NAVF'S EDB-SENTER FOR HUMANISTISK FORSKNING
NORGES ALMENVITENSKAPELIGE FORSKNINGSRÅD
Villavei 10, 5000 Bergen — Telefon 21 00 40
Postadresse: Postboks 53, 5014 Bergen - Universitetet

Knut Hofland: Implementering av en metode for syntaktisk analyse av norsk.

Foredrag til nordiska datalingvistikdagar i Göteborg 10. - 11. oktober 1977.

---

1. INNLEDNING

Denne artikkelen gir en oversikt over et samarbeidsprosjekt mellom stipendiat Svein Lie, Nordisk institutt, Universitetet i Oslo og NAVF's EDB-senter.

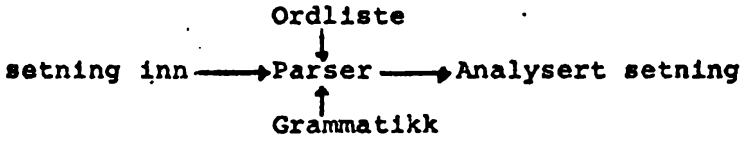
Formålet med prosjektet har vært å lage et programsystem som leser inn en vanlig norsk setning og som foretar en syntaktisk analyse av denne. Prosjektet har to hovedmål:

1. En ønsker å vinne økt innsikt i norsk syntaks ved å prøve ut grammatikken på et stort antall setninger og undersøke de ukorrekte analyser for å modifisere reglene og prøve på nytt.

2. Bygge opp et generelt programsystem for syntaktisk analyse som også kan nyttes av andre.

Prosjektet bygger på et notat av Martin Kay:"Morphological and syntactic analysis" utdelt på nordisk sommerskole i språklig databehandling sommeren 1974, samt et uferdig program av samme forfatter. Etter en første kontakt høst 75/vår 76 ble arbeidet ved NAVF's EDB-senter påbegynt i mai 1976. Sommeren 1976 var en første versjon av programmet klart. Dette opererte med en liten engelsk grammatikk beskrevet i Martin Kays notat og analyserte som kontrollsetning en setning fra et større eksempel i dette notatet. En norsk grammatikk ble så lagt inn og denne var i stadig utvikling høsten 1976/våren 1977. Våren 1977 ble programmet overført til Oslo slik at det nå kan kjøres både i Bergen og Oslo. NAVF's EDB-senter har bidradd i prosjektet med 2-3 månedsverk.

Oversiktsfigur:



Siden det er det syntaktiske aspektet som i denne forbindelse er det mest interessante, har en holdt utenfor en morfologisk analyse. Ordlisten består derfor av ordformer med opplysninger om ordklasse og et sett med egenskaper som f.eks. kjønn, tall, person, transitiv o.l. For ord som ikke står i ordlisten må brukeren slå inn disse opplysningene og ordet vil da bli satt inn i ordlisten.

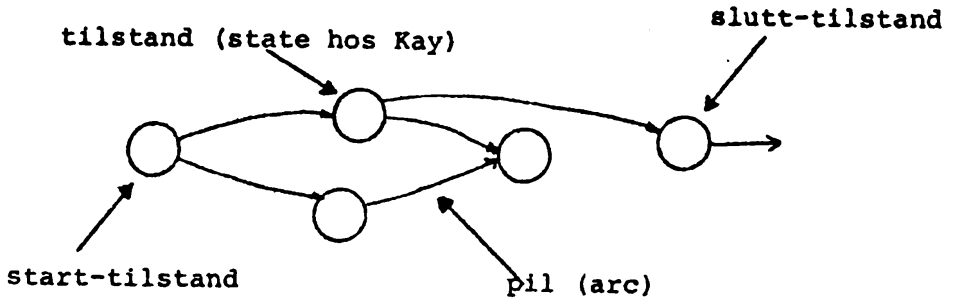
Implementering av en metode for syntaktisk analyse av norsk
Knut Hofland
Proceedings of NODALIDA 1977, pages 50-64

Eksempel på analyse.

```
GI SETNING :

>DA VI KOM FRAM BAD VI DIREKTØREN PÅ HOTELLET
>VÆRE SNILL OG GI OSS ET VÆRELSE MED BAD.

FØLGENDE ORD FINNES IKKE I ORDLISTE. ANGI ORDKLASSE MM.

VÆRE

>VERB INFINITIV HJVB COP
VÆRELSE

>SUBST NEUTR SG UBEST

S /
   •ADVL2
   ADVL
      •S
      S / ADV-LS
         •UKONJ
         UKONJ / ADVKNJ                          DA
         •SUBJ
         NP / NOMINATIV / BEST
            •OVERLEDD
            PRON / NOMINATIV                     VI
         •VERBFIN
         VF / PRET / INTRANS                     KOM
         •ADVL3
         ADVL
            •ADVERB
            ADV                                  FRAM
   •VERBFIN
   VF / PRET / SAMSEVB                           BAD
   •SUBJ
   NP / NOMINATIV / BEST
      •OVERLEDD
      PRON / NOMINATIV                           VI
   •OBJ
   NP / BEST / SG / MASK
      •OVERLEDD
      SUBST / MASK / SG / BEST                   DIREKTØREN
      •ADVL
      ADVL / PREPLEDD
         •PREP
         PREP                                    PÅ
         •STYRING
         NP / BEST / SG / NEUTR
            •OVERLEDD
            SUBST / NEUTR / SG / BEST            HOTELLET
   •OBJINF
   NP / NP-S/INF
      •INF-FRASE
      INF-FRASE
         •INFINITIV/OVERLEDD
         VERB / INFINITIV / HJVB / COP           VÆRE
         •PREDIKATIV
         ADJP
            •OVERLEDD
            ADJ                                  SNILL
      •SKONJ
      SKONJ                                      OG
      •INF-FRASE
      INF-FRASE
         •INFINITIV/OVERLEDD
         VERB / INFINITIV                        GI
         •INDIROBJ
         NP / BEST
            •OVERLEDD
            PRON / AKK                           OSS
         •OBJ
         NP / UBEST / SG / NEUTR
            •BEST
            ART / NEUTR / SG / UBEST             ET
            •OVERLEDD
            SUBST / NEUTR / SG / UBEST           VÆRELSE
            •ADVL
            ADVL / PREPLEDD
               •PREP
               PREP                              MED
               •STYRING
               NP / UBEST / SG / NEUTR
                  •OVERLEDD
                  SUBST / NEUTR / SG / UBEST     BAD
```

2. OVERSIKT OVER METODEN

Det gis her en oversikt over Martin Kays metode med de forandringer som er gjort av oss.
Grammatikken beskrives som et nettverk.



Et nettverk er et sett med tilstander (states) som er forbundet med (overgangs)piler (arcs). Pilene ut fra en tilstand gir de mulige etterfølgende tilstander. En tilstand som ikke har noen innløpende piler vil være en starttilstand. Tilsvarende vil en tilstand som ikke har noen neste tilstand, være en slutt-tilstand. Til enhver pil i nettverket vil det være et sett med betingelser som må være oppfylt dersom en skal ta denne veien i nettverket (dette settet kan være tomt og det finnes ubetingete overganger i nettverket). Til en pil kan det også høre et sett av aksjoner som skal utføres dersom denne overgangen i nettverket blir foretatt. En betingelse kan f.eks. være knyttet til ordklasse og egenskaper til et aktuelt ord i en setning mens aksjonen kan være å sette navn på et ord eller et setningsledd for senere å kunne bruke dette navnet for å sjekke egenskaper ved ordet. Denne fysiske sammenknytningen av et navn med en del av setningen kalles et register, og det vil bli opprettet nye registre underveis i analysen. Disse er metodens hukommelse og er en av de viktigste delene i hele prosessen.

Deler av nettverket som brukes flere ganger f.eks. for å analysere nominale uttrykk, skilles ut som egne undernettverk og det knyttes forbindelse fra hovednettverket til undernettverkene.

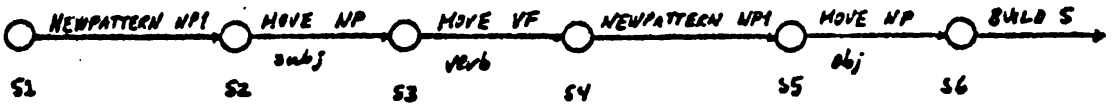
Det er 4 hovedtyper av overgangspiler i nettverket:

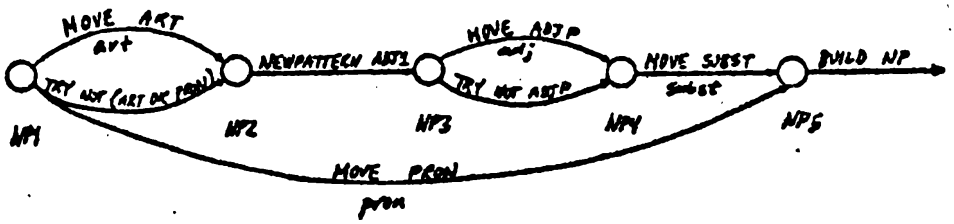
- overganger som godkjenner ord eller setningsledd på grunnlag av ordklasseopplysninger (MOVE, GO TO hos Kay) og går videre i setning
- overganger som starter søking etter nye konstituenter i et subnettverk (NEWPATTERN)
- overganger som gjør det mulig å gå ubetinget fram i nettverket (TRY)
- overganger som knytter sammen ord eller setningsledd til større enheter (BUILD)

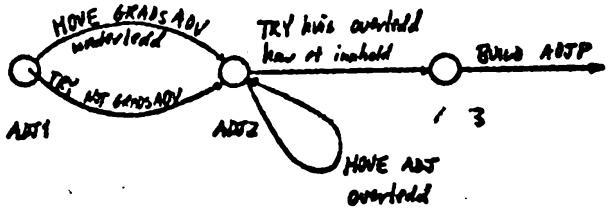
Vi har valgt å la NEWPATTERN bare være en pil som starter søking i et subnettverk. Undersøkelsene om en fant det en lette etter i subnettverket, gjøres i tilstanden etter NEWPATTERN pilen.

Eksempel på en grammatikk:

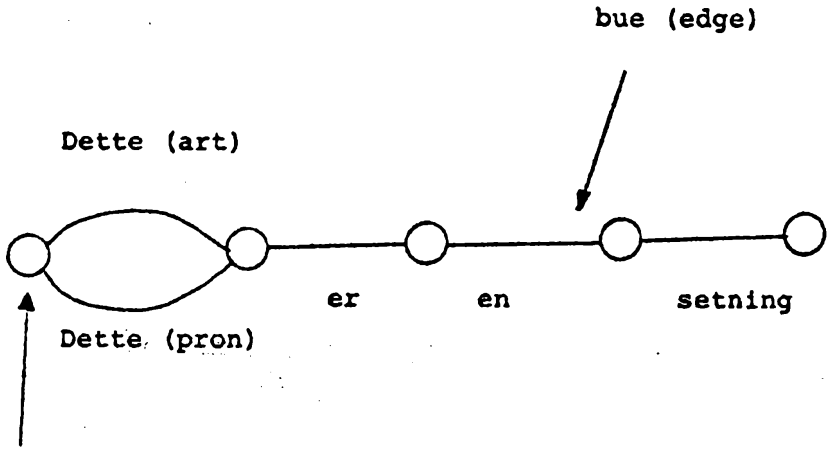
Symbolene etter MOVE eller TRY er en betingelse som er knyttet til ordklassen til den aktuelle bue. Navnene under en pil er et registernavn som den aktuelle bue blir satt i dersom en velger denne pilen.







Setningen som skal analyseres (og senere resultatene som fremkommer fra analysen), lagres i et nettverk som har fått navnet kart (chart hos Kay).



Et kart består av et sett med knuter, en for hvert ordmellomrom. Mellom to knuter går det en bue (edge). En bue inneholder et ord med tilhørende opplysninger som ordklasse o.l. som kan være fremkommet fra en morfologisk analyse eller tatt fra en ordliste. Dersom et ord har flere betydninger, vil det gå flere buer mellom de samme knutene.

Analysen foregår på den måten at alle mulige veier i kartet prøves mot alle mulige veier i grammatikken. Dersom slutt-tilstanden i grammatikken nås og en har kommet gjennom hele setningen, er setningen grammatisk. Registrene vil da inneholde opplysninger om hva som er subjekt, objekt o.l.

Når en finner konstituenter i setningen, så innføres det nye buer i kartet som spenner over de enkelte ordene som hører sammen.

Prøvingen av alle veier i kartet mot alle veier i grammatikken gjøres ved hjelp av en jobbliste. Et element i denne listen består av en bue (et ord eller en konstituent), en tilstand og registerlisten slik den var idet en forlot forrige tilstand.

På grunnlag av opplysningene om buen (edge) og innholdet i registrene, kan en velge ingen, en eller flere av pilene ut fra tilstanden, og dette gjøres ved at en produserer nye elementer (jobber) og henger disse på jobblisten (ett element for hver pil).

For også å kunne prøve nye buer som blir laget underveis mot grammatikken, har en innført begrepet venteliste (wait list). Dette er en liste med tilstander som er knyttet til en knute. For hver tilstand inneholder ventelisten også registerlisten slik den var da en kom til knuten. Dersom en tilstand inneholder en pil som starter en søking etter et nytt ledd (NEWPATTERN) setter en neste tilstand på ventelisten til den venstre knuten til den aktuelle bue.

Siden en jobb inneholder alle opplysninger som skal til for å sammenligne en bue mot et sett med piler fra en tilstand, er det likegyldig i hvilken rekkefølge en velger jobbene. Det mest vanlige er å velge den jobben som sist ble satt på jobblisten.

DEN FULLSTENDIGE ALGORITMEN

I Ved start.

For hver bue b_i som går ut fra knute nr. 1 sett R til registeret start = b_i, lag en ny jobb

b_i, S1, R

og sett denne på jobblisten.

S1 er første tilstand i grammatikken.

II Idet en setter nye jobber på jobblisten, undersøkes først om jobben finnes der fra før, eller har blitt utført tidligere.

III Så lenge det er jobber igjen på jobblisten, tar en ut og utfører denne.

Aktuell jobb har bue B, tilstand T og registerliste R.
For hver pil P_i fra tilstand T (som går til tilstand t_i) utføres følgende:

1. fortsett dersom betingelsene er oppfylt

2. utfør aksjonene som hører til den aktuelle pil, R' er nå den eventuelt ajourførte registerlisten

3. hvis P_i er MOVE.
   Lag en ny jobb for hver etterfølgende bue b_j til B (dersom t_i har en pil ut som er en BUILD ordre, lages jobb bare for en bue) og med oppdatert registerliste.
   ny jobb: b_j, t_i, R'

4. hvis P_i er TRY,
lag en ny jobb med samme bue og oppdatert registerliste.
Ny jobb: B, t_i, R'

5. hvis P_i er NEWPATTERN t,
t er første tilstand i et subnettverk.

a) Sett tilstand t_i og registerliste R på venteliste til B's venstre knute (dersom disse ikke finnes der fra før).

b) Sett R' til start = B
Lag to nye jobber:
B, t_i, R (denne jobben er for å komme videre i hovednettverk dersom vi ikke finner det ønskede ledd).
B, t_i, R'.

6. hvis P_i er BUILD K

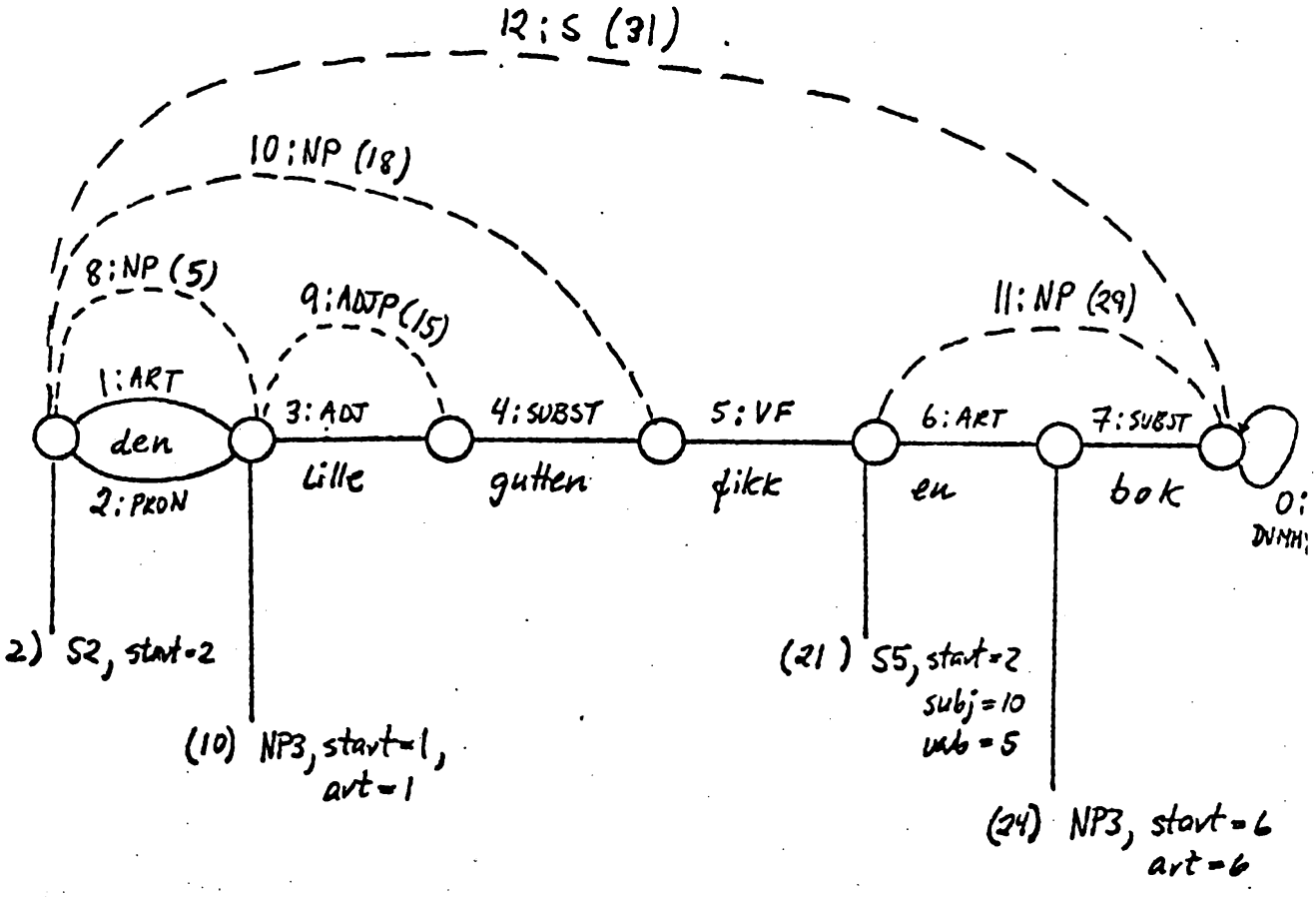
Lag en ny bue b_1 med ordklasseopplysning K og registerlisten R som innhold, fra knute K_1 (venstre knute til bue i registeret start) og til knute K_2 (venstre knute for bue B).

For hver tilstand t_j og registerliste r_j på ventelisten i K_1 lag en ny jobb:

b_1, t_j, r_j

Et eksempel.

Buene i kartet nummereres fortløpende. De heltrukne buene er kartet slik det var ved starten. I parentes står jobb-nummeret der buen ble laget eller der elementet ble satt på venteliste. Ventelisten henger under en knute. Den jobben som sist ble satt på jobblisten blir valgt som neste jobb.



jobb-nr.	produsert av jobb-nr.	utførelses-rekkefølge	bue	tilstand	registerliste
1 *)	start	7	1	S1	start=1
2	start	1	2	S1	start=2
3	2	6	2	S2	start=2
4	2	2	2	NP1	start=2
5	4	3	3	NP5	start=2, pron=2
6	5	4	8	S2	start=2
7	6	5	3	S3	start=2, subj=8
8	1	33	1	S2	start=1
9	1	8	1	NP1	start=1
10	9	9	3	NP2	start=1, art=1
11	10	31	3	NP3	start=1, art=1
12	10	10	3	ADJ1	start=3
13	12	11	3	ADJ2	start=3
14	13	12	4	ADJ2	start=3, overledd=3
15	14	13	4	ADJ3	start=3, overledd=3
16	15	14	9	NP3	start=1, art=1
17	16	15	4	NP4	start=1, art=1, adj=9
18	17	16	5	NP5	start=1, art=1, adj=9 subst=4
19	18	17	10	S2	start=2
20	19	18	5	S3	start=2, subj=10
21	20	19	6	S4	start=2, subj=10, verb=5
22	21	30	6	S5	start=2, subj=10, verb=5
23	21	20	6	NP1	start=6
24	23	21	7	NP2	start=6, art=6
25	21	24	7	NP3	start=6, art=6
26	21	22	7	ADJ1	start=7
27	26	23	7	ADJ2	start=7
28	25	25	7	NP4	start=6, art=6
29	28	26	0	NP5	start=6, art=6, subst=7
30	29	27	11	S5	start=2, subj=10, verb=5
31	30	28	0	S6	start=2, subj=10, verb=5 obj=11
32	31	29	12	S2	start=2
33	11	32	3	NP4	start=1, art=1

*) Her settes ikke nytt element på ventelisten til første knute fordi tilstanden er den samme og fordi registrene "start" inneholder buer som har samme venstre knute.

3. IMPLEMENTASJONEN I SIMULA

SIMULA er et programmeringsspråk med innebyggede funksjoner for listemanipulering. Det uferdige programmet fra Martin Kay's side var også skrevet i SIMULA. Dette er grunnene til at SIMULA ble valgt som programmeringsspråk.

Grammatikken beskrives direkte i SIMULA og oversettes sammen med resten av programmet. En tilstand i grammatikken er definert som en klasse i SIMULA. Dette er et sett med (data og) instruksjoner som gis et navn. En kan definere en forekomst av en klasse med navn og utføre denne ved å aktivisere forekomsten En forekomst kan videre aktiviseres og passiviseres. Først i alle tilstander er en ordre for passivisering av tilstanden (siden dette er felles for alle klassene gjøres dette i en prototyp som alle tilstandene defineres som underklasse av). Før analysen starter blir alle tilstandene aktivisert (for så å passiviseres med en gang).

Når en tar ut en jobb fra jobblisten, blir den aktuelle tilstand aktivisert. Instruksjonene for denne tilstanden lager nye jobber og når instruksjonene er utført, passiviseres tilstanden. Tilstanden i neste jobb som tas fra jobblisten blir så aktivisert.

Det er skrevet en del prosedyrer for å beskrive grammatikken i SIMULA. Her skal nevnes noen:

CE ("ordklasse")	boolsk funksjon som returnerer true dersom den aktuelle bue har samme ordklasse som angitt mellom anførselstegnene.
NEWREG("regnavn")	setter registeret "regnavn" til å peke på aktuell bue.
REG("regnavn")	gir oss bue som registeret "regnavn" peker på, er NONE dersom det ikke finnes et register med dette navn.
TEST("regnavn", "egenskap")	boolsk funksjon som er true dersom buen i registeret "regnavn" har den aktuelle egenskapen. "regnavn" kan være "CUR" og buen blir da den aktuelle bue .
MOVE(t)	godkjenner den aktuelle bue og lar neste tilstand være t . Går videre i kartet.
TRY(t)	blir stående på den aktuelle bue og lar neste tilstand være t.
NEWPATTERN(t_1,t_2)	lar neste tilstand være t_1 (start på et subnettverk). t_2 blir første tilstand etter en er ferdig med subnettverket .
BUILD("ordklasse")	lager en ny bue i kartet .
PLACE("regnavn")	lar buen som registeret "regnavn" inneholder, henge på den buen som skal bygges

SREG("regnavn","verdi")	lager et register som ikke inneholder en bue, men en tekstkonstant.
GREG("regnavn")	henter ut tekstkonstanten fra registeret.
NREG("regnavn1", "regnavn2")	lar et register med navn "regnavn1" inneholde samme bue som "regnavn2".

Grammatikken i eksempelet på side 4 vil nå se slik ut:

Alle tilstander har C foran navnet i definisjonen.
Tilstandene som utføres (forekomstene) har navn uten C og det er disse som bruke i MOVE, TRY, og NEWPATTERN.

```
tilstand class CS1;
newpattern (NP1,S2);

tilstand class CS2;
If CE("NP") then
begin
  newreg ("subj");
  move (S3);
end;

tilstand class CS3;
if CE ("VF") then
begin
  newreg ("verb");
end;

tilstand class CS4;
newpattern (NP1,S5);

tilstand class CS5;
if CE ("NP") then
begin
  newreg ("obj")
  move (S6);
end;

tilstand class CS6;
begin
  place ("subj");
  place ("verb");
  place ("obj");
  build ("S") ;
end;
```

```
tilstand class CNP1;
if CE("ART") then
begin
  newreg ("art");
  move (NP2);
end
else
if CE("PRON") then
begin
  newreg ("pron")
  move (NP5);
end
else
  try (NP2);

tilstand class CNP2;
newpattern (ADJ1,NP3);

tilstand class CNP2;
if CE ("ADJP") then
begin
  newreg ("adj");
  move (NP4);
end
else
  try (NP4);

tilstand class CNP4;
if CE ("SUBST") then
begin
  newreg ("subst");
  move (NP5);
end;
```

MOVE art pil

MOVE art pil

TRY pil

```
tilstand class CNP5;
begin
  place ("art");
  place ("adj");
  place ("subst");
  place ("pron");
  build ("NP");
end;
```

```
tilstand class CADJ1;
if CE ("GRADSADV") then
begin
  newreg ("Underledd");
  place ("underledd");
  move (ADJ2);
end
else
  try (ADJ2);
```

```
tilstand class CADJ2;
if CE ("ADJ") then
begin
  newreg ("overledd");
  place ("overledd");
  move (ADJ2);
end
else
if reg ("overledd") =/=NONE then
  try (ADJ3);
```

```
tilstand class CADJ3;
build ("ADJP");
```

```
ref (CS1) S1;
      "
      "          } Definisjon av forekomst,
      "            en for hver tilstand
      "
ref (CADJ3) ADJ3;
```

```
S1: - new (C1("S1",false));
          "
          "      } Aktivisering, en for
          "        hver tilstand
          "
ADJ3:- newreg(CADJ3("ADJ3",true));
```

Andre parameter er true dersom tilstanden inneholder en BUILD ordre (pil).

Før en slår inn en setning kan en angi hvor mye en vil ha skrevet ut mens analysen pågår. En kan her velge mellom ingen utskrift (bare den analyserte setningen) moderat utskrift og full utskrift. En utskrift som vist under der en får skrevet ut tilstandene og buene i den rekkefølge de sammenlignes, er til god hjelp ved analyse av feilanalyser.

```
S6A      30    NP-1     30    S7       9    ANCHOR   9
ADV1      9    ADV2      9    ADV4     9    ANCHOR   9
S10       9    ADV4A     9    S7A      9    S8       9
S9        9    S10       9    ANCHOR   9    ADJ1     9
ADJ2      9    S10A      9    S11      9    ANCHOR   9
NP1       9    NP2       9    ANCHOR   9    ADJ1     9
```

4. GRAMMATIKKEN

Programmet behandler jobblisten som en stack, dvs. den velger ut som neste jobb den siste som ble satt på jobblisten.

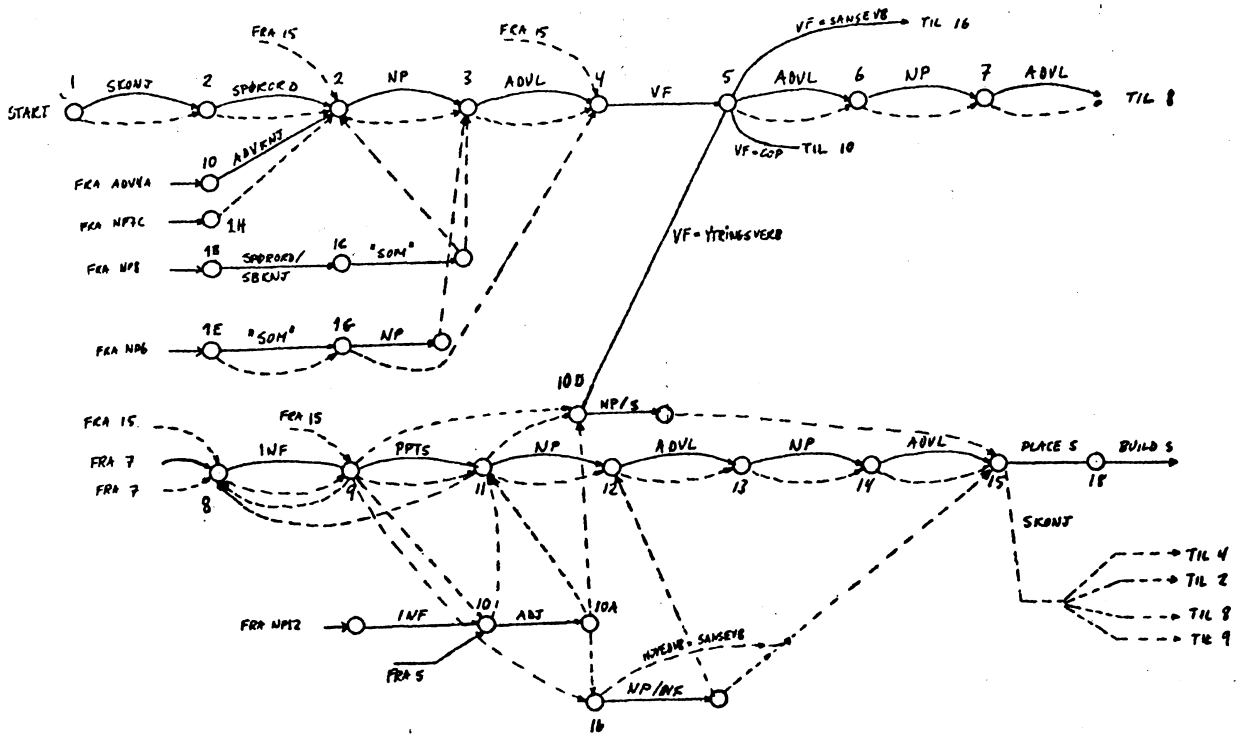
Dette forholdet har en utnyttet idet en har beskrevet grammatikken på en slik måte at en får først ut den mest vanlige tolkningen av en setning.

Grammatikken består av et hovednettverk for setning og 3 subnettverk, et for nominale ledd, et for adjektiviske ledd og et for adverbialer.
Fra NP nettverket eller adverbialnettverket går en igjen tilbake til setningsnettverket for å se etter substantiviske, adjektiviske leddsetninger og adverbiale leddsetninger.
Grammatikken kan behandle relativsetninger eller at-setninger uten som eller at. Det siste som er lagt inn er sideordning på alle nivåer i grammatikken.

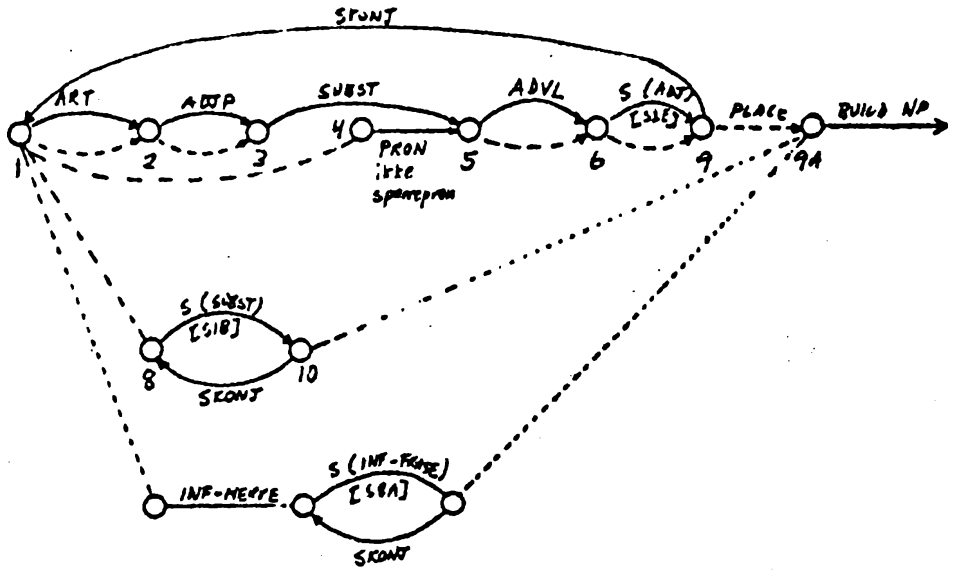
I setningsgrammatikken setter en først inn NP'ene i registre som har navn etter hvilket nummer NP'en har i setningen NP-1, NP-2 ..... Først når en skal bygge setningen blir NP'ene satt inn i registre for subjekt, objekt, predikat o.l.

På de neste sidene er de grammatiske nettverk litt forenklet. De prikkede linjene står for TRY piler. Piler som har NP, ADVL eller ADJP er en kombinasjon av en NEWPATTERN pil og en MOVE pil.

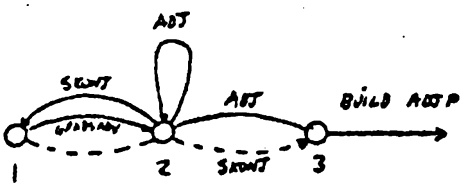
SETNINGSGRAMMATIKKEN (forenklet)



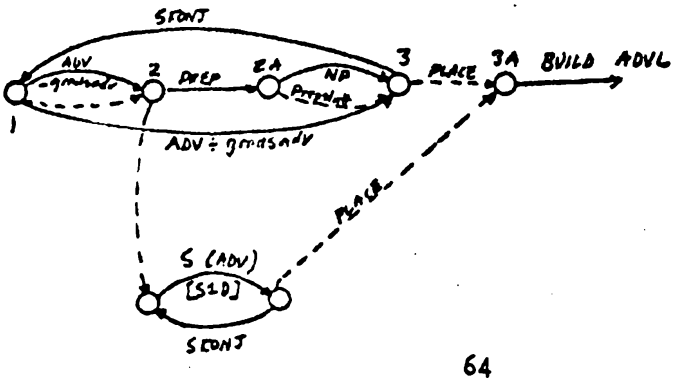
NP


SKONJ
ART
ADJP
SUBST
ADVL
S (ADJ)
[S2E]
PLACE
BUILD NP
1
2
3
4
PRON
irte
sparepron
5
6
9
9A
S (SUBST)
[S1B]
8
10
SKONJ
INF-MERKE
S (INF-FRASE)
[S8A]
SKONJ

ADJP


ADJ
SKONJ
ADJ
BUILD ADJP
1
2
SKONJ
3

ADVL


SKONJ
ADV
2
PREP
EA
NP
3
PLACE
3A
BUILD ADVL
1
ADV ÷ gmasadv
PLACE
S (ADV)
[S1D]
SKONJ