[*Proceedings of a Workshop on Machine Translation*, July 1990, UMIST]

Mel'čuk's Explanatory-Combinatorial Dictionary and its relevance for Machine Translation

Andy WAY
Department of Language and Linguistics
University of Essex
Wivenhoe Park
Colchester C04 3SQ, UK

July, 1990

Introduction

The intention of this paper is to introduce the reader to the 'Explanatory Combinatorial Dictionary' (ECD), an essential component of Melchuk's 'Meaning-Text Model' (MTM), which is a fully-fledged linguistic descriptive theory, and furthermore to investigate the relevance of such a theory to Machine Translation (MT).

There are four linguistic levels of description in the MTM : semantics, syntax (with deep and surface levels), morphology (also with a deep-surface distinction) and phonetics (or orthographies, again with the same subdivisions). The process of establishing correspondences between texts and meanings has several stages : one 'translates' a meaning (i.e. a semantic representation) from one level to another until one of the texts expressing that meaning is found. Obviously, if the MTM is to produce all the correct texts from a given 'translation' (and vice-versa), then enormous amounts of specific information need to be included in the ECD.

It may seem a little extreme to have so many levels of linguistic description, but by doing so one allows each stage of the mapping process to be stated in a simple manner whilst at the same time enabling a wider range of phenomena to be covered by the system. The phonetics levels do not play a great role in English (as they are related by relatively trivial mappings). The semantic level is represented by semantic nets (Melchuk & Polguere '87, p.262 ff), a widely used AI technique which permits the expression of default inheritance, whilst the syntactic levels use dependency trees.

Melchuk's system has been around for over 20 years now, and takes some getting used to. In the MTT, the levels of representation listed above are seen as an appendix to the ECD, unlike most other approaches in computational linguistics, which are generally syntax-oriented. The lexicon and the grammar together form the MTT, with the lexicon at its foundations. The grammatical levels express generalizations over the lexicon as well as containing all the necessary procedures for manipulating the data contained in the lexicon.

At first sight this dictionary is complex, but this is necessarily so, for the primary aims of the dictionary are to avoid circular definitions and to be fully comprehensive and consistent in approach. Such a dictionary must include all the semantic and combinatorial relationships of a given word to other words in a particular language. This illustrates one of the main advantages of this type of dictionary, namely that parts of entries are linked with those of others, so that relevant fields of information are connected. The major work of reference for this study was Melchuk's

"Dictionnaire explicatif et combinatoire du francais contemporain" (Melchuk et al '84).

The set of Lexical Functions (LF's) forms the basis of the combinatory process. They do seem complicated at first, but once one is familiar with their usage one appreciates the expressive power inherent in them. For this reason it is rather important that the major, more widely used LF's are explained at this juncture so that the rest of the paper becomes a little more intelligible.

Lexical Functions

The LF's appear in a strict order, as they would in the dictionary. The first group consists of the synonyms and antonyms, together with the 'Gener', 'Conv' and 'Figur' functions (which can be seen as quasi synonyms and antonyms), followed by those functions incorporating syntactic derivations, and lastly the rest of the LF's. According to their area of discourse, the LF's appear according to the following convention : nouns, adjectives, adverbs and verbs. As a general rule, more abstract LF's precede more concrete ones, so that the LF 'S(i)' - typical nouns for actant (i.e. surface syntactic argument) 'i', see (7) below - precedes 'Sing' (see (9) below) or 'Mult' (see (10) below), which are more specific instances of the usage of a particular noun.

It should be pointed out that the term 'function' is used rather loosely here, in that it does not coincide with the mathematical description of a function. For instance, in Melchuk's system the following is possible :-

A(l)(beard) = bearded
A(l)(beard) = hirsute

where it can be seen that the accepted notion of a unique relationship between members of the domain and the range is violated. Strictly speaking, therefore, the LF's are relations and not functions.

1. Syn, Syn(<), Syn(>), Syn(^) : synonyms and quasi-synonyms, or inexact synonyms, where the arrows signify having a narrower / wider / intersecting meaning than that of f (X) .

e.g. Syn(aid) = help
Syn(<)(respect) = veneration
[a particular kind of 'respect']
Syn(>)(veneration) = respect
Syn(^)(demonstration) = exhibition
Syn(^)(demonstration) = protest

The last two examples show that 'demonstration' sometimes means 'exhibition', and sometimes 'protest'.

2. Conv(ijkl) : converse relation to that of the key-word, where the subscripts refer to the actants associated with the key-word, i.e. Conv(312)(Key) means that the 3rd, 1st and 2nd actants of the key-word become the 1st, 2nd and 3rd actants respectively of Conv(Key).

e.g. Conv(3214)(sell) = buy

```
i.e. The shop sold a paper   to me     for 50p
        3              2       1         4
=>      1              2       3         4
        I   bought a paper from the shop for 50p
```

3. Anti, Anti(<), Anti(>), Anti(^) : antonyms (see 1 for details) .

4. Gener : generic word whose combination with a syntactic category derived from the key-word is synonymous with that key-word.

e.g. Gener(republic) = [republican] state

5. Figur : figurative use of a word, where combined with the key-word is synonymous with the key-word.

e.g. Figur(fog) = blanket [of fog]

6. Syntactic derivations : S(0), V(0), A(0), Adv(0), where 'syntactic derivation of a lexeme C(0)' means another lexeme or syntagm having the same meaning as C(0) but being of a different syntactic category.

```
e.g.   S(0)(honest) = honesty
       A(0)(school) = scholarly
       V(0) (loss)  = lose
     Adv(0)(honest) = honestly
```

7. S(l), S(2), S(3) ... : typical noun for the 1st, 2nd, 3rd ... actant of the key-word.

```
e.g. S(l)(crime) = criminal
     S(2)(crime) = victim
     S(1)(buy)   = buyer
     S(2)(buy)   = product
     S(3)(buy)   = seller
     S(4)(buy)   = price
```

8. S(instr), S(loc), S(med), S(mod), S(res) : typical nouns for the instrument, location, medium, mode, result of the key-word.

e.g. S(instr)(paint) = brush
S(med)(paint) = paint
S(res)(paint) = picture

9. Sing : a 'part' of a 'whole'.

e.g. Sing(fleet) = ship
Sing(flock) = sheep

10. Mult : converse of Sing.

e.g. Mult(ship) = fleet

11. A(l), A(2) ... : typical attribute for the 1st, 2nd actant of the key-word.

e.g. A(l)(beard) = bearded
A(2)(consider) = in question

i.e. the attribute to describe a person with a beard is 'bearded', where that person is the first actant of 'beard'. The second example illustrates the synonomy of the two phrases : "If you consider the author" and "The author in question", where 'author' is the second actant in both cases.

12. Able(l), Able(2) ... : function describing 'being capable of

e.g. Able1(fear) = the frightened (person)
Able2(fear) = the frightener

i.e. 'the frightened (person)' is capable of experiencing fear, whereas 'the frightener' is capable of inducing fear.

13. Magn : modifiers which added to the key-word intensify the meaning of the key-word.

e.g. Magn(memory) = excellent, prodigious, stunning, elephantine

It is normally the case that only salient values are listed under *a* particular lexical function for the lexeme in question. For this reason, the reader may have grounds for disputing the presence of 'excellent' and 'stunning' here. However, the LF 'Magn' can take a list of values increasing in strength from left to right, and the principal justification for their inclusion in this instance is to illustrate

this feature, i.e. 'excellent' is the lowest degree of 'Magn(memory)' here, whilst 'elephantine' (or perhaps 'elephant-like' or 'of an elephant') is the most forceful modifier of 'memory' in the list. Note that 'photographic', although being perhaps the most salient modifier of 'memory', is too specific to be included under this LF. Perhaps it could be inserted under the LF 'Sing'.

14. Bon : modifier expressing justification of one's being in the state of the key-word.

e.g. Bon(anger) = righteous

15. Pos(l), Pos(2) ... : positive evaluation characterising the actants of the key-word.

e.g. Pos(2)(opinion) = favourable

16. Oper(l), Oper(2) ... : semantically empty verb, taking 1st, 2nd key-word C(0) as its main object complement (CO).

e.g. Oper(1)(attention) = pay
Oper(2)(attention) = attract

By 'semantically empty', I mean semantically empty when used with the key-word mentioned with it. Hence 'pay' is semantically empty when used with 'attention', but not with 'bill'.

17. Func(0), Func(l) ... : semantically empty verb, taking C(0) as its GS and, if C(0) has actants, then lst/2nd ... actant of C(0) as its main CO.

e.g. Func(0)(wind) = blow
Func(1)(help) = come, provide
Func(2) (list) = contain, consist [of sthg]

18. Incep, Cont, Fin : Start, Continuation & End of the situation portrayed by the key-word.

e.g. IncepOper(1)(shape) = take
ContOper(l)(influence) = lose
FinFunc(0)(doubt) = evaporate

The second example here is one of Melchuk's, but perhaps it is not sufficiently descriptive. The following,

ContLiquOper(1)(influence) = lose

illustrates the decline in 'influence' inherent in the verb 'lose', which is not portrayed in the original (for more details of the LF 'Liqu', see (19) below).

N.B. Fin(P) = Incep(~P)
Cont(P) = not(Fin(P)) = not(Incep(~P))

19. Caus, Liqu, Perm : Cause, Bring an end to, Permit the situation portrayed by the key-word.

e.g. CausOper(l)(tears) = reduce [sb to ~~]
CausFunc(0)(problem) = pose
LiquFunc(0)(crowd) = disperse

N.B. Liqu(P) = Caus(~P)
Perm(P) = not(Liqu(P)) = not(Caus(~P))

As seen in the last two classes of LF, the functions combine easily with each other.

20. Degrad : verb portraying the fact that the situation exemplified by the key-word is becoming worse.

e.g. Degrad(milk) = turn, curdle
Degrad(cabbage) = go to seed

21. Excess : verb expressing the fact that the key-word is functioning excessively.

e.g. Excess(heart) = palpitate

There are many other functions, but these exemplifications serve to illustrate the sort of concepts which can be coped with by the LF's.

A further point to note is that it seems possible that generalizations can be expressed over regularities in the formation of collocations for members of semantically homogeneous classes of lexemes , as (18) and (19) above, where certain LF's are grouped together. Heid (Heid '89, p6) states that such LF's are 'explicitly linked to one another', while Heid and Raab (Heid & Raab '89, p.133) give a table of such LF's, which is used for the description of collocations. As well as those functions already shown to be linked, they include [Magn, Pos, Ver] (meaning '(high) degree'), [Able, Qual] ('ability'), and [Mult, Sing] ('count **<->** mass'). An advantage of grouping LF's together is that it may be possible to express generalizations over collocation formation for members of a particular group. Heid and Raab give the example of nouns which express information about computers, with semantic classes '*I-NOUNS(g)*' and '*I-NOUNS(fr)*' for German and French respectively as in :-

```
*I-NOUNS(g)* =  {Datei, Information,
                 Nachrichten, Verzeichnis}

*I-NOUNS(fr)* = {fichier, information,
                 messages, repertoire}
LiquFunc(0) (*I-NOUNS(g)*)  = loeschen
LiquFunc(0) (*I-NOUNS(fr)*) = supprimer
```

which says that one can use 'loeschen' in German and 'supprimer' in French to describe any of the nouns belonging to the set of '*I-NOUNS*' for the language in question, which saves one having to enter the fact in the lexicon for each individual noun belonging to that class.

Lexical entries

Having explained the LF's, we are now in a position to describe a lexical entry. This serves as a complete description of a lexeme, and has the following structure :-

(i) Introduction
- (a) key-word
- (b) morphological information *
- (c) syntactic information *

(ii) Semantics
- (d) propositional form
- (e) definition
- (f) connotations

(iii) Syntactic combinations
- (g) regime
- (h) restrictions on the regime
- (j) examples illustrating the regime and any restrictions on it
- (k) syntactic modifications of the key-word

(iv) Lexical combinations
- (1) lexical functions
- (m) examples illustrating the usage (both its meaning and possible combinations) of the key-word

(v) Phraseology

[* Of course, these two areas should not form part of the introduction. The first of them ought to constitute an independent morphological area, whereas the second belongs as part of the syntactic area (iii). However, Melchuk decides to include them in the introduction for the time being owing

to lexicographic tradition and the fact that at present, of the levels of grammatical representation in the MTT, these two areas are the least developed.]

Of the above, only (a), (b), (e) and (m) must be complete in all lexical descriptions. The others may in principle remain empty.

(g) Regime

The regime is a table where the number of columns corresponds to the number of semantic variables in the definition (i.e. the number of semantic actants of the lexeme), and the number of lines corresponds to the maximum number of possible realisations of these actants at surface level, as in (Melchuk et al '84, p.21) :-

HAINE, nom, fem.
1. ... Haine de X pour Y = ...

Regime

1 = X	2 = Y
1. de N	1. de N
2. PossPron	2. contre N
3. A	3. pour N
	4. envers N
	5. vis-a-vis de N
1. entre Npl < Nl et N2 >	

This is the regime for the French word 'haine' (= 'hatred'). We see that it is the key-word ((a), and appears in capital letters. The relevant morphological information (b) accompanies this. Any necessary syntactic information (c) would also appear at this point), and its combination with two variants of its actants appears directly beneath it (i.e. The propositional form (d) - this would be accompanied by a definition (e), and followed by any connotations (f)). We then have the regime itself (g), where the possible first actants are listed under X, with the possible realizations of the 2nd actant occurring under Y, as in :-

La haine de Pierre(X) pour Marie(Y)
Ma haine(X) contre Marie(Y)
Une haine acharnee(X) envers Marie(Y)

One point to note here is that this particular type of construction is rare in French, which prefers a construction with a past-participle or a relative rather than two PP's, as in :-

La haine eprouvee par Pierre pour Marie
La haine que Pierre eprouve pour Marie

Note also that the construction with two 'de' PP's would probably be ruled out here (or at least marked as stylistically poor).

The 'entre' PP spans both columns, indicating that it can unify either with X or Y. [Note that here we have a plural NP, consisting of two (of course, more are possible) concatenated NP's.] The rest of the entry's information follows.

Translating using the ECD

Melchuk (1988,pp 99-101) illustrates how a French speaker knowing no Russian is able to translate a sentence from French to Russian using the ECD (from Iordanskaja and Arbatchewsky-Jumarie '82). The sentence involved is the following : -

(Fr)
Les Anglais avaient une grande admiration pour son talent.
=^=
The English had a great admiration for his talent

The literal Russian translation is, as one would expect, completely ungrammatical, but by using the ECD one is able to find the required equivalent :-

(Russ)
Anglicane ispytyvali glubokoe vosxiscenie pered ego talantom
=^=
English experienced profound admiration before his talent

There are five steps involved in this operation :

(1) Find the Russian equivalent of 'admiration' in a bilingual index, namely 'vosxiscenie'.

(2) Understand that 'grande' is a value of *a* particular Lexical Function ('Magn(admiration)'), and that 'avoir' is an LF of a different kind ('Oper1(admiration)').

(3) Look up 'Magn(vosxiscenie)' and 'Oper1(vosxiscenie)' in the Russian ECD, and select the appropriate equivalent where alternatives exist, i.e. 'grande' is the lowest degree of Magn for French, so one would select 'glubokoe' as a modifier of 'vosxiscenie', as this is the lowest degree of Magn for Russian, rather than 'bezmernoe'. By the same token one would not choose 'byt' as the value of Oper1 as this does not allow any modifier with 'vosxiscenie', and as we have

just seen one exists ('glubokoe'). Note again that as salient values only are inserted under an LF for a particular lexeme, 'grande' would not appear as a value for 'Magn(admiration)', but would be inherited from a default value elsewhere in the dictionary, probably from a list of generic values for the set of LF's.

(4) Understand that 'pour' (with 'admiration') is a surface marker of the second syntactic argument (actant).

(5) Look up the corresponding marker in the pattern for Oper1 for Russian, 'ispytyvat'; it is 'pered'.

In this way it is possible to build the required target translation of the source input without too many problems. Nevertheless it is sometimes the case that a particular expression cannot be translated by following the sort of structured pattern above. For instance it might arise that an LF in one language is undefined (or disallowed) in another. Consider the following :-

(Fr) Son admiration pour ce pays durait toujours
=^= His admiration for this country lasted forever

where 'ContFunc(admiration) = durer'. However, there is no ContFunc LF for 'vosxiscenie', so one would have to invoke a paraphrase for the translation. ContFunc is equivalent to 'notFinFunc' (see (18) and (19) for other equivalents), and there is indeed a value for 'FinFunc(vosxiscenie)', this being 'proxodit', so we can translate the French 'durer' (N.B. with 'admiration' only, of course) as 'proxodit' plus negation, thus :-

(Russ) Ego vosxiscenie etoj stranoj ne proxodilo
=^= His admiration for this country did not pass

One important point to consider is how one might organize the search space. For instance, how does one know which functions to apply at the outset ? Does one begin by looking for related verbs first ? This would seem *a* sensible approach to start with, especially as the deep syntactic representations are organized in dependency terms. However, in the example cited here, it would be fairly fruitless to start looking for translations of 'avoir'. Melchuk says nothing on this matter.

What are the prospects for doing MT with this approach ?

On the face of it, it seems as though the process of producing an MT system using the ECD is just a question of implementing the technique outlined above. However, the idea of approaching MT problems via language generation is a relatively new one (in NLP as a whole, in fact, generation is a

recent preoccupation after years of neglect post-Schank). Previously the accent has been put on language understanding (i.e. analysis and transfer) with the result that the synthesis component has had to take a back seat.

The language understanding problem is so great that most if not all MT systems around today impose some restrictions on the domain of input, so that smaller dictionaries and grammars can be used. Other approaches permit only 'doctored' input, i.e. texts without ambiguities or problematic structures. These attempts at simplifying the understanding problem have led certain people to consider doing away completely with automated text-translation in favour of automatic composition, where both source and target languages have a common representation of content.

The first of three systems which attempt to use this approach is DIOGENES (Nirenburg et al '88), which is influenced by the MTT mainly in its organization of the lexicon, and which uses blackboard architecture as its control structure. The basic idea behind this approach is that a group of independent, knowledgeable 'experts' cooperate in solving a problem on a 'blackboard' (or blackboards), and they communicate only via the blackboard (a multi-level 'working-memory'). The processing in DIOGENES is concentrated in the 'knowledge sources' which are triggered by the state of the various blackboards, which contain both the input to generation and all intermediate and final results of the operation.

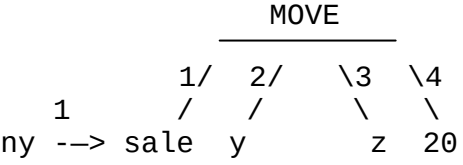
The second is the GOSSIP system (Kittredge et al '88), containing a direct implementation of the MTT for sentence generation with a domain-oriented planner (i.e. the planner restricts the lexicon by deriving the intended content of the text, thus reducing the search-space). One of the criticisms which the authors of GOSSIP level at previous similar approaches (e.g. the TAUM-METEO prototype for translating weather forecasts from French to English, Isabelle '84) is that they are too restricted in their domain of application, yet the GOSSIP system deals only with computer operating systems (note that DIOGENES was also used in this domain). Nevertheless, they make the interesting point that the MTT's semantic level deals only with the meaning of single sentences, so "any attempt to add text planning or rhetorical principles of text organisation is really external to the theory" (Kittredge et al '88, p.9).

Thirdly, Boyer and Lapalme (Boyer and Lapalme '85) set up a system to generate all the sentences which have the 'meaning' described by a given network, by using a dictionary and tree transformation rules. The system, based on the MTT, is implemented in PROLOG, which is especially suited to producing multiple solutions owing to its backtracking facility. The system uses most of the facets of the MTT as expounded

by Melchuk and Polguere (Melchuk and Polguere '87), in that semantic representations are translated to morphological representations via syntactic representations, as in the predicate (adapted from Boyer & Lapalme '85, p.181) :-

```
mtt(Sem_Rep,Text):-
        sem_comp(Sem_Rep,D_Synt_Rep),
        deep_synt_comp (D_Synt_Rep, S_Synt_Rep),
        surf_synt_comp (S_Synt_Rep,D_Morph_Comp) ,
        deep_morph_comp(D_Morph_Comp,Text).
```

Semantic networks are represented as follows (op cit, p.182):-



which is a representation of the predicate 'move(x,y,z,w)', meaning that 'x' moved up 'w' units (w=20 here) from 'y' units to 'z' units (i.e. 'Sales increased by 20 units'). The integers seen in the network are labels used later in the process of building a D_Synt_Rep. This deep syntactic representation is formed by pruning the network into connected pieces, searching the lexicon for a tree corresponding to each of these pieces and then gluing these trees together. Tree 'surgery' is then performed using tree unification to form the deep syntactic representations. Once these representations have been formed, lexical rules such as 'syn(shoot,fire).' (i.e. the synonymy rule) can apply to build other synonymous deep structures. Another such rule is that representing the LF 'S(0)', namely 's0(increase1,increase2).', which indicates that the noun corresponding to the verb 'increase1' is 'increase2'. Examples of the 'Oper1' rule are :-

```
oper1(increase2, show).
oper1(analysis,perform).
```

Melchuk believes that the ECD would "facilitate the general task of computational linguistics (CL) by dividing it into two more or less autonomous subtasks: a linguistic and a computational one" (Melchuk & Polguere '87, p.261), as borne out by the work just described on GOSSIP. The MTM, meanwhile, seeks either to produce the greatest possible number of synonomous utterances for a given semantic representation, or to reduce a number of such utterances to their semantic invariant, depending on whether one is doing analysis or generation. In this way it can be seen that the paraphrasing rules, of which there are about sixty, are reversible, viz (op cit, p.272) :-

```
C0(v) <=> Oper1 (S0 (C0)) + S0 (C0)
```

where C0 stands for the head lexeme ('v' indicates that it is a verb here) and S0 for the deverbal noun derived from C0. Such a rule can be seen at work in the following example:-

The crowd received [C0] Gorbachev well **<=>** The crowd gave [Oper1 (S0 (C0))] Gorbachev a good reception [S0(C0)].

where 'C0 = receive', 'S0(receive) = reception', and 'Oper1(reception) = give'. If we substitute these values into the rule above, we see that "receive <=> give + reception", indicating the synonomy (or, more precisely, the derivation of synonomous sentences from the same deep syntactic structure) between the verb 'receive' and the phrase 'give a reception'.

Although the rules of the MTT should be fully reversible, to have such a facility using PROLOG would be problematic, since PROLOG gives the same solution as often as there are ways to prove it, and due to the combinatorial explosions which follow if no control is taken over the backtracking procedure (by the use of 'cuts', which are reversability killers), Boyer and Lapalme only concern themselves with text generation (note also that by their very nature one's MT system incorporating such rules comes with a built-in monolingual paraphraser at no extra cost).

The authors were not primarily interested in the morphological component, and so reduced the process to a set of ad hoc procedures to cope with it. As for the surface syntactic element (i.e. getting the finished text), the PROLOG program executes a depth-first search to find the possible constituents of a sentence, and when these are found the first legal ordering of these constituents is kept as the final text. All other possibilities are then killed.

If we look at the MT systems available, most of them use a flat, declarative representation to encode lexical information. These lexical entries are by and large independent propositions in that they generally make no reference to other entries. As Melchuk says (Melchuk & Polguere '87, p. 273), "The structure of typical computational lexica is not relational: they are ... sequences of entries". He calls such an approach 'non-structuring', not meaning that these methods are without structure, but rather to emphasize the insufficiency of the structuring that they do use.

One advantage of the 'non-structuring' approach is that it is relatively simple to load up a mini-lexicon sufficient for a sub-domain that one might be working on, i.e. they are easily manipulable. In order to analyse or generate a text with such a system, one needs only to deal with the words present in the relevant text, and computationally this is very practical, as we all know that the greater the size of

the lexicon, so the slower one's system becomes (although whether the correlation is linear or exponential is unknown). Nevertheless, a non-structured lexicon prevents a system from performing high-level linguistic tasks, such as looking for alternative translations and making subtle stylistic choices.

The ECD, on the other hand, is rigorously but consistently structured, with the consequence that, as far as its meaning and coocurrence are concerned, a lexeme is specified in terms of other lexemes. One advantage of this type of dictionary is that it is possible to paraphrase a deep syntactic representation by means of the lexical functions of certain lexemes. Another is that the lexicalisation of a semantic representation follows naturally from the fact that a semantic item used for labelling a node is the actual sense of a word, i.e. a lexeme, which itself is fully characterized by a hierarchical set of properties, one of which is the presence of this lexeme in the definition of others, as well as the participation of other lexemes in its own definition. Thus by examining the semantic nodes together with their properties, one should be able to successfully analyse semantic representations in a given language by the process of lexicalisation.

It should be added at this point that these semantic representations exclude any reference to real-world knowledge, but represent the linguistic information. Therefore a fully-fledged linguistic description would need a link (namely the 'Reality-Meaning Model' in this system) between the real world and the semantics. Melchuk justifies this approach (op cit p.268) with a reference to MT. He says that the only common denominator of two texts in different languages is the state of affairs referred to, and not a semantic representation of either text. Thus these semantic representations would not be able to serve as an interlingua per se, as they reflect the characteristics of the particular language in question. They might, however, be used as a stepping stone towards an interlingua, as there can be no doubt that they bring the source and target languages closer together.

An objection to using the ECD is that the information encoded in such a lexicon is far too complicated for most computational applications. If we look at a couple of traditional problems for MT, we see that they just fall away entirely owing to the general set-up of the ECD. One of these is the notion of 'support verb'. This is taken care of by the Lexical Function 'Oper1', as in :-

```
Oper1(Question) = ask      (Eng)
Oper1(Question) = poser    (Fr)
Oper1(Pregunta) = hacer    (Sp)
Oper1(Frage) = stellen     (Ger)
```

Another problem case for syntax-based MT systems is that of anaphora resolution, e.g. "Twelve Conservative members of the club returned votes but three of them were spoiled". However, this problem would be simplified greatly in a Melchuk-style dictionary (although strictly speaking, lexicons need not (and most cannot) say anything about this phenomenon - it is just that an ECD helps to decipher such ambiguities) by the LF 'AntiVer', as 'vote' would have an entry under this LF whereas 'member' would not, i.e. 'AntiVer(vote) = spoil'.

Besides, why should it be the case that most CL lexicons are at present incomplete, inconsistent and suffer from a lack of detail ? None of these criticisms could be levelled at the ECD, after all. Such a dictionary could be employed in many areas of CL research. Perhaps the greatest advantage the ECD has over present CL lexicons is that it is reusable. All too often CL dictionaries (and this applies to knowledge representation as a whole), are written specifically for the job in hand, with the consequence that they constantly need to be revised or still worse completely rewritten if one's research changes tack even slightly. Surely a description of linguistic phenomena which is formulated in a consistent and exhaustive manner according to a particular theory is of immense value to CL applications if only because of its reusability. As we have found to our cost, the writing of a Melchuk-style entry is a non-trivial task, and one entry (for 'lose' and 'loss') took two of us about three weeks to complete, but this is merely a problem of manpower, and should not be used as a criticism of the general approach. In any case, the relationship between the time taken to write an ECD entry and that taken to write a normal dictionary entry may show the Melchuk approach to be preferable, when one considers the respective amount of information contained in the two entries. Indeed, as Melchuk says (op cit, p.274), "...the rigorous description of even one lexeme in a precise and formal framework appears as an improvement, in itself, on what has been achieved in this domain".

REFERENCES

1, Boyer, M. & G. Lapalme : "Generating Sentences from Semantic Networks" in "Natural Language Understanding and Logic Programming"; ed. Dahl, V. & P. Saint-Dizier; North Holland, Amsterdam, 1985.

2, Cruse, D.A. : "Lexical Semantics"; CUP, Cambridge, 1986.

3, Heid, U. : "Lexikalische Funktionen zur Beschreibung von Kollokationen - Notizen, Materialsammlung"; Internal Paper, Universitaet Stuttgart, 1989.

4, Heid, U. & S. Raab : "Collocations in Multilingual Generation" in : "Fourth Conference of the European Chapter of the ACL" (Proceedings); UMIST, Manchester, 1989.

5, Iordanskaja, L. & N. Arbatchewsky-Jumarie : "Lexicographic Applications *of* Lexical Functions : Two Sample Lexical Entries from an Explanatory Combinatorial Dictionary"; BLS8, Berkeley, 1982.

6, Isabelle, P. : "Machine Translation at the TAUM group"; paper presented *at* The ISSCO "Tutorial on Machine Translation", 1984.

7, Kittredge, R. et al : "Multi-Lingual Text Generation and the Meaning-Text Theory" in : "Second International Conference on Theoretical and Methodological Issues in Machine Translation of Natural Languages" (Proceedings); Pittsburgh, 1988.

8, Melchuk, I.A. et al. : "Dictionnaire explicatif et combinatoire du francais contemporain"; Les Presses de l'Universtite de Montreal, Montreal, 1984.

9, Melchuk, I.A. : "Dependency Syntax"; Suny, New York, 1988.

10, Melchuk, I.A. & A. Polguere : "A Formal Lexicon in the Meaning-Text Theory (or How to Do Lexica with Words)"; Computational Linguistics 13, ACL, Morristown NJ, 1987.

11, Melchuk, I.A. & A.K. Zholkovsky : "Explanatory Combinatorial Dictionary of Modern Russian"; Wiener Slawistischer Almanach, Sonderband 14, Vienna, 1984.

12, Nirenburg, S. et al : "Lexical Realization in Natural Language Generation" in : "Second International Conference on Theoretical and Methodological Issues in Machine Translation of Natural Languages" (Proceedings); Pittsburgh, 1988.

13, Sinclair, J. (ed.) : "COBUILD English Language Dictionary"; Collins, London, 1987.

14, Zubizarretta, M.L. : "Levels of Representation in the Lexicon and in the Syntax"; Foris, Dordrecht, 1987.

'Explanatory Combinatorial Dictionary' (ECD), an essential component of Melchuk's 'Meaning-Text Model' (MTM),

1) Levels of description in the MTM :
semantics,
syntax (with deep and surface levels),
morphology (also with a deep-surface distinction)
phonetics (or orthographies, again with the same subdivisions).

2) Lexical Functions

1. Syn, Syn(<), Syn(>), Syn(^) : synonyms and quasi-synonyms, or inexact synonyms, where the arrows signify having a narrower / wider / intersecting meaning than that of f(X).

e.g. Syn (aid) = help
Syn(<) (respect) = veneration [a particular kind of 'respect']
Syn(>)(veneration) = respect
Syn(^)(demonstration) = exhibition
Syn(^)(demonstration) - protest

The last two examples show that 'demonstration' sometimes means 'exhibition', and sometimes 'protest'.

2. Conv(ijkl) : converse relation to that of the key-word, where the subscripts refer to the actants associated with the key-word, i.e. Conv(312)(Key) means that the 3rd, 1st and 2nd actants of the key-word become the 1st, 2nd and 3rd actants respectively of Conv(Key).

e.g. Conv(3214)(sell) - buy

i.e. The shop sold a paper to me for 50p
3 2 1 4
=> 1 2 3 4
I bought a paper from the shop for 5Op

3. Anti, Anti(<), Anti(>), Anti(^) : antonyms (see 1 for details).

4. Gener : generic word whose combination with a syntactic category derived from the key-word is synonymous with that key-word.

e.g. Gener(republic) = [republican] state

5. Figur : figurative use of a word, where combined with the key-word is synonymous with the key-word.

e.g. Figur(fog) = blanket [of fog]

6. Syntactic derivations : S(0), V(0), A(0), Adv(0), where 'syntactic derivation of a lexeme C(0)' means another lexeme or syntagm having the same meaning as C(0) but being of a different syntactic category.

e.g. S(0)(honest) = honesty
A(0)(school) = scholarly
V(0)(loss) = lose
Adv(0)(honest) = honestly

7. S(l), S(2), S(3) ... : typical noun for the 1st, 2nd, 3rd ... actant of the key-word.

e.g. S(l)(crime) = criminal
S(2)(crime) = victim
S(l)(buy) = buyer
S(2)(buy) = product
S(3)(buy) = seller
S(4)(buy) = price

8. S(instr), S(loc), S(med), S(mod), S(res) : typical nouns for the instrument, location, medium, mode, result of the key-word.

e.g. S(instr)(paint) = brush
S(med)(paint) = paint
S(res)(paint) = picture

9. Sing : a 'part' of a 'whole'.

e.g. Sing(fleet) = ship
Sing(flock) = sheep

10. Mult : converse of Sing.

e.g. Mult(ship) = fleet

11. A(l), A(2) ... : typical attribute for the 1st, 2nd actant of the key-word.

e.g. A(l)(beard) = bearded
A(2)(consider) = in question

i.e. the attribute to describe a person with a beard is 'bearded', where that person is the first actant of 'beard'. The second example illustrates the synonomy of the two phrases : "If you consider the author" and "The author in question", where 'author' is the second actant in both cases.

12. Able(l), Able(2) ... : function describing 'being capable of ...'

e.g. Able1(fear) = the frightened (person)
Able2(fear) = the frightener

i.e. 'the frightened (person)' is capable of experiencing fear, whereas 'the frightener' is capable of inducing fear.

13. Magn : modifiers which added to the key-word intensify the meaning of the key-word.

e.g. Magn(memory) = excellent, prodigious, stunning, elephantine

14. Bon : modifier expressing justification of one's being in the state of the key-word.

e.g. Bon(anger) - righteous

15. Pos(l), Pos(2) ... : positive evaluation characterising the actants of the key-word.

e.g. Pos(2)(opinion) = favourable

16. Oper(l), Oper(2) ... : semantically empty verb, taking 1st, 2nd ... actants of situation C(0) as its grammatical subject (GS) and the key-word C(0) as its main object complement (CO).

e.g. Oper(1)(attention) = pay
Oper(2)(attention) = attract

By 'Semantically empty', I mean semantically empty when used with the key-word mentioned with it. Hence 'pay' is semantically empty when used with 'attention', but not with 'bill'.

17. Func(0), Func(l) ... : semantically empty verb, taking C(0) as its GS and, if C(0) has actants, then lst/2nd ... actant of C(0) as its main C0.

e.g. Func(0)(wind) = blow
Func(1)(help) = come, provide
Func(2)(list) = contain, consist [of sthg]

18. Incep, Cont, Fin : Start, Continuation & End of the situation portrayed by the key-word.

e.g. IncepOper(l)(shape) = take
ContOper(1) (influence) = lose
FinFunc(0)(doubt) = evaporate

The second example here is one of Melchuk's, but perhaps it is not sufficiently descriptive. The following,

ContLiquOper(1)(influence) = lose

illustrates the decline in 'influence' inherent in the verb 'lose', which is not portrayed in the original (for more details of the LF 'Liqu', see (19) below).

N.B. Fin(P) = Incep(~P)
Cont(P) = not(Fin(P)) = not(Incep(~P))

19. Caus, Liqu, Perm : Cause, Bring an end to, Permit the situation portrayed by the key-word.

e.g. CausOper(1)(tears) = reduce [sb to ~~]
CausFunc(0)(problem) = pose
LiquFunc(0) (crowd) = disperse

N.B. Liqu(P) = Caus(~P)
Perm(P) = not(Liqu(P)) = not(Caus(~P))

As seen in the last two classes of LF, the functions combine easily with each other.

20. Degrad : verb portraying the fact that the situation exemplified by the key-word is becoming worse.

e.g. Degrad(milk) = turn, curdle
Degrad(cabbage) = go to seed

21. Excess : verb expressing the fact that the key-word is functioning excessively.

e.g. Excess(heart) = palpitate

3) Generalizations can be expressed over regularities in the formation of collocations for members of semantically homogeneous classes of lexemes.

Advantages of this ?

Heid and Raab (Heid & Raab '89, p.133) give the example of nouns which express information about computers, with semantic classes '*I-NOUNS(g)*' and '*I-NOUNS(fr)*' for German and French respectively.

I-NOUNS(g) = {Datei, Information, Nachrichten, Verzeichnis}

I-NOUNS(fr) = {fichier, information, messages, repertoire}

LiquFunc(0) (*I-NOUNS(g)*) = loeschen

LiquFunc(0) (*I-NOUNS(fr)*) = supprimer

which says that one can use 'loeschen' in German and 'supprimer' in French to describe any of the nouns belonging to the set of '*I-NOUNS*' for the language in question, which saves one having to enter the fact in the lexicon for each individual noun belonging to that class.

═ ═

4) Lexical entries

HAINE, nom, fem.
1. ... Haine de X pour Y = ...

	Regime	
	1 = X	2 = Y
	1. de N	1. de N
	2. PossPron	2. contre N
	3. A	3. pour N
		4. envers N
		5. vis-a-vis de N
	1. entre Npl <,N1 et N2 >	

═ ═

5) Translating using the BCD

Melchuk (1988,pp 99-101) (from Iordanskaja and Arbatchewsky-Jumarie '82) shows how a French speaker knowing no Russian can translate from Fr > Russ. The sentence involved is the following :-

5a. (Fr) Les Anglais avaient une grande admiration pour son talent,
[lit - The English had a great admiration for his talent]

5b. (Russ) Anglicane ispytyvali glubokoe vosxiscenie pered ego talantom.
[lit - English experienced profound admiration before his talent]

There are five steps involved in this operation.

(1) Look up 'admiration' in a bilingual index, = 'vosxiscenie'.

(2) Magn(admiration) = grande

Oper1(admiration)= avoir

(3) (a) Look up 'Magn(vosxiscenie)' and 'Oper1(vosxiscenie)' in the Russian BCD,
(b)(i) select the appropriate equivalent where alternatives exist, i.e. 'grande' is the lowest degree of Magn for French, so select 'glubokoe' as a modifier of 'vosxiscenie'
(ii) do not choose 'byt' as the value of Oper1 as this does not allow any modifier with 'vosxiscenie', and as we have just seen one exists ('glubokoe').

(4) Understand that 'pour' (with 'admiration') is a surface marker of the second syntactic argument (actant).

(5) Look up the corresponding marker in the pattern for Oper1 for Russian, 'ispytyvat'; it is 'pared'.

Now a more problematic case.

5c. (Fr) Son admiration pour ce pays durait toujours
[= His admiration for this country lasted forever]

Problem : 'ContFunc(admiration) = durer'. But there is no 'ContFunc' LF for 'vosxiscenie', therefore invoke a paraphrase for the translation.

Solution : ContFunc is equivalent to 'notFinFunc' (see (18) and (19) for other equivalents), and 'FinFunc(vosxiscenie) = proxodit', ==> translate 'durer' (N.B. with 'admiration' only, of course) as 'proxodit' plus negation, thus :-

5d. (Russ) Ego vosxiscenie etoj stranoj ne proxodilo
[lit - His admiration for this country did not pass]

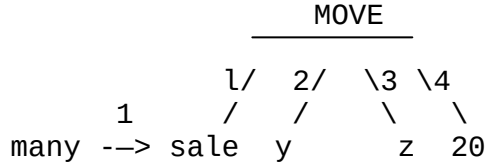
═ ═

6. What are the prospects for doing MT with this approach ?

6a. DIOGENES (Nirenburg et al '88),
— influenced by the MTT mainly in its organization of the lexicon
— uses blackboard architecture as its control structure.

6b. GOSSIP (Kittredge et al '88),
— contains a direct implementation of the MTT for sentence generation
— has a domain-oriented planner, which reduces the search-space.

6c. Boyer and Lapalme (Boyer and Lapalme '85)
— generates all the sentences having the 'meaning' described by a network, using a dictionary and tree transformation rules.
— implemented in PROLOG
— semantic networks are represented as follows :



— representation of the predicate 'move(x,y,z,w)', i.e. 'x' moved up 'w' units (w=20 here) from 'y' units to 'z' units (= 'Sales increased by 20 units').

— Lexical Rules
e.g. syn(shoot,fire).
s0(increase1,increase2).
oper1(increase2,show).

oper1(analysis,perform).

Paraphrasing Rules (about 60), are reversible

e.g. C0 (v) <=> Oper1 (S0 (C0)) + S0 (C0)

where C0 stands for the head lexeme ('v' - verb) and S0 for the deverbal noun derived from C0.

Example ?

The crowd received [C0] Gorbachov well

<==>

The crowd gave [Oper1 (S0 (C0))] Gorbachov a good reception [S0(C0)].

i.e. 'C0 = receive',
'S0(receive) = reception',
'Oper1(reception) = give'.

By substitution of these values into the rule above, i.e. "receive <-> give + reception", indicating the synonomy between the verb 'receive' and the phrase 'give a reception'.

Lexical Design in NT systems

Advantages of the 'non-structuring' approach :
— simple to load up a mini-lexicon sufficient for a sub-domain, i.e. they are easily manipulable
— to analyse or generate a text with such a system, only have to deal with the words present in the text. Computationally this is very practical.

Disadvantages :
— system is prevented from performing high-level linguistic tasks, such as looking for alternative translations and making subtle stylistic choices.

Advantages of the ECD :
— rigorously but consistently structured
— possible to paraphrase a deep syntactic representation by means of the lexical functions
— the lexicalisation of a semantic representation follows naturally from the fact that a semantic item used for labelling a node is the actual sense of a word, i.e. a lexeme

NB, Melchuk (op cit p.268) : "The only common denominator of two texts in different languages is the state of affairs referred to, and not a semantic representation of either text."

Disadvantages of the ECD :
— it is claimed that the information encoded in it is far too complicated for most computational applications.

But if we look at a couple of traditional problems for NT, we see that they just fall away entirely owing to the general set-up of the ECD.

(i) Support Verbs

Oper1(Question) = ask (Eng)
Oper1(Question) = poser (Fr)
Oper1(Pregunta) = hacer (Sp)
Oper1(Frage) = stellen (Ger)

(ii) Anaphora Resolution

e.g. "Twelve Conservative members of the club returned votes but three of them were spoiled".

AntiVer(vote) = spoiled

REFERENCES

1, Boyer, M. & G. Lapalme : "Generating Sentences from Semantic Networks" in "Natural Language Understanding and Logic Programming"; ed. Dahl, V. & P. Saint-Dizier; North Holland, Amsterdam, 1985.

2, Cruse, D.A. : "Lexical Semantics"; CUP, Cambridge, 1986.

3, Heid, U. : "Lexikalische Funktionen zur Beschreibung von Kollokationen - Notizen, Materialsammlung"; Internal Paper, Universitaet Stuttgart, 1989.

4, Heid, U. & S. Raab : "Collocations in Multilingual Generation" in : "Fourth Conference of the European Chapter of the ACL" (Proceedings); UMIST, Manchester, 1989.

5, Iordanskaja, L. & N. Arbatchewsky-Jumarie : "Lexicographic Applications of Lexical Functions : Two Sample Lexical Entries from an Explanatory Combinatorial Dictionary"; BLS8, Berkeley, 1982.

6, Isabelle, P. : "Machine Translation at the TAUM group"; paper presented at The ISSCO "Tutorial on Machine Translation", 1984.

7, Kittredge, R. et al : "Multi-Lingual Text Generation and the Meaning-Text Theory" in : "Second International Conference on Theoretical and Methodological Issues in Machine Translation of Natural Languages" (Proceedings); Pittsburgh, 1988.

8, Melchuk, I.A. et al. : "Dictionnaire explicatif et combinatoire du francais contemperain"; Les Presses de l'Universite de Montreal, Montreal, 1984.

9, Melchuk, I.A. : "Dependency Syntax"; Suny, New York, 1988.

10, Melchuk, I.A. & A. Polguere : "A Formal Lexicon in the Meaning-Text Theory (or How to Do Lexica with Words)"; Computational Linguistics 13, ACL, Morristown NJ, 1987.

11, Melchuk, I.A. & A.K. Zholkovsky : "Explanatory Combinatorial Dictionary of Modern Russian"; Wiener Slawistischer Almanach, Sonderband 14, Vienna, 1984.

12, Nirenburg, S. et al : "Lexical Realization in Natural Language Generation" in : "Second International Conference on Theoretical and Methodological Issues in Machine Translation of Natural Languages" (Proceedings); Pittsburgh, 1988.

13, Sinclair, J. (ed.) : "COBUILD English Language Dictionary"; Collins, London, 1987.

14, Zubizarretta, M.L. : "Levels of Representation in the Lexicon and in the Syntax"; Foris, Dordrecht, 1987.