AN EDITOR FOR THE EXPLANATORY AND COMBINATORY DICTIONARY OF CONTEMPORARY FRENCH (DECFC)

Michel Décary
Guy Lapalme

Département d'Informatique et Recherche Opérationnelle
Université de Montréal
C.P. 6128, Succursale A
Montréal, Quebec H3C 3J7
Canada

This paper presents a specialized editor for a highly structured dictionary. The basic goal in building that editor was to provide an adequate tool to help lexicologists produce a valid and coherent dictionary on the basis of a linguistic theory. If we want valuable lexicons and grammars to achieve complex natural language processing, we must provide very powerful tools to help create and ensure the validity of such complex linguistic databases. Our most important task in building the editor was to define a set of coherence rules that could be computationally applied to ensure the validity of lexical entries. A customized interface for browsing and editing was also designed and implemented.

1 INTRODUCTION (WHAT IS THE DECFC?)

The *Dictionnaire Explicatif et Combinatoire du Français Contemporain* (DECFC) is an attempt to provide a formally complete and adequate description of the French lexicon. It is based on the "Meaning–Text" theory (Melčuk 1973), which was the source of several projects in natural language processing and especially in automatic translation. One of the most important principles of the DECFC is that the greater part of the information needed to describe a natural language should be compiled within the dictionary. This is in contrast with the current practice of giving preference to grammars.

> Far from being a modest and secondary appendix to a good grammar, the dictionary becomes the main (in effect, only) basis of all grammars and, in general, of all linguistic descriptions (Melčuk 1973).

As the dictionary is used as the basis of linguistic description, it becomes a very complex database for different types of information with many links and constraints. A well-defined methodology is needed to build such a dictionary; adequate computerized tools are also needed, otherwise the task becomes almost impossible. Our editor is an attempt to provide such a tool.

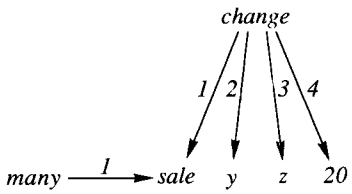
1.1 OVERVIEW OF THE MEANING-TEXT THEORY

To really understand the goal of the DECFC, it is important to put it in the perspective of the Meaning–Text Theory (MTT) for which it is the foundation. We now briefly sketch the MTT and show its implications for the dictionary. A comprehensive presentation can be found in Melčuk (1973); a more computer science–oriented view is Boyer and Lapalme (1984), where it is used as the basis of a system for generating paraphrases. Melčuk and Polguère (1987) describe the formal approach that underlies the construction of the DECFC.

As stated by Melčuk, the purpose of the MTT "consists in establishing correspondences between any given meaning and (ideally) all synonymous texts having this meaning." The MTT is essentially descriptive and is not concerned with procedures for moving from meanings to texts and vice versa. In MTT, an utterance u is represented at seven levels:

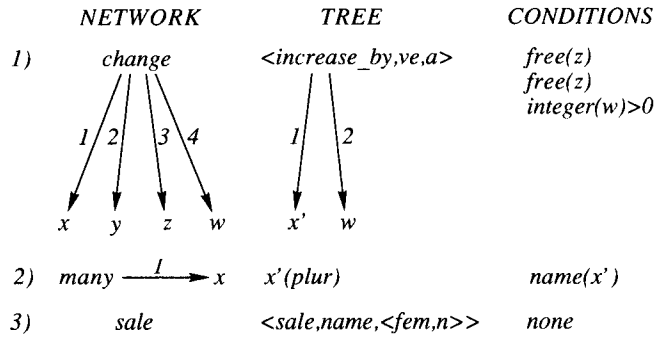
- the *Sem(antic)R(epresentation)*, which is a linguistic object, an utterance in a pictorial language. Its role is to represent a class of synonymous sentences, weeding them out of all their syntactic information. A semantic graph is a connected directed graph with labeled nodes and arcs. The node labels are either predicates or names of objects. The arc labels are integers; the arc labeled by i leads to the ith argument of the predicate.
- the *D(eep-)Synt(actic)Representation* is a tree whose nodes are labeled with "meaningful lexemes" of u.
- the *S(urface-)Synt(actic)R(epresentation)* is also a tree, but its nodes are labeled with all actual lexemes of u.
- MTT also introduces the *Deep* and *Surface* representations for morphology and phonology.

For example, consider the following simple network,

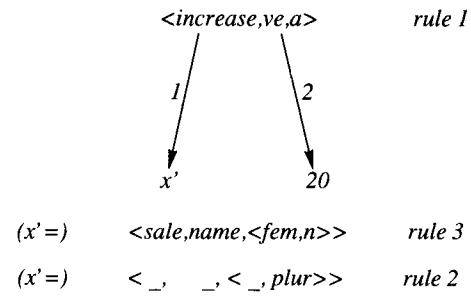


which represents the sentence "The sales have increased by 20 units." This network is incomplete because there is no indication about the time of the action and about the determination of the sales (do we know exactly what sales we are talking about?). To transform a *SemR* to a *DSyntR*, we have to cover all the nodes of the *SemR* with "network schemata" found in the DECFC and merge the corresponding trees also given by the DECFC.

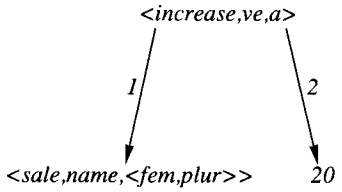
Suppose now that our dictionary contains only of three definitions composed of the "network schema," the corresponding tree and the conditions under which the definition can apply to a network.



The first rule indicates that if the *y* and *z* arguments are free variables and that *w* is a positive integer, then the *change*(*x*,*y*,*z*,*w*) predicate can be transformed to the tree corresponding to *x' increase_by w*. *x'* corresponds here to the *x* node obtained by using the dictionary definitions. In this case, the *sale* node gives the node ⟨*sale*,*name*,⟨*fem*,*n*⟩⟩. So applying rules 1), 2), and 3) we obtain the three following trees:



Boyer and Lapalme (1984) describe a variant of the classical unification algorithm for merging these trees (where underlines indicate free variables); we get the following *DSyntR*:



We could continue by applying similar rules to transform between the different levels of representation. The transformation rules have to be very precise, and thus the dictionary becomes a complex database involving many relations between words. A formal approach to the dictionary building is needed because the transformations are essentially automatic and data-driven (in our case dictionary-driven). Building the DECFC is an enormous task because it has to deal not only with the lexemes but also with their intricate relations. The appendix gives the full entry for the lexeme *respect* where it can be appreciated that a tool to help in writing and checking entries and relations would be very useful.

However, the basic goal of the DECFC project is not to create a version of the dictionary that could be used by a computer program (for text analysis or generation, for instance) but rather to edit a printable version for human readers. This means that the way information is presented and edited is not always as formal as could be expected from the theory. For instance, definitions of lexemes, which are represented by semantic networks in the theory, are represented in the DECFC by French sentences derived from the network by following a set of principles (but no formal rules). Despite this, the DECFC is built applying a systematic methodology that could in principle be programmed, the main difference being that the information is not always as explicit as it could be. A lexicologist could, for instance, retrieve the exact semantic network from a DECFC definition, but a computer could not. Furthermore, the DECFC includes some redundancies that would not be needed from a theoretical point of view. Even if there are a few differences between the DECFC and the formal lexicon of the MTT, there is a clear and direct correspondence between the two.

Our discussion emphasizes the validations the system has to ensure and the way to implement them. We first give a description of the DECFC structure. We then concentrate on specific problems of coherence and verification through the dictionary. We finally discuss the way lexicographers can interact with the system through a specialized interface to the editor. Melčuk and Polguére (1987) give more details about the structure of this explanatory and combinatorial dictionary. We only give here what is relevant for our system.

2 Contents of the DECFC

Each lexical entry of the DECFC gives two basic kinds of information: semantic information, a systematic and rigor-

ous description of the meanings of words and phrases, and combinatorial information, a description of the way individual words can combine syntactically and lexically. The DECFC is thus an "explanatory" and "combinatorial" dictionary.

2.1 UNITS OF LEXICOGRAPHIC DESCRIPTION

The basic unit of description is the **lexeme.** An entry corresponds to only one lexeme, and a lexeme is described in only one entry. A lexeme is a single word taken in only one precise meaning, which is a practice that differs from the traditional dictionaries. Each exception of a word which, in a traditional dictionary, would be described within the same article is given a separate description because each lexeme has its own semantics and combinatorics.

But obviously, certain lexemes have similar meanings and behavior; to take this into account, the DECFC uses the concept **vocable.** A vocable is the set of all lexemes that have the same form (they are written the same way) and share a nontrivial meaning component. Within a vocable, each lexeme is numbered according to its meaning proximity with the other lexemes. The DECFC uses three levels of numbering, corresponding to three levels of semantic distances. Because lexemes have different meanings and different syntax, distinguishing lexemes from each other is essential for the theory. However, the fact that those numbers also represent a certain measure of meaning proximity is a redundancy introduced only for human readers of the dictionary, as this is a current practice of most printed dictionaries. Nevertheless, there is a well-defined methodology for attributing those numbers.[1]

For example, the vocable *RESPECT* contains four lexemes numbered as follows:[2]

respect I: attitude émotionelle favorable
e.g., le respect pour les parents
respect II.1: fait de tenir compte des prescriptions
e.g., le respect des lois
respect II.2a: fait de tenir compte de quelque chose en ne lui portant pas atteinte
e.g., le respect de la propriété des parcs
respect II.2b: fait de ne pas porter atteinte à quelque chose
e.g., le temps n'a pas de respect pour quiconque

2.2 SEMANTIC INFORMATION

Semantic information contained in a DECFC entry opens the possibility of building a global semantic network corresponding to the meaning of a sentence according to MTT.

Lexemes can have semantic arguments, expressing the various actants involved in the meaning defined by the lexeme, and syntactic arguments realizing the semantic arguments in a text. In the DECFC semantic arguments are represented by capital letters, and syntactic arguments are represented by numbers. When the meaning is represented by a network, arguments are explicitly represented in the network. However, in the "text" version of definitions, we must first specify the number and name of semantic arguments and then use those arguments in the definition. The first part is called the **definiendum** and the second the **definiens.** For example, the definition of *respect I* looks as follows:

Respect de X envers Y =
attitude émotionelle favorable de X à l'égard de Y . . .

The expression to the left of the equality sign (**definiendum**) expresses the fact that *respect I* uses two semantic actants. Theoretically, it would have been enough to express it as *respect*(X,Y); however, as it is intended for human readers, the DECFC prefers to include the argument's name into an expression closer to the normal syntax of language. The semantic informations must also take into account the following:

- Definitions must never use ambiguous words: each lexeme appearing in a definition is distinguished by its number. But during the writing of the dictionary, it is not always possible to know these numbers before the corresponding lexemes have been defined. So our editor will have to deal with these "vague" references that later can be made more precise when more information becomes available.
- Definitions must not create "vicious circles." In other words, a systematic replacement of lexemes by their definition, and this recursively at all levels, must never use the initial lexeme. This implies that there are lexemes that we will not be able to define; these lexemes would then be identified as semantic primitives. One of the goals of the work on the DECFC is to find these primitives.

2.3 THE GOVERNMENT PATTERN

The government pattern describes the way semantic actants can be syntactically manifested in a correct sentence. It is a table where each column corresponds to a semantic actant and specifies for it the corresponding syntactic actant and the different ways it can be expressed. Take for example the lexeme *respect I*, whose government pattern is given in Figure 1.

For example, the expression "X = 1" in column 1 means that the semantic actant referred to as "X" in the definition can be expressed syntactically by a dependency relation

X = 1	Y = 2	Z = 3
1. de N	1. de N	1. pour N
2. Aposs	2. pour N	
3. A	3. envers N	

Figure 1 Government Pattern for the Lexeme *respect I.*

numbered "1." This relation can have the form *de N*, which means *de* (*of*) followed by a noun. Row 2 of column 1 expresses the fact that *X* can be expressed by a possessive pronoun (as in *his respect*).

In the following sentence:
Le peuple respecte le président pour son courage,
X = "le peuple,"
Y = "le président,"
Z = "le courage du président."
Looking at the government pattern, we can deduce that it is correct to say:

for the first actant:
le respect *du peuple* . . . ,
son respect . . . ,
le respect *populaire* . . . ,

for the second actant:
son respect *du président*,
le respect *pour le président*,
le respect *envers le président*,

for the third actant:
le respect du peuple envers le président *pour son courage*.

The government pattern is supplied with so-called restrictions; these are constraints on the combination of forms occurring in different columns.

For *respect I*, the restrictions are

1) C1.1 + C2.1
   C2.2 + C3.1 are impossible
   C3 without C2
2) C1.3 + C2.1: is not desirable.

The symbol "C" followed by a number refers to a column in the government pattern; if followed by a dot and a number, it refers to a row in that column. The symbol "+" means "together with." The first expression in the example could thus be read "to have the first realization of the first syntactic actant together with the first realization of the second actant is impossible" (ie: the sentence **Le respect du peuple du président* is not possible). Other examples are:

*Le respect au peuple du président[3]
*Le respect du peuple pour le président pour son courage

are impossible, while

?Le respect populaire au président

is not desirable.

2.4 LEXICAL COMBINATORIC

Usually, the meaning of a group of lexemes is the combination of the meanings of the original lexemes. For example, the meaning of *respect du peuple* is the combination of the meanings of *respect* and *peuple*. But it often happens that the resulting meaning is not this combination, for example *pomme de terre* is not the combination of the meanings of *pomme* and *terre*.

The lexical combinatoric of the DECFC describes the syntax and meaning of those idiomatic or semi-idiomatic expressions containing the lexeme. The authors of the DECFC have isolated about 50 elementary meanings (with specific syntactic roles) the terms of which, taken either alone or in combination, can express the meanings of many semi-idiomatic expressions. These elementary meanings and their legitimate combinations are called *lexical functions*. Lexical functions also include a set of "substitution functions," which express semantic or syntactic relations between lexemes. Examples of lexical functions are:

Magn, meaning *very intense* when applied to *appétit*, defines the expressions [*appétit*] *de loup* and [*appétit*] *gargantuesque*.
Oper$_1$ represents a semantically empty verb taking the first actant of the head lexeme as its grammatical subject and the lexeme itself as its direct object. When **Oper**$_1$ is applied to *respect* it defines *avoir* [*du respect*] or *éprouver* [*du respect*].
Func$_0$ represents a semantically empty verb taking the lexeme as its subject. When applied to *feu*, for instance, it yields [*faire rage*].
Syn represents synonyms of the lexeme.
Anti represents antonyms of the lexeme.

Each lexical function along with its results is expressed in the DECFC following a specific syntax. Furthermore, there are semantic constraints on the results, and our editor has to enforce them.

2.5 OTHER INFORMATION

The dictionary also gives other morphologic information, such as syntactic category, gender, etc. It also describes a few syntactic peculiarities, such as the position of an adjective around a noun. And finally, a list of examples of use of the lexeme is given with other textual information (like the one usually found in traditional dictionaries but of not special interest for the formal part). This information is not of any real use for the automatic processing of natural language, but it helps the human reader.

3 THE NEED FOR AN EDITOR

Lexicographers working on the DECFC were faced very early with the problem of verifying the correctness of lexical entries. Because of the very complex structure of lexical information and the many links between various pieces of information, manual verification becomes nearly impossible as soon as the number of lexemes in the dictionary reaches a few hundred. In many cases, a small modification in the description of one lexeme may require checking many others to ensure its validity. These verifications are of two kinds: a *syntax* verification that ensures that each piece of information respects the formal language of

representation used in the DECFC, and a *coherence* verification that makes sure that no piece of information is in contradiction with another and that some general rules of construction (e.g., the avoiding of circular definition) are respected.

3.1 SYNTACTIC VERIFICATION

Syntactic verification is actually not a very difficult problem as it only has to deal with local rules that bear no relation outside the point of verification. These verifications are also the easiest ones to do by computer. A large part of ensuring the syntactic correctness is done simply by the way the editing process is constrained within the system. The user often has to select keywords in a menu or fill a predefined template. Less constrained sections of information are checked using an appropriate grammar of representation for this kind of information. Nevertheless, syntactic correctness is essential, as a formal and structured representation of information is a prerequisite for defining and applying more complex coherence rules.

3.2 COHERENCE VERIFICATION

The problem of ensuring coherence in the DECFC is a very complex one. It is not limited to the simple problem of conflict or contradiction between pairs of information, but it leads to a broader range of difficulties. Among them is the fact that much information in the dictionary is expressed with words, the same words being described in the dictionary. This means that a complete verification of coherence would have to ensure that each word in the dictionary is used in accordance with its own description.

Furthermore, overall coherence is only verifiable when the whole dictionary is completed. Before that point we are always dealing with incomplete information. For example, very often a word inside a definition is not described in the dictionary when it is first mentioned. But the sole fact that a word is used somewhere in the dictionary already gives information about this word. In this sense, to check coherence implies having to know what is correct and incorrect about actual information and also an ability to construct deductions about words that are not actually in the dictionary but that will eventually be.

Before building our intelligent editor for the DECFC, we first identified and formalized coherence rules and then defined how they could be implemented. We show only the study of two coherence problems: synonymic relations and circularity of definitions.

3.2.1 COHERENCE OF SYNONYMIC RELATIONS

Synonymic relations are used to name different kinds of lexical relations where lexemes share the same (or approximately the same) meaning. This includes the following (the symbol used by the DECFC is shown inside parentheses):

- *synonyms* (*SYN*): same meaning and same syntactic category
- *antonyms* (*ANTI*): same meaning except that the definition of one of the two lexemes includes a negation
- *syntactic derivates*: same meaning but different syntactic categories such as: *Nominalization* (*So*), *Verbalization* (*Vo*), *Adverbialization* (*Advo*), *Adjectivization* (*Adjo*)
- *Converses* (*CONV*): same meaning and same syntactic category, but the order of syntactic actants with respect to semantic actants is different.

Furthermore, the DECFC distinguishes four different degrees in which those relations can occur:

- *Exact* (=): same meaning
- *Larger* (>): the meaning of the first lexeme includes the second
- *Smaller* (<): the meaning of the first lexeme is included in the second
- *Intersection* (<>): the meanings of the two lexemes intersect

The degree of synonymic relations is defined manually on the base of the comparison of two semantic networks. This information is thus redundant and could be computed automatically if the networks were available. It is mostly intended for human readers. However, as synonyms express a direct relation between definitions of lexemes, they can be used by our editor to check overall coherence of definitions without having to rely much on the definitions themselves.

Synonymic relations are subject to numerous rules of coherence. Apart from the more general rule of coherence between relations themselves, which we will look at more closely here, some others were studied. For instance:

- Synonymy and definition: depending on the kind of relation, a lexeme in a synonymic relation with another must, can, or must not appear in its definition. This problem is simpler when dealing with semantic networks. For example, the semantic network for a larger synonym of a word A must be included in the semantic network of A (this is the definition of larger synonyms!). When the definition is expressed in terms of a sentence representing the network, the rules are a little trickier. For example, if a word A has exact or larger synonyms, one of the synonyms will have to appear in the definition of A. Exact synonyms will be preferred to larger ones; derivates will be preferred to converses, etc . . .
- Synonymy and numbering: both numbering of lexemes and synonymous relations between lexemes are measures of semantic proximity and thus must comply with some identified rules. For example, if two lexemes differ only by the third level of numbering, they must be synonyms. These rules are derived from the methodology used to define numbering and the methodology used to identify synonyms (both methodologies being based on an analysis of the semantic network).
- Synonymy and government: there is a relation between sharing meaning and sharing government. Among other

things, the number of syntactic actants of each lexeme must be the same (exact synonyms have the same number of semantic arguments).

The reason that synonymic relations are so present in the problem of verifying coherence is they express a direct and explicit link between units of description inside the dictionary. Furthermore, these relations possess two important properties:

- each relation has an opposite;
- with some limitations, relations can be composed to form new relations (i.e., If A is in relation with B and B is in relation with C, in most cases there is a relation between A and C).

The most important consequence of those properties is that from a set of correct relations it is possible to validate any new relation given by the user or even to propose a list of new relations. Of course, we have to define clearly what is the opposite of each relation and how relations combine to form new ones. For instance, if we have the relations $SYN_{>}(huge) = big$ and $ANTI(big) = small$, those rules of derivation will be able to verify that $ANTI_{<}(small) = huge$.

The problem is easy when we deal with exact synonyms, but it becomes fuzzy when meanings only intersect. For example, if we have

$$SYN_{>}(huge) = big$$
$$SYN_{>}(gigantic) = big$$

(i.e., the meaning of *huge* contains the meaning of *big*, and the meaning of *gigantic* also contains the meaning of *big*), then what is the relation between A and C? Is it possible that no relation exists? Is $SYN_{>}(huge) = gigantic$ more probable than $SYN_{<>}(huge) = gigantic$? To answer those types of questions, each case (relations and degree) has been studied and the results are shown in Figures 2 and 3. What appears clearly is that the type and degree of synonymic relations are independent in regard to the result of relation composition. Figure 2 shows how types of relation are composed. This always gives a unique result (except in the case of *CONV* and *CONV*, where the result could be *SYN* in the trivial case where the second converse relation puts the syntactic arguments back in place).

Not all compositions give an existing relation as their result. We have used the symbol *Comp* in that case (e.g., if $ANTI(A)=B$ and $CONV(B)=C$, then the relation resulting from the composition is simply $ANTI(CONV(A))=C$). Those compound relations are not indicated in the dictionary.

But even then, it is important to consider them because when composed with another relation they can eventually simplify and return a single relation. For example, if we have the following relations: $CONV213(buy) = sell$, $So(sell) = sale$, $CONV213(sale) = purchase$, no compositions can be made according to Figure 2. However, combining the three relations would lead to $CONV213(So(Conv213)buy))) = purchase$. This larger expression can be reduced to $So(buy) = purchase$ if some rules of simplification are used (commutativity, elimination, simplification of derivates, etc.). A set of rules for simplifying composition of synonymic relation was found, and some mathematical proofs of those rules are given by Décary (1986). Those rules when applied, for example, to a large compound like:

$$CONV213(Vo(So(ANTI(CONV213(A))))) = D$$

would simply give

$$ANTI(A) = D$$

when simplified (If A is a noun).

Simplifying these synonymic relations is important, because it helps validate any new relations and it gives the system the capacity to generate hypotheses about semantic links between lexemes. These hypotheses could then be used by lexicographs to build new entries or to correct existing ones. According to MTT, the meanings of lexemes are built upon the meaning of other lexemes, and there is a

		R2						
		Syn	**Anti**	**Conv**	**So**	**Vo**	**Ao**	**Advo**
	Syn	Syn	Anti	Conv	So	Vo	Ao	Advo
	Anti	Anti	Syn	Comp	Comp	Comp	Comp	Comp
	Conv	Conv	Comp	Conv?	Comp	Comp	Comp	Comp
R1	**So**	So	Comp	Comp	So	So	So	So
	Vo	Vo	Comp	Vo	Vo	Vo	Vo	Vo
	Ao	Ao	Comp	Comp	Ao	Ao	Ao	Ao
	Advo	Advo	Comp	Comp	Advo	Advo	Advo	Advo

Figure 2 Composition of Synonymic Relations.

		R2			
		exact(=)	smaller(<)	larger(>)	intersection(<>)
	=	=	<	>	<>
R1	<	<	<	=,<,>,<>,*	<,<>,*
	>	>	=,<,>,<>	>	>,<>
	<>	<>	<,<>	>,<>,*	=,<,>,<>,*

Figure 3 Composition of Degrees in Synonymic Relations.

set of semantic primitives that cannot be defined. The DECFC lexicologists think that the surest way to find those primitives is to start by defining complex words and working their way down to simpler lexemes. In doing so, lexicologists try to use a clear methodology, but they also have to rely on intuition. Identification of definition as well as of synonymic relations relies on intuition at some point in process. So, when a lexicologist enters a new synonymic relation, he or she expresses by a different mean, the same intuitions about equivalence and hierarchy of meaning as in definitions. The composition of synonymic relations done in our editor has to generate all possible consequences from such choices. For instance, if a lexeme uses a word in its definition, our editor could find out that this word has to be a near synonym and, by that, generates a list of relations that would have to be true. Lexicologists could then see clearly the scope and consequences of their intuition, which would result eventually in a better structuring and integration of those intuitions.

As synonymic relations also have degrees, those must be combined as well. Figure 3 expresses the way relation degrees are combined. This is clearly different from relation composition, because in many cases more than one result is possible (the symbol * indicates the possibility of having no relation). It is important to note here that the inclusion of meanings does not follow the same rules as the inclusion of sets. For example, if A contains B and B contains C, then A contains C if A,B,C are sets, but it is not always true for synonymic relations. This is due to the fact that a synonymic relation exists only if the meaning shared is *important enough* (i.e., both definitions share a nontrivial[4] part). Thus, it is possible that C is too different from A to lead to a synonymic relation.

This shows that when near-synonymic relations are combined, we do not always get a single result but often a set of possibilities. This is sufficient for the purpose of ensuring the coherence of new information entered in the system. To be more precise would involve comparing the definitions of both lexemes, considering the numbering of the lexemes within their vocable and using statistical data.

We have now defined some properties of synonymic relations that are helpful in checking if relations in a given state of the dictionary are coherent. We still have to know how this verification takes place.

If we take all synonymic relations in the dictionary and represent them as two nodes linked by an arrow, we obtain a set of networks like the one in Figure 4. From there, we can check any new relations entered in the system by considering the network in which the new relation appears (this could imply the merging of two networks). We look at

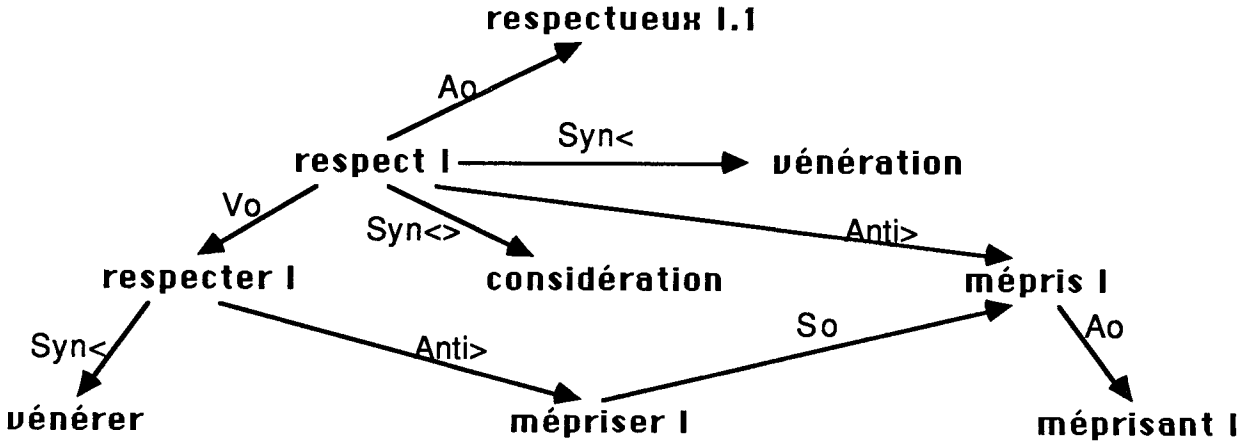


Figure 4 Network of Synonymic Relations for *respect I*.

all the paths linking the two nodes in the new relations (even the paths going in the reverse direction of an arrow as we can define an inverse for all relations). We then combine the relation on each of these paths using the rules we defined. The relation is coherent if no contradiction occurs between the new relation and what is obtained on any of the paths.[5]

Using this method, our editor ensures that an overall coherence is maintained between those relations, which is close to impossible to do manually. Now we look at another kind of coherence verification: circular definition.

3.2.2 AVOIDING CIRCULAR DEFINITIONS

When the meaning of any word in a dictionary is expressed in terms of other words in the same language, circular definitions become unavoidable. This means that if we take a definition and replace each word by its definition and so on, either the first word (the one we started up with) is found somewhere in the process (we call this **strong circularity**) or we have to use a word that has a strong circular definition (we call this **weak circularity**). This is due to the fact that the process of replacing words by their definition is infinite but the lexicon is not. In fact these conditions imply that each definition in the dictionary is circular.

To avoid that situation we accept that some words may not have a definition. They are the **semantic primitives** on which more complex meanings are built. One of the objectives of the DECFC is to find those primitives. The authors of the theory believe that the identification of semantic primitives can only be done by experimentation through the building of an actual dictionary. In respect to that goal, it becomes essential that each case of circularity be detected. Once again, this task is nearly impossible to realize without the help of an automatic tool.

As we have seen, there are two kinds of circularity. But, as weak circularity presupposes the existence of strong circularity in the dictionary, only the latter must be looked for. There are two ways to analyze a definition for that purpose: top-down or bottom-up. The top-down approach consists of trying to find the word being defined in its own definition, and then in the definition of the words used to define it, and so on. The bottom-up method tries to find all the words that are not allowed and then to compare this list with the definition. For example, let's say we want to check if the definition of *eye* is circular. We first build a list of all the words in whose definitions *eye* appears. We then add to this list by doing the same thing for all the words in the list and so on. These two methods give the same result, but the latter has the advantage of generating a list of forbidden words that can be of some use to the lexicographers when writing definitions.

Unfortunately, these simple mechanisms are not enough to get rid of circular definitions. This is because circular definitions are not created when the word being defined is repeated, but more precisely when its meaning is used to define it. This means that using an exact synonym in a definition also causes a definition to be circular. This is true for exact synonyms, but we need some precisions for imperfect ones. Let's look at the four possible cases given that B is

$SYN(A)_{=}$, using B to define A is forbidden;
$SYN(A)_{>}$, using B to define A is perfectly acceptable because B has a more simple meaning than A (i.e. the meaning of A is B plus something else);
$SYN(A)_{<}$, using B to define A is forbidden because B is more complex than A;
$SYN(A)_{<>}$, using B to define A is forbidden because B has a part of meaning that A does not have. On the other hand, it would not create a circular definition but an incoherent one;

Thus, to detect circular definitions, we have to take the synonymic relations into account. For example, in the top-down method we check not only for the word being defined but also for exact, larger, and intersection synonyms (and of course antonyms, converse and derivates). As a list of other relations can be deduced from a single set of synonymic relations, many "deductions" are only a list of possibilities. To ensure maximum validation, those relations have to be taken into account so that the system indicates potential circular definitions and explains what are the assumptions. But more important is that a circle can be introduced in a definition simply by adding a new synonymic relation, and this implies watching for circular definition each time a synonymic relation is added or modified.

Circular definition and synonymous relations are amongst the major coherence problems of DECFC we have worked on. Many others have been studied, and still more needs to be defined and analyzed.

4 IMPLEMENTATION

A prototype of the DECFC editor has been implemented on a Xerox 1108 Lisp Machine. We now only present the data structure, the definition of algorithms for verification, and the user interface.

We implemented our editor on a Xerox Lisp Machine because it provides a multi-windowing environment that enables different processes to be going on at the same time on different parts of the screen. For our editor, this is especially important since the validity of information is always related to other pieces of information elsewhere in the dictionary and it is essential for the user to be able to view, on a single screen, different parts of the dictionary. This becomes even more important when the system reports a coherence error: at this point, the user can see the two (or more) chunks of information that are in conflict and can browse elsewhere in the dictionary to really understand what the problem is. Furthermore, users often need to compare different entries or to use older entries as a model for new ones. For those reasons, we need much more than the usual single context view.

We also defined a data structure that reflects the real structure of the dictionary and eases the application of coherence rules to information. The general mechanism used for syntactic verification is a BNF grammar interpreter. When analyzing a section of information, the system applies an appropriate BNF grammar to the data. In some cases, context-free rules are not powerful enough and specific functions are used for verifying contextual rules of formation.

The implementation of coherence rule verification poses two problems. First, the algorithms for our theoretically defined coherence rules, and second, the order of application of the verification rules. In particular, we had to define what actions to take when a piece of information is added, modified, or deleted. For example when a synonymic relation is modified, many things would have to be checked (Is the syntax ok? Did the change remove some incoherences present in the system? Is the relation incoherent? Could the relation be incoherent if some hypothesis were true? Does the relation introduce a vicious circle?). A general flow of control for applying rules was designed but not implemented.

Finally, we designed an interface that allows a useful and efficient use of the system. The interface has the following facilities:

- Browsing is simple and flexible because it follows the structure of the DECFC.
- Information is presented similarly as to what lexicographers are used to seeing (in the actual printed version of the DECFC, for instance).
- The system shows different parts of the dictionary at the same time with few or no constraints.
- The system communicates efficiently with the user by presenting and explaining incoherences and errors, by guiding the user for correction, and by showing general information about the actual state of the dictionary.
- Editing is customized for each section of information (e.g. a definition is not edited like a lexical function).

We defined a model of structured editing in which a structure editor is viewed as a set of specialized editors and a set of specialized selectors. Each node in the structure of the dictionary is then assigned either an editor (when it is an editable information) or a selector (when it is seen as a structure itself). For instance, as lexemes are simply lists of

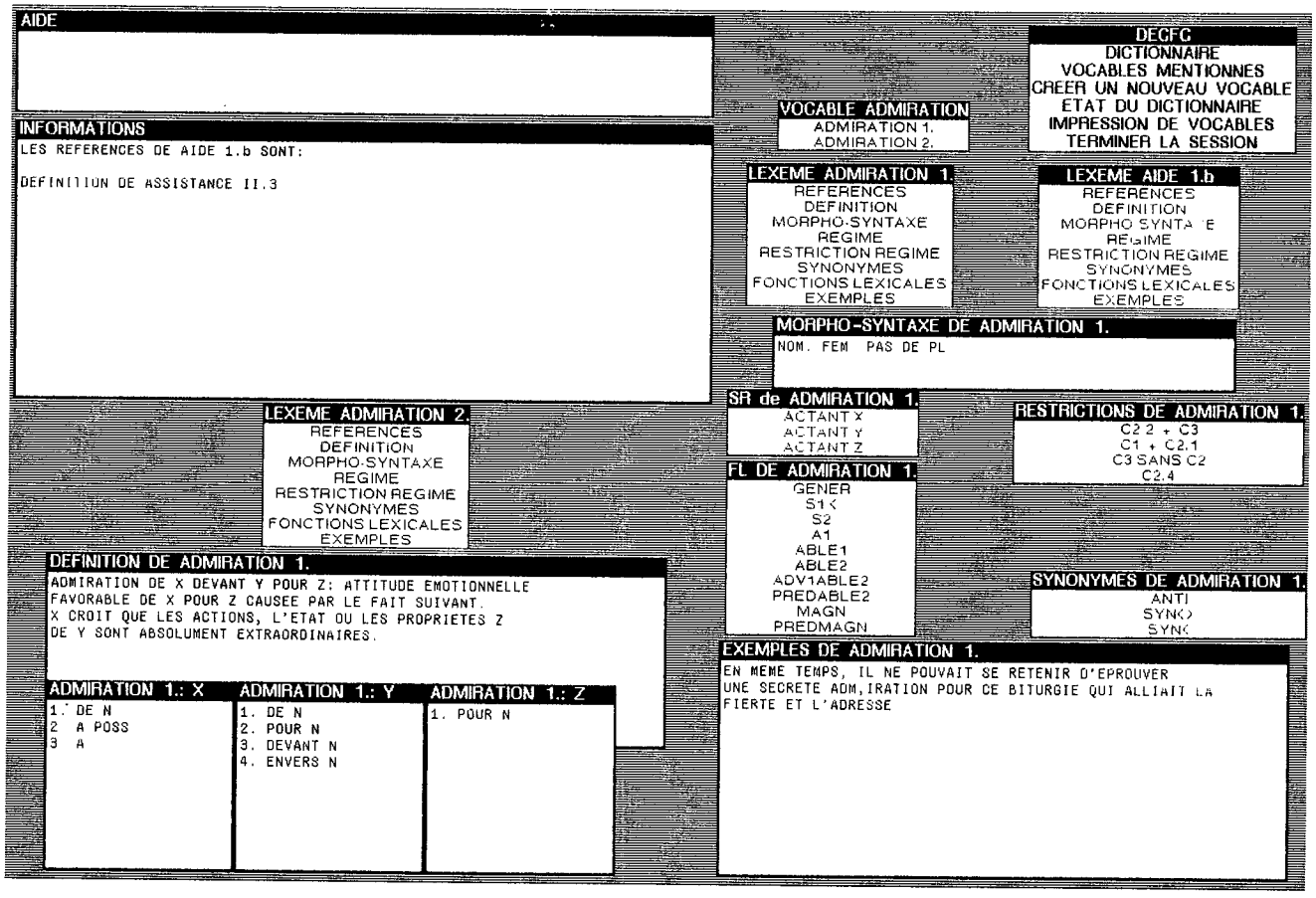


Figure 5 State of the Screen at One Point during the Editing Process.

fields of information, the specialized selector is a menu for selecting or deleting a field. In the case of definition, it is a text editor based on a general template for definition with some specific validation functions.

In our model, a specialized editor is made out of four elements: an interface (a text editor for instance), an output filter (a function that maps the internal representation of the information into the editable form), an input filter (the opposite), and a validation filter (a function that applies coherence and syntactic rules). This reflects the fact that complex structures are often made out of very different substructures. Each of those substructures must then be viewed and edited in a different way.

The actual prototype contains an implementation of this specialized interface. Browsing and editing through the whole dictionary is possible, but only syntactic checking of information and some simple coherence rules checking have actually been implemented. Figure 5 shows a copy of the screen at one point during the editing process.

5 Conclusion

We have described some problems and solutions related to the building of a specialized dictionary editor. Two tasks were identified:

- defining and implementing a set of validation procedures to ensure an overall coherence and well-formedness throughout the dictionary;
- defining and implementing a customized interface for an efficient interaction between lexicographs and the editor.

We have implemented a prototype of the interface that will evolve as lexicographers express new or more specific needs. The team of lexicologists working on the DECFC showed a great interest for the project and the prototype. We are presently discussing with them the possibility of creating a fully functional system for their use.

Some of the validation rules and procedures implemented in our project were discussed in this paper and some not, but these are few compared to the large quantity of coherence rules that are still to be found. Building a complete formal description of something as complex as a natural language has never been done thoroughly. We believe that one of the main reasons for this is the lack of efficient and powerful tools. The DECFC editor is an attempt to provide such a tool for a specific linguistic theory. In other words, our efforts were not directed toward the creation of a linguistic database for natural language processing, but mainly toward providing efficient tools to help linguists achieve good and complete descriptions of natural languages. Those descriptions could then be used as the base of NLP systems.

Acknowledgments

We would like to thank M. Igor Melčuk for his continued interest in our work and his patience in going through previous versions of this report. We thank also Lise Cinq-Mars, who did some very useful programming. Finally, we are grateful to an anonymous referee who helped us very much by pointing out important issues that were not so clearly addressed in a previous version of this paper.

References

Alterman R. 1985 A Dictionary Based on Concept Coherence. *Artificial Intelligence* 25: 153–186.

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

Décary, M. 1986 *Un éditeur spécialisé pour le dictionnaire explicatif et combinatoire du Français contemporain.* Document de travail #181, Dept. Informatique et recherche opérationnelle, Université de Montréal.

Melčuk, I. 1973 Towards a Linguistic "Meaning <=> Text" Model. In: *Trends in Soviet Theoretical Linguistics*. F. Kiefer, ed., Reidel Publishing Co., Dordrecht, The Netherlands.

Melčuk, I. and Polguère, A. 1987 A Formal Lexicon in the Meaning–Text Theory (or How to Do Lexica with Words). *Computational Linguistics* 13: 261–274.

Melčuk, I., A. Clas et al. 1984-1988 *Dictionnaire explicatif et combinatoire du Français contemporain,* Vol. 11, Les Presses de l'Université de Montréal.

Notes

1. The finest level of numbering (letters) indicates a lexical transfer (like metonymy) that is productive in the language. The second level (numbers) shows a lexical transfer that is not generally productive in the language. The first level (roman numbers) indicates any kind of semantic proximity.
2. We use examples in French because our work dealt with French words. Of course, the same principles would apply to another language, and we do not dare invent equivalent examples in English.
3. Sentences preceded by an asterisk are "impossible," while those preceded by a question mark are "not desirable."
4. They share a part that is at least 50% of the whole definition. This part is in the head position and is not a semantic primitive.
5. In doing this we used two assumptions (which we proved in Décary [1986]):
   - paths passing more than one time by the same arrow should not be considered;
   - if the network is coherent before the adding of the new relation no contradiction can appear between two paths but only between a path and the new relation.