Steven KRAUWER & Louis DES TOMBE
Institute for General Linguistics
Utrecht State University
Utrecht, Holland

REFLECTIONS ON TRANSFER IN MACHINE TRANSLATION

Note : We are very grateful to nearly all the participants of the Eurotra project for the stimulating environment they are.

1. Introduction

The research described in this paper was done in the framework of the Eurotra project. However, the ideas presented are not accepted Eurotra doctrine, though there is a distinct possibility that they will exert a certain amount of influence on it.

The paper is relevant to all members of the class of multilingual MT systems of the analysis-transfer-synthesis family, but it has a definite Eurotrian basis, so it may be useful to mention briefly some general aspects of Eurotra that make up the world in which our ideas were born. There are four things to say.

First of all, in the Eurotra project linguistics is pretty independent from computation. In principle the system will be created by linguists on the basis of a linguistics-oriented software. Note that by linguists we mean a rather heterogeneous bunch of people, including translators, lexicographers, bilinguals, syntacticians, and so on.

Second, it is fundamental that the transfer component must be as simple as possible. In Eurotra, the primary reason for this is that there are many transfer components compared to analysis and synthesis components.

Third, the approach is also *linguistic* in the sense that it is based as much as possible on 'linguistic' knowledge as opposed to 'extralinguistic' or 'world' knowledge. Of course, it is known since a very long time that for some cases of translation one needs knowledge that is hardly linguistic (like whether a box can be in certain things called pens). However, on the one hand it is clear that this kind of knowledge is extremely difficult to use in this field; and on the other, it is not at all clear how close one can get to 'acceptable translation' on the basis of linguistic knowledge only.

Lastly, there is a strict separation between *monolingual* (analysis and synthesis) and *bilingual* (transfer) tasks. It is not assumed that any knowledge of any other than the own language enters directly in the construction of analysis and synthesis systems. This idea follows more or less immediately from the fact that Eurotra is a multilingual system : it would be rather confusing if in analysing one's own language one would have to take six other languages into account. So the writer of analysis or synthesis is acting as if unaware of the fact that he was participating in the construction of a translation system.

The general linguistic approach given with these remarks is typical for Eurotra, though other systems may share some or all of these characteristics with it. Given this approach, a number of problems arise. In this paper, we address just one of them : *the nature of the transfer component* (though we will touch on some other issues in passing).

This question is certainly one of the most fundamental ones in a project with the characteristics mentioned ; we will briefly examine why this is so. First, in order to be able to specify the task for the analysis and synthesis writer, you must know what kind of interface representations (the objects passed from analysis to transfer and from transfer to synthesis) you want. But in order to know that, you must know what kinds of things you want transfer to do. It makes rather a difference whether you want to have an interlingual system (where transfer does nothing at all) or a system where transfer is allowed to perform arbitrary transformations on the representations used (where it is not really necessary that analysis and transfer do nothing at all). So the first thing one wants to know is what transfer is supposed to do.

The second reason is that it may turn out not at all simple to design a general strategy for transfer. In a multilingual system, this problem would have to be tackled several times, which is a waste of money no project can afford.

In short, it is important to examine what transfer does, and how it does it. In this paper, we present some ideas about this. However, these ideas are highly speculative and need more research. Moreover, we express them here in a rather informal way. The reason for this is that the only technical version we have was developed in the Eurotra framework, but this is hardly the place to give an overview of that framework. Instead of being technical, we have decided that an informal presentation may be far more informative as to the kinds of problems that arise in a certain project these days, and the kinds of ideas some Eurotrians have about them.

2. Simple transfer

Clearly, acceptable translation involves preservation of something ; let us say that the something is meaning. This is not at all an unproblematic thing to say, by the way ; but that will not be our topic for this paper. So we formulate a first piece of theory :

(1) A theory of *analysis, transfer*, *and synthesis :*
(i)
Lexical meaning is translated *by rules of the transfer* component;
(ii)
0ther meaning *(like thematic relations, tense, aspect*) *is represented* in a universal *way in the interface representation.*

This view is not particularly original ; in fact, it is rather common to think this in the MT world. The standard motivation for it is feasibility. Note that, again, we are touching an interesting problem here: for some kinds of meaning it is not an entirely trivial question whether they are to be considered 'lexical' or not. A good example is modality, which causes confusion in Eurotra for precisely this reason (here we want to thank Douglas Arnold for the excellent part he played in the detection of this problem). We will ignore this problem here.

Given (1), it is clear that transfer has rules like this :

(2) LU (source) $\rightarrow$ LU (target) in the context X

Here, LU means 'lexical unit', whatever that means exactly. The context is necessary to decide in cases of lexical ambiguity in transfer (note that if this kind of ambiguity did not exist, theory (1) would be tanta-mount to the statement that one tries to construct an interlingua). Note the highly informal description of the rule type ; the specification of a rule syntax is another issue that does not concern us here.

In the ideal case, transfer consists entirely of rules like (2). Transfer is considered 'simple' in this case, because the transfer writer is only responsible for the construction of a bilingual dictionary. It may be worthwhile to point out that by simplicity we mean *simplicity for the transfer writer*. There is no way to see how simplicity in the way of machine efficiency could affect transfer more than analysis or synthesis. However, there is more to be said about simplicity of transfer. A complication for the transfer writer occurs if rules can interact; in such cases, he will not just be writing a dictionary, but he will also have to solve problems of strategy. So, in the ideal case, the rules do not interact at all. Note that, if it can be guaranteed that all context specifications in rules of type (2) refer to the source language only, then this ideal is attainable : transfer can then be a 'one-off process.

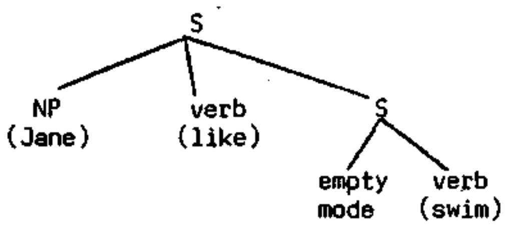
3. Complex transfer

A crucial problem with the idea of simple transfer is that lexical substitution may entail other changes of representation. Let us give an example. The following two sentences are the best possible translations of each other :

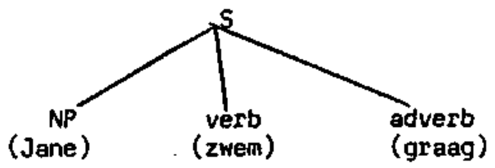
(3)English : Jane likes to swim
(4)Dutch : Jane zwemt graag

The Dutch word 'graag' is an adverb, meaning 'like-to' ; similarly, German has 'gern'.

We now give two representations :the representation that some analysis system for English could assign to (3), and the one that some analysis system for Dutch could assign to (4).

(5)


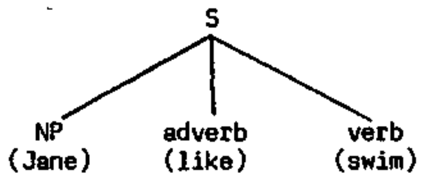
(6)



Actually, these two representations are abstract sketches of the representations that would be used in Eurotra. It is clear that simple transfer, as defined in the preceding section, cannot relate (5) and (6) to each other, as lexical substitution rules cannot change tree geometry. Whether this is or is not a problem for some given translation system, depends on two conditions : the representation theory, and the task set for transfer. Let us examine each of them briefly.

With respect to the representation theory, one can easily imagine a representation theory that does not cause this geometrical difference. All one needs is a theory that analyses the English example as (7) instead of (6) :

(7)



Of course, in order to be useful, a representation theory must have a certain generality ; the representation theory saying that English represents (3) as (7) might easily cause other representations to be problematic in the way of the example. It is not clear at the moment whether there is a representation theory for which it is possible to show that problems like this will never occur, and which is at the same time compatible with the idea of lexical transfer (note that without the latter condition, an interlingua would do the job). In practice, it seems to be the right attitude to be prepared for this sort of problem.

The second condition that renders the example a problem case is the assumption that transfer must relate the 'correct' representation of the source language to the 'correct' representation of the target language (in terms of the grammar of this language). It is entirely reasonable to question this. However, if one does not require that the output from transfer is in some way 'correct' in terms of the target language grammar, then the synthesis writer will be in trouble, especially in a multilingual system, since he will not know what kind of input structures he has to expect. In Eurotra, we support a sophisticated version of this assumption ; we will not pursue the details here.

The conclusion is that given certain reasonable assumptions, lexical transfer causes concomitant non-lexical transfer in some cases. So we will have other kinds of transfer rules as well. We will call these complex transfer rules. We will not examine their nature here, but think of them as arbitrary rewrite rules.

4. A possible transfer strategy

In this section, we show how one could construct a transfer system that incorporates complex transfer rules but that at the same time does not force the transfer writer to develop a complicated strategy. The main problem to be solved is that complex transfer rules change the representation in arbitrary ways, so that the operation of simple transfer rules becomes hard to define. The scheme described works for complex transfer rules as long as they are triggered by lexical units, as in the case of *like - graag.* One may wonder whether other kinds of complex transfer rules exist. We think they will exist in systems like Eurotra, if only as a consequence of insufficient task specifications to the writers of analysis and synthesis.

The view expressed here is incomplete. For example, it ignores complex transfer that is not lexically based. A more elaborate description, that however is formulated entirely in terms of the Eurotra project, can be found in the Eurotra report ETL-1-NLB, Chapter 3, section 3.

The idea behind the strategy described here is that transfer, though it contains arbitrary transformations, will still enable the transfer writer to write rules without having to think about strategy. The strategy described should therefore be built into the system that interprets transfer rules. The strategy assumes that the data structure is a tree, with some arbitrary kind of labeling.

First of all, we give an impression of the strategy as a whole :

(i)
The data structure in transfer consists of 2 trees : a complete source tree and a target tree that is built gradually in the course of the transfer process ; .

(ii)
The source tree is stable throughout transfer ;

(iii)
The nodes of the tree are translated in a systematic, structurally defined way ; to each node a piece of target tree is associated ;

(iv)
As soon as the root node of the source tree is assigned its target associate tree, the main transfer process is finished.

Let us now look more closely at the data structure. Usually, at least in Eurotra, the data structure is looked upon as a single tree. This principle is preserved in transfer. However, as transfer is concerned with 2 languages, there will be 2 trees. We will call them s-tree and t-tree (and will use the prefixes s- and t- generally for source language and target language). Note that we are not suggesting that there will be more than one data structure. Initially, there is only the s-tree. The t-tree is built gradually. In fact, nondeterministic processing may result in several t-trees, we assume that these will be represented in the usual way. The s-tree has a special feature : each of its nodes has a property that is a reference to its associated t-node. This property is of the same kind as other 'binding' properties as used for pronominal reference, for instance. We will call this property _itn_ (for index-of-t-node). Initially, all _itn_ properties have the value _e_, indicating that they must get a real value yet. We will refer to the t-node indicated by _itn_ as _itn^_; this does not mean, however, that _itn_ is meant to be of type 'pointer'; it is not relevant here to think about the data type of _itn_, but, deep in our hearts, we know that it must be of type index. In the rest of this paper, we will refer to properties of t-nodes as 'property of _itn^_ of _A_' (where A is a s-node), and similar.

Given two trees, we obtain a certain degree of stability : no transfer rule will assign to the s-tree, and this rule will be without exception. As a consequence, the s-tree is stable throughout transfer. This has 2 advantages :

(i)
For each transfer rule, the rule writer knows at least what the s-tree is, independently of strategy ;
(ii)
Obviously, (i) does not apply to the t-tree; but the stable s-tree is a basis for a stable strategy in building the t-tree.

We now turn to the order of rule application. In principle, one could take the t-tree as the basis for a transfer strategy. The process would be one of generating a (partial) t-tree, conditioned by the s-tree. A difficult problem with such a strategy is that it *is* difficult to be certain that all of the s-tree has been translated (and exactly once). We have not examined this possibility further.

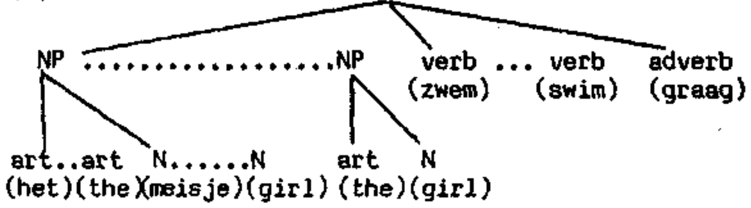
We feel that the transfer strategy should be based on some systematic treatment of the nodes of the s-tree, so that at every moment it is clear which nodes have received a translation (itn<>e), and which have not (itn is e). The best strategy seems to be a 'daughters-first' approach, i.e. a mother mode will not be translated until all of its daughters have been translated. We do not have very strong arguments for this, only two weak ones :

(i)
In this way it looks relatively easy to write rules for lexically based transfer of structure;

(ii)
It can easily be seen when the process has come to a result : this is the case when the root mode of the s-tree has been translated.

As an example, we look at the application of the *graag-like* rule in transfer from Dutch to English. In (8), the dotted lines indicate the relation between s-node and t-node.

(8)



The adverb is not translated by a simple transfer rule; the daughter-first strategy has constructed t-trees for all nodes that satisfy the condition that all their daughters have been translated. On this structure, a complex rule will be applied. A highly informal description of the rule is :

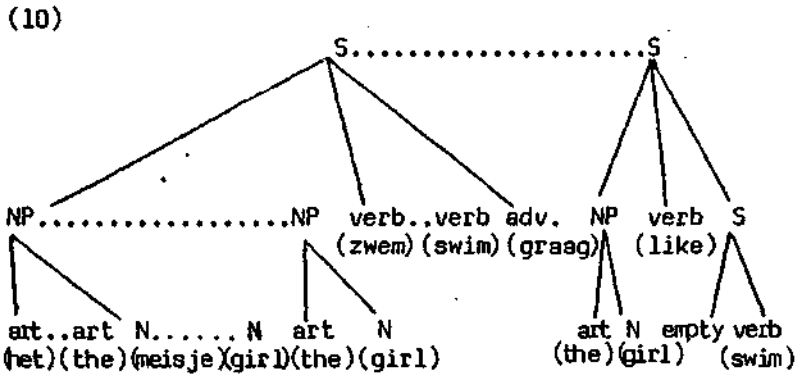
(9) left hand side :

$[_S$subject, adverb (graag), X]

right hand side :
$[_S$itn^ of subject, verb (like), $[_S$empty (bound to subject), itn^ of X]]

The result is :

(10)



5. Conclusion

The proposal given is very tentative. We feel more that we can prove that it is good to have two trees in transfer anyway, because it is so natural. We also feel that the strategy described will allow the transfer writer to use simple and complex transfer rules in combination, without having to invent his own strategy.

A more general conclusion is that we need if not this scheme, then in any case some solution for the problems indicated. Especially in multilingual translation systems with transfer, the number of transfer components is rather large. Problems like the one described should not be left to the responsibility of transfer writers.