DANISH FIELD GRAMMAR IN TYPED PROLOG

Henrik Rue
UNI-C, Danish Computing Center for Research and Education
Vermundsgade 5, DK 2100 Ø, Copenhagen, Denmark

ABSTRACT

This paper describes a field grammar for Danish and its implementations in a Prolog version with predeclared types. In comparison to the ususal S -> NP VP schema, this kind of grammar, where the first rule is S -> CNF FF NF CF enhances analysis effeciency because the fields specify constituents and syntactic function at the same time. The field grammar tradition is outlinedand an overview of the major rules of the Prolog program, which implements the grammar, is given.

FIELD GRAMMAR

A Syntactic Strategy

In terms of computational linguistics, field grammar may be viewed as a syntactic strategy, which offers the user the immediate constituents while at the same time giving their syntactic functions and the functional sentence perspective, in part at least. Field grammar furthermore facilitates the handling of discontinuous constituents, as will be shown.

Background

The field grammar of the Danish linguist Paul Diderichsen adequately describes constituent structure in Danish, while at the same time capturing both topicalization and syntactic roles. Diderichsens grammar "Elementær dansk grammatik" (1946) was developed from the 1940's onwards with the intention that it should be used as a common framework for grammar teaching in secondary school as well as on university level. This grammar has since served as one cornerstone of Danish grammatical thought.

Diderichsen's grammar is distinguished by a high degree of formalization, and it is one of the aims of the work presented in this paper to see how much of the original formalism can be implemented directly as a Prolog program, and whether it is necessary to make substantial changes in the definition and inventory of fields in order to make an executable program.

Prolog Dialect

The Prolog dialect used is the Danish prototype of Borland's TurboProlog. This is a typed prolog, and may be termed a hybrid between Prolog and Pascal. When seeing a sample grammar written in this dialect, one is impressed by the clarity it achieves: grammatical structures are statically described in the declaration of types. The dynamic part which enables one to get at these structures are the rules of the program. A further aim of this work, then, is to explore whether this clarity will prevail also in an elaborate grammar program.

Other Purposes

Apart from the purpose implicit in the aims we believe that field theory offers a sound (read: economic) starting point for a great variety of parsing purposes. As mentioned, the theory offers a combination of constituent structure analysis with syntactic and thematic analysis.

This will not only hold for the Scandinavian languages, but presumably also for other Germanic language like English, where one might abandon the S -> NP VP in favour of something on the lines of the SVC SVA SV SVO etc. clause patterns of Quirk (1972) et al.

In the work presented here, however, there is no exploitation of the topicalization facilities offered by the grammar.

A DANISH FIELD GRAMMAR

According to Diderichsen, the Danish sentence structure has four major fields, the connector field, the fundament field, the nexus field and the content field.

The four types are present in main sentences

S -> CONN FF NF CF

and three of them in subordinate ones:

SS -> CONN S-NF CF

where all fields except the nexus field (NF or S-NF) may be empty.

The CONN is the field for conjunctions.

The FF (for Fundament Field, which is the Danish topicalization device) may contain any complete constituent, which is there as a result of a movement from its field in the sentence: 'Moderen giver drengen gaven' vs. 'Gaven giver moderen drengen', ('The mother gives the boy a gift') where the second version differes in its thematical content only: it stresses the direct object as the theme.

The NF, for Nexus Field, contains a finite verbform, a possible subject plus adverbials modifying the verb; the internal structure of the nexus field differs in main and subordinate clauses.

The CF, for Content Field, contains two possible infinite verbforms, the objects and predicates plus adverbial and other modifiers.

The Grammar Declaration

So far the project has implemented field analysis of both main and subordinate sentences. However, not all topicalizations are handled yet: in questions, the fundament field may be empty too, but this is not incorporated in the program, as it remains to be seen whether an anlysis with the finite topicalized, that is moved into the fundament field, would be more fit for the purpose.

Clause structure

The following declarations describe main and subordinate clauses and furthermore the internal structure of the major fields:

```
S = s( CONN, FUNDF, NEXUSF, CONTENTF );
    nil;
    s_s( CONN, NEXUSF_S, CONTENTF )

CONN =
    nil;
    konj( KONJ )

FUNDF = fundf_n( NOMINAL );  /* No nil */
        fundf_a( ADVERBIAL );
        fundf_i( INF );
        fundf_c( CONTENTF )

NEXUSF = nexusf( FINIT, SUBJ, NADV )

NEXUSF_S = nexusf_s( SUBJ, NADV, FINIT )

CONTENTF = nil;
           contentf( INFFLD, OBJFLD,
                     CADVFLD )
```

These are the major fields. They may in turn be divided into subfields:

```
INFFLD = nil;
         inffld( INF1, INF2 )
```

means that Danish has a possibility of two auxiliaries, (the finite + one infinite), and implicitly that if INF2 is filled, then this will be the content verb. This treatment is not quite adequate, actually, but it follows Diderichsen's schema.

```
OBJFLD = nil;
         objfld( NOMINAL, PREPG, NOMINAL )
```

the object field, which at the moment contains a quick-and-dirty solution to the problem that the indirect object may be expressed by a prepositional phrase in Danish, the solution being the incorporation of an unwarranted PREP subfield.

It should be noted in passing, that the connector field in Diderichsen's formalism is one of the places where the system will not be able to hold on to the original. This field is part of scemata not only for sentences, but also for noun- and adverbial phrases, where it may contain i.a. preposition. The system thus has to distinguish between the two types of connector fields in order to avoid the generation of spurious analysis results.

Discontinuous Verbal Particles

In Danish some verbs are either prefigated or obligatorly constructed with a particle, a preposition actually, which moves to the end of the sentence with all finite forms: 'oplade' ('charge') but 'han lader batteriet op', ('he charges the battery'); 'lukke op' ('open up') but 'han lukker døren op' ('he opens the-door up'). The same phenomenon exists in German: 'Peter gab sein rauchen auf'. This is one of the places where field grammar shows its force as a syntactic strategy, because the phenomenon of discontinuity is handled in a straightforward way at the first level of analysis:

```
ADVFLD = nil;
         cadvfld( CADF, CADF )
```

with

```
CADF = nil;
       prep( PREP );
       cadf( ADVERBIAL )
```

where CADF is the field for i.a. contential adverbs, but also for disjunct verbal

particles. These are acommodated by splitting the original Diderichsen subfield for content adverbials into two further subfields, one of which will contain the verbal particle (if any) the other the regular content adverbials. This is sufficient for the declaration of the grammar; how our analysis handles the various fields will be shown in a later section.

Phrasal structure

Syntagmatic structures are also divided into fields. As the system stands it is implemented for adverbial phrases, but not yet for noun phrases. These are at the moment structured in a way, that is pretty much on the NP -> Det AdjP N lines. As regards adverbials, the structure given is only one of several possible:

```
NOMINAL = nil;
      nominal( ART, ADJEKTIVAL, SUBKERN
               PREPP, CS )

ADVERBIAL = nil;
            adverbial( CONN,
            DEGREEF, SITUATF, ADVKERN,
            PREPP, CS )
```

The CS is a symbol representing subordinate sentences, which have the form:

```
CS = nil;
     cs( S, SYNT )
```

where S is the field structure, and SYNT the corresponding syntactical structure of the subordinate sentence represented by the token of the symbol type CS.

Verb phrases, on the other hand, do not exist as such. Instead we have:

```
FINIT = finit( VERB, VERB, TEMPG )
INFINIT = infinit( VERB, VERB, TEMPG )

VERB = Symbol
```

which means that a verb, whether it be finite or infinite, is described by a structure, which consists of 1) the verbal form itself as it is found in the sentence (the first 'VERB'), 2) a lexical unit, (the second 'VERB', which will be found as a result of the analysis of the sentence, and which will leave the fields for infinite form empty) and 3) a complex description, TEMPG, of tense, aspect, voice, modality and the telic/atelic property of the situation described by the verb. This TEMPG is used of the sentence as a whole also.

In this way a 'FINIT' in a sentence will have either an auxiliary, a finite verbform missing the verbal prefix or the full, finite form of the content verb in the first 'VERB' slot when field analysis is carried out. The result of the syntactical analysis which follows, will be in the second 'VERB' slot.

Syntax

The system also comprises a syntactic part, based on traditional school grammar:

```
SYNT = synt( SUBJ, VERB, NADV, SUBJPRED,
             OBJ, OBJPRED, IOBJ, CADV,
             TEMPG )
```

where NADV and CADV are the adverbial modifiers of the nexus and the contentfield respectivily. The other mnemonics should be self evident.

The Dictionary

As the dictionary of the system has not been given much attention yet, and as it works on a purely ad hoc basis, it will not be treated in this paper.

ANALYSIS

Analysis runs in two steps, one carrying out the field analysis, the other handling the syntactical interpretation of the result of the field analysis.

Field Analysys

Field analysis is carried out by a call to the following major rule:

```
is_s( I, O, s( CONN, FUNDF, NEXUSF,
                CONTENTF ) ):-
    is_forb( I, I1, CONN, FEATC ),
    FEATC <> subord,
    is_fundf( I1, I2, FUNDF ),
    is_nexusf( I2, I3, NEXUSF ),
    is_contentf( I3, O, CONTENTF ).
```

which applies the following rules in order to succeed (or fail):

```
is_fundf( I, O, fundf_n( NOMINAL ) ):-
    is_nomen( I, O, NOMINAL ), I <> O.

is_fundf( I, O, fundf_a( ADVERBIAL ) ):-
    is_adverbial( I, O, ADVERBIAL, _ ),
    I <> O.

is_nexusf( I, O, nexusf( FINIT, NOMINAL,
                         ADVERBIAL ) ):-
    is_finit( I, I1, FINIT ),
    is_nomen( I1, I2, NOMINAL, _, _ ),
    is_adverbial( I2, O, ADVERBIAL, _ ).
```

and

```
is_contentf( I, O, contentf( INFFLD,
                    OBJFLD, CADVFLD ) ):-
    is_inffld( I, I1, INFFLD ),
    is_objfld( I1, I2, OBJFLD ),
    is_cadvfld( I2, O, CADVFLD ),
    I <> O.

is_contentf( I, I, nil ).
```

As a consequence of having a possible nil-filling for a major field, the content field, it becomes necessary to explode the number of rules which identify and collect compound verb forms, or in other words what is gained in the simplicity of the grammar is lost again by the number of rules.

Discontinous Verbal Particles

As an example of the rules handling the major fields, we shall take a look at the rule, which picks out discontinous verbal particles.

The rules which handle the adverbial subfield of the content field contain a specification for the particles, as they allow for the class of prepositional adverbs:

```
is_cadvfld( I, O, cadvfld( PREPG,
                    C_ADVERBIAL ) ):-
    is_advprep( I, I1, PREPG ),
    is_c_adverbial( I1, O, C_ADVERBIAL ),
    I <> O.

is_cadvfld( I, O, cadvfld( C_ADVERBIAL,
                           PREPG ) ):-
    is_c_adverbial( I, I1, C_ADVERBIAL ),
    is_advprep( I1, O, PREPG ),
    not_nom( O ), I <> O.
```

The prepositional adverbs are then picked up by the rule:

```
  is_advprep( I, O, prep( PREP ) ):-
    fronttoken( I, PREP, O ),
    dic_prep( X ), X = PREP.
```

which in fact is an ad hoc rule to circumvent the restrictions posed on the system be the typing facility. During syntactic analysis the disjunct particles are collected with the verb by the rule extract_disco_vpart, as will be demonstrated in the following.

Syntactic Analysis

There is one major clause for syntactic analysis, 'is_syn', which is called by the top level anlysis clause 'start':

```
start:-
  write("Skriv en sætning"),nl,
  readln( Line ),
  is_s( Line, "", S ),
  is_syn( S, SYNT ),
  nl, write("Feltanalyse:"),nl,
  skriv_s( S, 0 ), nl,
  nl, write("Syntaktisk analyse:"), nl,
  skriv( SYNT, 0 ), nl, fail.

is_syn( S, SYNT ):-
  extract_vg( S, VERB1, TEMPG ),
  extract_disco_vpart( VERB1, S, VERB ),
  extract_advg( S, NADV, CADV ),
  interpret_nominals( S, VERB, SUBJ,
                      SUBJPRED, OBJ,
                      OBJPRED, IOBJ ),
  collect_synt( VERB, NADV, SUBJ,
                SUBJPRED, OBJ, OBJPRED,
                IOBJ, CADV, TEMPG, SYNT ).

is_syn( nil, nil ).
```

The claim was that field grammar facilitates syntactic analysis, and we shall now endeavour to support this claim by looking at the handling of the noun phrases.

The major rule is 'interpret_nominals', which has the form:

```
interpret_nominals(
   s( _, FUNDF, NEXUSF, CONTENTF ),
   VERB, SUBJ, SUBJPRED,
         OBJ, OBJPRED, IOBJ ):-
   syn_nomfund( FUNDF, NEXUSF, CONTENTF,
                VERB, SUBJ, SUBJPRED,
                      OBJ, OBJPRED, IOBJ).
```

For transitive verbs the following version of a 'syn_nomfund' rule generates the filler in the fundament field as subject, and two fillers to the object and indirect object slots; if there is only one filler in the object subfield this will be the object:

```
syn_nomfund(
   fundf_n( FUNDFN_I ),
   nexusf( _, nil, _ ),
   CONTENTF,
   VERB, subj( FUNDFN_O ), nil,
   OBJS, nil, IOBJS ):-
     trans_verb( VERB, DITRANS ),
     check_sentcomp( FUNDFN_I, FUNDFN_O ),
     extract_obj( nil, DITRANS, CONTENTF,
                  OBJS, IOBJS ),!.
```

where the interesting call is the one to 'extract_obj', where the following will match (the 'check_sentcomp' in the following rules should be disregarded, as it has nothing to do with the analysis of the arguments proper, it only activates a syntactic analysis of a possible clausal complement to the given nominal kernels):

```
extract_obj( nil, _,
  contentf( _, objfld( NOM_I, nil, nil ),
            _ ),
  obj( NOM_O ), nil ):-
     check_sentcomp( NOM_I, NOM_O ),!,
     is_noprep( NOM_O ).

extract_obj( nil, DITRA,
  contentf( _,
            objfld( NOM1_I, nil, NOM2_I ),
            _ ),
  obj( NOM2_O ), iobj( NOM1_O ) ):-
      DITRA <> nil,
      is_noprep( NOM1_I ),
      check_sentcomp( NOM1_I, NOM1_O ),
      check_sentcomp( NOM2_I, NOM2_O ),!.

extract_obj( nil, DITRA,
  contentf( _,
            objfld( NOM1_I, prep( PREP ),
                    NOM2_I ),
            _ ),
  obj( NOM1_O ), iobj( NOM2_O ) ):-
      DITRA <> nil,
      is_noprep( NOM1_I ),
      check_tilfor( PREP ),
      check_sentcomp( NOM1_I, NOM1_O ),
      check_sentcomp( NOM2_I, NOM2_O ),!.

extract_obj( nil, _,
             contentf( _, nil, _ ),
             nil, nil ).

extract_obj( nil, _, nil, nil, nil ).
```

Even if simplicity is in the eye of the beholder, we are confident that the rules above are not very complicated.

It is evident, however, that at least one necessary modification to the claim must be that the two structures for 'The mother gives the boy a present' example:

```
s(fundf_n(X),nexusf(finit(Y),nil,_),
  contentf(objfld(nominal(XX),_,
                  nominal(YY))))

s(fundf_n(X),nexusf(finit(Y),subj(Z),_),
  contentf(objfld(obj1(XX),_,nil))
```

can only be distinguished from each other in analysis by a call to a rule that operates at the lexical level of the verb and its arguments.

Discontinouos Verbal Particles

In the syntactic analysis, a possible discontinous verbal particles is discovered by the rule extract_disco_vpart, which has the form:

```
extract_disco_vpart(
    VERBIN,
    s( _, _, _,
       contentf( _, _,
                 cadvfld( prep( PREPIN ),
                          _ ))),
    VERBOUT ):-
  dic_v( VERB, _,_,_,_,_,_,_, discon, _ ),
  VERB = VERBIN,
  dic_v_discon( VERB, PREP, _, _ ),
  VERB = VERBIN, PREPIN = PREP,
  concat( VERB, " ", X ),
  concat( X, PREP, VERBOUT ).
```

PERFORMANCE

The system consists of 35 complex grammatical objects, eg. FUNDF, NOMINAL, with a total of 69 possible internal structurings. There are 18 simple grammatical types, eg. INF, ADV.

There are 77 predicate types for the analysis proper, and another 36 types used for prettyprinting the results of the analysis.

There are 72 rules for the handling of the field grammar analysis, and 74 rules for the syntactic analysis.

Finally there are 70 actual rules to the 36 types of prettyprinting.

This reflects on one of the shortcomings of the typing system: you need a separate predicate for each object type you want to type out. Up to a certain point one may have one predicate type handle several object types, but what happens is that instead the compiler generates different predicate types behind your back. All in all one must say, that running on an IBM XT you will very soon hit the upper limits of the various tables in the compiler, when you attempt to exploit the typing facilities offered.

The sentence 'den meget gode dreng som giver moderen gaven lukker øl op med et redskab' ('The very good boy who gives the-mother the-gift opens beer up with a tool') takes a total of 21.13 seconds in field and syntactic analysis:

```
Field analysis:
FUNDAMENTFIELD
  FUNDF
    NOM      dreng
      DET    den
      ADJ    gode
        ADV  meget
```

```
      CONJ          som
      NEXUSFIELD
        FINIT
          VERB      giver
      CONTENTFIELD
        OBJ-SUBPRED FIELD
          OBJ1/SP
            NOM     moderen
          OBJ2/OP   gaven

NEXUSFIELD
  FINIT
    VERB      lukker
CONTENTFIELD
  OBJ-SUBPRED FIELD
    OBJ1/SP
      NOM     øl
  CONTENT ADVERBIAL FIELD
    VB-PART   op
    CF-ADV
      PREP    med
      NOM     redskab
        DET   et


SYNTACTIC ANALYSIS
SUBJ    NOM   dreng
        DET   den
        ADJ   gode
          ADV     meget
        SUBJ      NOM   RelT^
        VERB            give
        DIR-OBJ   NOM   gaven
        DAT-OBJ   NOM   moderen
        TEMP      tempg(pres,contmp,act,
                        nil,imperf,atelic)
VERB          oplukke
DIR-OBJ   NOM  øl
CF-ADV    PREP med
          NOM  redskab
          DET  et
```

CONCLUSIONS

As the project is still running, it is too early to propose any firm conclusions. It has been seen ,though, that a field analysis for Danish is easily implemented in Prolog, that for the most part shortcuts are merely programming conveniences, and that typed Prolog using mnemotecnic variable names enhance readability and thereby adaptability.

On the other hand, our experience has shown that expanding the system is easy but expensive in process time. When eg. subordinate clauses were introduced to noun phrases and adverbial phrases, this was a very simple operation in the grammar (it required the addition of a single symbol) but it had severe consequenses for execution time: roughly a 25% increase in analysis time for the sentence 'den meget gode dreng vil gerne få givet moderen den gode gave' ('The very good boy will be-happy-to manage-to give the-mother the-present'): 1.21 seconds before, 1:60 after the extension.

Experience has also shown that typed Prolog is a hindrance for the writing of rules, which handle different constructors: the compiler generates separate rules for each cnstructor, and that leaves you with a severe problem of adequacy of space in the rule tables, when running on an IBM XT.

REFERENCES

Paul Diderichsen, Elementær dansk grammatik, Copenhagen 1946

Randolph Quirk, Sidney Greenbaum, Geoffry Leech & Jan Svartvik, A Grammar of Contemporary English, London 1972

PC PROLOG, Tutorial and User's guide, Prolog Development Center, Copenhagen 1985, 1986.