Constraints, tlinks and MT

ACQUILEX II Working paper 16

Ann Copestake
Computer Laboratory, University of Cambridge
Ann.Copestake@cl.cam.ac.uk
November 16, 1994

1 Introduction

This document is intended as a description of the current mechanisms in the ACQUILEX LKB for constraint resolution and for the application of tlinks. It does not contain any detailed account of particular translation mismatches etc — these will be provided (at some point) in Copestake and Sanfilippo (in preparation). This document incorporates some information from previously published sources, on tlinks in particular, in order to provide a single consistent description (see, in particular, Copestake and Sanfilippo, 1993). Eventually this document will form part of the full description of the LKB.

2 Types and constraints

2.1 Well-formedness

The notion of well-formedness which has been used in the LRL up till now is relatively weak.[1]It is possible to have well-formed feature structures which cannot be extended to a structure which contains only most specific types. One disadvantage of this is that it is not possible, in general, to enforce cooccurrence restrictions, even of a quite limited sort. For example, suppose we have a type **t** with two features, F and G which both take boolean values, but we wish to state that when F has value **true**, G must have value **false** and vice versa. We might try to specify this by defining two subtypes of **t**, **t1** and **t2** with the following constraints:

$$\begin{bmatrix} \textbf{t1} \\ \text{F} = \textbf{true} \\ \text{G} = \textbf{false} \end{bmatrix} \qquad\qquad \begin{bmatrix} \textbf{t2} \\ \text{F} = \textbf{false} \\ \text{G} = \textbf{true} \end{bmatrix}$$

A type system in the LKB is assumed to be completely specified, in that, for example, two types can only be unified if they have a specified meet (greatest lower bound). Therefore we might assume that this fragment of a type system specifies the restriction that if F is **true** then G is **false**, and if F is **false** then G is **true**, since **t** has only the two subtypes **t1** and **t2**. However this is not the case. Consider the following structure:

$$\begin{bmatrix} \textbf{t} \\ \text{F} = \textbf{true} \\ \text{G} = \textbf{true} \end{bmatrix}$$

This is still a well-formed feature structure, by the definition given in Copestake (1992) despite the fact that it cannot be extended to be a well-formed structure with a type corresponding to that of any leaf node in the type hierarchy. Thus we have no good way of enforcing cooccurrence restrictions. Note that an equivalent result applies to Carpenter's appropriateness specifications (see Carpenter, 1992:144f).

To implement cooccurrence restrictions in this way we need a notion of typing which enforces the idea that the type system is complete. We can do this by checking that a feature structure can be extended to a well-formed structure where every type is a leaf type (cf the HPSG concept of a sort-resolved feature structure (Pollard and Sag, forthcoming)). We refer to such a feature structure as 'ultimately well-formed' and we can enforce such cooccurrence restrictions by checking for ultimate well-formedness.

[1]This section is mostly taken from Copestake (1992).

Definition 1 *(Ultimate well-formedness) A feature structure F is said to be ultimately well-formed iff it subsumes some well-formed type-resolved feature structure F', where a feature structure $\langle Q, q_0, \delta, \alpha \rangle$ is type-resolved if the types of all its nodes are leaf types, that is for all nodes $q \in Q$ there is no type t such that $t \sqsubset \alpha(q)$*

However, in order to check for ultimate well-formedness, we have to consider not only the immediate subtypes of a type, but their subtypes, and so on, and recursively apply this process to every node in the feature structure.

The notion of a type-resolved feature structure has similarities with Carpenter's description of ground feature structures (Carpenter, 1992:144f). A ground feature structure according to Carpenter is one which may not be further specialised, that is there is no possible (well-formed) feature structure which is strictly subsumed by a ground feature structure. However, a type-resolved feature structure by the definition above is only fully specified with respect to path values; a more specific feature structure may be possible if reentrancy is introduced. Our definition is equivalent to Zajac's (1991, in press) definition of ground feature structure.

3 Constraint solving

If instead of simply checking whether a feature structure is ultimately well-formed by ensuring that it can be extended to at least one type-resolved feature structure, we enumerate all such type resolved structures, we have a notion of type inference which is, in effect, a a constraint resolution system. This is a very powerful concept, which can be used, for example, to drive parsing and generation. Carpenter (1992:227f) and Zajac (1991, in press) have discussed how typed feature structures may be viewed as encoding constraint systems. The basic concept used in the LKB constraint system is similar, thus an LKB FS can be regarded as a description of a set of type-resolved or ground feature structures as defined above. Solving a constraint FS involves enumerating all the ground feature structures of which it is a description. Unlike Zajac and Carpenter, we assume, as before, that the constraint on a type is itself simply a feature structure.[2]

For example, assume we have the following type system:

top ().

list (**top**).

e-list (**list**).

ne-list (**list**)
< `HEAD` > = **thing**
< `TAIL` > = **list**.

thing (**top**).

a (**thing**).

b (**thing**).

The FS specified as:

< `TAIL : TAIL` > = **e-list**.

[2]The LKB constraint resolution is equivalent to Carpenter's description, on the assumption of "coverage", that is we assume that the type system is complete in that the subtypes of a type completely cover all the possibilities. Carpenter allows disjunctive constraints, but the effect of these can always be mimicked in our system by adding types, so there is no difference in power. Thus in the example in the previous section, type **t** is in effect associated with a disjunctive constraint on the valid feature structures such that either F is **true** and G is **false** or G is **true** and F is **false**. Zajac allows conditional constraints, which again make the system more expressive but do not (as far as I can tell) add power since their effect can be obtained by the use of "junk slots" (see below and also Carpenter, 1992:238f).

can be solved in this type system to give the following ground FSs:

$$\begin{bmatrix} \textbf{ne-list} \\ \text{HEAD} = \textbf{a} \\ \text{TAIL} = \begin{bmatrix} \textbf{ne-list} \\ \text{HEAD} = \textbf{a} \\ \text{TAIL} = \textbf{e-list} \end{bmatrix} \end{bmatrix}$$

$$\begin{bmatrix} \textbf{ne-list} \\ \text{HEAD} = \textbf{a} \\ \text{TAIL} = \begin{bmatrix} \textbf{ne-list} \\ \text{HEAD} = \textbf{b} \\ \text{TAIL} = \textbf{e-list} \end{bmatrix} \end{bmatrix}$$

$$\begin{bmatrix} \textbf{ne-list} \\ \text{HEAD} = \textbf{b} \\ \text{TAIL} = \begin{bmatrix} \textbf{ne-list} \\ \text{HEAD} = \textbf{a} \\ \text{TAIL} = \textbf{e-list} \end{bmatrix} \end{bmatrix}$$

$$\begin{bmatrix} \textbf{ne-list} \\ \text{HEAD} = \textbf{b} \\ \text{TAIL} = \begin{bmatrix} \textbf{ne-list} \\ \text{HEAD} = \textbf{b} \\ \text{TAIL} = \textbf{e-list} \end{bmatrix} \end{bmatrix}$$

In contrast the FS with the description:

```
< HEAD > = a
< TAIL : HEAD > = b
```

has an infinite number of solutions, because the list can be extended indefinitely. Because of this possibility, constraint resolution is, in general, undecidable.

The algorithm used for type resolution in the LKB is quite straightforward (and not at all efficient!). We start at the initial node of the feature structure and attempt to "push down" its type to a leaf type. That is, for each subtype of the type, we check to see whether its constraint unifies with the given feature structure. If it does, we attempt to push down the unified structure. If we arrive at a leaf type, we add the new structure to the list of possibilities. If unification fails, we stop investigating that possibility. For example, when resolving:

$$\begin{bmatrix} \textbf{list} \\ \text{TAIL} = \begin{bmatrix} \textbf{top} \\ \text{TAIL} = \textbf{e-list} \end{bmatrix} \end{bmatrix}$$

the structure can only be a **ne-list**. [3] At the end of this phase, we will have a set of possibilities, with the initial node typed as a leaf type. In this case there is only one member of the set:

$$\begin{bmatrix} \textbf{ne-list} \\ \text{HEAD} = \textbf{thing} \\ \text{TAIL} = \begin{bmatrix} \textbf{top} \\ \text{TAIL} = \textbf{e-list} \end{bmatrix} \end{bmatrix}$$

For each of the possibilities, we attempt to push down all the nodes at the end of paths of length 1, effectively in parallel. In this case, we try to resolve **thing** and **top**. This results in the set of feature structures:

$$\begin{bmatrix} \textbf{ne-list} \\ \text{HEAD} = \textbf{a} \\ \text{TAIL} = \begin{bmatrix} \textbf{ne-list} \\ \text{HEAD} = \textbf{thing} \\ \text{TAIL} = \textbf{e-list} \end{bmatrix} \end{bmatrix}$$

$$\begin{bmatrix} \textbf{ne-list} \\ \text{HEAD} = \textbf{b} \\ \text{TAIL} = \begin{bmatrix} \textbf{ne-list} \\ \text{HEAD} = \textbf{thing} \\ \text{TAIL} = \textbf{e-list} \end{bmatrix} \end{bmatrix}$$

This process is then repeated with paths of length 2, and so on, until all the nodes in the feature structure are type resolved.

For a more complex example, assume that the following types are added to the system:

[3] This is slightly misleading since the actual algorithm the LKB uses to do type inference involves making the feature structure well-formed according to the original definition first, since this is a relatively efficient process. However the end effect is the same.

```
append (top)
   < LEFT > = list
   < RIGHT > = list
   < RESULT > = list.

append1 (append)
   < LEFT > = e-list
   < RIGHT > = < RESULT >.

append2 (append)
   < JUNK > = append
   < LEFT : HEAD > = < RESULT : HEAD >
   < LEFT : TAIL > = < JUNK : LEFT >
   < RIGHT > = < JUNK : RIGHT >
   < RESULT : TAIL > = < JUNK : RESULT >.
```

The idea is that the solutions to a goal feature structure of type **append** will be structures where the RESULT feature is the concatenation of the lists in the LEFT and RIGHT slots. This illustrates the use of "junk slots" which in effect allow for recursion. The type system encoding of append is equivalent to the Prolog code:

```
append([],L,L)
append([X|L1],L2,[X|L3]) :- append(L1,L2,L3)
```

where `[X|L1]` is the Prolog version of the head/tail notation — that is X is the head of the list and L1 the tail. In both cases there are two possibilities for append — the first case is the end condition where the left list is empty, in which case the right hand list is equal to the result. This identity is implemented by reentrancy in typed feature structures and by the shared variable, L, in Prolog. The second case is where the left hand list is not empty, in which case append is satisfied if the head of the result is equal to the head of the left hand list given that the tail of the left hand list, the whole of the right hand list and the tail of the result themselves satisfy append. This recursion is implemented in Prolog by the conditional statement and in the LKB type system by the use of the extra "junk" slot, which itself takes a feature structure of type **append**. Just as in Prolog, **append** can be used not just in the 'conventional' direction to concatenate two lists, but also to enumerate the possible solutions that will give a particular result or to give the results for partially instantiated structures.

For example, the following structure specifies part of each of the lists:

$$
\begin{bmatrix}
\textbf{append} & \\
\text{LEFT} = \begin{bmatrix} \textbf{ne-list} \\ \text{HEAD} = \textbf{a} \\ \text{TAIL} = \textbf{list} \end{bmatrix} \\
\text{RIGHT} = \begin{bmatrix} \textbf{ne-list} \\ \text{HEAD} = \textbf{thing} \\ \text{TAIL} = \begin{bmatrix} \textbf{ne-list} \\ \text{HEAD} = \textbf{c} \\ \text{TAIL} = \textbf{list} \end{bmatrix} \end{bmatrix} \\
\text{RESULT} = \begin{bmatrix} \textbf{ne-list} \\ \text{HEAD} = \textbf{thing} \\ \text{TAIL} = \begin{bmatrix} \textbf{ne-list} \\ \text{HEAD} = \textbf{b} \\ \text{TAIL} = \begin{bmatrix} \textbf{ne-list} \\ \text{HEAD} = \textbf{thing} \\ \text{TAIL} = \textbf{e-list} \end{bmatrix} \end{bmatrix} \end{bmatrix}
\end{bmatrix}
$$

The result of giving this as a goal to the constraint system is the following:

$$\left[\begin{array}{ll}\textbf{append2} & \\ \text{LEFT} = & \left[\begin{array}{l}\textbf{ne-list}\\ \text{HEAD} = \boxed{2}\ \textbf{a}\\ \text{TAIL} = \boxed{0}\ \textbf{e-list}\end{array}\right] \\ \text{RIGHT} = & \boxed{1}\left[\begin{array}{ll}\textbf{ne-list} & \\ \text{HEAD} = \textbf{b} & \\ \text{TAIL} = & \left[\begin{array}{l}\textbf{ne-list}\\ \text{HEAD} = \textbf{c}\\ \text{TAIL} = \textbf{e-list}\end{array}\right]\end{array}\right] \\ \text{RESULT} = & \left[\begin{array}{l}\textbf{ne-list}\\ \text{HEAD} = \boxed{2}\\ \text{TAIL} = \boxed{1}\end{array}\right] \\ \text{JUNK} = & \left[\begin{array}{l}\textbf{append1}\\ \text{LEFT} = \boxed{0}\\ \text{RIGHT} = \boxed{1}\\ \text{RESULT} = \boxed{1}\end{array}\right]\end{array}\right]$$

Constraint resolution is sufficiently powerful to allow grammars encoded in the type system to be exploited for conventional parsing, generation from logical form or generation from any partially specified structures. In such a type system, instead of conventional grammar rules, there are constraints on phrases which partially specify their daughters. For a trivial example, instead of having a grammar rule

```
S -> NP VP
```

the type **s** could be defined as having an attribute DTRS which takes a value which is a list of feature structures of type **np** and **vp**.

$$\left[\begin{array}{ll}\textbf{s} & \\ \text{DTRS} = & \left[\begin{array}{ll}\textbf{ne-list} & \\ \text{HEAD} = \textbf{np} & \\ \text{TAIL} = & \left[\begin{array}{l}\textbf{ne-list}\\ \text{HEAD} = \textbf{vp}\\ \text{TAIL} = \textbf{e-list}\end{array}\right]\end{array}\right]\end{array}\right]$$

I will sometimes use angle bracket notation to indicate a list with elements separated by commas, for example:

$$\left[\begin{array}{l}\textbf{s}\\ \text{DTRS} = \langle \textbf{np}, \textbf{vp} \rangle\end{array}\right]$$

Instead of the two rules:

```
VP -> Vtrans NP
```

and

```
VP -> Vintrans
```

we can define the type **vp** to have two subtypes **vp1** and **vp2** such that,

$$\left[\begin{array}{ll}\textbf{vp1} & \\ \text{DTRS} = & \left[\begin{array}{ll}\textbf{ne-list} & \\ \text{HEAD} = \textbf{vtrans} & \\ \text{TAIL} = & \left[\begin{array}{l}\textbf{ne-list}\\ \text{HEAD} = \textbf{np}\\ \text{TAIL} = \textbf{e-list}\end{array}\right]\end{array}\right]\end{array}\right]$$

$$\left[\begin{array}{ll}\textbf{vp2} & \\ \text{DTRS} = & \left[\begin{array}{l}\textbf{ne-list}\\ \text{HEAD} = \textbf{vintrans}\\ \text{TAIL} = \textbf{e-list}\end{array}\right]\end{array}\right]$$

If we now set **john** and **mary** to be subtypes of **np**, **sleeps** to be a subtype of **vintrans** and **likes** to be a subtype of **vtrans**, the feature structure:

$$\left[\ \textbf{s}\ \right]$$

will be resolved to generate structures (almost) isomorphic to the tree structures produced by the grammar rules:

$$\begin{bmatrix}\mathbf{s}\\ \text{DTRS} = \left\langle \mathbf{john}, \begin{bmatrix}\mathbf{vp}\\ \text{DTRS} = \langle\mathbf{sleeps}\rangle\end{bmatrix}\right\rangle\end{bmatrix}$$

$$\begin{bmatrix}\mathbf{s}\\ \text{DTRS} = \left\langle \mathbf{mary}, \begin{bmatrix}\mathbf{vp}\\ \text{DTRS} = \langle\mathbf{sleeps}\rangle\end{bmatrix}\right\rangle\end{bmatrix}$$

$$\begin{bmatrix}\mathbf{s}\\ \text{DTRS} = \left\langle \mathbf{john}, \begin{bmatrix}\mathbf{vp}\\ \text{DTRS} = \langle\mathbf{likes},\ \mathbf{mary}\rangle\end{bmatrix}\right\rangle\end{bmatrix} \quad \ldots$$

Naturally a much more complex series of constraints must be used to encode a realistic grammar as is done in HSPG (Pollard and Sag, 1987, forthcoming). In the following examples I will use a type system which is similar to that found in Pollard and Sag (forthcoming), although considerably simplified.[4] The structure shown in Figure 1 is that assigned to the sentence *john likes mary* by resolution using this type system. Note that the arguments of the lexical signs have been co-indexed. This structure can be produced by constraint resolving either of the structures shown in Figure 2, the first of which has only the orthographies of the sentence sign instantiated (parsing) and the second only the logical form (generation).

In Figure 1, **s-sign** is a specialisation of the type **idp1** which implements the HPSG immediate dominance principle schema 1. **idp2** corresponds to schema 2. I have omitted several features from the figure. In particular the type **head-comp-struc** actually has features which take as values a list of all the orthographies and synsems of the daughter structures, and the same is true of the types **ne-dtrs-list**. All structures also have a feature which specifies the language, in this case English, which is necessary when translating using a single type system, to ensure that only types and lexical entries which are valid for that language are used. In all cases the language of all the daughters of a phrase must be identical and equal to that of the phrase itself, but I have left out the feature here for simplicity. I have omitted the orthographies of the phrasal signs except for that of the sentence sign itself. I have not shown the syntactic part of the signs in any detail, but both types of phrasal sign obey the subcategorisation principle which states that the SUBCAT value of the HEAD-DTR is the concatenation (append) of the phrases SUBCAT list with the list of the SYNSEM values of the COMP-DTRS. They also obey the head feature principle, that the HEAD of the phrase is the HEAD of the HEAD-DTR. For further details, see Pollard and Sag (forthcoming).

The simplest treatment of lexical entries is to make them all distinct leaf types and to specify all their orthographies as distinct leaf types. This is cumbersome, so I have assumed that the type **string** covers arbitrary (Lisp) strings as usual in the LKB and that lexical signs are "pseudo-types". The system makes no attempt to resolve a node of type **string**. Pseudo-types can be regarded as (unnameable) types which cannot introduce new features and cannot have multiple parents. Thus lexical signs behave with respect to constraint resolution as though they are subtypes of types such as **tverb-sign**, but the set of lexical signs can be varied without redefining the type system.

Note that in the LKB system there is no separate linear precedence component. The phrasal types are defined so that their orthography is the result of appending the orthographies of their daughters in an order specified by the sign. Note also that lexical rules in this system have to be regarded in the conventional way as expanding the lexicon in a preliminary processing phase. This is discussed further in Section 6. The main reason for investigating the use of a constraint system in the LKB is its naturalness as a way of implementing a testbed MT system. This is the subject of the rest of this document.

4 Multilingual lexical representation

It should be emphasised from the outset that the intention of this work is to concentrate on the linguistic aspects of multilingual representation and the problems of the lexicon in particular, rather

[4]The existing ACQUILEX type system could be converted into a constraint based system in a way which is quite straightforward in principle: instead of the existing three types of grammar rules there would be three types of phrasal sign, which explicitly incorporated the daughters of the rule. In practice, some less straightforward modifications would have to be made, since not all signs allowed by the grammar in the ACQUILEX system can be extended to ground feature structures.

$$
\left[\begin{array}{l}
\textbf{s-sign} \\
\text{ORTH} = \langle \textbf{john, likes, mary} \rangle \\
\text{LANG} = \textbf{english} \\
\text{SYNSEM} = \left[\begin{array}{l}
\textbf{s-synsem} \\
\text{CAT} = \boxed{\textbf{scat}} \\
\text{SEM} = \fbox{8} \left[\begin{array}{l}
\textbf{tvsem} \\
\text{PRED} = \textbf{and} \\
\text{ARG1} = \left[\begin{array}{l}
\textbf{tvformula} \\
\text{PRED} = \textbf{like\_1} \\
\text{ARG1} = \fbox{9}\ \textbf{obj} \\
\text{ARG2} = \fbox{10}\ \textbf{obj}
\end{array}\right] \\
\text{ARG2} = \left[\begin{array}{l}
\textbf{compl-formula} \\
\text{PRED} = \textbf{and} \\
\text{ARG1} = \fbox{11} \left[\begin{array}{l}
\textbf{npsem} \\
\text{PRED} = \textbf{john\_1} \\
\text{ARG1} = \fbox{9}
\end{array}\right] \\
\text{ARG2} = \fbox{12} \left[\begin{array}{l}
\textbf{npsem} \\
\text{PRED} = \textbf{mary\_1} \\
\text{ARG1} = \fbox{10}
\end{array}\right]
\end{array}\right]
\end{array}\right]
\end{array}\right] \\
\text{DTRS} = \left[\begin{array}{l}
\textbf{head-comp-struc} \\
\text{HEAD-DTR} = \left[\begin{array}{l}
\textbf{idp2} \\
\text{SYNSEM} = \boxed{\textbf{vp-synsem}} \\
\text{DTRS} = \left[\begin{array}{l}
\textbf{head-comp-struc} \\
\text{HEAD-DTR} = \left[\begin{array}{l}
\textbf{tverb-sign} \\
\text{ORTH} = \langle \textbf{likes} \rangle \\
\text{SYNSEM} = \left[\begin{array}{l}
\textbf{tv-synsem} \\
\text{CAT} = \boxed{\textbf{tvcat}} \\
\text{SEM} = \fbox{8}
\end{array}\right]
\end{array}\right] \\
\text{C/DTRS} = \left[\begin{array}{l}
\textbf{ne-dtrs-list} \\
\text{HD-DTR} = \left[\begin{array}{l}
\textbf{lex-np-sign} \\
\text{ORTH} = \langle \textbf{mary} \rangle \\
\text{SYNSEM} = \left[\begin{array}{l}
\textbf{np-synsem} \\
\text{CAT} = \boxed{\textbf{npcat}} \\
\text{SEM} = \fbox{12}
\end{array}\right]
\end{array}\right] \\
\text{TL-DTR} = \boxed{\textbf{e-dtrs-list}}
\end{array}\right]
\end{array}\right]
\end{array}\right] \\
\text{C/DTRS} = \left[\begin{array}{l}
\textbf{ne-dtrs-list} \\
\text{HD-DTR} = \left[\begin{array}{l}
\textbf{lex-np-sign} \\
\text{ORTH} = \langle \textbf{john} \rangle \\
\text{SYNSEM} = \left[\begin{array}{l}
\textbf{np-synsem} \\
\text{CAT} = \boxed{\textbf{npcat}} \\
\text{SEM} = \fbox{11}
\end{array}\right]
\end{array}\right] \\
\text{TL-DTR} = \boxed{\textbf{e-dtrs-list}}
\end{array}\right]
\end{array}\right]
\end{array}\right]
$$

Figure 1: Structure for parse of "John likes Mary". (C/DTRS is used as an abbreviation for COMP-DTRS).

$$
\left[\begin{array}{l}
\textbf{s-sign} \\
\text{ORTH} = \langle \textbf{john, likes, mary} \rangle \\
\text{LANG} = \textbf{english}
\end{array}\right]
$$

$$
\left[\begin{array}{l}
\textbf{s-sign} \\
\text{LANG} = \textbf{english} \\
\text{SYNSEM} = \left[\begin{array}{l}
\textbf{s-synsem} \\
\text{SEM} = \left[\begin{array}{l}
\textbf{sem} \\
\text{PRED} = \textbf{and} \\
\text{ARG1} = \left[\begin{array}{l}
\textbf{formula} \\
\text{PRED} = \textbf{like\_1} \\
\text{ARG1} = \fbox{9}\ \textbf{obj} \\
\text{ARG2} = \fbox{10}\ \textbf{obj}
\end{array}\right] \\
\text{ARG2} = \left[\begin{array}{l}
\textbf{formula} \\
\text{PRED} = \textbf{and} \\
\text{ARG1} = \left[\begin{array}{l}
\textbf{sem} \\
\text{PRED} = \textbf{john\_1} \\
\text{ARG1} = \fbox{9}
\end{array}\right] \\
\text{ARG2} = \left[\begin{array}{l}
\textbf{sem} \\
\text{PRED} = \textbf{mary\_1} \\
\text{ARG1} = \fbox{10}
\end{array}\right]
\end{array}\right]
\end{array}\right]
\end{array}\right]
\end{array}\right]
$$

Figure 2: Goal structures which can be resolved to parse and generate from logical form

than MT as such. MT is not the only possible application for a multilingual LKB — ultimately such an object should be able to support the needs of linguists, lexicographers and others who need access to substantial quantities of cross-linguistic information. Therefore here we concentrate on whether the lexicon contains adequate syntactic and semantic information for MT, leaving open issues such as how to support disambiguation, pragmatically appropriate lexical choice and so on which are crucial to real MT but depend heavily on general and domain-specific real world knowledge.[5] For example, it would clearly be problematic for real applications if when translating (1) into German, *Ufer* rather than *Bank* was used, but giving both options would be the correct behaviour for the LKB system.

(1) John went to the bank and paid in the cheque for £36.49.

We take it as a goal that our encoding of multilingual lexical information should support approaches which are declarative and bidirectional and that, as far as possible, we should avoid encoding information which is specific to particular techniques for MT. Thus we have no interest in producing a practical MT system within the ACQUILEX project, but we clearly have to support construction of the multilingual lexicon and show that it contains sufficient information to allow MT, and the best way of achieving this is to build a test-bed MT system.

4.1 MT in unification based formalisms

Like parsing and generation, MT can be regarded very generally as a process of constraint solving. The target language (TL) sentence generated is constrained by the monolingual grammar and lexicon and also by constraints derived from the parse of the source language (SL) sentence. Different approaches to MT can be characterised according to what sort of cross-linguistic constraints are imposed, in addition to the monolingual constraints. Here we will briefly discuss interlingua and then consider some transfer based approaches.

The archetypal use of an interlingua corresponds to the situation where the semantic structure produced by the parse of the SL sentence is identical to that which would be produced by the parse of the TL sentence (see Emele *et al* 1992, for an MT system which uses constraint resolution to produce an interlingual representation). The difficulty with designing interlingual representations which can be used in this way is to ensure that they can be generated by the grammars of the individual languages, without the grammars becoming too unnatural or the job of the grammar writer unreasonably hard. One problem is that structures which may be intended to be interpreted as having identical meanings may not be identical with respect to the grammar in that they will not generate identical strings (see Shieber, 1993, for the problem of logical form equivalence in generation). To take a simple example, the structures

$$
\begin{bmatrix}
\textbf{formula} & & \\
\text{PRED} & = & \textbf{and} \\
\text{ARG1} & = & \begin{bmatrix} \textbf{formula} & & \\ \text{PRED} & = & \textbf{young} \\ \text{ARG1} & = & \boxed{1}\ \textbf{x} \end{bmatrix} \\
\text{ARG2} & = & \begin{bmatrix} \textbf{formula} & & \\ \text{PRED} & = & \textbf{man} \\ \text{ARG1} & = & \textbf{x} \end{bmatrix}
\end{bmatrix}
\quad
\begin{bmatrix}
\textbf{formula} & & \\
\text{PRED} & = & \textbf{and} \\
\text{ARG1} & = & \begin{bmatrix} \textbf{formula} & & \\ \text{PRED} & = & \textbf{man} \\ \text{ARG1} & = & \boxed{1}\ \textbf{x} \end{bmatrix} \\
\text{ARG2} & = & \begin{bmatrix} \textbf{formula} & & \\ \text{PRED} & = & \textbf{young} \\ \text{ARG1} & = & \textbf{x} \end{bmatrix}
\end{bmatrix}
$$

might be interpreted as $man(x) \wedge young(x)$ and $young(x) \wedge man(x)$ and as therefore being equivalent but this does not mean that they will generate identical strings in a grammar or indeed that a grammar will accept both. Emele *et al*'s use of a constraint-based formalism reduces the problem to some extent, because it makes it possible to write grammars where the semantic structure does not have to mirror the constituent structure, but it still leaves open the problem of ensuring that each grammar will generate compatible interlingua. It is difficult to see how this can be done in the absence of a solution to the AI-complete meaning representation problem unless the grammar

[5] Nothing we shall say is incompatible with the assumption that some knowledge needed for such processing is associated with the lexicon. There is clearly an issue of how to integrate the mechanism with the further processing needed and this is briefly addressed in Section 5.

writer has detailed knowledge of at least two languages and writes or tunes the grammars so that the representations fit each other.

Transfer based approaches, in contrast, do not assume identity between any part of the SL structure and TL structure and this can allow the grammars for the different languages to be developed independently. However transfer systems differ with respect to the sort of correspondences which can be described. Just as grammar rules expressed in a unification-based formalism can be regarded non-procedurally as constraints on well-formed monolingual structures, transfer rules or translation links can be regarded as constraints on the mappings between source and target structures which are translation equivalent. These mappings may be stated in terms of orthography, syntax, semantics or a combination of these components, and may be described at a lexical, phrasal or sentential level. To some extent, the choice depends on the monolingual grammar framework used — for example, within sign-based lexicalist approaches such as HPSG, it is natural to think of transfer in terms of signs, and to concentrate on lexical transfer.

Thus we can classify MT systems in terms of such a model according to the sort of translation constraints they assume, and how they control the process of translation. For example in SRI's BCI system (Alshawi *et al.*, 1991) the source language string is parsed and transfer is carried out on the (quasi-)logical form representation. This produces a quasi-logical form appropriate for the TL, which can be used to drive a head-driven generator. In contrast, the Shake-and-Bake approach (Whitelock, 1992) relies on lexical transfer operating on lexical signs which have had their variable instantiated as a result of parsing. As in the BCI system, transfer operates before generation, but in Shake-and-Bake generation is constrained by 'bags' of instantiated lexical entries rather than an ordered representation of the sentence as a whole. This has the advantage that there is no problem of potential mismatch between the monolingual grammars. However it also means that normal generation techniques are not applicable; the algorithm actually used roughly involves producing all possible orderings of transferred target language signs and using standard parsing techniques to determine which of these orderings meet the constraints of the target language grammar (which means that the number of sentences which must be parsed is factorial in the length of the SL sentence). The ELU system (Estival *et al.*, 1990) demonstrates a third approach; transfer operates at both a lexical and a phrasal level, and transfer and generation are interwoven by making use of transfer variables which indicate explicitly that parts of the structure remain untranslated.

Transfer based approaches may put a heavy burden on the writers of the monolingual grammars, if the structures which are stated to be translation equivalent have to have some degree of isomorphism. It is also desirable to avoid the situation where detailed knowledge of the monolingual structures is required in order to write transfer rules. The particular approach to translation which we are exploring attempts to avoid these pitfalls. It is heavily lexicalist in orientation, in that constraints are described between signs in the SL and TL structures, but it is also possible to incorporate non-lexical cross-linguistic constraints.

We define lexical translation equivalence in terms of cross-linguistic links, *tlinks*, stated in terms of lexical entries in the monolingual lexicons. Because tlinks are defined in terms of inheritance from lexical entries and rules, translation equivalence can be stated through direct reference to properties of word syntax and semantics as they appear in the monolingual lexical descriptions. Tlinks can be viewed as information structures describing ways in which lexical entries from the source and target languages are mapped into translation-equivalent pairs. The simplest cases of tlinks are where the lexical entries themselves can be considered translation equivalent but the mechanism allows translation equivalence to be expressed between lexical entries which have been in some way modified or grouped together in a phrase. A fully instantiated tlink can be viewed a stating that a particular set of ground FSs in the SL are translation equivalent to a set of ground FSs in the TL. Thus they can be seen as acting as constraints on valid SL/TL translation equivalent pairings. As in Shake-and-Bake, the translation process is driven by stating that the variables in the semantic structures in the source and target are identical.

The LKB testbed system applies tlinks in a way which is reminiscent of Shake-and-Bake. The source language sentence is parsed, which results in instantiation of the variables in the lexical signs which make up part of the parsed structure. Tlinks are applied to appropriate subparts of the structure. This results in a "bag" of partially instantiated feature structures in the target language.

$$\begin{tikzcd} \text{SFS1} \arrow[r, "\text{source mapping}"] & \text{SFS0} \arrow[r, leftrightarrow, "\textbf{tlink}"] & \text{TFS0} & \text{TFS1} \arrow[l, "\text{target mapping}"'] \\ \text{source LE} & \text{source TE} & \text{target TE} & \text{target LE} \end{tikzcd}$$

Figure 3: Figurative representation of a word-word translation link.

$$\begin{tikzcd} \text{SFS1} \arrow[r, "\text{identity}"] & \text{SFS0} \arrow[r, leftrightarrow, "\textbf{tlink}"] & \text{TFS0} & \text{TFS1} \arrow[l, "\text{identity}"'] \\ \text{fly\_1} & \text{fly\_1} & \text{volare\_1} & \text{volare\_1} \end{tikzcd}$$

Figure 4: Tlink between *fly* (intransitive) and *volare*.

Generation then proceeds in a similar way as before, except that the lexical entries available to act as pseudo-types are restricted to those in the generation bag and (unlike in Shake-and-Bake) phrasal constraints may also be expressed. Furthermore, we allow correspondences to be stated in terms of feature structures which can be resolved to one or more lexical entries, rather than insisting that the actual lexical entry is known in all cases. This allows us to exploit a notion of a limited interlingua, as will become apparent below. In the rest of this section, we give a detailed description of tlinks and of the process of translation in the LKB.

4.2 Tlinks

Tlinks can be viewed as information structures describing ways in which input lexical entries (LE) from the source and target languages are mapped into output translation-equivalent (TE) pairs. Most tlinks are of the word to word type, which can be represented diagrammatically as in Figure 3. In these cases one lexical item in the source language is related to one lexical item in the target language, although the FS which are actually stated to be translation equivalent may be specialisations of the lexical entry, or be derived from the entry by rule application.[6] For example, the English lexical entry for (intransitive) *fly* is translation equivalent to the Italian *volare* (see Figure 4). In this case the source and target mappings are identity, since the base lexical entries themselves can be stated to be translation equivalent.

Somewhat rarer and more complex cases of linking arise when a translation mismatch involves transformations operating on FSs. For example, the equivalence class resulting from translation pairs such as 'English:*furniture* ≈ Spanish:*muebles*' can be represented by establishing a link between a word and its translation in the plural form. In this case the equivalence holds between a basic lexical entry and a lexical entry after rule application. Here the mapping on the English side is identity again, but the Spanish mapping is the morphological rule for plural formation, as shown in Figure 5.

In more complicated cases, more than one lexical item may be involved on one or both sides of the tlink. A word-to-phrase tlink is shown diagrammatically in Figure 6 (phrase-to-phrase or phrase-to-word tlinks are also possible). Here the constraint on translation equivalence is stated partially at the phrasal level and partly in terms of the lexical structures that the phrase contains.

[6] Tlinks are both symmetrical and reversible; we use the terminology source and target for ease of exposition.

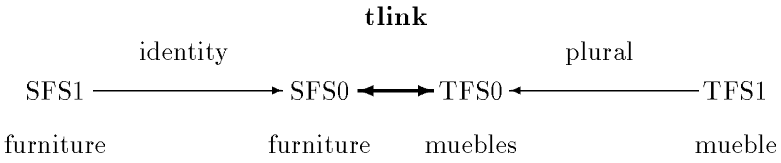

Figure 5: Tlink relating *furniture* and *muebles*

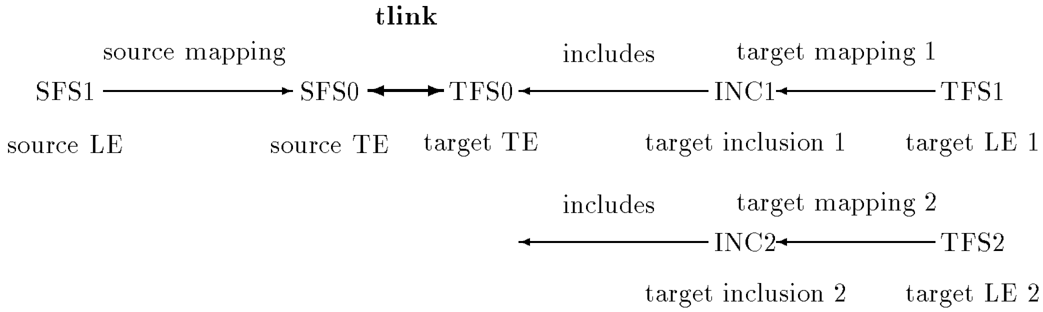

Figure 6: Figurative representation of a word-phrase translation link.

For example, the Spanish-English VOX dictionary gives *novillo* the translation *young bull*. Diagrammatically we can represent this as Figure 7. Note that this is less restrictive than stating the constraint as a simple FS, where the components were specified precisely. This tlink allows *novillo ingles* to be translated as *young English bull*, for example, where the inserted adjective is allowed because the specification is simply that the phrase includes *young* and *bull*, not that it consists of exactly those two lexical items.

More formally, tlinks are defined as FSs of type **tlink** which relates the source FS (SFS) and the target FS (TFS) which are both of type **rule**:

```
tlink (top)
< SFS > = rule
< TFS > = rule.
```

In the simplest case, a rule establishes a correspondence between an input sign (1) and an output sign (0) in the same language:

```
rule (top)
< 0 > = sign
< 1 > = sign
< 0 : LANG > = < 1 : LANG > .

unary-rule (rule) .
```

The rule-inputs of tlinks are meant to be instantiated by FS representations of word senses in the source and target languages; rule-outputs provide the TE pairs. Thus, when the FSs at the end of the paths `< SFS : 1 >` and `< TFS : 1 >` are instantiated by lexical entries, the FSs at the end of the paths `< SFS : 0 >` and `< TFS : 0 >` are defined to be translation equivalent.

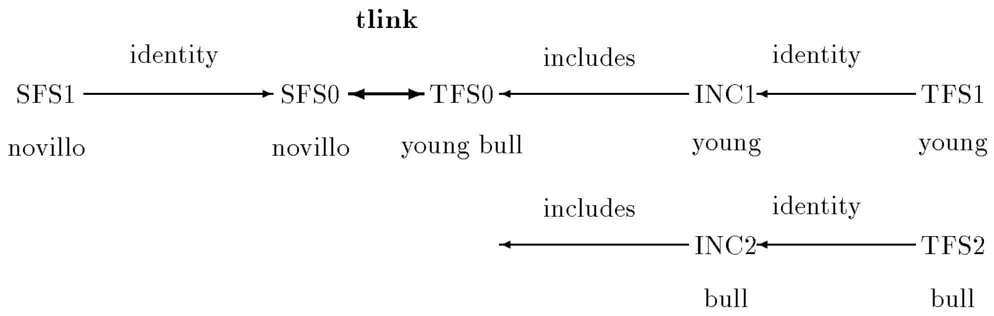

Figure 7: Tlink relating *novillo* and *young bull*

Tlinks can be regarded statically as defining constraints on translation equivalent pairings or dynamically as generating new FSs; given a FS in one language, and an appropriate tlink, unification with the FS at the end of the appropriate path, the *SL output*, (e.g. `< SFS : 0 >`) in the tlink, will result in the FS at the end of the other output path (the *TL output*) being returned (e.g. `< TFS : 0 >`).

The concept of translation equivalence is constrained by defining an inheritance network of tlink types encoding generalisations relative to classes of crosslinguistic equivalences. The commonest and simplest cases of noun translation equivalence can be represented as **simple-tlinks**.

```
simple-tlink (tlink)
< SFS : 0 > = < SFS : 1 >
< TFS : 0 > = < TFS : 1 >
< SFS : 0 : SYNSEM : SEM : ARG1 > =
   < TFS : 0 : SYNSEM : SEM : ARG1 >.
```

A **simple-tlink** is applicable in the case where two nouns are straightforwardly translation equivalent, without any transformation being necessary. The semantic variables relative to the semantic entity described by two output structures are specified to be identical. This variable equivalence is the basis for the use of tlinks within unification based approaches to MT and is crucial in expressing translation mismatches such as thematic divergence and head switching (Sanfilippo *et al.*, 1992). The particular paths equated will depend on the way that the type system is used to encode semantic structure — in the examples which follow a simplified encoding has been used for ease of exposition.

For example, assuming that the English word sense *chocolate* 4, is translation equivalent to the Dutch *chocolade* 2, we would have the tlink:

```
simple-tlink
< SFS : 1 > <= chocolate_4 <>
< TFS : 1 > <= chocolade_2 <>.
```

where `<=`, indicates non-default inheritance from a named FS (the particular lexical entries). This tlink will yield the English entry as output when the Dutch entry is supplied as input and vice versa. In order to make the description more readable we use a notation of the following form:

```
psort / psort
description .
```

where the psort on the LHS of the slash is taken as the value for the path `< SFS : 1 >` and the psort on the RHS of the slash is taken as the value for the path `< TFS : 1 >` and `description` is a normal FS description. Thus the tlink above is equivalent to:

```
chocolate_4 / chocolade_2
simple-tlink.
```

For intransitive verbs we make use of a slightly different tlink type:

```
iverb-tlink (tlink)
< SFS : 0 > = < SFS : 1 >
< TFS : 0 > = < TFS : 1 >
< SFS : 1 > = iverb-sign
< TFS : 1 > = iverb-sign
< SFS : 0 : SEM : ARG1 : ARG1 > =
  < TFS : 0 : SEM : ARG1 : ARG1 >.
```

For example the tlink for 'English:*fly* ≈ Italian:*volare*' would be:

```
fly_1 / volare_1
iverb-tlink.
```

Some restrictions on translation can be expressed by making the target or source FSs of tlinks more specific. For example, we can define a type **human-tlink** and state as a constraint that the values for the SEX feature must be the same in the translation equivalent feature structures:

```
human-tlink (simple-tlink)
< SFS : 0 : QUALIA : SEX > =
   < TFS : 0 : QUALIA : SEX >.
```

A tlink of this type would be suitable for establishing equivalences such as that between the word *teacher* in English and its translation in Spanish as either *maestro* or *maestra*. The restriction that *maestro* denotes a male teacher and *maestra* a female one — i.e. the values for `< QUALIA : SEX >` in the FS descriptions for the two words are **male** and **female** respectively — follows from the use of tlinks of type **human-tlink** to relate *teacher* with *maestro* and *maestra*:

```
teacher / maestro :
human-tlink.

teacher / maestra :
human-tlink.
```

The path equation induced by the **human-tlink** will ensure that translation equivalent words denote individuals of the same (natural) gender.[7] The use of a common type system makes it easier to express such relationships, although it is not essential, provided that comparable information is encoded in both monolingual representations.

In the case of examples such as 'English:*furniture* ≈ Spanish:*muebles*' we instantiate one half of the tlink with the appropriate lexical rule. This is shown in the tlink below where the target-language side (`< TFS >`) inherits from the FS representation for the plural rule whose input is the tlink's target input (`mueble`) and output is the tlink's target output.

```
furniture / mueble :
tlink
< SFS : 0 > = < SFS : 1 >
< TFS > <= plural <>.
```

Note that the lexical/morphological rule for plural formation used in the tlink is needed anyway for use in parsing/generation, so that its use in tlinks does not involve introduction of elements other than those needed in the monolingual grammars. Since the singular form *mueble* would not unify with the feature structure at the end of the output path `< TFS : 0 >`, a translation of *mueble*

[7] As we will see later, the morphological generalisation involved in this example can be captured, making it unnecessary to manually specify both tlinks.

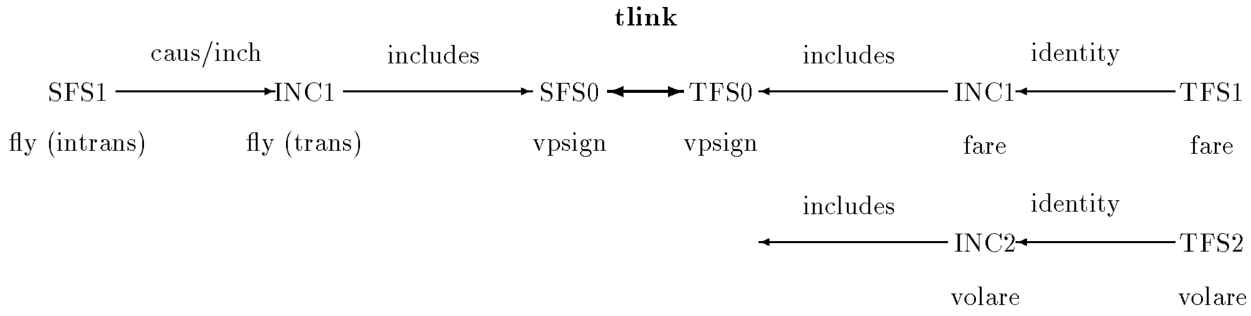

Figure 8: Tlink relating transitive *fly* to *fare volare*.

as *furniture* would not be generated and another tlink would be required to relate the two words when translating a phrase such as *a piece of furniture*.

As indicated by Figure 6 above, phrasal tlinks in general have a more complex structure. The translation equivalent structure is not a simple FS, but consists of a single FS corresponding to the phrase as a whole, plus one or more FSs describing lexical items which are components of the phrase. In order to allow the component lexical items to be non-basic forms, we allow for a mapping between the input signs and the included signs. To describe such tlinks we use a more complex type of **rule** on the phrasal side(s) of the tlink. The type constraint appropriate for the case where two lexical signs are described as being included in a phrase is shown below:

$$
\begin{bmatrix}
\textbf{binary-inclusion} \\
0 = \begin{bmatrix} \textbf{sign} \\ \text{LANG} = \boxed{1} \end{bmatrix} \\
1 = \boxed{0} \begin{bmatrix} \textbf{sign} \\ \text{LANG} = \boxed{1}\ \textbf{lang} \end{bmatrix} \\
2 = \boxed{2}\ \textbf{sign} \\
\text{INC1} = \begin{bmatrix} \textbf{unary-rule} \\ 1 = \boxed{0} \\ 0 = \begin{bmatrix} \textbf{sign} \\ \text{LANG} = \boxed{1} \end{bmatrix} \end{bmatrix} \\
\text{INC2} = \begin{bmatrix} \textbf{unary-rule} \\ 1 = \boxed{2} \\ 0 = \begin{bmatrix} \textbf{sign} \\ \text{LANG} = \boxed{1} \end{bmatrix} \end{bmatrix}
\end{bmatrix}
$$

The description of the phrase as a whole is given by `< 0 >`, the input lexical signs are the values of `< 1 >` and `< 2 >` (which are the same as the values of `< INC1 : 1 >` and `< INC2 : 1 >`) and the signs included in the phrase are found as the values of `< INC1 : 0 >` and `< INC2 : 0 >`. As before, the input lexical signs may be transformed by lexical rule application, but in the simple cases the mapping is identity and in such cases `< INC1 : 1 >` will be identical to `< INC1 : 0 >` and `< INC2 : 1 >` will be identical to `< INC2 : 0 >`. For inclusion of a single item 2 and INC2 will not be present, for three items we add 3 and INC3 and so on. There will normally be further constraints on how the included items interrelate.

For example, consider the case of the transitive use of motion verbs such as *fly* in English, which has to be translated in Italian with an explicit causative using *fare* (see Copestake and Sanfilippo, 1993):

(2) Jamie *flew* his kite
Jamie *fece volare* l'acquilone

Diagrammatically, a tlink which could be used to express this is shown in Figure 8 (there are other possibilities). The path description is partially shown below:

```
< SFS : 1 > <= fly_1 <>
< TFS : 1 > <= fare_1 <>
< TFS : 2 > <= volare_1 <>
< SFS : 0 > = idp2
< SFS : INC1 > <= causative-inchoative-alternation <>
< SFS : 0 : SYNSEM : SEM > =
   < SFS : INC1 : 0 : SYNSEM : SEM >
< TFS : 0 > = idp2
< TFS : INC1 > = identity
< TFS : INC2 > = identity
< TFS : 0 : SYNSEM : SEM > =
   < TFS : INC1 : 0 : SYNSEM : SEM > .
```

Again, the slashed notation is simpler:

```
fly_1 / fare_1 volare_1 :
< SFS : 0 > = idp2
< SFS : INC1 > <= causative-inchoative-alternation <>
< SFS : 0 : SYNSEM : SEM > =
   < SFS : INC1 : 0 : SYNSEM : SEM >
< TFS : 0 > = idp2
< TFS : INC1 > = identity
< TFS : INC2 > = identity
< TFS : 0 : SYNSEM : SEM > =
   < TFS : INC1 : 0 : SYNSEM : SEM > .
```

In this example, the English phrase is stated to be an **idp2**, with a semantic structure equal to that of the included lexical sign, which is the transitive version of *fly*. The Italian phrase is also a **idp2** including *fare* and *volare*, suitably constrained. Here I have not shown the path equality statements for the semantics, since this depends on the encoding and a full discussion of the possibilities is out of place here. One simple possibility would be for the English semantics to be equivalent to

$$[CAUSE(e, x, e') \wedge fly(e', y)]$$

(where the intransitive is $fly(e', y)$) and the Italian to be

$$[fare(e, x, e') \wedge volare(e', y)]$$

but more complex encodings are possible. The two semantic structures need not be isomorphic as they are here, but it has to be possible to relate the variables. In particular, the event variables have to be directly related to deal correctly with adverbs and with cases where the structure is embedded.

In fact this translation pattern generalises over a number of lexical items, such as *march*, *run* and so on, so it makes sense to define a type of tlink which captures the generalisations, which would leave us with a much simpler bilingual entry.[8] In Copestake and Sanfilippo (1993) we discussed the use of a "partial interlingua" using thematic roles which allows us to state the general translation pattern in a more linguistically motivated fashion.

In general, the set of ground FSs described by a phrasal tlink is infinite. In this example, the phrases *fly the kite*, *fly the big red kite*, *fly the aeroplane* and so on, will all match the description of the English phrase. However, these structures will not be generated unless licenced by other parts of the input. Thus *kite* will not be valid unless *acquilone* appears in the Italian sentence and there is a tlink that links the two. The conditions on translation are discussed in detail in section 4.5.

[8] As we will see below, this tlink could actually be derived automatically from the simpler tlink between intransitive *fly* and *volare*.

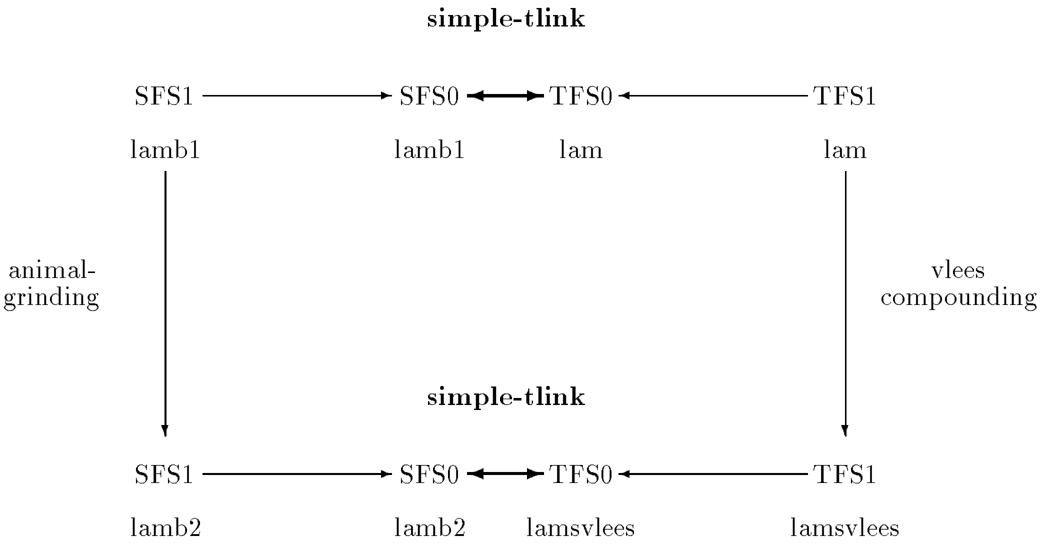

Figure 9: Tlink rule relating animal and meat senses of *lamb* to *lam* and *lamsvlees*

4.3 Tlink rules

In some cases the existence of a tlink between two lexical items implies a further translation relationship. For example, in English, there is a regular sense extension such that a word used primarily to denote an animal can also be used to denote the meat of that animal (e.g. *lamb*, *chicken*, see Copestake and Briscoe (1992)). A similar sense extension rule applies to Italian (Östling 1991) but in Dutch a compound is generally used (*lam*, *lamsvlees*). The correspondence between the two processes (e.g. sense extension in English and compounding in Dutch) and their import on translation equivalence can be expressed by means of a *tlink-rule* which allows transformations to be defined for tlinks. Tlink rules are analogous to lexical rules, that is they can be regarded as generating new tlinks by operating on existing ones. They are treated as applying in a phase which is preliminary to translation proper in order to expand the set of defined tlinks.

For example, we can automatically generate the relationship between the FS representing the animal sense of *lamb* (`lamb_1`) and `lamsvlees` by transforming the simple tlink between the FS representing the meat sense of *lamb* (`lamb_2`) and `lam` into a new tlink where the source input FS results from applying the sense extension rule 'animal-grinding' to `lamb_1` and the source input FS results from compounding *lam* with *vlees*, as shown diagrammatically in Figure 9.

Again tlink-rules are themselves FSs. The type tlink-rule is defined as follows:

```
tlink-rule (top)
< t0 > = tlink
< t1 > = tlink
< srule > = rule
< srule : 1 > = < t0 : sfs : 1 >
< srule : 0 > = < t1 : sfs : 1 >
< trule > = rule
< trule : 1 > = < t0 : tfs : 1 >
< trule : 0 > = < t1 : tfs : 1 > .
```

In the case of relationships involving unary rules and simple-tlinks the type **simple-tlink-rule** is appropriate:

```
simple-tlink-rule (top)
< t0 > = simple-tlink
< t1 > = simple-tlink
< srule > = unary-rule
< trule > = unary-rule
```

The tlink rule involved for the lamb example is then simply defined as:

```
simple-tlink-rule
< srule > <= animal-grinding <>
< trule > <= vleescompounding <> .
```

and again we use an abbreviated notation:

```
psort / psort:
description .
```

where the LHS psort is the value of `< srule >` and the RHS is the value of `< trule >`. The example above then becomes:

```
animal-grinding / vleescompounding:
simple-tlink-rule.
```

Since we are just making use of the monolingual sense extension mechanism here we can rely on that to handle cases where the sense extension is blocked (e.g. *pig*). It does not necessarily matter for translation purposes whether the lexical rule can fully predict the effects of the sense extension; even if it is used to encode the regular aspects of the relationship between two existing lexicalised items, an appropriate translation link will be generated if the monolingual processes are sufficiently similar.

There are many examples of such correspondences; for example the English sense extension between trees and their fruits (*pear* etc) is mirrored in Italian with a gender distinction; the trees are masculine but the fruits feminine (*pero*, *pera*). In fact, our earlier examples of *muebles* $\approx$ *furniture* and *maestro/a* $\approx$ *teacher* can also be described more generally. A tlink-rule can be defined for human-denoting nouns so that given the equivalence between two entries of one gender the tlink the other gender can be produced. Since gender in English is normally unmarked, one instantiation of this is with a 'null lexical rule', which states equality between two FSs. Thus given the tlink between *teacher* and *maestra* we can automatically generate that between *teacher* and *maestro* (see Figure 10).

The relationships at the 'sides' of the tlink-rule need not be lexical rules as they are in this case, indeed in many cases the motivation for a tlink rule is that a process which can be regarded as a lexical rule in one language has to be modelled by phrase formation in another language. Thus in the example of causative formation, the lexical rule in English which relates the causative and inchoative uses of movement verbs such as *fly* can be related to the process of forming a phrase with the explicit causative verb *fare* in Italian.

4.4 A full example

In this section I will discuss a relatively trivial example in some detail in order to give a better idea of the use of tlinks and to introduce the translation system. Consider the translation of the structure given for *John likes Mary* in Figure 1 into Spanish *Maria gusta a Juan* the structure for which is given in Figure 11.

I will assume the following, relatively straightforward tlinks.

```
john_1 / juan_1 :
simple-tlink .

mary_1 / maria_1 :
```

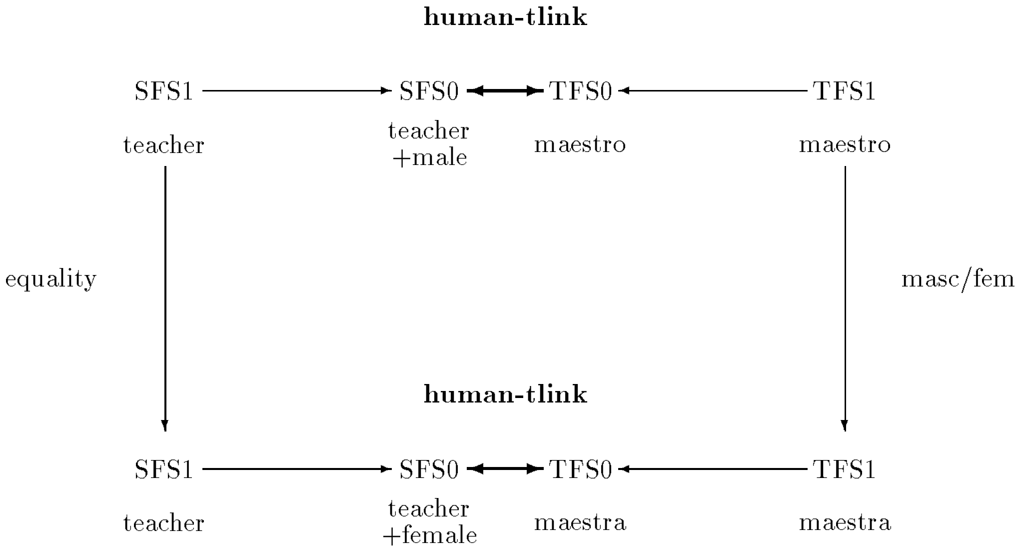

Figure 10: Tlink rule for *teacher* ≈ *maestro/a*

```
simple-tlink .

likes_1 / gusta_1 :
arg-switch-tlink .
```

The arg-switch-tlink type allows the subject of *like* to be the object of *gustar* and vice versa. Thus the normal **tverb-tlink** type is:

```
tverb-tlink (tlink)
< sfs : 0 > = < tfs : 0 >
< sfs : 1 > = < tfs : 1 >
< sfs : 0 : synsem : sem : arg1 : arg1 > =
   < tfs : 0 : synsem : sem : arg1 : arg1 >
< sfs : 0 : synsem : sem : arg1 : arg2 > =
   < tfs : 0 : synsem : sem : arg1 : arg2 > .
```

and **arg-switch-tlink** is:

```
arg-switch-tlink (tlink)
< sfs : 0 > = < tfs : 0 >
< sfs : 1 > = < tfs : 1 >
< sfs : 0 : synsem : sem : arg1 : arg1 > =
   < tfs : 0 : synsem : sem : arg1 : arg2 >
< sfs : 0 : synsem : sem : arg1 : arg2 > =
   < tfs : 0 : synsem : sem : arg1 : arg1 > .
```

The difficulty with this translation is accounting for the Spanish particle *a* which precedes the (animate) direct object. I will use this example to illustrate some of the tlink mechanisms.[9] I will assume that NPs in Spanish have the boolean valued feature OBJECT in their CAT which is

[9] The analysis here is for illustrative purposes only and should not be taken as a serious proposal for the treatment of Spanish.

$$
\left[\begin{array}{l}
\textbf{s-sign} \\
\text{ORTH} = \langle \textbf{maria, gusta, a, juan} \rangle \\
\text{LANG} = \textbf{spanish} \\
\text{SYNSEM} = \left[\begin{array}{l}
\textbf{s-synsem} \\
\text{CAT} = \boxed{\textbf{scat}} \\
\text{SEM} = \boxed{12} \left[\begin{array}{l}
\textbf{tvsem} \\
\text{PRED} = \textbf{and} \\
\text{ARG1} = \left[\begin{array}{l}
\textbf{tvformula} \\
\text{PRED} = \textbf{gustar\_1} \\
\text{ARG1} = \boxed{13}\ \textbf{obj1} \\
\text{ARG2} = \boxed{14}\ \textbf{obj2}
\end{array}\right] \\
\text{ARG2} = \left[\begin{array}{l}
\textbf{compl-formula} \\
\text{PRED} = \textbf{and} \\
\text{ARG1} = \boxed{15} \left[\begin{array}{l}
\textbf{npsem} \\
\text{PRED} = \textbf{maria\_1} \\
\text{ARG1} = \boxed{13}
\end{array}\right] \\
\text{ARG2} = \boxed{16} \left[\begin{array}{l}
\textbf{npsem} \\
\text{PRED} = \textbf{juan\_1} \\
\text{ARG1} = \boxed{14}
\end{array}\right]
\end{array}\right]
\end{array}\right]
\end{array}\right] \\
\text{D:HEAD} = \left[\begin{array}{l}
\textbf{idp2} \\
\text{SYNSEM} = \boxed{\textbf{vp-synsem}} \\
\text{D:HEAD} = \left[\begin{array}{l}
\textbf{tverb-sign} \\
\text{ORTH} = \textbf{gusta} \\
\text{SYNSEM} = \left[\begin{array}{l}
\textbf{tv-synsem} \\
\text{CAT} = \boxed{\textbf{tvcat}} \\
\text{SEM} = \boxed{12}
\end{array}\right]
\end{array}\right] \\
\text{D:COMP} = \left\langle \left[\begin{array}{l}
\textbf{non-lex-np-sign} \\
\text{SYNSEM} = \left[\begin{array}{l}
\textbf{np-synsem} \\
\text{CAT} = \boxed{\textbf{npcat}} \\
\text{SEM} = \boxed{16}
\end{array}\right] \\
\text{D:HEAD} = \left[\begin{array}{l}
\textbf{idp2} \\
\text{SYNSEM} = \boxed{\textbf{objn-synsem}} \\
\text{D:HEAD} = \left[\begin{array}{l}
\textbf{objn-sign} \\
\text{ORTH} = \langle \textbf{juan} \rangle \\
\text{SYNSEM} = \left[\begin{array}{l}
\textbf{objn-synsem} \\
\text{CAT} = \boxed{\textbf{objncat}} \\
\text{SEM} = \boxed{16}
\end{array}\right]
\end{array}\right] \\
\text{D:COMP} = \langle\rangle
\end{array}\right] \\
\text{D:COMP} = \left\langle \left[\begin{array}{l}
\textbf{a-sign} \\
\text{ORTH} = \langle \textbf{a} \rangle \\
\text{SYNSEM} = \left[\begin{array}{l}
\textbf{a-synsem} \\
\text{CAT} = \boxed{\textbf{acat}} \\
\text{SEM} = \textbf{dummy}
\end{array}\right]
\end{array}\right] \right\rangle
\end{array}\right] \right\rangle
\end{array}\right] \\
\text{D:COMP} = \left\langle \left[\begin{array}{l}
\textbf{lex-np-sign} \\
\text{ORTH} = \langle \textbf{maria} \rangle \\
\text{SYNSEM} = \left[\begin{array}{l}
\textbf{np-synsem} \\
\text{CAT} = \boxed{\textbf{npcat}} \\
\text{SEM} = \boxed{15}
\end{array}\right]
\end{array}\right] \right\rangle
\end{array}\right]
$$

Figure 11: The translated structure. I have abbreviated the DTRS : HEAD-DTR and DTRS : COMP-DTRS paths to D:HEAD and D:COMP respectively and shown COMP-DTRS as a list. All other omissions are as before.

set to **true** if they can be a direct object, and false otherwise, and that the entries for *juan* and *maria* have OBJECT set to **false**. A lexical rule, spanish-objn, can apply to animate **lex-np-sign**s producing a sign which can combine with *a* (using **idp2**) to give a **non-lex-np-sign** which has OBJECT set to **true**. Given this, and the psorts above, the following treatments are possible for the bilingual component:

1. Treat *a* as an item which makes no semantic contribution and can therefore be introduced at will. In Figure 11 it can be seen that the SEM value of *a* is **dummy**. The LKB can be parametrised to accept this (and/or other possibilities) as a specification of a semantically vacuous lexical entry. Thus when translating from English into Spanish, the generation bag will contain *a* as well as the FSs resulting from tlink application. The sentence *Maria gusta Juan* will not be generated, because of the monolingual condition that animate direct objects must be marked. The tlink specified above for *john* will only translate it into the non direct object form of *juan*. In order to get the form with the lexical rule applied into the bag, another tlink has to be specified:

```
john_1 / juan_1 :
< sfs : 0 > = < sfs : 1 >
< tfs > <= spanish-objn <> .
```

   Rather than doing this explicitly for each proper noun, we set up the following tlink-rule:

```
equality / spanish-objn :
simple-tlink-rule.
```

   This is applied to expand the tlinks. Note that only the name of the lexical rule need be known by whoever constructs the tlinks, and that there is no reference to the feature OBJECT in the bilingual component. [10]

2. Link the phrase *a juan* to *john*. The tlink involved is:

```
john_1 / a_1 juan_1 :
<> = tlink
< sfs > = unary-rule
< sfs : 0 > = < sfs : 1 >
< tfs > = binary-inclusion
< tfs : 0 > = non-lex-np-sign
< tfs : inc1 : 0 > = < tfs : incl1 : 1 >
< tfs : inc2 > <= spanish-objn <> .
```

   This is an example of a word to phrase tlink. The lexical entry for *john* is unchanged. The lexical entry for *a* and the result of applying the lexical rule spanish-objn to *juan* are included in the phrase, which is simply stated to be a **non-lex-np-sign**. Again this could be generated automatically, given a suitable tlink-rule, from the simple tlink between *john* and *juan*.

3. Link the phrase *john* to a phrase containing *juan* plus a psort. The tlink involved is:

```
john_1 / apsort juan_1 :
<> = tlink
< sfs > = unary-rule
< sfs : 0 > = < sfs : 1 >
< tfs > = binary-inclusion
< tfs : 0 > = non-lex-np-sign
< tfs : inc1 : 0 > = < tfs : incl1 : 1 >
< tfs : inc2 > <= spanish-objn <> .
```

[10] The system should perhaps be modified so that the tlink-rule does not have to be explicitly specified, in that a semantically vacuous transformation such as spanish-objn should always be allowed. The difficulty with this however is that it is not entirely clear how to define semantic vacuity.

This is very similar to the previous case except that apsort is an under-defined place holder for a lexical sign not the sign a_1 itself. It is necessary to specify that there is some lexical sign licensed by the tlink, even if nothing about it is specified. This is because one of the conditions on translation is that all lexical signs which do not have vacuous semantics have to be licensed by a translation link (see section 4.5).

The first of these possibilities is probably the most natural one to use in this case, but in examples where the lexical item cannot be treated as semantically vacuous the choice between specifying it explicitly and using an underspecified representation depends essentially on whether the grammar naturally allows a way of specifying the lexical item or items involved, without using language specific terms. If the phrase can be sufficiently constrained in this way, then this option is to be preferred because it avoids duplicating information in the monolingual grammar. For example, in translating *mueble* into English possible phrases include *piece of furniture*, *item of furniture*, *article of furniture* and (possibly) *stick of furniture*. It is clear that such individuating words depend on the mass noun involved, for example *blade of grass* is acceptable but **item of grass* is not, nor is **blade of furniture*. If it is possible to determine from the monolingual information which the valid individuating phrases for the mass noun are, then it is clearly inappropriate to encode this in the bilingual lexicon and all that should be necessary is to specify that an individuating phrase is present.[11]

In all these cases, the constraints given by the tlinks and the monolingual structure shown in Figure 1 are such that only the single structure shown in Figure 11 can be a solution. In the next section, I give a detailed description of the conditions on translation equivalence which determine the valid pairings. In the following section, the algorithm actually used for translation is described.

4.5 Conditions on translation

A source structure (SS)/target structure (TS) pair is valid under a given set of tlinks iff

1. Both the SS and the TS are type resolved.
2. For all lexical signs in the SS:
   (a) there is a tlink such that the tlink source unifies with the lexical sign and the tlink target matches some part of the TS, or
   (b) there is a tlink such that the tlink source unifies with a phrase which includes the lexical sign and a specification of an included sign from that tlink source unifies with the lexical sign and the tlink target matches some part of the TS, or
   (c) the lexical sign is specified to make no contribution to the semantics of the SS.
3. For all lexical signs in the TS:
   (a) there is a tlink such that the tlink target unifies with the lexical sign and the tlink source matches some part of the SS, or
   (b) there is a tlink such that the tlink target unifies with a phrase which includes the lexical sign and a specification of an included sign from that tlink target unifies with the lexical sign and the tlink source matches some part of the SS, or
   (c) the lexical sign is specified to make no contribution to the semantics of the TS.

Normally we are interested in translating sentences, but SS/TS pairs of other types of sign may also be valid under this definition. Clearly these conditions are symmetrical with respect to the SS and TS. The identification of the tlink source and target sides depends on the language pair —

[11] Clearly it is necessary in practice to avoid generating a ridiculous number of alternative translations by giving all these possibilities for a particular sentence containing *mueble*. But this should not be done by restricting the bilingual lexicon to one possibility as this would mean that the system were not reversible, since the other possibilities would not be translated. Of course this does not preclude weighting alternatives for acceptability in generation etc.

if the tlinks have been defined as L1/L2 tlinks then the SFS side of the tlink is the source and the TFS side the target when translating from L1 to L2 and the TFS side of the tlink is the source and the SFS side the target when translating from L2 to L1.

A tlink source or target is said to match part of an SS or TS if:

1. the tlink source or target is lexical (that is, its type is a subtype of **lexical sign**) and unifies with some part of the SS or TS, or
2. the tlink source or target is phrasal (that is, its type is a subtype of **phrasal sign**), it unifies with some phrase which is a part of the SS or TS and all included structures in the tlink unify with some structure which is a subpart of that phrase.

In some circumstances a further condition is necessary on translation, that the target structure produced be the minimal one that satisfies these conditions. For example, this condition might be necessary to avoid translations which iterated adjectives, though this depends on the semantics.

The inclusion condition requires an extension to the LRL. It can be formalised most simply as follows:

Definition 2 (Inclusion) *A feature structure* $F' = \langle Q', q'_0, \delta', \alpha' \rangle$ *includes another feature structure* $F = \langle Q, q_0, \delta, \alpha \rangle$ *if* $q_0 \in Q'$.

The definition is the same as that used in LFG. However it is important to note that we do not have cope with general inclusion as defined above in order to produce the effects we are seeking with tlinks. Firstly, we only define inclusion relationships between phrasal and lexical signs. Secondly, inclusion relationships need only be defined to hold between type-resolved feature structures, since both the SS and the TS will be type-resolved. Rather than implement inclusion in its full generality, the LKB uses task specific code to check for inclusion as described in the next section.

There is one more condition which is necessary for the Shake-and-Bake style approach, which is that variables which are (potentially) distinct in the source are also distinct in the target. This condition is necessary to avoid the possibility of conflating two variables and producing incorrect translations. For example, assume that we are translating the English sentence "John sees Mary and John sleeps" into English with a series of tlinks which specify identity. If we do not distinguish the variables pertaining to John and Mary, we will get not only the original sentence, but also "Mary sees Mary and John sleeps" etc. This is obviously a silly example, but the same problem can occur when doing more realistic translations. Again, formalising this requires an extension to the LRL, such as inequalities (Carpenter, 1992), but the implementation actually involves special purpose code.

4.6 The translation algorithm

The definition given above simply specifies what valid SS/TS pairings are, it does not describe how we actually translate a sentence in L1 into L2. It would in theory be possible to treat translation purely as a process of constraint resolution. For example, suppose we had only lexical tlinks and treated individual tlinks as pseudo-types like lexical entries and had no items with dummy semantics, then if the LRL was extended to allow set valued-features we could describe the translation relationship as follows:

```
[ translation
  SS : [ s-sign
         SET-OF-SIGNS = sign-set [ 1 ]]
  TS : [ s-sign
         SET-OF-SIGNS = sign-set [ 2 ]]
  TLINK-LIST : [ tlink-list
                 INPUT-SET = [ 1 ]
                 OUTPUT-SET = [ 2 ] ]]

tlink-list (top).
```

```
e-tlink-list (tlink-list)
< INPUT-SET > = {}
< OUTPUT-SET > = {}.

ne-tlink-list (tlink-list)
< HEAD > = tlink
< TAIL > = tlink-list
< INPUT-SET > = { < HEAD : SFS : 0 > } union < TAIL : INPUT-SET >
< OUTPUT-SET > = { < HEAD : TFS : 0 > } union < TAIL : OUTPUT-SET > .
```

Here the feature SIGN-SET on a phrase is assumed to be equal to the union of the SIGN-SET of each of its daughters, and the SIGN-SET value of a lexical entry is the singleton set containing that lexical entry.

Resolving the feature structure of type **translation** would give us the valid SS/TS pairings. However this would be totally impractical as an implementation technique. It is quite unnecessarily general — it would allow us to perform such operations as deciding whether a partial sentence in L1 was a possible translation of a partial sentence in L2, or translating given pieces of logical form, rather than orthographic strings for example, but all we are actually interested in doing is generating a sentence in L2 given an input sentence in L1 and vice versa. Thus we want the system to be reversible, but we do not need it to be able to resolve an arbitrary combination of information from the SS and TS.

The algorithm currently used in the LKB for translation is the following:

1. The source language sentence is parsed using general constraint resolution, that is the constraint program is given a goal of a feature structure of type **s-sign** (or other sign type) with the language specified as the source language and the orthography instantiated according to the input sentence.

2. All potentially distinct variables in the semantics of this structure are set to have distinct values. For example, if there are two nodes with type **obj** in the structure and **obj** is declared to be a type which applies to variables, then these values are further instantiated with the specially generated types **obj1** and **obj2** (these can be regarded as similar to the subtypes of **string**).

3. The available tlinks are applied to the source language structure to yield a *generation bag* of potential target language structures. This is done by traversing the target structure, checking for applicable tlinks at each node which is of a type applicable to phrasal or lexical signs.

    (a) For phrasal sign nodes, only phrasal input tlinks may apply. For all that do, the output of the tlink is added to the generation bag. Checking application of a phrasal tlink involves checking that the SS phrase to which the tlink is being applied contains all the appropriate lexical entries. As a side effect of application, these included nodes are marked.

    (b) For lexical sign nodes, all lexical sign input tlinks are tried and the output added to the generation bag. If no tlinks apply, the sign is checked to see whether it has "dummy semantics", that is whether it is specified that it will not make any contribution to the semantics of the sentence. If not, the node is checked for marks resulting from the previous application of phrasal constraints. If these exist it is noted that one of these phrasal constraints must apply in any TS. If none exist, translation fails, since there is no possibility of any output corresponding to the sign appearing in the TS. If any (phrasal or lexical) tlink does apply it is noted that this tlink translated this particular part of the SS.

4. The constraint resolution program is given a goal which is simply the feature structure of type **s-sign** (or other sign type). Normally resolving this would generate all the possible

sentences of the target language, but in this case the resolution is constrained so that instead of a free choice of lexical items only items in the generation bag may be used. Since these are mutually constraining, because of their instantiated semantics, the structures generated are further restricted. The two modifications to the general constraint resolution code are:

(a) Any phrasal constraints in the bag are treated as alternatives to the existing subtypes of a type in the push-down phase of resolution.

(b) Instead of all items in the lexicon being available as pseudo-types the only structures available are those with dummy semantics, those which unify with one of the set of lexical tlink outputs in the generation bag (which may be used anywhere in generating the TS) plus those which unify with one of the set of included lexical signs. These may only be used as subparts of phrases to which their including constraint has been applied.

5. All the generated structures are then checked to ensure that all the conditions are met, that is that they incorporate results from tlinks translating all semantically contributing lexical signs in the SS, that any phrasal input tlinks that were required were actually used, and that for all phrasal output tlinks used in the structure, all the included signs are actually included in the phrase as generated.

5 Limitations

As stated above, this approach is intended to allow expression of translation divergences where languages differ systematically in the ways in which a concept is expressed, such as the differing lexicalisation patterns mentioned. We do not attempt to address all the problems of translation mismatches. One standard example of a translation mismatch is that *wear* in English should be translated differently into Japanese according to the part of the body on which the garment is worn — *hateiru* for shoes, *kiteiru* for coat and so on (Wilks, 1991). The tlink mechanism is not particularly suitable for representing such mismatches. We do not want a phrasal tlink that links *hateiru* with *wear on the feet* or any similar phrase, since *on the feet* would usually be redundant.[12] But linking *hateiru* and *wear* and attempting to represent the distinction in meaning within the feature structure for *wear* by means of lexical semantic features is an even less promising solution, since it would lead to a proliferation of features which could not be motivated monolingually, and could not be instantiated in general by the sort of processing that we are assuming. If we introduced a feature BODY-LOC for *wear* we might envisage instantiating it with "foot" in the cases where the sentence was (3a) or even (3b) but it is hardly reasonable to assume that this could be done without some inference for (3c). [13]

(4) a John wore red shoes.
b John wore boots/slippers/Doc Martens.
c The company insisted that employees had regulation footwear.
John wore these symbols of his enslavement to the capitalist system to work every day.

---

[12] In fact it is possible to envisage situations for which some of the material is redundant for most examples of MT where a phrase has to be used to translate a lexical item, including the example which we gave earlier of *novillo* being translated as *young bull*.

[13] To some extent similar arguments could also be raised against the tlinks described above where we made *teacher* with a SEX instantiated to **male** or **female** translation equivalent to *maestro* or *maestra* respectively, given that the sex feature could not be instantiated without inference in sentences such as (3)

(3) The teacher had a beard.

The justification for the tlink's reference to sex is the assumption that it is sensible to encode SEX as a lexical semantic feature monolingually even in English, because of the personal pronouns and affixes such as *er* and *ess*.

It is clear that in some cases some degree of inference on real world knowledge is required for correct translation and the tlink mechanism is not designed to supply this. In this example tlinks should be described for each possible translation of *wear* and augmented with the discriminating information. The expression of this would depend on how the distinction was to be made, whether by some further component of the MT system or by the user (as in the BCI system). It would also be possible to accept the limited degree of discrimination possible within a system such as the LKB, on the grounds that sentences such as (4c) are very rare compared with the cases where a simple mechanism would be adequate, but if the signs are augmented to allow this, the additional information should still be distinguished from that needed for monolingual processing.

6 Conclusion

The are advantages and disadvantages to a uniform constraint based approach. The concept of ultimate well-formedness with respect to a type system appears natural and is very expressive, allowing complex constraints to be captured. As described in Section 2, the original definition of well-formedness with respect to a type system seems intuitively too weak, if the type system is viewed as a complete description of the set of possible feature structures. We have a uniform notion of constraint resolution which allows structures to be generated from any sufficient amount of information, whether it be the orthography of the sentence, the logical form or the description of the component parts of the sentence and their co-dependencies which arises from the lexicalist model of transfer adopted.

However, this process of constraint resolution can be very slow. While this is not too problematic for an application such as the ACQUILEX LKB, where the primary concern is to ensure sufficiency of information in lexical entries, it is a serious concern more generally. A second related disadvantage is that there is no distinction between categories of types which are intuitively distinct — types such as **phrasal-sign**, **agr** and **synsem-list** are treated identically. Yet these encode quite different concepts and while the first relates to an entity which intuitively has some clear linguistic significance, the third is really an artefact of the formalisation. The current account of types in the LKB does not allow a distinction to be made, but it is clear that a distinction is warranted both from the viewpoint of the grammar/lexicon writer and in terms of processing considerations, since efficiency in constraint resolution depends on applying the constraints in the most effective order.

In particular, it is unclear that the HPSG account of phrasal signs as feature structures which incorporate their daughters is the best one to adopt. Constraint resolution can be used to perform operations which cannot be mimicked by more conventional grammar rules. For example, as shown in Section 3, we can model the operation of appending two lists of arbitrary length in a constraint framework and we cannot do this by simple feature structure unification. However, it is not clear to me whether HPSG currently takes advantage of this possibility in any very significant way. There have to be good reasons to adopt an approach which makes most known parsing technology inapplicable. Furthermore the description seems too general, one would like the value of DTRS to be regarded as different in kind from the value of features such as AGR — in particular a phrase which has a phrasal sign as a daughter can have a head feature which is dependent on the value of the enclosed sign's AGR feature, but should not depend directly on any values encoded inside its DTRS.

Lexical and morphological rules are also problematic within a uniform constraint based account. The conventional option of treating lexical rules as expanding the lexicon is unsatisfactory for several reasons. It implies an enormous expansion of the size of the lexicon which is computationally problematic, given that lexical rules can be applied in sequence. It is not even clear whether this process would necessarily achieve closure without an arbitrary limit being placed on the number of applications of rules. There is no clearly principled restriction on affixation and if sense extensions such as those relating count and mass usages of nouns are encoded as lexical rules (see Copestake and Briscoe, 1992) it is not clear at what point the application of these rules should terminate. Furthermore, such processes can apply to phrases, to a limited extent at least (Copestake, 1992). But the alternative of treating signs derived by morphological or lexical rules as complex with some sort of DTRS feature (by analogy with phrasal rules) also seems unattractive, because there is no linguistic motivation for regarding this information as part of the derived sign. A third possibility is to encode lexical rule like effects directly in the type hierarchy, by making the input form underspecified with respect to the output form. For example, we could have a type representing a noun which was unspecified for number which would have two subtypes, one corresponding to the singular and another to the plural form. Constraint resolution would then result in all nouns being specified as one of these subtypes. While this is a plausible way of treating inflectional morphology, it does not allow for the effects found in derivational morphology and sense extension where the usual lexical rules involve a shift in type from one part of the hierarchy to another, not simply a specialisation. Although it might be possible to develop a concept of the lexicon where the lexical entries were underspecified with respect to type and derivational and sense extension rules acted to specialise types, this would be a big move to make.

It therefore seems that there are advantages in keeping the concept of the parse tree and constituent structure distinct from the description of signs. Blackburn *et al.* (1993) discusses how both trees and feature structures may be modelled in a *layered modal language*, one layer describing the trees and the other the feature structures which decorate the nodes. This allows the formalisation of trees and signs using different modal languages with the interrelationship between them also being formally described. At least in principle, this seems preferable to attempting to encode everything in terms of feature structures. It is still possible to view this as a constraint language and such an approach is largely compatible with the description of machine translation given here. In fact it is preferable to the monostratal account, since the interrelationship between phrasal signs and their constituents is treated as a special case within the current framework. In particular, in a layered account, it would not be necessary to describe inclusion relationships within feature structures, but only within trees.

Further work on using tlinks will be reported in Copestake and Sanfilippo (in preparation). There are many problems in determining how a multilingual lexical knowledge base can be used within a real MT system, some of which apply regardless of the architecture or approach to MT assumed. One major issue which I have not addressed at all is what we actually mean by the concept of translation equivalence. It is clear that a bilingual transfer lexicon will always be restrictive, in that it will not support all possible ways of translating a phrase into a (roughly) denotation equivalent phrase in another language. For example, if a word sense in English is close enough in meaning to one in Spanish that they are adequate translations of each other with respect to denotation in all contexts, the bilingual lexicon will presumably only give the word to word translation equivalence and not an alternative word to phrase relationship, even if the phrase was also a good translation (unless the word was rare or restricted in register). In such a case, the bilingual lexicon would be restricting the options for translation correctly but in doing so would be usurping part of the function of the monolingual system since avoiding periphrasis is a general principle. (In an idealised interlingual system, on the other hand, the generation of the phrase would have to be blocked by the existence of the word.) I suspect that our concepts of a multilingual lexicon are quite closely related to experience of bilingual dictionaries. (My original description of the tlink mechanism was closely motivated by Piek Vossen's analysis of the Van Dale bilingual, but tlinks are not very different from other approaches to unification transfer based MT.) Possibly looking at what makes a bilingual dictionary good for human use would be a useful way of approaching the problem of what should, in principle, go in a multilingual lexicon.

References

Alshawi, H., D. Carter, M. Rayner and B. Gambäck (1991) 'Translation by quasi logical form transfer', *Proceedings of the 29th Annual Meeting of the Association for Computational Linguistics (ACL-91),* Berkeley, California, pp. 161–168.

Beaven, J. L. (1992) 'Lexicalist unification-based machine translation', PhD thesis, University of Edinburgh.

Blackburn, P., C. Gardent and W. Meyer-Viol (1993) 'Talking about trees', *Proceedings of the 6th Conference of the European Chapter of the Association for Computational Linguistics (EACL-*

93), Utrecht, the Netherlands, pp. 21–29.

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

Copestake, A. (1992) 'The representation of lexical semantic information', D.Phil thesis, University of Sussex, Cognitive Science Research Paper, CSRP 280.

Copestake, A. and E. J. Briscoe (1992) 'Lexical operations in a unification based framework' in J. Pustejovsky and S. Bergler (ed.), *Lexical Semantics and Knowledge Representation. Proceedings of the first SIGLEX Workshop, Berkeley, CA,* Springer-Verlag, Berlin, pp. 101–119.

Estival, D., A. Ballim, G. Russell and S. Warwick (1990) 'A syntax and semantics for feature structure transfer', *Proceedings of the 3rd International Conference on theoretical and methodological issues in MT of NLs (TMI-90),* Austin,Texas, pp. 131–143.

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

Estival, D., A. Ballim, G. Russell and S. Warwick (1990) 'A syntax and semantics for feature structure transfer', *Proceedings of the 3rd International Conference on theoretical and methodological issues in MT of NLs (TMI-90),* Austin,Texas, pp. 131–143.

Pollard, C. and I. Sag (1987) *An information-based approach to syntax and semantics: Volume 1 fundamentals,* CSLI Lecture Notes 13, Stanford CA.

Pollard, C. and I. Sag (forthcoming) *Head-driven phrase structure grammar,* Chicago University Press, Chicago.

Shieber, S.M. (1993) 'The problem of logical form equivalence', *Computational Linguistics,* **19 (1)**, 179–190.

Whitelock, P. (1992) 'Shake-and-bake translation', *Proceedings of the 14th International Conference on Computational Linguistics (COLING-92),* Nantes, France.

Wilks, Y. (1991) 'More introduction and a bibliography', Machine Translation: ACL tutorial notes.

Zajac, R. (1991) 'Issues in the design of a language for representing linguistic information based on inheritance and feature structures' in E. J. Briscoe, A. Copestake and V. de Paiva (ed.), *Proceedings of the ACQUILEX Workshop on Default Inheritance in the Lexicon,* University of Cambridge Computer Laboratory, Technical Report No. 238.

Zajac, R. (in press) 'Issues in the design of a language for representing linguistic information based on inheritance and feature structures' in E. J. Briscoe, A. Copestake and V. de Paiva (ed.), *Inheritance, defaults and the lexicon,* Cambridge University Press, Cambridge, England.