NIELS JÆGER

Text Treatment and Morphology in the Analysis of Danish within EUROTRA

1 General Overview

The EUROTRA TRANSLATION SYSTEM consists basically of a C-programme which operates a front-end programme package and a core PROLOG programme, the Engine.

The Engine combines with a series of generators and translators written by the linguist in a user language and compiled into PROLOG code. The generators and the translators contain linguistic information according to the linguistic specifications in the EUROTRA model.

The EUROTRA model consists of three main parts: analysis, transfer and synthesis. The main parts are subdivided into levels. Each level in the EUROTRA model has a grammar (i.e. a generator) and between two levels there is a translator.

The EMS, EUROTRA Morphological structure, is the first level in analysis. The first syntactic level follows immediately after EMS, it is called ECS, EUROTRA constituent structure.

A front-end programme package is being tested in EUROTRA-Denmark, from July to December 1989. It consists of a wordscanner written in C and an SGML parser. SGML is an abbreviation of "Standard Generalized Markup Language".

In the present experimental phase the document which is going to be translated has to be written in "VI" or "Qone", two editors available for UNIX installations. A "VI" document may be formatted by "Nroff". But in principle, any text editor could be used for SGML text entry, just as SGML could deliver an output to any text editor.

SGML provides methods for marking up documents. For instance you can mark up the logical structure of a text (i.e. chapters, sections, paragraphs etc.).

SGML also provides a search-and-replace mechanism. We can make the SGML parser search for layout information and replace it by a marker.

We can also define translations between input character code and output character code.

In the front-end module we intend to use the SGML parser twice:
First the special formatting codes of the word processors will be changed into SGML mark up codes and simultaneously the 8-bit national character sets are turned into a standardised 7-bit Eurotrian character set.
Next the SGML parser creates a translator.

SGML may take in a whole text for the first parsing. But the paragraphs of the text will be forwarded to the second application of the parser and later to the Engine in strict rotation. The paragraphs will be translated one by one and placed in their translated form in the right position in the output file in synthesis.

Between the two applications of the parser it is possible to edit the first SGML document manually.

The last programme in the front-end package is a wordscanner written in C. The wordscanner segments the wordforms on the basis of the lexicon rules of the morphological level EMS.

2 An Example

Please consider the following example which consists of a title and a sentence:

```
Signaler

En kombineret 4 og 6 GHz polarizer konverterer det modtagne
signal.
```

Written in "VI" and supplied with "Nroff" commands the example would look like this:

```
.NH
Signaler
.PP
En kombineret 4 og 6 GHz polarizer konverterer
.UL det
.UL modtagne
.UL signal.
```

`.NH` means: what follows is a title—`.PP` means: here comes a paragraph. `.UL` in front of a word indicates that the word is underlined.

This piece of text is sent to the SGML parser. The "Nroff" commands are changed into SGML markers. The text now looks like this:

```
<DOC   LEVEL="emsda"  SCANLEV="ems"   SURF="dummy"  SCAN=yes
WP="Nroff" NAME=text">
<TEXT>
<CH>
<TI NROFF= "PP">
En kombineret 4 og 6 GHz polarizer konverterer
<STYLE TYPE="UL">
det
<STYLE TYPE="RN">
<SYLE TYPE="UL">
modtagne
<STYLE TYPE="RN">
<STYLE TYPE="UL">
signal.
<STYLE TYPE="RN">
</P></CH></TEXT></DOC>
```

Next the text is forwarded to the second application of the SGML parser which produces a translator:

```
:trans: dummy=>emsda.

:b:

surface0 = ~:{} => {cat=s, io='0'}

<{thcat=wordform, rend=no,upper=first}
%%SCAN signaler
>.

surface1 = ~:{} => {cat=s, io='1'}
<
{thcat=wordform, rend=no, upper=first}
%%SCAN en
,
{string='&bk'},
{thcat=wordform, rend=no, upper=no}
%%SCAN kombineret
,
{string='&bk'},
{string=numerus, lex='4', rend=no},
{string='&bk'},
{thcat=wordform, rend=no, upper=no}
%%SCAN og
,
{string='&bk'},
{string=numerus, lex='6', rend=no},
{string='&bk'},
{thcat=wordform, rend=no, upper=first}
%%SCAN ghz
,
{string='&bk'},
{thcat=wordform, rend=no, upper=no}
%%SCAN polarizer
,
{string='&bk'},
{thcat=wordform, rend=no, upper=no}
%%SCAN konverterer
,
{string='&bk'},
{thcat=wordform, rend=under, upper=no}
%%SCAN det
,
{string='&bk},
{thcat=wordform, rend=under, upper=no}
%%SCAN modtagne
,
{string='&bk'},
{thcat=wordform, rend=under, upper=no}
%%SCAN signal
,
{string='&fs'}
>.
.
```

&bk stands for blank; **&fs** stands for full stop.

The translator will translate from an empty object (i.e. the empty pair of curly brackets on the left side of the arrow) into the sentences we gave as input (on the right side of the arrow). **{cat=s}** means a sentence. The sentence in its turn contains several wordforms. The lay out information is featurized. 'upper' stands for upper case. The value 'first' means that the first letter in a word is capitalised. 'rend' means rendition and 'rend=upper' means that a given word is underlined. Double percentage signs and the capitalised word 'SCAN' is a signal to the wordscanner that the following word should be segmented.

Now, this translator is sent to the wordscanner. The result will be similar to the translator shown above except for the wordforms which will have been split up into basic strings. The first part of the translator will suffice to illustrate this:

```
surface1 = ~:{} =>
<
{thcat=wordform, rend=no, upper=first}
```

```
<{string=en}>,
{string='&bk'},
{thcat=wordform, rend=no, upper=no}
<{string='kombiner'}, {string='et'};
 {string='kombiner'}, {string='e'}, {string='t'}>,
{string='&bk'},
{string=numerus, lex='4', rend=no},
...
>.
.
```

The substructure is indicated by '<' and '>'.

The wordforms are split up into basic strings which figure as values to the attribute 'string'. A useful device is the possibility of using alternation in the translator. If, for instance, we have three dictionary entries for three different verbal endings '{string=e, ...}', '{string=t, ...}', and '{string=et, ...}' we will get two possible segmentations of 'kombineret' as shown above. The EMS should of course only accept one of the possible segmentations. In case of an Arabic number we have a standard value to the string attribute: 'string=numerus'. This means that we only have to write a single lexicon rule for all possible integers at the morphological level. The actual number is given as value to the attribute 'lex'. Since numbers are the same in all the Eurotrian languages no tranfer rules will be needed.

The wordscanner compares the wordforms with the lexicon rules of the morphological module. A lexicon rule has depth=0—it consists of one feature bundle. Among other things we assign category, a lexeme value (assigned to the attribute 'lu') and a string value—the latter will unify with the string value in the translator rule. We have lexicon rules for inflectional roots (marked by the feature 'infl=root'), we have lexicon rules for invariants (marked by 'infl=full') and we have lexicon rules for inflectional suffixes (marked by 'infl=infl_end'). The approach to inflectional morphology we have adopted is in accordance with the so-called 'word and paradigm model'. Consequently, the lexemes are divided into inflectional classes. Lexemes which can be grouped as an inflectional class receive the same number as value for the attribute 'flex_type'. Examples of lexicon rules (abbreviated):

```
{cat=art, lu=en, infl=full, string=en, ...}.

{cat=v, lu=konvertere, infl=root, string=kombiner, term=xx0,
 ...}.

{cat=v, infl=infl_end, string=e}.
{cat=v, infl=infl_end, string=t}.
{cat=adj, infl=infl_end, string=et}.
{cat=n, infl=infl_end, flex_type=fx3, msdefs=msindef,
 nb=plu, string=er}.
```

```
{cat=n, lu=signal, infl=root, flex_type=fx3, string=signal,
 term=xx00000837, ...}.

{cat=card, lu=numerus, infl=full, string=numerus}.
```

These single feature bundles combine at the morphological level in general constructors. The constructors needed for building past participles (as e.g. 'kombineret') will look like this (slightly abbreviated):

```
b_stem_v = {infl=stem, cat=v, lu=L, vform=inf, ...}
           [ {infl=root, cat=v, lu=L, ...},
             {infl=infl_end, cat=v, string=e} ].

b_full_v = {infl=full, cat=v, vform=pastpart, lu=L, ...}
           [ {infl=stem, cat=v, lu=L, vform=inf, ...},
             {infl=infl_end, cat=v, string=t, ...} ].
```

The constructors needed for building indefinite forms of nouns (as e.g. 'signaler') will have the form:

```
b_stem_n = {infl=stem, cat=n, lu=L, nb=N, msdefs=msindef,
            ...}
           [ {infl=root, cat=n, lu=L, flex_type=F, ...},
             {infl=infl_end, cat=n, flex_type=F, nb=N,
              msdefs=msindef}.


b_full_n = {infl=full, cat=n, lu=L, nb=N, msdefs=M,
            case=nge, ...}
           [ {infl=stem, cat=n, lu=L, nb=N, msdefs=M, ...},
             {infl=full, cat=separator} ].
```

The identifier occurs before the equal-sign to the left. In the constructor the mother node appears above, the daughter nodes appear between sharp brackets below.

The inflectional root may become an inflectional stem by addition of an inflectional ending (e.g. 'e' in the constructor for a verbal stem) which in that case only becomes a final wordform when another inflectional ending is added (e.g. 't' in the constructor which turns an infinitive stem into a past participle above). The finished wordform is marked 'infl=full'.

Variables (in the form of capital letters) are used a) to percolate values from daugthers to the mother node (cf 'lu=L') and b) to ensure that values of a certain attribute are the same in the feature bundles of two daugthers (cf 'flex_type=F').

By means of the segmentation and the subsequent unification of the basic strings we turn a wordform into a grammatical word—e.g. we change 'signaler' into 'signal, plural, indefinite, neuter'.

The sentence rule accepts one or more wordforms:

```
b_sentence = {cat=s}
             [{infl=full}].
```

The final representation at EMS has the form of a tree with a sentence node at the top and a series of binary subtrees (one for each wordform) which are hanging on the same level beneath:

```
                            cat=s
        ...-------------------|----------------...
                 (full)                   (full)
           ---------|---------            lu=den
        (stem)            string=r        string=det
   -------|-------
   |              |
 (root)        string=e
lu=konvertere
string=konverter
```

Adjectives, verbs and nouns are marked for termhood at EMS. 'term=xx0' means that a given word is not a term. A value different from 'xx0' means that the word is a term. Multi-word terms are identified at EMS in the Danish implementation. As an example of a constructor which builds a multi-word term I will give the one for term number 'x000000003':

```
b_000000003 = {cat=n, lu=kombineret_N-og_N_GHz_polarizer,
               term=xx000000003, gd=G, nb=N, msdefs=M,
               case=C, infl=term}
              [ {infl=full, cat=pastpart, lu=kombinere},
                {infl=full, cat=card, lu=numerus},
                {infl=full, cat=coord, lu=og},
                {infl=full, cat=card, lu=numerus},
                {infl=full, cat=n, lu=GHz},
                {infl=full, cat=n, lu=polarizer,
                 gd=G, nb=N, msdefs=M, case=C} ].
```

'polarizer' is the most important word in this multi-word term. We therefore percolate all relevant grammatical information from the main word to the top node, we assign a number to the attribute 'term' and a value to the 'lu' attribute. The number is assigned in order to avoid translation rules between two languages. A given term has the same number in all nine Eurotrian languages. The lexical unit value is for the benefit of grammarians and lexicographers who develop and maintain the system.

In the translator between the morphological level, EMS, and the constituent level, ECS, all substructure beneath the very top nodes is deleted in general translator rules.

When these general translator rules have applied the input object to ECS will therefore consist of a series of grammatical words and multi-word units hanging beneath the sentence nodes:

```
{cat=s}
  {cat=n, lu=signal, ...}
{cat=s}
  {cat=art, lu=en, ...}
  {cat=n, lu=kombineret_N_og_N_GHz_Polarizer, ...}
  {cat=v, lu=konvertere, ...}
  {cat=art, lu=den, ...}
  {cat=n, lu=modtaget_signal, ...}
  {cat=separator, lu=stop, ...}
```

EUROTRA-DK,
Njalsgade 80,
DK-2300 Copenhagen S
Denmark