Working Papers
in
Natural Language Processing

Editors

Frank van Eynde

Katholieke Universiteit Leuven
Maria Theresiastraat 21
3000 Leuven
Belgium

frank@mcvax!prlb2!kulet

Pius ten Hacken

Rijksuniversiteit Utrecht
Trans 10
3512 JK Utrecht
The Netherlands

pius@mcvax!accumv!ruuatw

1
An Introduction to the Eurotra Machine Translation System

Anthony Raw, Frank van Eynde, Pius ten Hacken,
Heleen Hoekstra, Bart Vandecapelle

Abstract

Eurotra is a project for machine translation sponsored and organised by the Commission of the European Communities. It has two aims: the development of a system for machine translation for the nine official EEC languages, and the spreading of expertise, research and education in the fields of machine translation, natural language processing and computational linguistics in the twelve EEC Member States.

This article first explains why the EEC set up this project and goes on to situate the Eurotra effort in a world-wide context. It then concentrates on the two main parts of the system: firstly, the formalism and mechanisms which underlie the software and, secondly, the linguistic specifications. There are two annexes. The first provides a concrete example of a machine-made translation. The second briefly discusses the internal organisation and planning of the project.

1. Machine Translation in the EEC

From its inception, the European Community has adhered to the principle of equal treatment of all Community languages. Initially there were four such languages: Dutch, French, German and Italian. Due to the successive enlargements of the Community this number has grown to nine: the four original ones plus Danish, English, Greek, Portuguese, and Spanish. As a consequence, the number of language pairs has increased from twelve (4 x 3) to seventy-two (9 x 8).

The translation needs within the EEC institutions are hence large and growing, and in order to keep up with the demand the EEC has created the largest translation and interpretation services in the world. Unfortunately, even these can only provide a limited service and due to a lack of resources, many documents are not translated into all EEC languages.

This mismatch between the demand for translation and the affordable resources - both human and financial - could be solved by reducing the number of official EEC languages (to Dutch and Danish, for instance), but this is unacceptable from a political point of view.

An alternative solution could be to enhance the productivity of the translation services by providing them with more powerful tools, such as electronic dictionaries, terminological databases, text formatting devices and, ideally, machine translation (MT) systems. From a technical point of view there have been doubts about the feasibility of these alternatives - especially about the feasibility of machine translation - but since various research and development projects in the USA, Canada and Western Europe had already come up with results in the domain of machine aided translation, the decision was taken to explore this possibility.

The first step was the acquisition in 1976 of the MT system Systran. Systran had been developed in La Jolla (California) for the language pairs Russian to English and English to French. The initial aim of the EEC was to extend the number of language pairs and to enlarge the dictionaries, so that it would become a useful tool for its translation services. In practice these extensions turned out to be far more problematic than foreseen, and this, together with the awareness that within the EEC itself some research centers were developing MT systems of a more advanced design (notably in Grenoble and Saarbrücken), led the Commission in 1978 to the conclusion that the time was ripe for launching a European R&D project in MT. The project was given the name Eurotra, and a group of representatives from some thirty European universities and research centers was formed for its preparation. This group was called the Eurotra Coordination Group.

It took this group and the Commission services four years to get the approval of the European Parliament for an R&D programme for the creation of a prototype translation system for all Community languages. It was further stipulated that the system should be of advanced design, that it should be geared to the translation of official EEC documents within the domain of information technology, and that the dictionaries should contain approximately 20.000 entries.

A second but equally important aim of the Eurotra programme is the encouragement of research and education in the areas of computational linguistics, natural language processing and machine translation within the EEC.

2. The place of Eurotra within Machine Translation

2.1. Design

The first attempts to build a system for machine translation were during the fifties. The design of the early systems was very simple: they identified the words in the source language text and mapped them one-by-one onto their target language equivalents.

It is easy to see, however, that this view of translation is too limited. It suggests that all a human translator does is to take a word, look it up in a bilingual dictionary, write down its translation, take another word, look it up, and so on. In other words, it suggests that the unit of translation is the word, whilst it is in fact at least the sentence. In order to be able to translate the very simple Dutch sentence

(1) De jongen zwemt

into its English equivalent, one has to know that the subject is third person singular, and not second, to avoid the translation

(2) * The boy swim

And in order to know that the subject is third person singular one has to know where the subject is; in other words, quality translation is impossible without a syntactic analysis of the entire sentence. This insight led to the creation of *transfer* systems. Their structure appears as follows :

$$\begin{tikzcd}
\text{source language sentence} \arrow[d, "\textit{morpho-syntactic analysis}"] \\
\text{morpho-syntactic representation of the source language sentence} \arrow[d, "\textit{transfer (lexical and structural)}"] \\
\text{morpho-syntactic representation of the corresponding target language sentence} \arrow[d, "\textit{synthesis}"] \\
\text{target language sentence}
\end{tikzcd}$$

The source language sentence is analysed morphologically and syntactically, and the result of this analysis is stored in a morpho-syntactic representation. The latter is then mapped onto a similar representation of the target language sentence, and finally the target language morpho-syntactic representation is mapped onto a target language sentence. The first step is called *analysis*, the second *transfer*, and the third *synthesis*.

The unit of translation in this approach is no longer the word but the analysed sentence. Different transfer systems take different views on

- the division of labour between transfer and the monolingual components (analysis and synthesis)
- the nature of the morpho-syntactic representations (dependency structures, augmented phrase structures, etc.)
- the internal structure of the analysis and synthesis components (whether or not via intermediate representations)

As for the division of labour between transfer and the monolingual components, it soon turned out that, the more work done by the monolingual components, the less extra effort it takes to handle additional language pairs. Today, many transfer systems are multilingual (as opposed to bilingual) systems, and consequently analyse the source language more thoroughly than the earlier transfer systems did (delivering to the transfer component a deep syntactic or semantic representation).

The next step would be to argue that, if the source language is analysed optimally, then the transfer component can be empty: analysis delivers a representation of the source language sentence which is so abstract as to lack any language specific properties and which can be fed directly into the target language synthesis component. This type of system is called an *interlingual* system.

Eurotra, being a very large MT system (9 languages and 72 language pairs), would seem to be a good candidate for the interlingual approach, for in that case it would only be necessary to devise 9 analysis and 9 synthesis components and it would not be necessary to write an additional 72 transfer components. However, it is not clear what an 'interlingua' should look like, especially for the lexical items.

What, for example, is the interlingual representation of a word like "horse"? The word itself is a language specific expression, and so is any translation of "horse", such as "cheval", "Pferd", "paard", "caballo", etc. Taking any of these as an interlingual representation would mean that one of the existing object languages (English, for instance) is promoted to the status of interlingua, but then one could wonder what the advantage is of mapping a word like "cheval" first onto "horse" and then onto "Pferd" instead of mapping it directly onto "Pferd".

A more plausible alternative would be to take an interlingual representation which is neutral with respect to the different object languages, but then, of course, the question arises of how this special representation language should look.

One thing we can safely state about it is that it should at least have the same expressive power as the object languages: it should, for example, not have one and the same representation for "horse" and "donkey", for in that case one could not prevent the translation of "horse" into "Esel".

In fact, it is easy to demonstrate that the representation language should have more expressive power than any of the object languages. It should, for instance, distinguish at least two kinds of "Pferd" : the warm-blooded quadruple of the equine family, for which English uses the word "horse", and the chess piece which makes L-type movements and for which English uses the word "knight". Hence, if we want to avoid "Pferd" always being translated as "horse" (also when it denotes a chess-piece), then we have to assign at least two interlingual representations to "Pferd". In general, the interlingual representations of the German words will have to be more specific and more differentiated than the German words themselves.

Notice, however, that the differentiation which is required for translating "Pferd" into English is not needed when one translates "Pferd" into Dutch, for the Dutch word "paard" is ambiguous in exactly the same way as the German word. This shows that the amount and the nature of the extra differentiation which one needs on the level of the interlingua is dependent on the nature of the target language. But, if the interlingual representations are target language dependent, then the analysis component, which assigns those interlingual representations to the source language texts, will also be target language dependent, as it will have to anticipate exactly those ambiguities which have to be solved with respect to that one particular target language.

It seems to follow, then, that an interlingual approach to translation is only possible at the cost of making the analysis component target language dependent.

It could be replied, of course, that an interlingua can also be developed in a way which would make it less target language dependent. It could, for instance, be argued that the interlingual representations should be language independent representations, i.e. representations with such a high degree of differentiation that they anticipate ambiguities with respect to any object language. This, however, is more than utopian, especially if one wants to develop it for the whole vocabulary of a language, since the enterprise of completely analysing the vocabulary of a language in terms of universal semantic primitives is a project which meets so many fundamental difficulties that it has never got off the ground, neither in theoretical nor in applied linguistics.[1] In short, there is little hope that a genuine semantic interlingua for the whole of a language can be developed, and the alternative, which would be a target language dependent interlingua, is not very attractive if one wants to extend the system to other target languages.

Since this kind of extensibility is very important for Eurotra, though, it appears that an interlingual approach is less felicitous. This is why Eurotra has chosen the transfer approach and adheres to the following general principles:

- transfer should be as simple as possible, preferably limited to the replacement of lexical units (the notion of *simple transfer*)
- analysis and synthesis are strictly monolingual, i.e. they are not devised with one or more target/source languages in mind
- the representations that are delivered by analysis and transfer, and that are input to transfer and synthesis, are called *interface structures*; they take the form of dependency structures, enriched with semantic information
- the mapping of sentences onto interface structures (and vice-versa) is not one-shot, but is performed via a number of intermediate representations (the principle of *stratification*)

1. One of the most vigorous advocates of this approach has been Jerrold Katz, especially in the late sixties and the early seventies, but from his more recent publications it appears that now even he has abandoned it.

2.2. Implementation

Apart from the basic design of the translation process, MT systems can also be classified with respect to the programming language used to implement them. When the first MT systems started most of the available programming languages, like Fortran, Cobol, Algol, etc. were geared towards mathematical applications, and not very adequate for the formulation of linguistic rules and translation operations (the only exception was Lisp). Also since the hardware was very slow, even Assembler language had a certain popularity. In the early 1970's the development of more adequate programming languages like Pascal and Prolog, together with the progress in hardware, brought about a considerable improvement from the perspective of natural language processing.

Prolog is especially popular at the moment, because it is simple to define special purpose tools within it. Such a tool would be a formalism geared towards a specific task, e.g. coding dictionary or grammar rules. The use of special purpose tools has obvious advantages over using general purpose programming languages, because frequently used concepts can be formulated in a simple way, however difficult they are in the general purpose language. In this way it is much more user-friendly for the linguists and lexicographers, who have to formalize and code their grammars and dictionaries.

Eurotra has also chosen to design a special purpose language, written in Prolog. Section 3 describes this 'Eurotra framework'. Some notions, like unification and proving mechanisms, remind one of the Prolog background, but rule writing in the Eurotra framework is much simpler than writing rules directly in Prolog.

3. Formalism and Mechanisms

The Eurotra framework is a model of translation which provides a linguist with the concepts and tools with which he is able to describe (for all official languages of the EEC):

- the analysis of a language from surface text to some abstract representation termed an interface structure
- the transfer of information between the interface structures of different languages
- the synthesis of a language from an interface structure to surface text

That is, the 'analysis/transfer/synthesis' model of translation.

The concepts which the framework provides comprise the linguistic specifications of Eurotra which are the topic of the next section. The tools provided by the framework to express these linguistic specifications will be discussed in this section.

The two most interesting 'meta-components' of this set of tools are:

- a formalism in which a linguist is able to describe a language
- a mechanism for applying these rules to text and to abstract representations of text

We call the first of these the ***user-language*** (where the 'user' is a linguist attempting to describe a language), and the second the ***virtual machine***.

Naturally there are other components in the set of tools, both internal and external to the Eurotra framework, which, although not topics of this article, should be mentioned. Internal to the framework is a compiler to translate rules written in the user-language into programs in the language of the virtual machine. External to the framework there is a compiler to translate virtual machine programs into the language of a machine, i.e. machine code. Lastly, of course, there is the machine itself.

Since the mapping of text onto interface structure is performed via a number of intermediate representations, the description that a linguist makes of a language is actually a series of descriptions of different abstract levels. It is therefore not only necessary to describe an abstract representation but also to define the relationship between two adjacent representations. We call these abstract levels ***representation levels***. The descriptions that define representation levels are called ***generators***. The relations between adjacent representation levels are defined in terms of the relations between the generators of those levels and are called ***translators***. The remainder of this section will examine these three concepts in some detail with special regard to the formalism and mechanisms of the framework.

3.1. Representation Levels

A main principle of the Eurotra framework is that both the analysis of text to an interface structure and the synthesis of text from an interface structure are performed as a series of steps between intermediate levels of representation. That is, in analysis, for example:

$$Text \Rightarrow i_1 \Rightarrow i_2 \Rightarrow i_3 \Rightarrow \dots \Rightarrow i_n$$

where i designates intermediate representation levels and n designates the interface structure.

The purpose of this stratification is that, because we want the gaps between interface structures of different languages to be as small as possible (the notion of 'simple transfer'), the gaps between text and interface structures become quite large. Thus it becomes an extremely complex task to relate text to interface structure in a single operation. By decomposing analysis and synthesis into a series of primitive translations between intermediate levels of representation the task becomes much more manageable.[2]

This model can be extended further because the transfer component between the interface structures of different languages can also be considered as a translation between two representation levels. Thus, the whole translation process can be illustrated as:

$$\begin{tikzcd}
Text^s \arrow[r, Rightarrow] & i^s_1 \arrow[r, Rightarrow] & i^s_2 \arrow[r, Rightarrow] & i^s_3 \arrow[r, Rightarrow] & \dots \arrow[r, Rightarrow] & i^s_n \arrow[d, Rightarrow] \\
Text^t & i^t_1 \arrow[l, Rightarrow] & i^t_2 \arrow[l, Rightarrow] & i^t_3 \arrow[l, Rightarrow] & \dots \arrow[l, Rightarrow] & i^t_n \arrow[l, Rightarrow]
\end{tikzcd}$$

where s designates the source language and t the target language.

In section 4 we will see what linguistic information is actually represented by these intermediate levels. In section 3.2 we will examine how generators define these representation levels and in section 3.3 how translators define the relations between adjacent levels.

First, however, let us consider the notion that each representation level is a formal language. Such a language is a set of objects which are either

- simple objects
- structures built from simple objects

In the Eurotra framework, simple objects are termed *feature bundles* which consist of sets of *features*, each feature in turn being built out of an *attribute* and a *value* where the value of the attribute may be a variable. For example:

lex is an attribute
eat is a value
lex=eat is a feature
{lex=eat,lu=eat,cat=v,nb=X,tense=pres} is a feature bundle

The set of legal simple objects which may exist within a level of representation is defined by the *feature theory* of that level. A feature theory of a representation level consists of:

- a set of feature declarations which defines which attribute=value pairs constitute legal features
- a set of co-occurrence restrictions which defines which sets of features constitute well-formed feature bundles

Structured objects are trees of feature bundles and can be represented, for example as:

2. Note that from now on, unless explicitly stated otherwise, when we refer to translation we mean translation between levels of representation and not direct text-to-text translation.

Filter rules can modify neither the structure of objects nor the contents of nodes within objects. They are used to check well-formedness, and any object that is deemed ill-formed is deleted. The main purpose of filter rules is to filter out any exceptional objects created by possible over-generalisations in structure building rules and in feature rules. There are two types of filter rules: strict and killer. Strict filter rules, like feature rules, contain condition and action parts. If the structure of a rule matches the structure of an object and the condition part is satisfied then the rule is applicable. Subsequently the action part must also be satisfied for the object to survive - otherwise it is considered ill-formed and is deleted. Killer filter rules contain condition parts only - the action is always deletion of the object. If the structure of a rule matches the structure of an object and the condition part is satisfied then the object is considered ill- formed and is deleted.

3.2.3. Example of Generator Rule Application

As a very simple example of the application of generator rules to objects consider the following unconsolidated object:

{cat=s} < {lex=herons} {lex=eat} {lex=fish} >

which is the hypothesis that the node *{cat=s}* dominates the nodes between angled brackets and that these nodes have the correct precedence relation. That is, 'Is the string "herons eat fish" a sentence and, if so, what are its characteristics ?'. The following 'trace' shows how this object is consolidated via parsing by unification. (Feature declarations and co-occurrence restrictions are omitted for the sake of brevity).

The generator receives this object bottom-up, so it first tries to consolidate the leaves of the object. This takes the form of applying lexical feature rules to each node (i.e. 'dictionary lookup') and may well result in a series of consolidated nodes of the form:

{lex=herons,lu=heron,cat=n,nb=plur}
{lex=eat,lu=eat,cat=v,nb=X,tense=pres}
{lex=fish,lu=fish,cat=v,nb=X,tense=pres}
{lex=fish,lu=fish,cat=n,nb=X}

Note that some of the values for *nb* are uninstantiated variables as their values are ambiguous. Also note that the consolidation of "fish" has resulted in two possibilities.

Although the feature bundles themselves can now be considered consolidated, the relations between each of the nodes and their mother are still weak. This structural consolidation of dominance and precedence is the next step. Assume a structure building rule of the form:

{cat=np,nb=X}
[{cat=n,nb=X}]

which states that the feature bundle description *{cat=np,nb=X}* immediately dominates the single feature bundle description *{cat=n,nb=X}*. The consolidated leaves are processed from left to right, and this rule unifies with the first of them resulting in the structure:

{cat=np,nb=plur}
{lex=herons,lu=heron,cat=n,nb=plur}

Note that the variable for *nb* has become instantiated by unification and has percolated up to the mother node.

Now assume a structure building rule of the form:

```
{cat=vp,nb=X}
   [ {cat=v,nb=X}
     {cat=np} ]
```

The second of the consolidated leaves unifies with the first argument in the body of the rule resulting in the structure:

```
{cat=vp,nb=X}
   {lex=eat,lu=eat,cat=v,nb=X,tense=pres}
   {cat=np}
```

This time the value for nb is not instantiated as it is still ambiguous. We are now looking for an object to unify with the second argument of this rule (i.e. *{cat=np}*). The first of our rules unifies with only one of our readings for "fish", resulting in the structure:

```
{cat=np,nb=X}
   {lex=fish,lu=fish,cat=n,nb=X}
```

which then completes the former rule resulting in the structure:

```
{cat=vp,nb=X1}
   {lex=eat,lu=eat,cat=v,nb=X1}
   {cat=np,nb=X2}
      {lex=fish,lu=fish,cat=n,nb=X2}
```

Note that no relationship is stated between the values of *nb* for some parts of the rule, between, for example, *{cat=vp,nb=X1}* and *{cat=np,nb=X2}*.

Finally, assume a structure building rule of the form:

```
{cat=s}
   [ {cat=np,nb=X}
     {cat=vp,nb=X} ]
```

This rule unifies with the two structures we have created so far resulting in the final structure:

```
{cat=s}
   {cat=np,nb=plur}
      {lex=herons,lu=heron,cat=n,nb=plur}
   {cat=vp,nb=plur}
      {lex=eat,lu=eat,cat=v,nb=plur,tense=pres}
      {cat=np,nb=X}
         {lex=fish,lu=fish,cat=n,nb=X}
```

Note that, wherever possible, percolation of values has occurred and that the number agreement restriction specified in the body of the *{cat=s}* rule has been satisfied.

The resulting structure is a fully consolidated object and the output from our simple generator. All feature bundle nodes have been proven as being well-formed and the structure of the input object has been modified to produce consolidated dominance and precedence relations. That is, we have proved the initial hypothesis that *{cat=s}* dominates the string of leaves and have produced a fully consolidated representation of the result.[4]

3.3. Translators

3.3.1. Translation

Translators are simple devices performing the minimum amount of tasks and leaving the bulk of the work to the generators. They are "one-shot" devices in that the output of a source generator becomes the input to a target generator without creating any intermediate representations within the translator.

The input to a translator is a single representation which is a fully consolidated object created by its source generator. The translator processes the representation top-down from the root node to the leaves, decomposing the input object into a number of unconsolidated subobjects which are immediately passed on to the target generator. The basic principle of the concept of translators is thus that of *compositionality* - the translation of an object is a function of the translation of its parts.

A translator is defined by three components: a feature theory, a default translation mechanism, and a set of user-defined translator rules.

The feature theory of a translator defines the set of basic units of data over which the translator can operate. For default translation, this is defined as the intersection of the feature theories of the source and target generators, i.e. the feature declarations and co-occurrence restrictions that exist at both levels. For user-defined translator rules it is defined as the union of the feature theories of the source and target generators.

Built in to the system is a mechanism for the default translation of objects from source to target generators. The mechanism is intended to simplify the job of rule-writing by linguists by providing a default translation for default cases. The mechanism will fire unless overridden by explicit user-defined translator rules. The structure of objects is translated by copying consolidated dominance and precedence relations between feature bundle nodes at the source level into unconsolidated relations at the target level. That is, the relations are maintained but 'weakened' so that they are subject to possible modification by the target generator. Similarly, the features contained in the nodes of objects are translated by copying consolidated feature bundle nodes at the source level into unconsolidated nodes at the target level (provided that those features are part of the feature declarations of the target level).

4. Note that the example and the rules are oversimplified. To describe even a small fragment of English requires a much larger set of more complex rules. The example should suffice, however, to give some insight into the workings of unification and bottom-up parsing.

3.3.2. Translator Rules

A user can define two types of translator rules: *structure translator* rules and *feature translator* rules.

Structure translator rules, if applicable, override structure translation by the default mechanism. The rules define the translation of consolidated dominance and precedence relations between feature bundle nodes of objects from the source level into unconsolidated relations at the target level.

The rules contain three elements: a left hand side (lhs), the translation operator (⇒), and a right hand side (rhs). The basic shape of the lhs of a structure translator rule is of the form:

$$A{:}fbd\ [\ B{:}arg1\ ,\ C{:}arg2\ ,\ \dots\ ,\ N{:}argn\ ]$$

where the capitals are indices, the head of the rule is a feature bundle description and each argument in the body of the rule is either a feature bundle description or itself a head with its own arguments (recursive).

The rhs of a structure translator rules is of the form:

$$A < B\ ,\ C\ ,\ \dots\ ,\ N >$$

where the capitals are the lhs indices specifying the new soft dominance and precedence relations between nodes. The lhs is a pattern to be matched against a source level object and the rhs specifies what unconsolidated object should be created on the basis of a translation of the lhs.

Structure translator rules perform several operations by altering the position of indices on the rhs of rules. These operations may have the effects of:

- modifying dominance relations between nodes
- modifying precedence relations between nodes
- removing the precedence relation between nodes (i.e. introducing an unordered set)
- deleting nodes
- inserting nodes

Some semi-formal examples of the above:

$$A{:}fbd\ [\ B{:}fbd_1\ ,\ C{:}fbd_2\ [\ D{:}fbd_3\ ]\ ]\ \Rightarrow\ A < B\ ,\ C\ ,\ D >$$

$$A{:}fbd\ [\ B{:}fbd_1\ ,\ C{:}fbd_2\ ,\ D{:}fbd_3\ ]\ \Rightarrow\ A < C\ ,\ B\ ,\ D >$$

$$A{:}fbd\ [\ B{:}fbd_1\ ,\ C{:}fbd_2\ ,\ D{:}fbd_3\ ]\ \Rightarrow\ A < B\ ,\ (\ C\ ,\ D\ ) >$$

$$A{:}fbd\ [\ B{:}fbd_1\ ,\ \sim{:}fbd_2\ ,\ D{:}fbd_3\ ]\ \Rightarrow\ A < B\ ,\ D >$$

$$A{:}fbd\ [\ B{:}fbd_1\ ,\ C{:}fbd_2\ ]\ \Rightarrow\ A < B\ ,\ fbd_3\ ,\ C >$$

Note the use of the set operator (parentheses) in the rhs of the third example, the deletion marker (~) in the lhs of the fourth example, and the insertion of a new feature bundle description (fbd_3) in the final example.

Feature translator rules, if applicable, override feature translation by the default mechanism. The rules define the translation of consolidated features contained in feature bundle nodes from the source level into unconsolidated features at the target level. The rules have the same basic form as structure translator rules. That is, a lhs, the ⇒ operator, and a rhs:

$$fbd\,[\,arg_1\,,arg_2\,,\ldots\,,arg_n\,] \Rightarrow fbd'$$

The body of the lhs of a rule is usually empty. The rhs of the rule (fbd') is a translation of the feature bundle node matched by the head of the lhs of the rule (*fbd*). If structure is stated in the body of the lhs of the rule (as in the above semi-formal example) then it serves only as a context for the application of the rule and none of its arguments are actually affected. That is, the lhs of the rule is a pattern to be matched against a source level object and the rhs specifies what unconsolidated object should be created on the basis of a translation of the head of the lhs.

The type of operations that feature translator rules perform are changing the value of features, introducing new features and deleting features, all in the context of the pattern specified by the lhs of the rule. Feature translator rules are also able to state generalisations regarding feature translation that structure translator rules are unable to make.

3.3.3. Example of Translator Rule Application

As a simple example of the translation process, we can examine how the object created by the simple generator in section 3.2.3. might be translated to the next level of representation. Assume the structure translator rule:

```
      A:{cat=s}
         [ B:{cat=np}
          ~:{cat=vp}
             [ C:{cat=v}
               D:{cat=np} ] ]
⇒            A < C:{frame=subj_obj} , B , D >
```

which will match with our consolidated object from the previous example and create a rhs with altered dominance and precedence relations (the *{cat=vp}* node is deleted and the *{cat=v}* node is 'raised', moved, and given some extra information regarding the *frame* that it expects). Features will be translated by the default mechanism, with the result that the unconsolidated object created by the translator for input to the target generator will look something like:

```
{cat=s}
    {lu=eat,cat=v,nb=plur,tense=pres,frame=subj_obj}
    {cat=np,nb=plur}
        {lu=heron,cat=n,nb=plur}
    {cat=np,nb=X}
        {lu=fish,cat=n,nb=X}
```

Note that some information (in this case the feature *lex*) is no longer present, as this information is not considered to be relevant for higher levels of representation. (The transfer of a feature is blocked if the feature is not declared at the next level).

The target generator will then have the task of consolidating this object by unification with a new set of generator rules which define the linguistic specifications for the next level of representation (the introduction of the *frame* feature suggests that the next level will be concerned with syntactic dependency, i.e. finding the *subject* and *object* of the governing verb).

3.4. Software Implementation

The Eurotra Translation System (ETS) is still very much in a prototypical stage and will remain so until "industrialisation" of the software during the third phase of the project. Thus the project has seen, and will continue to see, quite a number of different software implementations over the years.

The following is a description of the current software, which includes an implementation of the user language and virtual machine as described in the previous sections and a supporting software environment.

3.4.1 Implementation of the Virtual Machine

The generator and translator components, i.e. the "core" of the system, are written in the programming language Prolog. Despite the relative inefficiency of the language in terms of time and space, it does offer several advantages.

- the virtual machine is a unification-based machine and Prolog's built in mechanism of unification offers an ideal environment
- the virtual machine is a non-deterministic machine (it always computes all possible alternatives) and Prolog's built in backtracking mechanism is a good means for achieving this sort of non-determinism
- due to the ease of program development (rapid prototyping) within Prolog, modifications to the Eurotra framework can be quickly implemented

It is envisaged that the core of the prototype will continue to be written in Prolog for the foreseeable future, although there may well be a transition from the currently used C-Prolog to a faster compiled Prolog known as Yap whose developers are working in close collaboration with the software team in Eurotra.

3.4.2. Software Environment

Surrounding the core, but still written in Prolog, are a number of tools to aid linguists in writing correct generator and translator rules. These include:

- a debugging mechanism to trace the application of rules and their effects
- a pretty-printer to display objects in various formats
- a command interpreter to manipulate objects

Rules are written in a formalism (i.e. the user language) which is not the language of the virtual machine (i.e. Prolog). These rules are translated from the formalism into Prolog clauses by a rule compiler written in Yacc prior to translation for improved time and space efficiency during translation.

ETS also contains an interface to the Unify relational database system where a large number of dictionary items for each representation level of a language can be entered, stored and updated. A direct interface between the Prolog core and Unify is currently under development.

Finally, the top-level interface between the user and ETS is in the form of a menu-driven interface which gives the user access to all components of the system plus access to external tools such as editors and the operating system UNIX[5] on top of which the entire system resides.

5. UNIX is a trademark of AT&T Bell Laboratories

4. Linguistic Specifications

In the preceding section the definitions and the formal properties of generators and translators were introduced. This section focuses on the linguistic contents of the representation levels. There are two key ideas to be discussed here before going into detail with regard to each representation level separately.

Firstly, the representation languages must be linguistic in nature as the translation relation is fundamentally a relation between linguistic objects. Consequently, the intermediate levels cannot be completely neutral with regard to different natural languages, in the way a real 'interlingua' would be.

Secondly, there is the observation that in the adopted transfer-model the gap between text and interface structure is quite large. So, it seems impossible to relate text to interface structure (and vice versa) in one go. For this reason the stratification idea is introduced: the analysis and synthesis steps are decomposed into a sequence of primitive mappings between a number of intermediate levels of representation.[6]

The current 'standard' hypothesis is that there are three intermediate representation levels between text and interface structure (IS) for each language:

$$\begin{tikzcd}
Text^s \arrow[r, Rightarrow] & EMS^s \arrow[r, Rightarrow] & ECS^s \arrow[r, Rightarrow] & ERS^s \arrow[r, Rightarrow] & IS^s \arrow[d, Rightarrow] \\
Text^t & EMS^t \arrow[l, Rightarrow] & ECS^t \arrow[l, Rightarrow] & ERS^t \arrow[l, Rightarrow] & IS^t \arrow[l, Rightarrow]
\end{tikzcd}$$

4.1. The Frontend

The frontend consists of three levels: ETS (Eurotra text structure), ENT (Eurotra normalised text) and EMS (Eurotra morphological structure). The first two are not very linguistic in nature. They map text onto a uniform machine readable representation where characters are normalised, etc. This comes down to some variant of the ASCII character set. These levels make it possible for the linguist to abstract away from typographical and typesetting information in the actual text, so that he can assume a standard text format as input for analysis.

The idea of the EMS generator is that it builds representations of the morpho-syntactic structure of word-forms by means of general morphological rules. Inflection is handled at this level by means of the featurisation of inflection endings. The example below shows how the lexical form (i.e. actual word appearing in the input text) of a word is mapped onto its corresponding lexical unit (stem-form), and what kind of features are added to the lexical unit:

6. It is also possible that this strategy can play a role in making the system robust. Robustness means that if at some point in the translation process the system fails (due to ungrammatical input, or simply because of a real system deficiency), it can still produce some (probably incorrect) translation on the basis of objects created on lower levels.

{lex=plays} ⇒ *{lu=play,cat=n,nb=plur}*

{lex=plays} ⇒ *{lu=play,cat=v,nb=sing,pers=3}*

It is clear from the example that one lexical form can receive more than one morphological interpretation.

Derivation and compounding are focuses of current research.

4.2. ECS (Eurotra configurational structure)

ECS is a level of phrase structure closely related to the level of c-structure in Lexical-Functional Grammar. An ECS representation is a labelled bracketing which maintains the linear order of the string and indicates its surface constituent structure. ECS therefore does not contain empty elements nor coindexing (unlike Generalised Phrase Structure Grammar or s-structures in Government and Binding) and it has no notion of head or governor (unlike Dependency Grammar).

One of the important aspects of writing an ECS grammar is to determine the correct set of categories for a specific language. In addition to the traditional categories (noun, adverb, verb, etc.) other lexical categories are included: coordinator, quantifier, complementiser, etc. Also phrasal categories (e.g. noun phrase, prepositional phrase, adverbial phrase) are needed as the function of ECS is to group sequences of words together into phrases which will form constituents at the next level.

Note that the specifications for ECS do not put any severe restrictions on the category set to be used; much scope for language-specific divergences is left here, e.g. the notion of VP (verb phrase) seems suitable for certain languages, whilst others can do perfectly well without it. This is entirely appropriate, since ECS is presumably the level where Eurotra languages differ most.

Apart from defining categories and constituents, the ECS level contains generalisations about surface word order (e.g. verb second in German and Dutch) and about certain types of agreement.

As an example one of the structure building rules for NP's in Dutch is shown. Notice that it also takes care of agreement inside the NP:

```
{cat=np,nb=N,gender=G,ncase=Y,ntype=T}
    [ ^ {cat=detp,nb=N,gender=G,msdefs=D}
      * {cat=ap,nb=N,gender=G,msdefs=D}
        {cat=n,nb=N,gender=G,ncase=Y,ntype=T}
      * {cat=pp} ]
```

This rule says that an NP consists of a noun (*n*) which can be preceded (optionality marker) by a determiner phrase (*detp*) and any number (Kleene star marker) of adjectival phrases (*ap*), and which can be followed by any number of prepositional phrases (*pp*). The features stated on the daughter nodes require certain attributes to have the same value (variable sharing). If these conditions are met, a linguistic object at ECS can be created.

The rule will accept NP's such as:

de rode auto
een huis in de stad

but not:

* het man

as the gender information of the determiner and the noun is contradictory.

The rules and structures exemplified in section 3.2.3 also belong to ECS.

4.3. The Deeper Levels

The next two levels are ERS (Eurotra Relational Structure) and IS (Interface Structure). Both are lowered-governor dependency grammars, and are much like the f-structure of LFG. They differ from ECS in the following respects:

- the leading notion is syntactic dependency
- the order of the constituents is canonicalised
- certain ECS nodes have no equivalent at the deeper levels, e.g. articles, auxiliaries, verb particles, argument bound prepositions, etc. and so their meaning is expressed by a feature on another (usually dominating) node - this mechanism is called elevation[7]

There are two classes of dependents:

- those filling a slot in the frame of their governor (valency bound), called complements or arguments:

  John hits Bill
  subj gov obj

- those not filling a slot because they are not required by a certain governor, called modifiers:

  Bill hit Tom in the kitchen
  modifier

Both ERS and IS representations obey the completeness and coherence constraint. A representation is complete if and only if all the complements that the frame of its governor requires are present in the representation as the governor's dependents; it is coherent if and only if it contains no more complements than the frame of its governor admits.

7. An example is the auxiliary of the future tense ("will", "zullen", werden", etc.) which is not treated as a separate constituent but rather as a feature on the main verb. In this way, the representation of "he will come" is more similar to that of "il viendra" and this makes transfer between English and French considerably simpler.

4.3.1. ERS (Eurotra Relational Structure)

The leading notion of ERS is surface syntactic dependency, which means that the slots of a frame represent surface syntactic functions (subject, object, etc.). (For an example of an unconsolidated ERS structure, see section 3.3.3.)

A typical example of a phenomenon treated at ERS is subject-verb agreement. As subjects and verbs can appear in more than one surface configuration, it is impossible to treat their agreement at ECS in a simple and unified way. At ERS, however, where syntactic functions are explicit, it is easy to capture the generalisation. The following strict filter rule takes care of subject-verb agreement:

```
{/cat=s}
    [ {nb=N,pers=P/sf=gov,cat=v}
      {nb=N,pers=P/sf=subj,cat=np}
    *{} ]
```

The slash ('/') separates the condition and the action part of the rule. Everything on the right hand side of the slash defines the context required for the rule to apply, everything on the left hand side defines what should be there in order for the object to be accepted.

Agreement is a relation between the verb and the surface subject:

```
John was hit by his teachers
 sg   sg             plur
```

This is why active sentences and their passive counterparts get different representations at ERS.

Other phenomena that are handled at ERS are control, raising, and long distance dependencies. The problem with these constructions is that they violate the completeness and coherence constraint. Thus, in order to arrive at ERS representations which are complete and coherent, some constituents are moved or inserted.[8]

4.3.2. IS (Interface Structure)

The IS level is the most abstract level in Eurotra. It serves as input to the transfer components. Because of the 'simple transfer' principle, the IS level should be as neutral as possible with regard to the different languages.

For the moment, IS is a rather straightforwardly 'linguistic' representation, with little or no provision for the expression or use of 'real world knowledge'. This is the result of the typically linguistic view of translation taken in Eurotra as opposed to a more cognition-based approach.

8. Eurotra linguists have been experimenting with a special tool (the so-called 'coindexing tool') for the treatment of unbounded dependencies.

The leading notion of IS is deep syntactic dependency, which means that the slots of a frame represent deep syntactic functions (called arg1, arg2, etc.), augmented with semantic features (e.g. human, animate, abstract, collective). These features can play a role in the disambiguation of structures produced at the lower levels. The following Dutch sentence, for example, can be interpreted in two ways at ERS:

<u>Water</u> drinkt <u>de man</u>
subj *obj*
obj *subj*
<u>Water</u> drinks <u>the man</u>

At IS, however, a selectional restriction based on the fact that only animates can drink, rules out the first interpretation.

Note that active sentences and their passive counterparts get the same representation at IS. The following structure building rule, for instance, creates an IS object which consists of a verbal governor whose frame requires an arg1 (deep subject) and an arg2 (deep object). The rule also applies to passive sentences; in that case the first slot is filled by the 'by-phrase' (which is the deep subject), and the second by the surface subject (which is the deep object). The rule further allows for an arbitrary number of modifiers:

```
{cat=s}
    [ {role=gov,cat=v,is_frame=arg1_arg2}
      {role=arg1}
      {role=arg2}
     *{role=mod} ]
```

The rule accepts both the following sentences:

the boy hit the girl yesterday
the girl was hit by the boy yesterday

The deep syntactic basis is enriched with formal semantic analyses of phenomena such as tense and aspect, mood, determination, quantification and negation. Some of these are still under development.

Two further, non-dependency, relations exist at IS:

- transconstructionals: constituents modifying the construction as a whole rather than the governor, such as the adverb "fortunately" in "I fortunately found the exit" [9]
- conjuncts: constituents building a coordinate structure

These relations imply the existence, at IS, of non-lexical governors (in the case of transconstructionals) and governor-less constructions (in the case of conjuncts). They form an example of a well-defined deviation of standard dependency grammar, as a concession to linguistic naturalness.

9. The adverb does not modify the verb "found" but rather the proposition "I found the exit".

5. Topics of Current Research

The system outlined in this paper is far from complete, but it presents an environment for developing and testing solutions to the problems encountered. Some of these problems have been around in machine translation for a long time, others were discovered by Eurotra, because of its multilingual approach involving many new language pairs.

Some of the areas under research with regard to the formalism and mechanisms of the system include:

Improved parsing mechanisms: by the investigation of techniques such as constraint-based parsing, improved filtering and structure-sharing.

Order and unorder: some linear precedence (LP) tool is required to enable the linguist to adequately describe the relatively free word order languages, such as Italian, Dutch and German.

Ambiguity resolution: some "preference mechanism" is required to select the best alternative between competing solutions.

Most of the linguistic problems that have not yet been solved are semantic or related to the lexicon. Some of the areas where research is being done at the moment include:

Determination:

> *Les* hérons mangent du poisson
> Herons eat fish
> *Les* hérons que j'ai vus ...
> *The* herons I have seen ...

Generic noun phrases have a definite article in French, but not in English. How can we determine whether or not the definite article in French has to be mapped onto a definite article in English?

Pronominal reference:

> John has *a dog*. He beats *it* very much.
> John a *un chien*. Il *le* bat beaucoup.

In order to know that 'it' has to be translated as 'le' and not as 'la', for instance, the system has to know that it refers to 'a dog'/ 'un chien'.

Support verbs:

> een overeenkomst *sluiten* (litt.: 'close an agreement')
> *enter / make* an agreement

It would be wrong to translate 'sluiten' as 'enter' or 'make', without referring to a very specific context. The verbs in the above example have very little semantic content. They cannot be translated independently, since the choice of their target language equivalent depends on the noun. On the other hand the constructions behave as independent lexical units, e.g.:

> *enter* an unfair agreement
> *enter* a new one

How can we deal with these semi-idiomatic expressions in the grammars and the dictionaries?

In other areas results have been reached, but extensions are still being elaborated. This is the case with tense and aspect for example. A system has been developed to represent the meaning of tense and aspect in e.g.

he *has lived* here for six years
hij *woont* hier al zes jaar

Rather than translating the present perfect into a simple present, both are reduced to the same label, reflecting the temporal information of the sentence as a whole (cf. van Eynde,1988).

This system is now being extended to the temporal analysis of non-finite clauses and time adverbials.

6. Summing Up

In the above article we have given a brief overview of the Eurotra translation system. Much of the work is still under debate, revision or development (cf. section 5). Rather than engaging in any of those debates we have tried to keep this presentation neutral and short, leaving the "bones of contention" for forthcoming papers.

References

Arnold, D. (1986). *Eurotra : a European Perspective on MT.* In: *Proceedings of the IEEE, vol.74, no.7.*

Clocksin, W.F. and Mellish, C.S. (1987). *Programming in Prolog.* Third Edition, Springer-Verlag.

Damas, L. (forthcoming). *YAP User's Manual.*

Earley, J. (1970). *An Efficient Context Free Parsing Algorithm.* In: *Communications of the ACM, vol.13, no.2.*

EUROTRA-Reference Manual, version 5.0 (1988). Internal Eurotra document, Luxembourg.

Johnson, R., King, M. and des Tombe, L. (1985). *EUROTRA : a multilingual system under development.* In: *Computational Linguistics 11, 2-3.*

Johnson, S.C. (1975). *Yacc : Yet Another Compiler-Compiler.* In: *Computing Science Technical Report no.32.* AT&T Bell Laboratories.

Katz, J. (1972). *Semantic Theory.* Harper, New York.

King, M. and Perschke, S. (1987). *Eurotra.* In: *Machine Translation today,* M. King (ed.), Edinburgh University Press, Edinburgh.

Multilingua 5-3 (1986). Mouton De Gruyter, Amsterdam.

Pereira, F. (1984). *C-Prolog User's Manual version 1.5.* EdCAAD, Edinburgh.

Raw, A., Vandecapelle, B. and Van Eynde, F. (forthcoming). *Eurotra: An Overview.* In: *Interface, June 1989.* Brussels.

Slocum, J. (1985). *A survey of machine translation : its history, current status and future prospects.* In: *Computational Linguistics 11-1.*

UNIFY Relational Data Base Management System : Reference Manual Release 3.2

Van Eynde, F. (1988). *The analysis of tense and aspect in Eurotra.* In: *Proceedings of the 12th International Conference on Computational Linguistics (Coling-88).* Budapest.

Annex 1: An Example

In this last section we present the machine-made translation from English to Dutch for the sentence:

the commission has sent the proposal to the council

This example is meant to clarify the Eurotra translation system and to give a more concrete idea of what the different levels of representation are like. For ease of presentation we have left out wrong representations at lower levels which eventually get filtered out at higher ones. Generator rules and translator rules are not shown as these modules are quite big (say 100 pages for analysis and synthesis together not including the lexicon). We give both a tree representation and an exhaustive labelled bracketing, accompanied by some very short remarks. Note that we go from text immediately to ECS, thus skipping the frontend.

ECS (analysis):

```
                     s
                     |
      _______________|_______________
      np                            vp
      |                             |
  ____|_____             ___________|______________
  det      n           vgrp          np              pp
  the  commission     __|___      ____|____       ____|____
                      v    v      det     n       p       np
                     has  sent    the  proposal   to   ____|___
                                                       det     n
                                                       the  council
```

```
{cat=s}
    {cat=np,ncase=nongen}
        {cat=det,lu=the,lex=the,msdefs=msdef}
        {cat=n,lu=commission,lex=commission,nb=sing,ncase=nogen,cattype=common}
    {cat=vp}
        {cat=vgrp}
            {cat=v,lu=have,lex=has,vform=fiv,finform=tsg,cattype=aux}
            {cat=v,lu=send,lex=sent,vform=past_part,cattype=main}
        {cat=np,ncase=nongen} cat=det,lu=the,lex=the,msdefs=msdef}
            {cat=n,lu=proposal,lex=proposal,nb=sing,ncase=nongen, cattype=common}
        {cat=pp}
            {cat=p,lu=to,lex=to}
            {cat=np,ncase=nongen}
                {cat=det,lu=the,lex=the,msdefs=msdef}
                {cat=n,lu=council,lex=council,nb=sing,ncase=nongen,cattype=proper}
```

Elevation of the auxiliary; deletion of the vp-node; assignment of surface syntactic func (subject, object, etc.) and normalisation of word order.

ERS (analysis):

```
                            s
        ____________________!____________________
   v          np                np            pp
  gov        subj              obj           obl
 send     ______!______     _____!_____    ____!____
             n        det  n          det  p       np
            gov       mod gov         mod gov    objofp
        commission    the proposal    the to    ____!_____
                                                 n       det
                                                gov      mod
                                              council    the
```

{cat=s}
{sf=gov,cat=v,lu=send,msaspect=perf,mstns=pres,ers_frame=subj_obj_obl,
diathesis=act,nb=sing}
{sf=subj,cat=np,nb=sing,ncase=nongen,cattype=common}
{sf=gov,cat=n,lu=commission,nb=sing,ncase=nongen,ers_frame=none,
cattype=com mon}
{sf=mod,cat=det,lu=the,npdiacr=nonpro,msdefs=msdef}
{sf=obj,cat=np,nb=sing,ncase=nongen,cattype=common}
{sf=gov,cat=n,lu=proposal,nb=sing,nmorphol=deverbal,ncase=nongen,
ers_frame=none,cattype=common}
{sf=mod,cat=det,lu=the,npdiacr=nonpro,msdefs=msdef}
{sf=obl,cat=pp,smod=yes,pform=to,pdist=nonadjl}
{sf=gov,cat=p,lu=to}
{sf=objofp,cat=np,,nb=sing,ncase=nongen,cattype=proper}
{sf=gov,cat=n,lu=council,nb=sing,ncase=nongen,ers_frame=none,
cattype=proper}
{sf=mod,cat=det,lu=the,npdiacr=nonpro,msdefs=msdef}

Elevation of determiners and valency bound prepositions; assignment of roles (gov, arg1, etc.).

IS (analysis):

```
                              s
                              |
 -----------------------------|----------------------------
 v                  np                   np                np
  gov               arg1                 arg2              arg3
 send                |                    |                 |
                     n                    n                 n
                    gov                  gov               gov
                 commission            proposal          council
```

```
{cat=s,s_type=main}
   {role=gov,sf=gov,cat=v,lu=send,msaspect=perf,mstns=pres,pform_of_arg3=to,
    is_frame=arg1_arg2_arg3,diathesis=act,nb=sing}
   {role=arg1,sf=subj,cat=np,nb=sing,ncase=nongen,cattype=common, msdefs=msdef}
      {role=gov,sf=gov,cat=n,lu=commission,nb=sing,ncase=nongen,is_frame=none,
       cattype=common}
   {role=arg2,sf=obj,cat=np,nb=sing,ncase=nongen,cattype=common, msdefs=msdef}
      {role=gov,sf=gov,cat=n,lu=proposal,nb=sing,nmorphol=deverbal,
       ncase=nongen,cattype=common}
   {role=arg3,sf=objofp,cat=np,nb=sing,ncase=nongen,cattype=proper,msdefs=msdef}
      {role=gov,sf=gov,cat=n,lu=council,nb=sing,ncase=nongen,is_frame=none,
       cattype=proper}
```

The next step is simple transfer. Only the lexical items in the tree are translated. The interaction of dictionary entries and feature translation rules takes care of correct feature-assignment.

IS (synthesis):

```
                                    s
                                    |
 -----------------------------------|-------------------------
 v                 np                    np                np
gov               arg1                  arg2              arg3
sturen             |                     |                 |
                   n                     n                 n
                  gov                   gov               gov
               commissie              voorstel            raad
```

```
{cat=s,s_type=main,diath=act}
   {role=gov,sf=gov,cat=v,lu=sturen,msaspect=perf,mstns=pres,cs=no,ers_frame=sobobl,
    is_frame=arg1_arg2_arg3,pform_of_arg3=naar,vtype=main,nb=sing}
   {role=arg1,sf=subj,cat=np,ntype=ordn,ncase=norm,elev=yes,gender=nneut,
    msdefs=msdef,nb=sing}
      {role=gov,sf=gov,cat=n,lu=commissie,ntype=ordn,ncase=norm,ers_frame=empty,
       is_frame=empty,gender=nneut, pers=t,nb=sing}
   {role=arg2,sf=obj,cat=np,ntype=ordn,ncase=norm,elev=yes,gender=neut,
    msdefs=msdef,nb=sing}
      {role=gov,sf=gov,cat=n,lu=voorstel,ntype=ordn,ncase=norm,ers_frame=empty,
       is_frame=empty,gender=neut,pers=t,nb=sing}
   {role=arg3,sf=pcomp,cat=np,ntype=ordn,ncase=norm,elev=yes,gender=nneut,
    msdefs=msdef,nb=sing}
      {role=gov,sf=gov,cat=n,lu=raad,ntype=ordn,ncase=norm,ers_frame=empty,
       is_frame=empty,gender=nneut,pers=t,nb=sing}
```

Insertion of determiners and valency bound prepositions; function assignment.

ERS (synthesis):

```
                         s
                         |
 ________________________|_______________________
 gov          subj             obj            obl
  v           np               np             pp
sturen    ____|_____       ____|_____     _____|______
          gov     mod      gov     mod    gov    pcomp
           n      detp      n      detp   prep     np
       commissie   |     voorstel   |     naar   ___|____
                  gov              gov           gov   mod
                  det              det            n    detp
                  de               het           raad   |
                                                       gov
                                                       det
                                                       de
```

{cat=s,s_type=main,diath=act}
    {sf=gov,cat=v,lu=sturen,msaspect=perf,mstns=pres,ers_frame=sobobl,
     pform=naar,vtype=main,nb=sing}
    {sf=subj,cat=np,ntype=ordn,ncase=norm,gender=nneut,nb=sing}
        {sf=gov,cat=n,lu=commissie,ntype=ordn,ncase=norm,ers_frame=empty,
         gender=nneut,nb=sing}
        {sf=mod,cat=detp,dtype=art,elev=yes,gender=nneut, msdefs=msdef,nb=sing}
            {sf=gov,cat=det,lu=de,dtype=art,elev=yes,det=de, gender=nneut,msdefs=msdef}
    {sf=obj,cat=np,ntype=ordn,ncase=norm,gender=neut,nb=sing}
        {sf=gov,cat=n,lu=voorstel,ntype=ordn,ncase=norm,ers_frame=empty,
         gender=neut,pers=t,nb=sing}
        {sf=mod,cat=detp,dtype=art,elev=yes.,gender=neut, msdefs=msdef,nb=sing}
            {sf=gov,cat=det,lu=het,dtype=art,elev=yes,det=het,gender=neut,
             msdefs=msdef,nb=sing}
    {sf=obl,cat=pp,pform=naar}
        {sf=gov,cat=prep,lu=naar,ers_frame=np_compl}
        {sf=pcomp,cat=np,ntype=ordn,ncase=norm, gender=nneut,nb=sing}
            {sf=gov,cat=n,lu=raad,ntype=ordn,ncase=norm, ers_frame=empty,
             gender=nneut, nb=sing}
            {sf=mod,cat=detp,dtype=art,elev=yes,gender=nneut, msdefs=msdef,nb=sing}
                {sf=gov,cat=det,lu=de,dtype=art,elev=yes,det=de,gender=nneut,
                 msdefs=msdef,nb=sing}

Insertion of auxiliary; note that the Dutch grammar does not use a vp-node. Surface word order is determined.

ECS (synthesis):

```
                                 s
                                 |
    _____________________________|_____________________________
    np              v       np              pp                v
_____|_____      heeft  _____|____      ______|_____       gestuurd
detp      n             detp     n      prep     np
 |    commissie           |   voorstel  naar  ____|___
det                     det                   detp    n
de                      het                     |    raad
                                              det
                                              de
```

```
{cat=s,s_type=main}
    {cat=np,ntype=ordn,ncase=norm,gender=nneut,nb=sing}
        {cat=detp,dtype=art,msdefs=msdef,gender=nneut,nb=sing}
            {cat=det,lu=de,lex=de,dtype=art,msdefs=msdef, gender=nneut,nb=sing}
        {cat=n,lu=commissie,lex=commissie,ntype=ordn,ncase=norm, gender=nneut,nb=sing}
    {cat=v,lu=hebben,lex=heeft,vtype=aux,finite=fin,vform=fin,nb=sing}
    {cat=np,ntype=ordn,ncase=norm,gender=neut,nb=sing}
        {cat=detp,dtype=art,msdefs=msdef,gender=neut,nb=sing}
            {cat=det,lex=het,lu=het,dtype=art,msdefs=msdef, gender=neut,nb=sing}
        {cat=n,lu=voorstel,lex=voorstel,ntype=ordn,ncase=norm, gender=neut,nb=sing}
    {cat=pp,preptype=no}
        {cat=prep,lex=naar,lu=naar}
        {cat=np,ntype=ordn,ncase=norm,gender=nneut,nb=sing}
            {cat=detp,dtype=art,msdefs=msdef,gender=nneut,nb=sing}
                {cat=det,lex=de,lu=de,dtype=art,msdefs=msdef, gender=nneut,nb=sing}
            {cat=n,lu=raad,lex=raad,ntype=ordn,ncase=norm, gender=nneut,nb=sing}
    {cat=v,lu=sturen,lex=gestuurd,,vtype=main, finite=nonfin,vform=past_part}
```

From this ECS-representation the Dutch sentence is generated simply by concatenation of the lexical nodes (i.e. the values for lex):

de commissie heeft het voorstel naar de raad gestuurd

Annex 2: Organisation and Planning

One of the characteristic features of the Eurotra project is its highly decentralised organisation. Apart from the general management and coordination, which rest with the Commission of the European Communities - more specifically with the Directorate-General for Telecommunications, Information Industries and Innovation - all tasks are performed by the Eurotra research units in the twelve member states.

There are two reasons for this decentralisation: primo, the lack of a special EEC research centre for natural language processing and, secundo, the explicit wish to spread the expertise over the different member states.

The consequence of this policy is that also tasks of central importance, such as the definition of the linguistic specifications and the development of the core formalism, are carried out by teams whose members are spread all over Europe. At the moment there are three such teams: the linguistic research group, the framework group and the dictionary task force. They provide the tools, the legislation and the guidelines for the nine language groups.

The main task of a language group is to develop modules for the analysis and generation of its own language plus the modules for transfer from the eight other languages. The following is a list of the nine EEC languages with references to the corresponding research unit(s):

Dansk
Københavns Universitet (Danmark)

Deutsch
Institut für Angewandte Informationsforschung (Saarbrücken, BRD)
Institut für Kommunikationsforschung (Bonn, BRD)

Ελληνικα
Eurotra Greece (Athens, Greece)
University of Rethymnon (Creta, Greece)

English
UMIST (Manchester, United Kingdom)
University of Essex (Colchester, United Kingdom)

Español
Eurotra España (Barcelona, España)
Universidad Autonoma de Madrid (España)

Français
Université de Nancy II (France)
Université de Paris VII (France)
Université de Liége (Belgique)

Italiano
Gruppo Dima (Torino, Italia)
Università di Pisa (Italia)

Nederlands
    Rijksuniversiteit Utrecht (Nederland)
    Katholieke Universiteit Leuven (België)

Português
    Universidade de Lisboa (Portugal)

Ireland and Luxembourg, the two remaining member states, have been assigned tasks of a general nature. NIHE (Ireland) monitors the work on terminology and CRETA (Luxembourg) functions as a documentation centre and a software clearing house.

All management decisions are taken by the Liaison Group, which consists of representatives of the different research units (one voting member per member state) and the head of the project. This group meets once per month in Luxembourg. It is mainly responsible for management, coordination and planning.

On a very general level the planning distinguishes three phases: the first phase was used for setting up the research units in the various member states and for making them operational. In the second phase the implementation work started and by June 1988 when this phase finished most language groups had developed analysis and synthesis modules for their languages containing 2500 lexical entries and covering a restricted set of syntactic constructions. Apart from these monolingual modules the language groups also developed modules for transfer into their own language.

The third phase which started in July 1988 and which will finish in 1990 will be used for extending the syntactic and the lexical coverage of the system: from 2500 to 20.000 entries and from a restricted set of syntactic constructions to a much larger set.

By mid 1990 the project will have produced a prototype system which will be further developed on an industrial basis.