A computationally tractable formal language for the encoding of lexical relations

Sylvain Kahane & Alain Polguère

Abstract

We propose a formal device for the encoding of relations holding between lexical units of natural languages, based on the concept of lexical function introduced by Meaning-Text theory. Because the language traditionally used for encoding lexical functions is not fully specified from a formal point of view, we propose a new language which we believe to be computationally tractable. Moreover, this formal language allows us to give a formal basis to the traditional language and to fill its gaps.

1 Introduction

The concept of *LEXICAL FUNCTION* was introduced some 35 years ago (º olkovskij & Mel' uk 1965) to describe two types of relations holding between lexical units (= lexemes[1]) of a natural language: semantic derivations and base-collocate relations. A formal language for describing these relations by means of lexical functions has been developed and used extensively in lexicographic descriptions found in *EXPLANATORY COMBINATORIAL DICTIONARIES* (ECDs) — see Mel' uk & Zholkovsky (1984) (for Russian) and Mel' uk *et al.* (1999) (for French). However, the formal bases of the *ECD ENCODING* of lexical functions have never been made totally explicit, leaving researchers with a formal device that seems loose from both a computational and conceptual point of view. We believe it is essential to address this issue as the concept of lexical function itself has proved to be particularly suited for applications in computational lexicography (see Heid 1996, Fontenelle 1997) and natural language processing (see Iordanskaja *et al.* 1996). The alternative encoding we propose is computationally tractable and makes explicit the inner value and role of lexical functions in natural language, thus making it easier for Meaning-Text outsiders to understand and manipulate them. Because it makes explicit all formal properties of lexical relations, we hereafter refer to the proposed formalism as the *EXPLICIT ENCODING*.

[1] Meaning-Text linguistics distinguishes two types of lexical units, lexemes and phrasemes, the latter being multi-lexemic lexical units (kICK THE BUCKL, kJUMP THE GUNL, etc.). For the sake of simplicity we use here the term *lexeme* to refer to both types.

Section 2 introduces some basic terminology and notation that will be used in the remainder of this paper. Section 3 presents the explicit encoding and Section 4 proposes a formal definition of the ECD encoding, describing it relative to the explicit encoding.

2 Basic notions

2.1 Semantic derivations and base-collocate relations

The Meaning-Text concept of *LEXICAL FUNCTION* (hereafter, LF) is used to model two types of relations holding between lexemes of a natural language: semantic derivations and base-collocate relations. A *SEMANTIC DERIVATION* holds between two lexemes L_1 and L_2 in any of the following three cases: 1) L_1 and L_2 have (roughly) the same meaning —this is either a case of (quasi) synonymy if L_1 and L_2 belong to the same part of speech, or a case of verbal, nominal, adjectival or adverbial translation (in Tesnière's sense); 2) L_1 and L_2 have opposite meanings —the link here is that of (quasi) antonymy; 3) L_2 designates a participant or a circonstant of the situation designated by L_1 —e.g. CRIME is linked by semantic derivations with AUTHOR [*of a crime*] or CRIMINAL, VICTIM, INSTRUMENT [*of a crime*], etc. Such relations between lexemes are called *semantic* derivations as no morphological link needs to exist between lexemes involved, contrary to standard (morphological) derivation. A *BASE-COLLOCATE RELATION* holds between two lexemes L_1 and L_2 when they can be combined to form a semi-idiomatic construction $L_1\ L_2$ —called a *COLLOCA-*

TION— where one of the components —called the *COLLOCATE*— is chosen in a (partially) arbitrary way to express a given meaning and/or a grammatical structure contingent upon the choice of the other component —called the *BASE* of the collocation. Such is the case of collocates expressing intensification (***heavy** bombardment*, ***sharp** contrast*, …), collocates that function as support verbs for the base (***to run** a fever*, ***to stage** a coup*, …), etc.

A relation of semantic derivation holding between two lexemes L_1 and L_2 is always oriented: it associates L_2 to L_1. For instance, MURDERER is the standard name of the first actant of $MURDER_V$: $MURDER_v$ =1st actant⇒ MURDERER. Of course, we can also associate $MURDER_v$ to MURDERER, but in this case we have a different semantic derivation, namely: MURDERER =action⇒ $MURDER_V$. For base-collocate relations, the link is even more obviously oriented. L_2 will be connected to L_1 as one of its collocates, i.e. a lexeme with which L_1 forms a collocation L_1 L_2, of which L_1 is the base. For instance, *heavy* is a collocate (acting as intensifier) of the base *bombardment* in *heavy bombardment*, and not the other way round. We will see immediately below how the fact of modeling lexical relations by means of **functions** accounts for the inherent orientation of such links.

2.2 Formal perspective on LFs

Each LF **f** encodes a subset of the set *R* of all possible lexical relations. The notation $\mathbf{f}(L_1)=L_2$ means that a lexical relation **f** holds from L_1 to L_2. We call L_1 the *KEYWORD* and L_2 the *VALUE* of **f**. Using this functional notation, it is therefore possible to rewrite the two above-mentioned relations holding between $MURDER_V$ and MURDERER as:

1st actant(*to murder*) = *murderer*
action(*murderer*) = *to murder*.[2]

Because several lexemes can be linked to a lexeme L_1 by **f**, **f** is not exactly a function. Rather, each LF **f** corresponds to a map (a true function) **f**• which associates to each lexeme L_1 a **set** of values $\mathbf{f}\bullet(L_1)$, such that for each lexeme L_2 belonging to this set, $\mathbf{f}(L_1)=L_2$. The set $\mathbf{f}\bullet(L_1)$ will be the empty set if no lexeme is linked to the keyword L_1 by **f**.

An *ENCODING OF LEXICAL RELATIONS* is a correspondence φ between the set *R* of all possible lexical relations and a formal language *E*, called the *ENCODING LANGUAGE OF LFS*. A given element **f** of *E* encodes the subset of lexical relations corresponding to **f** through φ.

An encoding $\varphi : R \rightarrow E$ defines a partition of *R*. A given encoding φ_1 is said to be more *GRANULAR* than another encoding φ_2 if φ_1 defines a finer partition of *R* than φ_2, that is, if φ_2 collapses together some LFs which are considered separately by φ_1.

2.3 Linguistic perspective on LFs

Each LF **f** corresponds to a linguistically homogenous set of lexical relations. In other words, if $\mathbf{f}(L_1)=L_2$ and $\mathbf{f}(L'_1)=L'_2$, then L_2 provides the same linguistic features to L_1 as L'_2 to L'_1, that is, the same modification of semantic content and/or the same modification of syntactic behavior. In addition, an LF can be viewed as some sort of "generalized" lexeme (see Wanner 1996:23). Contrary to a true lexeme, an LF **f** is not associated with specific realizations. Its realizations depend on its context of application, that is on the keywords; for each keyword L there exists a set $\mathbf{f}\bullet(L)$ of possible realizations of **f**.

In order to be able to postulate LFs, i.e. generalizations upon lexical relations, the linguistic "content" associated with each particular LF has to remain vague. To illustrate this point, we will take the standard LF of intensification, **Magn**, which is somehow an idealization of "pure" intensification. It is never expressed as such for at least two reasons. First, the intensification of the meaning of a lexeme L is in fact always the intensification of a component of this meaning. For instance, while *deadly* in *deadly combat* applies to the number of casualties, *fierce* in *fierce combat* applies to the actual intensity of the combat.[3] The second reason

[2] To be accurate, we should write the keyword and value of a LF in upper case, as they both are lexemes and not surface linguistic realizations. However, to make our formulas more readable, we will follow here the ECD convention and make use of italics instead.

[3] The case of the LF of "realization" $\mathbf{Real_i}$ is even more striking. The meaning expressed by $\mathbf{Real_1}$(*recommendation*) in *to follow a recommendation* is obviously distinct from the meaning expressed by $\mathbf{Real_1}$(*car*) in *to drive a car.*

why **Magn** is never expressed as such is that values returned by **Magn**(L) themselves correspond to full lexemes that have they own specific meaning. Therefore, even if we should consider that *intense* and *fierce* are both equivalent **Magn** of *combat*, it is still possible to identify semantic differences between the two collocations *intense combat* and *fierce combat* on the basis of the definitional meaning of the two corresponding collocates. Whether for **Magn**, $\mathbf{Real_1}$ or any other LF, it is theoretically possible to opt for more granularity in the encoding and postulate more than just one LF for a given set of lexical links. However, if the concept of LF has to retain its descriptive and generalization power, there is no doubt that each unit of description has to be rather coarse.

2.4 The ECD encoding of lexical relations

The modeling of LF relations offered by Meaning-Text theory provides computational linguistics with a conceptual foundation that we believe should be kept almost as it is. What we propose to deeply revise is **how** LFs should be formally accounted for: what we termed in Section 1 the *ECD ENCODING*. This encoding is made up of a set of about sixty "primitive" LF relations and of rules for combining them (for a presentation, see Mel' uk 1996). While Meaning-Text literature usually describes the basic lexicon of the ECD encoding, it is interesting to note that no detailed account has been made of all the rules governing the combination of the units of this lexicon.

Each LF provided in the ECD encoding pertains to either a semantic derivation or a base-collocate relation, thus suggesting a natural bipartition of the set of LFs: *PARADIGMATIC LFS*, which account for semantic derivations, *vs. SYNTAGMATIC LFS*, which account for base-collocate relations. However, the ECD encoding considers that some LFs have both collocational values V and *FUSED* derivational values, noted / V', such that L+V and / V' are paraphrases. For instance, the ECD encoding accounts for two possible values of **Magn**(*rain*): *torrential* and / *downpour*. This entails that *torrential rain* and *downpour* are paraphrases, both expressing (rain) together with intensification.

The concept of *FUSION*, which has just been examined, reveals something very important about the ECD encoding. Formulas used in this encoding, although linear and apparently homogeneous in nature (**Magn**, $\mathbf{Oper_1}$, $\mathbf{CausOper_1}$, etc.), account for two distinct properties of the lexical relation they encode: a *SEMANTIC CONTENT* and a *SYNTACTIC FRAME* of behavior. Thus, for instance, $\mathbf{IncepOper_1}$(*disease*) = *to contract* [ART ~] encodes the following information:

1. semantic content: *to contract a disease* means (to start [cf **Incep**] to experience [cf $\mathbf{Oper_1}$] a disease);
2. syntactic frame: *to contract* is a verb that takes the noun *disease* as complement and the first actant of this noun as grammatical subject in order to express the above semantic content.

3 Towards an explicit encoding of lexical relations

The explicit encoding describes each LF by means of two distinct formulas: the encoding of the LF's semantic content and the encoding of its associated syntactic frame. We will examine successively (Sections 3.1 and 3.2) each of these two representational components. Notice that all illustrations and examples given in this section are borrowed from a test study we conducted on LF relations controlled by Fr. COLÈRE ((anger)). We selected this lexeme for two reasons: first, it has already been described in great detail using ECD encoding (see Mel' uk *et al.* 1984); second, nouns of feeling control very rich and diverse sets of LF relations and are therefore particularly suited for studying encoding problems.

3.1 Representation of semantic content

The semantic content of an LF is a configuration of predicate-argument relations holding between *PRIMITIVE LF MEANINGS*. These primitive meanings correspond to some of the LFs already identified by Meaning-Text theory. They are primitive in the sense that they will not be accounted for by means of other LF meanings. Primitive meanings are named using the standard symbols found in ECDs (although these symbols refer to the whole LF in the ECD encoding) followed by the argument structure between square-brackets. We list below all primitive meaning that will be used in this paper, using the following template of presentation: <Formal encoding>: (<Semantic gloss>).

Incep[*Arg*]: (*Arg* begins)

Caus[Arg_1,Arg_2]: (Arg_1 causes Arg_2)
Magn[*Arg*]: (*Arg* is intense)
AntiMagn[*Arg*]: (*Arg* is little)
Plus[*Arg*]: (*Arg* increases)
Minus[*Arg*]: (*Arg* decreases)
Fact[*Arg*]: (*Arg* functions)
Real[Arg_1,Arg_2]: (Arg_1 realizes Arg_2)
Manif[*Arg*]: (*Arg* manifests itself)
Manif[Arg_1,Arg_2]: (Arg_1 manifests itself in Arg_2)
Sympt[Arg_1,Arg_2]: (Arg_1 is revealed by Arg_2)
Non[*Arg*]: (*Arg* does not hold)

In addition to primitive LF meanings, some special notations are used to refer to specific meanings:

- #: meaning of the keyword;
- 1, 2, 3, …: first, second, third, … semantic actants of the keyword;
- Ω: other (unspecified) semantic participant.

Some formulas may have to include non standard components. These cannot be formalized and are simply introduced between semantic quotes ((…)). Examples below illustrate the use of special keywords and symbols for arguments. We use actual LF relations controlled by Fr. COLÈRE, whose predicate-argument structure is (colère de X envers Y à cause de Z) ((X's anger towards Y due to Z)):

- Caus[2/3,#]
  [= (Y/Z causes anger)]
  E.g. *Y/Z* ***met*** *X en colère* (lit. (Y/Z-puts-X-in-anger))
- Sympt[#,(poings de X)]
  [= (anger is revealed by X's fists)]
  E.g. *X* ***sert les poings*** *de colère* (lit. (X-squeezes-the-fists-of-anger))

Primitive meanings can be combined to form more complex meanings through predicates recursively taking arguments:

- Caus[1,Minus[Manif[#]]]
  [= (X causes a decrease in the manifestation of anger)]
  E.g. *X* ***étouffe*** *sa colère* (lit. (X-suffocates-his-anger))

Components can also be combined by using the infix operator ^ expressing specification/characterization:

- #^Magn
  [= (anger which is intense)][4]
  E.g. ***rage***

Curly brackets {…} indicate a meaning functioning as context; i.e. it is not part of what is explicitly expressed by the LF. In addition, parentheses (…) are used to specify the scope of operators when needed:

- {1}^(#^Magn)
  [= ([X] such that his anger is intense)]
  E.g. [*X*] ***très en*** *colère* (lit. (very-in-anger))

3.2 Representation of syntactic frame

Syntactic frames contain two types of information:

1. The part of speech of the value. There are only four parts of speech: V(erb), N(oun), A(djective) and Adv(erb).
2. The diathesis of the value (roughly, the complement structure it controls). In accordance with what can be found in the ECD encoding, we limit ourselves to a deep-syntactic level of encoding, which indicates the relative oblicity of the syntactic dependents of the value.

For instance, the formula V[1,2] denotes a verbal value taking the first actant of the keyword as subject and its second actant as first complement, while V[2,1] denotes a conversive structure. Information on the value's diathesis is essential in modeling the relation that link it to the keyword. For instance, in French (as in English), the first argument of a verb is by default the theme of the proposition; consider the communicative contrast between the three values obtained below:

$\begin{pmatrix} \# \\ V[\#,1] \end{pmatrix}$ (*colère*) = *habiter* (to live in) [N=X]

$\begin{pmatrix} \# \\ V[1,\#] \end{pmatrix}$ (*colère*) = *éprouver* (to feel) [ART ~]

$\begin{pmatrix} \# \\ V[2,\#] \end{pmatrix}$ (*colère*) = *encourir* (to incur) [ART ~]

The above examples show the use of complete formulas: they are matrices made up of semantic content (first row) and syntactic frame (second row) subformulas. The first formula expresses that *habiter* is an "empty" verb that takes *colère* as grammatical subject and the first actant of *colère* as complement in order to only express that anger is experienced by someone: *La colère habite Paul.* lit. (anger-lives in-Paul). In ECD encoding terms, it is a case of $\mathbf{Func_1}$. The LFs in the second and third

[4] By convention, we do not indicate the argument of Magn[]: it is identical to its semantic governor (i.e. what appears at the left-hand side of the ^ operator).

examples correspond to $\mathbf{Oper_1}$ and $\mathbf{Oper_2}$, respectively.

For A and Adv values, which are meant to function as syntactic modifiers, the governor is indicated as first element in the argument list, followed by ^. For instance, A[1^] denotes an adjectival value that functions as modifier of the first actant of the keyword:

$\begin{pmatrix} \{1\}\text{^}\# \\ A[1\text{^}] \end{pmatrix}$ (*colère*) = *fâché* (angry).

This example shows that *fâché* is an adjectival constituent that can function as modifier of the first actant of *colère* (*la colère de X* (X's anger) → *X fâché* (angry X)) in order to characterize this actant as being involved in a "situation of anger." This LF corresponds to $\mathbf{A_1}$ in the ECD encoding. We will return to this particular LF in Section 4.1.

Recall the loose concepts of *PARADIGMATIC* vs. *SYNTAGMATIC* LF introduced in 2.4. They happen to be irrelevant in the explicit encoding as the only thing that matters is whether a given LF encodes a semantic derivation or a base-collocate relation. This information is directly available in the explicit encoding: if, as is the case in the above formula, # does not appear in the syntactic frame component of an LF formula, the value of $\mathbf{f}$(L) is a semantic derivation; otherwise, the value is a collocate. Contrast the following formula with the preceding one:

$\begin{pmatrix} \{1\}\text{^}\# \\ A[1\text{^},\#] \end{pmatrix}$ (*colère*) = *en* [~] (in [~]).

While A[1^] represents an adjective or an adjectival phrase, A[1^,#] represents a word (here, a preposition) or a phrase which once combined with the keyword will form an adjectival phrase.

4 Relating the explicit encoding to other modes of encoding

4.1 Defining the ECD encoding

The ECD encoding is based on linear unidimensional formulas that synthesize both the information on the semantic content and syntactic frame of LFs. It uses a finite number of *SIMPLE LFs*, all other LFs being expressed by some form of concatenation of the simple LFs. After introducing the definition of simple LFs, we will show that more complex formulas can be obtained by means of operations performed on simple LFs.

Simple LF

We show below how a few simple LFs can be expressed in the explicit encoding. (For lack of space, we are forced to presuppose that the reader has at least an approximate knowledge of the ECD encoding.)

$\mathbf{Func_0} := \begin{pmatrix} \# \\ V[\#] \end{pmatrix}$; $\mathbf{Func_i} := \begin{pmatrix} \# \\ V[\#,i] \end{pmatrix}$;

$\mathbf{Oper_i} := \begin{pmatrix} \# \\ V[i,\#] \end{pmatrix}$; $\mathbf{Caus} := \begin{pmatrix} Caus[\Omega,\#] \\ V[\Omega,\#] \end{pmatrix}$;

$\mathbf{Manif} := \begin{pmatrix} Manif[\#,\Omega] \\ V[\#,\Omega] \end{pmatrix}$;

$\mathbf{Non}:=\begin{pmatrix} Non[\#] \\ pos(\#)[\#] \end{pmatrix}$; $\mathbf{Incep}:=\begin{pmatrix} Incep[\#] \\ pos(\#)[\#] \end{pmatrix}$;

$\mathbf{Magn} := \begin{pmatrix} \{\#\}\text{^}Magn \\ A[\#\text{^}] \end{pmatrix}$ or $\begin{pmatrix} \{\#\}\text{^}Magn \\ Adv[\#\text{^}] \end{pmatrix}$

$\mathbf{A_i} := \begin{pmatrix} \{i\}\text{^}\# \\ A[i\text{^},\#] \end{pmatrix}$; $\mathbf{Adv_i} := \begin{pmatrix} \{i\}\text{^}\# \\ Adv[i\text{^},\#] \end{pmatrix}$

The expression pos(#), which appears in the syntactic frames of **Non** and **Incep**, denotes the part of speech of the keyword. **Non** and **Incep** do not impose any part of speech for their value, but they behave as some kind of syntactic "chameleons."

Composition of LFs

The first "natural" operation on LFs that we may think of is that of *COMPOSITION*, that is, the application of an LF **g** to the application of another LF **f**: **g**o**f**(L) = **g**(**f**(L)). As has already been mentioned in Meaning-Text literature, this operation bears very little interest in terms of lexicographic description: if $\mathbf{f}(L_1) = L_2$ and $\mathbf{g}(\mathbf{f}(L_1)) = L_3$, this does not imply that any LF relation holds between L_1 and L_3. Take for instance the case of the adjective *sour* (*a sour apple*/*dish*/*liquid*/…). The most neutral value for $\mathbf{Oper_1}$(*sour*) is *to be* (*This apple is sour*). Now, what would be a possible collocational value for **Incep**(*to be*)? There does not seem to be any other choice than the very general *to start*: *This is salty* → *This starts to be salty.* (We ignore here the non collocational, fused value *to become*.) Clearly, it would be farfetched to pretend that an LF relation holds between *sour* and *to start*. On the other hand, as far as the semantic content of **Incep** is concerned, there is definitely a special relation holding between *sour* and *to turn* (*It turned sour*), which does not hold between *to be* and *to turn*: it would be equally farfetched to sate

that **Incep**(*to be*) = *to turn*. As we will soon demonstrate, the relation which holds between *sour* and *to turn* is not the result of a composition of LFs, but the result of a product of LFs.

Product of LFs

The *PRODUCT* is the more productive mode of combination of LFs. Some additional formalism is necessary in order to explain how a product is obtained. Let c(**f**) be the content of **f**, d(**f**) be the diathesis of **f**, pos(**f**) be the part of speech of **f**. In first approximation, the *PRODUCT* of **g** and **f**, written **g.f**, is defined as:

$$\mathbf{g.f} := \begin{pmatrix} c(g):\#\rightarrow c(f) \\ pos(g)[d(g):\#\rightarrow d(f)] \end{pmatrix}$$

In other words, c(**g.f**) is equal to c(**g**) where # is replaced with c(**f**), and d(**g.f**) is equal to d(**g**) where # is replaced with d(**f**). For instance, the products of **Incep** and **Caus** with an LF **f** return the following formulas:

$$\mathbf{Incep.f} = \begin{pmatrix} Incep[c(f)] \\ pos(f)[d(f)] \end{pmatrix};$$

$$\mathbf{Caus.f} = \begin{pmatrix} Caus[\Omega, c(f)] \\ V[\Omega, d(f)] \end{pmatrix}.$$

It is important to note that in this paper we are dealing only with products of collocational LFs, i.e. LFs where the # symbol appears in the syntactic frames of **f**, **g** and **g.f**.

Our *sour*/*to turn* problem, mentioned above, finds now its solution. It is a case of LF product, i.e. $\mathbf{Incep.Oper_1}$(*sour*) = *to turn* [~], and the collocation *to turn sour* is properly accounted for by the following explicit encoding formula:

$$\mathbf{Incep.Oper_1} = \begin{pmatrix} Incep[\#] \\ V[1,\#] \end{pmatrix}$$

Note that even though composition and product of LFs are two different operations, they can however be related. If **g** is a syntagmatic LF, the value of **g.f**(L) is a (potentially less idiomatic) paraphrase for the value of **g**o**f**(L) + the value of **f**(L): *to turn* [*sour*] ≡ *to start to be* [*sour*]. It is this relation that makes the product be an equally "natural" operation, from a formal point of view.

All that seems very neat. However, the case of $\mathbf{A_i}$ (remember $\mathbf{A_1}$(*colère*) in 3.2 above) will force us to refine the definition of **g.f**. We want that:

$$\mathbf{A_i.f} = \begin{pmatrix} \{i(f)\}\hat{}c(f) \\ A[i(f)\hat{},d'(f)] \end{pmatrix}$$

where i(**f**) is the i[th] actant of c(**f**) or d(**f**), and d'(**f**) is d(**f**) minus i(**f**). For instance:

$$\mathbf{A_2.Manif} = \begin{pmatrix} \{\Omega\}\hat{}Manif[\#,\Omega] \\ A[\Omega\hat{},\#] \end{pmatrix}$$

E.g.: (*Un geste*/*regard*/…) *rempli* [*de colère*] (lit. (a-gesture/look/…-filled with-anger)).

Taking into consideration the above remark about $\mathbf{A_i}$ (which applies to $\mathbf{Adv_i}$ as well), the *PRODUCT* of **g** and **f** can be defined in more general terms as follows:

$$\mathbf{g.f} := \begin{pmatrix} c(g):\#\rightarrow c(f), i\rightarrow i(f) \\ pos(g)[d(g):\#\rightarrow d'(f), i\rightarrow i(f)] \end{pmatrix}$$

Of course, only when the first member of the product is of the $\mathbf{g_i}$ form should actant replacement (i→i(**f**)) be performed in the content and diathesis components.

Incorporation and paradigmatic LFs

In order to account for fused values (mentioned in Section 2.4), we introduce the operation of *INCORPORATION*. It associates to each syntagmatic LF **f** a corresponding paradigmatic LF / **f** such that / **f**(L) is a paraphrase of L+**f**(L). The effect of the incorporation operator / is to remove # from the syntactic frame of **f**. For instance:

$$/\ \mathbf{Oper_i} = \begin{pmatrix} \# \\ V[i] \end{pmatrix};\ /\ \mathbf{A_i} = \begin{pmatrix} \{i\}\hat{}\# \\ A[\hat{}i] \end{pmatrix}$$

Notice that the incorporation operator does not exist as such in the ECD encoding, which uses the / symbol as only a "mark" on a value indicating that it is fused: **Magn**(*colère*) = / *rage*. We believe that this notation, while convenient when fused and non fused values have to be listed together, hides the fact that a fused value does not entertain the same relation with the keyword as a non fused value. As the relation is different, the name of the relation has to be different in an explicit encoding.

4.2 Deriving other encodings

Our attempt to define a new explicit encoding by no means implies that the ECD encoding should be discarded. While it is not suited for performing formal computations, it is highly relevant for human encoders of LF relations. ECD formulas are not so dissimilar from expressions in natural language. A

formula such as **Caus.Non.Manif**(L), for instance, can be very directly translated into pseudo-English as *to cause the non-manifestation of L* and, therefore, can be used as pseudo-paraphrase for the value itsef in English sentences. Because of this, the ECD encoding is a metalanguage with which the lexicographer and the linguist can "think." However, we propose this metalanguage to be defined on top of the explicit encoding, and not the other way round, in order to provide it with safer formal foundations.

Furthermore, we believe that linear formulas of the ECD type could be automatically derived from the explicit encoding, and that these formulas could themselves be automatically translated into expressions in "simplified" English, French, etc. For the purpose of our experimentation with Fr. COLÈRE we manually produced these translations and we give below a few illustrations of how they look like. Note that the translation procedure takes as parameter the general semantic value (what we term the *semantic label*) of the keyword. Thus, the sample translations that follow are valid for nouns of feeling only and the reader may replace # with *feeling* in reading the proposed translations:

Oper$_1$≡ *experiences #*; **Oper$_2$**≡ *is the target of #*;
Oper$_3$≡ *is the reason for #*;
Func$_0$≡ *takes place*;
Func$_1$≡ *is in X*; **Func$_2$**≡ *is targetting Y.*

For lack of space, we will not elaborate further on the problem of bridging the gap between different modes of encoding. Suffice it to say that this problem has to be carefully addressed, especially in contexts where one wishes to popularize the concept of lexical function (for language learning, layman dictionaries, etc.).

Conclusions

Our main purpose in developing an explicit encoding for LF relations was to make available a formalism that would be computationally tractable. Such a formalism should allow for the implementation of operations such as: 1) automatic search on lexical databases that encode LF relations (to retrieve data, identify inconsistencies, etc.), and 2) various sorting of lexical function relations in lexical entries (according to the part of speech of values, according to semantic content, etc.).

We believe our explicit encoding to be suitable for these tasks as it meets the following three requirements: 1) it is entirely defined in terms of its syntax and semantics; 2) it allows us to express all information that seems relevant to the automatic processing of lexical databases; i.e., it has sufficient granularity and can be connected in a rather direct fashion to the ECD encoding; 3) it allows for the definition of formal operations such as composition, product and incorporation.

However, the task is not yet completed. Our next goal is to extend the coverage of the proposed encoding so that it can cover the whole spectrum of LF relations already accounted for by means of the ECD encoding. This implies a complete description of all simple LFs in explicit encoding terms and the definition of more operations on LFs.

References

Heid, U. (1996) Using Lexical Functions for the Extraction of Collocations From Dictionaries and Corpora. In Wanner (1996), 115-146.

Iordanskaja, L., M. Kim & A. Polguère (1996) Some Procedural Problems in the Implementation of Lexical Functions for Text Generation. In Wanner (1996), 279-297.

Fontenelle, T. (1997) *Turning a Bilingual Dictionary into a Lexical-Semantic Database*, Tübingen: Niemeyer.

Mel' uk, I. A. (1996) Lexical Functions: A Tool for the Description of Lexical Relations in a Lexicon. In Wanner (1996), 37-102.

Mel' uk, I. A., N. Arbatchewsky-jumarie, L. Elnitsky, L. Iordanskaja & A. Lessard (1984) *Dictionnaire explicatif et combinatoire du français contemporain: Recherches lexico-sémantiques I*, Montreal: Presses de l'Uni versité de Montréal.

Mel' uk, I. A., N. Arbatchewsky-jumarie, L. Iordanskaja, S. Mantha & A. Polguère (1999) *Dictionnaire explicatif et combinatoire du français contemporain: Recherches lexico-sémantiques IV*, Montreal: Presses de l'Uni versité de Montréal.

Mel' uk, I. A. & A. K. Zholkovsky (1984) *Explanatory Combinatorial Dictionary of Modern Russian*, Vienna: Wiener Slawistischer Almanach.

Wanner, L. (ed.) (1996) *Lexical Functions in Lexicography and Natural Language Processing*, Amsterdam/ Philadelphia: Benjamins.

º olkovskij, A. K. & I. A. Mel' uk (1965) O vozmoì nom metode i instrumentax semanti eskogo sinteza. *Nau no-texni eskaja informacija*, 5, 23-28.

Addendum to "A computationally tractable formal language for the encoding of lexical relations"

Sylvain Kahane & Alain Polguère

5 Introduction

The present text is an addendum to our paper "A computationally tractable formal language for the encoding of lexical relations" [hereafter, KP]. It contains text that we were unable to present in our paper for lack of space, as well as considerations on aspects of our research which are still in a phase of gestation. Additionally, we are trying to introduce here our formalism with more accuracy, specially when it comes to position it relative to the standard way of encoding lexical relations. In KP, we referred to two distinct encodings of lexical relations, that we termed the *EXPLICIT ENCODING* and the *ECD ENCODING*. This last encoding, as we presented it, is in fact not exactly the encoding used in ECDs. We will now call it the *ALGEBRAIC ENCODING* and we will compare it with the "true" ECD encoding, explaining the role it plays in the formal modeling we propose for LF relations.

This addendum is structured as follows. Section 2 gives more details on some aspects of our explicit encoding. Section 3 defines the algebraic encoding, which is then put into relation with actual ECD encoding, in Section 4. Section 5 presents two tables with the description —in terms of ECD, algebraic and explicit encoding— of all LF values for Fr. COLÈRE which have 1) `V[1,#]` as syntactic pattern in our explicit encoding (first table) and 2) `A` as part of speech (table 2). Additional references introduced in this addentum are listed in the final *References* section.

6 More on the Explicit Encoding

6.1 Remarks on the semantic content

Coordination in semantic content

The explicit encoding presentend in KP is only a primary and incomplete proposal. There are many aspects of this encoding that needs to be developed. Additionally, we were not able, for lack of space, to present a complete account of all problems and solutions that have been examined when conducting our "test study" on COLÈRE. Let us just mention here the case of the an additional operator that needs to be introduced in the formal language for semantic content formulas, namely the coordination operator `&`:[5]

- `Caus[Ω,Minus[#&Manif[#]]]`
  [= (Ω causes the decrease of anger and of its manifestation)]
  Ex. *Qqch./Qqn.* ***adoucit*** *la colère* (*de X*) (lit. (Sth./Sb.-softens-the-anger(-of-X)))

The above collocation is the only instance we found in the lexical entry for COLÈRE where the use of the coordination operator seems to be needed. It is unfortunate that this example is not very clear-cut: one could argue that in *Celà a adouci la colère de Paul* ((This softened Paul's anger)), only the decrease of the anger itself, and not of its manifestation is expressed. However, it seems legitimate to believe that cases where coordination has to be encoded will necessarily surface while applying the explicit encoding to LF data.

About the semantic content of paradigmatic LFs

In KP, we have decided to use the same formula to encode the semantic content of a syntagmatic LF **`f`** and the corresponding paradigmatic (via incorporation) LF / **`f`**; the encoding will vary only in its syntactic frame component. Cf., for instance:

$$\mathbf{A_1} := \begin{pmatrix} \{1\}\hat{}\# \\ A[1\hat{},\#] \end{pmatrix} : \textit{en} \sim$$

$$/\ \mathbf{A_1} := \begin{pmatrix} \{1\}\hat{}\# \\ A[1\hat{}] \end{pmatrix} : \textit{fâché}$$

We must insist on the fact that the semantic content cannot be interpreted in the same way in both cases: it is not *en* which is synonymous to *faché*, but *en colère*. Therefore, for a non-incorporated syntagmatic LF **`f`**, `C(f)` does not express the con-

[5] Remember that all our examples concern the keyword COLÈRE, unless we make explicit mention of a different keyword.

tent of the value —the collocate, but the content of the syntagmatic combination of the key-word and the value —the whole collocation.

Another solution would consist in using different expressions for `C(f)` and `C(/ f)`, for example, by puting # beween {…} in `C(f)`. We think that it is not necessary, because a paradigmatic LF can be trivially spotted by the absence of # in the syntactic pattern and its content interpreted as it must be in this case. The only case where we have to do this is when we deal with a syntagmatic LF whose value modifies the keyword, such as **Magn**:

$$\textbf{Magn} := \begin{pmatrix} \{\#\}\hat{}\text{Magn} \\ \text{A/Adv}[\#\hat{}] \end{pmatrix}$$

$$/\ \textbf{Magn} := \begin{pmatrix} \#\hat{}\text{Magn} \\ \text{pos}(\#)[\text{d}(\#)] \end{pmatrix}$$

Here I had = in your text, above it was :=. I think it should be :=, according to the meaning you gave me for this operator in our paper.

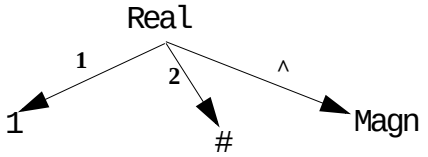
In the case of / **Magn**, we want to express the fact that #, and not Magn, is the dominant node of the content.

Graphical representation

A semantic content formula, as we encode it, is formally equivalent to a tree whose branches are either a predicate-argument (**1**, **2**, etc.), a modification (^) or a coordination (**&**) relation. For instance, the following semantic formula :

- Real[1,#]^Magn
  [= (X realizes # with intensity)]
  E.g. *X **déchaîne** sa colère sur Y* (lit. (X-unleashes-his-anger-on-Y))

formally corresponds to the following tree:



See also Dymetman & Copperman (1996) and Kahane (2000), where semantic graphs are explicitly encoded as trees. We are aware of the fact that this encoding makes our semantic formulas less "semantic;" they are actually very close to Meaning-Text deep-syntactic structures.

6.2 Remarks on the syntactic pattern

Part of speech

Let us return to our preceding example of $\textbf{A}_1$ *vs* / $\textbf{A}_1$ encoding; we cite it again for the reader's convenience:

$$\textbf{A}_1 := \begin{pmatrix} \{1\}\hat{}\# \\ \text{A}[1\hat{},\#] \end{pmatrix} : \textit{en} \sim$$

$$/\ \textbf{A}_1 := \begin{pmatrix} \{1\}\hat{}\# \\ \text{A}[1\hat{}] \end{pmatrix} : \textit{fâché}$$

The symbol A in the explicit encoding is not exactly the part of speech of the value. It means that the value is the head of an adjectival phrase. For the syntagmatic LF $\textbf{A}_1$, the value (here, *en*) is not an adjective but a translatif of noun in adjective (using Tesnière's terminology), that is, a preposition. More generally, the value of a syntagmatic LF is a translatif of the part of speech of the keyword into the part of speech given in the syntactic pattern of the LF. For this reason, it could be judicious to mention also the part of speech of the keyword in the explicit encoding (even if it it can be determined by general syntactic rules, from the information already found in the present encoding).

Paradigmatic LFs and syntactic pattern

For paradigmatic LFs, the syntactic pattern depends on the syntactic pattern of the keyword. For instance:

$$\textbf{Syn} := \begin{pmatrix} \# \\ \text{pos}(\#)[\text{d}(\#)] \end{pmatrix},$$

where pos(#) and d(#) are respectively the part of speech and the diathesis of #.

6.3 Handling of multiple values

In the ECD encoding, multiple values of an LF for a same keyword are presented as a list separated by three types of separators:

1. comma: V_1**,** V_2 means that values V_1 and V_2 are of equal semantic status (these values are ordered alphabetically);
2. less-than: $V_1 < V_2$ means that V_2 ranks higher than V_1 on a "semantic intensity scale";
3. semi-colon: V_1**;** V_2 means that a non trivial semantic gap exists between V_1 and V_2 (i.e., in a given context, V_2 may not be considered as being an adequate paraphrase for V_1).

In our proposal, LF links are described in an atomic way. In other words, explicit formulas encode a relationship between a keyword and one given value. Sometimes, all values encoded under a single ECD formula will receive identical formulas in the explicit encoding too. In such cases, it is a very straighforward business to automatically connect all these values together. However, this will not always be the case. For instance, in cases of incorporation, we will need to be able to automaticlly relate incorporated and non-incorporated values, even though they do not receive identical encoding. This problem is part of the much more global task of developing a complete and consistent algebra based on the proposed encoding.

6.4 Granularity

Let us recall the definition of the granularity:

An encoding $\varphi : R \rightarrow E$ defines a partition of *R*, where *R* is the set of lexical relations. A given encoding φ_1 is said to be more *GRANULAR* than another encoding φ_2 if φ_1 defines a finer partition of *R* than φ_2, that is, if φ_2 collapses together some LFs which are considered separately by φ_1.

In other words, more an encoding is granular, less each LF covers a great number of lexical relations.

We have adopted more or less the same granularity as the traditional encoding, but we did not consider whether this granularity is adequate. For instance, can we consider that **Fact$_i$** and **Real$_i$** are converse or must we consider that **Fact$_i$** and **Real$_i$** express different meanings ? Or on the contrary, must **Real$_i$** be split in order to account for huge semantic gaps such as the one mentioned in KP, Footnote 3 (The meaning expressed by **Real$_1$**(*recommendation*) in *to follow a recommendation* is obviously distinct from the meaning expressed by **Real$_1$**(*car*) in *to drive a car.*) ? Is the oblicity of syntactic arguments of the value always a good criterion to split the lexical relations? This last question touches upon the modeling of thematic roles in LF values. Thematic roles have to be taken into consideration for handling paraphrasing with LF relations: *to be the target of, to be the reason of...* A study is necessary to correlate the thematic role of the argument of the keywords with the value of the LF. In particular, for a given LF content, a V[2,#] where 2 has the thematic role θ is probably nearer from a V[3,#] where 3 has the same thematic role θ than a V[2,#] where 2 has another thematic role.

7 An Algebraic Encoding

As we said in KP, there is many reasons to define an "algebraic encoding" such as the ECD encoding. But why do not condider directly the ECD encoding ? Let us define what we call an algebraic encoding and explain why the ECD is not exactly a pure algebraic encoding.

An encoding of lexical relations is a correspondence $\varphi : R \rightarrow E$ where *R* is the set of lexical relations and *E*, a formal language. An encoding $\varphi : R \rightarrow E$ can be said algebraic if *E* is a set of algebraic expression, such as a set of arithmetic expressions or a set of logic formula. A set E of algebraic expressions is buils on a set of atoms by the applications of a finite number of operations. In the case of arithmetic expressions, atoms are numbers and operations are +, -, ×…giving us expressions such as -3×(4+7). In our case, atoms are simple LFs and operations are incorporation, product…

The ECD encoding resemble an algebraic encoding because it use a set of expression on a sets of atoms. The only "operation" considered is called the combination and is noted by simple concatenation. Neverthless, as we will see below, the combination is not an algebraic operation: it corresponds to different operations according to the FL which are combined.

7.1 Complements on simple LFs

Let us give some complements to the list of simple LFs given in KP.

V_0 := (#,V); S_0 := (#,N);

A_0 := (#,A); Adv_0 := (#,Adv);

S_i := (i,N[d'(#)]) with d'(#) = d(#)\{i}

$Fact_0$:= (Fact[#], V[#]); $Fact_i$:= (Fact[#], V[#,i])

$Real_i$:= (Real[i,#], V[i,#])

$Caus_i$:= (Caus[i,#], V[i,#])

Sympt ?? (Comment va-t-on faire avec les μ ?)

Conv$_{21}$ = $(\#, pos(\#)[d(\#)_{1\rightarrow 2, 2\rightarrow 1}])$ = (#,pos(#)[2,1,d'(#)]) with d'(#) = d(#)\{1,2}.

In other words, **Conv$_{21}$** is a paradigmatic LF which permutes the two first elements of the diathesis of #. Note that it is paradigmatic LF which is not obtained by applying the operator // to a syntagmatic LF.

7.2 Properties of incorporation

For an **f**(L) that normally functions as modifier of L (for instance **Magn**), the syntactic frame associated to the value of / **f**(L) is the syntactic frame of the keyword:

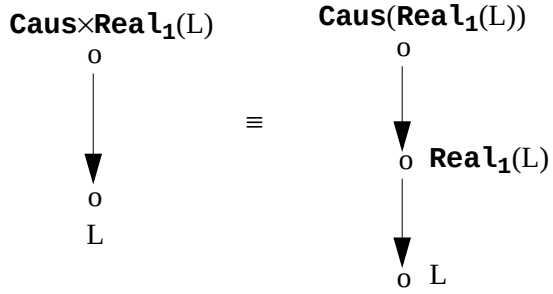
$$/\ \mathbf{Magn} = \begin{pmatrix} \#\text{^Magn} \\ \text{pos}(\#)[\text{d}(\#)] \end{pmatrix}$$

For instance, / **Magn**(*eat*) = *devour*, where *devour* are associated with the following syntactic frame: V[1,2].

However, this is just a default rule and it is not very difficult to find counter-examples. For instance, we have in the DiCo database[6] / **Magn**(Fr. *défaite* (defeat)) = *râclée* (trashing), but *râclée* does not possess the same diathesis as *défaite*; in fact, they are conversive units: if we write that d(*défaite*) = [1,2,3] (*X's defeat in the hands of Y in the event Z*) then we have d(*râclée*) = [2,1,3] (*trashing inflicted by Y to X in the event Z*). While we have good reasons to believe that such discrepancies between the syntactic frame of incorporated values of **Magn** and that of the keyword are not uncommon, it is a fact that the standard ECD encoding will never allow us to exhaustively retrieve such data, thus forbidding us from conducting a systematic study of the question. In point of fact, the ECD encoding will force us to state two distinct FL links between DÉFAITE and RÂCLÉE: the fact that the latter is a fused (i.e. incorporated) intensifier of the former and the fact that it is an "approximate" conversive. We believe our encoding to offer both a more compact, more elegant and more accurate way of modeling these facts.

7.3 Products

For sake of convenience, let us introduce a new, more explicit operator for LF product: ×. The definition of the product reflects the fact we want the two deep-syntactic configurations **f**×**g**(L)→L and **f**(**g**(L))→**g**(L)→L, where **f** and **g** are two syntagmatic LFs for which the value is the governor of the keyword, to be in a relation of paraphrase. For instance:



As we will show, this equation determines completely the semantic content of **f**×**g** and its part of speech, but not its diathesis.

The semantic content c(**f**×**g**) must be equal to the semantic content c(**f**) where the semantic content of keyword is equal to c(**g**) and the ith argument of the keyword is i(**g**). Consequently:

c(**f**×**g**) := c(**f**)$_{\#\rightarrow \text{c}(\mathbf{g}),\text{i}\rightarrow \text{i}(\mathbf{g})}$.

The part of speech pos(**f**×**g**) must be equal to pos(**f**).

The case of the diathesis d(**f**×**g**) is more problematic. We want the diathesis of **f**×**g**(L) to be as close as possible to the diathesis of **f**(**g**(L))→**g**(L).

But as this is a deep-syntactic configuration and not a lexical unit, there is no perfect solution. A first choice is to take the diathesis of **f**(**g**(L)) and insert the diathesis of **g**(L) at the place occupied by **g**(L).

The diathesis of **f**(**g**(L)) is the diathesis of **f**, d(**f**), where each i stands for the ith argument i(g) of the keyword g(L), that is d(f)$_{\text{i->i(g)}}$.

The diathesis of **g**(L) is d(**g**) and g(L) occupies the place # in the diathesis of f(g(L)). Hence the solution is:

d(f×g) := d(f)$_{\text{i->i(g),\#->d(g)}}$.

Nevertheless, this is not a good solution. It must be noted that, in a construction f(L')->L', the argument of L' which are expressed on f(L') cannot be exressed on L'. For instance, with f = Oper$_1$, 1 is expressed on f(L') and cannot be expressed on L (*Pierre*$_1$ *éprouve de la colère*, but **Pierre*$_1$ *éprouve*

[6] For a presentation of the DiCo project, see Polguère (to appear —maybe).

sa$_1$ *colère*). Therefore the diathesis of g(L) when it is governed by f(g(L)) is not d(g) but $d_f(g)$, where $d_f(g)$ is a sublist of d(g) prived to all the argument i(g) expressed on f(g(L)) (containing # is # appear in d(g)):

$d(f \times g) := d(f)_{i->i(g),\#->df(g)}$

with $d_f^-(g) \leq d_f(g) \leq d_f^+(g)$,

$d_f^-(g)$ = # (or is empty if # does not appear in d(g)) and $d_f^+(g)$ = d(g)\{i(g) expressed in d(f)}.

The exact value of $d_f(g)$ depends on f.

Rather to consider the true product, we will introduce to near operations, the concatenation . and the light product ∗.

$f.g := (c(f)_{\#->c(g)}, pos(f)[d(f)_{\#->d(g)}]$

$f*g := (c(f)_{\#->c(g),i->i(g)}, pos(f)[d(f)_{i->i(g)}]$

The light product is particular case of product with $d_f(g)$ = #. The concatenation is equal to a product (with $d_f(g)$ = d(g)) only when there is no argument i of # in the syntactic pattern of f.

Examples:

Incep.g = (Incep[c(g)],V[d(g)])

Caus.g = (Caus[Ω,c(g)],V[Ω,d(g)])

$Caus_i$.g = (Caus[i,c(g)],V[i,d(g)])

$Incep.Func_0$ = **Incep**

$Caus.Func_0$ = **Caus**

$Caus_i$∗g = (Caus[i(g),c(g)],V[i(g),#])

Generally, $Caus_i$.g and $Caus_i$∗g differ in semantic content and in syntactic pattern. For instance:

$Caus_1.Oper_2$ = (Caus[1,#],V[1,2,#])

$Caus_1*Oper_2$ = (Caus[2,#],V[2,#]) = $Caus_2$

Only $Caus_1*Oper_2$ is apparented with the composition $Caus_1 \circ Oper_2$. Indeed, $Caus_1(Oper_2(L))$ take as causer the first argument of its keyword $Oper_2(L)$ wich is 2.

A_i∗g = ({i(g)}^c(g),A[i(g)^,#]

A_1∗$Oper_1$ = **A_1**

A_1∗$Oper_2$ = **A_2**

A_2∗Manif = ({Ω}^Manif[#,Ω],A[Ω^,#])

A_1∗Manif = ({#}^Manif[#],A[#^])

In this last case, there is a little problem: 1(g) = # but the syntactic pattern cannot be A[^#,#]. So the diathesis is reduced to #^.

$Conv_{21}$∗g = $(c(g),pos(g)[d(g)_{1(g)\to 2(g),2(g)\to 1(g)}])$

= (c(g),pos(g)[1(g),2(g),d'(g)])

$Conv_{21}$∗$Func_1$ = **$Oper_1$**

$Conv_{21}$∗Manif = (Manif[#,Ω],V[Ω,#])

$Conv_{21}$∗Manif(*colère*) = *traduire* [ART ~]

//Magn∗g = (c(g)^Magn,pos(g)[d(g)])

f∗//Magn = $(c(f)_{\#->\#\wedge Magn}$,pos(f)[d(f)])

A_2 = **A_1∗$Conv_{21}$**

Paradigmatic LFs and product

When f and g are two syntagmatic LFs where the value is the governor of the keyword, f×g is a syntagmatic LF and f×g(L) -> L and f(g(L)) -> g(L) -> L are substituable. The same property occurs for the light product f∗g which is a particular case of product with $d_f(g) = d_f^-(g)$ = #.

But when f is a syntagmatic LF and g is a paradigmatic LF, f×g is a paradigmatic LF and f×g(L) et f(g(L)) -> g(L) are substituable. This is due to the fact that $d_f(g) \leq d(g)$ does not contain # because g is paradigmatic. With our definition of the light product, f∗g is a syntagmatic LF and f(g(L)) -> g(L) substitutes with f∗g(L) -> L rather than f∗g(L). This is due to the fact that we have not substitute # in d(f), while $d_f(g)$, which substitute with # in the definition of ×, does not contain #.

7.4 About the coverage of the Algebraic Encoding

Diathesis

Our set of simple LFs does not allow to express all the possible diathesis of LF. Consider, for instance:

(Manif[#]^Magn,V[1,#,2]) fulminer de ~ contre Y.

Due to its semantic content, this LF is of the form //Magn.Manif. But the syntactic pattern of //Magn.Manif is V[#].

Therefore, we need a LF which can add an argument in the diathesis:

$Perm_i$ = (#,pos(#)[i,d(#)])

$Perm_i$.g = (c(g), pos(g)[i,d(g)])

In fact, a LF $Perm_i$ is already considered in the ECD encoding with a meaning 'i permits #'. We think that this meaning is not imputable to the semantic content but only to a modification of the communicative structure due to the addition of an argument.

Example:

$Perm_1.Func_0 = Oper_1$

Combinations of $Perm_i$ and $Conv_{ji}$ allow to obtain all the possible diathesis.

$Oper_{12} = Conv_{32}*Conv_{21}*Perm_2.Oper_1$

Perhaps, it would be useful to give a name to a combination such as $Conv_{32}*Conv_{21}*Perm_2$, which allows to add the argument 2 in the third position.

Other problems

How could we express the following LF ?

(Incep[Real[1,#'[1,2']]]^Fin[Real[1,#]],V[1,#,2']) tourner sa ~ contre Y.

All the {Ω[1]}^... are problematic. Following the ECD encoding, we can introduce a special notation:

Propt := ({Ω[1]}^Caus[#,Ω],**Adv**[^Ω,#]);

Another notation is needed for ({Ω[1]}^#,**Adv**[^Ω,#]) (which is encoded **Adv_1**, as ({1}^#,**Adv**[^1,#]) in the ECD encoding, which is rather confusing).

Another solution could be to introduce a LF which introduce the semantic content {Ω[1]}^...

7.5 From the Explicit Encoding to the Algebraic Encoding

It is easy to go from an algebraic formula to an explicit formula: we have just to take the definition of the simple LFs considered and to calculate the result of the combination, which is easy because the two operations we use, product and incorporation, are well defined. It is exactly as to go from the algebraic expression 2x(4+3) to the explicit result 14.

But how to go into the reverse direction ?

It must be noted first that the Algebraic Encoding is redundant. For instance:

$Incep.Func_0 = Incep$

$//Caus_i.Func_j = //Caus_i.Oper_j = (Caus[i,\#],V[i, j])$

Therefore, if we want to express an explicit formula by a algebraic formula, we must make choices.

A suivre

8 From the Algebraic Encoding to the true ECD Encoding

Simple LFs

Our Algebraic Encoding use a lot of simple LF of the ECD Encoding: Incep, Caus, Non...

The problem of the subspecification: $Oper_i$ and $Oper_{ij}$.

Sympt

Pred ?

The problem of A_i and Adv_i: in the ECD encoding Ai is considered as a paradigmatic LF. Consequently, the collocation *en colère* is considered as semantic derivation of colère. It is not false: more generally, if f is a syntagmatic LF, L+f(L) can be considered as a value of //f(L). And it could be possible to describe all the collocations with paradigmatic LFs. But if we think that there is an interest to describe the collocations as own, me must have a LF to describe the collocation *en colère*.

Combinations of LFs

The ECD encoding use an operation of combination (noted by concatenation of simple LF) which is not homogenous. According to the first LF f, an ECD combination fg can corresponds to f.g or f∗g.

Incepf := Incep.f

Causf := Caus.f

$Caus_i f := Caus_i.f$

$A_i f := A_i*f$

$Conv_{ij} f := Conv_{ij}*f$

Sum

The ECD encoding consider an operation noted + and called the sum. This operation is reductible to the light product:

g+f := f∗//g

where g is syntagmatic LF which is a modifier of #, such as Magn.

For instance,

Magn+f = f*//Magn = $(c(f)_{\#->\#\wedge Magn}, pos(f)[d(f)])$

Same here. Let's see where this has to go and we'll right something decent.

9 More on the LAF encoding

$Conv_{21}$ cannot be expressed as own by a linguistically reasonable expression. For each binary simple LF f, we must have an expression of f but also of $Conv_{21}$f.

Quid de $Perm_i$?

10 A sample dictionary entry (in a table)

Example 1: We give all the value of LF with syntactic pattern V[1,#] for the keyword fr. *colère*

Il faudrait ajouter l'encodage alg brique

Je suis d'accord là-dessus. Ne touche à rien dans cette section, j'y travaille. C'est de la job de bras, de toute façon.

#	*éprouver, ressentir, être en ~*
#^Magn	*bouillir, bouillonner de (ART) ~*
Incep[#]	*se mettre, se foutre en ~*
Caus[1,Minus[Manif[#]]]	*étouffer ART ~*
Real[1,#]	*laisser sortir, libérer sa ~*
Perm[1,Fact[#]]	*s'abandonner, céder, se laisser aller à ART ~*
Non[Perm[1,Fact[#]]]	*retenir, réprimer, réfréner, dominer, contenir ART ~*
Sympt[#,'parole']	*bégayer, bafouiller de ~*
Sympt[#,'cri']	*crier, hurler de ~*
Sympt[#^Magn,'souffle']	*étouffer, s'étouffer, s'étrangler, suffoquer de ~*
Sympt[#^Magn,'sécrétion']	*écumer de (ART) ~*
Sympt[#,'gestuelle']	*trépigner de ~*

Example 2: We give all the value of LF with part of speech A.

({1}^#, A[^1])*//fâché*

({1}^#, A[^1,#])*en ~*

({1}^(#^Magn^Manif[#]), A[^1,#])*fou, ivre de ~*

({1}^(#^Magn^Manif[#]), A[^1])*_hors de soi_*

({1}^#, A[^1,2])*//monté contre 2*

({μ/1}^Sympt[#,'visage'], A[^μ/1,#]) *blanc, blême, pale, rouge, écarlate, cramoisi de ~*

({#}^Magn, A[^#])*~ forte, grande, incroyable, indescriptible, insurmontable, terrible, _sans nom_, aveugle, hystérique, folle, sauvage*

({#}^Non[Manif[#]], A[^#])*~ froide, sourde*

({◊}^Manif[#,◊], A[^◊,#])*plein, rempli, empreint, chargé de Ø/ART ~]*

({#}^Non[Perm[1,Manif[#]]], A[^#])*~ rentrée*

({μ/1}^Sympt[#,'visage'], A[^μ/1,#]) *blanc, blême, pale, rouge, écarlate, cramoisi de ~*

References

Dymetman, M. & M. Copperman (1996) Extended Dependency Structures and their Formal Interpretation. In: *Proceedings of COLING'96*, Copenhagen, 255-261.

Kahane, S. (2000) Extractions dans une grammaire de dépendance léxicalisée à bulles. *T.A.L.*, 41:1, Paris, 30p.

Polguère, A. (draft) Toward a theoretically-motivated general public dictionary of semantic derivations and collocations for French.