VALENCY AND MT: RECENT DEVELOPMENTS IN THE METAL SYSTEM

Rudi Gebruers

Siemens-METAL project
Katholieke Universiteit Leuven
Maria Theresiastraat 21
B-3000 LEUVEN
BELGIUM

ABSTRACT

This paper describes a valency model, developed within the Belgian METAL project, aimed at enhancing the modularity and multilinguality of the METAL system. The introduction provides background, section 1 discusses the existing valency framework, and section 2 presents the alternative model. The final section deals with some results and problems with this model.

0. Introduction

The task of MT is to map between equivalent linguistic objects. One of the central design questions in MT is that of the best method to decompose the translation relation. The ideal would be to have a system that produces a (natural) language-independent representation from a source language (SL) text, which could then be synthesized in any target language (TL). However, this ideal not being feasible for real-world texts, it has become customary to adopt a model where a transfer module, specific to one language pair, defines a mapping between language-dependent structural representations. In principle it should be possible to design a 'transfer' model in such a way that the analysis module for mapping surface strings onto structural representations and the synthesis module for mapping structural representations onto surface strings remain the same, regardless of the TL and SL, respectively. The advantage of this 'multilingual' design is that existing modules will not be seriously affected by the addition of a new language to the system. A still more attractive, but also more ambitious, design would be one in which the same grammar can be used for both parsing and generating, and the same translation rules for translating between two languages in either direction (see Jin and Simmons, 1986 for an example of a 'symmetric' translation system).

Whereas early MT systems blended the rules of grammar and the analysis procedure for efficiency reasons, it has also become customary, given current system optimization techniques, to make a clear separation between programming logic and data on the one hand, and linguistic logic and data on the other. This separation is convenient for the division of labour between the linguist and the programmer, and it enables the former to revise and complete his rule systems without the latter having to constantly change his programs.
The METAL automatic translation system tries to be multilingual in the above sense. Moreover, it makes an attempt at separating software and lingware (= linguistic knowledge written in a specialised user language). In the following, I will show how the adoption of a new kind of valency framework, developed at the K.U.Leuven in the course of the last two years, enhances the multilinguality and modularity even further. For the sake of clarity, I will first review the relevant aspects of the valency framework in the current METAL system.

1. Valency in the METAL system

Since the main claim to be advanced in this paper bears on the relation between a valency framework and the general design of an MT system, I will first say a few words on the METAL system architecture. I will then review and comment upon the valency framework in this system.

1.1. The METAL architecture

In METAL the translation process proceeds in three phases. During the *analysis* phase an input sentence is mapped onto one or more interpretations. Each interpretation is represented as a flattened phrase structure, consisting of a predicate node followed by one or more arguments (and zero or more modifiers). Anaphoric links are resolved during the *integration* phase. The resulting analysis trees are not intended to be language-independent representations, but are passed to a bilingual *transfer* phase. During transfer, analysis trees are structurally and lexically modified according to TL specifications. The output sentence is the string of terminal nodes of this transformed tree.

The METAL system accommodates two kinds of transfer/generation approaches. Most transfer instructions are paired one-to-one with the grammar rules used to perform the SL analysis. However, provisions have also been made to complement this 'direct transfer' approach with an independent transfer grammar (see Root, 1985). The latter approach is becoming more and more important in METAL because it greatly enhances the modularity of the system (viz. with an eye on using it for several different language pairs).

1.2. The valency framework

It is well-known that the dependency relations between a verb and its arguments can influence greatly the lexical and structural transfer of both, as well as the structural transfer of the clause as a whole. Though the dependency relations themselves may be language-independent, their encoding varies from one language to another, and, within one language, from one verb to another. It is therefore essential to know for each verb what its dependents should look like. This topic being central in Valency Grammar, it is not surprising that many MT systems (e.g. TAUM-AVIATION, SUSY, GETA, ARIANA-78, EUROTRA) incorporate valency notions (see Somers, 1986). One essential notion borrowed from valency theory is that of 'valency frame', i.e. a pattern listing all the complements allowed and/or required by a verb, together with associated morpho-syntactic and/or syntactic features.

Since German-English is the furthest developed language pair of METAL at present, I will now discuss what the valency frames look like for German. In the METAL system the German valency frames mainly include morpho-syntactic information (syntactic (sub)category and/or surface case) with respect to non-subject arguments. For instance, the pattern *(A-X (CP TH))* signals that a German verb carrying it may take, besides a nominative subject, an accusative reflexive pronoun *(A-X)* plus a complement phrase introduced by *dass (CP TH)*. The optionality of arguments is not signalled in the frame itself, but in a separate feature on the verbal predicate.

In *analysis*, predicate-argument structures, resulting from a flattening and rearrangement of constituent structures, are passed to a *case frame processor*. The latter attempts to 'use up' the available sentence constituents by matching them against the argument specifications in the valency frame(s) specified for the verbal predicate. When it finds a constituent that matches an argument specification in the frame, it updates this constituent with a grammatical role (SUBJECT, (IN)DIRECT OBJECT, etc.) according to some general implication relation holding between grammatical roles and case markings (or some other sort of coding). For a clause to be well-formed, at least one of its verb's frames must have a 'filler' for each of its argument 'slots'. If a frame is found to be applicable in more than one way, preference is given to one application on the basis of word order criteria and/or semantic properties of the subject argument. Eventually, the case frame processor returns either an analysis tree that has been updated with grammatical role information, or it discards the input sentence.

During the *transfer* phase, morpho-syntactic information of the sort present in valency frames may be used, both in tests and in transformations associated with lexical transfer entries, to attune SL argument specifications to the TL. In addition, transfer entries may contain further semantico-syntactic restrictions on argument positions, which may help in choosing the right translation for a verb. The grammatical roles are used to convert the canonical ordering of the translations of the verb and its arguments into the appropriate TL ordering, as well as to generate the appropriate forms of the TL constituents.

1.3. Discussion

Before discussing the new, extended valency framework, we will briefly point out how the existing system does not yet exploit the potential of a valency grammar to the full.

The only valency frames referred to by the case framing package in the course of translation are SL valency frames. There is no procedure for mapping SL frames onto TL frames, and the information provided in TL frames is not used when TL strings are generated. The underlying assumption seems to be that argument structures are *more or less the same across languages*. Any discrepancies with respect to the argument structure are resolved by means of a small set of transformations specified in the relevant lexical transfer entries. Any discrepancies with respect to the expression of the argument structure (e.g case-marked vs. unmarked NPs) are handled in the relevant grammar rules.

The assumption that argument structures are more or less the same across languages is also reflected in the status of the canonical clause representations employed in the METAL system. The latter are considered to be some sort of interlingual structures from which TL surface strings are to be generated directly (cf. the direct transfer approach). However, the general philosophy in the METAL system has been to start off with a rather 'shallow' level of analysis, rather than a 'deep representation' of some sort (see [SLOCUM 83]). Thus, there seems to be a conflict between the reluctance to work with a more semantically oriented analysis and the desire to have an interlingua. This conflict may have been negligible for the German-English system, because these languages are 'cut' along very similar patterns. Nevertheless, even these two languages display subtle differences as to the way they 'model' extralinguistic reality. For instance, 'helping' is a real-world relationship involving two entities, A and B. In English, this relationship is construed as an action of A which affects B (*A helps B* is similar to *A hits B*); in German, it is modelled as if A transferred something to B (*A* [nom] *hilft B* [dat] is similar to *A* [nom] *gibt B* [dat] *C* [acc]). Similar differences may be expected to increase as languages more divergent than English and German are to be handled. If, for some reason or other, it is not feasible or desirable to reduce language-specific models of some real-world relationship to a language-independent case frame, there is nothing but to state translation equivalences between clause structures in terms of equivalences between *language-dependent* argument structures. (For similar views, see Alam, 1986; Kudo and Nomura, 1986; Van der Korst, 1987.)

Although there is little doubt that the framing facilities provided in the system work quite well and yield very good results for translations from German into English, we have tried to improve the framing module beyond this language-pair. One should also bear in mind that, with a less well-structured MT system than METAL, we could never have developed a more language-independent valency mechanism in such an easy and straightforward way.

2. An alternative valency framework

2.1. The architecture

The general philosophy behind the development of the Leuven valency framework has been to maintain an essentially syntax-driven MT system, while enhancing the latter's modularity in view of extensions to other language pairs. This required reconsidering not only the relation between lingware and software, but also the general architecture behind the system.

With respect to the general translation theory behind the METAL system, enhancing the modularity boils down to increasing the relative independence of the analysis, transfer, and synthesis modules. More specifically, we assume that

(a) an analysis module must provide representations which are useful starting-points for translation into multiple TLs;
(b) major parts of a synthesis module must be independent of the SL under consideration so that they can also be used for translation from other SLs;
(c) mappings between SL and TL representations must be defined in terms of a minimum of, preferably, lexically governed transformations.

Though we are still far away from the ideal transfer-based MT system, we believe that the alternative valency framework may be an important step in the right direction.

2.2. The linguistic fundamentals of the alternative valency framework

The basic assumption is that simple clauses have a predicational structure and that (partial) equivalences between SL and TL clauses can be defined in terms of (partial) equivalences between SL and TL predications. The structural centre of a predication is a lexical predicate with which one or more valency frames are associated. Each valency frame is a sequence of typed *argument* slots to be filled with appropriate terms, i.e. sentence constituents of the appropriate types. Sentence constituents which cannot be related to any of a predicate's argument slots should be 'legal *satellites*', i.e. legal circumstantial modifiers, of the predication as a whole.

The structure of valency frames is language-independent, and can be defined as follows:

```
<frame>          ::=  "(" <slot>+ [ "OPT" <slot>+] ")"
<slot>           ::=  "(" <slot_label> <key>+ ")"
<slot_label>     ::=  one of a set of user-definable atoms,
                        starting with a "$"-sign
<key>            ::=  <code_pointer> | <feature-value-pair>
<code_pointer>   ::= one of a set of user-definable atoms
<feature-value-pair> ::= "(" <feat_name> <feat_val>+ ")"
<feat_name>      ::= one of a set of user-definable atoms
<feat_val>       ::= one of a set of user-definable atoms
```

The number of argument slots for a given frame is primarily determined on the basis of formal, language-specific criteria. Thus tests to distinguish between arguments (*args*) and satellites (*sats*) include, besides the elimination test, paraphrase tests (cf. the *do so* and *und zwar* tests, for English and German, respectively), as well as distributional and substitutional criteria (e.g. *sats* are freer to move than *args*, whereas elements of pronominal paradigms substitute more easily for *args* than for *sats*). Whenever those tests are not decisive with respect to the status of a sentence constituent, the latter is assumed to be an arg, since, for transfer, it is arguable that it is easier to operate on args than on sats.

Argument slots may be obligatory or optional. Optional slots, which need not be present, but are always semantically implied, are set apart from obligatory ones by means of the symbol OPT. In fact, OPT is a means to collapse frames whose obligatory slots are identical and whose optional slots are not mutually exclusive. Thus, a frame containing *n* optional slots is an abbreviation for *2expn* different frames.

Argument slots are not in themselves labelled semantically, though they do tie up with semantic relations (deep cases) as all valency relations are ultimately semantically motivated. (See Helbig and Schenkel (1973) for a discussion of the relation between logical, semantic, and syntactic valency.) Instead, a slot label is taken to signal that certain rules or regularities apply to all the args carrying that label. Our basic principles for labelling slots are the following:

(a) Args labelled *$0* are 'deep subjects'; typical surface reflexes in Dutch and French are *'nominative* case', position to the left of the finite verb in unmarked declaratives, ability to become the agentive phrase in passives (under certain conditions);

(b) Args labelled *$1* are 'deep objects'; typical surface reflexes in Dutch and French are *'accusative* case', position strongly tied to the main verb, ability to become the 'surface subject' in passives (under certain conditions);

(c) Args labelled *$2* are 'indirect objects'; typical surface reflexes in Dutch and French are indirect object prepositions (*aan* vs. *a*), alternation between PP[*aan* vs. *a*] and NP[non-clitical and lexical vs. clitical and pronominal];

(d) Args labelled *$3* are 'oblique objects'; a typical surface reflex in Dutch and French is that these args can be replaced by adverbial constructions;

(e) Args labelled *$4* are 'prepositional objects'; in both Dutch and French these args are PPs, with strongly governed, idiosyncratic prepositions;

(f) Args labelled *$5* are 'subjective complements'; these args are attributes of the subject with bivalent verbs (e.g. *zijn/etre* 'be');

(g) Args labelled *$6* are 'objective complements'; these args are attributes of the direct object with trivalent verbs (e.g. *noemen/appeler* 'call').

It is important to note that those principles are rules of thumb, rather than clear-cut definitions in terms of necessary and sufficient conditions. However, far from being arbitrary, the inventory of arg labels should be justified both language-internally and cross-linguistically. *Language-internally*, this means that one has to come up with a number of indications that a given arg label allows for significant language-specific generalizations. *Cross-linguistically*, this means that, in assigning the same label to slots in different languages, it must be possible to reveal a reasonable degree of overlap in the behaviour of fillers for the slots in the respective languages. Of course, we do not pretend that our list of arg labels is in any way exhaustive and we grant that it may have to be adjusted in the light of further research.

Apart from a slot label, an argument slot contains a number of 'keys' which refer to procedures, frame tests and frame constructors, to be called during analysis and synthesis, respectively.

Frame tests consist of morpho-syntactic and, possibly, semantic conditions which constituents must satisfy in order to become potential slot fillers. The actual contents of the morpho-syntactic constraints is co-determined by such parameters as the slot label and the clause's Mood and Voice values. Regarding the use of semantic selection restrictions, a fairly pragmatic course has been pursued. That is, we started off with a rather limited inventory of semantic features ([± person(alized)], [± abstract], etc.) which we think to be consistently applicable and flexible enough to be extended and/or changed when the need arises. Apart from semantic selection restrictions on filler constituents, it is possible to include (canonical) lexical forms in a slot specification list. These may refer to the form required in either the 'relator' (e.g. the preposition in a PP or the conjunction in a subordinate clause) or the 'head' of filler constituents. The latter functionality has been provided in order to handle more or less idiomatic NPs (e.g. *een keuze* in *een keuze doen*, 'make a choice') which are still sensitive to regular syntactic operations (e.g. passivization).

Frame constructors consist of instructions according to which the system should generate the appropriate surface form required for fillers of the slots from which those constructors are called. Again, the actual content of the instructions is co-determined by such parameters as the slot label and the clause's Mood and Voice values, as well as by lexical information provided in the slot.

It is important to note two things. First, it is actually the same codes that are used for both tests (Analysis) and constructors (Synthesis). The system knows from the translation phase whether it has to interpret the code as a test or as a constructor. Second, both frame tests and frame constructors are stored in separate files, in order to enhance the modularity of the lingware. However, they are written in the same format as ordinary grammar rules, and a special interface for loading, inspecting and editing these procedures has been developed, so that they can be *easily accessed and updated by the linguist*.

As an illustration of the above, let us consider the following (incomplete) list of valency frames which occur as elements in a value list to the feature ARGS for the verb *faire*:

```
(
 (($0 N1 P1) ($1 N1 P0)
 OPT ($2 N1 P1 (PREP pour)))     ;'make'
 (($0 N1)($5 A))                 ;'look'
 (($0 N1)($3 MEA))               ;'do'
 ...
)
```

What is said here is that *faire* can show up in (at least) three different frames. The first one contains two necessary and one optional slot. The first slot requires a nominal filler with selection restriction [+personal(ized)], the second one a nominal filler with selection restriction [-personal], and the optional one a prepositional complement introduced by *pour* and having selection restriction [+personal(ized)]. The second frame contains two obligatory slots, the first of which requiring a nominal filler and the second one an adjectival complement. Finally, the third frame requires a nominal filler and a measure constituent. The respective frames are illustrated in the following sentences:

(1) *Je* ($0) *fais ce jouet* ($1) *pour mon ami* ($2)
'I make this toy for my friend'

(2) *Elle* ($0) *fait vieille* ($5)
'She looks old'

(3) *Cette voiture* ($0) *fait 100 km/h* ($3)
'This car does 100 km/h'

An example of a (French) frame test is given in fig. 1. It is invoked by the key P1, when called for the slot labelled $2 (as in the first frame of *faire*). Comments explaining the test instructions are given in italics. Note how different contextual restrictions have been assembled in one test procedure.

In fig. 2, we give an example of one (French) frame constructor which is called by the key N1 from a slot labelled $2.

```
P1-$2

(SONS ($2SON = (INT ? $2 N1))
        single out nodes which passed the $2-N1 test
     DO
     for each of these nodes,
     (COND ((INT $2SON TY HUM HI)
            if it has semantic type [+human]
            or [+human intervention],
            then succeed
            (PUT ($2 P1)))
           ((INT $2SON KP REL)
            if it contains a relative pronoun,
          then succeed
            (PUT ($2 P1) (TR HUM HI)))
           ((RET $2SON KP)
            if it contains a pronoun,
            then succeed
            (PUT ($2 P1)))
           (T
            else, unmark the node as a candidate
            for the $2 slot
            (RMV $2SON $2))))
----------
```

fig.1: frame test

```
N1-$2

(AND (RET 0 PREP)
    if there is a PREP feature on the father node
    (this feature has been retrieved from a
    verbal entry)
    (OR (XFM (&:1 (--:2 (NP:3 NIL (INT 3 ROL $2)) --:4))
     and if there is a NP son marked $2,
     then create a TL-PREP node in front of
     it and make both dependent on a
     new PP node marked $2
           (&:1 (--:2
           (PP:5 ((PREP:6 NIL (TRF 1 PREP))
           (NP:3 NIL (RMV ROL)))
           (PUT (ROL $2)))
           --:4)))
      (XFM (&:1 (--:2
           (PP:3 (PREP:4 &:5) (INT 3 ROL $2))
            --:6))
     or if there is a PP node marked $2,
     then translate its PREP node
         (&:1 (--:2
         (PP:3 ((PREP:7 NIL (TRF 1 PREP)) &:5))
         --:6)))))
----------
```

fig. 2: frame constructor

2.3. The valency procedure

The valency procedure is composed of three subprocedures (see fig. 3). Two of them use purely language-specific material, while the third one has to establish a link between material from two different languages. What is important to know, however, is that we hold the overall organization of all three subprocedures to be completely language-independent. As a consequence, it should be easy to plug language and language-pair specific information into these procedures, without the latter having to be adapted for each new language pair. We think this modularity is a substantial improvement as compared with the valency procedure incorporated in the LRC-METAL system.

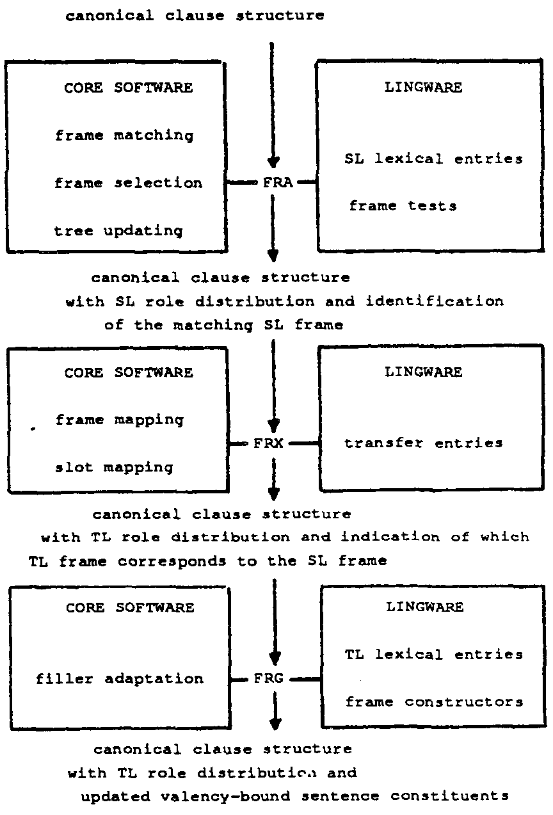
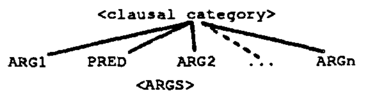

canonical clause structure
CORE SOFTWARE
frame matching
frame selection
tree updating
FRA
LINGWARE
SL lexical entries
frame tests
canonical clause structure
with SL role distribution and identification
of the matching SL frame
CORE SOFTWARE
frame mapping
slot mapping
FRX
LINGWARE
transfer entries
canonical clause structure
with TL role distribution and indication of which
TL frame corresponds to the SL frame
CORE SOFTWARE
filler adaptation
FRG
LINGWARE
TL lexical entries
frame constructors
canonical clause structure
with TL role distribution and
updated valency-bound sentence constituents

fig. 3 : valency procedure

(FRA, FRX, FRG are the drive functions called from within grammar rules during analysis, transfer, and synthesis, respectively)

2.3.1. During *analysis*, the valency procedure is invoked from within grammar rules for building clausal structures of the following format:



Given a verb with a set of valency frames and a set of sentence constituents, this procedure has to make sure that one of these frames is realized in the sentence at issue, and, if so, it has to make clear how that frame is realized. It will take a frame to be realized in a tree, if and only if each of the frame's non-optional slots, and possibly, one or more of its optional slots, is matched by at least one sentence constituent. If the system finds a matching frame, it will ultimately return a tree structure in which each of the valency-bound constituents is marked for a slot and whose root node contains a reference to the matching frame.

Now, the general idea is to have the procedure look for the *most ambitious* frame matching the tree structure, as well as for the *best realization* of this frame in the structure, while avoiding superfluous processing as much as possible. This implies that two kinds of preference mechanisms had to be introduced in the valency procedure: one to choose the best candidate from a set of potential fillers for a given slot (instead of always choosing the first constituent matching the specifications of that slot), and one to choose the most ambitious frame from a set of successful frames (instead of always choosing the first frame that matches a given analysis tree).

The first preference mechanism is implemented in the following way. When checking whether a frame matches the tree, the valency procedure collects for each slot *($i key1...keyN)* all sentence constituents which pass all the frame tests associated with the keys in that slot. Furthermore, in the action part of frame tests, each potential filler gets a number indicating the probability that this constituent will be taken as the ultimate filler for a slot. As the linguist can easily alter this number, he has significant control over the assignment of constituents to slots.

The actual assignment procedure is fairly economical and runs as follows. Whenever there is only one potential filler for a slot (which may be either an obligatory or an optional one), this constituent loses its marking as a candidate filler for other slots, if it was one. Furthermore, it is marked as the only remaining filler for the slot that is being matched. A side-effect of this marking may be that one of the other slots will now have only one candidate filler. In that case, the latter constituent will be marked as the actual filler for that slot and lose its marking as a potential filler for still other slots, if it was one. This may again cause the number of potential fillers for yet another slot to be reduced to one, in which case the above marking (and unmarking) procedure starts over again. If, eventually, there is more than one potential filler for a slot, the procedure takes the leftmost candidate which received the highest preference value for the slot in the frame tests.

The second preference mechanism is fairly economical as well. The general assumption is that, in order to find the most 'ambitious' frame, one should look for the frame which has the largest number of slots realized. In order to avoid superfluous processing, we let the more complex frames (i.e. the ones with the larger number of slots, whether obligatory or optional) precede the less complex ones in the ARGS value of a verb. Consequently, the system comes across the former before it sees the latter. Whenever two or more alternative frames happen to have the same number of slots, the lexicographer has to determine (e.g., on the basis of frequency of occurrence) which frame to try first. Given that the system has found a matching frame, it will only explore an alternative frame if the number of (optional and non-optional) slots contained in the alternative frame outnumbers the number of slots found to be realized during the matching of the first frame. An alternative frame will be preferred only when it has more slots realized than the previous matching frame.

2.3.2. During *transfer*, since we take argument structures to be language-specific entities, the valency procedure has to accomplish two tasks. First, it has to determine which TL frame corresponds with the SL frame that has been found to be applicable to the analysis tree. Secondly, it has to specify which slots in the TL frame correspond with which slots in the SL frame. It performs those tasks in the following way.

First, the mechanism looks up all the transfers for the SL verbal predicate. Each of the verbal transfer entries describes a transition between the SL verb with one of its frames, and an equivalent TL verb with one of its frames. It does so in terms of (a) *conditions* on the transition and (b) (possibly partial) *mappings* between SL slots and TL slots. The condition part of a verbal transfer entry may be empty or take any of the following forms, for disambiguation w.r.t. the TL:

(a) a test on the presence (absence) of some slot filler in (from) the SL tree;
(b) a test on the presence (absence) of certain lexical or grammatical information on some slot filler in (from) the SL tree;
(c) a test on the presence (absence) of some feature on (from) the root node of the SL tree.

As for the mappings between SL slots and TL slots, three possibilities have been catered for so far:

(a) *SL slot maps onto TL slot*. In this case, we only state contrastive information which is strictly necessary to effect an appropriate mapping between SL slots and TL slots. That is, in general, equivalences between distinct slot labels will suffice, though additional information may be provided for disambiguation.
(b) *SL slot without TL counterpart*. Again, only minimal information needs to be specified in order to identify the SL slot whose filler must be removed from the tree structure.
(c) *TL slot without SL counterpart*. Here, the coder should be able to describe the internal structure of a TL constituent to be created at clause level in the tree structure. At the moment, the functionality provided is limited to the creation of new TL nodes without internal structure.

Having retrieved all the transfers for the SL verb, the system reduces the potential transfer ambiguity of a SL verb in two steps. It first discards any transfers for which not all conditions are fulfilled. Afterwards, it checks which of the remaining transfer entries (there should be at least one) provides a frame equivalence whose 'left-hand side' can be linked to the frame realized in the SL tree. Once this equivalence has been found, the TL frame matched by the 'right-hand side' of the frame equivalence will be substituted for the SL frame referenced by a feature on the root node of the tree. At this stage, nodes can be pruned from or added to the tree structure, if lexical instructions tell the system to do so.

Finally, after translation of the verbal predicate, the system exploits the equivalences between SL slots and TL slots in order to determine how the translations of the sentence-level constituents fit into the slots of the TL frame.

The above matching procedure complicates the lexicographer's task considerably (see section 3. for how we try to remedy this situation). This is because it requires that verbal transfer entries be written in such a way that each of them provides

(a) a link with a SL verb and *exactly one* of its frames;
(b) a link with a TL verb and *exactly one* of its frames;
(c) sufficient information concerning the slot equivalences holding between these two frames.

On the other hand, it has the advantage of giving the dictionary writer significant control over verbal and clausal transfer.

2.3.3. The task of the valency procedure during *synthesis*, as we conceived of it, consists in guiding the generation of the appropriate surface form of valency-bound sentence constituents. Furthermore, the synthesis component contains constituent ordering procedures which may refer to slot labels in order to rearrange the canonical clause structure into the appropriate TL ordering.

First, the valency procedure retrieves the TL-frame from the root of the tree. For each slot contained in this frame, it then checks whether it contains any lexical information specific to the TL verb and the frame that are at issue (e.g., the third (optional) slot in the first frame of *faire* has to be filled by a prepositional phrase introduced by *pour*) and makes this information available for further processing. Next, it calls all of the frame constructors associated with the keys in the slot. During those calls the relevant sentence constituents will be modified and updated according to the instructions that the linguist specified in the constructors. Eventually, the valency procedure should return a tree all of whose valency-bound constituents contain sufficient information so that morphological rules and linearization rules can generate the appropriate TL forms and constituent ordering (the latter rules may occasionally alter the constituents' forms).

Again, the linguist has significant control over the generation process as he can easily specify and update the contents of both frame constructors and linearization rules.

3. Results and Problems

In the last year, the alternative valency framework presented in this paper has been applied to the translation of Dutch into French, and vice versa. Though the application has been limited to "kernel" sentences, i.e. simple active and passive declarative sentences (possibly containing relative clauses of the same type), the results seem fairly promising. At the moment, provisions are being made to handle non-finite valency-bound subclauses.

In the meantime, small conversion experiments have been conducted for the language pairs German-English (at K.U.Leuven) and German-Spanish (at CDS Barcelona). These experiments have shown that, at least for the admittedly very limited domain of application, the Leuven valency framework works very well. Its main advantages seem to be the following:

(a) it provides the skeleton for a really transfer-based MT system, since it clearly separates three subprocedures to be invoked during Analysis, Transfer, and Synthesis, respectively;
(b) it allows for a neat separation of kernel software and application-specific lingware and provides user-friendly facilities to access and update the latter;
(c) its methodological underpinning to a certain extent allows languages to be treated independently of one another.

Because of these advantages, the Leuven valency framework has recently been adopted by all sites of the METAL project.

However, serious problems remain to be tackled, with respect to both lexicon coding and grammatical parsing. The first kind of problems can be traced to the rigidity of the frame mapping schema itself. As has been pointed out in section 2.3.2., the main requirement for frame mapping to be possible is that verb lexicons be coded consistently across languages. This may indeed be profitable in an experimental environment. However, it is doubtful whether we can expect the average end user to have both source, target, and transfer codings in mind at the same time and to make sure that no aspect of the mapping between frames is overlooked. Therefore, we have developed provisional relaxations on the rigid schema to the extent that at least the mandatory slots of the TL frame must have a counterpart in the SL frame. Eventually, however, it may turn out to be more effective not to have these relaxations at run time, but to sort out inconsistencies between verbal lexicons at coding time.

The second kind of problems (among which the notorious difficult problem of PP-attachment) has to do with a need for still greater functionality in the system. In order to provide for this functionality, we envisage two paths of further research. One path concerns how we can make our valency mechanism interact with a mechanism to identify peripheral constituents, which orbit around the verb and its arguments. The other path concerns the extensibility of the valency framework to nouns and adjectives. Preliminary research along both lines has revealed that
there are no objections of principle against the valency framework presented in this paper.

ACKNOWLEDGMENTS

Thanks are due to all the members of the Leuven METAL-project who helped develop the valency framework presented in this paper. The author would also like to thank Geert Adriaens en Herman Caeyers for their helpful comments on earlier drafts of this paper.

REFERENCES

Alam, Y.S. 1986. 'A lexical-functional approach to Japanese for the purpose of machine translation.' *Computers and Translation* 1(4): 199-214.

Bennett, W.S. & J. Slocum 1985. 'The LRC machine translation system.' *Computational Linguistics* 11 (2-3): 112-121.

Helbig, G. & W. Schenkel 1973. *Woerterbuch zur Valenz und Distribution deutscher Verben.* Leipzig: VEB Verlag Enzyklopaedie.

Jin, W. & R.F. Simmons 1986. 'Symmetric rules for translation of English and Chinese.' *Computers and Translation* 1(3): 153-167.

Kudo, I. & H. Nomura 1986. 'Lexical-functional transfer: a transfer framework in a machine translation system based on LFG.' *Proceedings COLING 86*, 112-114.

Root, R. 1985. 'A two-way approach to structural transfer in MT.' *Proceedings of the 2nd ACL 1985*, 70-72.

Slocum, J. 1983. 'A status report on the LRC machine translation system.' *Proceedings of the ACL-NRL Conference on Applied Natural Language Processing*, 166-173.

Somers, H.L. 1986. *Valency and Case in Computational Linguistics.* Edinburgh: Edinburgh University Press.

Van der Korst, B. 1987. 'Twelve sentences: a translation procedure in terms of Functional Grammar.' *Working papers in Functional Grammar* 19, University of Amsterdam.