[*Proceedings of a Workshop on Machine Translation*, July 1990, UMIST]

Generation for MT in English in Eurotra

Ian Crookston
University of Essex
Colchester, UK

SALT Club MT Workshop,
UMIST, 2-3 June 1990

1 The Synthesis Task and Formalisms

In multilingual MT, the "interface" representation that is the input to and output from transfer is heavily constrained. For modularity, it must not be "tuned" to specific target languages, as argued in Krauwer & des Tombe (1984), but must be a representation of the language concerned per se. However, to maximise the simplicity of the bilingual components of the system, the representation of any given sentence must be maximally similar to the representation of translationally equivalent sentences in other languages, as argued in van Eynde (1986); Arnold, des Tombe & Jaspaert (1985, section 2.2.3); Leermakers & Rous (1986).

These two apparently conflicting requirements can in fact be used like two compass bearings from different points to "fix" the best possible interface representation theory. The Eurotra Reference Manual is mostly taken up with such an enterprise, and reports on this will appear in Allegranza et al (forthcoming). The theory adopted in Eurotra is a predicate-argument type of theory, a fact which even without further elaboration has rather definite implications for aspects of the synthesis task, as will be seen below.

Such a fix obviously also fixes the input to synthesis. The task of synthesis comes to be definable as the task of mapping from the representation defined by the interface theory to a plausible surface tree covering a correct surface string.

There are certain further givens specific to Eurotra synthesis components. The interface representation has always taken the form of a tree, rather than, say, a dag. Also, the translation process is performed sequentially, rather than being embodied in an asequential set of constraints as in some recent work such

as Kaplan et al (1989). This latter given leads to synthesis (like analysis and transfer) being isolable as an independent module of the system.

With all these givens, the most basic imaginable tool for synthesis is probably the so-called "t-rule" of the CAT formalism of Arnold et al (1986). In the most compact version of this notation, that of Sharp (1988), the t-rules consist of little more than the annotated subtree on each end of the mapping. For instance, in a synthesis component, there might be a t-rule like

(1)

```
(?,{cat=s}).[ $GOV: (gov,{cat=v}),
              $ARG1: arg1,
              $ARG2: arg2 ]
=>
(s).[ $ARG1,
      vp.[ $GOV,
           $ARG2 ]]
```

This had the attractiveness of explicitness and clarity, but when it was applied over a wider range of phenomena, two problems emerged. Firstly, the right-hand side (RHS) of the t-rules repeated the target grammar. In (1), the RHS repeats the statement of the surface grammar that the verb and object are dominated by a VP. Secondly, the set of t-rules exploded combinatorially. How this emerged depended on details of the grammars involved. For example, in the above case, if a second rule were added for passive sentences, then a third and fourth would be needed inserting *will* on the RHS in future active and passive sentences. It is impossible to make separate provision for passive and future—there is a passive simple-tense rule and a passive future rule. To add provision to lower the negation operator into its surface adverbial position now requires not another rule but another four rules.

The current Eurotra formalism, the E-Framework described in Bech & Nygaard (1988), Raw et al (1988), is a reaction to these difficulties. The general route of the EFW to solving these problems is to separate the output of the t-grammar from the finished representation in the following way, as described in Bech & Nygaard (1988):

(2) Representation —t-grammar→ DESCRIPTOR—grammar→ Representation

The RHS of each t-rule specifies a local part of a special representation called a descriptor. This descriptor is then further processed by the grammar to produce

a true representation. Nodes such as *will* can be added in this processing, and an unordered descriptor can be specified (using round brackets). So (1) becomes

(3)

```
{cat=s,voice=active}[ GOV:  {role=gov,cat=v}
                      ARG1:  {role=arg1}
                      ARG2:  {role=arg2} ]
=>
{voice=active}< ( GOV, ARG1, ARG2 ) >
```

and (3) can replace the four t-rules covering future *will* and voice alternation mentioned above.

To make such a strategy work a number of more powerful devices than are available in CAT are needed within the grammar of a level of representation, in addition to the unorderedness device in t-rules just mentioned. Among these devices are

(4)
(a) parsing of descriptors
(b) insertion of leaves
(c) downward expansion of leaves
(d) the Eurotra Coindexation Tools (Allegranza & Bech (1989))

These devices have been exploited in the English synthesis component to produce a component virtually free of t-rules.

2 Synthesis Tasks

The fixed points of interface and surface representations, and the types of difference which must of necessity exist between them, define a number of tasks which must be performed by a synthesis component. Among these tasks are the creation of surface word order, of a VP node, and others which will be discussed in detail directly. To aid in these tasks, the English synthesis component features a third level of representation called Eurotra Relational Structure (ERS) as a stepping stone between the fixed points mentioned above, which are usually referred to as Interface Structure (IS) and Eurotra Constituent Structure (ECS).

2.1 Surface word order

IS and ERS have the same, conventional, order of constituents. In the case of IS this is necessary as a neutralisation of the differing surface word orders of the Eurotra languages. Between ERS and ECS this language-neutral and conventional order must therefore be turned into the observed word order of English.

The order is first discarded by what is almost the only t-rule in the module, which looks like this:

(5) `M:{}[ D:+{} ]  => M < ( D )  >`

That is, "from any local tree at ERS, make a descriptor with the translation of the mother as mother and the translation of the daughters as daughters, the latter being unordered". Almost all local trees are translated to ECS by this rule.

The ECS grammar then has to be written in such a way as to accept as few as possible of the many possible orderings which (5) defines. The ECS rules thus have to be written in a relatively fine-grained way (something which would arguably be undesirable in an analysis grammar), for example the NP rule:

(6)

```
np = {cat=np,ncase=nongen,nb=N,head=H}[
             * {cat=advp,sf=mod,aptype=noncompl}
             ^ ( {cat=np,ncase=gen} ; {cat=detp} )
             ^ {cat=ordp}
             ^ {cat=cardp}
             * {cat=ap,aptype=noncompl ,pos=i}
             * {cat=ap,aptype=noncompl,pos=ii}
             * {cat=s,wh=no,compval=nonq,
                aptype=noncompl,mstype=pastpart}
             * {cat=s,wh=no,compval=nonq,
                aptype=noncompl,mstype=prespart}
             * {cat=ap,aptype=noncompl,pos=iii}
             * {cat=np,ncase=nongen,nb=sing,role=attrib}
               {cat=n,nb=N,gb_lu=H}
             ^ {cat=pp,role=arg1}
             ^ {cat=pp,role=arg2}
             ^ {cat=pp,role=arg3}
             *  {cat=pp,role=mod}
             *  {role=mod,cat=np,barenpadv=yes,ncase=nongen}
             *  {cat=ap,aptype=compl}
```

```
* {cat=s,aptype=compl,mstype=pastpart}
* {cat=s,aptype=compl,mstype=prespart}
* {cat=s,mstype=infin}
* {cat=s,mstype=finite} ] .
```

Each daughter is fairly heavily subtyped: for example, reduced relatives with nothing after the verbal head ("cat=s,aptype=noncompl") are accepted at a different point from those with something after it. The sentence rules are similarly subtyped, making great use of syntactic function annotations which are added by the ERS grammar.

2.2 Strongly governed prepositions

A strongly governed preposition such as the *on* of *rely on* is not a predicate (or a likely translational unit) and cannot feature in IS as a word. It therefore must come into existence at some point in synthesis.

The IS representation, which is identical to the ERS descriptor, is

(7)

```
{cat=s}[ {role=gov,word=rely,pformarg2=on>
       …
       {role=arg2,cat=np,pform=on} ]
```

The sentence node will also be decorated with a feature signifying its voice. This descriptor then has to be processed by the following ERS cf rules:

(8)(a)

```
{cat=8,voice=active} [ {role=gov,sf=gov,pformarg2=P}
                         …
                      {role=arg2,sf=obl,cat=pp,head=P} ]
```

(b)

```
{cat=pp,role=R,sf=obl,head=P}[!  {cat=p,word=P}
                                 {cat=np,role=R,
                                  pform=P} ]
```

(c)

```
{cat=s,voice=passive}[  {role=gov,sf=gov,pformarg2=P}
                         ….
                        {role=arg2, cat=np, sf=subj}
                    !   {cat=pp,sf=obl,head=P} ]
```

(8)(b) makes use of the insertion device (4)(b) in the obvious way. The node marked "!" is added to the tree. If active, the descriptor is parsed as mentioned in (4)(a), which means that a PP node is built above the arg2 by (8)(b), and this arg2 is then acceptable to (8)(a). If the descriptor is passive, (8)(b) will not fire. The arg2 is accepted by (8)(c) directly, which then inserts an extra PP daughter with the head on. The sf markings subj and obl are used in the reordering process described in the preceding section to order the constituents appropriately: the result will be surface strings like *The system was relied on*, with a stranded preposition.

2.3 Aspect and Voice

Aspect and voice are represented as feature decorations on the IS tree. This is so that they can be translated orthogonally to the main lexical and relational content of the sentence. For example:

(9) `{voice=passive,aspect=progressive}`

These annotations must be mapped into sequences of surface auxiliaries, and for a compact and perspicuous treatment each pairing of an annotation and an auxiliary must somehow be expressed individually. If t-rules were used for this kind of mapping, each possible combination of annotations would have to be treated separately, since these devices do not permit a mapping between an annotation and a piece of tree structure.

An attractive option is to use the "insert and parse" technique illustrated with (8)(b) and (8)(a) above. This is done in the ECS grammar. First a VP node is added to the tree above the verb and its complements, and then a short cascade of rules of the following type is available:

(10)(a)

```
{cat=vp,voice=passive}[! {word=be}
                         {cat=vp,mstype=pastpart,
                          voice=none} ]
```

(b)

```
{cat=vp,voice=V,aspect=progressive}[ !  {word=be}
                                        {cat=vp,voice=V,
                                         mstype=prespart,
                                         aspect=none} ]
```

The main VP is accepted as the VP daughter of (10)(a), and the VP thus produced is accepted as the VP daughter of (10)(b). Only with the application of (10)(b) is a VP acceptable to the sentence rule produced.

Another option, much more suited to the bottom-up parsing techniques of the present implementation of the E-Framework, is to create the auxiliaries at a different stage from the creation of the VP nodes over them. This is what is done in the actual English synthesis component. The IS verb is decorated with aspect and voice features, and is turned into a "verb group" in the ERS descriptor:

(11)
IS representation:
`{cat=v,voice=passive,aspect=progressive}`
ERS descriptor:
`{cat=vgrp,voice=passive,aspect=progressive}`

This verb group node is then expanded downwards (the device mentioned in (4)(c)), recursively, by a short cascade of such rules as

(12)(a)

```
{cat=vgrp,voice=V,aspect=progressive} [{word=be}
                                       {cat=vgrp, voice=V,
                                        mstype=prespart,
                                        aspect=none> ]
```

(b)

```
{cat=vgrp,voice=passive,aspect=none}[ {word=be}
                                      {cat=v,
                                       mstype=pastpart} ]
```

so that the ERS representation is

(13)

```
{cat=vgrp,voice=V,
 aspect=progressive}[ {word=be}
                     {cat=vgrp,
                      voice=passive,
                      mstype=prespart,
                      aspect=none}[ {word=be}
                                   {cat=v,
                                    mstype=pastpart,
                                    voice=none} ]]
```

This procedure observes the requirement of pairing annotations and auxiliaries individually (there is in principle one rule in (12) for each annotation) while avoiding the computational drawbacks associated with a bottom-up parser operating on rules such as (10).

2.4 Raising

The most awkward difference between a predicate-argument representation and a surface one will be that which can be termed the raising phenomenon. This extends far beyond the narrower theoretical definitions of the phenomenon: the difficulty potentially exists in many places where a predicate does not selectionally restrict its surface subject. Thus the synthesis component is faced with the probability that it will have to undo long (in principle infinite) chains of raising movements: a mapping such as (14) is not unlikely.

(14)
may(happen(be-going(seem(fail(they))))) =>
They may happen to be going to seem to fail

The E-Framework's facility for treating this is the Eurotra Coindexation Tools mentioned in (4)(d). The English synthesis component contains a special rule something like

(15)

```
raising = {cat=s}[ {cat=vgrp}
                   {att<=>I,cat=np,sf=subj}
                #1 {cat=s}[ {raising=yes}
                          #2 {cat=s}[ +{}
                                      {cat=np, sf=subj,att=I}
                                     *{}],
                             *{} ],
                  *{} ].
```

This contains the "recursion markers" #1 and #2, which means that it fires when it encounters infinite repetitions of the subtree spanned by them, in this case an infinite cascade of sentence nodes dominating raising predicates. It also contains the "copy operator" <=>, which is an instruction to copy the appropriately-marked subtree (here the subject of the bottom sentence of the cascade, identified by the variable I matching the copy operator's variable) onto its site.

Thus

(16) may(happen(be-going(seem(fail(they)))))

at IS is converted into

(17) may(they,happen(be-going(seem(fail(they)))))

at ERS. Between ERS and ECS the lower copy of the raised subject is deleted, producing the desired output with the normal reordering and other spelling-out processes.

3 Conclusion

The E-Framework does make it possible to implement MT synthesis for a substantial fragment of English without the use of a combinatorially-exploded set of t-rules as would be necessary in CAT. The most obvious outstanding "empirical" problem is idioms: though some common types, such as verb-object idioms, can be treated satisfactorily, a general solution is unavailable where the working notion is the whole local tree (cf Arnold &: Sadler (1987)). The effect of the parsing device (4)(a), which inserts a VP in an ECS sentence descriptor, on a coordinate structure is another problem (cf Crookston (to appear)).

References

Allegranza, V, &: A Bech (1989) "A Versatile Tool for Treating Unbounded Dependency Constructions in NLP and MT Systems", Gruppo Dima Working Papers 1, Gruppo Dima, Turin

Allegranza, V, S Krauwer & E Steiner (forthcoming) special issue of *Machine Translation* on EUROTRA

Arnold, D, L des Tombe & L Jaspaert (1985) "Eurotra Linguistic Specifications Version 3", DG XIII, CEC, Luxembourg

Arnold, D, S Krauwer, M Rosner, L des Tombe & G B Varile (1986) "The <C,A>,T Framework in EUROTRA: A Theoretically Committed Notation for MT", in *Proceedings of the 11th International Conference on Computational Linguistics (COLING 86),* Association for Computational Linguistics, 297-303

Arnold, D, & L Sadler (1987), "(Non)-Compositionality and Translation", in *Recent Developments and Applications of Natural Language Understanding,* Unicom Seminars Ltd, Uxbridge, 44-67

Bech, A, & A Nygaard (1988) "The E-Framework: A Formalism for Natural Language Processing", in *Proceedings of the 12th International Conference on Computational Linguistics (COLING 88),* Association for Computational Linguistics, 36-39

Crookston, I (to appear) "The E-Framework: Emerging Problems", to appear in *Proceedings of the 13th International Conference on Computational Linguistics (COLING 90),* Association for Computational Linguistics

Krauwer, S, & L des Tombe (1984) "Transfer in a Multilingual MT System", in *Proceedings of the 10th International Conference on Computational Linguistics (COLING 84),* Association for Computational Linguistics, 464-467

Leermakers, R, & J Rous (1986) "The Translation Method of ROSETTA", in *Computers and Translation 1, 169-183*

Raw, A, B Vandecapelle, & F van Eynde (1988) "Eurotra: An Overview", in *Interface 3, 5-32*

Sharp, R (1988), "CAT-2—Implementing a Formalism for Multi-Lingual MT", in *Second International Conference on Theoretical and Methodological Issues in Machine Translation of Natural Languages,* Carnegie Mellon Univ, Pittsburgh

van Eynde (1986) "The interface structure level of representation", in *Multilingua 5, 145-146*