A SURVEY ON TRANSFER BASED APPROACHES TO MT USING FEATURE STRUCTURES

Toni Badia & Antoni Tuells

Sèrie Informes, 16

Barcelona
Universitat Pompeu Fabra. Institut Universitari de Lingüística Aplicada
1997

Direcció de les Publicacions de l'IULA: M. Teresa Cabré

Primera edició: 1997

La Rambla, 30-32
08002 Barcelona

Dipòsit legal: B-34.226-2002

A Survey on Transfer based approaches to MT using Feature Structures

Toni Badia, Antoni Tuells
Institut Universitari de Lingüística Aplicada
Universitat Pompeu Fabra
tbadia@upf.es , tuells@upf.es
Project Number: MLAP 93-15

Abstract

In this document we review some well known Transfer based approaches to MT that use Feature Structures. This work has been done within project MLAP 93-15, a project on the investigation of the EUROTRA linguistic specifications for industrial standards.

Abstract

Aquest treball és una revisió d'alguns sistemes de Traducció Automàtica que segueixen l'estratègia de Transfer i fan servir estructures de trets com a eina de representació. El treball s'integra dins el projecte MLAP-9315, projecte que investiga la reutilització de les especificacions lingüístiques del projecte EUROTRA per estàndards industrials.

Contents

1 INTRODUCTION

This a review of several well known transfer-based approaches to MT, though we don't claim it to be exhaustive. In selecting the list of approaches to evaluate we considered the following criteria:

- They are transfer-based
- They put into practice some of the advances in Computational Linguistics, namely the use of unification as the main operation for combining information contained in complex signs (Feature Structures, which contain constraints at various levels; syntactic, semantic and phonological)
- They are widely mentioned in the MT research community.

It is important to note that the different approaches reviewed do not have the same theoretical status; some often are directly built in the formalism associated with a linguistic theory while others are just independent formalisms or devices created for NLP uses. Nonetheless, our review is always based on the formalism presented so that the different approaches become comparable.

All of the approaches reviewed share another common characteristic : they go about defining linguistically possible translation, or translations which employ linguistic knowledge only. Though it is well known that correct or best translation requires other components (world knowledge, common sense reasoning,...), to try to build a system with such requirements at present is unrealistic.

We present all of the approaches in a sketchy, though sufficient manner, and comment briefly on their adequacy.

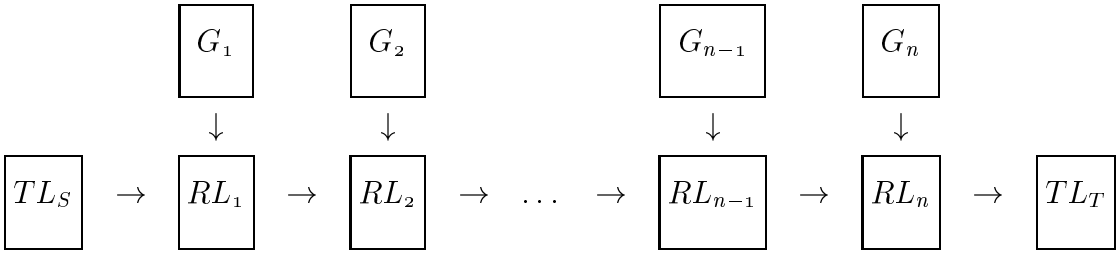
2 CAT2

CAT2 [Sharp, 1991] is an instantiation of the Eurotra MT methodology, and it was mainly developed to test MT theories within the Eurotra framework.

2.1 Basic Assumptions

CAT2 follows the Eurotra translation methodology [Arnold and des Tombe, 1987], i.e., stratified levels of representations described by generators, and mappings between levels described by translators:

(1)



where TL_S and TL_T stand for source and target language texts respectively; each RL_i is a representation language (or level), each G_i is meant to represent the corresponding generators and '$\rightarrow$' stands for the respective translators.

As is apparent in the figure above, there is no principled distinction between those translators that perform the mapping from one level of monolingual analysis to the next and those that relate representation levels of different languages (the transfer component).

Two conditions are imposed on all primitive translators:

1. They must be compositional
2. They must be one-shot

The idea behind compositionality is that translations of expressions are translations of its subexpressions in a systematic way. The idea behind the 'one-shot' condition is that the mappings are effected directly (without intermediate processing), in order to eliminate any interaction between the rules that make up a translator.

2.2 The Formalism

The basic representation structure in this formalism is the tree. Trees are meant to represent objects. Conceptually, objects fall into two main classes:

- sentential objects
  A sentential object is a representation of a sentence; it may be composed of subobjects, in general representing thinks like noun phrases, verb phrases, etc...
- semantic objects
  Subobjects can also be semantic objects, which represent semantic constituents, such as "entity", "process", etc ...

The primary object is a sentential object that is a representation of the sentence. An object O is a tree defined as follows (taken from [Sharp, 1991]):

(2) O = ROOT. BODY

where ROOT is the top node of the tree and BODY is a (possibly empty) list of subtrees under ROOT, each subtree having also the form in (2). The leafs of a tree are called atomic objects, which are objects whose BODY list is empty. Each node in the tree is represented as a feature set of attribute-value pairs:

(3) $[a_1 = v_1, a_2 = v_2, \ldots, a_{n-1} = v_{n-1}, a_n = v_n|\_]$

where each a_i is an atomic attribute name and each v_i a value, which may be an atomic constant, a variable or a complex feature . The tail of the list is an implementation detail: it is a Prolog (anonymous) variable used to collect (via unification) additional features while the processing of the object. Since CAT2 representational structures are trees and not DAGs (directed acyclic graphs) as in current unification-based foralisms (like PATR-II, HPSG,...), concepts like structure sharing cannot be properly captured; other aspects that conform most recent Feature Structures (FS) formalisms, like typing, are simply inexistent.

Let us look at an example: the representation of the sentence "John goes":

(4)
```
[cat=s,tense=pres|_].
 [[cat=np,agr=[pers=3,num=sing|_]|_].
   [[cat=n,lex='John',lu='John',agr=[pers=3,
                                    num=sing|_]|_].[]],
 [[cat=vp,agr=[pers=3,num=sing|_],tense=pres|_].
   [[cat=v,lex=goes,lu=go,agr=[pers=3,num=sing|_],
                              tense=pres|_].[]]]
```

2.2.1 Generators

A generator G consists of a set of b-rules and f-rules that define the well-formedness of an object at a given level of representation. B-rules are context-free rewrite rules that consolidate an object, and f-rules are rules that validate the object. Both concepts, validating and consolidating, are common with the Eurotra framework. F-rules are used to assign default feature values, to enforce feature conditions on representations, and to filter out objects with unsound features. They do not affect structure, but rather modify the feature content of objects (alternatively they simply filter out unwanted objects).

2.2.2 Translators

Translators define the mapping from objects of one level of representation onto objects of the next level of representation. Thus, a translator can be thought as a relation between levels. It has to be mentioned that a translator is a composition of structural-modifying relations (t-rules) and feature-modifying relations (translator f-rules), rather than a single application of a unique relation. Given that they may modify the structure of the object

(from one level to the next), translators introduce procedurality to the overall system. Syntactically, translator rules are of the form:

(5) ROOT.BODY $\Longrightarrow$ ROOT.BODY

A simple t-rule is the following:

(6) `{lu = gehen}.[] => {lu = go}.[] .`

where one atomic object is mapped onto another one, in which the value of the feature *lu* is changed.

A more complex example can be the following:

(7) `{cat = s}.[1:{cat = np},{cat = vp}.[2:{cat = v},*3]] => _.[ 2, 1. 3]`

The left hand side of this rule describes an object which is a sentence with two constituents: a noun phrase and a verb phrase. The VP has a verb and possibly other constituents. The translator deletes the VP node and maintains the verb and any other constituent that followed it under the VP.

2.3 Comments

CAT2 is a formalism derived from the Eurotra methodology. It shares with the ET framework the overall architecture. Nonetheless the CAT2 formalism is neater and more explicit in the semantics of its operators ("=" and "==" are distinguished, insertion of nodes is only allowed in translators...). At the same time, complex feature values are allowed, so that many linguistic relations are easier to express (as, say, subcategorization).

CAT2 follows the stratificational approach to modularity. This means that the different parts of the processing (the modules) are seen as conforming a step in a process. It is a linear sequence of different grammars to each of which corresponds a different sort of linguistic information.

A consequence of the stratificational approach is that transfer operates only on one level, in the objects of which all the information relevant for translation has to be present. This conforms a hybrid (usually badly defined) level of representation as the turning point of transfer.

Finally, although the different generators are declarative, the translators introduce procedurality and non monotonicity to the system. Thus, in the translators trees may be altered (nodes can be deleted, added, interchanged...) and the information may be changed.

3 LFG

[Kaplan *et al.*, 89] describe a transfer-based approach to MT within the LFG framework; it is based on codescription, which is one of the basic mechanisms of LFG. In general, linguistic constraints (syntactic and semantic) are expressed by means of the LFG mechanism of codescriptions, so that the different levels are put into correspondence with one another by projections or mappings. As a result, a set of levels of linguistic representation is obtained, rather than a single hybrid level of representation ready for the transfer component. The key assumption is (taken directly from [Kaplan *et al.*, 89]) :

> "This approach permits the mapping between source and target to depend on information from various levels of linguistic abstraction, while still preserving the modularity of linguistic components and of source and target grammars and lexicons."

Note that we address this approach from the point of view of the formalism; the theoretical aspects of LFG are not considered.

3.1 The Architecture for Monolingual Descriptions

Roughly speaking, we might describe an LFG grammar as a grammar assigning two levels of syntactic and one of semantic representation to every sentence (see figures 1, 2 and 3). The syntactic levels are called c-structure and f-structure: the first is a phrase-structure tree obtained by a set of context-free rules, while the second deals with grammatical functions (subject, object,etc.) described by ordinary untyped FS. Functions ϕ,σ are used to establish correspondences between c-structure/f-structure and f-structure/semantic structure respectively.

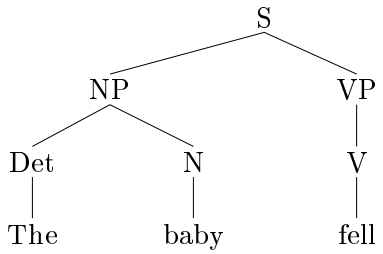


Figure 1: c-structure of sentence "The baby fell"

$$f_1\begin{bmatrix}
\text{PRED} & \text{fall}\left\langle\left[\text{BABY}\right]\right\rangle & \\
\text{TENSE} & \text{past} & \\
\text{SUBJ} & f_2\begin{bmatrix}
\text{PRED} & \text{baby} & \\
\text{NUMB} & \text{sing} & \\
\text{SPEC} & \begin{bmatrix}\text{DEF} & +\\ \text{PRED} & \text{the}\end{bmatrix} &
\end{bmatrix} &
\end{bmatrix}$$

Figure 2: f-structure of sentence "The baby fell"

$$\theta_1\begin{bmatrix}
\text{REL} & \text{fall} & \\
\text{ARG1} & \theta_2\begin{bmatrix}
\text{IND} & \boxed{1}\left[\text{ID} \quad \text{IND}\right] & \\
\text{SPEC} & \left[\text{DET} \quad \text{THE}\right] & \\
\text{COND} & \begin{bmatrix}\text{REL} & \text{baby}\\ \text{ARG1} & \boxed{1}\\ \text{POL} & 1\end{bmatrix} &
\end{bmatrix} & \\
\text{LOC} & \begin{bmatrix}\text{IND} & 1\\ \text{COND} & 2\end{bmatrix} & \\
\text{POL} & 1 &
\end{bmatrix}$$

Figure 3: semantic structure of sentence "The baby fell"

The equations establishing the projections from one level to another are stated both in the lexicon and in the c-structure rules. It is then the combining performed by the structure-building rules and the information contained in the lexical entries that produce the different levels of representation by the equation-solving mechanism. As a result, final structures (c-, f-, or semantic) are related with one another by means of functions ϕ (from c- to f-structures) and σ (from c- and f-structures to semantic structure). Thus, in the figures above the following can be established:

$$\phi(\text{S}) = \text{f1} \quad \sigma(\text{f1}) = \theta_1$$

In other words, since linguistic structures are put in correspondence by functions ϕ, σ, their construction is mutually constrained.

3.2 The Architecture for Transfer

The method described in section 3.1 can be extended to deal with the relationships between source and target languages (SL and TL) as follows:
Functions τ, τ' relate f-structures in SL to f-structures in TL, and semantic structures in SL to semantic structures in TL respectively.

For example, for an English-French module we might define the lexical entry for *like* in two different ways: either as establishing the translation by means of τ or by means of τ', as shown in the following two lexical entries:[1]

(8) a. *like*, V:
(τPRED) = like<*SUBJ, OBJ*>
(τPRED FN) = plaire<*SUBJ,AOBJ*>
(τAOBJ OBJ) = τ(SUBJ)
(τSUBJ) = τ(OBJ)

b. *like*, V:
($\sigma\phi$REL) = like
($\sigma\phi$ARG1) = σ(ϕSUBJ)
($\sigma\phi$ARG2) = σ(ϕOBJ)
($\tau'\sigma\phi$REL FN) = plaire
($\tau'\sigma\phi$ARG1) = τ'($\sigma\phi$ARG1)
($\tau'\sigma\phi$ARG2) = τ'($\sigma\phi$ARG2)

As can be easily seen from this example, there is nothing in the formalism that prevents using both τ and τ' constraints, so that the translation of a particular linguistic expression may be dependent on factors that are stated both in the f-structure and in the semantic one. This framework, then, provides a mechanism for imposing constraints on the form of a target sentence by relating them to information that appears at different levels of the source language description.

As shown by ([Sadler and Thompson, 1991] [Sadler, 92]) there are problems with this approach particularly when the translation units (i.e., the elements upon which transfer has to be performed) do not correspond to the analysis units of the source language (i.e., the units upon which the analysis of source language sentences is organised). This is a consequence of the architecture of the whole system: the τ and τ' projections are stated in the source language lexicon and c-structure building rules. It is then difficult to state projections that are not based on the structural units of the source language analysis.

Examples of this problem appear when a predicate plus one of its arguments are jointly translated into a single predicate, or when a head plus adjunct construction is translated by just a single element. The examples [Sadler, 92] discusses are respectively:

(9) a. commit suicide

[1] These examples are taken from [Sadler, 92].

b. se suicider

(10) a. toy library

b. ludothèque

This problem can be solved provided that the monolingual and the transfer component are separated, and the latter is provided with some powerful mechanisms. The separation of the two lexica is a necessary step to overcome the dependency of the transfer component on the monolingual distinctions and as such has to be seen as a clear improvement of the model. On the other hand, the proposal implies the use of powerful mechanisms, such as priority union and conjunction of entries.

Another problematic case for the LFG translation model is that of head-switching. Here again it is difficult to organise the translation projections in a lexicon and c-structure rules for monolingual analysis. The proposals to deal with these cases either introduce large elements in the grammar that are not monolingually motivated, or are based on a powerful mechanism again, such as functional uncertainty.

3.3 Comments

In this proposal the translation relation is seen as a relation between sets of representational levels, and not just between a single level of description (which would contain all sorts of different information that is relevant for translating). It is also based on description (codescription, actually) so that translating is not seen as building structure but as stating the correspondence between the linguistic units of information of one language and those of another one.

MT within LFG thus conforms to mainstream linguistics in that it is constraint-based, in a way that is consistent with constraint-based approaches to grammar in general. It is just an additional information type that is added to the set of constraints used to characterise linguistic units of the source language.

However, there are some fundamental problems with LFG-based MT. The first one is that the translational correspondences are established in the monolingual lexicon and c-structure rules. Consequently the transfer component is dependent on the monolingual one, thus diminishing the modularity of the system. In fact, proposals like Sadler's of separating the two lexica are not sufficiently worked out to know their exact practical impact.

It is also important to notice that there are expressive mechanisms in LFG in general, and in its translation formalization, which are far from being lean. The use of such tools as priority union and functional uncertainty makes the system far from expressible in a lean formalism like the one assumed in MLAP-9315.

4 TFS

TFS ([Emele *et al.*, 1992], [Zajac, 1993])is a typed feature structure rewriting system, which means that it consists of an inheritance network of typed feature structures together with a mechanism for rewriting a given feature structure. Obviously, the inheritance network of feature terms drives the rewriting process, which aims at producing a set of feature terms that are more specific (are subsumed by) than the initial feature structure. Thus, the rewriting process can be viewed as an alternative to the Context Free backbone in other Feature Structures formalisms (PATR-II, ALE, ALEP,...) . These concepts will be clarified shortly.

4.1 The Inheritance Network of Feature Terms

An inheritance network of feature terms defines a partial ordering on kinds of (linguistic) available information; to meet this objective we need the following:

- A set of type symbols $\mathcal{T}$ together with a partial ordering $\sqsubseteq$ on $\mathcal{T}$.
- Feature terms: (partial) descriptions of linguistic objects via sets of attribute-value pairs, as usual.
- A method of combining types and feature terms

4.1.1 Types

The set $\mathcal{T}$ of possible types is a partially ordered set (poset) ($\mathcal{T}$,$\sqsubseteq$) . The ordering $\sqsubseteq$ defines the subtype relation: A $\sqsubseteq$ B means 'A is a subtype of B'.

Other conditions are added in order to obtain a 'well-behaved type hierarchy' :

- The type hierarchy has two special symbols: $\top$ (top) and $\bot$ (bottom),where the top is the greatest element and bottom is the least element of $\mathcal{T}$. The top is interpreted as the whole universe and bottom is interpreted as the empty set. Any type of the hierarchy is interpreted as a subset of the top and $\sqsubseteq$ is interpreted as set inclusion.
- Any two type symbols A anb B of $\mathcal{T}$ have a greatest common lower bound called infimum of A,B and written inf{A,B}. A poset where this condition holds is called a meet semi-lattice. A new operation is introduced: A $\sqcup$ B = inf{A,B}, where A $\sqcup$ B is called the meet of A and B; this operation is known as *unification* in most of the literature on Feature Structures.

These conditions together mean that unification of any two type symbols A,B never fails: if the result of the operation is bottom, then it should be interpreted as a failure in Carpenter's ([Carpenter, 1992]) approach. The poset $\mathcal{T}$ does not need to be consistent:

the unification of A and B may return more than one result that should be interpreted disjunctively. This is a significant difference with respect to [Carpenter, 1992], which includes the consistent condition in order to have deterministic unification.

4.1.2 Linking Types and Feature Terms

The linking process is realized associating a type symbol to every Feature Term together with the following constraints:

- Every type defines a collection of features (and possibly restrictions on their values) which are appropriate for it.
- Every feature defines its own appropriate types.
- A subtype inherits all features appropriate for its supertypes, though it may put restrictions on their values (strong typing).

These conditions mean the following: a type cannot have a feature which is not appropriate for it and conversely, a feature-value pair should always be defined for some type, though there is no maximal-introduction constraint on features for types like those in [Carpenter, 1992]. Furthermore, a subtype inherits all constraints from its supertypes monotonically. In TFS, the inheritance network is specified via (possibly recursive) type definitions:

```
LIST = NIL | CONS.
CONS = [first: T, rest: LIST].

APPEND0 = APPEND[1:NIL, 2: #1=LIST, 3: 3l]
APPEND1 = APPEND[1: CONS[first: #x, rest: #l1],
                 2: #l2 = LIST,
                 3: CONS[first: #x, rest: #l3]]
          :- APPEND[1: #l1, 2:#l2, 3: #l3].
```

Figure 4: Type definitions for LIST and APPEND using the TFS syntax

Figure 4 shows the type definitions for LIST and APPEND using the TFS syntax; note that disjunction and recursive definitions are allowed. Figure 5 shows the inheritance network for LIST and APPEND which corresponds to the type definitions in figure 4 . Note how subtypes restrict the values of features inherited from its supertypes.

4.2 The Rewriting process

From the inheritance network the system computes a set of rewrite rules (figure 6) as follows: each direct link between a type α and a subtype β generates a rewrite rule of the

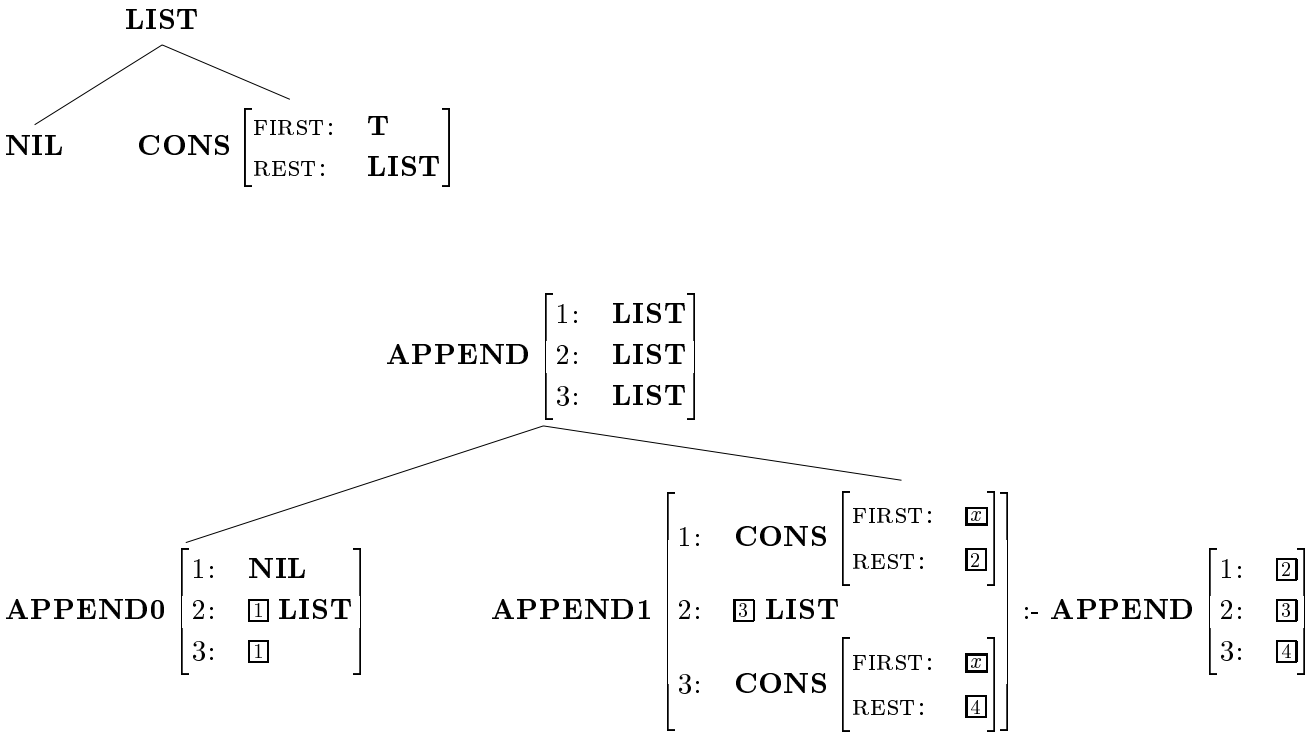


Figure 5: Inheritance Network for LIST and APPEND ($\top$ and $\bot$ omitted)

form $\alpha[F_a] \longrightarrow \beta[F_b]$, where F_a and F_b stand for the definitions of α and β respectively. Then these rules are used to evaluate a feature term (a 'query' in TFS terms): first, the system checks whether the input term is well-typed; if it is consistent, then it incrementally adds more information to that term using the rewrite rules, until all types are minimal and no further rule is applicable. One step of rewriting can be decomposed into two parts:

1. Inheritance : The feature term of type U inherits all definitions of the supertypes of U.

2. Specialization : The feature term is made more specific by unifying it with the immediate subtypes of U, if there is any. In general, subtypes lead to disjunctions and then to multiple solutions.

The result of this process is a (possibly empty) set of "ground" (most specific) feature terms which are regarded as the "denotation" of the input term. If this set is empty, the input feature term is inconsistent with the feature type system (the inheritance network of feature terms). A thorough explanation of this process can be found in [Zajac, 1993]. It is important to point out that the rewriting process defines how partial descriptions can be combined, just as other systems do using context free rules; the authors claim the gain is reversibility [2]: the same "grammar" can be used for both parsing and generation.

[2] [Emele *et al.*, 1992] prefer to use the term "non-directionality" rather than "reversibility", since TFS

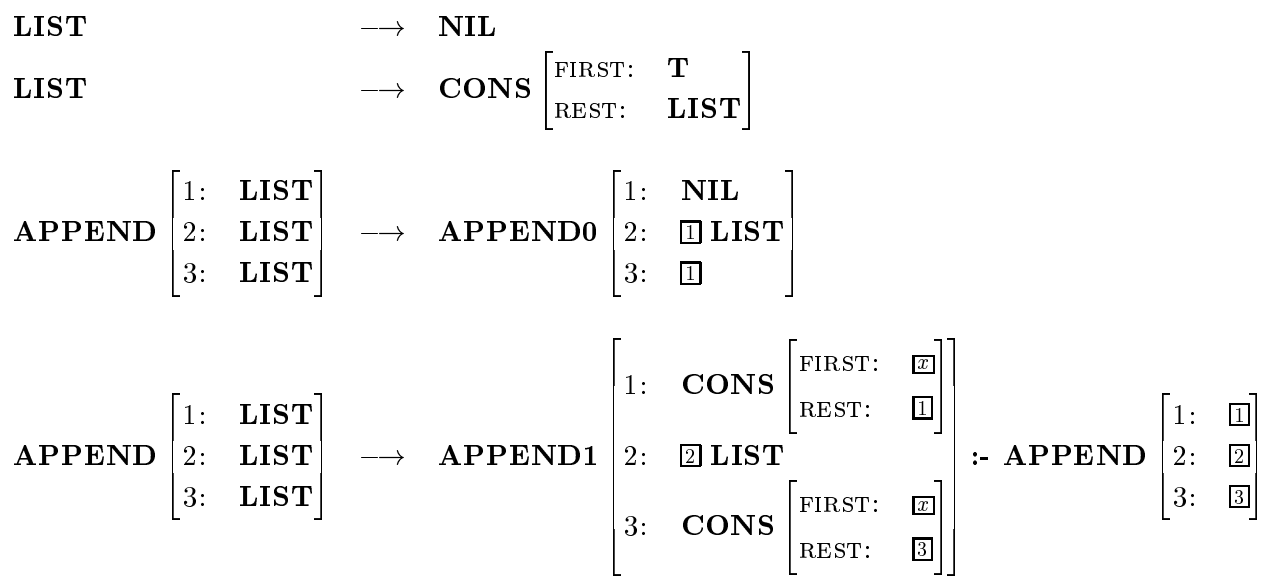


Figure 6: Rewrite rules for LIST and APPEND

4.3 TFS and MT

The translation process between languages is seen as a process of complex interactions among several levels (syntactic, lexical, semantic,...) of linguistic description, which can be modelled as constraints among linguistic levels (relations, in this framework). Thus, translating consists of simultaneous satisfaction of constraints from all levels. Obviously, relations are defined via TFS specifications. To produce a translation system between two languages, the following relations need to be defined :

1. Semantic classification of linguistic objects
2. Syntactic classification of linguistic objects
3. Relation between syntactic and semantic descriptions
4. Relation between SL and TL semantic descriptions

Relations 1-3 describe language-specific knowledge while relation 4 describes bilingual knowledge. It is important to point out that the former relations give a declarative view of the translation process, rather than a derivational view. The monolingual relations specify which constraints must be satisfied between utterances and semantic representations; they do not give a specific strategy for parsing. Likewise, bilingual knowledge is encoded defining which constraints must be satisfied between the SL and TL semantic descriptions. Thus, from the translation point of view, language-specific and cross-linguistic definitions are applied simultaneously to this structure; the final result is expected to be a set of utterances for both languages.

---

is a constraint-based language

In order to avoid complex structural transfer, cross-linguistic relations are suppossed to be stated at an abstract level of conceptual representation.

4.4 Comments

The TFS translation strategy is a reformulation of LFG translation by co-description.The formalism is very attractive, since linguistic knowledge is expressed in a very declarative way, thus avoiding a derivational strategy of the translation process (like CAT2 and MIMO2). Furthermore, it might be possible to add new relations (for instance, common sense knowledge,...) to the already existing ones.

A serious drawback of the formalism is its efficiency; since the system generates a rule for any type-subtype link, and type hierarchies are expected to have a great deal of nodes, it can be easily seen that we could end up having a huge set of rewrite rules. Furthermore, since the system generates a rewrite step for every applicable rule, it is expected to obtain (possibly too many) multiple solutions for a given query.

The TFS formalism for MT has not been developed yet to handle complex or difficult translations. In other words, though the formalism is very appealing, it is still in a very preliminary state of development. Its efficiency problems may represent a serious drawback for extended implementations.

5 MIMO2

5.1 The basic Model

([van Noord *et al.*, 1991],[van Noord, 1993]) proposes a model where the translation relation between two languages is defined as the composition of three reversible relations. The first relation deals with the phonological and semantic representation of SL, the second deals with both SL and TL semantic representations and the third deals with the TL semantic and phonological representation. Each translation relation is defined by a constraint-based grammar [3]. The second grammar is the transfer component, as commonly understood.

Figure 7: Reversible translation system.

[3] According to [van Noord, 1993], a grammar is reversible if it is capable of both parsing and generation on the basis of a single characterization of the relation between semantic and phonological structures; see [van Noord, 1993] for motivation for reversible grammars

Each Language may have its own semantic representation formalism, though it is assumed to be some kind of formal logic. Thus, the transfer component functions as an interface between both representations , and grammars can be developed independently from each other and only under monolingual considerations.

5.2 Transfer in MIMO2

As far as the transfer component is concerned, it should be mentioned that while the output of the parsing phase is a feature structure containing morphological, syntactic, and semantic information, only the semantic information is input to transfer. Conversely, the output of the transfer component is a semantic representation, which is input for the generation phase, whose role is thus to fill in other values, and ultimately produce a string. This implements a particular theory of translation, which has been also employed in Alep.

The key assumption for the transfer component is that translation of some structure is defined in terms of the translations of the parts of that structure.[4] As we noted, the SL semantic representation is input for the transfer component. Some of these representations may be related in a straightforward way to their TL equivalents (figure 9); others may be related in a more complicated way. For instance, consider the following rule (taken directly from [van Noord, 1993]), which translates 'open_fire_on' into 'het_vuur_openen_op':

$$\text{sign}\left(\begin{bmatrix} \text{GB:} & \begin{bmatrix} \text{SORT} & \text{binary} \\ \text{PRED} & \text{open\_fire\_on} \\ \text{ARG1} & \text{G1} \\ \text{ARG2} & \text{G2} \\ \text{NEG} & \text{Neg} \end{bmatrix} \\ \text{NL} & \begin{bmatrix} \text{SORT} & \text{binary} \\ \text{PRED} & \text{het\_vuur\_openen\_op} \\ \text{ARG1} & \text{N1} \\ \text{ARG2} & \text{N2} \\ \text{NEG} & \text{Neg} \end{bmatrix} \end{bmatrix}\right) \text{ :- } \text{sign}\left(\begin{bmatrix} \text{GB} & \text{G1} \\ \text{NL} & \text{N1} \end{bmatrix}\right), \text{sign}\left(\begin{bmatrix} \text{GB} & \text{G2} \\ \text{NL} & \text{N2} \end{bmatrix}\right).$$

Figure 8: A rule that translates 'open_fire_on' into 'het_vuur_openen_op'

This rule (figure 8) shows the main MIMO2 translation strategy: translation of an argument structure is composed of the translation of its arguments. Note that we could use the power of unification grammars to "thread" translation relevant parameters, such as style and subject field....

[4]Some types of non-compositional translations can also be handled by a transfer component. See [van Noord, 1993] for details.

$$\text{sign}\left(\begin{bmatrix} \text{GB} & \begin{bmatrix} \text{SORT} & \text{nullary} \\ \text{PRED} & \text{soldier} \\ \text{NUM} & \text{Num} \end{bmatrix} \\ \text{NL} & \begin{bmatrix} \text{SORT} & \text{nullary} \\ \text{PRED} & \text{militair} \\ \text{NUM} & \text{Num} \end{bmatrix} \end{bmatrix}\right).$$

Figure 9: A bilingual lexical entry.

5.3 Comments

MIMO2 is very similar to the PATR-II style FS formalism, which means that it cannot implement aspects of more modern theories (HPSG, for instance), though it should be mentioned that FS representing semantic information are typed: in the example above, the sort type specifies how many arguments the predicate takes.

The main drawback of this approach to MT is that the semantic representation generates too many solutions; for instance, it generates the active and the passive form of a given semantic FS, unless our grammar deals only with one of the forms, which does not seem reasonable. The same applies to word order and other phenomena, which means that we need to enrich our semantic representation with syntactic features in order to produce some notion of best translation. In other words, the problem is to ensure a sufficiently rich semantic representation which is not too "far away" from its syntactic realization.

MIMO2 has been used to translate between pairs of these languages: Dutch, English and Spanish. It remains to be seen how powerful this technique is for achieving reasonably good translations.

6 TAGS

[Egedi *et al.*, 1994] present a prototype system for MT between English and Korean which is implemented in the Synchronous Tree Adjoining Grammar (STAG) Formalism ([Shieber and Schabes, 1990]), an extension of Lexicalized ([Schabes *et al.*, 1988]), Feature Based ([Vijay-Shanker *et al.*, 1991]) Tree Adjoining Grammars (FB-LTAG) ([Joshi *et al.*,1975]).

6.1 The Formalism

6.1.1 Primitive elements

The primitive elements of the standard TAG formalism are elementary trees, which are of two types:

- initial trees
- auxiliary trees.

These trees form the basic building blocks of the formalism; operations of adjunction and substitution build derived trees from elementary trees.

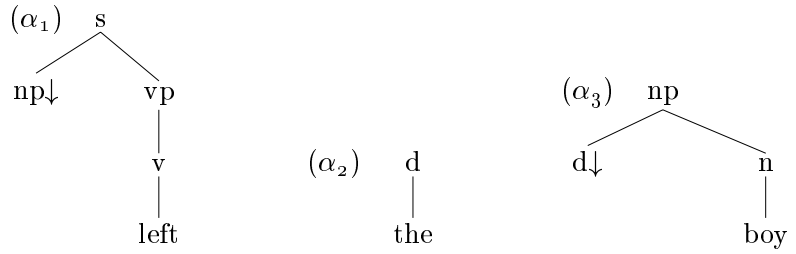


Figure 10: Examples of initial trees

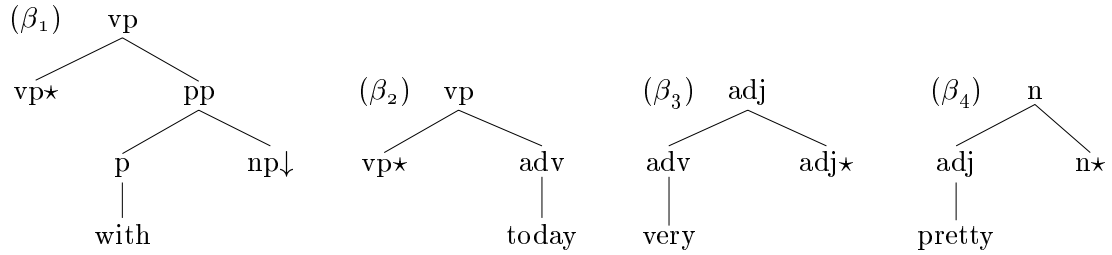


Figure 11: Examples of auxiliary trees

Initial trees are minimal linguistic structures that contain no recursion, i.e. trees containing the phrasal structure of simple sentences, NPs, PPs , and so forth. Technically, they have the following properties :

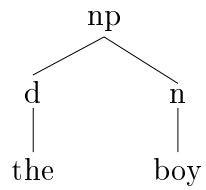
- all internal nodes are labeled by non-terminal
- all leaf nodes are labeled by terminals or by non-terminals nodes marked for substitution ($\downarrow$)

An auxiliary tree is defined as an initial tree, except that exactly one of its boundary nodes must be marked as foot node ($*$). The foot node must be labelled with a non-terminal symbol which is the same as the label of the root node. Auxiliary trees are meant to represent recursive structures (e.g. adverbials).

Each internal node of an elementary tree is associated with two feature structures, the top and the bottom. The bottom FS contains information relating to the subtree rooted at the node, and the top FS contains information relating to the supertree tree at that node. Substitution nodes have only a top FS, while all other nodes have both a top and bottom FS.

Substitution and adjunction are the only permitted operations on trees for building derived trees: in the substitution operation, a node marked for substitution in an elementary tree is replaced by another elementary tree whose root label is the same as the non-terminal. The top FS of the node results from the unification of the top FS of the two original nodes, while the bottom FS of the new node is simply the bottom FS of the root node of the substituting tree (since the substitution tree has no bottom feature). For example, substituting α_2 in α_3 gives the following tree[5]:

(11)



Only initial trees and derived trees can be substituted in another tree (note that in the example above no top and bottom FS are present).

In an adjunction operation, an auxiliary tree is inserted into an elementary tree. The root node and foot nodes of the auxiliary tree must match the node label at which the tree adjoins. The node being adjoined to splits, and its top FS unifies with the top FS of the foot node of the auxiliary tree, while its bottom FS unifies with the bottom FS of the foot node of the auxiliary tree.

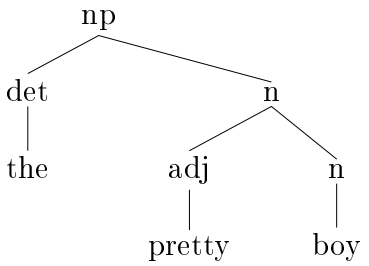


Figure 12: Result of adjoining β_4 at the node labelled 'n', of the derived tree np(d(the),n(boy)).

Feature Structures may have syntactic as well as semantic information for selectional purposes; syntactic and semantic features constrain the type of trees a lexical item may select. For instance, the Xtag system [Doran *et al.*, 1994] employs 30 features, (almost all of them syntactically motivated); (e.g. agr, case, conditional,wh,definite,...). Other features are semantically motivated and are employed for accurate lexical selection of polysemous verbs (for instance, the well known problem of translating *to wear* into the appropriate Japanase/Korean equivalent depending on the type of article being worn).

Apparently, syntactically motivated features encode monolingual knowledge, and semantic features may encode bilingual knowledge.

[5]Note that our substitution and adjunction examples are in standard TAGs (i.e. no unifications of top and bottom FS are involved), since our aim is to show the clearest possible examples of these operations

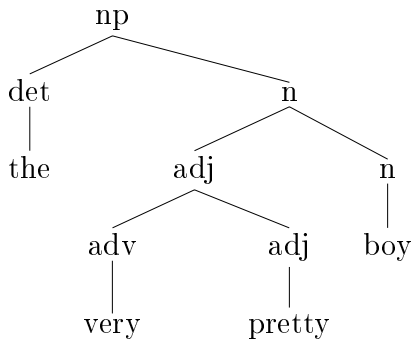


Figure 13: Result of adjoining β_3 at the 'adj' node of the derived tree np(d(the),n(adj(pretty),n(boy))).

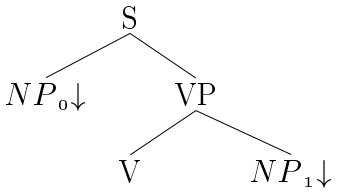
6.1.2 Grammar and Lexicon

A STAG grammar is a lexicalized grammar: the lexical items contain syntactic information, which means that grammar writers work in a bottom-up style, starting from words and stating how they interact. More specifically, a grammar consists of the following components:

- a tree database
- a syntactic lexicon
- lexical rules (called metarules in this framework).
- a transfer lexicon (for MT applications)

The tree database contains all the trees available to lexical items. There are individual trees, generally anchored by lexical items other than main verbs, and tree families, which represent subcategorization frames:

(12)



The example above represents the transitive tree family.

The syntactic lexicon specifies all the syntactic information about a lexical item, including which trees it selects and what constraints it places on these trees:

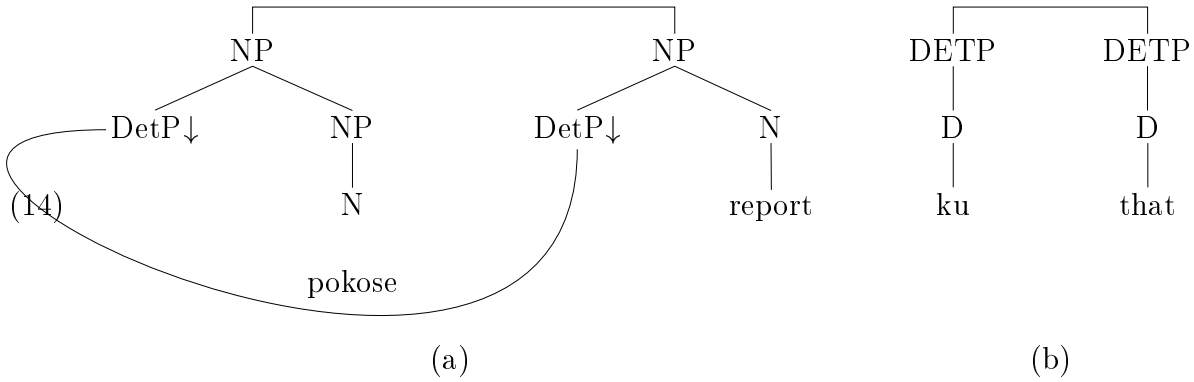
(13)

INDEX:	think	INDEX:	think
ENTRY:	think	ENTRY:	think
POS:	VERB	POS:	Verb
FRAME:	Sentential_Complement	FRAME:	Sentential_Complement
FS:	Indicative_Complement	FS:	Infinitive_Complement

The example above shows syntactic entries for *think* that require a sentential complement.

Lexical rules are used to capture the similarities between different trees for a lexical entry (for instance, they can cover morphological as well as syntactic phenomena).

The transfer lexicon specifies correspondences between lexicalized trees from the source and the target grammar:



The example above shows Lexicalized Synchronous trees for *report* and *that*, taken from [Egedi *et al.*, 1994]. Note how transfer rules can cover a large domain of locality.

6.2 TAGS and MT

As far as MT is concerned, the general translation procedure (taken from [Egedi *et al.*, 1994]) is the following:

1. First parse the SL sentence according to the source grammar. Feature unification is used for lexical selection purposes.
2. Each elementary tree in the source derivation tree (which roughly corresponds to a lexical item) is put in correspondence with an elementary tree of the target grammar using the transfer lexicon.
3. These trees obtained from phase 2 are then combined for creating the target sentence.

6.3 Comments

From the linguistic point of view, the formalism resembles HPSG in merging syntactic and semantic information in the representation of linguistic objects. Also the lexicon, carries most of the linguistic information in both approaches.

As far as MT is concerned, current work is in a very preliminary state; only simple translation divergences have been tested. Furthermore, there are not clear criteria for specifying the transfer lexicon and the generation process is far from being straightforward.

This approach is strikingly similar to Shake & Bake, (see next section). It has the same advantages and suffers from the same problems.

7 SHAKE & BAKE

Of all the approaches to MT, Shake & Bake (S&B) ([Whitelock, 1992],[Whitelock, 1994]) is the most lexicalist one. It addresses some of the problems encountered in other transfer approaches: lack of portability and modularity, as well as recursive mapping from a structure representing the source sentence into a structure representing the target sentence. To get around them, [Whitelock, 1992] proposes the independent writing of the monolingual grammars, and the description of the information specific to the translation process in the bilingual lexicon, a lexicon of linguistically rich bilingual signs. Accordingly, difficult translations, as those described in [Alshawi *et al.*, 1991], are directly handled in the bilingual lexicon.

7.1 The Method

The following is a brief summary of the S&B method:

1. The first step is the parsing of the Source Language (SL) sentence according to the SL grammar and lexicon only. The output is some kind of SL structure (usually, a tree) the leafs of which are the lexical entries which contain semantic information in form of a logical formula. In particular, they may contain variables filling roles in the semantics and indices to the appropriate fillers of these roles.

2. For each succesful parse, do the following:

   (a) Keep only the lexical entries and ignore the parse tree, or in other words, ignore the SL sentence structure.

   (b) Look up these entries in the bilingual lexicon. As a result, we will get a bag of Target Language (TL) entries, which are more instantiated than the entries in the TL lexicon, since they arise from the SL lookup.

(c) From the bag of words generate the TL sentence. The TL grammar and lexical entries drive this process.

3. When all the succesful parses of the SL have been processed, all possible translations have been found.

Let's see a simple example taken from [Beaven, 1992], namely, translation of the sentence *María leyó el libro* (Spanish) to *Mary read the book* (English):

The Spanish monolingual entries are the following:

(15)
$$\begin{bmatrix} \text{ORTHO} & \text{'María'} \\ \text{CAT} & \text{s/}\begin{bmatrix} \text{S} & \\ \text{I1} & \text{Sem1} \end{bmatrix} \\ \text{SEM} & I1 : role(I1, R1, F3), name(F3, maria) \cup Sem1 \\ \text{ARG0} & \text{F3} \end{bmatrix}$$

(16)
$$\begin{bmatrix} \text{ORTHO} & \text{leyó} \\ \text{CAT} & \text{s/}\begin{bmatrix} \text{S/}\begin{bmatrix} \text{S} \\ E : leer(E), role(E, agt, X), role(E, pat, Y) \end{bmatrix} \\ \text{SEM} \end{bmatrix} \\ \text{SEM} & \text{Sem} \\ \text{ARG0} & \text{E} \\ \text{ARG1} & \text{X} \\ \text{ARG2} & \text{Y} \end{bmatrix}$$

(17)
$$\begin{bmatrix} \text{ORTHO} & \text{el} \\ \text{CAT} & \text{s/}\begin{bmatrix} \text{N} & \\ \text{I1} & \text{Sem1} \\ \text{S} & \\ \text{I2} & \text{Sem2} \end{bmatrix} \\ \text{SEM} & I2 : definite(I1) \cup Sem1 \cup Sem2 \\ \text{ARG0} & \text{I1} \end{bmatrix}$$

(18)
$$\begin{bmatrix} \text{ORTHO} & \text{libro} \\ \text{CAT} & \text{n} \\ \text{SEM} & L : libro(L) \\ \text{ARG0} & \text{L} \end{bmatrix}$$

The English monolingual entries are the following:

(19)
$$\begin{bmatrix} \text{ORTHO} & \text{read} \\ \text{CAT} & \text{s/}\begin{bmatrix}\text{NP}\\ \text{Y:SEM2}\\ \text{NP}\\ \text{X:SEM1}\end{bmatrix} \\ \text{SEM} & E : reading(E), role(E, agt, X), role(E, pat, Y) \cup Sem1 \cup Sem2 \\ \text{ARG0} & \text{E} \\ \text{ARG1} & \text{X} \\ \text{ARG2} & \text{Y} \end{bmatrix}$$

(20)
$$\begin{bmatrix} \text{ORTHO} & \text{'Mary'} \\ \text{CAT} & \text{np} \\ \text{SEM} & F3 : name(F3, mary) \\ \text{ARG0} & \text{F3} \end{bmatrix}$$

(21)
$$\begin{bmatrix} \text{ORTHO} & \text{the} \\ \text{CAT} & \text{np/}\begin{bmatrix}\text{N}\\ \text{I:SEM}\end{bmatrix} \\ \text{SEM} & I : definite(I1) \cup Sem1 \\ \text{ARG0} & \text{I1} \end{bmatrix}$$

(22)
$$\begin{bmatrix} \text{ORTHO} & \text{book} \\ \text{CAT} & \text{n} \\ \text{SEM} & B.book(L) \\ \text{ARG0} & \text{B} \end{bmatrix}$$

The bilingual entry for *read-leyó* is the following:

(23)
$$\begin{bmatrix} \text{SPANISH} & \boxed{16} \begin{bmatrix} \text{ARG0} & \text{E} \\ \text{ARG1} & \text{X} \\ \text{ARG2} & \text{Y} \end{bmatrix} \\ \text{ENGLISH} & \boxed{19} \begin{bmatrix} \text{ARG0} & \text{E} \\ \text{ARG1} & \text{X} \\ \text{ARG2} & \text{Y} \end{bmatrix} \end{bmatrix}$$

After parsing 'María leyó el libro' we have:

(24)
$$\begin{bmatrix} \text{ORTHO} & \text{'María leyó el libro'} \\ \text{CAT} & \text{s} \\ \text{SEM} & E : leer(E), role(E, agt, F3), role(E, pat, L), def(L), libro(L), name(F3, maria) \end{bmatrix}$$

After the bilingual lookup, the semantics of the corresponding TL signs have become more instantiated (new instantiations are indicated by boldface); the following example is the English entry for *read*:

(25)
$$\begin{bmatrix} \text{ORTHO} & \text{read} \\ \text{CAT} & \text{s/}\begin{bmatrix}\text{NP}\\ \mathbf{L}\text{:Sem2}\\ \text{NP}\\ \mathbf{F3}\text{ :Sem1}\end{bmatrix} \\ \text{SEM} & E : \mathit{reading}(\mathbf{E}), \text{role}(\mathbf{E}, \text{agt}, \mathbf{F3}), \text{role}(\mathbf{E}, \text{pat}, \mathbf{L}) \cup \text{Sem1} \cup \text{Sem2} \\ \text{ARG0} & \mathbf{E} \\ \text{ARG1} & \mathbf{F3} \\ \text{ARG2} & \mathbf{L} \end{bmatrix}$$

Then, generating from the TL signs is to consider each possible permutation of the signs and try to parse them.

7.2 Comments

The advantatges of this approach are the following:

- SL and TL Grammars and Lexicons are developed under monolingual considerations only:

  The monolingual components are written independently from each other, using only monolingual considerations. They can be used for parsing and generation, not exclusively for MT purposes, thus achieving modularity and reusability. A single monolingual grammar serves for a multi-lingual system; only the bilingual lexicons need developing for each pair of languages.

- Maximal lexicalization in the transfer process:

  Transfer divergences are directly handled within the bilingual lexicon. Thus, Shake & Bake is a good example of the lexicalization trend in modern NLP.

The problems are the following:

- Bag Generation is NP-Complete
- Can we put all translation constraints in the lexical entries?
- The Bilingual Lexicon: lexical lookup and lexical disambiguation
- Lexical Rules and generalized Head Switching

Bag generation is NP-complete ([Brew, 1992]) : For an input of size n (where n is the number of items in the bag) all permutations (n!) have to be explored to find a correct answer. In the worst case, they must be explored to assure that there is no possible answer. Several authors have employed some techniques to improve bag generation efficiency ([Popowich, 1994],[Poznański *et al.*, 1995]); the most promising one is [Poznański *et al.*, 1995]; its complexity is polynomial (O(n4)), which means that generation turns from an intractable problem into a tractable one. This reduction in theoretical complexity is mainly achieved by placing constraints on the power of the target grammar when operating on instantiated signs; this permits the use of a greedy algorithm and avoids backtracking. [Poznański *et al.*, 1995] are currently investigating the mathematical characterization of grammars which obey these constraints. Another direction for future reasearch may be Head-driven bag generation: we could start the generation process from the head, (provided it could be determined from the bag of words) ; this would mean a more linguistically based generation process, which cuts the search space down and thus improves efficiency.

Can we put all translation constraints in the lexical entries? In order to generalize this approach to other relevant aspects of the translation process (for instance, "style"), we need to assume that this information is projected from the lexical entries; it remains to be seen whether this is always possible or linguistically satisfactory ([van Noord, 1993]).

Little attention has been paid so far to the lexical disambiguation process; [Beaven, 1992] assumes that the semantics of each lexical entry (which have become highly instantiated during the SL parsing) together with other features (orthography, for instance) give enough information to find the corresponding TL word (It is well worth pointing out that the semantics of SL and TL do not need to share predicate names, role names, etc...) . We do believe that this process deserves more attention.

Lexical rules are employed for dealing with morphological aspects and generalized head switching; for instance, consider the Spanish sentence *María cruzó el río nadando* which translates into English as *Mary swam across the river* (literally *Mary crossed the river swimming*). In the S&B approach, we need to specify that *cruzó* corresponds to *across*, and *nadando* to *swam*, in addition to the standard correspondences *cruzó-crossed* and *swam-nadó*. As this is part of a quite general pattern, for avoiding a huge lexicon of bilingual signs we need some general lexical rule applicable to many verbs of movement.

Finally, it should be mentioned that though most of the known work on S&B deals with UCG (Unification Categorial Grammar) and untyped Feature Structures, nothing prevents the use of other grammar formalisms (like HPSG) and typed Feature Structures.

8 ALEP

This section discusses the expressive power of the Alep formalism and the translation strategies it permits, though at the current state of development of the project, all that can be said about the issue is only a tentative evaluation.

8.1 Expressivity of the Formalism

We assume the reader is familiar with the Alep formalism. The expressive power of Alep can be seen in the following transfer rule, which transfers the input structure root node while calling for recursive translation on the semantic feature structure:[6]

(26)
```
trule(de,en,<=>,
sign:{
          syn => major:{
                   unit_num => NUM1,
                   bar => max,
                   punct => yes,
                   head => v:{},
                   subj => [] },
          sem => SEM1 } < _,
sign:{
          syn => major:{
                   unit_num => NUM2,
                   bar => s,
                   punct => yes,
                   head => v:{},
                   subj => [],
                   subcat => [] },
          sem => SEM2 },
[NUM1 = NUM2,
 SEM1 == SEM2]).
```

The left hand side (lhs) of this rule matches the relevant characteristics of the SL sign after the monolingual analysis, and the right hand side (rhs) shows what the TL sign, which will be input for the generation component, should look like.

Other conditions for applying the rule are stated using the "=" and "==" operators. The first one is equivalent to the Prolog "=" matching operator, and the second one calls for recursive translation on the appropriate features, thus completing the TL sign that serves for input of the synthesis component.

8.2 Transfer Methodology and tentative Evaluation

The main advantage of the formalism is that it allows specifcation of both syntactic and semantic information of the SL and TL signs in writing transfer rules. This expressivity permits several translation strategies:

[6] Figures in this section have been taken from [Theofilidis, 1994]

1. The head of the SL sign contains all of the relevant information for translation; this information could appear within the semantic feature structure of the head or within a syntactic-semantic feature structure, as shown in figure 14.

```
trule(de,en,<=>,
sem_fs:{
        gov => v_sem:{
                pred => geben,
                predtype => PT1 },
        args => ARGS1 => trival:{},
        mods => MODS1 },
sem_fs:{
        gov => v_sem:{
                pred => give,
                predtype => PT2 },
        args => ARGS2 => trival:{},
        mods => MODS2 },
[ ARGS1 == ARGS2,
  MODS1 == MODS2,
  PT1 = PT2]).
```

Figure 14: This rule translates the German predicate *geben* into the corresponding English predicate *give*

   In other words, from the parse tree, only its head is input for transfer; the rest of the tree is not needed in this process. Thus, the problem is to ensure that a rich enough semantics for translation is collected in the head of the SL sign (see figure 17).

2. The necessary information for the transfer component is distributed among the head of the SL sign and its daughters, though it is always present in the same feature structure (a semantic FS or a syntactic-semantic FS):

   As shown in figure 15, translation operates in a recursive fashion; it can be thought of as a composition of translations on the head and its daughters (see figure 16).

3. It also seems possible to simulate also the S&B approach (or any other lexicalist approach) with the Alep formalism: in order to do this, transfer rules should only operate on the leafs of the SL sign (see figure 18), thus building a bag of signs that are input for the synthesis component. Thus, the synthesis component should assume no order among daughters of the head of the synthesis rules. Obviously, as it stands, this makes the synthesis component intractable.

4. It seems possible to simulate TFS descriptions in Alep, though at a high cost; since the functionality and well-formedness conditions of both type systems is different,

```
trule(de,en,<=>,
sem_fs:{
        gov => GOV1,
        args => ARGS1,
        mods => MODS1 },
sem_fs:{
        gov => GOV2,
        args => ARGS2,
        mods => MODS2 },
[ GOV1 == GOV2,
  ARGS1 == ARGS2,
  MODS1 == MODS2 ]).
```

Figure 15: Translation by letting semantic feature structures be translated by a general (recursive) rule

we have to write TFS rewrite rules as Alep rules. Furthermore, each Alep rule should make no real use of the Alep type system, which means that each rule should have a 'false' type (i.e a type which permits compilation in the Alep system but which is useless in run time). Unfortunately, that means that the appropriateness and well formedness conditions on Feature Structures and types must be directly handled by the user.

9 CONCLUSIONS

This conclusion contains a number of remarks on the transfer approaches we have reviewed:

- The transfer approaches we have reviewed can be classified as follows:
  - Lexicalist approaches: Shake & Bake, STAGS, Alep?
  - Non-lexicalist approaches: CAT2, LFG, TFS, MIMO2, Alep. Some of these assume a single structure containing all necessary information for the translation process (Alep); others assume translation is compositional, and thus, the relevant information for translation is distributed in several structures rather than collapsed into a single structure (MIMO2,ALEP). The rest (TFS,LFG) are the most declarative: they just put linguistic (monolingual and bilingual) constraints on the transfer process.
- Except CAT2, the rest of approaches deal only with single transfer problems. The development of a translation system capable of handling complex translations is an element of further research.

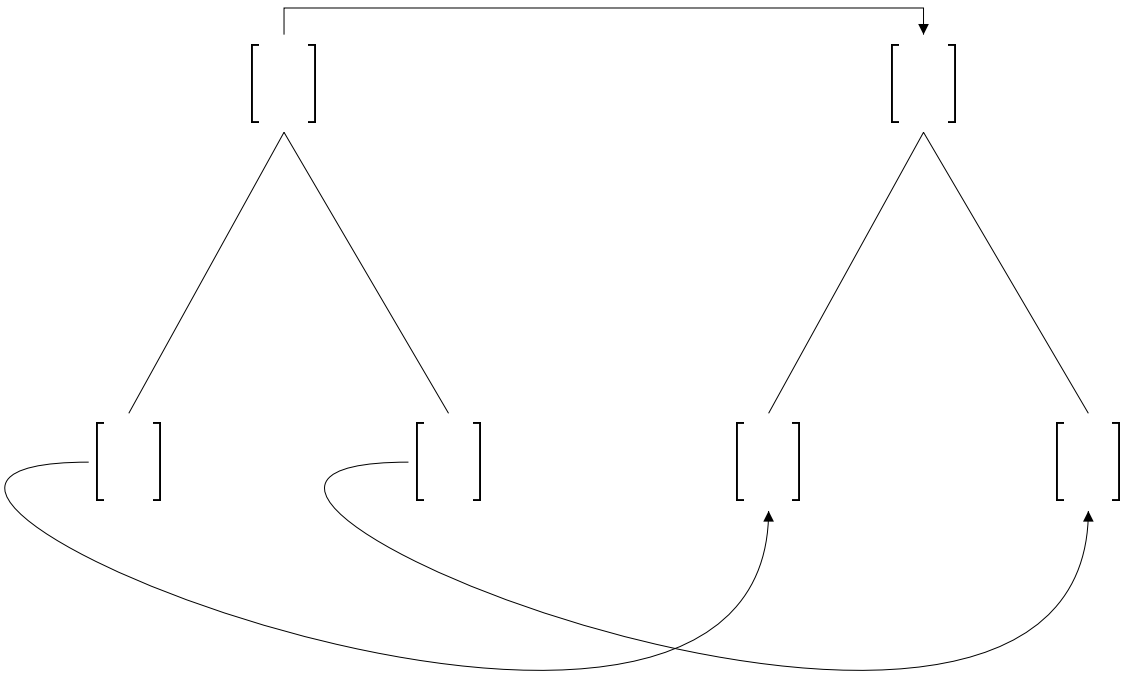


Figure 16: Recursive translation along the parse tree.

- Only TFS makes an essential use of type hierarchy in the translation process. In the rest of approaches, the type hierarchy (if present) is only monolingually motivated; type hierarchy is used for putting constraints on the well formedness of signs according to monolingual considerations, but it is not used in the transfer process.
- The Alep formalism appears to be powerful enough to test several translation strategies, though the actual tests are to be camed out in the course of project MLAP-9315.

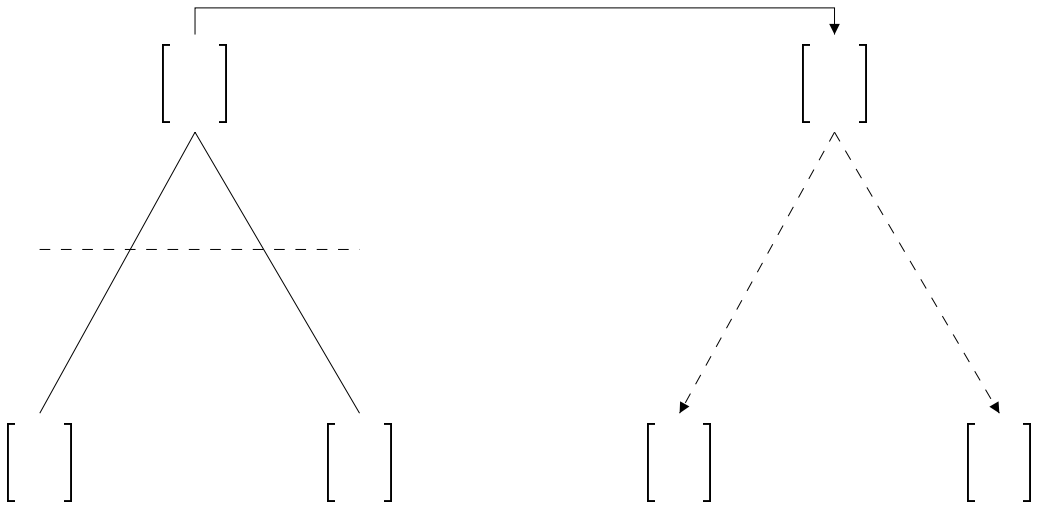


Figure 17: Translation of the top node

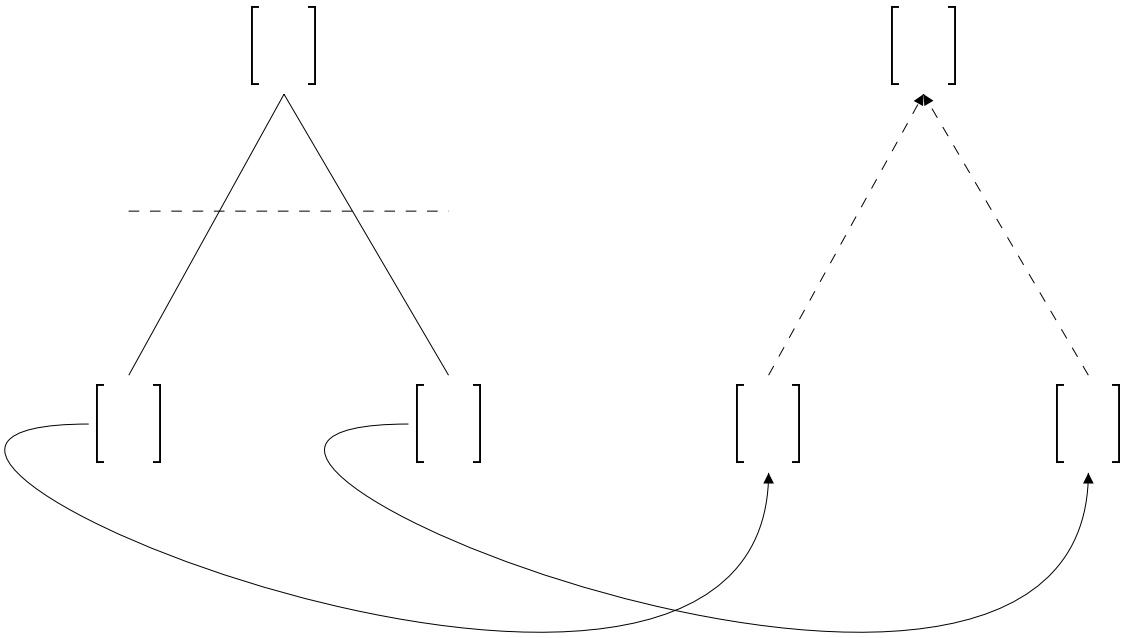


Figure 18: Translation of the terminal nodes

References

[Alshawi *et al.*, 1991] H.Alshawi, D.J.Arnold, R.Backofen, D.M.Carter, J.Lindop, K.Netter, S.Pulman, J.Tsuji and H.Uszkoreit. Eurotra ET6/1:Rule Formalism and Virtual Design Study (final Report), CEC 1991.

[Arnold and des Tombe, 1987] D.Arnold, L.des Tombe. Basic theory and methodology in Eurotra. In S.Nirenburg, editor, *Machine translation:theoretical and methodological issues*, Cambridge University Press, 1987.

[Beaven, 1992] J.L.Beaven. *Lexicalist Unification-based Machine Translation*, Ph.D thesis, University of Edinburgh, Edinburgh, 1992

[Brew, 1992] C.Brew. Letting the cat out of the bag: generation for Shake & Bake MT, In *Proceedings of the 14th International Conference on Computational Linguistics (COLING-92)*, pages 610-616. Nantes, France, 1992.

[Carpenter, 1992] B.Carpenter. *The Logic of Typed Feature Structures*. Cambridge University Press, 1992.

[Doran *et al.*, 1994] C.Doran, D.Egedi, B.A.Hockey, B.Srinivas and M.Zaidel. Xtag system - a wide coverage grammar for English. In *Proceedings of the 17th International Conference on Computational Linguistics (COLING'94)*, Kyoto, Japan, August 1994.

[Egedi *et al.*, 1994] D.Egedi, M.Palmer, H.Park and A.Joshi. Korean to English Translation Using Synchronous TAGs, In *Proceedings of the First Conference of the Association for Machine Translation in The Americas*, pages 48-55, Columbia, U.S.A.

[Emele *et al.*, 1992] M.Emele, U.Heid, S.Momma and R.Zajac. Interactions between Linguistic Constraints: Procedural vs. Declarative Approaches. In *Machine Translation*,1992

[Joshi *et al.*,1975] A.Joshi, L.Levy,L and M.Takahashi. Tree Adjunct Grammars. In *Journal of Computer and System Sciences*, 1975.

[Kaplan *et al.*, 89] R.Kaplan, K.Netter, J.Wedekind and A.Zaenen. Translation by Structural Correspondences. In *Proceedings of the Fourth Conference of the European Chapter of the Association for Computational Linguistics*, pages 272-281, Manchester,1989.

[Popowich, 1994] F.Popowich. Improving the Efficiency of a Generation Algorithm for Shake and Bake Machine Translation using Head-Driven Phrase Structure Grammar. Technical Report CMPT-TR 94-07, School of Computing Science, Simon Fraser University, Burnaby British Columbia, Canada.

[Poznański *et al.*, 1995] V.Poznański, J.L.Beaven and P.Whitelock. An Efficient Generation Algoritm for Lexicalist MT. In *Proceedings of the ACL-95.*

[Sadler and Thompson, 1991] L.Sadler and H.S.Thompson. Structural Non-Correspondence in Translation. In *Proceedings of the 5th Conference of the European Chapter of the Association for Computational Linguistics*, pages 293-298, Berlin, Germany,1991.

[Sadler, 92] L.Sadler. Co-description and translation. In Frank van Eynde, editor, *Linguistic Issues in Machine Translation*, Communication in Artificial Intelligence Series, Pinter Publishers, 1993.

[Schabes *et al.*, 1988] Y.Schabes, A.Abeille and A.Joshi. Parsing strategies with 'lexicalized' grammars: Application to Tree Adjoining Grammars. In *Proceedings of the 12th International Conference on Computational Linguistics (COLING-88)*, Budapest, Hungary, 1988.

[Shieber and Schabes, 1990] S.Shieber and Y.Schabes. Synchronous Tree Adjoining Grammars. In *Proceedings of the 13th Conference on Computational Linguistics (COLING'90)*, Helsinki, Finland, 1990.

[Sharp, 1991] Sharp, R. CAT2: An Experimental Eurotra Alternative. In *Machine Translation 6*, pages 215-228, 1991.

[Srinivas *et al.*, 1994] B.Srinivas , D.Egedi, C.Doran and T.Becker. Lexicalization and Grammar Development, In *Proceedings of KONVENS-94*, Vienna, Austria, 1994.

[Theofilidis, 1994] A.Thefilidis. ET9/1 Lingware Development. Final Documentation. CEC and IAI, 1993-1994.

[van Noord *et al.*, 1991] G.van Noord, J. Dorrepaal, P. van der Eijk, M. Florenza, H. Ruessink and L. des Tombe. An Overview of MiMo2. In *Machine Translation 6*, pages 201-214, 1991.

[van Noord, 1993] van Noord, G.*Reversibility in Natural Language Processing*, Ph.D thesis, University of Utrecht, 1993.

[Vijay-Shanker *et al.*, 1991] K.Vijay-Shanker and A.Joshi. Unification based Tree Adjoining Grammars, In J.Wedwkind, editor, *Unification-based Grammars.* MIT Press, Cambridge, M.A., 1991.

[Whitelock, 1992] P.Whitelock. Shake and Bake Translation. In *Proceedings of COLING 92*, pages 610-616.

[Whitelock, 1994] P.Whitelock. Shake and Bake Translation, In C.Rupp, M.Rosner and R.Johnson, editors, *Constraints, Language and Computation*, Academic Press, 1994.

[Zajac, 1993] R.Zajac. Issues in the Design of a Language for Representing Linguistic Information Based on Inheritance and Feature Structures, In T. Briscoe, A.Copestake,V. da Paiva (editors), *Inheritance,Defaults and the Lexicon*, Cambridge University Press, 1993.