Machine Translation with the ACQUILEX LKB

Arturo Trujillo
Computer Laboratory, University of Cambridge[1]
July 1, 1995

1 Introduction

This working paper describes a prototype lexicalist machine translation (MT) system which extends the Acquilex Lexical Knowledge Base (LKB) in order to demonstrate the feasibility of using lexicographic tools in MT. The system is based on the so-called 'Sake-and-Bake' (SB) approach to translation due to Whitelock (1992). Under this approach, translation consists of three phases: analysis, lexical transfer and bag generation. Analysis results in a set of instantiated lexical signs (Pollard and Sag 1987) constituting the input to the transfer phase.

In the original formulation of SB the input to transfer was treated as a bag or multiset which ignored source language (SL) linear order. However, in our implementation, efficiency considerations have led us to treat the input as a list. Thus, in this paper transfer involves mapping a list of SL lexical signs into a bag of target language (TL) lexical signs. Bag generation consists of ordering the TL bag into a grammatical sentence; this is done through a modified chart parser which effectively attempts every permutation of the TL bag. Much effort in lexicalist translation is needed to tackle the inefficiencies of this generation process, and the paper will address this issue in some detail.

The paper begins with an introduction to the LKB and Typed Feature Structures (TFSs). This is followed by a description of a representative sample of rule-based transfer machine translation systems and a fuller description of lexicalist translation and related approaches. A number of heuristics for bag generation are described and the results of a small experiment into their efficacy are presented. The conclusion summarizes the main ideas explored here, and points to areas for future research.

2 Lexical Knowledge Base

The LKB was developed as part of the ACQUILEX project (Copestake et al. 1993) as a tool for the representation, acquisition and study of multilingual lexical knowledge. It uses TFSs (Carpenter 1992) as its representation formalism for describing monolingual and bilingual lexicons (bilexicons henceforth) and lexical processes. It is possible to write grammars, parse sentences and carry out a limited amount of translation with the LKB. Furthermore, a number of substantial lexicons have been developed semi-automatically from machine readable dictionaries (Sanfilippo and Poznański 1992; Vossen and Copestake 1993), and proposals have been made for representing translation mismatches in the LKB (Sanfilippo et al. 1992).

We begin with a description of lexical entries. A lexical entry in the LKB is called a *psort*. This is simply a TFS which has been given an identifier (cf. sense identifiers in other systems). For instance, the identifier 'lamb_1' has been assigned to the following TFS:

[1]Now at SCMS, The Robert Gordon University, St Andrew Street, Aberdeen AB1 1HG, Scotland. E-mail: `iat@scms.rgu.ac.uk`

$$\texttt{lamb\_1} \qquad \begin{bmatrix} \textbf{n-com} \\ \textsc{orth} = \textbf{lamb} \\ \textsc{syn} = \boxed{\textbf{syn}} \\ \textsc{sem} = \boxed{\textbf{sem}} \end{bmatrix}$$

The TFS shown on the right follows sign-based approaches to grammar (Pollard and Sag 1987). Types are in bold face, features are in small capitals. The boxed or "shrunk" notation (e.g. $\boxed{\textbf{syn}}$) indicates that the TFS has been hidden to ease presentation. Reentrancy (not shown in this example) will be depicted by boxed integers (e.g. $\boxed{2}$).

Lexical processes are expressed by lexical rules which derive new lexical entries from other lexical entries (Shieber et al. 1983; Briscoe and Copestake 1991). For example, the simplified rule that derives a mass noun from a count noun is:

$$\begin{bmatrix} \textbf{lexical-rule} & \\ 1 = & \begin{bmatrix} \textbf{n-com} \\ \textsc{orth} = \boxed{0}\,\textbf{orth} \\ \textsc{syn:noun-type} = \textbf{count} \\ \textsc{sem} = \boxed{\textbf{sem}} \end{bmatrix} \\ 0 = & \begin{bmatrix} \textbf{n-com} \\ \textsc{orth} = \boxed{0} \\ \textsc{syn:noun-type} = \textbf{mass} \\ \textsc{sem} = \boxed{\textbf{sem}} \end{bmatrix} \end{bmatrix}$$

The path SYN:NOUN-TYPE identifies the value of the feature NOUN-TYPE in the value of SYN. The above rule may form the basis of, for instance, a more complex rule that constructs the meat sense of a noun from its animal sense:

The shepherd lost a *lamb* in the fog.
Kebabs are made of *lamb*.

Shared value $\boxed{0}$ ensures that the orthography of the output lexical sign is the same as the input orthography[2].

Bilingual knowledge in the LKB is expressed through *tlinks*. These are pairs of source and target language TFSs which establish a transfer relation. One type of *tlink* consists of source and target lexical rules, the outputs of which are in transfer correspondence. The simplest of such *tlinks* involves identity rules which simply share their input and output signs; these *tlinks* are equivalent to the standard notion of a simple lexical transfer relation as represented by an entry in a bilexicon.

However, many transfer relations are best captured by using contentful lexical rules. For example, the translation of 'furniture' into Spanish is *muebles* which is the plural of *mueble* (a piece of furniture). While it is natural to expect a Spanish lexicon to have the singular noun *mueble*, its plural is best derived through a plural lexical rule such as the one below:

$$\begin{bmatrix} \textbf{plural-rule} & \\ 1 = & \begin{bmatrix} \textbf{n-com} \\ \textsc{orth} = \boxed{0}\,\textbf{orth} \\ \textsc{syn:number} = \textbf{sg} \\ \textsc{sem} = \boxed{\textbf{sem}} \end{bmatrix} \\ 0 = & \begin{bmatrix} \textbf{n-com} \\ \textsc{orth} = \boxed{0}\,\textbf{-s} \\ \textsc{syn:number} = \textbf{pl} \\ \textsc{sem} = \boxed{\textbf{sem}} \end{bmatrix} \end{bmatrix}$$

The *tlink* to express the equivalence 'furniture - *muebles*' may then be depicted schematically as:

$$\begin{array}{lcl} \text{furniture} & & \text{mueble} \\ \Downarrow \text{identity-rule} & & \Downarrow \text{plural-rule} \\ \text{furniture} & \leftrightarrow & \text{mueble-s} \end{array}$$

[2] How information is shared between input and output lexical signs in a lexical rule is not well-understood yet (Meurers 1994; Lascarides et al. in press), particularly as it concerns the type of the input and the output signs, and the amount of information shared between the two.

It is the bottom line which represents the appropriate translation relation. The simplified TFS for this *tlink* is:

$$
\left[\begin{array}{ll}
\multicolumn{2}{l}{\textbf{simple-tlink}} \\
\textsc{sfs} = & \left[\begin{array}{ll}
\multicolumn{2}{l}{\textbf{identity-rule}} \\
1 = & \boxed{0}\ \boxed{\textbf{n-com}} \\
0 = & \boxed{0}
\end{array}\right] \\
\textsc{tfs} = & \left[\begin{array}{ll}
\multicolumn{2}{l}{\textbf{plural-rule}} \\
1 = & \boxed{\textbf{n-com-sg}} \\
0 = & \boxed{\textbf{n-com-pl}}
\end{array}\right]
\end{array}\right]
$$

In this TFS, the English lexical rule is found at SFS, while the Spanish rule is at TFS. Translation of a lexical sign with a *tlink* is done by unifying the lexical sign in the source sentence with the value of the path SFS:0 or TFS:0, depending on the direction of translation. If unification is successful, the target language (TL) sign is the value of TFS:0 or SFS:0 respectively.

Another type of *tlink* relates a phrasal TFS either to a lexical sign or to another phrasal TFS. This kind of *tlink* is used for translations which involve more than one lexical sign on the source or target language. For example, the translation of Spanish *novillo* as 'young bull' would require such a *tlink*; its structure may be depicted as follows:

$$
\begin{array}{ccc}
\text{N1} & \Leftrightarrow & \text{novillo} \\
\{\text{young, bull}\} & &
\end{array}
$$

This says that the translation of *novillo* is an N1 phrase containing the lexical signs 'young' and 'bull'. Its simplified TFS is shown below:

$$
\left[\begin{array}{ll}
\multicolumn{2}{l}{\textbf{phrasal-tlink}} \\
\textsc{sfs} = & \left[\begin{array}{ll}
\multicolumn{2}{l}{\textbf{2lexeme-phrase}} \\
0 = & \boxed{\textbf{n1-phrase}} \\
1 = & \boxed{1}\ \boxed{\textbf{`young'}} \\
2 = & \boxed{2}\ \boxed{\textbf{`bull'}} \\
\textsc{inc}1 = & \left[\begin{array}{ll}
\multicolumn{2}{l}{\textbf{identity-rule}} \\
0 = & \boxed{1} \\
1 = & \boxed{1}
\end{array}\right] \\
\textsc{inc}2 = & \left[\begin{array}{ll}
\multicolumn{2}{l}{\textbf{identity-rule}} \\
0 = & \boxed{2} \\
1 = & \boxed{2}
\end{array}\right]
\end{array}\right] \\
\textsc{tfs} = & \left[\begin{array}{ll}
\multicolumn{2}{l}{\textbf{indentity-rule}} \\
0 = & \boxed{0}\ \boxed{\textbf{`novillo'}} \\
1 = & \boxed{0}
\end{array}\right]
\end{array}\right]
$$

Features INC1 and INC2 indicate that translation involves finding these lexical signs (or more precisely the output of these lexical rules) in the SL such that they form part of a phrasal sign of type **n1-phrase**. For this example these features contain no contentful lexical rules, but for example in the equivalence 'washing machine - *lavadora*', one can imagine that the gerundive rule is applied to 'wash' before it is used to translate *lavadora*. How such *tlinks* are used during translation will be explained in Section 4.1.

Just as lexical rules capture relations in the monolingual lexicon, so *tlink-rules* capture relations within the bilexicon. A *tlink-rule* derives a new *tlink* from an existing one in order to create a new translation relation. For example, there is a relationship between the translation of fruits and the translation of their trees in the English-Spanish bilexicon, as exemplified by the table below:

Fruit		**Tree**	
English	*Spanish*	*English*	*Spanish*
almond	almendr**a**	almond	almendr**o**
apple	manzan**a**	apple	manzan**o**
lemon	limón	cherry	limon**ero**

An English-Spanish fruit translation is matched by a tree translation in which the name of the fruit in Spanish changes morphologically. Such a relationship may be captured through a *tlink-rule* with the following schematic structure:

$$\begin{array}{lcl} \text{Ne} & \leftrightarrow & \text{Ns} \\ \Downarrow \text{e-fruit-tree} & & \Downarrow \text{s-fruit-tree} \\ \text{Ne}' & \leftrightarrow & \text{Ns}' \end{array}$$

Where, ‘Ne’ and ‘Ns’ are English and Spanish nouns respectively. This *tlink-rule* would effect the following mapping:

$$\begin{array}{lcl} \text{apple} & \leftrightarrow & \text{manzan}\mathbf{a} \\ \Downarrow \text{e-fruit-tree} & & \Downarrow \text{s-fruit-tree} \\ \text{apple} & \leftrightarrow & \text{manzan}\mathbf{o} \end{array}$$

The simplified TFS for this *tlink-rule* is:

$$\left[\begin{array}{ll} \textbf{tlink-rule} & \\ \text{T0} = & \left[\begin{array}{ll} \textbf{simple-tlink} & \\ \text{SFS} = & \left[\begin{array}{ll} \textbf{identity-rule} & \\ 1 = & \boxed{0}\,\boxed{\textbf{n-com}} \\ 0 = & \boxed{0} \end{array}\right] \\ \text{TFS} = & \left[\begin{array}{ll} \textbf{identity-rule} & \\ 1 = & \boxed{1}\,\boxed{\textbf{n-com}} \\ 0 = & \boxed{1} \end{array}\right] \end{array}\right] \\ \text{T1} = & \left[\begin{array}{ll} \textbf{simple-tlink} & \\ \text{SFS} = & \left[\begin{array}{ll} \textbf{identity-rule} & \\ 1 = & \boxed{2}\,\boxed{\textbf{n-com}} \\ 0 = & \boxed{2} \end{array}\right] \\ \text{TFS} = & \left[\begin{array}{ll} \textbf{identity-rule} & \\ 1 = & \boxed{3}\,\boxed{\textbf{n-com}} \\ 0 = & \boxed{3} \end{array}\right] \end{array}\right] \\ \text{SRULE} = & \left[\begin{array}{ll} \textbf{e-fruit-tree} & \\ 1 = & \boxed{0} \\ 0 = & \boxed{2} \end{array}\right] \\ \text{TRULE} = & \left[\begin{array}{ll} \textbf{s-fruit-tree} & \\ 1 = & \boxed{1} \\ 0 = & \boxed{3} \end{array}\right] \end{array}\right]$$

Features SRULE and TRULE contain the English and Spanish lexical rules which derive tree denoting nouns from fruit denoting ones.

The above mechanisms are intended as generic translation relations which can be used in a variety of transfer MT systems. A brief review of representative transfer systems now follows; for a more extensive discussion the reader is referred to the original sources.

3 Transfer Machine Translation

MT systems have traditionally been classified as either interlingua or transfer (Tucker 1987)[3]. Interlingua systems construct a language independent representation of a sentence from which generation into any TL takes place. For example, Dorr (1994) analyses sentences into lexical conceptual structures (Jackendoff 1990) from which translation into any TL is carried out. Transfer systems include a bilingual component which maps SL representations into TL ones from which generation takes place.

3.1 Interlingua vs Transfer

The interlingua approach is very attractive from the point of view of modularity and extendibility. Unfortunately, there is at present no general mechanism which will map

[3] But see Kay et al. (1994) for arguments against this distinction.

interlingua structures constructed from some arbitrary sentence in one language into sentences in another language. Part of the reason stems from what Shieber (1993) calls the logical equivalence problem: currently, there is no meaning representation formalism nor terminating algorithm which will determine whether two sentences mean the same, where the standard of meaning equivalence is what a native speaker would judge to be equivalent meanings. This argument may be extended to the problem of Interlingua translation in the following manner. We consider two approaches to using an Interlingua for MT: one assumes a unique representation common to SL and TL, while the other allows the Interlingua produced by one language to be mapped into a more appropriate Interlingua for the TL.

The first approach requires analysers and generators which will construct identical representations for translationally equivalent sentences. If such devices were constructed, however, the logical equivalence problem as formulated by Shieber would be solved, since there would be a mechanism for establishing the meaning equivalence of two sentences. This is because, assuming that translation is meaning preserving, and that the assumed Interlingua translator constructed a finite number of translations, meaning equivalence could be established by first translating one of the two sentences into TL sentences, then translating these sentences back into the SL, and finally checking that the other sentence was among these SL sentences. Although Shieber does not prove that meaning equivalence cannot be established nor that the logical equivalence problem cannot be solved, he highlights the difficulties of the problem, noting for instance that first order logic and other more powerful formalisms are undecidable with respect to equivalence. Obviously this does not prove that an Interlingua cannot be built either; we present the argument as our justification for avoiding a commitment to a particular Interlingua, and for our assumption that if an Interlingua for MT is to be built, cross-linguistic studies of the type necessary for transfer-based MT are at least a reasonable methodology to adopt at present.

Under the second approach to Interlingua MT, an algorithm which will map between two representations is required. This algorithm, however, assumes that a representation exists which can be manipulated algorithmically in a meaning preserving way. While this is possible to a certain extent (consider conversion to and from logical normal forms), it has associated problems. For example, since one expects that in a multilingual system grammars will be written independently, it will be difficult to find a mapping which will derive an interlingua structure that ensured adequate TL generation; that is, the interlingua-to-interlingua transformation may not be general enough to be compatible with the input and output of all the possible grammars used in a multilingual system. Again, this does not prove that this type of interlingua approach is not achievable in practice; for example, careful grammar writing and some degree of coordination between language modules could force adequate transformations. Yet, this would have a negative effect on modularity and development which should perhaps be avoided.

From a more practical point of view, many interlingua systems are tailored to specific languages and it is not clear whether the relevant Interlingua would be adequate for describing sentences in a language not envisaged during interlingua construction.

While not disputing the many disadvantages of transfer systems such as multiplicity of transfer components, expensive addition of new languages and, like Interlingua systems, inability to guarantee meaning equivalence between SL and TL, we have adopted a transfer approach because it is not committed to a language independent representation and therefore permits a more flexible development strategy.

3.2 Transfer Systems

One may distinguish three types of transfer system: syntactic, semantic and mixed. The principal operations in syntactic transfer systems are tree-to-tree transformations which convert SL syntactic structures into TL ones. Semantic transfer systems construct a representation of the meaning of a sentence which is nevertheless dependent on the SL. This representation may take the form of a predicate-argument structure or some other formalized representation of meaning. Transfer in such systems involves principally predicate translation, although structural transfer rules are also used to overcome discrepancies in SL and TL semantic structures. In mixed transfer systems, syntactic, functional, semantic and sometimes pragmatic information is brought to bear on the expression of transfer relations. By using multiple levels of information, mixed transfer systems can encode translation equivalences at the level which is most appropriate to the languages and constructions at hand. For instance, the translation of a passive sentence into a language which has passives can best be effected at the functional level rather than at the predicate-argument level in order to preserve grammatical structure.

3.3 Metal

The Metal MT system is described by Bennett and Slocum (1988), Schneider (1991), Hutchins and Somers (1992:260-74) and Fontenelle et al. (1994). It is a syntactic transfer system in which translation follows the sequence: morphological analysis; parsing; transfer including lexical and structural or tree-to-tree rules, and synthesis. Parsing is done with a context-free (CF) grammar augmented with tests on features and with transformations. Transfer rules convert SL syntactic trees and transform them, in conjunction with the bilexicon, into trees appropriate for the TL generator. Generation involves further structural transformations to construct a valid TL structure. There are many other features of Metal which we will not consider in this brief description.

To elaborate on the translation procedure, consider the translation of the sentence below into German (Hutchins and Somers 1992:273-74):

It will have been tested.

After parsing, a structure equivalent to that shown on the left in Figure 1 is constructed. One important feature of this tree is that any structural transfer rules that need to be applied during transfer are appended to the node to which they apply. These rules are specified for each grammar rule in the SL grammar and are added to the analysis tree as it is constructed. Thus, for node PRED the leftmost daughter will become rightmost after transfer. Application of the transfer routine to this parse tree leads to the structure on the right in which lexical and structural transfer has taken place. The transfer routine uses the bilexicon in conjunction with the structural transformations attached to the nodes to derive the input to the generator. Transfer proceeds from the top node downwards, applying transfer rules as they are found and effecting lexical transfer on the leaves of the tree when they are reached.

Local rearrangement of constituents is not the only type of transfer rule; more complex structural transformations are allowed although for this particular example complex transformations take place during generation only, as shown by the intermediate and final structures in Figure 2. During generation the TL tree can undergo further transformations in order to construct an appropriate TL sentence. Morphological synthesis then takes place from the resulting tree.

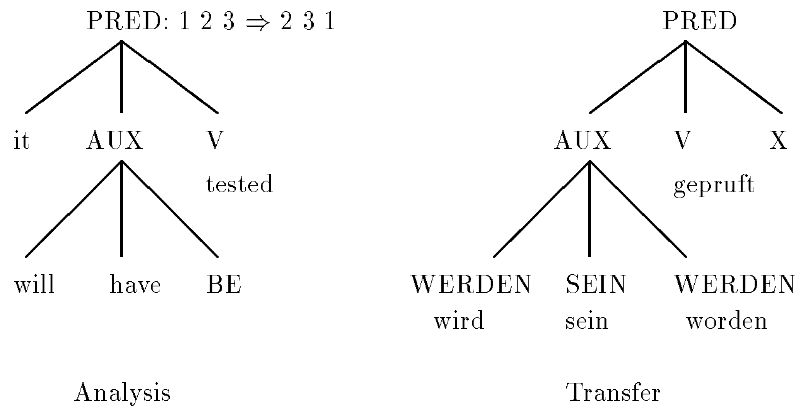


Figure 1: Result of parsing in Metal.

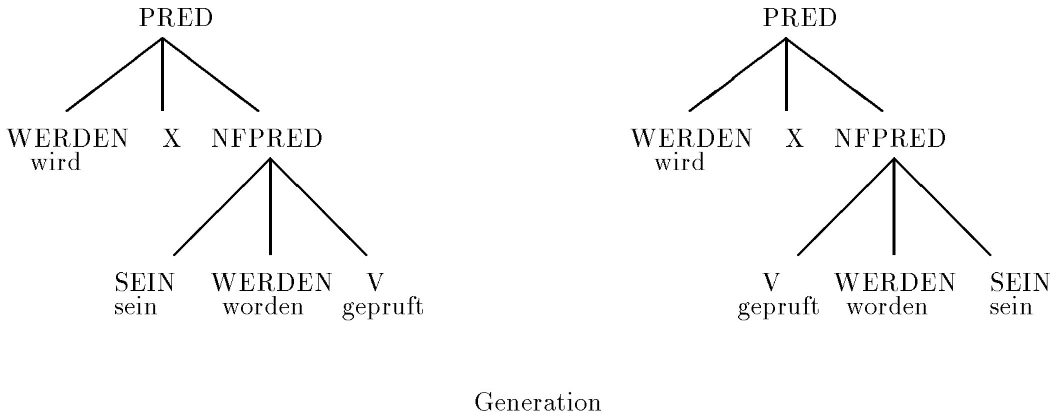


Figure 2: Generation with restructuring in Metal.

3.3.1 Comment

One problem with Metal is that it is not bidirectional and therefore different transfer modules have to be written for different language pairs. This leads to redundancy and possible inconsistency. Furthermore, transfer operations are unlikely to be reversible even in principle as they are underspecific on the conditions and restrictions on the TL output.

Transfer and generation routines in Metal are procedural, which means complex interactions between the bilexicon, the structural transfer rules and the rules for generation; in turn, this leads to difficulties in the debugging, maintenance and expandability of the system.

Another problem arises with the transformations allowed during generation. These are designed to convert the output of transfer into a correct TL tree. Unfortunately this technique cannot lead to sufficient modularity since appropriate generation transformations will have to be specified for different SLs, making the size and complexity of each generator grow considerably with the addition of a new SL.

There is also the problem of monolingual reversibility: the grammars used for analysis are different from the transformations used for generation. This creates an enormous amount of overlap in information content between analysis and generation rules. Although the Metal developers claim that the generation rules can cover a narrower set of expressions than the analysis grammar, one should aim at using reversible grammars whenever possible (Isabelle 1989).

Finally, it is worth noting that the preceding description does not emphasize sufficiently the lexical component in Metal. As Magnúsdóttir (1993) points out, Metal's extensive lexical encoding is a very important feature of its operation, thus highlighting the importance of the lexicon in MT.

3.4 Bilingual Conversation Interpreter

An example of a semantic transfer system is the Bilingual Conversation Interpreter (BCI) described by Alshawi et al. (1992). This is an application of the CLE (Core Language Engine) for enabling communication between two speakers using typed text. The CLE has evolved in a number of ways (Agnäs et al. 1994; Rayner et al. 1993) but transfer continues to be effected at the semantic representation level.

Transfer in the BCI is from quasi-logical form (QLF), which is a semantic representation derived purely from the syntactic structure of a sentence. That is, in QLF quantified expressions are not assigned scope, anaphoric terms are not resolved, and implicit relations are not disambiguated. One novel aspect of this representation is that it contains grammatical information such as number and syntactic category.

Translation begins by parsing the SL sentence in order to derive its QLF. This source QLF is converted into a TL QLF using transfer rules. The TL QLF then becomes the input to the generator and a TL sentence is produced. Transfer rules in the BCI take the following form:

```
trans(<QLF pattern 1>   <Operator>   <QLF pattern 2>)
```

A transfer rule relates portions of the source and target QLF: the patterns identifying these portions may be atoms, predicate structures or other more complex formulae. Although most transfer rules are bidirectional, it is possible to restrict their application to one direction only by using one of the operators `=>` or `<=`; bidirectionality is represented by the operator `<=>`. During transfer, the SL QLF is traversed recursively from the top downwards, following its recursive structure. At each stage the SL side of every transfer rule is compared against the current portion of the SL QLF; if there is a match, the TL side of the rule is used to build the target QLF. Recursion is introduced in a rule by using transfer variables. For example, in the transfer rule below, the two transfer variables `tr(cat)` and `tr(rest)` recursively invoke the transfer algorithm with whatever appears in their place in the SL QLF:

```
trans(qterm(tr(cat),X,tr(rest))
      <=>
      qterm(tr(cat),X,tr(rest))).
```

This rule states that *qterms* (a *qterm* is part of the QLF syntax and represents an unscoped variable plus restrictions) translate as *qterms* and that their first and third argument should be translated by invoking the transfer rules on their value.

The following is an example of transfer at QLF for the translation of Spanish *el gatito duerme* to 'the little cat sleeps'. The Spanish QLF is:

```
[dcl,
    [duerme1,
      qterm(<t=quant,n=sing,l=ex>,E,
            [and,[event,E],
                  a_form(<tense=pres,aspect=imperf>,P,[P,E])])
      qterm(<t=ref,p=def,l=el,n=sing>,N,[gatito1,N])]]
```

The interpretation of this QLF is as follows: `dcl` indicates the declarative mood of the sentence; `qterm` introduces the event and noun variables which are arguments to the predicate `duerme1`. The first `qterm` states that the event should be bound by an existential quantifier; the event variable is `E` and it is restricted by the predicates `event` and `P`. The latter predicate is part of an anaphoric formula which has been left unresolved as this is deemed unnecessary for translation applications; it represents unresolved tense and aspect information as derived from the syntactic structure of the sentence. The second `qterm` identifies the noun variable `N` introduced by the referring, definite, singular determiner `el` and restricted by the predicate `gatito1`.

The following transfer rules would be necessary for translating this QLF into its equivalent English QLF:

```
trans([dcl, tr(for)]                  <=> [dcl, tr(for)])                  (1)
trans([duerme1, tr(arg1), tr(arg2)]) <=> [sleep1, tr(arg1), tr(arg2)]     (2)
trans(qterm(tr(cat),X,tr(rest))      <=> qterm(tr(cat),X,tr(rest)))       (3)
trans(<t=quant,n=sing,l=ex>          <=> <t=quant,n=sing,l=ex>)           (4)
trans([and, tr(first), tr(rest)]     <=> [and, tr(first), tr(rest)])      (5)
trans([event,X]                      <=> [event,X])                       (6)
trans(a_form(tr(cat),X,tr(rest))     <=> a_form(tr(cat),X,tr(rest)))      (7)
trans(<tense=pres,aspect=imperf>     <=> <tense=pres,aspect=imperf>)      (8)
trans([P,X]                          <=> [P,X])                           (9)
trans(<t=ref,p=def,l=el,n=sing>      <=> <t=ref,p=def,l=the,n=sing>)     (10)
trans([gatito1, X]                   <=> [and,[little1,X],[cat1,X]])     (11)
```

The first transfer rule that matches is (1) recursively invoking the transfer procedure on the formula covered by `tr(for)`; rule (2) matches then, invoking again all the transfer rules on the arguments of `duerme1`, and so on. The resulting English QLF is:

```
[dcl,
  [sleep1,
      qterm(<t=quant,n=sing,l=ex>,E,
            [and,[event,E],
                  a_form(<tense=pres,aspect=imperf>,P,[P,E])]])
      qterm(<t=ref,p=def,l=the,n=sing>,N,[and,[little1,N],[cat1,N]])]]
```

Generation of the TL sentence from this QLF is done by using an adaptation of the algorithm of Shieber et al. (1990). Note that there is no guarantee that the TL QLF corresponds to a QLF derivable from the TL grammar. In some instances this is done deliberately in order to simplify the transfer component; what happens in such cases is that a number of TL QLFs are produced during transfer and are filtered by the TL grammar. In doing this, modularity is enhanced since the translation of a fragment of QLF might depend on information solely present in the TL grammar. For example, in English-Swedish translation the article 'the' can translate as nothing or as *den/det*, unless it comes before an adjective in which case it must translate as an article. To incorporate such information into the transfer rule for 'the' would be difficult and it would diminish the modularity of the system; therefore, transfer produces QLFs with and without articles and it is left to the generation step to select between them.

During transfer-based translation ambiguity may be present during analysis, transfer and generation. In one version of the BCI, disambiguation during analysis is done by interacting with the SL user in order to resolve structural and word sense ambiguity. In the speech translation application of the CLE translation engine, disambiguation is done stochastically.

During transfer, ambiguity is tackled by using unidirectional transfer rules such as that shown below on the right:

```
trans(alrededor_de <=> around)   trans(alrededor_de <= round)
```

or by introducing pseudo-words like `clock/watch` to translate their single Spanish equivalent `reloj`.

Much of the ambiguity encountered during generation in the BCI consists of stylistic variants with the same or almost the same meaning; this is because, generally, sentences deriving from the same QLF will have equivalent meanings. In the BCI this type of ambiguity is resolved by presenting the first sentence produced by the generator.

3.4.1 Comment

One minor problem with the BCI is that there is no apparent way of encoding relationships in the bilexicon relating entries such as 'apple - *manzan*a' to 'apple (tree) - *manzan*o', as described in Section 2. Remedying this problem should be relatively straightforward, however, through the implementation of a rule mechanism for deriving new predicate translations from existing translations. The only possible obstacle might be integrating such a mechanism with whatever lexical or morphological rules are necessary for analysing Spanish *manzan*a and *manzan*o.

Another difficulty is caused by discontinuous lexical relations. Take the following simplified (i.e. ignoring the non-intersective nature of 'young') QLF representation of the NP 'young English bull':

```
[and, [young,X], [and, [english,X], [bull,X]]]
```

Translation of this phrase into Spanish would require the pairing 'young' and 'bull' in one QLF translation rule to give *novillo*, and a different rule for 'English' to give *inglés*:

```
trans([and, [young,X], [bull,X]] <=> [novillo,X])
trans(english <=> inglés)
```

However, these rules would not work in the appropriate way for the above example since the predicate 'English' stands between 'young' and 'bull'. The problem arises because the recursive transfer algorithm of the BCI cannot easily cope with discontinuities in the transfer structure. To cope with this translation, it would be necessary to serially traverse the leaves of the QLF, matching portions of transfer rules against the source QLF.

While the compositionality of the transfer component in the BCI is adequate for an ample range of constructions (Gambäck et al. 1991), Bretan and Collins (1994) note that special mechanisms are necessary to handle phenomena such as head switching and discrepancies in the type of control verb required by the source and target language. Most of these problems arise partly because of the recursive descent nature of the transfer algorithm.

Finally, the addition of new constructions to the BCI monolingual grammars and transfer components can be expensive in the sense that they are difficult to acquire automatically. This is because the semantic rules which build the QLF structures at which transfer takes place need to be added manually in order to take into account their import into the subsequent computation of the truth-conditions of the sentence. Clearly this problem affects all present systems whose analyses are required to have a formal semantic interpretation. From our point of view a semantic representation whose construction can be automated is highly desirable, and in the absence of such a representation, an alternative to QLFs will be sought.

3.5 Structural Correspondences

As an example of a mixed transfer approach one may cite the proposal of Kaplan et al. (1989) and Kaplan and Wedekind (1993) in which transfer is seen as a set of relations expressed as structural correspondences between different levels of linguistic description in the theory of Lexical-Functional Grammar (LFG). They distinguish this view from other approaches to transfer in which a single level of representation is unnaturally required to hold all information necessary for transfer. LFG as a theory of language is described by Kaplan and Bresnan (1982); below we only consider its application to MT.

In the conception of Kaplan et al. (1989) the structural correspondence approach to transfer may be represented by Figure 3. This diagram indicates that transfer mappings are

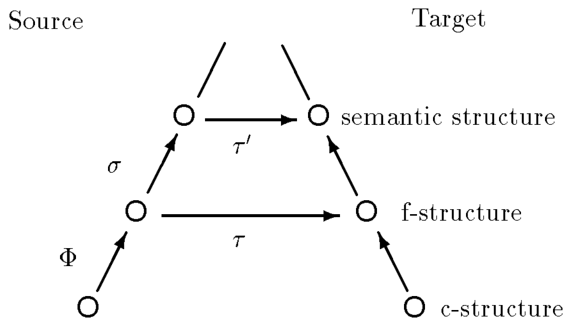


Figure 3: Structural correspondences at different linguistic levels.

established between f-structures (functional structures describing grammatical relations) and semantic structures, as indicated by the arrows.

To see how different levels are appropriate for different translation problems, we will present two examples, one where transfer takes place at the f-structure level and the other where it is mostly done at the semantic structure level. For the first example, consider the sentences:

Eng: John saw Mary.
Spa: Juan vio a María.

Transfer at a more abstract level than f-structure would lead to undue ambiguity during generation because important surface information would have been lost. Therefore the transfer step maps between the f-structures of the two sentences. The English f-structure is as follows:

$$\begin{bmatrix} \text{PRED} = \textbf{`saw[}\uparrow \text{SUBJ},\uparrow \text{OBJ]'} \\ \text{SUBJ} = \textbf{John} \\ \text{OBJ} = \textbf{Mary} \end{bmatrix}$$

while the target f-structure to be constructed by transfer is:

$$\begin{bmatrix} \text{PRED} = \textbf{`vio[}\uparrow \text{SUBJ},\uparrow \text{AOBJ]'} \\ \text{SUBJ} = \textbf{Juan} \\ \text{AOBJ} = \begin{bmatrix} \text{PRED} = \textbf{`a[}\uparrow \text{OBJ]'} \\ \text{OBJ} = \textbf{María} \end{bmatrix} \end{bmatrix}$$

To obtain this structure, the following τ mappings in the bilexicon entry for 'saw' would be necessary (direct equivalences for 'John' and 'Mary' are also assumed).

$(\tau \uparrow)$ PRED = vio
$(\tau \uparrow)$ SUBJ = $\tau(\uparrow$ SUBJ$)$
$(\tau \uparrow)$ AOBJ OBJ = $\tau(\uparrow$ OBJ$)$

Equation $(\tau\ \uparrow)$ SUBJ $= \tau(\uparrow$ SUBJ$)$, for example, indicates that the subject of the TL structure (lhs) is set to the translation of the SL subject (rhs). During transfer, the above equations, in conjunction with the bilingual entries for 'John' and 'Mary' and the equations in the Spanish lexical entry for *vio*, would be solved by a process of step-wise constraint resolution (Kaplan and Bresnan 1982) to find the smallest f-structure which satisfied all the equations; the result is a TL f-structure from which generation can take place.

Transfer in which mapping at the level of semantic structure is required proceeds in a similar fashion. For example, consider the translation pair:

Eng: I think that the baby just fell.
Spa: Yo creo que el bebé acaba de caerse.

Sadler and Thompson (1991) show that structural correspondences at the level of f-structure are not satisfactory for this type of sentence because they have subordinate clauses with different heads: in the English case the head is 'fell' whereas in Spanish it is the translation of 'just' namely the verb *acaba de*. The problem arises when having to provide, at f-structure, a compositional translation of 'just fell' which will embed the translation of 'fell' into that of 'just'. In the most adequate grammatical analysis of this phrase, 'just' acts as an adjunct to 'fell'; given this, the transfer of 'just' as the syntactic head of the Spanish sentence does not lead to correct TL f-structures. For example, from the highly simplified SL f-structure shown below:

$$\left[\begin{array}{lcl} \text{PRED} & = & \textbf{think} \\ \text{COMP} & = & \left[\begin{array}{lcl} \text{PRED} & = & \textbf{fell} \\ \text{SUBJ} & = & \textbf{baby} \\ \text{SADJ} & = & \textbf{just} \end{array}\right] \end{array}\right]$$

two mappings are necessary: 1) the translation of 'think' requires its complement to be the translation of 'the baby *just* fell' (i.e. the value of COMP); 2) the translation of 'just' requires its complement to be the translation of 'the baby fell' (i.e. part of the value of COMP). To achieve both of these goals, Kaplan et al. (1989) formulate the equations

(1) VP $\longrightarrow$	Vsent	S	
	$(\tau\ \uparrow)$ COMP $= \tau(\uparrow$ COMP$)$		
(2) S $\longrightarrow$	NP	ADVP	VP
		$(\tau\ \uparrow) = \tau(\uparrow$ SADJ$)$ XCOMP	

The first equation is the expected compositional transfer of the complement of a verb. Equation (2) however, creates a structure in which the translation of the adverb is no longer part of the translation of the sentence but instead is treated as a predicate over the translation of the sentence. The combination of these and other relevant τ equations leads to a doubly rooted graph because both (1) and (2) construct topmost feature structures as shown below (note that in this case reentrancy is intended between otherwise unrelated structures):

$$\left[\begin{array}{lcl} \text{PRED} & = & \textbf{`pienso}[\uparrow \text{SUBJ}, \uparrow \text{COMP}]\text{'} \\ \text{SUBJ} & = & \textbf{yo} \\ \text{COMP} & = & \boxed{0} \left[\begin{array}{lcl} \text{PRED} & = & \textbf{`caerse}[\uparrow \text{PRED}]\text{'} \\ \text{PRED} & = & \boxed{1}\, \textbf{bebé} \end{array}\right] \end{array}\right] \quad \left[\begin{array}{lcl} \text{PRED} & = & \textbf{`acaba\_de}[\uparrow \text{XCOMP}]\uparrow \text{SUBJ'} \\ \text{SUBJ} & = & \boxed{1} \\ \text{XCOMP} & = & \boxed{0} \end{array}\right]$$

Clearly this is incorrect since it does not represent a valid f-structure.

The semantic solution to this problem is described by Kaplan and Wedekind (1993) and consists in assigning the following semantic representation to the English sentence:

$$\left[\begin{array}{lcl} \text{REL} & = & \textbf{think} \\ \text{ARG1} & = & \textbf{I} \\ \text{ARG2} & = & \left[\begin{array}{lcl} \text{REL} & = & \textbf{just} \\ \text{ARG1} & = & \left[\begin{array}{lcl} \text{REL} & = & \textbf{fall} \\ \text{ARG1} & = & \textbf{baby} \end{array}\right] \end{array}\right] \end{array}\right]$$

This semantic structure is isomorphic to its Spanish counterpart and therefore no major difficulties arise during transfer. However, analysis is complicated since part of the semantic mapping involves deriving a hierarchical structure from a linear one, as shown below:

$$\begin{bmatrix} \text{PRED} = \textbf{fell} \\ \text{SUBJ} = \textbf{baby} \\ \text{SADJ} = \textbf{just} \end{bmatrix} \xrightarrow{\sigma} \begin{bmatrix} \text{REL} = \textbf{just} \\ \text{ARG1} = \begin{bmatrix} \text{REL} = \textbf{fall} \\ \text{ARG1} = \textbf{baby} \end{bmatrix} \end{bmatrix}$$

The details of the solution need not concern us, save that they involve the introduction of a restriction operation which deletes information from a feature structure. For instance, applying restriction on the feature SADJ is depicted below:

$$\begin{bmatrix} \text{PRED} = \textbf{fell} \\ \text{SUBJ} = \textbf{baby} \\ \text{SADJ} = \textbf{just} \end{bmatrix} \overset{\textbf{restriction}}{\Longrightarrow} \begin{bmatrix} \text{PRED} = \textbf{fell} \\ \text{SUBJ} = \textbf{baby} \end{bmatrix}$$

Using this restriction, the adverb is first made the head of the main structure and then the restricted feature structure is made its complement.

3.5.1 Comment

Although the above approach to transfer is attractive, it has a number of shortcomings. One problem is the complete dependence on specific syntactic and semantic analyses for both SL and TL structures. There are two main problems with this. Firstly, each language in the system needs to adopt the same formalism (e.g. LFG); unfortunately there are many linguistic theories at present and it is not yet clear which one is the best for describing all languages. Secondly, commitment to a particular theory implies that changes in the treatment of different monolingual phenomena will require changes to the transfer component, thus diminishing the modularity of the system. For example, Kaplan et al. (1989) proposed transfer of head switching constructions at f-structure level. However, in Kaplan and Wedekind (1993) they prefer to effect transfer at the semantic level by augmenting their semantic interpretation function (see above). In a multilingual system such a change could induce modifications in all the transfer components in which English was involved.

Although the flexibility of specifying transfer relations at different levels may be desirable, it can also lead to compromises having to be made. Put differently, the question is, why not effect all transfer at the semantic level? One could argue that this level would not preserve grammatical information useful for translation between related languages, but equally, one could say that this grammatical information is the result of more abstract levels of interpretation which could be used for translation. For instance, f-structure may be argued to be the best level for translating passives between English and Spanish. However, the use of passives is the result of factors such as the informational structure of the text, its style and the formal devices available in the language. Therefore it would be possible in principle to preserve passive constructions even when transfer was at an abstract level of representation simply by encoding the relevant factors which determined their use in the semantic representation. For this reason, the expression of transfer relations at f-structure may be seen as a pragmatic decision whose usefulness depends on the level of abstraction available in the grammatical theory adopted. This means that the transfer component will have to be modified considerably in the face of developments in syntactic and semantic theory.

A different issue regarding mixed transfer approaches is that phenomena as pervasive in translation as lexical gaps, (semi-)idiomatic translations and differences in lexicalization patterns are not naturally handled by the above approach. For instance, transfer relations of the type 'commit suicide - *suicidarse*' are considered by Sadler et al. (1990); they propose a separate transfer lexicon in which this type of correspondence would be stored and which would override other applicable transfer relations. Theoretically, this points to a shortcoming in the approach since special mechanisms have to be included to cope with such prevalent transfer phenomena. Another difficulty with a separate lexicon of this sort is that it cannot encode regularities which exist between different types of lexical transfer relations. For example, patterns such as 'apple - *manzan***a**' and 'apple (tree) - *manzan***o**' are not uncommon, and these would be overlooked in the special bilexicon suggested by Sadler et al. (1990).

3.6 Conclusion

All the transfer approaches described above rely on the structure constructed by the analysis component. While such structures make explicit functional and semantic relationships relevant to translation they also introduce problems for the transfer procedure. Thus, syntactic transfer requires tree-to-tree transformations whose interaction is complex. Semantic transfer with recursive descent algorithms do not cope easily with structural discontinuities and mismatches in transfer relations. Mixed transfer approaches have dependent analysis, transfer and generation components which make them difficult to maintain. Below we present lexicalist machine translation as an alternative to transfer systems of the above type.

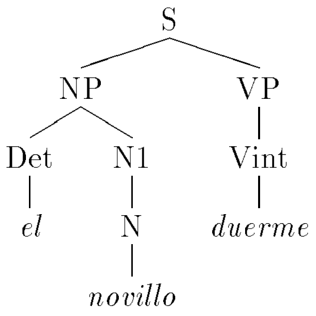
4 Lexicalist Machine Translation

In this section we consider two approaches to translation which directly relate to our translation mechanism, and follow this by a description of our approach. The two approaches are presented here because of their relevance to our system.

4.1 Constraint Resolution MT

The first mechanism to consider is implemented in the LKB and is described by Copestake (1993) and Copestake and Sanfilippo (1993); the approach may be understood as a combination of ideas from Shake-and-Bake (Whitelock 1992) and Type Rewriting (Zajac 1989; Emele et al. 1992) translation. The intuitive idea is that the transfer algorithm, in conjunction with the *tlinks*, constructs a set of TL lexical and phrasal signs which are used to constrain the generation phase. These signs are derived from the target side of a *tlink* based on those *tlinks* whose source side unifies with some part of the analysed SL sentence. The generation step applies constraint resolution to these signs, using the TL grammar to guarantee a grammatical sentence. Constraint resolution is the process by which a TFS is specialized to one or more TFSs with all their types being leaf types (i.e. maximally specific).

The following graphic example will clarify the constraint resolution approach to MT. Consider translating *el novillo duerme* into English 'the young bull sleeps'. The analysis tree will be:



Possible *tlinks* to translate this sentence may be:

el	⇔	the
novillo	⇔	N1 AP N1 {young, bull}
duerme	⇔	sleeps

Note that this time, following Pollard and Sag (1994), the phrasal sign (i.e. N1 in the *tlink* for *novillo*) is depicted as incorporating daughter information. While this is not necessary for the approach of Copestake (1993), it is used there to simplify the integration of HPSG into the system.

During transfer, the source side of these *tlinks* will match the leaves of the source tree; the result of this phase is the following bag, which contains a phrasal category N1:

$$\{\ \text{the},\ \underset{\text{AP}\quad\text{N1}}{\text{N1}},\ \text{young},\ \text{bull},\ \text{sleeps}\ \}$$

Further restrictions will also indicate that the 'N1' phrase must contain both 'young' and 'bull' within its leaves, and that the semantic structure of this phrase be instantiated to the semantics of 'young bull'. Generation proceeds through constraint resolution, where the input is an unresolved sentential sign together with the bag of constraints, and the output is a fully resolved sign with its orthography being the translation of the SL sentence. For the present example, resolution starts with the sign S and proceeds through the steps shown in Figure 4, presented deterministically for brevity. At each step the current sign is further resolved either by using the grammar rules or by using the phrasal (shown in bold in Figure 4) and lexical signs in the TL bag. For example, a fuller depiction of S above is:

$$\begin{bmatrix} \textbf{sent-sign} \\ \textsc{orth} = \textbf{orth} \\ \textsc{syn} = \boxed{\textbf{syn}} \\ \textsc{daughters} = \textbf{rule} \end{bmatrix}$$

Further resolution of **rule** gives, among other signs:

$$\begin{bmatrix} \textbf{sent-sign} & \\ \textsc{orth} = \textbf{orth} & \\ \textsc{syn} = \boxed{\textbf{syn}} & \\ \textsc{daughters} = & \begin{bmatrix} \textbf{head-comp-struct} \\ \textsc{head} = \boxed{\textbf{vp-sign}} \\ \textsc{comps} = \boxed{\textbf{np-sign}} \end{bmatrix} \end{bmatrix}$$

Each of the daughter signs is further resolved into more specific types until a complete sentence using all the signs in the TL bag is constructed. Each resolution step searches

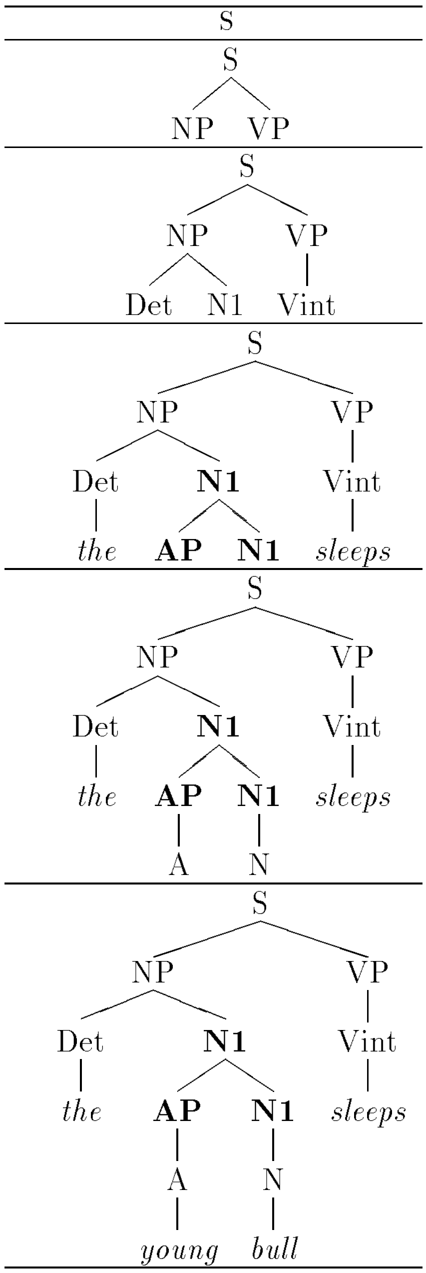


Figure 4: Constraint resolution generation.

the type hierarchy for all the subtypes of the type being resolved; the constraints of these types are unified with the current structure in order to produce a number of more specific TFSs which are further resolved until all types in the TFS are leaf types. At this point, the orthography of the TFS will be fully instantiated because of the various constraints on phrasal and daughter signs.

A special case of the above technique is one where no phrasal signs are present in the *tlinks*. In this case, the only phrasal constraints used during generation are those found in the TL grammar. The main advantage then is a reduction in complexity since the generator needs to consider less phrasal types during the type resolution phase; the disadvantage is that certain translation constraints not easily expressible through lexical equivalences are difficult to impose (e.g. noun-noun modification). We will consider such equivalences below.

Copestake (1993) further suggests that in cases where the source side of a *tlink* contains a phrasal sign, the phrasal component can be exploited during transfer. For example, to translate *novillo inglés* into English the transfer algorithm will traverse the SL analysis tree searching for a phrasal sign node which will match the N1 component of the respective *tlink*. Matching a phrasal *tlink* in this way involves unifying with the TFS at the respective node, and ensuring that any specified lexical signs are also found at the leaves of the corresponding subtree. If matching is successful the signs on the TL side of the *tlink* are included in the TL bag. Once again, it is possible to ignore these SL phrasal constraints, and simply match lexical signs; the trade off, between computational efficiency versus expressivity is the same as for the generation case and will be taken up later.

4.2 Shake-and-Bake Translation

The Shake-and-Bake (SB) approach to translation, as described by Whitelock (1992) and Beaven (1992), is in essence a special case of the previous technique where only lexical signs are used both during transfer and generation. In SB, contrastive knowledge is found in the bilexicon only, in the form of source and target bags (i.e. sets in which repeated elements are significant) of lexical signs standing in translation correspondence. Transfer maps a bag of source language (SL) lexical signs into a set of TL lexical signs. Generation is via a modified parser which effectively re-arranges the TL bag into a valid TL string (Brew 1992).

In SB, the analysis phase of translation constructs a logical formula from the SL sentence. In this formula, uninstantiated variables corresponding to arguments in predicates are assigned unique values; these instantiations permeate to the leaves of the analysis tree which in turn are used for transfer via the bilexicon. Since only the bilexicon is used, the result of transfer is a bag of TL lexical signs. Therefore, the task of the generator is to reorder this bag into a valid TL string. Generation is through a process similar to parsing but where the input is treated as a set rather than as a list, thus effectively ignoring the order of the input lexical signs.

The following example shows the steps needed for translating *un hombre duerme* into English. I will use an informal AVM notation which allows λ-functions (Dowty et al. 1981) to be encoded as the value of an attribute, with shared values between SL and TL signs represented by boxed integers. Assume the following bilexicon:

$$\begin{bmatrix} \text{ORTH} = \textbf{un} \\ \text{SYN} = \textbf{Det} \\ \text{SEM} = \lambda\mathbf{S}\lambda\mathbf{P}(\exists\boxed{0}.\ \mathbf{S}(\boxed{0})\ \&\ \mathbf{P}(\boxed{0})) \end{bmatrix} \iff \begin{bmatrix} \text{ORTH} = \textbf{a} \\ \text{SYN} = \textbf{Det} \\ \text{SEM} = \lambda\mathbf{S}\lambda\mathbf{P}(\exists\boxed{0}.\ \mathbf{S}(\boxed{0})\ \&\ \mathbf{P}(\boxed{0})) \end{bmatrix}$$

$$\begin{bmatrix} \text{ORTH} = \textbf{hombre} \\ \text{SYN} = \textbf{N} \\ \text{SEM} = \textbf{hombre}(\boxed{1}) \end{bmatrix} \iff \begin{bmatrix} \text{ORTH} = \textbf{man} \\ \text{SYN} = \textbf{N} \\ \text{SEM} = \textbf{man}(\boxed{1}) \end{bmatrix}$$

$$\begin{bmatrix} \text{ORTH} = \textbf{duerme} \\ \text{SYN} = \textbf{Vtr} \\ \text{SEM} = \textbf{dormir}(\boxed{2}) \end{bmatrix} \iff \begin{bmatrix} \text{ORTH} = \textbf{sleeps} \\ \text{SYN} = \textbf{Vtr} \\ \text{SEM} = \textbf{sleep}(\boxed{2}) \end{bmatrix}$$

Analysis of the SL sentence, using lexemes equivalent to the SL side of the bilexicon and an appropriate SL grammar, results in the structure:

$$\begin{bmatrix} \text{ORTH} = \textbf{un hombre duerme} \\ \text{SYN} = \textbf{S} \\ \text{SEM} = \exists\boxed{3}(\textbf{hombre}(\boxed{3})\ \&\ \textbf{duerme}(\boxed{3})) \end{bmatrix}$$

The next step in SB is to assign a unique constant to the shared variable $\boxed{3}$; this step may be thought of as assigning a unique range or type to the variable, effectively disallowing any additional bindings; for simple cases, this process may also be thought of as skolemization. It is essential that this assigned value permeates through the analysis tree down to the lexical signs at the leaves. Then, given the value sharings between SL and TL in the bilexical entries, unification of each SL leaf with the relevant SL side in the bilexicon will result in a unique constant also being assigned to the TL side. The result of these two operations is the set of bilingual entries shown below, where the unique constant is indicated by `0` (i.e. the reentrant structures below have this value).

$$\begin{bmatrix} \text{ORTH} = \textbf{un} \\ \text{SYN} = \textbf{Det} \\ \text{SEM} = \lambda\mathbf{S}\lambda\mathbf{P}(\exists\boxed{0}{:}0.\ \mathbf{S}(\boxed{0})\ \&\ \mathbf{P}(\boxed{0})) \end{bmatrix} \iff \begin{bmatrix} \text{ORTH} = \textbf{a} \\ \text{SYN} = \textbf{Det} \\ \text{SEM} = \lambda\mathbf{S}\lambda\mathbf{P}(\exists\boxed{0}.\ \mathbf{S}(\boxed{0})\ \&\ \mathbf{P}(\boxed{0})) \end{bmatrix}$$

$$\begin{bmatrix} \text{ORTH} = \textbf{hombre} \\ \text{SYN} = \textbf{N} \\ \text{SEM} = \textbf{hombre}(\boxed{1}{:}0) \end{bmatrix} \iff \begin{bmatrix} \text{ORTH} = \textbf{man} \\ \text{SYN} = \textbf{N} \\ \text{SEM} = \textbf{man}(\boxed{1}) \end{bmatrix}$$

$$\begin{bmatrix} \text{ORTH} = \textbf{duerme} \\ \text{SYN} = \textbf{Vtr} \\ \text{SEM} = \textbf{dormir}(\boxed{2}{:}0) \end{bmatrix} \iff \begin{bmatrix} \text{ORTH} = \textbf{sleeps} \\ \text{SYN} = \textbf{Vtr} \\ \text{SEM} = \textbf{sleep}(\boxed{2}) \end{bmatrix}$$

Generation using the TL side of these bilexical entries would take place by effectively attempting to parse all possible permutations of the TL lexemes. The result would be the structure:

$$\begin{bmatrix} \text{ORTH} = \textbf{a man sleeps} \\ \text{SYN} = \textbf{S} \\ \text{SEM} = \exists\boxed{4}{:}0.\ \textbf{man}(\boxed{4})\ \&\ \textbf{sleeps}(\boxed{4}) \end{bmatrix}$$

The transfer and generation algorithms in SB are similar to those considered in Section 5. Although the generation algorithm is NP-complete and therefore may not have a polynomial solution, some techniques will also be described there for alleviating this problem.

To appreciate the need for instantiating variables to unique constants, recall first that generation in SB effectively attempts every possible linear ordering of TL lexical entries. For instance, in a simplified Prolog-like notation one could express the result of parsing 'Mary hits John' as the bag of lexemes (upper case indicates variables):

[mary(X), hits(X,Y), john(Y)]

Without constant assignment the corresponding Spanish bag would give:

[mary(X), golpea_a(X,Y), john(Y)]

Now, note that variables X and Y can unify. Since SB attempts every possible ordering of the TL bag, it will successfully construct two analyses with the following predicates and argument values:

mary(X) golpea_a(X,Y) john(Y) ⇒ Mary golpea_a John
john(X) golpea_a(X,X) mary(X) ⇒ John golpea_a Mary

However, the second expression is incorrect. In the absence of case or thematic information, this expression will be incorrectly constructed as the translation of the English sentence. In many cases this problem can be overcome by instantiating variables in the semantic representation of a sentence. Continuing with our Prolog-like example, instantiation of the representation for the sentence above would be:

[mary(c1), golpea_a(c1,c2), john(c2)]
⇒ mary(c1) golpea_a(c1,c2) john(c2)
⇒ Mary golpea_a John

This ensures that value sharings not explicitly defined during analysis are not permitted during generation.

4.2.1 Comment

One advantage of SB is that it relates source and target structures via the bilexicon only, thus eliminating many problems arising from structural differences in the source and target language. For example, the 'likes - *graag*' case of head switching found in the translation of 'John likes swimming' into Dutch can be tackled using the following (simplified) bilingual entries (Whitelock 1992) and the SB generation algorithm[4]:

john(X) ⇔ jan(X)
enjoy(X,Y) ⇔ graag(X,Y)
swim(X,Y) ⇔ zwimt(X,Y)

Since no specific structural description of the SL or TL is assumed, maximum modularity is achieved between the monolingual and bilingual components.

Nonetheless, there remain certain issues to be addressed in the SB approach. The first one is the treatment of determiners within the semantic representation, and the way instantiations are performed on that representation. Apart from indefinite determiners which may be handled through existential quantifiers and skolemization, there are details relating to other determiners which are not made explicit in the available descriptions of SB. Rather than developing a full solution to this problem, we have adopted a simplified view of determiners whereby they are simply expressed as predicates over the noun variable they modify (thus effectively ignoring issues of scope).

The second issue is the status of function words. Whitelock (1992), following Calder et al. (1989), treats prepositions such as French *à* as having 'identity semantics' when used in expressions such as *plaît à* (likes): he includes them as part of the bilexicon and indexes them to the object of the verb. Nevertheless, it is not clear what the treatment of other function words such as relativiser 'that', pleonastic 'it' and auxiliary verbs should be.

The third issue is whether complex structural differences between languages can be handled through lexical transfer alone. Whitelock (1992) analyses the Dutch adverb *graag* as having two arguments in order to cope with head switching. However, it is necessary to show that other structural problems can be overcome through bilexical entries.

Since SB is not intended as a semantic theory but rather as an alternative paradigm for MT, it is somewhat unfair to raise the above issues. This is done, however, to highlight some of the areas in which our approach differs from the original SB formulation.

[4]Note that the adverb *graag* has to be made a two-place predicate.

5 Lexicalist MT in the LKB

The system described here follows SB in carrying out transfer through the bilexicon only. However, we allow *tlink-rules* to dynamically expand the bilexicon with new transfer knowledge. In addition, we represent determiners and syntactically dependent words as semantically empty predicates over the variable of their syntactic governor. Other changes have also been made to improve efficiency, such as interpreting bilexical entries as pairs of lists rather than bags, and generalising the constraint propagation technique of Brew (1992) to tackle the inefficiency problem in the generation step.

Translation starts by analysing the input sentence using the LKB chart parser. The output of the analysis module is a list of spanning edges from which instantiated lexical signs are extracted by reconstructing the parse tree, thus ensuring that the effect of unifications is percolated to the lexical leaves. The resulting list of instantiated lexical signs then becomes the input to the transfer module.

5.1 Lexicalist Transfer

The objective in lexicalist transfer, as understood here, is to exactly cover the SL list with the SL side of the bilexicon. By exact cover it is meant that every lexical sign in the input sentence must be part of some bilexical entry. Consider for example the following bilexicon (maximally simplified to ease description):

the	⇔	la
tall	⇔	alta
woman	⇔	mujer
wake up early	⇔	madrugar

The SL side of this bilexicon exactly covers the English sentence below:

The tall woman woke up early

The transfer algorithm starts by using the SL side of a bilexical entry to cover part of the input. If this is possible, the covered items in the input sentence are deleted and the process is called recursively on the remaining items. The result of transfer is a bag consisting of those TL items which must appear in a valid translation of the SL sentence. The TL bag for this example is shown below:

{la, alta, mujer, madrugó}

Generation from such a bag results in *la mujer alta madrugó*; this final phase will be considered in the next section.

Two modifications to the LKB have been necessary to implement the above transfer strategy. First, apart from expressing a relation between the output of lexical rules, *tlinks* are allowed to express a transfer relation between lists of lexical signs. That is, transfer via a *tlink* proceeds by unifying the lexical signs in the SL sentence with one side of the *tlink*, with the output being the list of lexical signs on the TL side. The implementation of this approach using TFSs has involved defining types with a feature for each lexical sign on either side of the input. For example, a modified *tlink* for representing the bilexical entry:

young bull ⇔ novillo

is shown below:

$$\left[\begin{array}{l}\textbf{2-1lexeme-tlink}\\ \textsc{sfs} = \left[\begin{array}{l}\textbf{2lexeme-side}\\ 1 = \boxed{\textbf{`young'}}\\ 2 = \boxed{\textbf{`bull'}}\end{array}\right]\\ \textsc{tfs} = \left[\begin{array}{l}\textbf{1lexeme-side}\\ 1 = \boxed{\textbf{`novillo'}}\end{array}\right]\end{array}\right]$$

The notation $\boxed{\textbf{`young'}}$ indicates that the lexical psort for 'young' has been shrunk. An alternative representation here would be to define *tlinks* as equating list types whose elements were lexemes or lexical rules. Since this would allow more flexibility in the number of lexical elements present on each side of the *tlink*, this possibility ought to be considered in more detail.

The other modification to the LKB is the identification of features indicating semantic variables. These are necessary for instantiating variables to unique constants in order to avoid invalid bindings during generation, as explained in Section 4.2. A list of these features (typically those indicating arguments in the TFS representation of a predicate) are sufficient for effecting the necessary bindings. An outline of the transfer algorithm is given in Figure 11 in the Appendix; the algorithm accepts as input a list of lexical signs and returns the list of all covers of the input as defined by the bilingual lexicon. A cover is a set of bilexical entries which effect one possible translation of the input. The input to the generation module is the union of the TL sides in the cover.

The approach just described does not incorporate phrasal constraints. The main reason for this is computational: as explained earlier, phrasal constraints add an extra level of non-determinism during transfer and generation. During transfer, the analysis tree has to be searched node by node in order to find a matching phrase; this would be in addition to the cover procedure for translating the lexical component of *tlinks*. During generation, the type resolution mechanism will consider constraints derived from the TL grammar and from the transfer phase. In the absence of such constraints, only the lexical aspects of transfer and generation need to be considered. There may be ways of overcoming the computational burden created by phrasal constraints; for example, one can precompile grammars, types and *tlinks* in order to efficiently determine which constraints are more likely to be applicable during transfer or generation. Nevertheless, this additional processing would hamper development and debugging without guaranteeing deterministic phrasal constraint application.

Yet eliminating phrasal constraints implies that structural conditions on translation cannot be naturally expressed. We have not found this restriction very problematic, but if it were necessary, lexical markers could be introduced which would signal structural arrangements at the lexical level. For example, if during translation one wished to exploit the fact that a noun appeared as modifier of another noun, as in:

The summer institute.

the feature-value pair NN-MOD = **true** could be incorporated into 'fire' to mark it as appearing as a noun modifier in the expression. During transfer and/or generation this marking could be exploited, for example, to introduce the additional preposition needed for translation into Spanish:

El instituto **de** verano

This feature could be used as a restriction in the respective bilexical entry:

$$\left[\begin{array}{l}\textbf{noun-lex}\\ \textsc{orth} = \textbf{summer}\\ \textsc{nn-mod} = \textbf{true}\end{array}\right] \Leftrightarrow \left[\begin{array}{l}\textbf{prep-lex}\\ \textsc{orth} = \textbf{de}\end{array}\right] \left[\begin{array}{l}\textbf{noun-lex}\\ \textsc{orth} = \textbf{verano}\end{array}\right]$$

Other structural conditions such as verb inversion and preposed constituents could be similarly handled, if necessary. Thus, one would encode structural information in lexical signs without unnecessarily increasing the complexity of the system.

5.2 Lexicalist Generation

Lexicalist generation is through a modified parser which treats the input as a bag rather than as a list. Thus, the fundamental rule of chart parsing (Gazdar and Mellish 1989:195), which normally tests for the adjacency of an active and an inactive edge, is changed to test for the empty intersection of the set of lexical leaves in the active and inactive edges. That is, edges are combined if they do not have any lexical signs in common; this condition is, in fact, implicit in normal parsers since they follow the strict linear ordering of the input and therefore cannot contain a lexical sign more than once within a single edge.

One consequence of this change is that the algorithm becomes NP-complete (i.e. almost certainly without a polynomial time solution) as demonstrated by Brew (1992). Hence efficiency becomes a very important issue for this type of generation; the rest of this section considers ways of addressing this inefficiency.

5.2.1 Constraint Propagation

Brew (1992) proposes the addition of a constraint propagation technique to prune the lexicalist generator's search space. Consider the bag:

{the, fierce, little, brown, cat, sleeps}

from which a sentence must be generated. Without heuristics, the following incomplete sentences, among others, are constructed:

1) The cat sleeps.
2) The brown cat sleeps.
3) The little brown cat sleeps.

These incomplete sentences are a major source of inefficiency in lexicalist generation. To eliminate them, Brew constructs a functor-argument graph for the decomposed categorial grammar signs of the input sentence and restricts parsing to a search for a spanning tree of the graph so constructed.

A variant of Brew's technique has been adopted in which the graph represents possible adjacencies between lexical signs in the input bag. In addition, phrase structure grammars have been used instead of the categorial framework that Brew adopts. For example, the graph shown in Figure 5 is the unconstrained graph that results from the input bag above. The grammar that gives rise to these adjacencies is Brew's adjective grammar, in which certain adjectives must precede certain others. This graph shows that an adjective like 'fierce' must precede all adjectives, 'brown' must follow every other adjective, and 'little' must precede 'brown' but follow 'fierce'. Given an adjacency graph, a valid parse (i.e. a generated sentence) will trace a Hamiltonian circuit through it; that is, it will trace a circuit in which every node in the graph is visited exactly once. Nodes START and END are added to simplify the algorithm; the dotted arrow[5] between them is included to comply with the definition of a Hamiltonian circuit (Garey and Johnson 1979:47). A graph such as this may be constructed by computing the FOLLOW function (Aho et al. 1986:189) for the phrase structure grammar of the TL, as described in Trujillo (1994).

[5]The word 'arrow' is used instead of the more standard graph-theoretic term 'edge' to avoid confusion with the edges in the chart

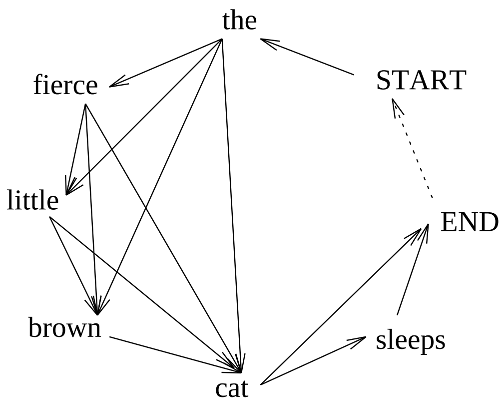
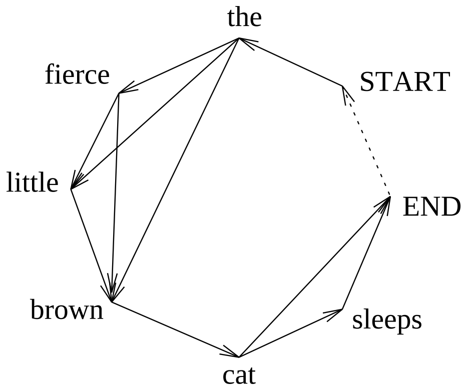


Figure 5: Unconstrained adjacency graph.

Once the adjacency graph is constructed, constraints are propagated through it to eliminate arrows which cannot be part of a Hamiltonian circuit. An arrow is eliminated if its being part of a circuit would dissallow some node from being in the final solution. For example, the arrow from 'the' and 'cat' prevents 'brown' from being in a circuit because the only way 'brown' can be part of a circuit is if it is followed by 'cat'. This means that one may delete the arrow 'the $\rightarrow$ cat'. During constraint propagation this process is repeated until no more deletions are possible. For this particular case, constraint propagation alone leads to a single path through the graph. Intermediate and final steps are shown in Figure 6. In general however, a single circuit will not be obtained by constraint propagation



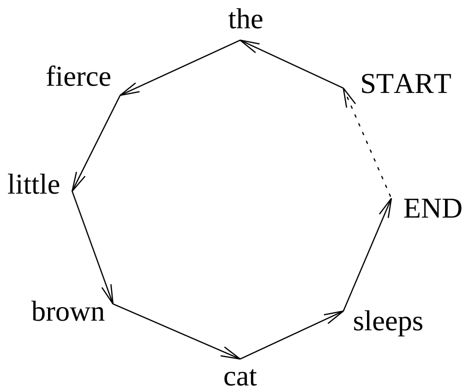


Figure 6: Middle and final stages in constraint propagation.

alone, in which case the lexicalist generation algorithm will inspect and update the graph in the predicting step before applying the fundamental rule. In doing this, many possible applications of the rule are blocked: first, if an arrow in the graph has been deleted and the proposed active-inactive pair assumes that arrow (i.e. their right-most and left-most leaves respectively correspond to the nodes of the deleted arrow) then application is disallowed. Similarly, if application of the rule, and hence selection of a particular sub-path in the graph, leads to an impasse, application is also disallowed. The initialization and non-destructive predictive algorithms for the adjacency graph are given in Figures 12 and 13 in the Appendix.

5.2.2 Reachability Constraints

The previous technique works best with strictly ordered constituents. However, if the order of adjectives, for example, is relaxed, the effectiveness of the algorithm is diminished. Consider the bag

{ the, thirsty, hungry, sleepy, dog }

Assuming there are no ordering constraints on these three adjectives, Figure 7 shows its adjacency graph before parsing starts, and after the subparse for 'the sleepy dog' has been constructed. Using constraint propagation alone it is difficult to detect the invalidity

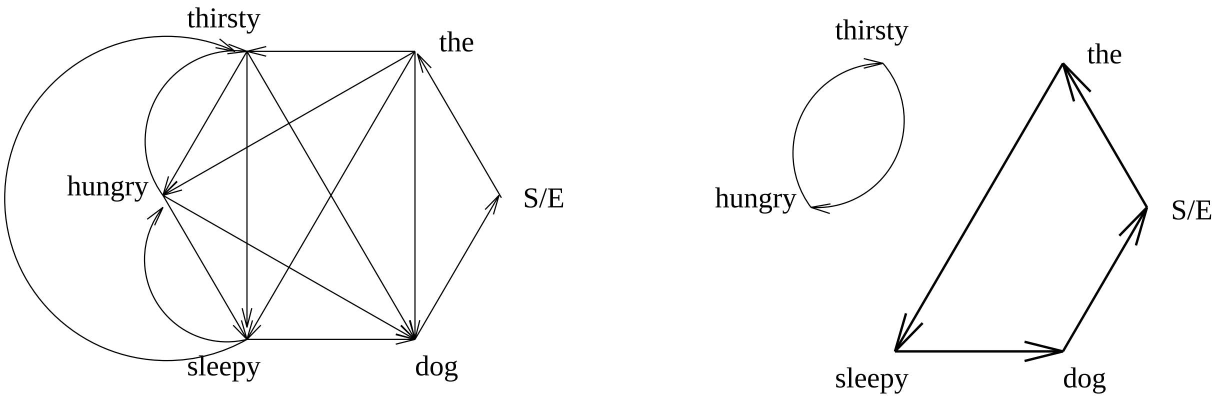


Figure 7: Constraint propagation with unordered adjectives.

of the graph on the right because each edge has an incoming and an outgoing edge. In order to overcome this problem it is necessary to introduce a reachability check which determines whether all nodes in the graph are reachable from the start node once constraint propagation has been performed. In the bag in question, application of the fundamental rule to the edges 'the' and 'sleepy dog' would be disallowed with a reachability check because the constrained graph that would result (right graph in Figure 7) has two nodes unreachable from the start node. An outline of the relevant algorithm is given in Figure 15 in the Appendix.

5.2.3 Connectivity Constraints

Another important source of inefficiency are modifiers such as PPs and relative clauses (Phillips 1993). The problem here is that neither constraint propagation nor reachability constraints prevent the construction of invalid parses. For example, a bag such as:

{the, boy, on, the, bus, from, London, saw, the, girl}

with intended interpretation 'the boy on the (bus from London) saw the girl' results in arrows connecting 'the girl' and 'from London' (actually, they connect 'girl' and 'from' but this is may be ignored) among other pairs, as shown on the left graph in Figure 8 (only relevant arrows shown). The problem with graphs like this one is that even after several applications of the fundamental rule, impossible parses go undetected because there are always sufficient arrows to ensure that every node is reachable from the start node. The graph on the right in Figure 8 shows the state of the adjacency graph after the subparse for 'the boy on the bus saw the girl' has been constructed; this subparse can never be part of a complete sentence because the PP that modifies 'bus' is missing. In fact, all of the following sentences are constructed, but only the last one is complete:

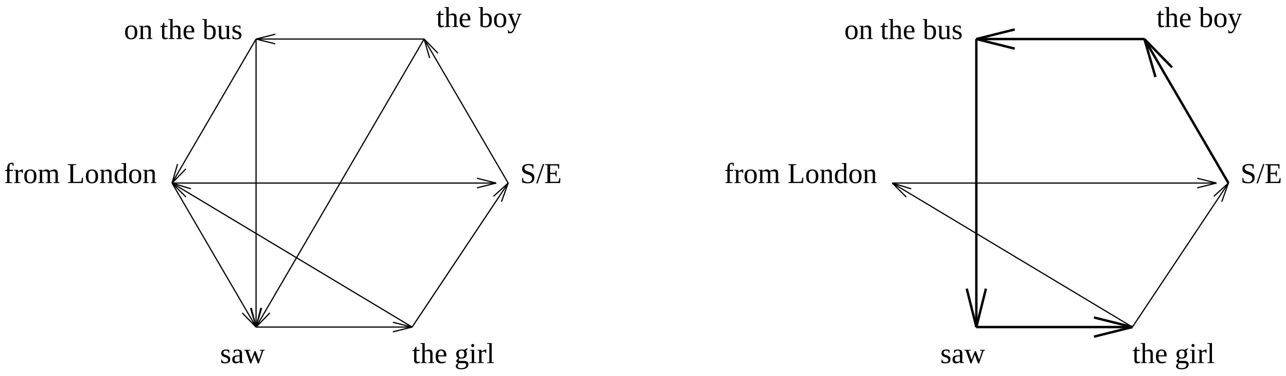


Figure 8: Arrows arising from PP modification.

The boy saw the girl.
The boy on the bus saw the girl.
The boy on the bus from London saw the girl.

The solution to this problem relies on imposing conditions on lexical signs and on the grammar rules. The condition on lexical signs is that they all must be connected. Two lexical signs are connected if they share one or more semantic variables; furthermore, if a and b are connected, and b and c are connected, then a and c are connected (i.e. the connected relation is transitive). This condition involves associating a semantic variable with lexical signs which are not normally assigned semantic content; for example, verb particles such as 'up' in 'call up friends' (Shieber et al. 1989) would have to be associated with the event variable of their respective verbs (see Section 4.2.1).

The condition on the grammar rules is that each mother must be associated with a distinguished lexical sign drawn from one of its leaves or daughters; furthermore, the distinguished lexical sign of a lexical sign is itself. Similar conditions are proposed in HPSG (Pollard and Sag 1994:Ch. 8), where the semantic nucleus of some daughter in a rule is shared with the mother.

With these restrictions, the invalid subparse shown on the right in Figure 8 above can now be detected and avoided as follows. First, in the construction of the adjacency graph, a parallel connectedness graph is derived which determines which lexical signs in the input bag are directly connected (i.e. share a variable). During lexicalist generation, whenever the fundamental rule is to be applied, a check is made to determine whether the lexical signs not included in the edge plus the distinguished lexical sign resulting from applying the fundamental rule, form a connected graph. If this is not the case, then the result of applying the fundamental rule is discarded, effectively forcing application failure. For example, after variable instantiation, the above sentence may have the following constant assignments in its (Prolog-like) transfer representation:

```
[the(1), boy(1), on(1,2), the(2), bus(2), from(2,3), london(3), saw(1,4), the(4), girl(4)]
```

The simplified connectivity graph for this bag is shown in Figure 9 (double arrows indicate directly connected lexical signs). To prevent the construction of an edge for 'the boy on the bus saw the girl', it is necessary to verify that the lexical sign 'saw', which is the distinguished lexical sign for this phrase, and the lexical signs remaining in the input bag (i.e. 'from' and 'London') are not connected. Since this is the case, the constructed edge is discarded, effectively forcing the fundamental rule to fail. The left of Figure 10 shows the resulting graph for such an edge, with the connected lexical signs encompassed by a line and the distinguished lexeme underlined. The algorithm for initializing the connectivity graph

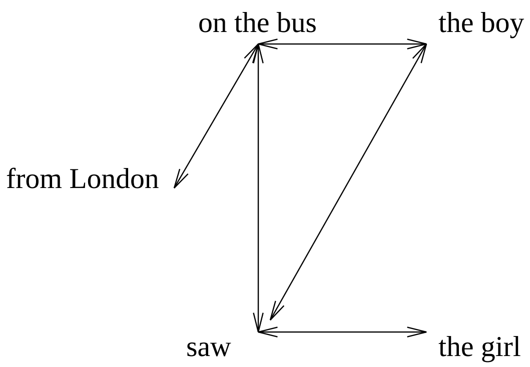


Figure 9: Connectivity graph.

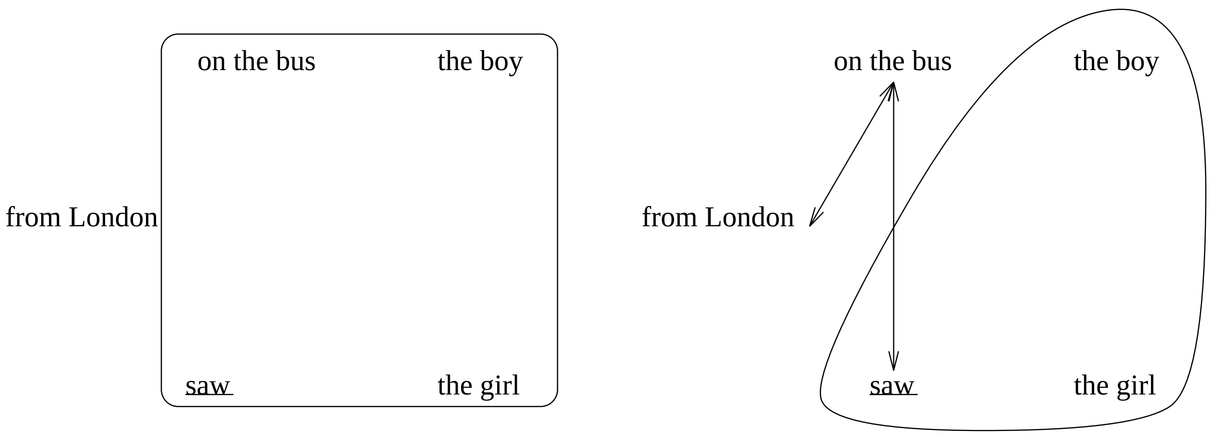


Figure 10: Connectivity graph for two impossible subparses.

is similar to that used for initializing the adjacency graph. The algorithm for checking that there are no disconnected lexical signs is given in Figure 16 in the Appendix.

Unfortunately, connectivity constraints allow the edge 'the boy saw the girl' to be constructed because 'saw' remains connected to the remains of the bag through 'on the bus' as shown on the right of Figure 10. A solution proposed by Martin Kay (personal communication) for this source of inefficiencies may be to restrict the indices included in the distinguished lexeme of an edge to those strictly necessary for further semantic binding. For example, since 'saw' includes an index for its subject 'boy', which in turn shares an index with 'on', connectivity cannot eliminate 'the boy saw the girl'. But if 'saw' were made to include only indices required for further binding during generation (e.g. the event index, needed for adverbial modification[6]), while indices such as the subject and object indices in 'saw' were not available after edge construction, this phrase would be rejected since it would lead to a disconnected graph. While this mechanism requires additional restrictions on the indices allowed in the distinguished lexeme of grammar rules, it seems that automatic checking of such restrictions is possible. This is a topic for further research.

5.3 Results

Table 1 shows the result of testing the three pruning techniques above on the LKB using CMU Common Lisp running on a Sun SPARCstation. The results are only for lexicalist generation, and do not include the time required for analysis and transfer; they do, however, include the time taken to find all grammatical permutations of the TL bag. One can see from the timings that initialization and maintenance of the various graphs is fairly expensive in the sense that for simple sentences, the overhead incurred causes generation with pruning to be slower than without. However, for more complicated sentences with several modifiers, there is a marked improvement in execution time. The column headed *+ Pruning* indicates that the lazy evaluation of the conjunction of all three techniques presented above (i.e. adjacency constraints, reachability and connectivity) is used to determine whether to allow successful application of the modified fundamental rule. The columns headed *Wfss* show the number of well-formed substrings constructed by the generator; this number is a better indicator of potential algorithm performance.

	Modified chart parser		*+ Pruning*	
Bag	*Time (secs)*	*Wfss*	*Time (secs)*	*Wfss*
The dog chases the cat	1	18	2	18
The fat dog chases the little cat	2	36	3	26
The fat brown dog chases the little tame cat	6	94	7	52
The dog in the park by the station chases the cat	3	47	7	42
The fat dog in the big park by the noisy station chases the tame cat	35	238	20	84

Table 1: Effect of pruning technique on different constructions.

6 Conclusion

This paper has considered in some detail the main features of lexicalist MT techniques and has attempted to contrast them with those of other approaches to transfer MT. One of the

[6]The issue is actually more complex, since subjects are sometimes necessary for further binding: John went to Scotland to study.

distinguishing features of lexicalist approaches is their simplified transfer component. Simplification stems mainly from the absence of structural transformations and from a uniform treatment of lexical gaps, idioms and other multi-lexeme translations (Trujillo 1995). Since the transfer component is normally expensive and more difficult to develop and maintain than the monolingual components, this simplification is attractive. It also opens the possibility of automatically or semi-automatically acquiring contrastive knowledge either from bilingual texts, or from terminology databases and bilingual dictionaries. Contrast this with other transfer approaches, where transfer rules are expressed in terms of the internal representation produced by the analyser, which makes their automatic acquisition more difficult.

Reversibility is another useful feature of lexicalist approaches such as the system described, particularly for bi-directional applications, as it reduces development costs. In the present approach, grammars are fully reversible since they are used in similar ways for analysis and generation: the difference is that during lexicalist generation, ordering information is disregarded. Yet, the process is complete because the generator is guaranteed to generate all the strings accepted by the TL grammar which satisfy the constraints imposed by the TL bag. Lexicalist generation is also sound because only strings which satisfy the constraints of the TL grammar are constructed. In addition, termination is guaranteed if it is guaranteed for parsing since one can at worst construct a generation algorithm which simply attempts all permutations of the TL bag and then parses them in order to test whether they are appropriate TL sentences.

The results of a small evaluation of our implementation reveal that while TL disambiguation, grammar and lexical coverage, and pronoun resolution remain crucial issues (in concert with most MT systems), the grammaticality and intelligibility of our output is at least comparable to that of most existing systems. Good intelligibility in our system is attributable to reversible grammar usage and adequate treatment of multi-lexeme translations. The former ensures that TL sentences are grammatical, while the latter allows natural translations to be included in the bilingual dictionary, regardless of the number of lexical signs required.

Efficiency in the generators has also been discussed thoroughly. The main sources of inefficiency have been identified, and several of these have been tackled. For example, order of lexical signs predictable from the grammar was exploited using constraint propagation; reachability was used to extend the use of constraint graphs, while connectivity eliminated certain cases of modifier inefficiency. One line of work on lexicalist generation has used greedy algorithms that start with an initial incorrect and/or incomplete parse and iteratively refine it until a complete parse is found (Poznański et al. 1995). However, such approaches place restrictions on the grammars which should be avoided if possible. Other work uses memoization in order to prevent the recalculation of previous grammar rule applications (Popowich 1995); this approach yields improvements in execution time, but does not address the inefficiencies caused by modifiers.

6.1 Future Work

The main short term issue to be tackled in lexicalist MT is the inefficiency of the generator. A variety of mechanisms are now available for tackling the NP complete character of this problem, and it seems that progress is being made towards efficient algorithms. One area to be pursued generalises connectivity constraints in order to make them applicable to most instances of modification inefficiencies. This would place few restrictions on the grammars while eliminating many inefficiencies due to modifiers.

In the longer term, it will be important to develop an adequate semantic representation which can be used for disambiguation and pronoun resolution. One representation that seems particularly relevant is the Minimal Recursion Semantics (MRS) described by Copestake et al. (1995). The principal characteristic of MRS is that scope information, instead of being represented via recursive structures, is instead encoded as labels associated with semantic predicates, where a particular label represents some scoping domain. The transfer and generation algorithms presented here seem well suited to translation through MRS, and it may be that predicates in MRS could replace the lexical signs in our approach, without affecting the main benefits of lexicalist MT.

There are issues which also need to be resolved in lexicalist MT, including the problem of multi-lexeme translations. Consider the following examples:

1) Eng: The troops **march across** the valley.
1′) Spa: Las tropas **cruzan** el valle **marchando**.

2) Eng: The general **marches** the troops.
2′) Spa: El general **hace marchar** a las tropas.

3) Eng: The general **marches** the troops **across** the valley.
3′) Spa: El general le **hizo cruzar** el valle a las tropas **marchando**.

The multi-lexeme translation problem stems from the interaction of two or more transfer relationships, and the way that these need to be combined to give natural translations. For 1) and 2) for example, the transfer relationship may be described as:

march across NP $\Leftrightarrow$ cruzar NP marchando
$\text{march}_{causative}$ NP $\Leftrightarrow$ hacer marchar NP

Unless one posits distinct predicates for the various meaning components present in each phrase, these multi-lexeme translation relations seem the most appropriate mechanism for such translations in lexicalist MT. This is because at the lexical level, there does not exist a straightforward one-to-one correspondence between individual source and target lexemes; instead, a relationship does exists between the bags of lexemes shown. The problem then, is that while the combination of these two bilexical entries should yield the relation:

$\text{march}_{causative}$ NP1 across NP2 $\Leftrightarrow$ hacer cruzar NP1 NP2 marchando

which is necessary for translating 3), this combination does not follow from application of the transfer algorithm alone. Additional procedures need to be introduced, which should take into account the various morphological, syntactic and semantic constraints required in 3) while allowing for reversibility. In Trujillo (1995), a partial solution to this problem is presented, based on a modified version of *tlink-rules* and the addition of contextual conditions to bilexical entries. However a full solution awaits further research.

References

Agnäs, M.-S., Alshawi, H., Bretan, I., Carter, D. M., Ceder, K., Collins, M., Crouch, R., Digalakis, V., Ekholm, B., Gambäck, B., Kaja, J., Karlgren, J., Lyberg, B., Price, P., Pulman, S., Samuelsson, M. R. C., and Svensson, T. (1994). Spoken language translator: First year report. Technical report, SRI/SICS, Menlo Park, CA, Cambridge, UK and Stockholm, Sweden.

Aho, A. V., Sethi, R., and Ullman, J. D. (1986). *Compilers - Principles , Techniques, and Tools.* Addison Wesley, Reading, MA.

Alshawi, H., ed. (1992). *The Core Language Engine.* MIT Press, Cambridge, MA.

Alshawi, H., Carter, D., Gambäck, B., and Rayner, M. (1992). Swedish-English QLF translation. In Alshawi (1992), ch. 14, pp. 277–309.

Beaven, J. L. (1992). *Lexicalist Unification Based Machine Translation.* PhD thesis, Department of Artificial Intelligence, University of Edinburgh, Edinburgh, UK.

Bennett, W. S. and Slocum, J. (1988). The LRC machine translation system. In Slocum (1988), pp. 111–40.

Bresnan, J., ed. (1982). *The Mental Representation of Grammatical Relations.* MIT Press Series on Cognitive Theory and Mental Representations. The MIT Press, Cambridge, MA.

Bretan, I. and Collins, M. (1994). QLF-based transfer for ATIS. Technical report, SRI/SICS, Menlo Park, CA, Cambridge, UK and Stockholm, Sweden. ch. 8.

Brew, C. (1992). Letting the cat out of the bag: Generation for Shake-and-Bake MT. In *Proceedings of the 14th COLING*, pp. 610–16, Nantes, France.

Briscoe, E., Copestake, A., and de Paiva, V., eds. (1993). *Inheritance, Defaults and the Lexicon.* Cambridge University Press, Cambridge, UK.

Briscoe, E. J. and Copestake, A. (1991). Sense extensions as lexical rules. In *Proceedings of the IJCAI Workshop on Computational Approaches to Non-Literal Language*, pp. 12–20, Sydney, Australia.

Calder, J., Reape, M., and Zeevat, H. (1989). An algorithm for generation in unification categorial grammar. In *Proceedings of the Fourth European Conference of the ACL*, pp. 233–40, Manchester, England.

Carpenter, R. (1992). *The Logic of Typed Feature Structures.* Tracts in Theoretical Computer Science. Cambridge University Press, Cambridge, UK.

Copestake, A. (1993). Constraints, tlinks and MT. Technical Report 3.1, ESPRIT BRA-7315 ACQUILEX II Working Paper, Publishing Division, Cambridge University Press, UK.

Copestake, A., Flickinger, D., Malouf, R., Riehemann, S., and Sag, I. (1995). Translation using minimal recursion semantics. In *Proceedings of the 6th International Conference on Theoretical and Methodological Issues in Machine Translation*, Leuven, Belgium.

Copestake, A. and Sanfilippo, A. (1993). Multilingual lexical representation. In Dorr, B., ed., *Building Lexicons for Machine Translation, Proceedings of the AAAI Spring Symposium*, Stanford, CA.

Copestake, A., Sanfilippo, A., Briscoe, E., and de Paiva, V. (1993). The ACQUILEX LKB: An introduction. In Briscoe et al. (1993), ch. 9, pp. 148–63.

Dorr, B. J. (1994). Machine translation divergences: A formal description and proposed solution. *Computational Linguistics*, 20(4) pp. 597–633.

Dowty, D. R., Wall, R., and Peters, P. S. (1981). *Introduction to Montague Semantics.* Reidel, Dordrecht, The Netherlands.

Emele, M., Heid, U., Momma, S., and Zajac, R. (1992). Interaction between linguistic constraints: Procedural vs. declarative approaches. *Machine Translation*, 7(1-2) pp. 61–98.

Fontenelle, T., Adriaens, G., and de Braekeleer, G. (1994). The lexical unit in the Metal MT system. *Machine Translation*, 9(1) pp. 1–20.

Gambäck, B., Alshawi, H., Carter, D., and Rayner, M. (1991). Measuring compositionality in transfer-based machine translation systems. In *Natural Language Processing Systems Evaluation Workshop*, Griffiss Air Force Base, NY 13441-5700. Rome Laboratory, Air Force Systems Command.

Garey, M. R. and Johnson, D. S. (1979). *Computers and Intractability: A Guide to the Theory of NP-Completeness.* W. H. Freeman, New York.

Gazdar, G. and Mellish, C. (1989). *Natural Language Processing in PROLOG: An Introduction to Computational Linguistics.* Addison-Wesley, Wokingham, England.

Hinrichs, E. W., Meurers, W. D., and Nakazawa, T. (1994). Partial-VP and Split-NP Topicalization in German – An HPSG Analysis and its Implementation. Technical Report Arbeitspapiere des SFB 340 Nr. 58, Universität Tübingen, Germany.

Hutchins, W. J. and Somers, H. L. (1992). *An Introduction to Machine Translation.* Academic Press, London.

Isabelle, P. (1989). Toward reversible MT systems. In *Proceedings MT Summit II*, Munich, Germany.

Jackendoff, R. S. (1990). *Semantic Structures.* Number 18 in Current Studies in Linguistics. MIT Press, Cambridge, MA.

Kaplan, R. M. and Bresnan, J. (1982). Lexical-functional grammar: A formal system for grammatical representation. In Bresnan (1982), ch. 4, pp. 173–281.

Kaplan, R. M., Netter, K., Wedekind, J., and Zaenen, A. (1989). Translation by structural correspondences. In *Proceedings of the Fourth Conference of the European Chapter of the ACL*, pp. 272–81, Manchester, UK.

Kaplan, R. M. and Wedekind, J. (1993). Restriction and correspondence-based translation. In *Proceedings of the Sixth Conference of the European Chapter of the ACL*, pp. 193–202, The Netherlands. OTS, Utrecht University.

Kay, M., Gawron, J. M., and Norvig, P. (1994). *Verbmobil: A Translation System for Face-to-Face Dialog.* Number 33 in Lecture Notes. Centre for the Study of Language and Information, Stanford, CA.

Lascarides, A., Briscoe, T., Asher, N., and Copestake, A. (in press). Order independent and persistent typed default unification. *Linguistics and Philosophy.*

Magnúsdóttir, G. (1993). Review of *An Introduction to Machine Translation* by Hutchins, W. J. and Somers, H. L. *Computational Linguistics*, 19(2) pp. 383–384.

Meurers, W. D. (1994). On implementing an HPSG theory – aspects of the logical architecture, the formalization, and the implementation of head-driven phrase structure grammars. In Hinrichs et al. (1994).

Nirenburg, S., ed. (1987). *Machine Translation - Theoretical and Methodological Issues.* Studies in Natural Language Processing. Cambridge University Press, Cambridge, UK.

Phillips, J. D. (1993). Generation of text from logical formulae. *Machine Translation*, 8(4) pp. 209–35.

Pollard, C. and Sag, I. (1987). *Information Based Syntax and Semantics: Vol. 1.* Lecture Notes. CSLI, Stanford, CA.

Pollard, C. and Sag, I. (1994). *Head Driven Phrase Structure Grammar.* Chicago University Press, IL.

Popowich, F. (1995). Improving the efficiency of a generation algorithm for Shake and Bake machine translation using Head-Driven Phrase Structure Grammar. In *Proceedings of Natural Language Understanding and Logic Programming V*, Lisbon, Portugal.

Poznański, V., Beaven, J. L., and Whitelock, P. (1995). An efficient generation algorithm for lexicalist MT. In *Proceedings of the 33rd Annual Meeting of the Association for Computational Linguistics*, Boston, MA.

Rayner, M., Alshawi, H., Bretan, I., Carter, D., Digalakis, V., Gambäck, B., Kaja, J., Karlgren, J., Lyberg, B., Price, P., Pulman, S., and Samuelsson, C. (1993). A speech to speech translation system built from standard components. In *Proceedings of the 1993 ARPA workshop on Human Language Technology*, Princeton, NJ.

Sadler, L., Crookston, I., and Way, A. (1990). LFG and translation. In *Proceedings of the Third International Conference on Theoretical and Methodological Issues in Machine Translation – TMI-90*, University of Texas at Austin. LRC.

Sadler, L. and Thompson, H. (1991). Structural non-correspondence in translation. In *Proceedings of the Fifth Conference of the European Chapter of the ACL*, pp. 293–98, Bonn, Germany.

Sanfilippo, A., Briscoe, E., Copestake, A., Marti, M., Taule, M., and Alonge, A. (1992). Translation equivalence and lexicalization in the ACQUILEX LKB. In *Proceedings of the Fourth International Conference on Theoretical and Methodological Issues in Machine Translation – TMI-92*, pp. 1–11, Montreal, Canada.

Sanfilippo, A. and Poznański, V. (1992). The acquisition of lexical knowledge from combined machine-readable sources. In *Proceedings of the 3rd Conference on Applied Natural Language Processing (ANLP-92)*, pp. 80–88, Trento, Italy. ESPRIT BRA-3030 ACQUILEX WP NO. 35.

Schneider, T. (1991). The METAL system. Status 1991. In *Proceedings MT Summit III*, pp. 41–44, Washington DC.

Shieber, S., van Noord, G., Pereira, F. C. N., and Moore, R. C. (1990). Semantic-head-driven generation. *Computational Linguistics*, 16(1) pp. 30–42.

Shieber, S. M. (1993). The problem of logical-form equivalence. *Computational Linguistics*, 19(1) pp. 179–90.

Shieber, S. M., Uszkoreit, H., Pereira, F. C. N., Robinson, J., and Tyson, M. (1983). The formalism and implementation of PATR-II. Research on Interactive Acquisition and Use of Knowledge, SRI Final Report 1894, SRI International, Menlo Park, CA.

Shieber, S. M., van Noord, G., Moore, R. C., and Pereira, F. C. N. (1989). A semantic-head-driven generation algorithm for unification-based formalisms. In *Proceedings of 27th Annual Conference of the ACL*, pp. 7–17, Vancouver, Canada.

Slocum, J., ed. (1988). *Machine Translation Systems*. Studies in Natural Language Processing. Cambridge University Press, Cambridge, UK.

Trujillo, A. (1994). Computing FIRST and FOLLOW functions for Feature-Theoretic grammars. In *Proceedings of the 15th COLING*, pp. 875–80, Kyoto, Japan.

Trujillo, A. (1995). Bi-lexical rules for multi-lexeme translation in lexicalist MT. In *Proceedings of the Sixth International Conference on Theoretical and Methodological Issues in Machine Translation – TMI-95*, Leuven, Belgium.

Tucker, A. B. (1987). Current strategies in machine translation research and development. In Nirenburg (1987), ch. 2, pp. 22–41.

Vossen, P. and Copestake, A. (1993). Untangling definition structure into knowledge representation. In Briscoe et al. (1993), ch. 13, pp. 246–74.

Whitelock, P. (1992). Shake-and-Bake translation. In *Proceedings of the 14th COLING*, pp. 784–91, Nantes, France.

Zajac, R. (1989). A transfer model using a typed feature structure rewriting system with inheritance. In *Proceedings of the 27th Annual Conference of the ACL*, pp. 1–6, Vancouver, Canada.

Appendix: Algorithms

Purpose: given a list of source lexical signs (LSs) construct all possible covers that the list can have using a bilexicon which defines allowed lexical correspondences

Inputs: a list of source lexical signs

Results: a covers list

Background: a bilingual lexicon

Working Structures:

Current LS: a lexical sign

Partial LS List: a list of LSs constructed by removing elements from the input list

Source List: a list of SL lexical signs taken from a bilingual entry

Current Bilingual Entry: a pair of source and target lists containing LSs standing in the transfer relation; its source side contains one of the LSs in the input

Cover: a set of bilingual entries describing the translation of a list of LSs.

Result: a list of all the covers possible from a source list of LSs; used as final value of **Cover-SL-List**

Partial Result: just like result but used as a temporary store for the value of the recursive step

Basic Method: Iteration through bilingual entries, recursive call on part of the input, iteration through result of recursion:

- If the input is the empty list then assign to result an empty cover as its only element and return result; this accounts for the recursion's base case
- Otherwise initialize result to the empty list in order to accumulate values from subsequent steps
- Remove the first member from the input list and assign it to the current LS.
- For each bilingual entry in the bilingual lexicon whose SL side contains the current LS do:
  - Make this the current bilingual entry
  - Assign to source list the SL side of current bilingual entry
  - Remove every element of source list from the input list, in the order defined by source list, and assign the remaining list to partial LS list
  - Assign partial result the value of applying **Cover-SL-List** to partial LS list
  - For each cover in partial result do:
    - Complete the result of the recursive step by adding the current bilingual entry to this cover, thus incorporating the bilingual entry used for the current level call
    - Append the cover thus obtained to result, thus collecting all possible translations of the input
- Return result

Figure 11: **Cover-SL-List** algorithm.

Purpose: given a list of numbered lexemes, construct a graph indicating which lexemes might follow which others

Inputs: a list of (integer,lexical-sign) pairs

Results: an adjacency graph

Background: a Follow function

Data Types:

Node: a record containing: the number of the lexical sign it represents, its TFS, a list of possible follow nodes, a unique preceding node if any, and two additional fields for working storage in non-destructive constraint propagation (the shadow fields).

Working Structures:

Adjacency Graph: a list of nodes

Basic Method: Compare each node with each other to determine whether they can be adjacent. At the end, apply constraint propagation.

- For each node N_i in the Adjacency Graph
  - For each node N_j in the Adjacency Graph:
    Based on Follow function, if N_j can follow N_i, add j to the follow list of N_i
- Apply constraint propagation algorithm to Adjacency Graph

Figure 12: Initializing adjacency graph.

Purpose: determine whether the combination of two edges would result in an impossible sub-parse

Inputs: an active and an inactive edge, where each edge has a list of the lexeme numbers it spans, arranged in orthographic order

Results: boolean

Background: an adjacency graph, with each node containing working storage (shadow fields) for effecting constraint propagation.

Working Structures:

Path Traced: A list of lexical sign numbers indicating the path traced by the two edges

Basic Method: Initialize Path Traced, apply constraint propagation to the result.

- Initialize: copy the follow and preceding information of each node into its respective shadow field; concatenate the leaf identifiers of active and inactive edges and assign it to Path Traced.
- For each element in Path Traced, get its corresponding Adjacency Graph node and mark its follow shadow field with the lexeme number that follows the element in Path Traced; similarly, mark the preceding shadow field of the node with the number that precedes it in Path Traced.
- Apply the constraint propagation algorithm to the shadow fields of Adjacency Graph and return its result.

Figure 13: Predicting step for adjacency constraints.

Purpose: propagate adjacency constraints on an adjacency graph

Inputs: an adjacency graph and a field indicator which determines whether the main (for initialization) or shadow (for predicting) fields are to be used

Results: boolean

Basic Method: Iterate through adjacency graph eliminating arrows until no more changes are made or an impass in the form of a node without outgoing arrows is found.

- Initialize: valid := true, changed := true
- Repeat until not valid or not changed
  - changed := false
  - For every node N_i in Adjacency Graph
    - If an outgoing arrow from N_i points to a node with a different incoming arrow, delete the outgoing arrow from N_i and set changed := true. Do this for every outgoing arrow in N_i.
    - If only one outgoing arrow j remains in N_i, then make i the value of incoming arrows of node N_j.
    - If the number of remaining outgoing arrows from N_i is 0, then set valid := false since N_i cannot be part of a circuit. Abort current loop.
- Return valid

Figure 14: Constraint propagation.

Purpose: determine whether every node in an adjacency graph is reachable from the input node. When this function is used, the length of its result determines whether all nodes are reachable from the start node.

Function Name: Reachable-Nodes

Inputs: an adjacency graph node (initially the start node), an adjacency graph to which constraint propagation has been applied and a list of reached nodes (initially nil).

Results: list of node identifiers reached since the top-most recursive call was made; on returning from this call the value will be a list of all the nodes reachable from the start node.

Working Structures:

Reached Nodes: a cumulative list of the node identifiers reached

Basic Method: call function recursively on each of its immediately reachable nodes (i.e. nodes in the shadow follow field), accumulating results for each recursive call.

- Initialize: set Reached Nodes to the value of the input reached nodes.
- For each node N_i which is immediately reachable from the input node:
  - If N_i has not been reached (i.e. $i \notin$ Reached Nodes)
    - Add i to Reached Nodes
    - Reached Nodes := Reachable-Nodes(N_i, adjacency graph, Reached Nodes)
- Return Reached Nodes

Figure 15: Reachability calculation.

Purpose: determine whether a list of (identifier,sign) pairs is connected in the sense that distinguished lexemes in the signs share semantic variables.

Inputs: a pair (identifier,phrasal-sign) and a list of pairs (identifier,lexical-sign); the single pair includes the result of combining an active and an inactive edge; the pairs list includes the lexical signs not spanned by either edge.

Results: true if the signs are connected, where the connected relation is transitive; false otherwise.

Background: an array indicating which lexical signs are directly connected to which others.

Working Structures:

Disconnected Nodes: A list of sign identifiers indicating which nodes have not been connected to the first node in the input list (a global variable).

Basic Method: call auxiliary function; check Disconnected Nodes.

- Initialize: set Disconnected nodes to a list of all the input identifiers.
- Call Total-Connected-Aux(input pair, input pair list).
- If every element in Disconnected is false, return true and false otherwise

---

Auxiliary Function: Total-Connected-Aux

Purpose: Update Disconnected by changing to false all elements which are connected to the sign in its first input.

Inputs: an (identifier,sign) pair and a list of such pairs

Results: none; called for its side effect on Disconnected

Background: same as main function

Working Structures:

Disconnected Nodes: from enclosing environment

Basic Method: recursive call on directly connected elements.

- Replace the input identifier with false in Disconnected
- For every element P_i in the input list of pairs:
  If P_i is directly connected to the input pair, and $i \in$ Disconnected then Total-Connected-Aux(P_i,input pair list)

Figure 16: Connectivity functions.