[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 PRELIMINARY LINGUISTIC FRAMEWORK FOR EUROTRA, JUNE 1985

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

ABSTRACT

The work described here was the consequence of the idea that we wanted to make a new, more interesting theoretical start in EUROTRA. It is preliminary and not fully developed yet; it should be seen as the reflection of a way of thinking about MT. Currently, we are making it more precise, and experimenting with it. In this paper, we sketch the general outlines of the new EUROTRA framework; some more exemplification can be found in the paper by D. J. Arnold et al. (this volume).

1. Aims and background

It may be worthwhile to summarize the basic principles of the EUROTRA project.

(i) Project started by the EC; this organization has, amongst others, the problem of multilinguality. Large size, some 350 man/years; rather short duration: 5.5 years.

(ii) Seven languages, with possible additions if new members enter the EC (at the moment, Spain and Portugal are entering).

(iii) End result after 5 years: a 'prototype' system, exemplified by a version covering 20,000 lexical units per language, producing reasonable translation, and rather extensible.

(iv) Basic design: transfer system, with at least the sentence as the unit of translation, but probably more. Transfer as simple as possible, analysis and synthesis components constructed by language groups that work as independently of each other as possible.

(v) Same basic software used in all parts of the system.

2. Levels of abstraction for the description of EUROTRA

We will look at the EUROTRA system at four levels:

mu-0:A working translation system, containing actual linguistic 'rules', or whatever they are; constructed conforming to:

mu-l:Some substantial linguistic theory or theories, including some language for the expression of linguistic knowledge at mu-0; this theory should fit into:

mu-2:the abstract theory of the translation system, stating the basic design of the system, and defining the basic software tools. This abstract theory should then be evaluated with respect to:

mu-3:the informally stated basic principles of the project.

In this article, we concentrate on mu-2.

3. Compositionality

Intuitive basis: A translation of some object A is based, in some systematic way, on the translations of the parts of A.

Idea: Describe levels by recursive generative devices such that well-formed objects are atoms or applications of constructors to well-formed objects. Describe translations as simple relations between

elements of different generative devices.

A more formal description: We define the notion of a set of representations by means of a generative device G, which is a pair <C,A> such that

C is a finite set of constructors
A is a finite set of atoms

and define the membership of L(G) as follows:

every a in A is member of L(G)
if c is in C and ul...un are in L(G) then <c, ul...un> is a member of L(G).

(in fact, atoms are constructors with arity 0; however, for intuitive purposes, we prefer to call them atoms).

We call a mapping T between two sets of representations <C1,A1> and <C2,A2> (called source and target, respectively) a strictly compositional mapping if:

(i) T maps A1 into A2

(ii) there is a mapping t from C1 into C2 such that for all u if u = <c, ul...un> then T(u) = <tc, Tul...Tun>

This idea is the basis for the current EUROTRA framework. It is certainly an elegant way of looking at translation. Below, we will see that it is too simple for MT. It is an interesting problem in itself, how one proceeds from a too simple theory towards a more adequate one in a project like EUROTRA. However, in this paper, we will not concentrate on methodological issues.

4. The model for EUROTRA

The translation problem is broken down into a **sequence of primitive translations,** which then should be more 'simple' than the full translation.

We will call a translation **primitive** if it conforms to the 'one-shot principle', which means the following: the translation can be written as a finite collection of rules, where each rule has a lhs and a rhs, such that the lhs defines a class of source representations, and the rhs defines the class of target objects that are the translations of the source objects.

This means that we obtain some number of representation languages, also called 'levels'. These levels are to be defined by generative devices, in some way related to the description given above; and the translations from level to level are to be one-shot and compositional in some way, related to the principle of strict compositionality.

5. Vertical notions (i.e.: notions about the ways in which generative devices define representation languages).

Actually, our generative devices will be somewhat more complicated than what was described above. We think that, for the characterization of representation languages, it is necessary to be able to state conditions on applications of constructors to representations. Therefore, we have a notion of 'application' of constructors.

(i) Rough syntax for generative devices:

feature description ::= *pair**
pair ::= attribute = value | attribute = variable

(where attributes, values, and variables are enumerated in some way).

atom ::= (name, feature description)

(where the set of names is given extensionally. The idea of the 'name' is that it is the basis for some linguistically leading principle for the level concerned).

constructor ::= (name, feature description) [*constructor**]

(ii) Rough semantics:

A generative device G will now generate a language L(G) as follows:

(1) An atom is a member of L(G);

(2) A constructor can be **applied** to members of L(G); here, 'applied' means some version of (parametrized) unification.

At the moment, we have not yet a final idea about how to use unification in our context. A very intuitive description: We call the description of a constructor inside the [ ] an 'argspec', and the members of L(G) that it is applied to, 'constructs'. Now an argspec and a construct are 'compatible' if the names and feature descriptions do not contradict each other. Contradiction between feature descriptions means, for the moment, that they assign different values to the same attribute. Now if an argspec and a construct are compatible, unification produces a new construct which 'merges' the information contained in both. If we identify each description as consistent with some set of things, then we might expect to identify the unification of the descriptions with the intersection of the two sets. If the argspec and the construct are incompatible, then unification fails, and nothing new is produced.

(3) We can put additional requirements on the class of members of L(G), like 'complete' feature descriptions (in some sense to be defined).

(iii) Examples for some hypothetical level of 'syntactic relations':

Atoms:

(_, (word=the, cat=det, num = VARIABLE))
(_, (word=example, cat=n, num = sing))

Constructors:

$C_{det\text{-}n}$ =
(_, (cat=np)) [(gov, (cat=n)),
mod, (cat=det))
]

$C_{sub/ob}$ =
(_, (cat=s)) [(gov, (cat = v, frame = 'sub-np/ob-np')),
(subject, (cat=np)),
(object, (cat = np))
]

Here, the leading linguistic idea, expressed by the 'name' of the constructors and atoms, would be 'syntactic relation'. Application of the Cdet-n to the atoms for 'the' and 'example' yields the following construct:

(-, (cat = np)) [(gov, (word = example, cat = n, num = sing)
(mod, (word = the, cat = det, num = sing)
]

6. Horizontal notions

I.e., notions related to translations between levels.

The objects of translation are source constructs, and the results are target constructs. This applies to whatever pair of adjacent levels. The translations are defined by means of *T-rules,* that describe classes of constructs in terms of atoms, constructors, and variables. T-rules conform to the intuitive idea 'one-shot' in that they define pairs of classes of constructs. They apply recursively, in the way of a compositional translation process.

However, the notion of strict compositionality as defined above is probably too simple for machine translation. Sometimes, it may be desirable to have a translation like:

<c1 .....> => <c'1 ...<c'2...>...>

That is, the shapes of the constructs in terms of the constructors may differ.

A good concrete example of this is the translation between the English

'John likes to swim'

and the Dutch

'John zwemt graag'

The word *graag* is an adverb, meaning 'like-to', like the German *gern.*

It may happen in a case like this that the Dutch linguist finds it more natural to have one constructor 'sentence', applied to NP, verb, Adverb-phrase, and that the English linguist finds it more natural to describe the English sentence by two applications of some 'sentence' constructor.

More generally speaking, it may happen that the linguists constructing generators for adjacent levels do not obtain a relation that is so simple as strict compositionality between the two levels. In a project like Rosetta (Landsbergen, 1984), a rather strict version of compositionality is used, and rule writers (at mu-0 level) of the three languages work in a coordinated way. This approach is not taken in EUROTRA, if only for the reason that a detailed coordination of the work of the language groups is impossible in this case.

Therefore, in EUROTRA we relax compositionally. A current view on compositionality is the following. We still call a mapping between a source level and a target level compositional if the following holds:

(i) T(u) = <f, T(u1) ..... T(un)>, and

(ii) f is definable as a lambda expression in terms of target atoms, target constructors, and variables.

Intuitively, one can look upon this constraint as some sort of structure-preservingness: f can only construct objects that are well-formed target level representations.

We express translations between levels of representation in terms of various types of T-rules.

The most simple case is atom-to-atom. Example:

(fiets, (num = sing)) -> (bicycle, (num = sing))

Meaning: All atoms in the source language with name *fiets* and whose feature descriptions unify with (num = sing) are translated to all atoms in the target language with name = bicycle and whose feature descriptions unify with (num = sing).

Another simple case is constructor-to-constructor. Example:

c27->c112

Meaning: c27 and c112 are names of source and target constructors, respectively. The result is only defined if both constructors have the same number of arguments. For every X_i in the argument list of c27, the target constructor will operate on the translation of X_i. The rule translates all the constructs resulting from applications of c27 to constructs of the source language into tall the constructs of the target language resulting from the application of c112 to the translations of the arguments of c27.

A concrete example of this. Suppose we have, at some 'semantic representations' level, the following constructor:

$C_{ag/pat}$ = (_,(cat=S)) [(gov, (cat=v, frame-agent-patient))
(agent, ( )),
(patient, ( ))
]

A t-rule from 'relational syntax' to 'semantic relations' could then just be:

$C_{sub/ob}$ -> $C_{ag/pat}$

(in fact, this example rule oversimplifies and is empirically wrong).

For more complex cases we have the possibility to introduce complex constructor descriptions (CCD) in T-rules. Syntax:

CCD :: = constructor name [sub-item *]
sub-item ::= CCD | atom | c-variable

(where a c-variable is an integer that stands for some class of source constructs (at lhs) or for their respective translations (at rhs).

Concrete example, from some level of 'configuration syntax' to some level of 'relational syntax':

C_s [1, C_{VP} [2,3]]-> $C_{sub/ob}$ [2,1,3]

We also allow t-rules between atoms and constructors, like in the following case of translation between German and English:

(Schimmel, (...)) -> C_{NP} [(white, (...)), (horse, (....))]

At the moment, we try to find out what the optimal constraints on T-rules are.

7. Why this seems to be a 'good' mu-2

The framework satisfies the original aims of being rigorous (though many details still have to be figured out), and concentrating on relations between levels of representation. It may be a good interface between linguistics and computation too; at least, there are some good feelings about compositionality and unification. However, the latter issue has not been examined yet.

We also expect that the framework will give rise to highly modular systems, which may be important for the rule writer.

Another good thing seems to us that the framework uses well-known and important notions such as compositionality and unification, and in so doing connects EUROTRA more than before to existing (computational) linguistics.

8. Near future

Various linguists in EUROTRA are now trying to use this framework for the description of their languages. An experimental mu-1 description is available. However, the main point of the exercise is to get a first evaluation of the mu-2 description. A rough implementation is in preparation, though, of course, many decisions cannot yet be taken in a principled way.

Some main points for development of the theory seem to be:

(i) a theory of the feature descriptions; Here, in the vertical domain, one may think of a more structured feature world, such that agreement, percolation, co-occurrence restrictions can be expressed in a more satisfactory way.

(ii) linguistic pragmatics; or, how are linguists going to use this framework;

(iii) the relation between mu-1 and mu-0; or, how to express substantive linguistic theories in a more formal way, e.g., by means of universal constructors at the mu-1 level of description;

(iv) the power of T-rules or, what is the optimal version of compositionality for the EUROTRA project.