PARSING DANISH TEXT IN EUROTRA

Ole Togeby
University of Copenhagen
and
EUROTRA DK

Abstract.
The machine translation project Eurotra is described as a multi language modular translation system with 9 monolingual analysis modules, 72 bilingual transfer modules, and 9 monolingual synthesis modules. The analysis module for Danish is described as a 3 step parser with structure generation rules for immediate constituent structure, syntactic structure, and semantic structure, and translation rules between them. The topological grammatical description of Danish proposed by Paul Diderichsen, is shown to be usefull in building the parser for Danish, especially with respect to the interaction between empty slots and filled slot in the topological pattern. At last the special problem with parsing and disambiguation of sentences that allow many pp attachments patterns is mentioned and a solution is suggested.

Parsing Danish Text in EUROTRA
Ole Togeby
Proceedings of NODALIDA 1987, pages 35-57

Introduction

The Council of the European Communities decided in November 1982 to launch a research and development project aimed at the production of a pre-industrial prototype machine translation system of advanced design covering all the official languages in the Community. This project is called Eurotra, and it is a multilingual machine translation system covering 72 language pairs, each of the nine EEC languages being translated into all the other EEC languages. Eurotra is run on a collaborative basis by decentralized groups. In this article I will describe some of the problems we have had in the Danish language group working with translation to and from Danish. So what is reported here is the result partly of the 'linguistic legislation' common for all the language groups i Eurotra, partly of the work in the Danish language group from which many persons have participated in the discussions about how to build a parser of Danish.

The translation is performed in three stages using three independent modules: 1) a source language analysis module consisting of a source language monolingual dictionary and a parsing grammar yielding an interface structure which is language independent formal tree representation of the sentence, decorated with the lexical material from the source language text; 2) a transfer module using a bilingual dictionary by which the lexical items are translated into the target language, and using translation rules by which the interface structure is transferred into, in most cases, an identical target language interface representation; 3) a synthesis module consisting of a monolingual target language dictionary and a grammar, in many respects a mirror image of the grammar used in analysis of that language; this module generates the target language text from the transferred interface representation.

Because the whole translation system consists of 72 transfer modules, but only of 9 analysis modules and 9 synthesis modules, we try to make as much of the work in analysis as possible, yielding an interface respresentation which is the same for the translational equivalents of the source language and target language. The 'only' difference between the interface representations is the lexical material of the sentence being translated.

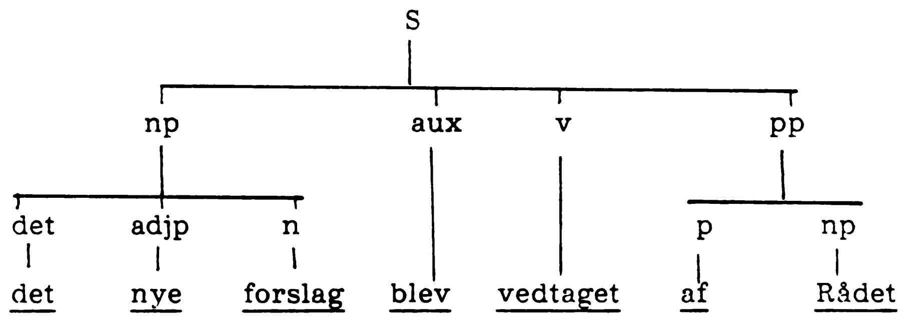
In this article I will describe the analysis module used by the Danish language group in Eurotra. The parsing of a sentence is done in 3 steps, primarily to provide modularity so that it is easy for all the linguists working in the project to recognize what is going on in the grammar rules, and so that errors can easily be found and corrected.

From the natural language text we parse to a level called Eurotra Constituent Structure, ECS, where the immediate constituents of the sentence are represented in a tree as np, auxiliary, v, advp and pp, and the immediate constituents of these sentence constituents are represented as daughter nodes with the names adjp, determiner, quantifier, cardinal and so on. From ECS we translate to a level called Eurotra Relational Structure, ERS, where the grammatical constituents of the sentence are represented in a tree with decorated nodes as subject, main verb, object, indirect object, attributive object, complement and modifier, and the constituents of these constituents are represented as modifiers and complements. From ERS we then translate into the Interface Structure, IS, where the dependency structure constituents of the sentence are represented in a tree in canonical order as: first: the predicate, i.e. the verbal head of the sentence, then: argument 1, 2 and 3 of the predicate, and finally sentence modifiers, and the dependents of the dependent constituents as arguments or modifiers of their heads.

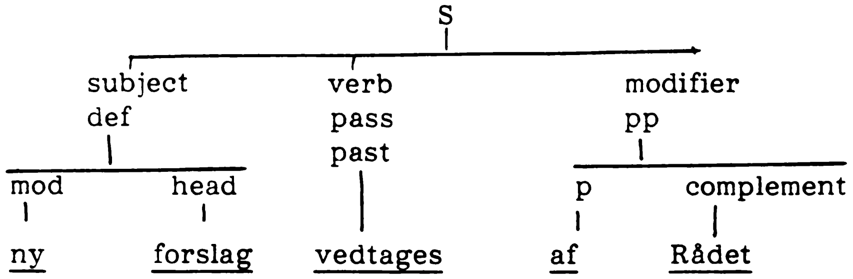
An example can illustrate the parsing process from text to IS:

text: Det nye forslag blev vedtaget af Rådet.

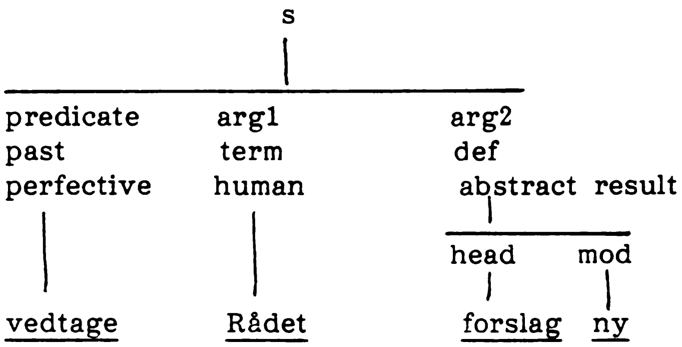
ECS:



ERS:



IS:



This parsing strategy means that we use three types of rules: 1) building rules, which are normal phrase structure rewriting rules. These rules generate the tree structure on each level. 2) Feature rules create the feature decorations on each node of the tree and exclude (kill) generated trees where the features do not match according to the feature match rules specified in the grammar. 3) Translation rules translate a decorated tree from one level into another decorated tree on the next level. In analysis the order of the levels is: text - ECS - ERS - IS, and in synthesis the order is: IS - ERS - ECS - text.

In the next paragraph I will describe some of the problems we have met making an ECS parser of Danish, using Paul Diderichsens topological grammar for Danish.

Overgeneration in a topological parser

It is not surprising that the parsing strategy will not be the same for case languages as Finnish or German and a non free word order language as Danish. A morphological parser has proved to be very efficient for languages with a rich morphology, but it is not at all sufficient for languages where much of the grammatical information is found in the word order. The alternative to a morphological parser is a topological parser, where the information found in the order of the words is transformed into the grammatical tree with canonical order of the decorated nodes.

But it is not clear how to write phrase structure rules generating a grammatical analysis, using the knowledge of the topology of Danish sentences, without overgeneration, i.e. without making many wrong analyses of a given sentence in addition to the wanted analyses.

As described by Paul Diderichsen in *Elementær Dansk Grammatik*, (Diderichsen, 1946) and elsewhere (Diderichsen, 1945) the order of the constituents in a Danish sentence is the following:

Base	actualisation field			content field		
	v^f	np	$advp^1$	v^{if}	np np	$advp^2$
så	ville	Petra	ikke	følge	børnene	hjem
then	would	Petra	not	follow	the children	home

And in subordinate clauses the order of the constituents is the following:

con- junction	actualisation field			content field		
	np	$advp^1$	v^f	v^{if}	np np	$advp^2$
hvis	Petra	ikke	ville	følge	børnene	hjem
if	Petra	not	would	follow	the children	home

The idea of this topological description is that this pattern is the order of the constituents in the sentence *if they are all present in the same sentence*; it is a maximally filled frame. If all the slots in the frame are not filled, the internal order of the constituents present in the sentence, will be the same:

Base	actual. field			content field				Heavy
	v^f	np^1	$advp^1$	v^{if}	np^2	np^3	$advp^2$	field
derfor	har	Rådet		vedtaget		planen		
derfor	vedtog	Rådet				planen		
Rådet	vedtog					planen		
i 1982	sendte	Rådet			Kommissionen	forslaget		

Literal, i.e. wordorder preserving translation of the sentences:
derfor har Rådet vedtaget planen
therefore has the Council passed the plan

derfor vedtog Rådet planen
therefore passed the Council the plan

Rådet vedtog planen
The Council passed the plan

i 1982 sendte Rådet Kommissionen forslaget
in 1982 sent the Council the Commission the proposal

The positions in this maximally filled scheme correspond systematically to the grammatical functions of the constituents:

In the actualization field the np^1 position after the v^f position is the slot for the subject and the $advp^1$ is the slot for the sentence adverbial; in the content field the np^2 is filled by the indirect object, np^3 by the direct object, and the $advp^2$ position consists of the adverbials modifying the main verb.

In the base all kinds of constituents can be found, except the finite verb; in fact they are moved from their normal position to the base position of the sentence if they are topicalized or marked for contrast to something in the preceding sentence. When a constituent is moved to the base position its grammatical function is indicated by the fact that its position slot in the frame will be empty - a rule which holds for the Germanic languages except for English. In Danish the position of the subject is after the finite verb when something else but the subject is topicalized in the base position; but in English the subject remains in front of the finite verb even if some other constituens, as for example the object, have been topicalized.

In the pedagogical practice where students are taught how to fill in the words in the slots of the pattern correctly, it is said that if you can not see whether a word in the base is, say, subject or object, you move another constituent in the base position than the one which is there, and then you can see from which slot it has been moved: What is the function of Den plan in the sentence Den plan vedtog Rådet ikke enstemmigt? Put the constituent back again to the position from where it has been moved: Rådet vedtog ikke den plan enstemmigt. Answer: Den plan is the object moved from the content field to the base position.

In addition to the three mentioned fields, there is an final field, called the 'heavy' constituent field, because only heavy np constituents, i.e. constituents consisting of many words, often whole clauses, are placed there for stylistic reasons. The constituent placed in the heavy field is moved from either the np^1 position, the np^2 position or the np^3 position without any change in their grammatical or pragmatical function. But it is only placed there, and you can only see that it is placed there, if the $advp^2$ position is filled, normally with a one word constituent. So the h position is never filled when the $advp^2$ is empty. And if $advp^2$ is filled, the np constituent is either placed in its normal position in actualization field or content field or it is moved to the heavy field:

Base // actual. field // content field // heavy
// v^f / np^1 /$advp^1$ // v^{if} / np^2 np^3 /$advp^2$ // field
derfor // har / Rådet // taget /forslaget / op //
derfor // har / Rådet // taget / / op // det forslag der skulle imødegå alle de mulige invendinger der kunne komme fra 3. landes side

Rådet // opvervejer/ / // /at vedtage planen/ //
Rådet // tøver / / // / / med //at vedtage planen
derfor // har / Rådet/ ikke // anbefalet/ Kommissionen at vedtage planen/ //
derfor // har / Rådet/ ikke // givet/ Kommissionen tilsagn/om// at vedtage planen

Literal translation of the Danish sentences:

derfor har Rådet taget forslaget op
therefore has the Council taken the prosal up

derfor har Rådet taget op det forslag der skulle imodegå
therefore has the Council taken up the proposal which should oppo-

alle de mulige invendinger der kunne komme fra 3. lande
se all the possible objections which could come from 3 countries

Rådet overvejer at vedtage planen
The Council considers to pass the plan

Rådet tøver med at vedtage planen
The Council hesitates with to pass the plan .

derfor har Rådet ikke anbefalet Kommissionen at vedtage
therefore has the Council not recommended the Commission to pass

planen
the plan

derfor har Rådet ikke givet Kommissionen tilsagn om at
therefore has the Council not given the Commission promise about
to

vedtage planen
pass the plan

If you should write formal rewriting rules which can be implemented and run in a computer, this knowledge of the topology of the Danish sentence could be formulated in a formal (ECS) grammar like this:

(^x indicates that the x is optional, i.e. occurs zero or one time,
*x indicates that x occurs zero, one, or more times.)

G.I.
1. S -> ^b, v^f,^np, *$advp^1$, *v^{if}, *np, ^prt, *$advp^2$, ^h

2. b -> np
$advp^2$,

3. h -> v^2, *np, *$advp^2$, ^h

np, *np
sc (subordinate clause)

4. advp^2 -> adv^2
pp

5. pp --> p, np

6. np -> ˆdetp, *adjp, n, *pp, ˆsc

7. sc -> ˆconj, ˆnp, *adv^1, v^f, *v^{if}, *np, *advp^2.

This grammar will give the correct analysis of most Danish sentences (except for some refinement about 'light' constituents, and a special negation position which I will not discuss here). All positions except the finite verbs are optional; so a given position may be filled by the constituent that fits into the slot, or it may be empty if no constituent fits into the slot. But the problem is that when the analysis of a sentence is computed not only the correct analysis will be the result, but also a lot of wrong analyses.

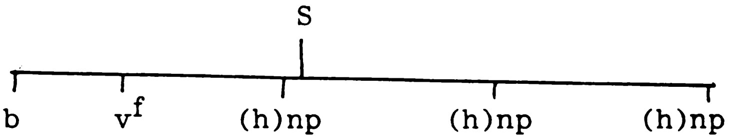
Here it is necessary to distinguish between sentences which from a grammatical point of view are ambiguous, and sentences which are grammatical unambiguous but will nevertheless result in grammatical wrong analyses in addition to the correct one.

If we analyse the sentence Adam elskede Eva, 'Adam loved Eve', we want the machine to give two analyses: one with Adam as subject placed in the base and Eva on np^3, and one with Eva as subject placed on np^1 and Adam as object placed in the base, corresponding to Adam måtte elske Eva and Adam måtte Eva elske respectively. The same will hold for the sentence Dette forslag vedtog Rådet, literal translation: 'this proposal passed the Council'; from a purely grammatical point of view this second sentence is ambiguous in the same way. This problem cannot be solved by a grammatical parser.

The problem with the grammar G. I is that it will give 6 analyses of the sentence: I 1982 sendte Kommissionen Rådet forslaget , literally:'in 1982 sent the Commission the Council the proposal' although it is not grammatical ambiguous:

ˆb	v^f	ˆnp	*advp^1	*v^{if}	*np	*advp^2	ˆh
i 1982	sendte	Kom.			Rådet forslaget		
i 1982	sendte				Kom. Rådet forslaget		
i 1982	sendte	Kom.			Rådet		forslaget
i 1982	sendte				Kom.Rådet		forslaget
i 1982	sendte				Kom.		Rådet forslaget
i 1982	sendte						Kom. Råd. forsl.

And in all 6 cases the tree structure will be the same:



In other words the parsing in the machine according to G.I. would yield 6 resulting trees with the only difference that in some of them one, two or three of the last np's would be represented by a mother node h.

The problem is that the interrelation between the empty slots in the pattern is not taken into account by the rules. The interrelations are in this example: np^1 will only be empty when the subject is placed in b; np^2 will only be filled in if np^3 is filled in; h will only be filled by an np if either np^1 or np$^{2-3}$ is empty and advp^2 is filled. The hat, ˆ, indicating optionality, and the star, *, indicating iterativity are not contextsensitive, so the interrelations cannot be reflected in the rules of G.I.

The Danish Eurotra-parser

Because of the overgeneration of the G.I grammar, the linguists in the Danish language group have built a grammar in which we have tried to describe thee interrelation between filled slots and empty slots. It looks like the following:

G.II.

1. s -> (ˆconj, sva, *v^{if}, ˆnpp, *advp^2, ˆsc
        (ˆconj, vsa, *v^{if}, ˆnpp, *advp^2, ˆsc

2. sva -> np, v^{f}, advp^{1}

3. vsa -> (advp^{2}, v^{1}, np, 'nadvp^{1}
(ap, v^{1}, np, 'nadvp^{1} ap = adjectival phrase
(pp, v^{1}, np, 'nadvp^{1} (= either... or
(sc, v^{1}, np, 'nadvp^{1} (
(np (demonstrative), v^{1}, np, 'nadvp^{1}

4. sc -> sbb, *v^{if}, ^npp, *advp^{2}

5. sbb -> (np, *advp^{1}, v^{f}
(subconj, np, *advp^{1}, v^{f}
(relpron, np, *advp^{1}, v^{f}
(relpron, *advp^{1}, v^{f}

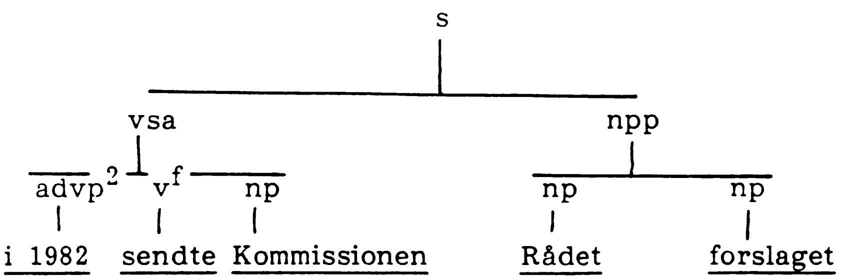
6. npp -> (^np, np
(^np, ap
(^np, sc

7. np -> ^detp, *ap, n, *pp, ^sc

8. advp^{2} -> (prep, ^h
(prt, ^h
(pp, ^h

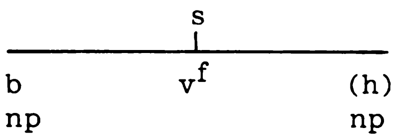
9. h -> (*advp^{2}, v^{if}, ^npp, *advp^{2}, ^h
(^np, np
(sc.

This G.II. will generate deeper trees than G. I because of the intermediate nodes sva, vsa or npp. But it will only generate one analysis of the sentence: I 1982 sendte Kommissionen Rådet forslaget:



The reason is that np^1 is only filled in if something else but the subject is placed in the base; it means that rule 2. cannot be used; and np^2 will only be filled if np^3 is filled according to rule 6; and h will only be filled if $advp^2$ is filled according to rule 7.

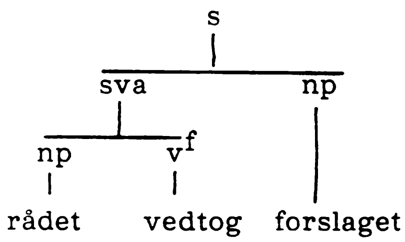
Both G.I and G.II are sets of ECS building rules, but G.II will make the translation rules from ECS to ERS much simpler than G.I would, even in the cases of grammatical ambiguity. Take the example: Rådet vedtog forslaget. G.I will create three nearly identical trees:

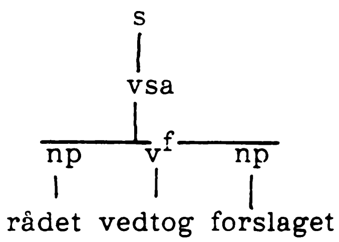


And from each of the three created trees the transformation rule used would be:

1. b(np), v^f, np => (subj, vb, obj
(obj, vb, subj.

G.II would only create two trees out of the sentence:





And there would be one translation rule for each tree:

1. sva(np, v^f), np => subj, vb, obj
2. vsa(np, v^f, np) => obj, vb, subj.

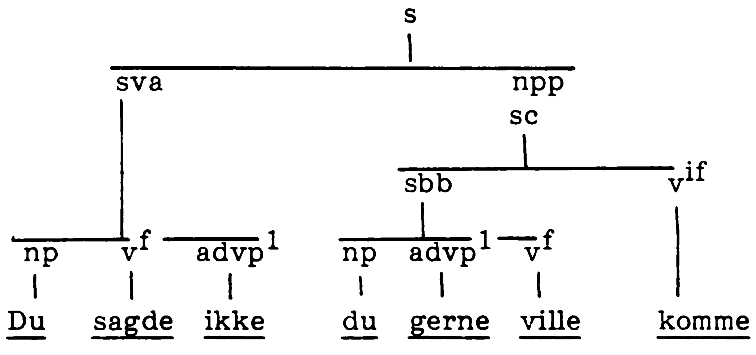
So G.I and the corresponding translation rules would create 6 ERS analyses of the sentence, while G.II and the corresponding translation rules will only create 2 ERS analyses of the sentence.

G. II is better than G.I in disambiguation power because the grammatical information indicated by the word order is used for disambiguation by G.II every time it is present, and the information can be indicated by the fact that a slot is not filled. In the sentence it is indicated that forslaget is not in the heavy constituent field, because adv^2 is not filled.

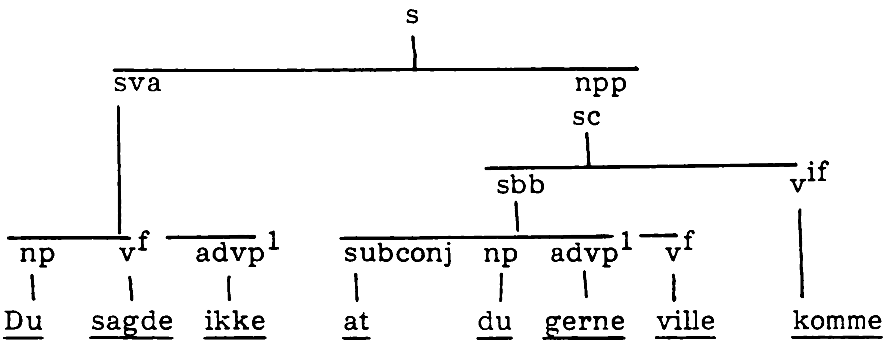
So the generalisations of a topological grammer, the topological interrelationship between constituents, the fact that one constituent can only have a certain position if another constituent has another position, can be registered by a grammar like G.II using more cycles in the generation, i.e. deeper trees with mother nodes indicating the word order of the sentence.

The G.II grammar has been designed by the Danish language group to solve quite a lot of the problematic examples in Danish. In the following I will show some examples of resulting analysis trees:

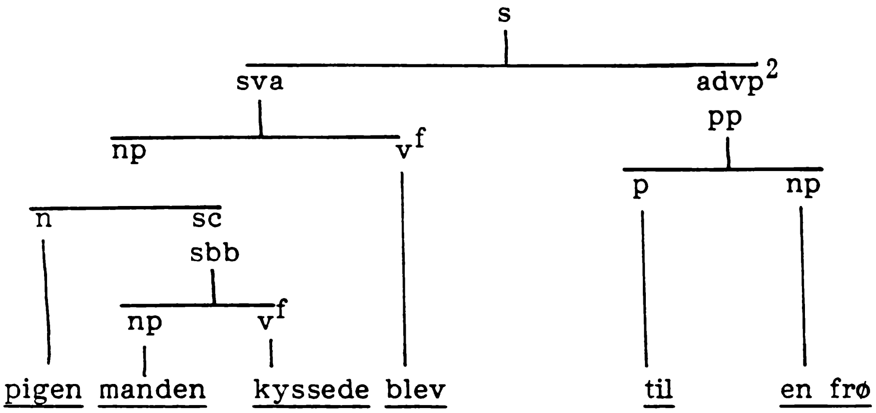
1. Subordinate clauses without conjunction:



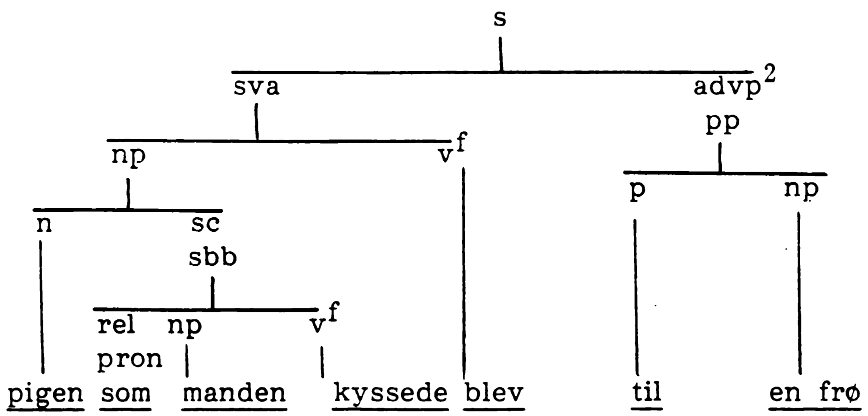
2. Subordinate clause with conjunction:



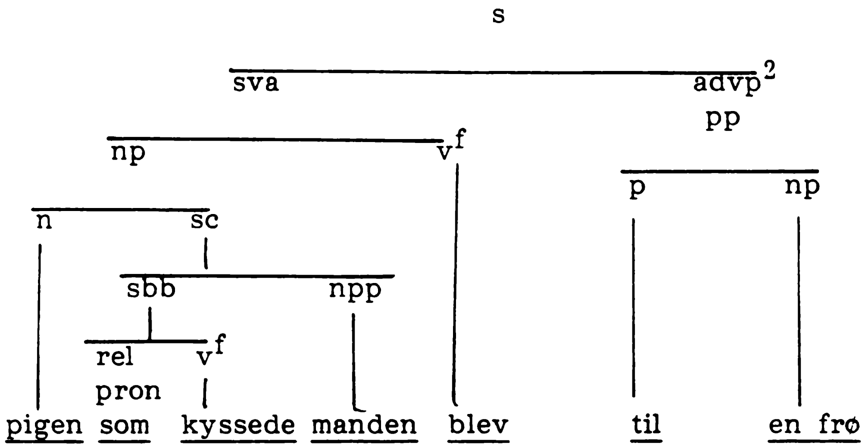
3. Relative clause without relative pronoun:



4. Relative clause with relative pronoun:



5. Relative clause with relative pronoun as subject:



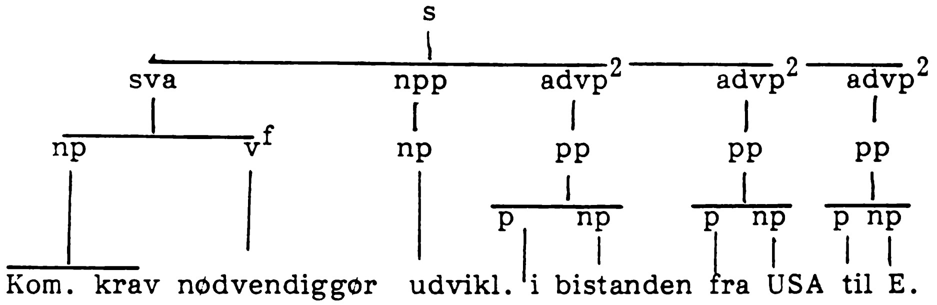
We have not solved all problems in automatic syntactic parsing of Danish sentences: We cannot analyse relative clauses in a 'distance position' , i.e. detached from its head: Europæiske firmaer har taget den udfordring op som ligger i dette emne. The sentence will be parsed by the grammar, but the anaphora from som to udfordring cannot be stated. We cannot parse subordinate clauses with a base: Det betød at hvis aftalen skulle indgås, måtte medlemslandene... And we cannot parse conditional clauses with word order as the main clause: Fortsætter udviklingen ikke, er forudsætningerne bristet.

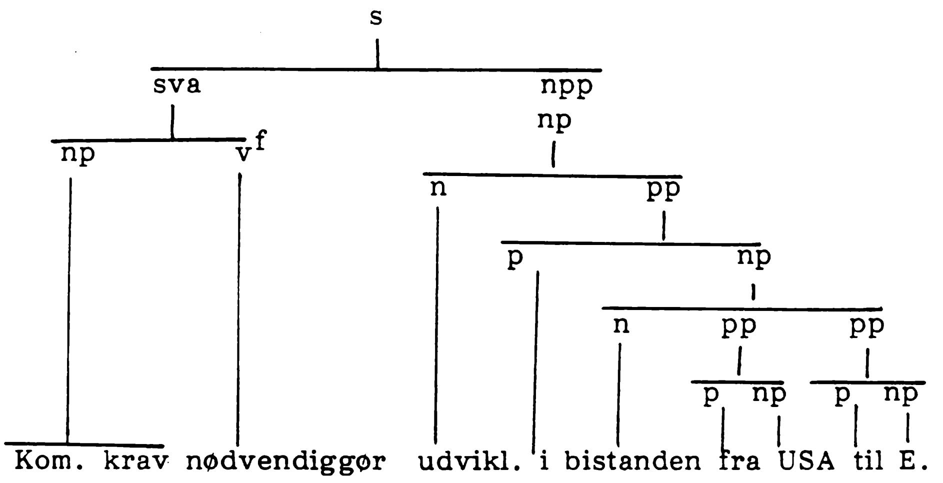
Semantic disambiguation

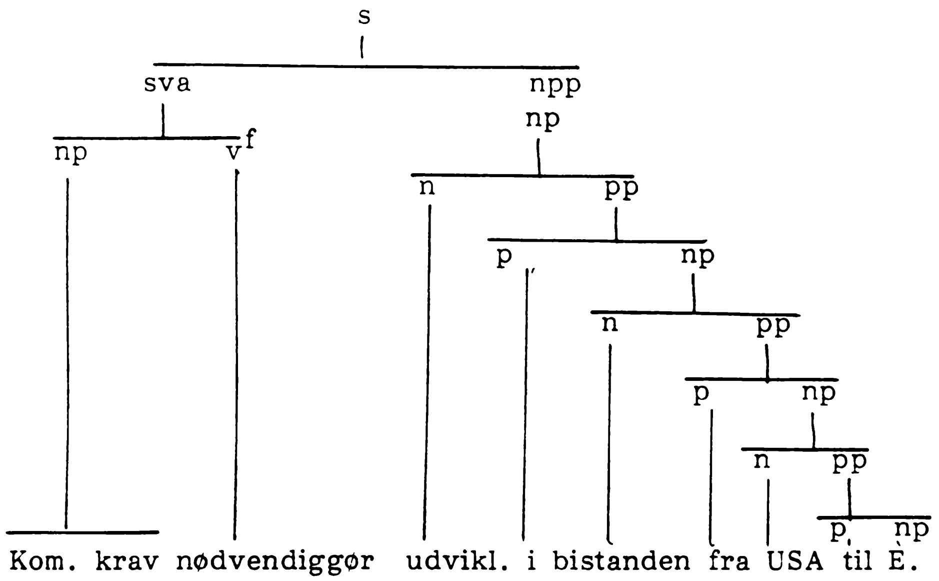
Sentences which are syntactically ambiguous but in many cases semantically unambiguous, are much more frequent than known from traditional grammars. Every time a sentence contains two or more pp's there will be many syntactically acceptable possibilities of pp attachment. The sentence

Kommissionens krav nødvendiggør udvikling i bistanden fra USA til Europa

will have 14 different resulting tree structures, when we parse it with the grammar G.II. I will here give 3 examples of attachment patterns, the flattest tree, the correct tree, and the deepest tree:







From a purely syntactical point of view all 14 attachment patterns are correct analyses of the sentences, and it is possible to find sentences with each of the 14 structures but other lexical material.

The problem should be solved by use of the feature rules mentioned earlier. What is described in the following is not part of the common Eurotra linguistic legislation, it is not even accepted or discussed in the Danish language group, so the only responsible for the ideas presented in the following is my self.

I imagine that to every noun in the IS dictionary there is assigned a semantic feature with the value chosen among a set og values organised in a hierarchy like the following:

```
        (--------------------------------------------------semiotic
        (
        (                      (-------------------------------time
        (                      (
        (         (abstract----(                 (--------- quality
entity--(         (            (           (state(-------- relation
        (         (            (           (     (---------- result
        (         (            (       (tem(     (--------- emotion
        (         (            (situ   (po (
        (         (            (ation-- (ral(nonstate(----- activity
        (  non    (                    (            (accomplishment
        (semio----(                    ( -------------- proposition
           tic    (                      (indivi-----(nomen agentis
                  (            (nonplace (dual      (--------person
                  (       (human(        (nonindi.(----organization
                  (       (     (        (vidual  (-communicat.tool
                  ( con-  (     ( ----------------------------place
                    crete (
                          (           (------------------------mass
                          ( nonhuman---(        (------natural kind
                                      (count----(arti    (-----part
                                                (ficial  (----whole
```

I will not in this paper give the definitions of these features but only show how the system is hierarchically organized, and give a list the lexical entries for the words in the example sentences:

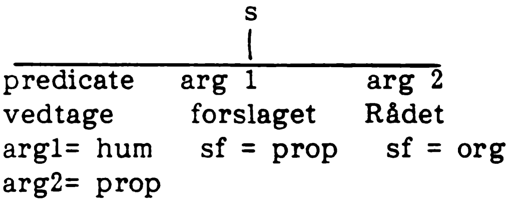
Rådet (semantic feature = organization)
forslag (semantic feature = proposition noun)
Kommissionen (semantic feature = organization)
krav: (semantic feature = proposition noun)
udvikling: (semantic feature = activity)
bistand : (semantic feature = result)
USA: (semantic feature = place)
Europa: (semantic feature = place)

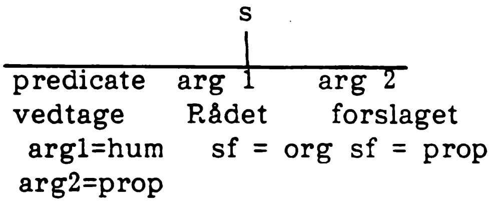
Then to every verb, noun (which has frames), adjective and preposition there is assigned a frame feature specifying the selection restriction from these words to their arguments and modifiers:

vedtage (sf of argument 1 = human, sf of argument 2 = proposition)
nøvendiggøre: (sf of argument 1 = entity, sf of argument 2 = situation)
krav: (sf of argument 1 = not non human, sf of argument 2 = entity, prep of argument 2 = til)
udvikling: (sf of arg 1 = human, sf of argument 2 = non state, prep of argument 2 = af, i)
i-1: (place where): (argument 1 = place)
i-2: (time during): (argument 1 = time)
i-3: (psychol cause): (argument 1 = emotion)
.
.

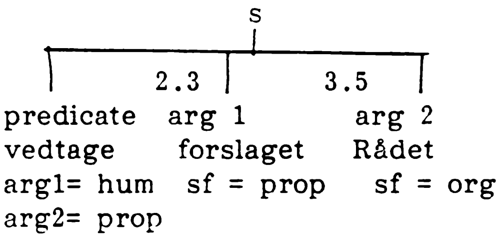
bistand: (sf of arg1= hum, sf of arg2 = nonstate, sf og arg 3= hum)
fra-1 (place from where): (argument 1 = not abstract)
.
.
til-1 (place to where): (argument 1 = not abstract)
til-2: (time until): (argument 1 = time)
til-3: ...
.
.

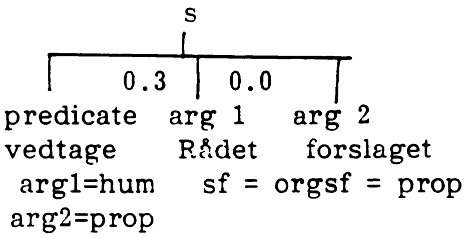
Now for each of the 2 generated is structures of the sentence Rådet vedtog forslaget, and for each of the 14 generated tree structures of the sentence Kommissionens krav nødvendiggør udvikling i bistanden fra USA til Europa, it is computed how well the semantic feature of the argument or modifier matches with the semantic feature selected by the frame of its head. We take the two IS trees :





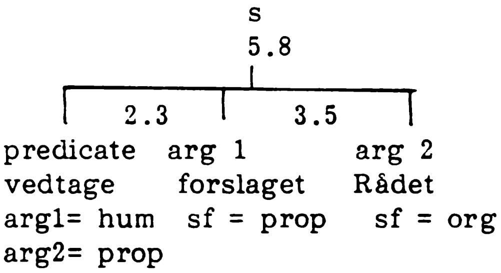
Then we measure the distance in semantic space from the feature value selected by the frame to the feature value of the slot filler in the hierarchy of features by walking from the frame value to the filler value counting 1.0 for every step upwards, and 0.1 for every step downwards. And then the generated tree structure with the shortest distance from frame value to filler value will be chosen automatically by the machine. This counting is a simulation of how unification works in the program hwen the hiararchy of feature values is implemented. It is possible to implement this preference mechanism.

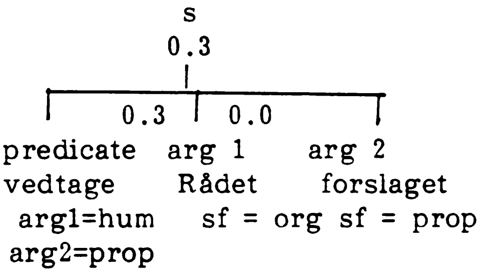




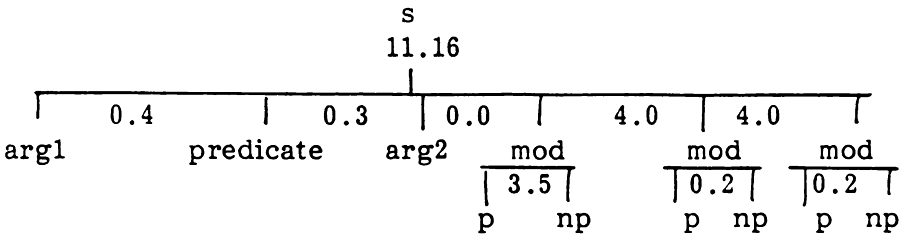
So the second tree will be selected by this preference mechanism. It is essential that it is a preference mechanism and not a killer rule which 'kill' all generated trees with mismatch between the value specified in the frame and the value of the slot filler, because if so, all the generated trees, even the wanted one of a metaphorical expression would be excluded: The new framework will solve the problems, the situation threatens to become worse.

If all the 14 generated IS trees of the second example should be computed there is an additional problem: The semantic distances to be compared by the preference mechanism are not distances of unifications in the same node in the tree. So we need to have a adding mechanism so that the two distances measured for argument 1 and argument 2 in the same tree can be added as a total value for the s node:

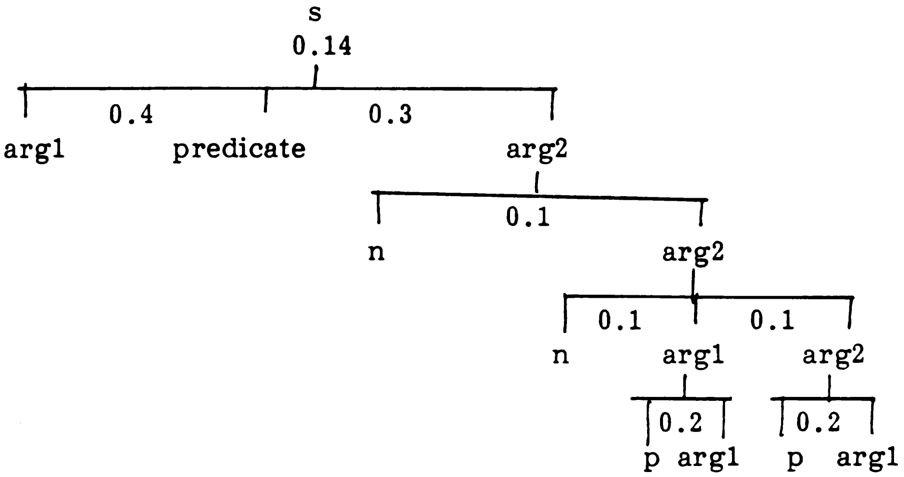




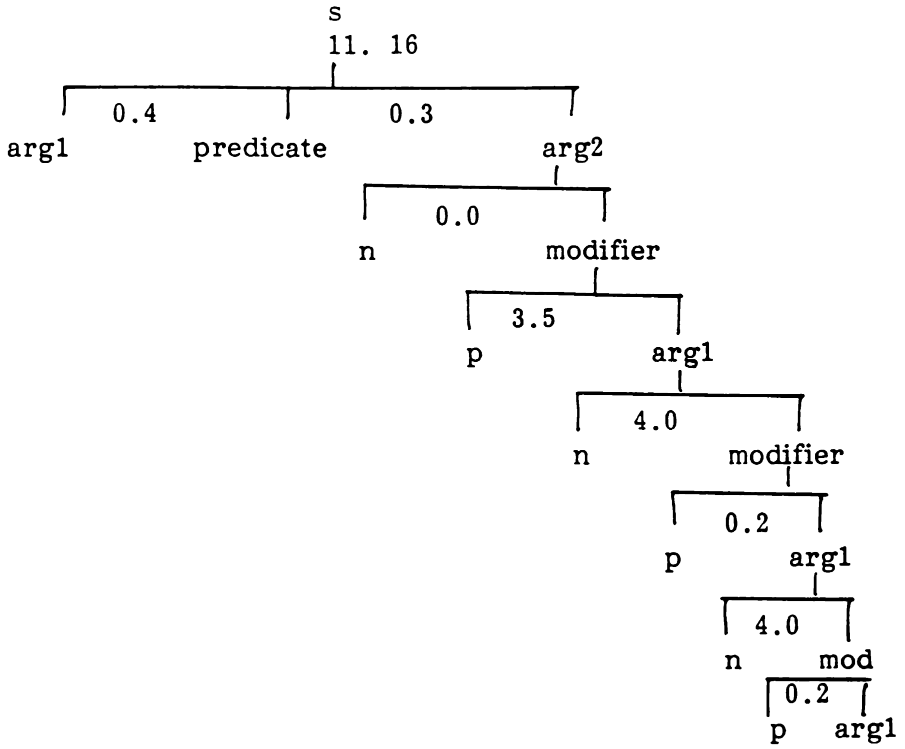
We have not implemented this mechanism yet. But if it can be done it will turn out that the tree structure which we want is the one which is selected automatically by the preference mechanism in the machine.



Kom. krav nødvendiggør udvikl. i bistanden fra USA til E.



Kom. krav nødvendiggør udvikl. i bistanden fra USA til E.



Kom. krav nødvendiggør udvikl. i bistanden fra USA til E.

References:

Diderichsen, Paul 1946. Elementær Dansk Grammatik. Gyldendal, København.

Diderichsen, Paul 1945: Dansk Sætningsanalyse. Dens Formaal og Metode. In
Meddelelser fra Dansklærerforeningen nr. 1 juni 1945.
Reprinted in
Heltoft, Lars og John E. Andersen (eds.) 1986: Sætningsskemaet og dets stilling - 50 år efter. NyS 16-17, Akademisk Forlag, København.

The Eurotra Reference Manual. Version 4.0 December 1987 Luxembourg.

1988: escdk.g Internal Eurotra document containing the Danish ECS grammar