Chapter 4

Machine Translation Engines

4.1 Introduction

In Chapter 2, we gave an overview of the environment in which a typical MT system might operate, and outlined the various processes and parts involved. In Chapter 3, we discussed how basic linguistic knowledge can be represented and used for automatic analysis and synthesis. It is now time to look inside the most important non-human component in MT — the component that actually performs automatic translation — what we will call the **translation engine**.

MT engines can be classified by their architecture — the overall processing organisation, or the abstract arrangement of its various processing modules. Traditionally, MT has been based on **direct** or **transformer** architecture engines, and this is still the architecture found in many of the more well-established commercial MT systems. We shall therefore look at this architecture in detail in Section 4.2 before moving on to consider the newer **indirect** or **linguistic knowledge** architectures which, having dominated MT research for several years, are starting to become available in commercial form (Section 4.3).

4.2 Transformer Architectures

The main idea behind transformer engines is that input (source language) sentences can be transformed into output (target language) sentences by carrying out the simplest possible parse, replacing source words with their target language equivalents as specified in a bilingual dictionary, and then roughly re-arranging their order to suit the rules of the target language. The overall arrangement of such an Engine is shown in Figure 4.1.

The first stage of processing involves the parser, which does some preliminary analysis of the source sentence. The result need not be a complete representation of the kind described in Chapter 3, but might just be a list of words with their parts of speech. This is passed to a package of rules which transform the sentence into a target sentence, using

— where necessary — information provided by the parsing process. The transformation rules include bilingual dictionary rules and various rules to re-order words. They may also include rules to change the form of target words, for example, to make sure verbs have the correct person, number, and tense suffixes.

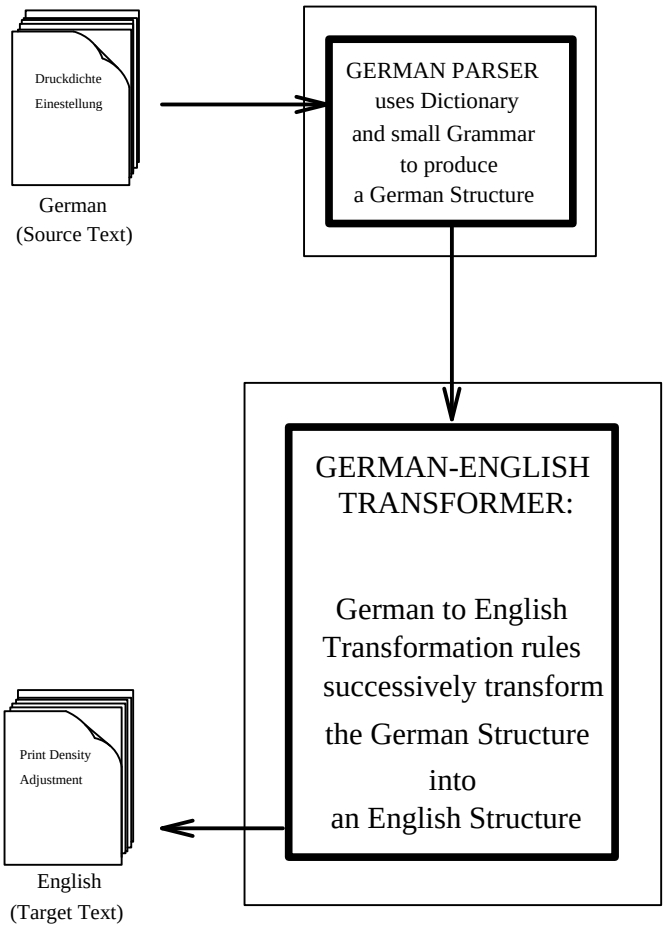


Figure 4.1 A Transformer Architecture (German to English)

To get a more detailed idea of how it works, we shall examine the steps in the translation of a sentence taken from the printer manual text in Chapter 2:

(1) Drehen Sie den Knopf eine Position zurück. 'Turn you the button one position back.' (Turn the button back one position.)

Step 1: The German words are looked up in a German electronic dictionary, and the appropriate category (for example, noun, verb) is assigned. In this particular case the look-up is easy: almost all the words in the sentence are present in their base form — the form they normally have as dictionary entries. The only exceptions to this are the determiners *den* and *eine*, which are inflected forms of *der* and *ein* and have to be recognised as such. After all, an electronic dictionary is likely to be similar to an ordinary paper dictionary

in that regularly inflected forms of verbs, nouns, adjectives and determiners are not given since they can be deduced from general rules. This is why most MT systems make use of a morphological component. This component contains specific rules that deal with the regularities of inflection. Take for example a verb like *drehen* ('turn'), which has the 3rd person singular form *dreht* ('turns'). This form is not shown in monolingual or bilingual paper dictionaries like Düden because other verbs of the same general form have the same form for 3rd person singular. If the input sentence contained *dreht*, the lookup system would first follow its general policy of looking up directly. Assuming that fails, it would then refer to some built-in inflection rules to see if they could be used to derive an infinitive or stem form. One rule might say (in effect) "If the word has *t* on the end, it might be a 3rd person singular verb. Try to confirm the hypothesis by removing the *t*, adding infinitive/imperative *en*, then looking for the resultant *drehen*." A detailed account of the type of rules that we can encounter in a morphological component is described in Chapter 5.

Note that the generalizations of a morphological component can also help the system to deal with words which are not in its dictionary in any form at all. In the past few years, German has acquired the verbs *faxen* and *mailen*, which are derived from English *to fax* and *to (electronically) mail*. Let us suppose they are not in the German dictionary. If *mailt* or *faxt* are encountered in the input, our 3rd person singular rule could apply and, as a result of the verb annotation on the RHS, it would 'guess' that the input forms might be 3rd person singular versions of the hypothesised verb *mailen* or *faxen*. Obviously this hypothesis cannot be confirmed in the available dictionary, but it is certainly useful: the parser can now work on the assumption that the unknown word is probably a verb — this is much more helpful in the parse process than having no idea at all what its category/part of speech might be.

One problem with which the system also has to deal is the fact that the two words *drehen* and *zurück* together form the main verb of the sentence: *zurückdrehen*. The recognition may be done by a rule which specifies that prepositions which stand alone (i.e. without a complement) at the end of a sentence can form part of the main verb. This possibility is then checked in the dictionary, which should contain an entry for the verb *zurückdrehen*.

Step 2: Some rules of a German grammar are used to try to parse the sentence. This parse might result in the assumption that the NP *den Knopf* ('the button') is the object of *zurückdrehen* and (possibly) that the next NP *eine Position* is a modifier of some sort. An advanced parser might work out that it is in fact a *measure* modifier. However, it is quite possible that the transformer Engine will not need any parse at all in this case (beyond identification of the category of the words in the string). This is because the difference between the German and some possible English translations is not great.

Step 3: The Engine now applies some German to English transformation rules. The first step here is to find translations of the German words in a German to English dictionary. Taking the simple cases, *der* — the nominative form of *den* — goes to *the*, *Knopf* goes to *button*, *ein* to *a*, *Position* to *position*. The rules might have the following form:

knopf{cat=n} → *button*{cat=n}

ein{cat=det} → *a*{cat=det}

...

...

and so on. That is, when *knopf* is a noun (cat=n) it is translated as *button*. Similarly, *ein* translates as the determiner *a* — in the present context, *ein* would be best translated as *one*, but let us assume that it is routinely translated as *a* by the Engine.

Turning to *zurückdrehen*, there needs to be a rule which says "If there is an imperative verb X, followed by the NP *Sie*, the translation is the translation of X. In this case, we have an imperative verb (*zurückdrehen*) followed by the NP *Sie*, so we will get *turn back* as the translation. This rule is intended to prevent the translation of the German NP *Sie* which functions as the subject. English imperatives do not have an overt subject and therefore the literal translation *Turn back you the button one position* is unacceptable. Our proposed rule would give *Turn back the button a position*, which is better[1].

In practice, the imperative translation might be handled by a pair of rules. The first could look like this:

X{cat=v,mood=imperative} *Sie*
→
X

The LHS matches cases where there is any imperative verb X followed by *Sie*. The RHS says that the translation of such a structure simply consists of the translation of the imperative verb.

As we have stated it, this first rule has not done any translation. What it has done is to re-order part of the German sentence prior to translation into English. The Engine can now simply apply the lexical translation rules to the re-ordered sentence:

zurückdrehen → turn_back

After applying all these rules, the Engine now has an internal representation of the form *Turn back the button a position.*

Step 4: The Engine would now apply rules which turn the stem or dictionary forms of English words to their inflected forms. As it happens, in the present example, the English

[1] Another possibility would be to have another rule which put the translated preposition immediately after the verb object, giving *Turn the button back a position*.

stem forms happen to be exactly what is wanted. For example, the stem form *turn* which the dictionary supplied is identical to imperative *turn*. Moreover, all the nouns are singular, so it is unnecessary to add any plural affixes (e.g. *s* or *es*).

This discussion is rather sketchy and we have ignored many details. For example, we have said very little about how the various types of transformation rule should be ordered: how should re-ordering rules be interleaved with the bilingual dictionary rules? We have also not said anything much here about how the system copes with ambiguities, or how rules are prevented from applying in the wrong circumstances; for example, it will not always be the case that a preposition at the end of a German clause 'belongs' to an earlier imperative verb. However, this should have given the reader an impression of what is involved in a transformer architecture. We can now summarize some of the distinctive design features of this sort of engine:

- Input sentences are automatically parsed only so far as it is necessary for the successful operation of the various lexical (word-based) and phrasal transformation rules. The transformer engine is often content to find out just a few incomplete pieces of information about the structure of some of the phrases in a sentence, and where the main verb might be, rather than worrying about getting a full and complete parse for the whole thing. In other words, parsing may stop before an S rule of the kind described in Chapter 3 has been applied.

  In practice, transformer systems tend not to have particularly large grammars for the language they translate from. Thus in the German to English transformer system discussed above, we assumed that the grammar covered only some features of German. As a consequence it would not be able to decide for many (or perhaps any) input sentences whether it is grammatically acceptable.

- The use of limited grammars and incomplete parsing means that transformer systems do not generally construct elaborate representations of input sentences — in many cases, not even the simplest surface constituent structure tree. As we will see, other types of MT system construct much more abstract and deep representations.

- Most of the engine's translational competence lies in the rules which transform bits of input sentence into bits of output sentence, including the bilingual dictionary rules. In a sense a transformer system has some knowledge of the **comparative grammar** of the two languages — of what makes the one structurally different from the other.

- Inflection rules aside, transformers generally have no independent linguistic knowledge of the target language because they have no independent grammar for that language. In the German-English system, there would be few, if any, independently stated rules about English — although you could perhaps infer some aspects of English grammar from the rules which transform bits of German into bits of 'English'.

Given these general features, we can describe the translational behaviour that can be expected from a system with a transformer engine.

Characteristic to the performance of such a system is the fact that the engine will not be particularly troubled when faced with unusual, marginally acceptable or frankly unacceptable source language sentences; it will rarely have sufficient source language grammatical knowledge to recognise something as ungrammatical. If the grammatical structures in the input sentence are not recognised by some transforming rule, that structure will pass through to the output sentence without any re-arrangement. We have seen this in the example above, where all the word order and structure of *Drehen Sie den Knopf eine Position zurück* apart from the relationship between *drehen* and *zurück* was passed through into the English output. Something similar is true for the words in the input sentence: if they are not found in the system's dictionary then they are passed through into the English output and remain untranslated. As a consequence of these features this type of architecture implies that, in the worst case, the whole input sentence could survive unchanged as the output sentence. This would happen in the highly unlikely case that none of the input words are found in the bilingual dictionary and none of the input sentence grammatical structure is recognised.

With regard to the target language performance of the system we can say that since the system has no detailed knowledge of target language grammar there is no guarantee that the transformed input sentence is actually a grammatical sentence in the target language. Although in most cases output will resemble the target language (especially the use of target language words), the result can sometimes be a completely unintelligible 'word salad'. In such cases one could say that the output does not belong to any known language — natural or artificial.

The typical design features of a transformer system pose some restrictions on the development of additional language modules. First, the engine will run in one direction only, for example, from German to English. If the engine developer wants it to go in the other direction she more or less has to completely rewrite the transformer rules. Since the transformer rules include bilingual dictionary rules, this can mean that the Engine has to be supplied with two bilingual dictionaries, for example, German-English and English-German. This is rather clumsy since, apart from the differences in their directionality, the dictionaries contain much the same information. Secondly, the engine links a single pair of languages only. If the developer wants it to translate into another target language then again she more or less has to completely re-write the transformer rules. Again, this amounts to rewriting most of the system. Grammatical knowledge of English and of German which is built into a German-English system cannot then be transferred to a English-French or a German-French system. Even in cases where a system contains only a rather limited grammatical knowledge of the languages it involves reproducing this knowledge for the development of other language pairs means an unnecessary time loss.

Drawing these various points together, we can summarise the situation of the transformer engine architecture as follows:

- It is highly **robust**. That is, the Engine does not break down or stop in an 'error condition' when it encounters input which contains unknown words or unknown grammatical constructions. Robustness is clearly important for general-purpose MT.

- In the worst case it can work rather badly, being prone to produce output that is simply unacceptable in the target language ('word salad').

- The translation process involves many different rules interacting in many different ways. This makes transformer systems rather hard to understand in practice — which means that they can be hard to extend or modify.

- The transformer approach is really designed with translation in one direction, between one pair of languages in mind, it is not conducive to the development of genuinely multi-lingual systems (as opposed to mere collections of independent one-pair, one-direction engines).

To close this section, we give an example of a German Teletext Travel News broadcast and a translation produced by an actual small transformer Engine (which is available commercially, and rather cheaply for use on PCs). The source text and the raw (unedited) MT output are given on page 70. The Engine is clearly struggling here with unfamiliar words and structures, occasionally producing completely unintelligible output which would be unsuitable even for gisting. This example represents the 'bottom end' of transformer performance, but gives a good idea of how useful even this quality of translation can be — readers with no knowledge of German will certainly get more information from the translation than they could from the original. Note, however, that the quality of the output could be improved considerably if the system were adapted to dealing with this particular text type and vocabulary. As we mentioned in Chapter 2, tuning the system to a particular text type is worthwhile if the input consists of many texts of that type.

Source Text

VEREINZELT BADEVERBOT
Sommerurlauber an den Küsten Südeuropas oder der Ost- und Nordsee müssen vereinzelt mit Beeinträchtigungen des Badespaßes rechnen.
An der Adria wird bei Eraclea Mare und Caorle wegen bakterieller Belastungen vom Baden abgeraten.
An der Cote d'Azur ist laut ADAC vereinzelt mit Verschmutzungen durch Teer und Öl zu rechnen.
Auch in Spanien werde an einigen Stellen bei Barcelona vom Baden abgeraten.
Zufriedenstellend lautet die Wertung für die Nordsee in Schleswig-Holstein und den Niederlanden.
Zugleich treten aber in der Nordsee vereinzelt tennisballgroße Phenolklumpen auf.

Unedited Output

ISOLATED BADEVERBOT
Summer vacationers at the coasts of South Europe or the east - and North Sea must calculate isolated with impairments of the bath joke.
At the Adria Mare and Caorle is dissuaded at Eraclea because of bacterial burdens from the bath.
At the Code D'Azur is to be calculated loudly ADAC isolated with pollutions through tar and oil. Also in Spain am dissuaded at some places at Barcelona from the bath.
Satisfactorily the appraisal sounds for the North Sea in Schleswig-Holstein and the Netherlands. At the same time tennisballegrosse appear however in the North Sea isolated Phenolklumpen.

4.3 Linguistic Knowledge Architectures

The second major architecture — **indirect** or **linguistic knowledge** (LK) architecture — has dominated research in MT design during the past decade and is starting to appear in a number of commercial systems. The idea behind LK engines is straightforward enough:

> High quality MT requires linguistic knowledge of *both* the source and the target languages as well as the differences between them.

We use the term 'linguistic knowledge' to refer to extensive formal grammars which permit abstract/relatively deep analyses in the sense of Chapter 3. We shall see later on just how deep the analysis can go.

With the Transformer architecture, the translation process relies on some knowledge of the source language and some knowledge about how to transform partly analysed source sentences into strings that look like target language sentences. With the LK architecture, on the other hand, translation relies on extensive knowledge of both the source *and* the target languages and of the relationships between analysed sentences in both languages. In short, LK architecture typically accords the target language the same status as the source language. As can be seen from Figure 4.2, the LK architecture requires two things:

- A substantial grammar of both the source language and the target language. These grammars are used by parsers to analyse sentences in each language into representations which show their underlying structure, and by generators to produce output sentences from such representations.

- An additional comparative grammar which is used to relate every source sentence representation to some corresponding target language representation — a representation which will form the basis for generating a target language translation.

The LK engine will have grammars for each language it deals with: in a German-English system, there would be one for German and one for English. Each of these grammars is an independent entity, i.e. there will be a set of rules which is identifiably for German, and another, separate set which is identifiably for English. In fact the physical and conceptual separation between the two grammars is such that in the initial stages of developing an LK engine, a group of English specialists might write the grammar for English entirely independently of another group of German specialists who are writing the system's German grammar. In such case both groups would have to aim though at a similar deep representation of their language, otherwise structural discrepancies can be created that would require extra transfer rules for mapping these different structures onto each other.

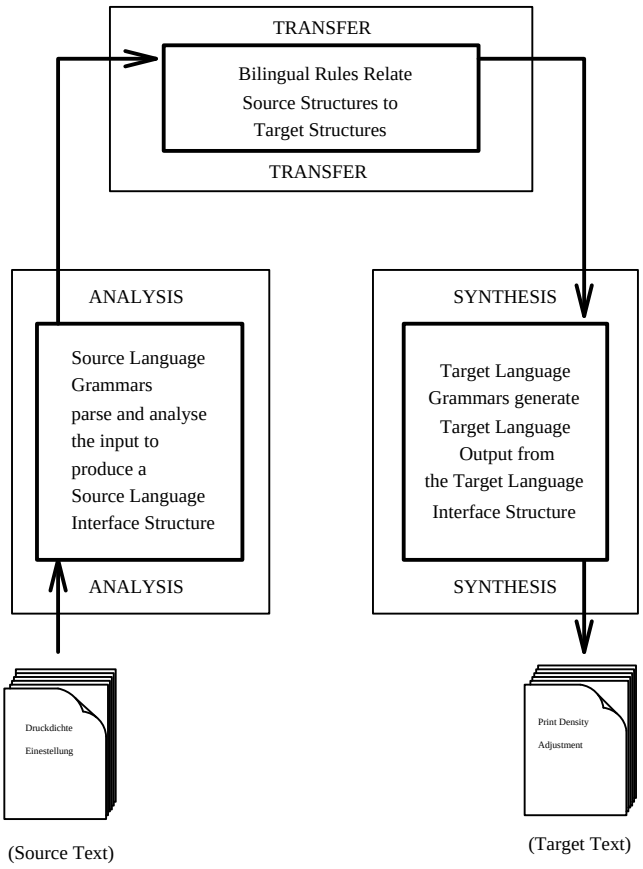


Figure 4.2 The Components of a Transfer System

Looking at Figure 4.2, it is clear that if (say) the system is translating from German to English, the first (analysis) step involves using the parser and the German grammar to analyse the German input. The second (transfer) step involves changing the underlying representation of the German sentence into an underlying representation of an English sentence. The third (synthesis) step and final major step involves changing the underlying English representation into an English sentence, using a generator and the English grammar. The fact that a proper English grammar is being used means that the output of the system — the English sentences — are far more likely to be grammatically correct than those of a German-English Transformer system (recall that the latter had no explicit English grammar to guide it). In fact, if (*per impossibile*) we had an LK German-English system with a 'perfect' English grammar the only sort of mistake it could make in the output would be errors in translational accuracy. That is, it would always produce perfectly well-formed English sentences even when it did not produce correct translations.

This also means that the whole Engine should be reversible, at least in theory. Taking the German-English LK engine in Figure 4.2, we could run the translation from right to left. That is, we could give it English sentences, which would then be analysed into underlying representations. These representations would be changed into German underlying representations and a German translation would then be synthesised from the result. The

same grammars for each language are used regardless of the direction of the translation. In practice few translation engines are reversible, since some rules that are necessary for correct translation in one direction could cause problems if the process was reversed. This is especially true for lexical transfer rules, as we will see later on in this chapter.

With this general picture in mind, the next subsection focusses on the so-called transfer component, which embodies the comparative grammar that links the analysis and synthesis components together — the module in the centre of Figure 4.2.

4.3.1 Comparative Grammar and Transfer

We have said that parsers in LK engines typically analyse to relatively abstract, or deep underlying representations. Of course individual systems differ radically in the precise sorts of representations they use, but suppose the Engine uses the English grammar to produce the sort of deep syntactic representation we described in Chapter 3 (this is far from being the most abstract representation one can imagine, of course). If we are translating sentence (2) into German, the analysis component might produce a representation along the lines of Figure 4.3

(2) The temperature has affected the print density.

We can look at how the comparative grammar relates such a representation to corresponding representations for target language sentences. Just as each monolingual grammar has a 'dictionary' of rules (e.g. `N` $\rightarrow$ `temperature`) so also the comparative grammar has bilingual dictionary rules. In the simplest case, these may just relate source lexical items ('words') to target lexical items:

`temperature` $\leftrightarrow$ `temperatur`

`print_density` $\leftrightarrow$ `druckdichte`

`affect` $\leftrightarrow$ `beeinflu`β`en`

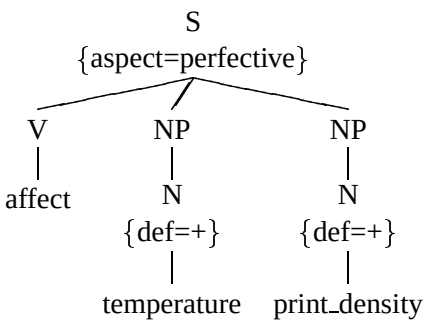


Figure 4.3 Abstract Tree Representation

One difference between these bilingual dictionary rules and those shown for the Transformer engine is that the latter were intended to be used in one direction only. The $\leftrightarrow$ in

the present rules indicates that they can (in principle) serve as English-German or German-English rules.

These dictionary rules can be seen as relating leaves (the word nodes) on the source language tree to leaves on the target language tree. The comparative grammar also contains some structural rules which relate other parts and nodes of the two trees to each other.

One such structural rule might be read as follows: "The translation of the whole sentence is normally made up of the translation of the verb + the translation of the subject + the translation of the object." Note that 'translation' in this context has the restricted sense of translation into the corresponding target language representation — this representation has to be input to synthesis before a 'full' translation is reached. The structural rule we need might be written in the following way (where the LHS describes an English structure and the RHS describes the German, and `$H`, `$S`, and `$O` are variables interpreted as standing for pieces of English structure on one side, and for their translations on the other side).

$[_S$ `HEAD:$HEAD, D-SUBJ:$SUBJECT, D-OBJ:$OBJECT` $]$
$\leftrightarrow$
$[_S$ `HEAD:$H, D-SUBJ:$S, D-OBJ:$O` $]$

The left and right hand sides of the rule reflect the 'canonical' order (HEAD, then DEEP SUBJECT, then DEEP OBJECT) that one finds in the source (and target) representations. In some systems, the rule application procedure might be set up so that rule would work regardless of the left-right order of the nodes in the source representation.

This rule says that in the translation of the sentence as a whole, the HEAD is whatever the HEAD in the source language translates as. The HEAD is the verb *affect*, and its translation is given by a bilingual dictionary rule. The DEEP SUBJECT and DEEP OBJECT just contain single content words (*temperature* and *print_density*) and so they too are translated by the appropriate dictionary rules.

The annotations on the nodes of the representations must also be translated in some way. The rules relevant to our example are straightforward, indicating that the given values are simply carried over from source structure to target structure:

{`def=+`} $\leftrightarrow$ {`def=+`}

{`aspect=perfective`} $\leftrightarrow$ {`aspect=perfective`}

Of course, one could imagine that this 'copying' of information without changes could occur by default, i.e. features are copied unless a rule explicitly says otherwise (although specifying how this sort of system should actually work turns out to be surprisingly difficult).

Applying these rules to the English representation in Figure 4.3 will result in the construction of the corresponding German representation in Figure 4.4.

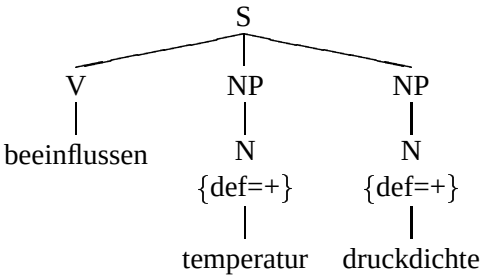


Figure 4.4 Tree Representation after Translation

This representation serves as input for the German synthesis module, which applies the rules of the German grammar to produce a German sentence. These rules will include one or more which require that the past participle of a verb is realised at the end of the clause when there is an auxiliary (*hat*, in this example). Thus, (3) should be produced as the translation.

(3) Die Temperatur hat die Druckdichte beeinflußt

It should be clear that LK and Transformer architectures handle the word order problem rather differently. A Transformer engine generally preserves the surface order of the source language and directly re-uses it — with modifications where appropriate — to order the target language words. An LK engine, on the other hand, extracts all the information it can from the source word order and recodes this information in a more or less abstract representation. The generator for the target language will use the information in the representation and in the target language grammar to construct a target language sentence with a word order that it is grammatically appropriate for that language. In short, ordering information is not normally carried over directly.

The only differences between the English and the German representation in this example is in the words on the leaf nodes; the geometry and annotations on the tree are the same. Ideally, this similarity will hold for most sentences, so that most of the work in constructing the representation is done by the dictionary rules. However, it is important to realise that the design of the comparative grammar anticipates the possibility that the structures *could* be very different indeed if the differences between the source and its target language translation are very great. We will look at some such examples in the following chapters (cf. especially Chapter 6).

The similarity of the representations is related to the simplicity of the rules. For example, according to the rule, DEEP SUBJECTS translate as DEEP SUBJECTS, and DEEP OBJECTS as DEEP OBJECTS, and the rules for translating the words are stated without any conditions. But in general, one would only want to say that subjects and objects are *normally* translated as subjects and objects, and it is easy to think of cases where one would want to put extra conditions on such lexical rules. For example, English *import* translates

as French *importer* when it is a verb, and *importation* when it is a noun, and the verb *effect* translates *réaliser* or *effet*, depending on whether it is a noun or a verb. Such examples can be multiplied at will. Similarly, one cannot always simply preserve the values of features such as `det`, or `aspect`. For example, in translating from English to French, one cannot generally expect to preserve the values of attributes indicating tense and aspect, if these are direct encodings of surface word forms (cf. Chapter 7).

A relatively straightforward example where a more complex rule is called for involves the translation of the English verb *like* into French *plaire*, as in (4), which shows the 'switching' of arguments.

(4) a. Sam likes the new laser printer.
b. La nouvelle imprimante à laser plaît à Sam.

Such a rule might look as follows:

$[_S$ `HEAD:like, SUBJ:$1, OBJ:$2` $]$
$\leftrightarrow$
$[_S$ `HEAD:plaire, SUBJ:$2, OBJ:$1` $]$

Switching of arguments occurs because the variables `$1`, and `$2` are associated with different grammatical relations on the two sides of the rule (`$1` will be bound to the representation of *Sam*, and `$2` will be bound to the representation of *the new laser printer* (on the English side of the rule), and *la nouvelle imprimante à laser* (on the French side of the rule)). The identity of the words that fill the HEAD relation has been given to prevent this rule applying to examples involving 'normal' verbs (one will also have to make sure that the 'normal' rules do not apply in translating *like* and *plaire*, of course). This process of argument switching is illustrated in Figure 4.5.

Special rules like the one given above have to be written for *every* case where there is some difference between the output of the source language analysis and the input expected by the target language generator. In practice, one would expect the contrastive grammar for an English-French, or English-German MT system whose most abstract representations involve surface grammatical relations to be quite large.

In general, the size and complexity of a comparative grammar can be reduced by increasing the depth of the parsing towards more abstract levels of representation. For example, the use of *Semantic Relations* (see Chapter 3) would remove the need for a special *like-plaire* rule, because both English and French sentences in (4) would have representations with *Sam* as EXPERIENCER, and *the new laser printer/la nouvelle imprimante à laser* as THEME.[2]

[2]The names of these particular Semantic Relations should not be taken too seriously. In fact, of course, it does not much matter *what* the relations are called, so long as they are the same in the source and target grammars.

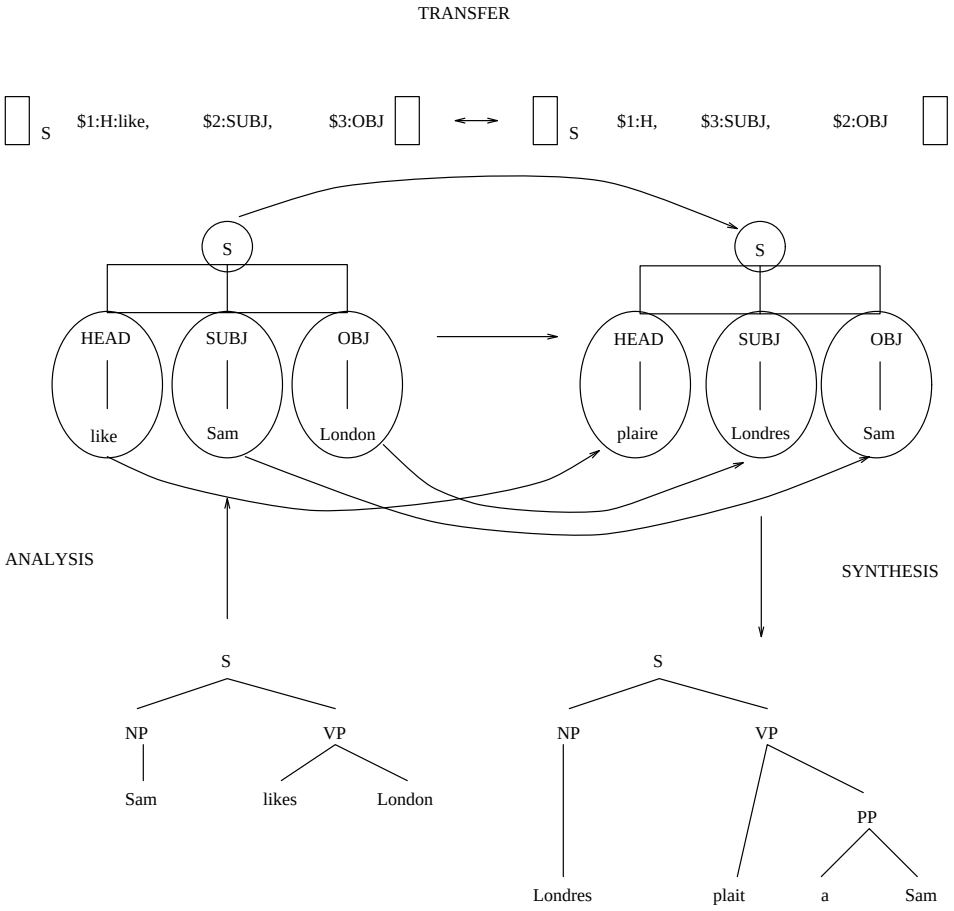


Figure 4.5 Complex Transfer

The discussion so far may give the impression that there is a single transfer approach to MT. But this is far from being the case. For one thing, different systems use different styles, and levels of representation. For another thing, we have only given one view of the relation of the various components. That other views are possible is indicated below, where we discuss some variable aspects of transfer systems.

Intermediate representations in transfer As we have described transfer, the mapping between source and target structure is direct in the sense that there are no intermediate structures. There are, for example, no structures which have target words, and source geometry. Some systems, however, make a distinction between lexical transfer (which simply changes source words to target words) and structural transfer (where rules actually change the shape of the tree) with one set of rules being applied before the other. Also, the rules we have given each deal with a structure in one step, without using an intermediate representation. But it is possible to have a transfer rule which changes the source tree in some way, producing an intermediate representation, that must have another rule applied to it before a genuine target structure results. The problem with systems that allow this is that problems of complex

rule interaction can occur, in the way that they do with a transformer architecture. We have allowed for a limited degree of collaboration between rules that deal with structure, and rules that deal with features, for example. The advantage of this is that we do not have to state facts about the relation between, for example, determination values in each rule. This seems both natural and economical in terms of effort involved. The disadvantage of this is that it increases the number of rules that must be applied in order to translate each tree. An alternative is to state the rules separately like this, but in some way compiling them together, to produce rules that deal with entire subtrees. The problem with this is that the set of compiled rules tends to be very large.

Symmetry Throughout this chapter the picture of transfer that we have described is rather **symmetric**. That is, it assumes the target structure is rather similar to the source structure in the sense of being of corresponding depth of analysis or linguistic abstraction. This suggests analysis and synthesis are to a large extent 'inverses' of each other. But this is not a requirement. It is possible to imagine systems where the input to transfer was a deep syntactic representation, and the output was a representation of surface syntactic structure. Moreover, in a one-directional system for one pair of languages, no real distinction might be drawn between transfer and synthesis. Symmetry is however desirable as soon as one deals with more than one language or direction. In such cases the advantages become obvious, having a separate synthesis component with a role broadly the inverse of to that of analysis — not only can the same synthesis component be used for all transfer pairs, but one will avoid duplicating work by using the same (or similar) grammars in analysis and synthesis.

Reversibility We noted that transfer rules could be reversible *in principle*, and though this is natural, and attractive (because it halves the number of transfer components one has to construct and makes testing easier, since, if a rule works in one direction it should work in the other), it is not obvious that reversible transfer rules are always possible, or desirable. This is because a system should be able to translate a wide variety of input strings, some of them the type of string that one would normally not want to produce as output. As a simple lexical example of the reversibility problem consider the slightly old-fashioned Dutch word *aanvangen*. One would like to be able to translate this into English as *begin*, but one would normally not want to translate *begin* into *aanvangen*. One would choose the more common verb *beginnen* instead. So the following translation rule cannot be reversible:

$$\texttt{aanvangen} \rightarrow \texttt{begin}$$

Well-formedness In order for transfer output to be useful for synthesis it is desirable that it is in some sense well-formed for the target language. To produce well-formed target language structures transfer components can become rather complex. Some systems supplement normal transfer with a set of adjustment rules which transform the output of transfer to make it more suitable for input to the target synthesis.

Instructions for synthesis The target structure that is produced by transfer has been described as a simple linguistic tree — it does not contain, for example, special instructions to guide synthesis. Some systems do contain this sort of information: transfer

attaches what are essentially small programs to nodes of the target tree, which are executed in synthesis.

Choosing between possible translations In general, several different transfer rules will be able to apply to a structure, giving alternative (not necessarily correct) translations. The question arises as to how to choose between these. One crude possibility is to organize the rules so they apply in sequence, taking the results of the first rule that produces a 'correct' target structure (correct in the sense of getting an acceptable target sentence, perhaps). Alternatively, one could apply all these rules and find some way of scoring the results, so as to prefer the better ones. A complementary question which arises in the case where no translation rule applies (because none matches the source structure) is whether one should leave the structure untranslated (it may be, for example, a proper name), or to try to force a rule to apply?

Declarative or procedural processing If the answer to the problem above is to organize the rules so they apply in sequence then the result is the contamination of **declarative** information in the comparative grammar with **procedural** information – information about the order in which things should be done. This violates a widely accepted principle that it should be possible to describe the relevant linguistic facts in an MT system independently of the ways the engine actually uses them. The advantages of a declarative system are (a) ease of understanding, modification and debugging, and (b) independence of particular implementations or algorithms: if a collection of rules is declarative, it will be possible to consider alternative algorithms for applying them, with some confidence that the same results will be produced, which allows one to find the most efficient way of processing. Despite these advantages of declarativity there is a strong temptation to introduce non-declarative characteristics (e.g. to ensure that the most likely transfer rules are tried early, and block the application of other rules, so cutting down the space of possibilities that have to be processed). Thus, though declarativity is a generally accepted goal, it is a property that systems have in different degrees, and it is not even generally agreed what the correct compromise between efficiency and declarativity is.

4.3.2 Interlinguas

The general idea suggested by the discussion of the *like-plaire* example at the end of the previous section is that comparative grammar (hence transfer) becomes simpler as linguistic analysis goes deeper — as the representations become more abstract. In fact, a major objective of MT research is to define a level of analysis which is so deep that the comparative grammar component disappears completely. Given such a level of representation, the output of analysis could be the direct input to the target synthesis component. Representations at such a level would have to capture whatever is common between sentences (and expressions of other categories) and their translations — that is they would have to be representations of 'meaning' (in some sense). Moreover, such a level of representation would have to be entirely language independent — for example, if it preserved features of the source language, one would still require a transfer component of some kind to produce the corresponding features of the target language. For this reason, such a level of repre-

sentation is normally called an **Interlingua**, and systems that use such a level are called **Interlingual**.

The relationship between transfer and interlingual systems can be pictured as in Figure 4.6. As one can see, the size of the contrastive grammar (hence the transfer component) between two languages decreases as the level of representation becomes more abstract. As this diagram perhaps suggests, the difference between transfer representations and interlinguas is a matter of degree rather than absolute distinction (for example, Chapter 7 shows how one might combine an interlingual representation of tense and aspect with a transfer approach to other phenomena).

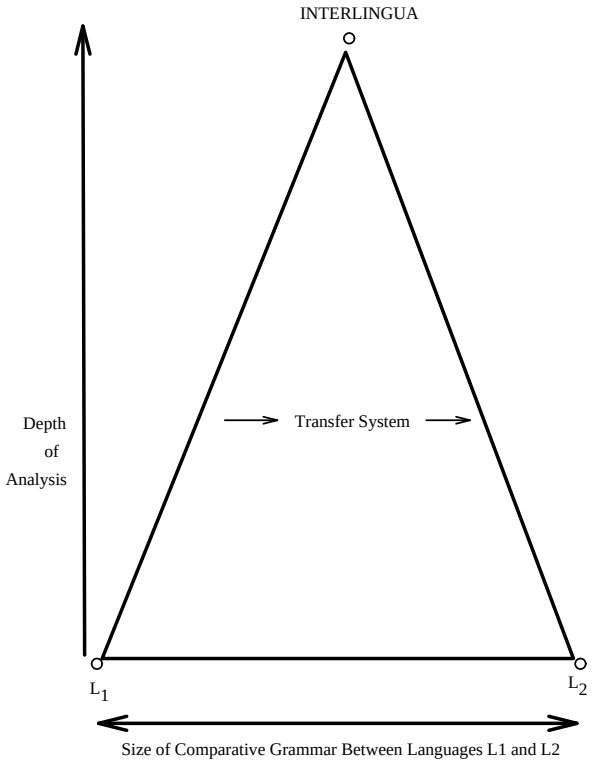
There are a number of clear attractions to an interlingual architecture. First, from a purely intellectual or scientific point of view, the idea of an interlingua is interesting, and exciting. Second, from a more practical point of view, an interlingual system promises to be much easier to extend by adding new language pairs, than a transfer system (or a transformer system). This is because, providing the interlingua is properly designed, it should be possible to add a new language to a system simply by adding analysis and synthesis components for it. Compare this with a transfer system, where one needs not only analysis and synthesis, but also transfer components into all the other languages involved in the system. Since there is one transfer for each language pair, N languages require $N \times N - 1$ transfer components (one does not need a transfer component from a language into itself). For example, extending a system for 3 languages into one for 5 means writing 14 new transfer components (as one goes from 6 to 20 transfer components), and going from a 5 language system to a 9 language system means going from 20 components to 72.

Ideas about interlinguas are intimately tied up with ideas about the representation of meaning. We will look at this in more detail in Chapter 7. However, one can get a flavour of the problems that are involved in defining an interlingua by considering the following.

Producing an interlingual representation involves producing a representation that is entirely language independent (for the languages one wants to translate, at least). This involves producing a language independent representation of words, and the structures they appear in. Under the latter heading, one would have to make sure one could represent the difference in meaning between examples like those in (5) — assuming one does not want them all to translate alike, that is — and find a way of representing the meaning that is expressed by various tenses, and by the distinction between definite, and indefinite NPs (e.g. *a printer* vs. *the printer*).

(5) a. It was the printer that was serviced yesterday.
b. It was yesterday that the printer was serviced.
c. The printer was serviced yesterday.

While this raises many unsolved linguistic problems, it is the language independent representation of word meaning that seems to pose the most difficult problems. The central problem is how to choose the vocabulary of the interlingua — what are the primitive concepts of the meaning representation to be. Notice that this is not a question of what *names*



The size of the comparative grammar that is required to translate between two languages gets smaller as the 'depth' of the representations used increases. As the representations become more abstract, there are fewer differences between source and target representations and it is easier to relate them. Ultimately, a level of representation may be achieved where source and target representations are identical, where no comparative grammar is needed. In this situation, the representations which are produced by analysis could be directly input to the target language synthesis component. Such a level of representation is called an **interlingua**, and a system that uses such a level is called an **interlingual** system.

Figure 4.6 Transfer and Interlingua

we should give the concepts — how we should write them down or represent them. Of course, we should make sure that we do not use one name for two concepts, which might be confusing, but beyond this, we can give them, for example, names from an existing language (e.g. English, or Esperanto), or numbers, or codes in some invented language — the only difference here will be how easy they are to write or remember. The problem is one of identity. For example, are we to include a concept that we might write as `CORNER` — this being the interlingual representation of the English noun *corner*? This seems natural enough from the point of view of English, but from the point of view of, for example, Spanish it is not so natural, because in Spanish there are different words for inside corners (*rincón*) and outside corners (*esquina*). Is there any reason why we should not choose a more specific primitive word for our representation, for example, `OUTSIDE-CORNER` and `INSIDE-CORNER`. Similar problems will arise wherever one language has several words that correspond to one word in another. The point is that different languages 'carve the world up' differently, so settling the choice of vocabulary for the interlingua will involve either (i) some apparently arbitrary decisions about which language's conceptualization to take as basic, or (ii) 'multiplying out' all the distinctions found in any language. In the latter case one will have two interlingual items for English *corner* (because of Spanish), two for English *river* (because of the distinction between *rivière* and *fleuve* in French), and two for English *eat*, because of the distinction between *essen* (for humans) and *fressen* (for animals) in German. When one consider more distant languages like Japanese, even more distinctions will arise — Japanese does not distinguish between wearing and putting on, as does English, but does make a distinction according to where the item is worn or put on (e.g. on the head vs on the hands). Of course, one solution to this multiplicity of concepts is to try to reduce the set of *primitive* concepts, defining complex concepts in terms of the primitive ones. For example, one might think that `EAT` is not a primitive, but that `INGEST` is, and that the interlingual representation of the meaning of *eat* should involve `INGEST`, and some other primitives. However, though this solves the problem of the number of concepts, it does not overcome the problem of arbitrariness, and it raises the problem of finding an adequate set of primitives to capture the relevant distinctions (the reader might, as an exercise, like to consider what a set of primitives would look like to distinguish a handful of verbs like *eat*, *drink*, *gobble up*, *feed on*, or find a set of primitives that will distinguish between different kinds of furniture (chairs, stools, tables, etc.)).

A further problem is that using an interlingua in MT can lead to extra, unnecessary work, in some cases. For example, suppose one has an interlingua intended for translation between English, French, and Japanese. Japanese distinguishes terms for older and younger brother and sister, and for various relatives depending on whether they belong to the speaker, or to the hearer (i.e. the term for *my mother* is different from that for *your mother*, or *mothers in general*). The problem is that this distinction has to be encoded in the interlingua, so one must decide if English *brother* is an older brother or a younger brother, even if one is not translating into Japanese. For example, translating *Sam's brother has already left* into French will involve dealing with an ambiguity, since there will be two interlingual representations differing as to whether the brother is older or younger than Sam. But of course, this is irrelevant for both English and French, and one can manage with a very simple transfer rule (along the lines of `brother` $\rightarrow$ `frère`).

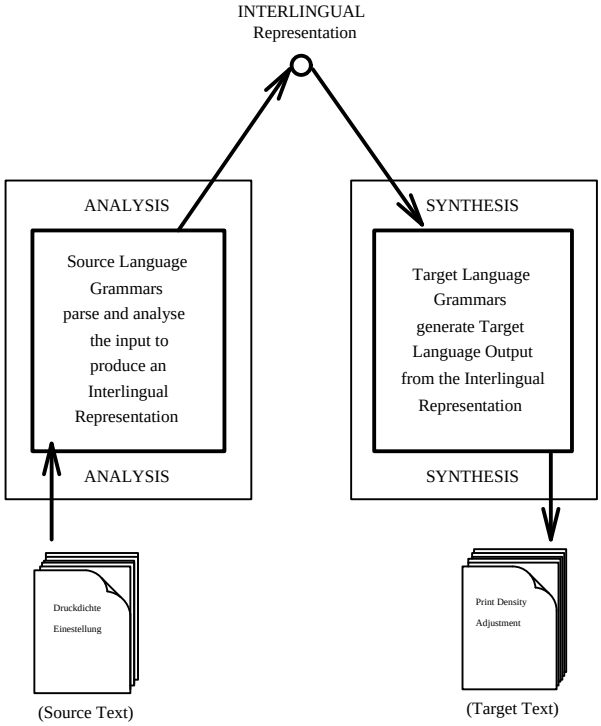


Figure 4.7 The Components of an Interlingual System

These are problems for general vocabulary. One should note, however, that these problems do not occur for all kinds of vocabulary. In particular, in domains where there is a codified system of terminology, the conceptual organization is generally relatively clear. In such cases, the set of concepts, and thus at least some of the vocabulary of the interlingua, is already settled. Interlinguas are rather metaphysical things. Implicitly or explicitly, they say what the universe is made of (events, processes, individuals, relations, etc.) and how it is put together. It is not at all surprising that many aspects of interlinguas are in dispute and are likely to remain so for some time to come. Given these difficulties, interlinguas in the sense described here are more popular as a basis for theoretical research in MT rather than for full-scale commercial development. For the next few years, most general purpose LK MT systems on the market are unlikely to analyse any deeper than to the level of semantic relations — and even that will be considered impractically deep by many developers and vendors. Nonetheless, we can certainly expect a tendency towards increasingly deep analysis over the next decade or so.

4.3.3 LK Engines Summarised

Having looked at some of the components of an LK engine and having seen something of how they might work, we can conclude this discussion of MT architectures by setting out what the performance characteristics of an LK engine might be.

- Because the system has a (partial) grammar of the target language, output will tend to be grammatical. At any rate, it will be far less strange and far less source-language grammar- dependent than output from transformer engines.
- Because the comparative grammar *completely* specifies a relationship between representations of two languages, translational quality will tend to be more reliable than for transformer engines.
- Because the system tends to separate language into separate modules (one grammar for each language and one comparative grammar for each pair of languages), it is relatively easy in principle to add new languages to the system. For example, adding Dutch to a German-English system would require only the addition of a Dutch grammar module and Dutch-English and German-English comparative grammar modules. Individual language modules can be designed and constructed without specifying which other language modules they will have to work with in the final system. Of course, this matters more to the developer than the user since it is the former that writes and supplies basic language modules.
- The system will be upset by unusual, marginally acceptable or frankly unacceptable input sentences because it has a grammar for the source language and hence a strong notion of grammaticality.
- Because the grammars that computational linguists are able to write are invariably less complete than the 'real' complete grammar of any language, there will be some complicated grammatical input sentences that the system fails to recognise.

From the engine manufacturer's point of view, the transformer architecture has the advantage that it accepts anything that is given to it (though the translations it produces are another matter). The LK architecture is at a disadvantage here: because it thinks it knows something about the languages involved, it tends to think that anything it doesn't know isn't language and hence unacceptable. As a consequence, a pure LK engine during its development phase tends to grind to a halt on anything unusual, or even on something quite common which the developer has forgotten to include.

For commercial purposes, this means that pure LK engines must be supplemented with various coping strategies. For example, if they cannot parse a particular sentence completely, then they at least ought to be able to use some of the information on those parts of the sentence for which they did find a parse — and perhaps they can guess how those well-parsed bits might be fitted together.

LK systems are clearly superior *in principle* to transformers. However, MT systems require a considerable development effort and some commercial transformer systems which have undergone extensive revision, refinement and updating over the years can achieve a good overall performance. Furthermore, some MT systems have sufficient flexibility in the design of the engine to allow developers to increase the depth and sophistication of their linguistic knowledge and even the overall arrangement of grammars. We can therefore expect highly developed transformer MT systems to survive in some sectors of the marketplace for some years to come.

4.4 Summary

In this chapter we have looked inside two different kinds of MT system, transformer systems, and linguistic knowledge systems, discussing, under the latter heading the distinction between transfer and interlingual systems. The following chapters will amplify this picture in various ways, by looking in more detail at the sorts of knowledge that are involved, for example, in dictionaries, and the representation of 'meaning', and looking at some particular translation problems. In Chapter 10 we will give some more discussion of the limitations of LK approaches, and describe a recently developed alternative.

4.5 Further Reading

Probably the most famous example of a system with what we have called a transformer architecture is SYSTRAN. This is described in Hutchins and Somers (1992). A recent discussion can be found in Wilks (1992).

A more detailed overview of transfer systems can be found in Arnold (1993).

Examples of transfer systems include the following, ARIANE Vauquois and Boitet (1985), SUSY Maas (1987), MU (the Japanese National Project) Nagao et al. (July 1986), METAL Slocum et al. (1987), Bennett and Slocum (1988), TAUM-AVIATION Isabelle (1987), ETAP-2 Apresian et al. (1992), LMT McCord (1989), EUROTRA Arnold (1986); Arnold and des Tombe (1987); Copeland et al. (1991a,b), , CAT-2 Sharp (1988), MIMO Arnold and Sadler (1990), MIMO-2 van Noord et al. (1990), ELU Estival et al. (1990). Several of these systems are discussed in detail in Hutchins and Somers (1992).

Among interlingual systems, the following are noteworthy: Rosetta Landsbergen (1987b,a), KBMT Goodman (1989), Goodman and Nirenburg (1991). A recent overview is given in Nirenburg (1993). (Hutchins and Somers, 1992, Chapter 6) is also recommended. One interlingual approach that we have not mentioned here is that which uses a human language as the interlingual. The best known example of this is DLT, which uses Esperanto, see Schubert (1992) and (Hutchins and Somers, 1992, Chapter 17).