[Proceedings of the *Conference on Theoretical and Methodological Issues in Machine Translation of Natural Languages*, Colgate University, Hamilton, New York, August 14-16, 1985]

A MU_1 View Of The <C,A>, T Framework In Eurotra

Doug Arnold, Lieven Jaspaert, Rod Johnson, Steven Krauwer, Mike Rosner, Louis des Tombe, Nino Varile, and Susan Warwick †

1. Background

The background to this paper is the attempt within EUROTRA to develop a general framework for research and development work in MT, providing in particular an environment which facilitates reasoning about the relationships between the representations that are necessary for automatic translation between natural languages. The more immediate background is the attempt to apply this framework experimentally on a small scale in developing a "proto-EUROTRA" (affectionately "the toy") over the summer and autumn of this year. The aim of this paper is to give a reasonably clear idea about the user language and theories of representation for this experiment (the Mul level in terms of the paper presented by Louis des Tombe), and to indicate en route some of the directions for further work. It reports work in progress, and is thus deliberately speculative, programmatic, and rather informal.

For concreteness section 2 gives a brief and rather casual restatement of some of the key assumptions behind this work.

2. Assumptions

-- Translation between natural languages involves a sequence of primitive translations between a number of levels of representation.

-- A representational level is a language L generated by a grammar G, and an interpretation I, specifying respectively the syntactically and semantically well-formed expressions of L. G consists of a set of atoms A, and constructors C. Syntactically, Cs are of the form:

(name {type}) [$argspec_1$,..., $argspec_N$]

and atoms are of the form:

(name {type})

where {type} is a set of features (conforming to some theory of features defining the notion of "well-formed type"). The same name may be shared by several atoms or constructors. Each constructor has in addition a constructor-name which identifies it uniquely, and which is used by the translation rules. The theory of features we assume here is extremely impoverished, allowing little more than that a type is a collection of attribute-value pairs. Syntactically, if c is a constructor of arity n, and each of u_1 .... u_N are well-formed, then

† This is a joint paper by the individuals listed alphabetically above. Arnold did the actual writing, and I am the referent of all uses of the first person singular; in the nature of things I am responsible for the places where this is poorly expressed or misrepresents the collective view. The paper is intended to fit together with the paper by the same authors presented by Louis des Tombe, but (apart from the title) it is supposed to be readable in isolation. I'd like to dedicate it to the memory of Dave Kilby, who died in June -- he deserved something clearer and more polished than this, but it is the first thing I have written since he died, and he was always generous and patient with those who knew less than he did and could not think as clearly as he did.

$$c\ [u_1, ..., u_N]$$

is well-formed. Semantically, it is only well-formed if each argspec of c unifies (according to some definition appropriate to that G) successfully with the appropriate u; its interpretation is the evaluation of this unification in the appropriate environment of variable bindings. “Constructs” are atoms and the result of applying constructors to atoms. (The examples in the next section will clarify most of this).

-- The relation between a level i and an “adjacent” level J is given by a translator described by a set of T rules, which relate the generating devices G_i and Gj.

-- To be “primitive” the operation of such a translator must be: (a) *compositional,* and (b) *"one shot."*

A translation relation T between $G_i = <C_i, A_i>$and $G_j = <C_j, A_j>$ is strictly compositional if T

maps A_i into A_j , and there is a mapping t from C_i into C_j such that if

exp= c $[u_1,...,u_N]$.

then the translation of exp is:

t(c) [$\mathrm{T}(u_1), ..., \mathrm{T}(u_N)$]

In addition to strict compositionality, the following relaxations are allowed:

(a) the number and/or order of arguments of c and t(c) may differ.

(b) rather than being a and actual member of the constructors for a given G, either c, or t(c) may be a function made up of variables, and atoms and constructors of G.

(Some further relaxations are may be motivated, however, e.g. allowing that in some cases as well as (functions composed of) actual constructors and atoms, translators might relate descriptions of constructions).

A translator is one-shot if it takes only well-formed expressions of G_i and yields only (syntactically and semantically) well-formed expressions of G_j.

Syntactically, T rules are written:

atom = = > atom

constructor-name = = > constructor name

construction-name (variables) = = > constructor name (variables)

c-function = = > c-function

where a c-function is of the form:

constructor-name[*sub-item*$_1$ ..., *sub-item*$_N$]

and where sub-items are either atoms, or constructor descriptions, or variables (integers standing for classes of source or target constructs, as appropriate). We will call a T rule *simple* if at most one side is a c-function: ideally, we would like all translators to be composed of simple T rules, but see below. Both sides of T rules in transfer are required to be simple.

The intention is that that whatever syntactically and semantically well-formed source language constructs result from interpreting the lhs of a T rule will be translated as whatever syntactically and semantically well-formed constructs result from interpreting the rhs of the rule.

As regards the *pragmatics* of using these languages, there are essentially three expressive devices available to the linguist: Gs (rules for defining individual languages), T rules (relating languages), and a feature theory (theory of types); and it is to some extent a matter of choice which is chosen for a particular task. However, it is the Gs that we are most interested in here, since we are interested in the idea of translation as a compositional process -- that is a relatively simple relationship between the basic items and the modes of combination available in two languages. Thus, users of these devices are encouraged to put the main burden of work on the Gs, and in keeping with this fairly strict restrictions have been imposed on the form of T rules, and the feature theory is extremely impoverished.

This is rather casual, and contains some obvious omissions (a proper account of a feature theory, and a procedural semantics, e.g.), but should serve the purpose of exposition.

3. Levels of Representation

The aim of the rest of the paper is to outline the substantive linguistic theory involved in the proto-EUROTRA experiment It should make the description of the framework more concrete, and give some idea of what it is like to work with in practice. Both this paper and the paper presented by Johnson will discuss some of the limitations of the framework we have assumed. This is as it should be: it is the purpose of this experiment to lead to the discovery of problems -- especially “natural classes” of problems.

At the lowest level(s) of representation, texts are simply strings (that is concatenations) of atoms. We will call a level *structural* if its constructors assign more structure than this. The hypothesis to be investigated in proto-EUROTRA is that there are three structural levels of representation in this sense:

- Eurotra Constituent Structure (ECS): where texts are represented as constructions of surface constituents.
- Eurotra Relational Structure (ERS): where texts are represented as constructions of surface constituents.
- Interface Structure (IS): a level based on semantic dependence relations -- the input to transfer, providing the interface between monolingual components.

In what follows, upper case normally indicates variables, lower case constants, underline is the anonymous variable, and ‘...’ is a sloppy variable for what is too tiresome to specify in an exposition like this. For brevity we will sometimes write exp’ to mean the translation of exp.

For the purpose of this experiment, we assume that the internal structure of words is never represented explicitly (the intuition behind this is that translation is fundamentally a relation between words -- rather than e.g. semantic formulae or morphemes, but there is clearly more than this to say on the topic of derivational morphology and compounding. More than this is said in [1], but it will not be pursued here). Given this lack of any systematic morphology the non-structural levels are not of great linguistic interest. It is, however rather easy to exemplify some simple T rules here. Suppose there is a level (1) at which atoms are simply word forms, and the single constructor is the string concatenator concat, of arbitrary arity; and a level (2) with the same constructor, but where atoms are words with associated lexical features. One would expect T rules such as the following:

```
  (1)                     (2)

concat     = = >          concat

(hit)      = = >          (hit {cat=v})

(hit)      = = >          (hit {cat=n})

(aux)      = = >          concat [(a {cat=p}), (les,{cat=det})]

concat [(computer), (terminal)] = = > (computer-terminal, {cat=n})
```

3.1. ECS

3.1.1. Overview of ECS

The hypothesis proposed here is that the first level of structural representation for a language should be sensitive to only the superficial distributional categories of the language: i.e. a configurational/categorial representation indicating surface syntactic category and constituent structure, with words as its basic expressions.

All the standard distributional evidence is that this will require more than one bar projection of each category (i.e. X^n for n>l -- if we think of S as a projection of V, allowing at least VP as well as S and V -- the precise number of the projections for particular categories and languages is a matter for investigation, and left open here). Such a level will not explicitly represent grammatical relations (such as government, or subject-of) or semantic relations (such as agent-of, or pronoun-antecedent relations). There thus seems no reason to include empty categories, whose major purpose is to make available information about such relations.

The existence of such a pure surface constituency level is a departure from "standard EUROTRA" (i.e. the theory of representation discussed in [1]), but is not otherwise very controversial: such descriptions are typically extremely easy to produce and process, and it is widely believed that it is much easier to describe the (grammatical and semantic) relational structure of texts in terms of such a representation than it is to describe it directly in terms of a nonstructural (string) representation.

3.1.2. ECS Constructors and atoms

Since the constructions at this level are surface constituents, it is natural to think of the constructors as essentially phrase structure rules. It also seems reasonable to abstract away from details of inflectional morphology at this level, so the ECS atoms should correspond to words (rather than the word-forms), and be named accordingly.

For example, the following might be atoms for English ECS (here and below, I have included a number of comments to aid readability, but I think the general intention of what is written will be clear to anyone who has worked with formalisms whose semantics involve unification:

```
(like, {cat=v, num = sing, per=3, tense=pres}) ; i.e. 'likes'
(like, {cat=v, num = sing, per=l, tense=pres}) ; i.e. '(I) like'
(example, {cat=n, num = sing})
(example, {cat=n, num = pl}) ; i.e. 'examples'
(computer-terminal, {cat=n, num = sing}) ; the internal structure of
(legalisation, {cat=n, num=sing})           ; compounds and derived
                                            ; words is ignored here
(I, {cat=prop-noun, per=l, num=sing})
(Jules, {cat=prop-noun, per = 3, num = sing})
(a, {cat=det, num = sing})
(the, {cat=det, num = X}   ; the number of 'the' is not specified
                           ; lexically: the rule below will give it
                           ; the same value as the n it modifies).
```

For the fragment of English described by the following rules we might product the following set of Cs:

PS rules:

```
S --> NP VP
NP --> DET N
NP --> Proper-N
VP --> V NP
```

Constructors:

Cs= (s, {declarative=yes, tense=Z }) [(np, {num=X, per=Y})
(vp, {num=X, per=Y, tense=Z})

Among other things, this enforces person-number agreement of np and vp, and passes the value of tense from vp to s.

Cnp/det-n = (np, {num-X, per = Y}) [(DET, {cat=det, num=X})
(N, {cat=n, num=X, per=Y})]

DET and N have to be variables, because they must match to the name of atoms (which are word names at this level), otherwise, det and n must agree in number, which is passed to the np, np shares the person value of n -- if this is not specified lexically by the N, Cs will ensure it is assigned by the vp, hence by the verb.

Cnp/proper= (np {num=X, per=Y}) [(PROPN, {cat-propn, num=X, per=Y})]

Cvp= (vp, {num=X, per+y, tense=Z}) [(V, }cat=v, num = X, per = Y, tense=Z})
(np, { })]

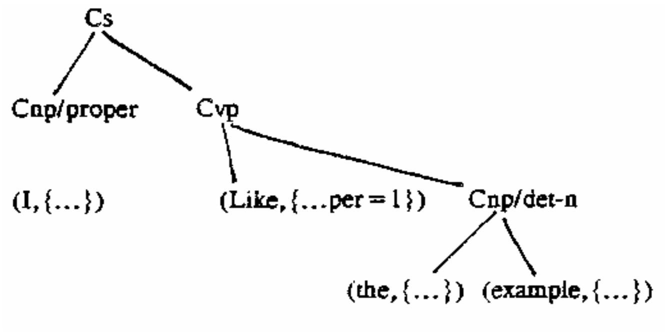
vp inherits person number and tense values from v; only the name of the object np is considered, its type (features) is ignored.

3.1.3. Derivation, interpretation, and inspection trees

Obviously, the hierarchical nature of expressions generated by Gs for structural levels means they are equivalent to trees. However, only in simple cases will these trees correspond to normal linguistic representations such as phrase structure and dependency trees. More generally, they will correspond to *derivation trees* where each terminal node is labeled with a basic expression (an atom), and each nonterminal node is labeled with a constructor-name; constructors are to be understood as applying to their substructure. The interpretation of a < C,A > expression is also equivalent to a tree (an *interpretation tree),* where each node is labeled by the name and features of an atom or constructor, and the daughters of a node appear in square brackets after it, Again these are somewhat distant from traditional linguistic trees: readability may involve abstracting away from some aspects of the derivation and interpretation trees to some other graphical representation (typically a normal phrase structure or dependency tree). This we call an *Inspection* tree. It exists purely for the convenience of the linguist, and plays no role in the working of the system.

As an example, application of constructors to atoms as indicated by the derivation tree (1) will evaluate by unification to the interpretation tree (2). This is clearly equivalent to a standard phrase structure tree, which is the obvious inspection representation.

(1)



(2)

```
(s {declarative=yes, tense = pres})
        [ (np {num = sing, per=l})
             [ (I, {cat=propn, num = sing, per=l}) ]
          (vp, {num=sing, per=l, tense=pres})
              [ (like, {cat=v, num = s, per=l, tense=pres})
                (np, {num = sing, per=Y})
                    [ (the, {cat=det, num = sing})
                      (example, {cat=n, num = sing, per = Y}) ] ]]
```

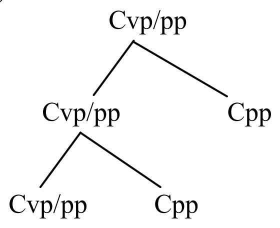
A problem that deserves special mention here is that of optional modifiers: The <C,A> syntax distinguishes Cs by their arity but it is well known that constructions allow for indeterminate numbers of modifiers. The solution to this proposed here is that each G contain a set of Cs whose intuitive purpose is to 'include' extra modifiers one at a time into existing constructions. The following would handle extra (optional) pps inside vp:

Cvp/pp = (vp, {num=X, per-Y...}) [(vp, {num = X, per=Y...}), (pp, { })]

The intended meaning is that given an VP construction and a pp construction, this will create a new construction, whose type is identical to that of the original vp, but which includes the pp. A similar treatment would seem appropriate for handling coordinate constructions.

The existence of such constructors complicates the relationship of derivation and interpretation trees and inspection trees. It will produce derivation trees such as (3), and inspection trees such as (4):

(3)

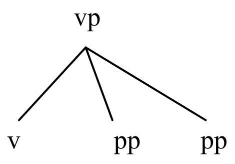


(4)

```
        (vp {...})
           [ (vp { })
                [(vp{ })]
                 (pp { }) ]
             (pp{  })]
```

The natural inspection level tree is probably along the lines of (5); it can be derived from (4) by collapsing nodes which are labeled identically:

(5)



We use traditional terms like geometry and labeling to refer to aspects of this inspection representation. We will also use the term 'construction' rather loosely to mean both appropriate linguistic entities (as in 'dependency construction'), and the result of applying a construction to some arguments. However, as the above example indicates, not every construction in the latter sense exactly corresponds to one in the former sense.

3.1.4. Translation of ECS and non-structural levels

For atoms, the translation from non-structural into structural levels is rather straightforward (depending mainly on whether atoms at prestructural levels correspond to words or word forms). For constructors, there are a number of options, depending on the nature of the pre-structural constructor (the exact nature of concat), and parsing strategy the translator is intended to mimick. For example, translation concat as Cs (with appropriate procedural semantics to ensure that e.g. the failure of Cs to unify with the translations of the arguments of concat results in the recursive application of appropriate np and vp constructors) will effectively produce a top down parse. There are ways in which this can be made more efficient, of course, but this transition from non-structural to structural level is something of a degenerate case of compositional translation, and is far from the most troublesome aspect of MT.

3.1.5. Overgeneration

Clearly, the ECS G will over-generate: semantically/pragmatically anomalous expressions apart, it will assign an analysis to things like:

(6) I like I the computer terminal.

The intention is that this kind of over generation will be harmless because of the effect of T-rules involving ECS (e.g. (6) will not be translatable in an ERS structure because ERS constructors will take account of the syntactic frames of verbs).

3.2. ERS

3.2.1. Overview of ERS

The intuitive basis of ERS is that of the grammatical relation of dependency that holds between the heads of phrases and their dependents (complements and modifiers). The head of a phrase is the item which intuitively:

(a) possesses a ‘frame’ of ‘slots’ to be filled by complements -- which are obligatory in the simplest case, and;

(b) is the focus of modification by other - typically optional -- items (see [1] for discussion). The head is said to **govern** the other members of the construction.

The intention is that ERS atoms correspond to (readings of) words, and that ERS abstracts away from configurational distortions of dependency relations (such as are caused by movement rules) by representing **complete** and **coherent** syntactic dependency constructions (the terminology is that of LFG [3]). A construction is complete if it contains all the dependents of its head, and coherent if it contains only dependents of its head. Extraposition as in (7):

(7) I read a book yesterday about phrenology.

thus produces a surface construction which is incomplete (the construction headed by *book)* and one which is incoherent (the construction headed by *read).* Thus the idea is that items that are superficially displaced from their syntactic constructions (e.g. extraposition) will be reunited with their syntactic co-dependents -- e.g. extraposition and wh-movement will be ‘undone’ at ERS. Similarly, the fact that e.g. subjects of control predicates *(try* etc.) are subjects of the associated embedded clauses will be represented explicitly. Roughly:

```
ECS:    Who do you think [ will win ].
ERS:    Do you think [ who will win ].

ECS     Jules tried [ to go ]
ERS     Jules --subj-- tried [ to go ]
                 \------ subj ------/
```

Atoms and constructions are subclassified according to category (lexical and phrasal category respectively).

The existence of such a level of pure syntactic dependency is something of a departure from standard EUROTRA, which attempts to combine configurational and relational syntactic representations by judicious use of empty elements. The motivation for ERS is the fact that languages appear to differ more in terms of phrase and configurational properties than they do in terms of relational ones: thus a relational representation is a better basis for translation than a configurational one, and IS should be relational. Furthermore, it is widely thought that a level that involves explicit representation of syntactic relations is a useful intermediate step between superficial and semantic representations. Since there is evidence that syntactic category is useful in transfer, at least some such information (major category, number, person ...) should be preserved at ERS.

As will appear, the version of IS proposed below requires heads to be lexical, but allows for distinctions between the type of a construction and the type of its head (i.e. it requires a lowered-gov, X^1 representation reminiscent of that in GETA [2], and in standard EUROTRA. The distinction between this and pure *(X^0)* is that the latter allows only lexical nodes, expressing government geometrically as domination, whereas an X^1 requires each node to have a lexical daughter which governs the rest of the construction, hence expressing government as a variety of command). It seems reasonable to adopt the same view at ERS.

3.2.2. Atoms and Constructors at ERS

Given this, there are a number of different candidates for ERS constructors (e.g. words (atoms), syntactic relations (phrasal), syntactic categories, ...); however, given the intuition that dependency relations are between constructions and words, and that at least some words share the same syntactic dependency structure, the obvious constructors are classes of lexical items -- as defined by their dependency structures (i.e. their syntactic dependency frames).

Thus, we will take the name of a constructor to be the name of a grammatical relation (gov, subj, obj, etc.), include category in the type, and refer to the frame of the gov in the constructor. The following constructors would construct transitive sentences involving verbs which require nominal subjects and objects, nps with a nominal head and det modifiers, and nps with proper noun heads.

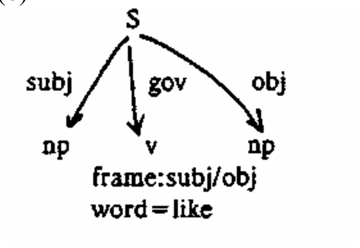
```
Csub/obj=  (_, {cat=s}) [ (gov, {cat=v, frame = subj/np-obj/np})
                          (subj, {cat=np})
                          (obj, {cat=np})
```

The name of the constructor as a whole is unspecified, since the grammatical relation of constructions is not determined intrinsically but by whatever construction they are themselves part of. For brevity, some ‘minor’ properties such as number and person, are ignored in these examples.

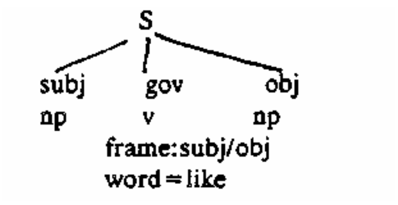
```
Cdefdet-n=   (_, {cat = np, definiteness = def}) [ (gov, {cat = n})
                          (mod, (cat=det})

Cpropn=   (-, {cat=np}) [ (gov, {cat=propn}) ]
```

Since it is not a simple configurational level, the relationship of the <C,A> representation of ERS to an inspection tree is slightly less obvious than with ECS. Either of the following are reasonable:

(8)



(9)



The set of atoms at ERS will be closely related to that at ECS -- in particular, since both represent words of the same language, it will never be the case that ECS words are translated as ERS constructions, or vice versa. There are two differences worth mentioning. First, ERS atoms should be differentiated according to their frames so that the relation of ECS atoms to ERS atoms will typically be one to many. Secondly, since we want the atoms at ERS to enter into dependency relations, we may want the name of each atom to be available for this; since it clearly cannot be specified lexically, the name of each atom would have to be 'blank' at ERS; the word it represents would then be part of the type; e.g.

(_, {word=like, cat=v, frame= subj/npobj/np, .,,.})

(an incomplete specification of the transitive verb 'like' ignoring number, person, etc ...)

3.2.3. Translation of ECS and ERS

For atoms, the ECS <---> ERS translations are always atom to atom, hence always straightforwardly primitive. Likewise, there will be many cases of straightforward constructor-constructor translations, involving at most re-ordering of arguments:

ECS	ERS
Cnp/proper	Cpropn
Cnp/det-n (1,2)	Cdet-n (2,1)

(the ECS constructor takes its arguments in the order det, n, the ERS one in the order n, det).

The translation rule that eliminates vps is also straightforward. Roughly: Cvp is assigned no translation at ERS, instead, Cs will be translated by a set of T rules, including the following:

Cs (A, Cvp (B, c)) = = > Csubj/obj (B, A, C')

in terms of the inspection trees:

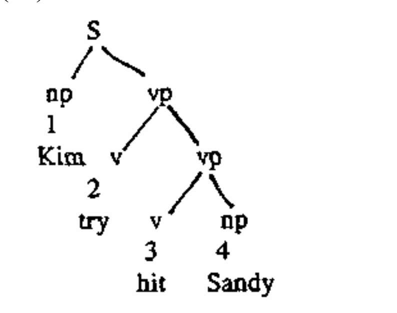
(10)

```
          S            ==>          S
  -----------------           ----------------------------------
  |                |           |              |             |
  np              vp          gov            subj          obj
  A          --------------    B              A             C
             |             |
             v            np
             B             C
```

3.2.4. Control and unbounded dependencies

Given the rather impoverished feature theory we have assumed, it is only possible to express the ‘control’ relation (i.e. the fact that the subject of verbs like ‘try’ is also the subject of their complement) through a non-simple T (i.e. a rule for which both sides are c-functions). In outline (i.e. ignoring complications like the presence of ‘to’, and the fact that the relation is dependent on particular predicates such as ‘try’), if the phrase structure of ‘Kim tried to hit Sandy’ is of the form:

(11)



then the ECS description is of the form:

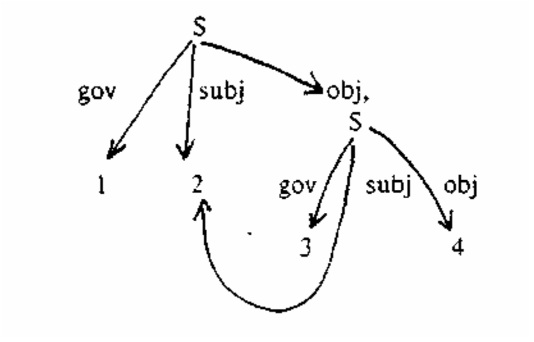
Cs (1, Cvp (2, Cvp (3, 4)))

and the corresponding ERS description is of the form:

Ssubjobj (1, *2,* Csubjobj (3, 2, 4))

i.e. in terms of the inspection tree:

(12)



Intuitively, this is not surprising: the phenomena of control involves more than one construction both at ECS and ERS -- if one is constrained so to refer to no more than one construction on one side of a T rule, then such phenomena will be difficult to handle.

Similar difficulties arise with unbounded dependencies (e.g. wh-movement) which can be reduced to a combination of local dependencies. The difference being that the problem cannot be handled by allowing non-simple T-rules, since they involve constructions which may be arbitrarily far apart (if unbounded dependencies are reduced to local ones then they may involve an arbitrarily large number of constructions.

There are two obvious solutions:

a) Find some way of stating these relations within one level (i.e. by Cs). The obvious way of doing this is to enrich the type theory so as to allow some attributes to take constructions as values: in this way one would be able to 'pass' descriptions of entire constructions around from construction to construction (e.g. for control at ERS make the identity of the subject of the embedded sentence part of the sentence's type description, and ensure that the actual subject of the matrix sentences unifies with this description) this is essentially how such things are handled in GPSG [4] and LFG [3].

b) Allow a relaxation of the spirit of the T-rules to allow that in addition to c-functions, they may contain expressions which *evaluate* to c-functions (i.e. essentially lambda abstractions of c-functions). Roughly, we could allow ECS constructors to translate (ambiguously) as lambda abstractions of ERS constructors:

   Cs (1, Cvp (2)) = = > λ X Csubj-obj (2, 1, X)

   if Ctop is the ECS constructor that combines topicalized elements with sentences, then we could say:

   Ctop(l,2) = = >2(l)

   The translation of *Jules, I like,* would then be the translation of *I like* applied to the translation of *Jules,* i.e. (13) which evaluates by lambda conversion to (14), which in turn evaluates to an ERS interpretation tree. This is essentially how such things are handled in theories whose semantics are inspired by Montague grammar, e.g. [4].

   (13) λ X Csubj-obj (like', I') : Jules'

   (14) Csubj-obj (like', I' Jules')

There is no reason why such a modification of ECS < -- > ERS should be extended to other translators, of course.

There is no doubt that a more sophisticated feature theory is necessary (at least as sophisticated as that of [4], and that the restrictions on T-rules will make certain kinds of translational relationship difficult or impossible to express. But it is not obvious what problems these solutions entail. And there is a sense in which while such constructions can be handled in this kind of approach (i.e. within what is essentially a context free grammar augmented by a feature and certain kinds of filter [the T rules]), they are at the boundaries within which it is 'natural.' The approach is most naturally suited to phenomena which are intuitively hierarchical and recursive in nature, and as such is not well suited for handling non-, or only quasi-hierarchical relations (such as e.g. those whose normal representation requires tree geometry to be augmented with co-indexation devices) -- what is not clear is whether phenomena such as control and unbounded dependencies (and the similar kinds of problem that arise in translating between other languages -- e.g. in transfer) are of this kind.

3.3. IS

3.3.1. Overview of IS

The intuitive basis of IS is that of semantic ('true') dependency: an IS construction is a (complete and coherent) true dependency construction differentiated according to the kind of relation (SR) which holds between each member of the construction and the head.

Roughly, the true members of a construction are those which make an independent contribution to its interpretation: true modifiers are items which semantically specify the head of the construction, true complements are items that satisfy an argument position in the frame of the predicate. True heads are items which possess an argument structure, or which are the focus of true modification. Typical non-true dependents are formal subjects, and (some) subordinating conjunctions and articles, typical non-true heads are aspectual auxiliaries, and strongly governed prepositions (this is all discussed in some detail in [1] p. 155 ff, and 168 ff). Requiring IS constructions to be coherent means that such items will not appear in IS. The atoms of IS are again (readings of) words. Some information about morpho-syntactic properties is preserved (e.g. major syntactic category), other superficial type information is removed (surface case, tense marking e.g.); some semantic properties (Semantic Features, and Time labels) are present.

Briefly, IS represents:

atoms	(i.e. readings of words)
SR	(Semantic Relations, semantic cases, theta roles, etc: e.g. agent, patient, experiencer, etc. see [1] for the proposed list)
Time	(to be precise, a subset of time meanings cf. and [1])
Syntactic Category	(cat, num, per, ...)

This is a proper subset of the standard EUROTRA IS, including some obvious omissions.

As with standard EUROTRA, we allow that there may be differences in type between a construction as a whole and the head of the construction; since we also require that heads be lexical, the intuitive geometry of IS contains lowered govs (an X^1 representation).

For the most part, the motivation for these representational devices is pretty well known: SRs play a crucial role in the translation of prepositions, the relationship between tenses in different languages is notoriously complex, and syntactic category is useful in cases such as English *know* ==> Dutch *weten* or *kennen* depending on the category of the patient, and English *talk* ==> French *parler* or *conversation* depending on whether it is a noun or a verb.

3.3.2. Atoms and Constructors at IS

As with ERS, we will treat the part of an atom which indicates the word of which it is a reading as part of the type, and leave the atom name 'blank' to receive a semantic function:

(_ , { word = like, frame = experiencer/patient, cat=v, ...})

Also as with ERS, the choice of constructors is not entirely obvious, but again, the intuition is that the semantic dependencies which are the basis of IS are relations between constructions and words, where words can be grouped according to their semantic relation frames. Thus, in most cases, the constructors of IS will correspond to semantic relation frames.

Some IS constructors:

```
Cexp/pat=   (_ , {cat=s}) [ (gov, {cat=v, frame = experiencer/patient}),
                            (experiencer, { }),
                            (patient, { }) ]
```

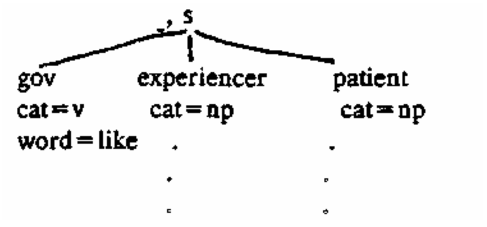
This will assign gov, experiencer, and patient roles to three arguments providing the first is a v with the indicated frame. The resulting construction is of category s.

Cnp/def= (_, {cat=np, definiteness=def}) [(gov, {cat=n})]

This allows definite nps to contain just a nominal gov; a similar C would have to be written for indefinite nps.

An example inspection tree:

(15)



3.3.3. Translation of ERS and IS

Since atoms correspond to words at both ERS and IS, T rules involving atoms will always be atom-atom in the ERS-IS and IS-ERS translators, the only complication being the extra readings of words that will be produced by atoms having SR frames rather than syntactic relation frames.

Given the pragmatic principle that assigns a major burden to the <C,A> descriptions of the levels and their interpretations, it should often be possible to state T-rules for constructors very simply, using the target language G and I to filter the apparent overgeneration of the translator. For example, the rule:

Csubj-obj = = > Cexp/pat

which works in the case of *Jules likes Sandy* appears to overgenerate, since in general subject-object constructions do not translate as experiencer-patient constructions. For example, if the ERS representation of *Jules hit Sandy* is:

Csubj-obj [hit, Jules, Sandy]

the T rule will attempt to produce:

Cexp-pat [hit', Jules', Sandy']

this will fail, however, since *hit* has an agent-patient SR frame, and will fail to unify with the first arg-spec of Cexp-pat.

Since a major difference between ERS and IS is the presence/absence of non-true dependents, one would expect T rules such as the following to be common:

Cdefdet-n (1,2) = = > Cnp/def (1)

3.3.4. Translation of IS and IS

Investigation of IS-IS translation (i.e. transfer) is the central preoccupation of the proto-EUROTRA experiment. This preoccupation is not reflected here, however, since in this framework transfer is not essentially different from any other step in the translation process, and the purpose of exposition is better served by describing relations between more superficial representations, where the phenomena are more familiar.

Transfer components are the most numerous components of MT systems, and their construction is the most demanding in terms of human effort. Accordingly, we would like to simplify the task of constructing the transfer translators as much as possible. Thus, where for most levels we are prepared to relax the requirement that some T-rules must be simple, it will be imposed strictly for transfer.

For transfer, both sides of T rules must be of the form

atom = = > atom
constructor-name = = > constructor-name
constructor-name (variables) = = > constructor-name (variables)

It is clear that this is unrealistic given the current representational theories. Gross structural differences apart, languages lexicalize differently, producing lexical holes and protuberances *(schimmel = = > white horse, kenner = = > someone who knows, graag = = > like to,* etc.). Imposing this restriction is intended to lead to the definition of classes of such cases, and a basis for modifications to the Mu1 level of the framework grounded on contrastive study.

4. References:

[1] Doug Arnold, Lieven Jaspaert, & Lous des Tombe (1985) ELS-3: Eurotra linguistic specifications, Report for Contract ETL-5, DG-XII, CEC Luxembourg.

[2] Boitet, Ch. & Nedobejkine, N. (1980) Russian-French at GETA: Outline of the method and a detailed example. RR 219 GETA, Grenoble.

[3] Bresnan, J. (ed.) The mental representation of grammatical relations, MIT Press, Cambridge, Mass.

[4] Gazdar, G. & Pullum, G. (1982) GPSG: A theoretical synopsis, IULC; Indiana.