UNIFICATION-BASED LEXICAL TRANSFER

Maria Vilkuna
University of Helsinki
Research Unit for Computational Linguistics

Artikkelissa esitellään Helsingin yliopiston Tietokonelingvistiikan tutkimusyksikössä vuodesta 1988 tehtyä transferpohjaista käännösjärjestelmää, jonka keskeiset piirteet ovat unifikaation käyttö perusoperaationa sekä leksikalistinen lähestymistapa. Kaikki järjestelmän käyttämä lingvistinen tieto on koodattu leksikoihin, joiden rakennetta, spesifikaatiokieltä, suhdetta toisiinsa ja lingvistisiä ratkaisuja artikkeli selostaa lingvistin kannalta.

This is a report on ongoing work on machine translation in the Research Unit for Computational Linguistics at the University of Helsinki, supported by IBM Finland. Initiated in 1987 as a member of a multilingual transfer-based project with a shared English analysis phase, the project adopted a unificational approach in 1988. The project is designed for translating technical texts, such as computer manuals, from English to Finnish. The experimental work done this far is lexically and structurally oriented; no attempts to solve discourse-related translation problems have been made yet. At the end of 1989, the system contained a transfer lexicon of some 500 entries and could manage simple declarative and imperative sentences with various types of complementation and modification, including sentential complements and adverbial clauses.

The point of view in this paper is that of an "ordinary working grammarian". The paper first discusses the motivation for choosing a unification-based framework and describes the specification language used in the linguistic descriptions. The organization of the lexicons is then discussed in more detail.

All linguistic information in our system, and hence in most examples in this paper, are ultimately represented in simple attribute-value graphs. For the formal properties of such graphs, as well as those of unification in general, the reader is referred to Shieber (1986), Carlson and Lindén (1987) and Carlson (1988). Carlson (this volume) explores some design issues concerning the lexicon formalism.

1. Introduction

It is customary to differentiate between two fundamental approaches to machine translation: interlingual and transfer-based. An interlingual system first maps the source language (SL) expression to a purportedly language-independent representation, interlingua (IL), and then performs a further mapping from this to the target language (TL). The IL representation is typically considered to be a complete semantic representation of the expression to be translated. We have chosen the transfer approach, which involves a more structure-oriented mapping between two language-particular representations (see Carlson, this volume, for discussion). But there are different ways of doing transfer.

A transfer based system typically consists of the following basic modules:

- **Analysis**: Building, without reference to TL, a SL specific syntactic representation, such as a tree with lexical items as terminal elements.

- **Transfer**: Choice of TL equivalents of SL lexical items by an algorithm operating on that syntactic representation, on the basis of a bilingual dictionary (**lexical transfer**); and a set of transfer operations, i.e. structural changes or transformations, on the resulting representation to produce a more TL-like representation (**structural transfer**).

- **Generation**: further structural operations, now based on purely TL specific information, to yield a TL sentence. Minimally, these operations involve the actual production of the TL word forms.

There are some inherent problems with this type of approach. The first is its procedural nature. The more sequential changes, cutting and pasting in the transfer phase, the greater the likelihood that the linguistic relations between the languages get obscured in the detail of the procedural execution, and that the procedures themselves become hard to understand and maintain.

Second, the precise nature and status of the intermediate representations remains obscure in the translation process. It is hard to justify a "half SL, half TL" representation, and the stage where purely monoligual generation begins is hard to define in practice.

Third, although different ways of integrating lexically conditioned and more general transfer operations can be developed, the exact relation between the two is unclear. Choosing one lexical equivalent over another requires reference to the general structural context the item is situated in. On the other hand, the effects of choosing one particular equivalent may be seen in the overall structure of the output.

Our solution to the above problems was to adopt

- a **unficational** approach that would enable us to state a static transfer relation between different pieces of information about SL and TL and leave its execution to be done by appropriate procedures in a completely order-free manner; and

- a **lexicalist** approach that would enable us to avoid sharp lines between lexical and structural transfer.

Our conviction is that the main effort in building a Machine (Assisted) Translation system must be dedicated to the lexicon(s). Since dictionary building is a heavily time-consuming activity, it is particularly important that the information be independent of particular procedures applying that information.

2. Transfer relations and transfer feature structures

Instead of formulating the translation process as a series of transformations, we state a symmetric transfer relation between independently motivated pieces of information about the two (or, in principle, more) languages in question. These pieces are often words, ie., lexcial entries, but they can be multiword phrases, individual features, semantic structures, and so on. Insofar as different pieces of partial information are consistent, they can combine into more complete structures that still conform to the transfer relation. The result is not affected by the order in which the combinations are made.

Unlike in the procedural approach sketched above, all linguistic representations processed by the transfer programs are representations of potential intertranslatability relations between the source and target languages, some applicable to the expression in question, some not.

Let us now look at what such transfer relations look like. In our model, the expression of the transfer relation relies on the property of structure sharing inherent in graph representation. A graph of the form

```
(1)     [SUBJ:#1
         VCOMP:[SUBJ:#1]]
```

partially describes a typical subject-control verb, whose subject is identical to the subject of its infinitival complement. This is encoded in the identical numbering. The unification formalism guarantees that the properties of the VCOMP's subject are always locally available, which is essential for selection of translation equivalents of the VCOMPs.

The same notion of structure sharing is applied to the statement of transfer relations. A simple example:

```
(2)     [E:[TENSE:#1]]
        [F:[TENSE:#1]]
```

To allow recursive statement of transfer relations, our system supplies each potentially translatable graph - the root graph, its subject and VCOMP, the VCOMP's object, etc. - with two particular attributes, E for English, and F for Finnish. Such graphs are called transfer feature structures (TFS). An example of a recursive TFS is (3). Here *open* and *aueta* are stated to be translation equivalens, and similarly their respective subjects. The graph thus (partially) represents the transfer relation between the sentences *The box opened* and *Laatikko aukeni*.

```
(3)     [E:[LEX:OPEN
            SUBJ:#1[E:[LEX:BOX]]
                   [F:[LEX:LAATIKKO]]
            TENSE:#2]]
        [F:[LEX:AUETA
            SUBJ:#1
            TENSE:#2]]
```

The following is an example of a slightly less straightforward correspondence. (4) is frequently needed when an English PP translates to a case-marked NP in Finnish. The value of the CASE attribute is here left open, as it depends on the English preposition, the nature of the object of the preposition, the nature of the governing word, to mention just a few things.

```
(4)    [E:[CAT:PREP
           OBJ:#1]]
       [F:[#1
           CASE:#2]]
```

As can be observed from (2) - (4), the transfer relations are symmetric; they do not reveal that we are translating *from* English *to* Finnish. Although our system as a whole is not automatically reversible, this symmetry in the basic representation gives us a good start in that direction.

3. The translation process

We can now summarize the translation phases in our system. To obtain the initial English feature structure, our implementation uses PEG, an English parser developed by IBM (Jensen 1986).[1] Since PEG was not built on unificational principles, its output must be pre-processed for our purposes.

The parser produces a record structure describing the English sentence, and this record structure is converted into a TFS acceptable to graph unification. This graph has the E attributes at appropriate places, waiting for their F counterparts, as in (5):

```
(5)    [E:[LEX:OPEN
            SUBJ:[E:[LEX:BOX]]
            TENSE:PRES]]
```

The transfer algorithm, given the English information in the TFS, completes the graph into a bilingual English-Finnish TFS, which would look like (3) (with tense specified). The algorithm goes through the nodes of the TFS and adds the compatible - both English and Finnish - information it finds in the transfer dictionary. All values of English LEX attributes - i.e., "words" - induce a check in the dictionary, but some grammatical features have transfer rules as well. The outcome has the contents of both E and F attributes fully specified. Dropping the E attributes gives us a Finnish graph representation of the sentence.

The Finnish elements in the TFS are then supplied with linear order and morphological form. The Finnish nodes with LEX attributes are ordered by Linear Precedence rules referring to various feature information in the graph. The base forms and all their morphologically relevant attributes in each lexical node are collected, and these specifications are turned into word forms by Koskenniemi's

[1] The use of an independent parsing module is the main difference between our system and that of the LFG-based unificational translation in Kaplan & al. (1989).

morphological generation program, based on his Two-level morphology.

Our present interest in this paper is the second phase, which we can call Transfer. Generation in this model is restricted to dealing with such aspects of the output that do not have a representation in attribute-value graphs. This means actual left to right order - as opposed to information regulating this order, which may very well be included in the graph - and word-form realization: morpheme concatenation and morphophonemic adjustments.

It might look like we had a broader concept of Transfer than some other systems. For example, since there is no other phase to add information about Finnish grammatical case, object case-marking is fully specified in the transfer output. As the details of Finnish case-marking are clearly not a bilingual matter, it is important to remember the role of Transfer in our system: it relates monolingual pieces of information. These pieces of information themselves reside in monolingual lexicons. Before turning to the organization of the lexicons, however, it is necessary to briefly describe the representation formalism and specification language.

4. Linguistic representation

4.1. Simple graph unification

All information, be it lexical entries (bilingual or monolingual), grammatical construction types, semantic types, or translation instructions, is given in the form of attribute-value graphs. The graphs are the internal representation the system sees when it is applied. What the linguist sees and writes are usually not attribute-value specifications but abbreviations of these, called templates. I shall first mention some properties of the graph formalism, then introduce templates.

Our system at this point applies the simplest possible graph unification formalism. Rules can only add positive definite information. There is no negation. We can give the "false" value for a binary attribute, or the *NONE*, i.e. 'absent' value for any attribute, but one particular value of some attribute cannot be simply denied. Instead, the attribute is given some other value that blocks the occurrence of the one not wanted.[2] Nor does the system provide for disjunctions in graphs. Disjunction in the actual entries is always expanded into distinct graphs. Thus, specification (6a) yields the two graphs in (6b) and (6c).

```
(6)   a. ((num sg) (case (!or ptv nom)))

      b. [NUM:SG
          CASE:PTV]

      c. [NUM:SG
          CASE:NOM]
```

[2] This was the situation at the time the paper was read. In early 1990, Krister Lindén implemented a monotonic version of atomic value and feature negation.

There is at the moment no "type checking" on information allowed by different types of graphs. Nothing prevents a finite clause from getting grammatical case, unless the grammar writer has made that impossible by writing (CASE *NONE*). Nothing prevents arguments from merging into one another, unless they are specified with conflicting values. There's no upper limit to the amount of attributes and values at any point.

Our only device outside simple graph unification is for ensuring completeness. As with the constraint equations in LFG (Bresnan 1982: 207 - 209), the transfer algorithm, after the transfer process proper, discards graphs containing an attribute with the value *ANY*. This is a special attribute that unifies with anything except *NONE*, and its presence in a graph reveals that it should have done so. *ANY* prevents a potential objectless output in the case of such verbs as *contain, sisältää*. It is also used in transfer entries when the presence of some participant is relevant for selection. Thus, to translate the verb *start*, the system selects Finnish *alkaa* when VCOMP is *ANY*, or OBJ is *NONE* in English; *aloittaa* is chosen when OBJ is *any* and VCOMP is *NONE*.

4.2. Templates

As mentioned, lexical entries are not stored in the form of attribute-value graphs in the lexicons. Only minor pieces of information are directly expressed by feature-value pairs. Pieces of graphs are abbreviated by named templates[3], which are typically referred to by other templates. The readability, extendability and maintainability of the lexicons depend crucially on how the templates are built and expressed.

There are two types of templates. Simple templates are lists of type (*a b c* ...), where *a* is the name of the template, and the rest consists of either other template names or atomic feature value pairs. In addition, simple templates can contain disjunctions of the above types of information. For illustration, the templates in (7) encode various information about predicate complements in Finnish. Spelled out as a graph, (7c) takes the the form of (8):

```
(7)   a. (ablativepredcomp predcompagr ((predcomp case) abl))

      b. (unmarkedpredcomp predcompagr
           (!or (@ ((subj num) sg)
                   ((predcomp case) nom)
                (@ ((subj num) pl)
                   ((predcomp case) ptv)))

      c. (predcompagr ((subj num) (predcomp num))

(8)   a. [SUBJ:[NUM:SG]
          PREDCOMP:[NUM:SG
                    CASE:NOM]]
```

[3] The template names are mnemonic for the linguist but otherwise arbitrary and subject to frequent changes.

```
     b. [SUBJ:[NUM:PL]
         PREDCOMP:[NUM:PL
                   CASE:PTV]]
```

The second type, parametric templates, allow attribute variables and, consequently, a more abstract way of formulating things. For example, there is a parametric template for each Finnish grammmatical case, where the function that is to receive this case is the parameter. (9a) below calls the parametric template PTV 'partitive', which is defined in (9b), in this case to be applied to the predicate complement, as the graph in (9c) shows.

```
(9)  a. (!use ptv predcomp)
     b. (ptv ((?x1 case) ptv))
     c. [PREDCOMP:[CASE:PTV]]
```

The number of variables is not restricted to one. When we need to equate an English and a Finnish graph, as in (10a), we use the generalized template TR1, defined in (10b). The simplest application is (10c), already represented in (2).

```
(10) a. ((e attr1) (f attr1))
     b. (tr1 ((e ?x1) (f ?x1)))
     c. (trtense (!use tr1 tense))
```

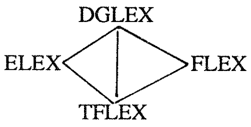
The template mechanism gives great freedom in choosing which pieces of information to put together. In the course of adding new types of linguistic information, existing templates are reformulated, and the information in them is frequently regrouped. The main thing is to build individual templates so that they can be referred to by other templates (cf. 7). Sets of templates thus form hiearchies on the basis of how they inherit information from each other.

Templates are essential in defining word classes, such as verbs with different argument structures. We expect the work with verb and adjective entries to consolidate the types that need to be used by the growing lexicon (section 6). This has already been experienced to some degree. A well-established set of such argument frame templates will be useful in devising automatic interactive lexicon building facilities for lexicographers or users. It should also be noted that template names can be reinterpreted as atomic features. These, in turn, can be given new interpretations in terms of some other implementation or linguistic theory. Our lexicons could thus be used by other applications, including non-unificational ones.

We are now ready to consider the organization and structure of the lexicons themselves.

5. The transfer lexicons

The information needed in transfer resides in four lexicon modules:



A transfer dictionary, TFLEX, states the lexical and grammatical correspondences between lexical items, features, etc. TFLEX refers to the two bilingual lexicons: An English dictionary, ELEX, describes the relevant syntactic, morphological, and semantic properties of the lexical entries in a purely English-specific way [4]. A Finnish dictionary, FLEX, does the same for Finnish. ELEX, FLEX, and TFLEX each rely on a module, DGLEX, which does not contain lexical items but, in template form, definitions common to both languages. With the exception of DGLEX, each lexicon is divided into a "lexicon" and a "templates" section.

From the point of view of the transfer algorithm, TFLEX acts as a passage to the other lexicons. This is illustrated by the schematized example of a TFLEX entry in (11). Information in ELEX, FLEX or DGLEX is prefixed with *e::*, *f::* and *dg::*, respectively. Each disjunction ("!or" clause) has three parts: a specification of the relevant English reading(s); a specification of the relevant Finnish reading(s); and a transfer template proper to say which attributes of the two are to be equated.

```
(11) (start (!or ((e (e::start dg::n)
                  (f (f::alku))
                  trn)
                 ((e (e::start dg::v e::simpleobj)
                  (f (f::aloittaa))
                  tra)
                 ((e (e::start dg::v e::noarg2))
                  (f (f::alkaa dg::novcomp))
                  tra)))
```

The noun *start* in (11) has one translation, the verb two translations depending on transitivity. It is important to note that the English and Finnish entries are not defined in TFLEX, but in their respective monolingual lexicons. The role of the English, Finnish and DGLEX templates in the TFLEX entry is to filter out, for each pair of words, the set of readings of the word that don't come into question. The relevant monolingual entries would then look like (12) for ELEX and (13) for FLEX.

[4] In our application, ELEX augments the often scarce lexical information provided by PEG. In another, imaginable application, ELEX would be the data base of the English parser.

```
(12)   (start
        (!or (n abstr)
             (v (!or simpleintran simpleobj simpletovcomp))))

(13)   (alku n abstr)
       (alkaa v (!or simpleintran simpleinfl))
       (aloittaa v simpleobj)
```

Multiword entries are treated on a par with simple entries in our system. If an entry can make reference to, say, the semantic features of a verb's object and those of the object's determiner, it is equally easy to refer to their LEX attributes. The modular organization of the lexicons induces a distinction between multiword entries. The monolingual lexicons must define idioms proper, such as *keep tabs*. But it is questionable whether the expression *have access to* is an idiom in English, although it corresponds to one word (*päästä*) or an idiom (*päästä käsiksi*) in Finnish. Such "transfer idioms" thus appear only in the transfer lexicon. A simplified part of the entry for *have* is given in (14):

```
(14)    (have (e (e::have simpleobj ((obj e lex) e::access))
              (f (f::päästä)))
```

6. Grammatical organization: Arguments and grammatical functions

We represent grammatical content by dependency graphs, influenced by Lexical-Functional Grammar (LFG; Bresnan 1982) and traditional Finnish grammar. Unlike in LFG, constituent trees do not figure in our system at all. A major difference from the LFG framework is that our graphs are not intended to be representations of pure functional structure; categorial and ordering information is freely included. To mention one further difference, we have not yet found any use for thematic roles, which are popular in current LFG.

One of the reasons for using a dependency organization rather than phrase structure is the widely accepted conclusion that the former is less language-particular than ordered constituent structure. In fact, the exact nature of Finnish constituent structure is unclear. The same can be said of the use of grammatical functions (GFs), which are the main labels in our graphs. The set of GFs used this far is the following:

- SUBJ(ect), OBJ(ect), OBL(ique), SCOMP (sentential complement), VCOMP (infinitival complement), PREDCOMP (predicate complement)
- (Finnish) GENITIVE,
- (English) OBJ2, OBL-BY, OBL-OF
- ADJUNCT, HEADW

We feel free to add the number of GFs, particularly different OBL and adjunct types.

In addition to GFs, we use an additional, more abstract level, argument structure. Arguments are linked to GFs by rules that partially define verb-argument frames. Arguments remain constant under alternations such as passive and dative shift, although their GF linkings differ. An important difference beteween English and Finnish is that argument/GF linkings are extremely constant in the latter.

As to the distinction between arguments and adjuncts, we don't want, at least not at this stage, to be too particular. (See Pajunen 1988 for the difficulties involved.) Our argument frames of particular verbs may be more inclusive than others', especially when it comes to inclusion of participants typical in the kinds of text our system is intended to be applied to. The following rules are followed in argument/GF linking.

- Arg1 is linked to SUBJ if there is a subject; impersonal VCOMP and SCOMP constructions have the complement as Arg1.
- Arg2 is linked to OBJ if there is one (in English, SUBJ of passive, OBJ2 of ditransitives), otherwise to OBL, VCOMP, or SCOMP;
- Arg3 is linked to OBL (In English, OBJ of ditransitives), or obj-controlled VCOMP, if Arg2 is already occupied.

In English, we distinguish two kinds of OBL. In one, shared with Finnish, OBL itself is linked to the argument in question. In the other, the argument corresponds to the object of the PP that has the OBL function. Examples of the two groups are *put* (locative) and *consist of* (non-locative).

The assumption is that genuinely locative verbs take locative arguments, which can be realized by PPs, but also by appropriate adverbs. In such cases, the choice of the preposition is more open, depending on the nature of the PP-object in part. The verbs that take the non-locative arrangement will select one preposition, or perhaps a couple. Assuming that selectional restrictions are ultimately stated on arguments, as they must be in order to stay constant under GF alternations, the non-locative arrangement implies that the relevant verbs directly know about the semantic status of their PP objects.

Argument structure is of great importance in TFLEX. Entries are simplified when transfer relations are stated between the arguments, rather than between the GFs linked to them. This allows for a simple and general formulation of predicate transfer, Tra or "translate arguments". This was used in (11).

For example, since the argument/GF linkings remain separate, Tra in the case of a simple transitive verb pair such as *delete/poistaa*, automatically pairs English SUBJ and Finnish SUBJ in active, but English SUBJ and Finnish OBJ in passive. Tra also takes care of translation equivalents like *like/pitää* and *discuss/keskustella*, whose second argument is an object in English but an OBL in Finnish (see 21 below). However, we can't always resort to simple Tra. A simple example is the verb *point* in the following context:

```
(15)    Point the cursor at the left window
        Osoita kursorilla vasenta ikkunaa.
        Point cursor-ADE left-PTV window-PTV
```

By the above rules of thumb, *point* has *cursor* as Arg2 and *window* as Arg3, whereas Finnish *osoittaa* treats 'window' as Arg2 (OBJ) and 'cursor' as Arg3, as shown by its case form which is the one typically encoding instruments. Our TFLEX entry for *point* on this reading must therefore contain a more detailed transfer instruction that equates Arg2 with Arg3 and vice versa.

There are roughly equivalent verbs whose argument and GF structure resemble that of *point* more closely, in particular, *suunnata*. However, it is *osoittaa* that gives the natural everyday translation in this case. We want ELEX and FLEX to be simple, natural, and linguistically motivated; TFLEX must at times give ad hoc, messy descriptions. There is no reason to assume that all transfer relations should obey linguistic generalizations or linguistic universals. This raises the question of the place of semantics in our kind of transfer system, to which we shall return at the end of the following section.

In addition to arguments proper, we have played with an "extra argument" (inspired by Pajunen 1988). This would only be linked to the function OBL2, would never be obligatory (i.e., never have an *ANY* value), and could be used to encode typical but not argument-like participants like *about* phrases of communication verbs, instrumentals of action verbs, or experiencers of attitudinal adjectives (*kind to me, ystävällinen minulle*).

7. Adjuncts and cyclic graphs

Arguments and GFs are unique for each head, and statements that refer to them are inherently simple in unificational grammar. But no amount of liberality in the representation of arguments would let us get rid of the phenomenon of multiple adjuncts[5]. Although the examples in this paper pretend that there is a single adjunct for each node, we actually represent sets of adjuncts as lists.

The treatment of adjuncts in our framework introduces cyclicity in the graphs. While predicates point to their unique arguments, adjuncts point to their respective heads (modified words), which are their unique arguments. We use the term HEADW for the "modified" function (cf. traditional Finnish "head word"). Such a cyclic graph is given in (16) (next page). Cyclic graphs could also have GFs point to their heads, making the representation closer to that of traditional Finnish grammar.

Adjectives act both as arguments (PREDCOMPs) and adjuncts. In lexical transfer, it seems important that they be able to refer to the semantics of either their subject (controller), or the head noun.

[5] The received wisdom is that grammatical functions are unique whereas adjuncts allow multiple occurrences. It seems to me entirely plausible that each adjunct type would be unique as well, if only we could establish an adjunct type classification revealing and fine-grained enough. I find it hard to imagine that one and the same verb be modified by two instrumentals or two genuine manner adverbials. The blatant exception, multiple locatives, is explained by their capability of forming "inclusive" relations, rather than the fact that they are not arguments. A phrase like *to sit on a bench under a tree* can thus be represented as containing only one locative argument or adjunct, with the ability of multiplying itself in a semantically coherent way, each location being included the next one (*to sit in a room in the park* would force us to conclude that the room was in the park). Of course, the technical problem of multiple locatives and of multiple adjective modifiers of nouns still remains.

```
(16) #1[E:[LEX:EXAMPLE
         CAT:NOUN
         ADJT:#2[E:[LEX:ADDITIONAL
                    CAT:ADJ
                    PRED:[ARG1:#1
                          ARG2:*NONE*
                          ARG3:*NONE*]
                    ADJT:[E:[CAT:ADV
                             MODIF:#2]]
                    MODIF:#1]
                 F:[LEX:LISA
                    CAT:NOUN
                    ADJT:[F:*NONE*]
                    MODIF:#1]]
                    NUM:PL
                    PERS:3]
      F:[LEX:ESIMERKKI
         CAT:NOUN
         NUM:PL
         PERS:3
         PRENOUN:#2]]
```

(Predicative and modifier adjectives also show agreement with these in Finnish, but follow different rules in that respect). This is represented as follows.

Each adjective has (at least) Arg1 and forms (at least) two graphs. In one of them, Arg1 is linked to the subject and the adjective is stated to be predicative; in the other, Arg1 is linked to the modified, whose category must be noun. Transfer rules for lexical choice can then refer to particular semantic or grammatical features in Arg1. This is needed in, e.g., choosing from *kova* and *vaikea* as equivalents of *hard*, where the matter is - roughly - resolved by concreteness vs. abstractness of the first argument.

As a further illustration of our approach, let us consider a more complex transfer relation that has to do with adjuncts. English-to-Finnish translation shares a feature often mentioned when considering translation from English to other European languages. The equivalents of the verb *like* do not accept a VCOMP, so that sentences of the type *I like to work* are often best translated with 'I work with pleasure', where, in effect, an adjunct replaces the whole upper-level predicate[6].

```
(17)    I like to work.
        Minä teen mielelläni työtä.
        I do with-pleasure work-PTV
```

We account for this relation as follows. An English predicate often translates simply as the equivalent of its VCOMP (or some other argument; see (4) above), with an addition of some attribute(s). This is the case with, e.g., passive and

[6] Here, we could nominalize the VCOMP and obtain a legitimate object: *Pidän työn tekemisestä.* Still, the adverbial alternative can't be omitted, for some nominalizations would give awkward, some downright ungrammatical results.

progressive *be* and *will* (Finnish has no expression for future in the unmarked situation). In addition to this, the transfer entry must then state that the Finnish equivalent of the VCOMP should contain the adjunct *mielelellään.* This is illustrated in (18). Examples similar to *like/mielellään* come up with *can* and *may/might*, often translated by adding *ehkä* 'maybe' to the equivalent of the VCOMP.

```
(18)    #6[E:[LEX:LIKE
              CAT:VERB
              SUBJ:#5[F:[NUM:#3
                         PERS:#2]]
              VCOMP:#10[E:[LEX:*ANY*
                           CAT:VERB
                           SUBJ:#5
                           VFORM:INF]
                         F:#4[SUBJ:#5
                              ADJUNCT:#1[F:[LEX:MIELELLAAN
                                             CAT:ADV
                                             NUM:#3
                                             PERS:#2]]]]
              PRED:[ARG1:#5
                    ARG2:#10
                    ARG3:*NONE*]
              VOICE:ACT]
           F:#4]
```

But isn't there a generalization being missed here? What is common to *mielellään* and *like to* is that they are predicates or functors that take another predication as their argument, as represented in Rupp (1989) and Kaplan & al. (1989). An alternative to our representation, then, is a more semantic representation, which makes the two languages more isomorphic, i.e., makes the "translate argument" template work. This can be sketched as in (19), where the *may/ehkä* pair is used for simplicity:

```
(19)    [E:[LEX:CAN
             PRED:[ARG1:#1[E[]
                           [F[]]]]
        [F:LEX:EHKA
           PRED:[ARG1:#1]]
```

As noted in Carlson and Vilkuna (1990) and Carlson (this volume), unificational transfer of the present type is flexible also in the sense that it is able to accommodate different levels of description simultaneously. In the approach of Kaplan & al. (1989), functional and semantic levels are represented as simultaneous but distinct projections of the same structure. In our graphs, semantic relations can be separately encoded in a particular attribute. Thus far, we have chosen a straightforwardly structural method for handling problems like *like to*, but a more "deep" one is not excluded in principle.

9. Thema

Attribute-value graphs, unlike phrase structure trees, abstract away from linear order. A potential advantage of phrase structure transformations as a transfer method might be seen in the possibility of preserving order where possible. For example, definite passive subjects in English correspond to various GFs in Finnish, but the typical translation equivalent of a passive sentence keeps the equivalent of the subject just where it is in English, i.e., in front of the finite verb:

```
(20) a. It was discussed.
     b. Siitä keskusteltiin.
        it-ELA discussed-PASS
```

Still, we would argue, it would be questionable to say that the Finnish OBL has the "same" position as the English SUBJ. Instead, we preserve this partly discourse-conditioned ordering fact in more abstract terms. In the transfer rule for passives, the English subject is said to correspond to the Finnish discourse-based function THEMA ("T" in Vilkuna 1989). Depending on the frame of the Finnish verb, then, THEMA may also have OBJ, OBL, or some other function in Finnish. The linearization rules then place THEMA in front of the verb. A bilingual graph illustrating (20) is given in (21).

```
(21)    [E:[LEX:BE
            CAT:VERB
            SUBJ:#3[E:[LEX:IT
                       CAT:PRON]]
                   [F:[LEX:SE
                       CAT:PRON
                       CASE:ELA]]
            VCOMP:#9[E:[LEX:DISCUSS
                        CAT:VERB
                        VFORM:PASTPART]]
                    [F:#10[LEX:KESKUSTELLA
                           CAT:VERB
                           THEMA:#3
                           SUBJ:#5[F:[LEX:*NONE*
                                      SEM:[HUM:T]]]
                           OBL:#3
                           TENSE:PAST
                           VOICE:PASS]]
            TENSE:PAST
            VOICE:PASS]
        [F:#10]]
```

The THEMA function is also used in Finnish verbs whose unmarkedly preverbal argument is not the grammatical subject, such as those in (22). The transfer rule that equates English SUBJ with Finnish THEMA thus has a wide application.

```
(22) a. Minulla on tietokone.
        I-ADE is computer
        'I have a computer'

     b. Siitä tuli hyvä.
        it-ELA came good
        'It became good'
```

10. Summary

This paper has reported a transfer based approach to machine translation that carefully separates declarative linguistic specification from its use by the transfer algorithms. All linguistic information, monolingual descriptions as well as specifications of transfer relations between items, are expressed as attribute-value graphs. The information is encoded in separate monolingual and transfer lexicons. A specification language with templates allows flexible statement of generalizations. No formal distinction is made between lexical and structural transfer.

The linguistic model used is a LFG-influenced dependency description, where grammatical functions and argument-function linkings play a central role. Transfer of larger constructs is achieved by equating arguments and adjuncts of each node according to the rules in the transfer lexicon. The paper illustrates the central types of such rules.

Acknowlegements

The basic design of our system is based on the ideas of Kimmo Koskenniemi and Lauri Carlson. Lauri Carlson implemented the first version of the system and directs the development work. Kimmo Koskenniemi has provided the morphological generation tools. Lately, Krister Linden has been responsible for the design and implementation of new versions. The present report would not have been possible without this team.

References

Bresnan, Joan (ed.) 1982: *The Mental Representation of Grammatical Relations*. Cambridge, Mass.: The MIT Press.

Carlson, Lauri 1988. Unifikaatiokielioppi formalismina. *SKY 1988*: 79 - 90. Helsinki, The Linguistic Association of Finland.

Carlson, Lauri and Krister Lindén 1987: Unification as a Grammatical Tool. *Nordic Journal of Linguistics* 10: 111 - 136.

Carlson, Lauri and Maria Vilkuna 1990: Independent Transfer using Graph Unification. Paper to be read at Coling '90.

Jensen, Karen 1986: PEG 1986: A Broad-coverage Computational Syntax of English. Technical Report, IBM T.J. Watson Research Center.

Kaplan, R. - Netter, K. - Wedekind, J. - Zaenen, A. 1989: Translation by Structural Correspondences. Proceedings of the Fourth Conference of the European Chapter of ACL, Manchester.

Pajunen, Anneli 1988: *Verbien leksikaalinen kuvaus*. Publications of the Department of General Linguistics 18, University of Helsinki.

Rupp, C.J. 1989: Situation Semantics and Machine Translation. Proceedings of the Fourth Conference of the European Chapter of ACL, Manchester.

Shieber, Stuart M. 1986: *An Introduction to Unification-based Approaches to Grammar*. CSLI Lecture Notes 4. Stanford: CSLI.

Vilkuna, Maria 1989: *Free Word Order in Finnish*. Helsinki: Suomalaisen kirjallisuuden seura.