entry, covers one lexical item — a word or a set phrase — taken in one well-specified sense. All such items called *lexemes* (respectively, *phrasemes*) are described in a rigorous and uniform way, so that a dictionary entry is divided into three major zones: the semantic zone, the syntactic zone, and the lexical cooccurrence zone. (We leave out of consideration all other zones, as irrelevant to the main purpose of this paper.)

2.1. SEMANTIC ZONE.

2.1.1. THE LEXICOGRAPHIC DEFINITION AS THE BASIS OF AN ECD ENTRY.

An ECD entry is centered around the definition of the head word, i.e. the representation of its meaning, or its SemR. All particularities of the ECD, as far as the semantic domain is concerned, follow from the fact that meaning is taken to be the first and foremost motivation for everything else in the dictionary: relevant properties of lexical items are discovered and described contingent upon their definitions. In order to make this important property more obvious, let us compare the MTT approach to the lexicon with certain current approaches known in computational linguistics. The approaches we mean here are rather syntax-motivated, so that they view the lexicon as an appendix to the grammar, the latter being equated with the formulation of syntactic well-formedness. In MTT, it is exactly the opposite: the grammar is considered to be an appendix to the lexicon, an appendix which, on the one hand, expresses useful generalizations over the lexicon, and on the other hand, embodies all procedures necessary for manipulating lexical data. As we have said, the lexicon and the grammar taken together constitute the MT model, with the lexicon at its foundations.

ECD definitions possess, among other things, the following two properties, which distinguish them from the definitions of many lexica used in computational linguistics:

a) An ECD definition must be adequate in the sense that all possible[4] correct usages of the lexeme defined are covered by it and all incorrect usages are excluded. In other terms, all the components of an ECD definition are necessary and the set of these is sufficient for the task just stated.

b) In various computational or formal approaches, lexicographic definitions are written in terms of a small set of prefabricated elements. It looks as if the researcher's goal were to make his definitional language as different as possible from his object language (cf. conceptual dependencies in Schank (1972), logical representations in Dowty (1979), and the like). One major drawback of this type of approach is that it does not provide for a direct and explicit expression of lexical cohesion in the language under analysis: possible relationships among lexical items have to be inferred from their definitions in a very indirect way. In sharp contrast to this, the ECD defines a lexeme L of language L in terms of other lexemes L_1, L_2, . . ., L_n of L in such a way that the meaning of an L_i is simpler than that of L, which precludes circularity (for further discussion, see 2.1.2.a). As a result, in the ECD, semantic relationships among lexemes are established and explicitly stated directly via their definitions.

2.1.2. FORMAL CHARACTER OF AN ECD DEFINITION.

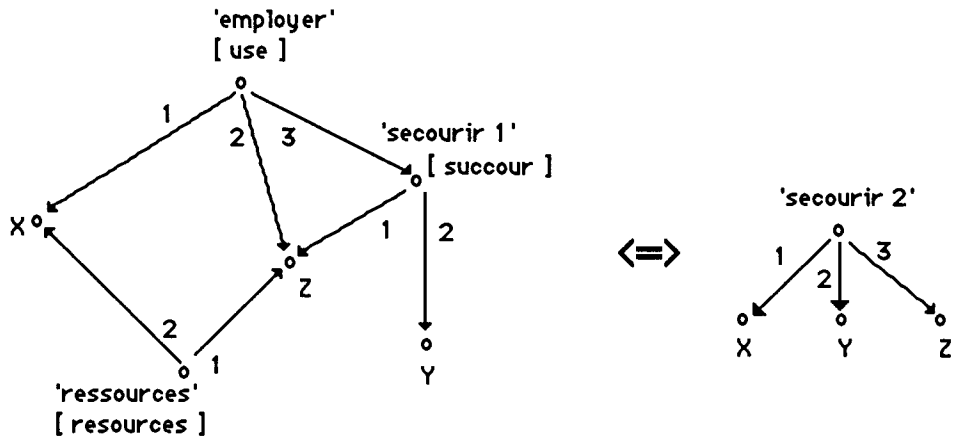
An ECD definition is a decomposition of the meaning of the corresponding lexeme. It is a semantic network whose nodes are labeled either with semantic units (actually, lexemes) of L, or with variables, and whose arcs are labeled with distinctive numbers which identify different arguments of a predicate. A lexical label represents the definition (the meaning) of the corresponding lexeme, rather than the lexeme itself. Therefore, each node of a definitional network stands, in its turn, for another network, whose nodes are replaceable by their corresponding networks, and so forth, until the bottom level primitives are reached. For practical reasons, though, one can take as primitives the lexemes of the level at which the researcher decides to stop the process of decomposing. This approach is directly related to the pioneering work of Wierzbicka: see, e.g., Wierzbicka (1980).

The elaboration of ECD definitions must satisfy a number of principles, of which we will mention here the following two:

a) *The Decomposition Principle* requires that a lexeme be defined in terms of lexemes which are semantically simpler than it; as mentioned above, this precludes circularity. This principle, if applied consistently, will

[3] Let it be emphasized that the abstract concept of an ECD as presented below is by no means the same thing as an actual ECD implemented in a specific form, such as, e.g., Mel'čuk *et al.* (1984) and Mel'čuk and Žholkovsky (1984). The particular concept of ECD which we propose here is no more than a logical constraint on an infinity of possible ways to build concrete ECDs. On the one hand, we insist on the concept of ECD only, without touching upon the methods of its realization; on the other hand, the actual ECDs of Russian and French, mentioned above, are not well adapted, under their present form, to computational treatment.

[4] By *possible* we mean 'possible in any imaginable context' — with the obvious exception of contexts involving either the phonetic form (as, e.g., in poetry) or metalinguistic use of lexical items.



Definition of the French verb SECOURIR 2

What we mean by a (lexicographic) definition of a lexical meaning is an equivalence between this lexical meaning itself taken together with its Sem-actants (the righthand part of the figure) and its semantic decomposition observing the Maximal Block Principle (the lefthand part).

In English, the network in the lefthand part can be read as: 'X uses Z which is X's resources in order to SECOURIR 1 Y'

Figure 4

lead to a set of semantic primitives. As is easily seen, we do not begin by postulating a set of semantic primitives: we hope to have them discovered by a long and painstaking process of semantic decomposition applied to thousands of actual lexical items. But let it be underscored that within the MTT framework it is not vital to have semantic primitives at hand in order to be able to proceed successfully with linguistic description.

b) *The Maximal Block Principle* requires that, within the definition of lexeme L, any subnetwork which is the definition of L′ be replaced by L′; this ensures the graduality of decomposition, which contributes to making explicit interlexical links in the language. In fact this principle forbids writing a definition in terms of semantic primitives if semantic units of a higher level are available; this makes our definitions immediately graspable and more workable.

2.1.3. AN EXAMPLE OF THE INTERACTION AMONG ECD DEFINITIONS.

To illustrate the way the vocabulary of L is semantically hierarchized using ECD definitions, we will consider some definitions involving semantically related lexemes.

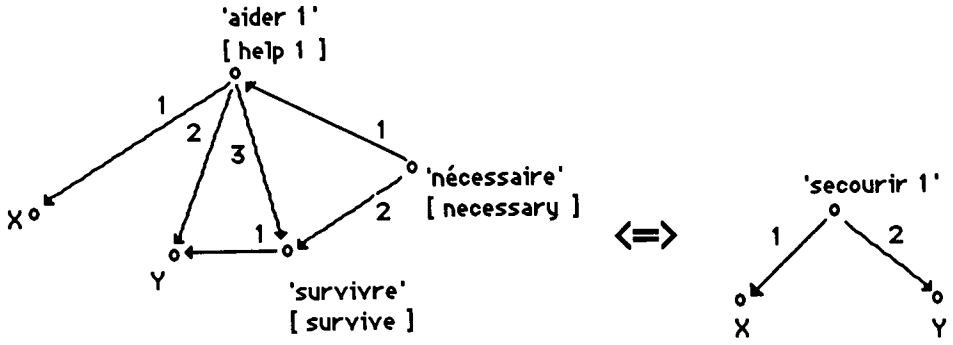
Let us take the French verb SECOURIR, roughly 'succour'. (Although this English gloss is not a very familiar word in Modern English, we cannot find a better equivalent, since there is no single English lexical item that covers exactly the meaning of this French verb. However we believe that this is a good example because it shows how different the lexicalizations of the same meaning in two different languages can be. We hope that our semantic description of SECOURIR will eventually make its meaning clear to the reader.) SECOURIR corresponds to two lexemes the meaning of which can be illustrated by the following examples:

(2) SECOURIR 1
Ce vaccin a secouru de nombreux enfants
'This vaccine helped/saved many children'.

SECOURIR 2
Le médecin a secouru de nombreux enfants avec ce vaccin
'The doctor helped/saved many children with this vaccine'.

There exists an obvious semantic relation between the two lexemes: a causative one; thus SECOURIR 2 means 'to use something for it to SECOURIR 1 someone', that is, very roughly, 'to cause something to SECOURIR 1 someone' (the concept of causation is implicit in 'use'). In a more formal way, we can represent the meaning of SECOURIR 2 by a semantic network which includes SECOURIR 1 (Figure 4).

SECOURIR 1, in its turn, can be defined in terms of AIDER 1, roughly 'help' (as in *La lumière aide les*



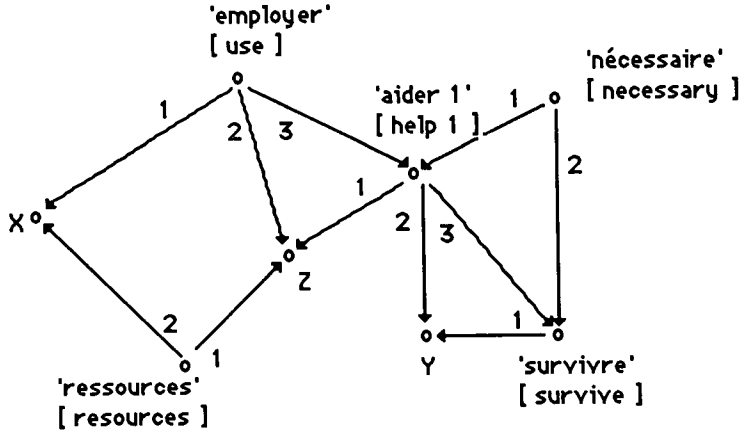
Definition of the French verb SECOURIR 1

In English, the network can be read as:
'X helps 1 Y to survive, this help1 being necessary for Y's survival'

Figure 5

plantes dans leur croissance, lit. 'Light helps plants in their growth'), plus the following two important semantic components: 'survive' and 'necessary'. More specifically, *X SECOURT 1 Y* means 'X AIDE 1 [=helps 1] Y to survive, X's helping 1 Y being necessary for Y's survival'. The node 'secourir 1' in the network of Fig. 4 can be replaced by its own definition, i.e. by the network of Fig. 5, giving Fig. 6. It is obvious that, generally speaking, the inverse substitution is also possible: a network representing a definition of a lexical meaning can be replaced by a single node representing this meaning. This type of network manipulation must be carried out automatically by substitution procedures based on lexico-semantic rules of the form:

$$\text{SemR}_i <=> \text{MEANING OF LEXEME L}_i$$

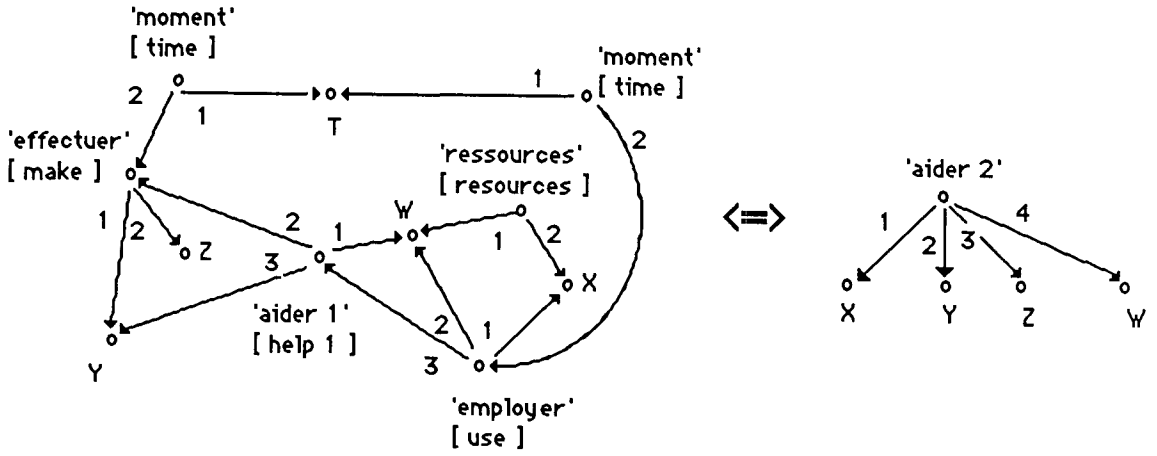


Deeper semantic decomposition for 'X SECOURT 2 Y'
(with the definition of SECOURIR 1 substituted for the latter)

Figure 6

Such a rule is nothing less than the central element of a lexical entry; more precisely, the core of its semantic zone.[5]

Note that the same causative relation that exists between SECOURIR 1 and SECOURIR 2 exists also between AIDER 1 and AIDER 2 (which appeared in our first example, sentence (1)), as can be illustrated by the definition of AIDER 2 in Fig. 7:



Definition of the French verb AIDER 2
In English, the network can be read as follows:
'Y carrying out Z, X uses his resources W in order for W to help 1 Y to carry out Z; the use of resources by X and the carrying out of Z by Y are simultaneous'.

Figure 7

To sum up the semantic inclusion relations among the four lexemes discussed, let us present these in the following obvious form:

'secourir 2' **includes** 'secourir 1', **which includes** 'aider 1', **which is included in** 'aider 2';

at the same time, there is an approximate proportionality:

'secourir 2' / 'secourir 1' ≅ 'aider 2' / 'aider 1'.

[5] In our framework, a (lexicographic) definition is a semantic rule which is part of the semantic component of the MTM. Semantic rules of this type serve to reduce a SemR network to a more compact form (if applied from left to right) and/or to expand a SemR network (if applied from right to left). A second type of semantic rule are rules matching a lexeme meaning with the corresponding lexeme, i.e. rules that carry out the transition between a SemR and a DSyntR (or vice versa). Generally speaking, all the rules of the MTM are subdivided into these two types: (a) rules establishing correspondences between units of the same linguistic level, or *equivalence rules* (semantic rules of the type illustrated in this paper in Figs. 4, 5, 7; paraphrasing rules — see (9) below); (b) rules establishing correspondences between units of two adjacent linguistic levels, or *manifestation rules* (e.g., Fig. 9). We are in no position to go into further details, but we would like to stake out this important distinction.

2.1.4. ON INTERPRETING THE SEMR.

As we have already said, the SemR in the MTT is aimed at representing the **linguistic** meaning, i.e., the common core of all synonymous utterances; it serves to reduce the enormous synonymy of natural language to a standard and easily manageable form — but it has no direct link with the real world. Therefore, the full-fledged description of linguistic behavior should include another representation — the representation of the state of affairs in the real world, and another model — the Reality-Meaning Model. The latter ensures the transition between the results of perception etc. and the SemR, which is the starting/end point of linguistic activity proper.

Our exclusion of real world knowledge from the SemR and therefore from the MTM opposes our approach to others, such as, e.g., Montague Grammar, which claims that the rules associating semantic representations with sentences and the rules interpreting the same representations in terms of extralinguistic reality (set-theoretical interpretation) are of the same nature and can be integrated within the same model. We insist on distinguishing the SemR proper from the real world/knowledge representation because we feel that the description of linguistic activity in the strict sense of the term is very different from the description of perceptual or logical activity.

To illustrate the relation between a SemR and the corresponding real-world representation, we can take the example of machine translation. We think that the only common denominator of two texts (in different

languages) equivalent under translation is the representation of the state of affairs referred to, not a SemR of either text. The SemR is not an ideal interlingua — although it can be effectively used to bring closer the source and the target languages — because a SemR, by definition, reflects the idiomatic characteristics of the language in question.

2.2. SYNTACTIC ZONE.

This zone stores the data on the syntactic behavior of the head lexeme — more specifically, on its capacity to participate in various syntactic configurations. Along with the part of speech (= syntactic category), the syntactic zone presents two major types of information:

— Syntactic features.

— The government pattern.

These two types are distinguished by their bearing on the semantic actants (= arguments) of the head lexeme.

2.2.1. SYNTACTIC FEATURES.

A syntactic feature of lexeme L specifies particular syntactic structures which accept L but which are not directly related to the semantic actants appearing in its definition (though this does not preclude a syntactic feature from being linked to the meaning of L in a different way). For instance, the feature "geogr" singles out English common nouns capable of appearing in the construction

$Det_{def} + N^1_{geogr} + of + N^2_{proper}$,
having the meaning 'the N^1 called N^2'

Such nouns are, e.g., *city, island, republic* (but not *mountain, peninsula, river, . . .*):

(3) a. *the city of London*
the island of Borneo
the Republic of Poland

b. **the Mountain of Montblanc <the Montblanc Mountain>*
**the Peninsula of Kamtchatka <the Kamtchatka Peninsula>*
**the River of Saint-Lawrence <the Saint-Lawrence River>*.

Note that "geogr" will block such an expression as **Mountain of London*, but not because London is not a mountain; this will happen because *a mountain* does not **syntactically** admit its name in the form of *of X*. *The London Mountain* will not be precluded by this feature since it is syntactically perfectly correct.

Syntactic features, which do not presuppose strictly disjoint sets, provide for a more flexible and multifaceted subclassification of lexemes than do parts of speech, which induce a strict partition of the lexical stock.

2.2.2. GOVERNMENT PATTERN.

The government pattern of lexeme L specifies the correspondence between L's semantic actants and their realization at the DSynt-level, SSynt-level and DMorph-level. It is a rectangular matrix with three rows:

— the upper one contains semantic actants of L (X,Y,Z,···);

— the middle one indicates the DSynt-roles (I,II,III,···) played by the manifestations of the Sem-actants on the DSynt-level with respect to L;

— the lower one indicates structural words and morphological forms necessary for the manifestations of the same Sem-actants on the SSynt- and DMorph-levels contingent on L.

The number of columns in this matrix is equal to the number of Sem-actants of L. Each column specifies the correspondence between a Sem-actant and its realizations on closer-to-surface levels.

To illustrate, we will use the government pattern of the French lexeme AIDER 2, the definition of which has been presented in Fig. 7. This definition involves four semantic actants (X, Y, Z and W), all of which are implemented in sentence (1), see the end of Section 1.1. The government pattern of AIDER 2 has the form shown in Fig. 8.

Let us now discuss the way the government pattern of a lexeme is used to ensure the appropriate correspondence between the representations of adjacent levels (within the framework of a transition from the SemR toward the sentence). We will focus on a fragment of sentence (1), more specifically, on the correspondence between the semantic actant Z of AIDER 2 and its syntactic correlates on both syntactic levels.

In the transition between the SemR of (1) and its DSyntR (see Figs. 1 and 2), the government pattern of AIDER 2 establishes that Z is DSynt-actant III of this lexeme. The correspondence "Z <=> III" is specified in the 3rd column of the government pattern (Fig. 8), which will be our working column.

The transition between the DSyntR and the SSyntR of (1), Figs. 2 and 3, may be thought of as being carried out in three steps:

1) Since the dependent of the DSynt-relation III, headed by AIDER 2, is a verb,[6] namely PASSER 'pass', we select lines 2 (*à* V) and 5 (*pour* V) of column III (only these two lines imply a verb occurrence).

2) We make our choice between lines 2 and 5. (In this specific case, the choice is made according to the restrictions imposed on the government pattern in question, but not given in this paper. More specifically, the preposition *pour* seems to imply the participation of X, agent of AIDER 2, in activity Z; since, however,

[6] The indication of the part of speech of a given lexeme, as well as its government pattern, its syntactic features etc., do not appear in the representation itself but are stored in the dictionary entry of the lexeme, which must be made easily available to all computational procedures.

X	Y	Z	W
I	II	III	IV
1. N	1. N	1. à N 2. à V 3. dans N 4. pour N 5. pour V	1. de N 2. avec N 3. par N 4. Adv

Government pattern of the French verb AIDER 2

X, Y, Z and W are the semantic actants of AIDER 2: X is the person who helps, Y the person who receives help, Z the activity of Y in which he needs help, and W the resources by which X helps Y.
I, II, III and IV are the Deep-Syntactic actants of AIDER 2: I refers to the noun phrase that expresses X etc.

Figure 8

PASSER SON BAC 'pass one's exams' is a purely individual action, one has to choose the preposition *à*, line 2.)

3) The selection of the preposition having been made, a DSynt-rule (cf. Figure 9) introduces (into the SSyntR) the lexical node À, with the concomitant SSynt-relations: "indirect objective" — from AIDER 2 to À, and "prepositional" — from À to PASSER (SON BAC).

This rule provides a general frame for the expression of DSynt- relations, but it is the government pattern of a specific lexeme, in this case **A** = AIDER 2, that supplies the specific preposition, in this case **PREP** = À. Thus a particular government pattern serves to instantiate in an appropriate way the variables in certain DSynt-rules.

In general, a government pattern has associated with it a number of restrictions concerning the cooccurrence and the realization of actants:

— an actant cannot appear together with/without another actant;
— a given surface form of an actant determines the surface form of another actant;
— a given realization of an actant is possible only under given conditions, semantic or otherwise (cf. the restriction mentioned above, step 2, concerning the choice between *à* and *pour*);
— etc.

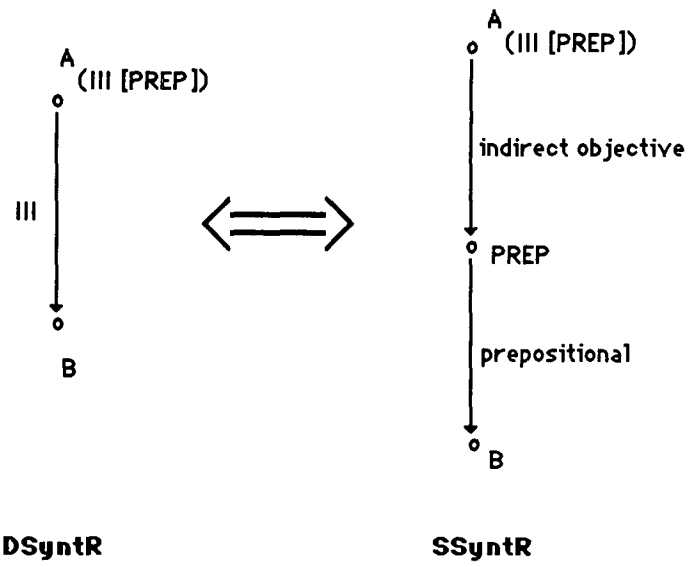
These restrictions function as filters screening possible forms and combinations of actants on the DSynt-, as well as on the SSynt-level.

2.2.3. A FEW REMARKS ON THE GOVERNMENT PATTERN *VS.* THE FEATURE APPROACH.

The notion of government pattern, as an element of lexical description, is not in itself revolutionary and is present in a number of linguistic theories. Thus what we call the government pattern is related to the concept of subcategorization in generative grammars. For instance, Generalized Phrase Structure Grammar (Gazdar *et al.* 1985) utilizes in its rules what are called *syntactic features*,[7] subcategorizing lexical items according to, in our terminology, their government pattern. GPSG postulates the existence of subcategories, for instance **V[32]**, i.e. a verb of type 32, etc., which constrain the application of grammar rules by selecting the lexical items they can deal with. It seems obvious that it is possible to make generalizations by grouping certain elements of a single syntactic category on the basis of certain similarities in their syntactic behavior; we do not believe, however, that one has to establish those groupings according to general syntactic rules rather than according to individual lexicographic descriptions. Tackling the problem from the viewpoint of the lexicon presents at least two advantages:

1) One can describe the syntactic behavior of lexeme L vis à vis its syntactic actants in L's lexical entry without having to examine a rather complex system of syntactic rules dealing with syntactic features attributed to L. Thus, syntactic features are a code that needs interpretation in terms of pre-existing syntactic rules; in

[7] Let it be strongly emphasized that the GPSG concept of syntactic feature does not correspond to syntactic features as used in the MTT (see 2.2.1 above).



A DSynt-rule for the realization of the DSynt-relation III

Figure 9

contrast to that, a government pattern is a "speaking" expression which explicitly specifies the syntactic micro-structure typical of L — independently of any syntactic rules that might use it.

2) One avoids postulating disjoint lexemic classes according to syntactic behavior (note that as a rule one has no means of evaluating, a priori, how many such classes there are and what their characteristics would be).

We could mention also that a very detailed description of lexemes in terms of government patterns allows one to consider nearly as many distinct syntactic behaviors (i.e., subclasses) as there are lexical items. For French, this seems to be what can be inferred from verb lists constructed by M. Gross and his associates (Gross 1975).

2.3. Lexical combinatorics zone.

2.3.1. What is restricted lexical cooccurrence?

The main novelty of the ECD is a systematic description of the restricted lexical cooccurrence of every head lexeme. Restricted lexical cooccurrence can be of two quite different types.

The first type is illustrated by the impossible cooccurrence observed in a sentence such as (4):

(4) *The telephone was drinking the sexy integrals.*

In (4), certain lexemes cannot cooccur only because of their meanings and of our knowledge of the world. The exact translation of (4) will be deviant in any language, and this means that the restrictions violated in (4) are of a non-linguistic nature. They should not be covered by a linguistic description.

In sharp contrast to (4), the phrases in (5) demonstrate the second type of restricted lexical cooccurrence, i.e. what are considered by MTT to be truly linguistic restrictions on lexical cooccurrence:

(5) a.
Eng. (*to*) ASK *a question*
Fr. POSER (litt. 'put') *une question*
Sp. HACER ('make') *una pregunta*
Russ. ZADAT' ('give') *vopros*

b.
Eng. (*to*) LET OUT *a cry*
Fr. POUSSER ('push') *un cri*
Sp. DAR ('give') *un grito*
Russ. ISPUSTIT' ('let out') *krik*

The selection of the appropriate verb to go with a given noun cannot be done according to the meaning of the latter; there is nothing in the meaning of 'question' to explain why in English you *ask* it while in Spanish you *make* it, in French you *put* it and in Russian you *give* it. This type of restricted lexical cooccurrence is the prime target of lexicographic description, which uses for this purpose so-called **lexical functions** (see below).

However, we think that our lexicographic description has no place for selectional restrictions formulated in terms of semantic features. Let us consider an example of selectional restrictions as used in some approaches in computational linguistics: McCord's Slot Grammar (McCord 1982), which has the advantage of giving, in a sample lexicon for a database questioning system, fully explicit information concerning the government pattern of each lexical entry. Thus we have for

the lexeme COURSE (as in *programming course, take a course etc.*):

(6) noun(**course,** crs(X,Y,_,_), nil, X:crs,
[npobj(*in*):Y:subject]).

In this Prolog expression, the last argument of the five-place predicate 'noun' indicates that the complement (= NP object) of the lexeme COURSE introduced by the preposition IN (*course in ancient history*) must be labeled with the semantic feature "subject". We, however, think that such indications should by no means appear in the description of lexemes as cooccurrence restrictions. Actually, one can give a course on anything at all (flowerpots, sexual habits of bedbugs, etc.). Just giving a course on something makes this something a subject. Therefore, we must indicate that the Y argument of the predicate 'course' is called *subject* but we cannot say that in order to be able to fill in the argument Y of COURSE a noun must be semantically labeled "subject".[8]

2.3.2. THE CONCEPT OF LEXICAL FUNCTION.

A lexical function **f** is a dependency that associates with a lexeme L, called the argument of **f**, another lexeme (or a set of (quasi-)synonymous lexemes) L' which expresses, with respect to L, a very abstract meaning (which can even be zero) and plays a specific syntactic role. For instance, for a noun N denoting an action, the lexical function $\mathbf{Oper_1}$ specifies a verb (semantically empty — or at least emptied) which takes as its grammatical subject the name of the agent of said action and as its direct object, the lexeme N itself. Thus the phrases in (5) are described as follows:

(7) a.
Engl. $\mathbf{Oper_1}$ (QUESTION) = ASK
Fr. $\mathbf{Oper_1}$ (QUESTION) = POSER
Sp. $\mathbf{Oper_1}$ (PREGUNTA) = HACER
Russ. $\mathbf{Oper_1}$ (VOPROS) = ZADAT'
b.
Engl. $\mathbf{Oper_1}$ (CRY) = LET OUT
Fr. $\mathbf{Oper_1}$ CRI) = POUSSER
Sp. $\mathbf{Oper_1}$ (GRITO) = DAR
Russ. $\mathbf{Oper_1}$ (KRIK) = ISPUSTIT'

There are about 60 lexical functions of the $\mathbf{Oper_1}$ type, called *standard elementary LFs*. They and their combinations allow one to describe exhaustively and in a highly systematic way almost the whole of restricted lexical cooccurrence in natural languages. Given the importance of lexical functions for "intelligent" linguistic systems, we will offer an abridged list thereof in the appendix (see also Mel'čuk 1982).

In the MTM LFs play a double role:

1) During the production of the text from a given SemR, LFs control the proper choice of lexical items linked to the lexeme in question by regular semantic relations. For instance, if we need to express in French the meaning 'badly wounded' ['badly' ⇒ Fr. *mal*, 'wounded' ⇒ *blessé*, but 'badly wounded' cannot be translated by **mal blessé*], we take the following steps:

(8) a. 'wounded' ⇒ Fr. *blessé;*
b. 'badly' with respect to 'wounded' is the LF **Magn** [meaning 'very', 'intensely']: **Magn** (*wounded*) ⇒ *badly* ;
c. therefore, 'badly wounded' ⇒ **Magn** (*blessé*) + *blessé*;
d. **Magn** (*blessé*) = *grièvement* ;
e. finally, 'badly wounded' ⇒ *grièvement blessé.*

During the analysis of a text, LFs help to resolve syntactic homonymy, since they indicate which word has the greater likelihood of going with which other word.

2) In text production, LFs are used to describe sentence synonymy, or, more precisely, the derivation of a set of synonymous sentences from the same DSynt-structure. This is done by formulating, in terms of LFs, a number of equivalences of the following type:

(9) $C_{0\,(V)} <=> \mathbf{Oper_1}\,(\,\mathbf{S_0}\,(\,C_0\,)\,) \xrightarrow{II} \mathbf{S_0}\,(\,C_0\,)$,

where C_0 stands for the head lexeme and $\mathbf{S_0}$ for the action nominal.

This equivalence can be illustrated by (10):

(10) *Alex received* [C_0] *me quite well* <=> *Alex gave* [$\mathbf{Oper_1}$ ($\mathbf{S_0}$ (C_0))] *me quite a good reception* [$\mathbf{S_0}$ (C_0)].

The operation carried out by this type of rule is called paraphrasing. About sixty paraphrasing rules of the form (9) are needed to cover all systematic paraphrases in any language (moreover, there must be about thirty syntactic rules which describe transformations of trees and "serve" the rules of the above type). A powerful paraphrasing system is necessary, not only because it is interesting in itself, but mainly because without such a system it seems impossible to produce texts of good quality for a given SemR: indeed when one is blocked during a derivation by linguistic restrictions, one can bypass the obstacle by recourse to paraphrases.

During text analysis, a powerful paraphrasing system helps to reduce the vast synonymy of natural language to a standard and therefore more manageable representation.

3. THE ECD IN THE COMPUTATIONAL IMPLEMENTATION OF THE MEANING-TEXT MODEL.

Within the framework of the global task of automatic natural language processing (which is involved in, e.g., natural language understanding, machine translation,

[8] Selectional restrictions of the type just indicated are needed in most cases for possible applications of a grammar, e.g., in order to facilitate the resolution of syntactic homonymy. We, on the other hand, do not want to mix the theoretical description of a grammar with tools appropriate for its applications. Note, however, that we use selectional restrictions as well, but only if they are necessary for the choice of the correct linguistic expression: for instance, with a given verb, one preposition is chosen with human nouns while the other one takes only abstract nouns, etc. (cf. the choice between *pour* vs. *a* in 2.2.2).

natural language interfaces, etc.) the MTM addresses one rather restricted but precise problem: the production of the highest possible number of synonymous utterances for a given meaning, presented in the standard form of a SemR, or, inversely, the reduction of a wealth of synonymous utterances to their standard semantic invariant — their SemR. All the other components of a natural language system could then be geared to a representation of meaning, thus avoiding all the capricious phenomena of natural language not directly related to meaning. We would like to discuss below two issues relevant to this aspect of natural language processing, an aspect which we consider to be strictly linguistic, and specifically relevant to the ECD: 1) using the lexicon in the transition between levels of linguistic representation and 2) structuring the lexicon.

3.1. USING THE LEXICON IN THE TRANSITION BETWEEN LEVELS OF REPRESENTATION.

To see how the ECD is applied by the MTM in the utilization of the latter, let us assume that the MTM has to deal with a pre-existent SemR which corresponds to a sentence (we disregard the problem of cutting any initial SemR, which can represent the global meaning of a whole text, into "smaller" SemRs, representing the meanings of separate sentences). We presuppose that the model is able to associate with such a SemR all DSyntRs of the sentences which a speaker of the language in question could produce for the given meaning. In other words, the model carries out the mapping $\{SemR_i\} <==> \{DSyntR_k\}$. This mapping is implemented via the following two operations which are logically distinct, even though we can conceive of them as constantly interacting and applying simultaneously:

1) The first operation groups semantic elements of the initial SemR into clusters each of which corresponds to a lexical unit of the language. Thus this operation selects **the lexical stock** for the target DSyntR and can be called **lexicalization.** (As the careful reader may have noticed, we have already discussed lexicalization without naming it at the end of Sec. 2.1.3.)

2) The second operation organizes the lexical units supplied by the first operation into a dependency tree under the control, on the one hand, of the information found under the dictionary entries for these units (various constraints on cooccurrence), and, on the other hand, of semantic links between their sources in the SemR. Thus this operation constructs **the syntactic structure** for the target DSyntR and can be called **syntacticization.**

It is obvious that the two operations are not completely independent of each other; but at present we do not know how they interact. There is an even more serious problem: the MTM does not offer procedural tools ensuring different groupings of semantic network elements, in order to join them into lexicalizable clusters. In other words, the MTM in its current state lacks mechanisms and devices for semantic network analysis. Nevertheless, whatever these mechanisms may be, they must rely in an essential way on the ECD, from which they have to draw all necessary semantic and syntactic information. After all, semantic units in a network, i.e. in a SemR, are nothing else but lexical meanings in the language under consideration.

Each DSyntR obtained in the way just described undergoes a set of *paraphrasing rules,* which involve lexical functions; an example of such a rule is rule (9). These rules derive from the initial DSyntR the set of DSyntRs which are synonymous to it, thus providing for necessary synonymic flexibility: as we have just said, under synthesis, the model displays the multitude of variants, so that the selection of an appropriate one becomes easier; under analysis, it reduces the host of variants to a standard representation, which facilitates its subsequent processing. The paraphrasing rules of the MTM have already been studied from the computational viewpoint (Boyer and Lapalme 1985). Note that theoretically these rules are reversible: although they have been implemented in the synthesis direction, they must function as well under analysis.

As for the operations and the corresponding components of the MTM which have the task of carrying out the transition between closer-to-surface levels (from DSyntR to SSyntR, from SSyntR to DMorphR, etc.), we will not consider them here for lack of space.

3.2. THE OVERALL STRUCTURING APPROACH TO THE LEXICON AS A TYPICAL TRAIT OF THE ECD.

Computational linguistics today knows many language analysis and synthesis systems relying on an explicit formalized lexicon. Traditionally, however, these systems use a "flat" declarative representation of lexical information. More specifically, a lexical entry in a classical computational lexicon is an independent proposition — i.e., a structure fully autonomous with respect to all other entries; such an entry contains a set of lexical data (concerning the head word) which is almost always isolated from other similar sets of lexical data. This boils down to the following principle: generally speaking, a lexeme is not described as a function of other lexemes of the language, so that the information supplied by the lexicon does not relate it to the rest of the vocabulary. The structure of typical computational lexica is not relational: they are basically sequences of entries, each of which is a set of features associated with the head lexeme. See, for instance, the entry of McCord's Slot Grammar lexicon cited in (6) above, or the lexicon of Pereira's well-known Chat-80 (Pereira 1983).

This approach can be loosely called non-structuring.[9] (Note that this state of affairs recalls what has been observed in the development of modern semantics:

[9] To be sure, this approach presupposes some degree of structuring; we, however, allow ourselves to use the term to emphasize the insufficiency of this structuring.

initially, linguists believed that one could do with unstructured sets of semantic features; later, they embarked on a research path with a heavy emphasis on hierarchization of semantic representations, such as "logical form"; finally, we see the advent of highly sophisticated semantic networks. One might expect a similar evolution in the organization of the lexicon.)

The non-structuring approach has certain advantages, the main one being the possibility of isolating, in the course of actual text processing, a mini-fragment of the whole lexicon, sufficient for a specific task and much easier to manipulate. In order to analyse or synthesize a text, a system using a non-structured lexicon will have to deal only with the lexemes actually present in this text, without being forced to reach out to other entries in the lexicon. Computationally, this is very practical — but then the system is restricted to scanty lexical information. The result is that a non- structured lexicon prevents the system from performing high-level linguistic jobs, which involves looking for new words and set phrases, making subtle choices, controlling style, and the like.

In sharp contrast to this, the ECD is consistently structured; this means that, with respect to its meaning and its cooccurrence, a lexeme is specified in terms of other lexemes. The necessity of such structuring manifests itself in many tasks of which we will consider the following two, considered previously: lexicalization and paraphrasing.

The lexicalization of a SemR, i.e. of a semantic network, is carried out due to the fact that a semantic unit labeling one of the network's nodes is actually the sense of a word — a lexeme, which is fully determined in the lexicon by a hierarchical set of properties. More specifically, one of these properties is the participation of the lexeme in question in the definitions of other lexemes, as well as the presence of certain lexemes in its own definition. The screening of semantic nodes with their properties will hopefully allow the model to carry out successfully the analysis of SemRs in a given language — in view of their lexicalization. We think that it will be possible to generalize this particular way of exploiting a structured lexicon to other levels of linguistic computation.

The paraphrasing of a DSyntR, i.e. of a DSynt-tree, which is so important to ensure the transition between the Sem- and the DSynt-levels (see 2.3.2), is of course possible only through Lexical Functions, which obviously represent another aspect of lexical structuring.

To conclude, we would like to mention that the whole of the information presented in a lexicon of the ECD format may sometimes seem too detailed with regard to certain kinds of computational applications. However, we do not see why, in computational as well as in more traditional linguistics, a lexical description could not be as complete, as consistent and as detailed as possible, so that it could be employed (even if partially modified) in all imaginable contexts of research and/or application. Such a description ideally contains all the lexical information that could be necessary for any task; for a specific task, one can extract from this source as much information as is deemed sufficient. A description of a linguistic phenomenon formulated in a theoretically consistent and exhaustive way is valuable for computational linguistics applications because it is "reusable": it can be utilized in many different applications. From this point of view, the rigorous description of even one lexeme in a precise and formal framework appears as an improvement, in itself, on what has been achieved in this domain.

Acknowledgments

Thanks to Geoffrey Hird for his comments on an earlier draft of this paper and to Kathleen Connors and Lidija Iordanskaja for their remarks and suggestions concerning the final draft. We are also grateful to two anonymous reviewers of *Computational Linguistics*, whose well-directed criticisms helped us considerably in reshaping our paper.

References

Boyer, Michel and Lapalme, Guy. 1985 Generating Paraphrases from Meaning- Text Semantic Networks. *Computational Intelligence* 1(3-4): 103-117.

Dowty, David R. 1979 *Word Meaning and Montague Grammar*. D. Reidel, Dordrecht, Holland.

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

Gross, Maurice. 1975 *Méthodes en Syntaxe*. Hermann, Paris, France.

Lamb, Sydney. 1966 *Outline of Stratificational Grammar*. Georgetown University Press, Washington.

McCord, Michael C. 1982 Using Slots and Modifiers in Logic Grammars for Natural Language. *Artificial Intelligence* 18: 327-367.

McKeown, Kathleen R. 1985 *Text Generation*. Cambridge University Press, Cambridge, England.

Mel'čuk, Igor A. 1974 *Opyt Teorii Lingvističeskix Modelej "Smysl-Tekst"*. Nauka, Moscow, USSR.

Mel'čuk, Igor A. 1981 Meaning-Text Models: A Recent Trend in Soviet Linguistics. *Annual Review of Anthropology* 10: 27-62.

Mel'čuk, Igor A. 1982 Lexical Functions in Lexicographic Description. In: *Proceedings of the VIIIth Annual Meeting of the Berkeley Linguistic Society*, Berkeley: UCB, 427-444.

Mel'čuk, Igor A. 1988. *Dependency Syntax: Theory and Practice*. SUNY Press, New York.

Mel'čuk, Igor A. *et al.* 1984 *Dictionnaire Explicatif et Combinatoire du Français Contemporain. Recherches Lexico-Sémantiques I.* Presses de l'Université de Montréal, Montréal, Canada.

Mel'čuk, Igor A. and Žholkovsky, Alexander K. 1984 *Explanatory Combinatorial Dictionary of Modern Russian*. [in Russian] Wiener Slawistischer Almanach, Vienna, Austria.

Pereira, Fernando C.N. 1983 *Logic for Natural Language Analysis*. Technical Note 275, SRI International, Menlo Park, California.

Schank, Roger C. 1972 Conceptual Dependency: A Theory of Natural Language Understanding. *Cognitive Psychology* 3: 552-631.

Sgall, Peter. 1967 *Generativiní Popis Javyka a Ceská Declinace*. Academia, Prague, Czechoslovakia.

Wierzbicka, Anna. 1980 *Lingua Mentalis*. Academic Press, New York, New York.

APPENDIX

An illustrative list of lexical functions

The argument of a LF is called its key word.

A_0
Derived adjective with the same meaning as the key word:

A_0 (*city*) = *urban*

A_1, A_2, A_3, . . .
Typical qualifier for the first, second, third, . . . actant of the key word:

A_1 (*surprise*) = *surprised*
A_2 (*surprise*) = *surprising*

Cont
Means 'continue':

$ContOper_1$(*contact*) = *stay, remain, keep* [*in contact with*]

(Here is a typical example of the way LFs can be "composed" to describe more complex new relations between lexical items.)

Contr
Contrastive term:

Contr (*top*) = *bottom*
Contr (*night*) = *day*

$Conv_{ijkl}$
Conversive, or a lexeme denoting a relation that is the converse of the one expressed by the argument of the LF. The indices i, j, k, l indicate the type of argument permutation which is effected:

$Conv_{21}$ (*more*) = *less*

Theo is MORE religious than Oeht <=> Oeht is LESS religious than Theo

$Conv_{3214}$ (*sell*) = *buy*

Ivan SOLD his soul to the Devil for three bucks <=> The Devil BOUGHT Ivan's soul from him for three bucks

Gener
Generic word:

Gener (*anger*) = *feeling* [*of anger*]
Gener (*pain*) = *sensation, feeling* [*of pain*]

$Labor_{ij}$
Semantically empty word which takes the actants i and j as its subject and direct object, respectively, and the key word as its indirect object:

$Labor_{12}$ (*esteem*) = *hold* [*someone in (high) esteem*]

Liqu
Means 'liquidate', 'eliminate':

Liqu (*meeting*) = *adjourn*

Mult
Standard word for a collectivity:

Mult (*ship*) = *fleet*

$Oper_1$, $Oper_2$, . . .
See 2.3.2 above. Semantically empty verb which takes the first, second, . . . actant of the key word as its subject and the key word as its direct object:

$Oper_1$ (*attention*) = *pay*
$Oper_2$ (*attention*) = *attract*

Magn
Means 'very', 'intense', 'intensely':

Magn (*escape*) = *narrow*
Magn (*bleed*) = *profusely*

S_0
Derived noun with the same meaning as the key word:

S_0 (*honest*) = *honesty*

S_1, S_2, S_3, . . .
Typical noun for the first, second, third, . . . actant of the key word:

S_1 (*sell*) = *vendor*
S_2 (*sell*) = *merchandise*
S_3 (*sell*) = *buyer*
S_4 (*sell*) = *price*

Syn, $Syn_{\subset}$, $Syn_{\supset}$, $Syn_{\cap}$
Synonymous and quasi-synonymous ("$\subset$" means 'narrower'; "$\supset$" means 'broader'; "$\cap$" means 'intersecting'):

Syn (*calling*) = *vocation*
$Syn_{\subset}$ (*respect*) = *veneration*
$Syn_{\supset}$ (*keen*) = *interested*
$Syn_{\cap}$ (*escape*) = *break out* [*of*], *run away* [*from*]