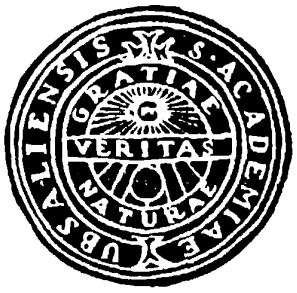
UCDL-R-84-1



FÖREDRAG

vid

De nordiska datalingvistik dagarna 1983

Uppsala den 3–4 oktober

Utgivare
Anna Sågvall Hein

Centrum för datorlingvistik,
Uppsala universitet

Förord

De Nordiska Datalingvistikdagarna utgör ett nordiskt forum för presentation och diskussion av datamaskinell metodik och tillämpning inom språkvetenskapen, initierat av Nordiska samarbetsgruppen för språklig databehandling. Den första sammankomsten ägde rum i Göteborg 1977 och därefter följde Köpenhamn 1979, Trondheim 1981 och Uppsala 1983. Nästa sammankomst i raden av Nordiska datalingvistikdagar kommer att äga rum 1985 i Helsingfors med Institutionen för allmän språkvetenskap som värd och arrangör.

I centrum för De Nordiska Datalingvistikdagarna i Uppsala 1983 stod datamaskinell morfologisk och syntaktisk analys (parsing). Ett flertal olika ansatser presenterades och diskuterades varigenom parsingproblematiken kom att få en mångfacetterad belysning. Den i föreliggande rapport presenterade samlingen föredrag från sammankomsten i Uppsala 1983 synes ge en god bild av det aktuella forskningsläget vad gäller parsing av naturligt språk i Norden.

Uppsala den 15 maj 1984

Anna Sågvall Hein

Innehåll

1. Beträffande denna presentation, se Ejerhed, E. & Church, K. (1983), Finite State Parsing, in F. Karlsson (ed), Papers from the Seventh Scandinavian Conference of Linguistics, Publ. 9, University of Helsinki, Department of Linguistics, pp 410-432.

Lars Ahrenberg
Institutionen för lingvistik
Uppsala universitet
Box 513
751 20 UPPSALA

DE GRAMMATISKA BESKRIVNINGARNA I SVE.UCP

1. Inledning. SVE.UCP är en parser för svenska som är under utveckling vid Uppsala Centrum för Datorlingvistik (UCDL). Grammatik och lexikon i SVE.UCP är proceduralt formulerade i en särskild formalism anpassad till parserns arbetande del, processorn, kallad UCP:n (the Uppsala Chart Processor)(1). Varje sträng som SVE.UCP accepterar ansätts en eller flera grammatiska beskrivningar. Ämnet för den här korta artikeln är just de grammatiska beskrivningarna, i synnerhet deras formella egenskaper och de möjligheter och begränsningar som ligger i dessa. Här och där kommer jag också att göra jämförelser med andra grammatiska beskrivningar av samma format, i första hand med de som används i Lexical Functional Grammar (LFG; Bresnan, 1982).

En grammatisk beskrivning kan utföras med olika syften. Den kan t ex utgöra ett steg på vägen i ett program som simulerar textförståelse eller vara ett datum i en textundersökning. Syftet bestämmer givetvis beskrivningens utseende till stor del. SVE.UCP utvecklas för närvarande inte med inriktning på någon speciell tillämpning, men är tänkt att vara användbar för bl a de nämnda syftena. Dess grammatiska beskrivningar är tämligen innehållsrika och omfattar både morfologi och syntax. Vad gäller syntaxen har utgångspunkten för beskrivningarna varit Teleman (1974) även om vi ibland har funnit det motiverat att avvika från hans analyser. I alla händelser är beskrivningarna innehållsligt sett mycket traditionella. i (1) visas den grammatiska beskrivning som SVE.UCP ger meningen Eva läser det(2).

2. Format och innehåll. SVE.UCP:s grammatiska beskrivningar kan enklast betraktas som en strukturerad mängd av påståenden. Ett påstående har formen av ett par vars första element vi kallar attribut och vars andra element kallas värde. Ett attribut är alltid en teckensträng medan ett värde kan vara en teckensträng, en grammatisk beskrivning eller en referens till ett annat värde. i (2)-(4) illustreras påståenden av alla tre typerna hämtade från (1).

```
(2)  NUMB = SING

(3)     OBJ = (PHR.CAT = NP
              NP.FEAT = (GENDER = NEUTR
                         NUMB = SING
                         PERS = 3
                         CASE = NIL(3)
                         LEX = !DEN)
              HEAD = (CHARS = (1 = D
                               2 = E
                               3 = T)
                       WORD.CAT = PRO
                       CASE = NIL))

(4)  SUBJ = (* SENT FUND)



(1)  EVA LÄSER DET.:

     (* = (SENT.CAT = SENT
           SENT.FEAT = (CPLX = SIMPLE)
           SENT = (PHR.CAT = CL
                   CL.FEAT = (TYPE = MAIN
                              MODE = DECL
                              TENSE = PRES)
                   FUND = (PHR.CAT = NP
                           NP.FEAT = (PROPR = +
                                      NUMB = SING
                                      GENDER = UTR
                                      CASE = NIL
                                      LEX = !EVA)
                           NAME1 = (WORD.CAT = NOUN
                                    NOUN.FEAT = (SEX = FEM
                                                 FIRST = +
                                                 CASE = NIL)
                                    STEM = (CHARS = (1 = E
                                                     2 = V
                                                     3 = A)
                                            MORPH.CAT = NOUNSTEM)))
                   PRED = (VP.FEAT = (FIN = +
                                      LEX = !LÄSA)
                           VFIN = (WORD.CAT = VERB
                                   VERB.FEAT = (FIN = +)
                                   STEM = (CHARS = (1 = L
                                                    2 = Ä
                                                    3 = S)
                                           MORPH.CAT = VERBSTEM
                                           CONJ = -ER)
                                   FLEX = (CHARS = (1 = E
                                                    2 = R)
                                           MORPH.CAT = VERB.FLEX
                                           CONJ = -ER))
                           OBJ = (PHR.CAT = NP
```

```
                    NP.FEAT = (GENDER = NEUTR
                               NUMB = SING
                               PERS = 3
                               CASE = NIL<5>
                               LEX = !DEN)
                    HEAD = (CHARS = (1 = D
                                     2 = E
                                     3 = T)
                                WORD.CAT = PRO
                                CASE = NIL)))
          SUBJ = (* SENT FUND))))
```

(2) kan utläsas "Numerus är singularis.", men det är då inte klart vilket elements numerus som avses. Detta framgår emellertid av den plats som påståendet befinner sig på i beskrivningen, närmare bestämt av den väg, eller följd av attribut som leder fram till det. Det finns två vägare genom (1) som leder till ett attribut/värde-par av typen (2), nämligen ⟨* SENT FUND.NP. FEAT⟩ och ⟨* SENT PRED OBJ NP.FEAT⟩. I det första fallet är det således fundamentets numerus som anges, i det andra objektets.

Uttryck av typen ⟨* SENT PRED OBJ NP.FEAT⟩ eller ⟨* SENT FUND⟩ som anger vägar genom en grammatisk beskrivning har själva värden. Deras värde är identiskt med det sista attributets värde. (4) anger alltså att värdet för attributet SUBJ är identiskt med värdet för attributet FUND, dvs att subjektet är identiskt med fundamentet i den givna meningen. Påståendet (3) ska tolkas så att objket är ett element med den grammatiska beskrivning som anges av uttrycket till höger om likhetstecknet. Vilket objket vi talar om framgår som förut av den väg som leder fram till beskrivningen, i detta fall ⟨* SENT PRED⟩. Märk att varje väg börjar i en och samma punkt, nämligen det särskilda attributet '*'. En grammtisk beskrivning för en given mening kommer alltid ut som värde till detta attribut.

För varje attribut i den grammtiska beskrivningen finns exakt en väg som förbinder det med *. Den grammatiska beskrivningen har således trädstruktur, med attributen som icke-terminala noder, * som rotnod och "enkla" värden, dvs teckensträngar och referenser som terminala noder. Till skillnad från gängse frasstrukturträd är dock döttrarna till en given nod inte ordnade i förhållande till varandra, vilket gör att den linjära ordningen mellan konstituenterna i en mening inte representeras strukturellt. Hur man kan göra istället ska jag återkomma till nedan.

De grammatiska beskrivningarna i SVE.UCP lyder under ytterligare ett villkor. Det är att inget attribut får ha mer än ett värde eller, om vi tänker i trädstrukturer, att om ett attribut (en nod) förgrenar sig så är alla döttrar skilda attribut. Detta villkor ska vi kalla unicitetsprincipen[4].

Trädstrukturen hos de grammatiska beskrivningarna gör det möjligt och naturligt att representera hierarkiska relationer i meningen strukturellt. Satsled representeras av delträd. (Omvändningen gäller dock inte, ett delträd behöver inte alltid svara mot ett satsled - se nedan). Avsaknaden av linjär ordning mellan beskrivningens delar medför att satsleden primärt beskrivs utifrån sina grammatiska funktioner (som subjekt, adverbial, etc) men dessutom ges uppgift om följande fyra egenskaper:

1/ Position i satsen,
2/ Morfosyntaktisk kategori,
3/ Grammatiska egenskaper (förutom kategori),
4/ Delkonstituenter.

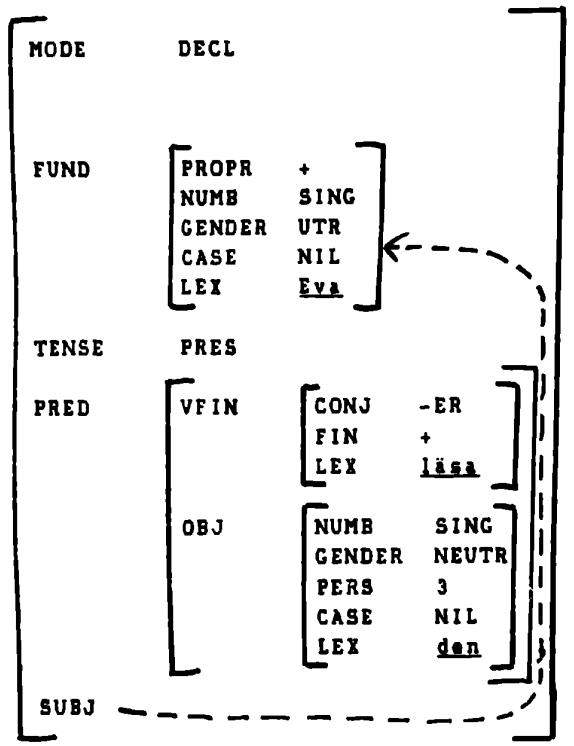
Position är inte direkt kodifierat i (1). Istället finns ett attribut CHARS, som anges för de minsta leden (morferna) och som anger den teckensträng som svarar mot dessa. För den nänskliga användaren är dock detta i de flesta fall tillräckligt för att man ska kunna avgöra vad som blivit analyserat som vad i beskrivningen. I (1) finns fyra förekomster av CHARS svarande mot de fyra morfer som strängen analyserats i.

Attribut som anger morfosyntaktisk katekori har alla sufficet CAT. Vi skiljer på fraskategorier (NP, PP, ADJP, ...), ordkategorier (NOUN, PREP, ADJ, ...) och morfkategorier (NOUNSTEM, ADJSTEM, VERBFLEX, ...). Attribut som anger grammatiska särdrag är samlade under ett attribut med suffixet FEAT och ett förled som utgörs av ett kategorinamn. Om fraskategorin för ett led är NP samlas således dess övriga egenskaper upp i en beskrivning under attributet NP.FEAT, så som i (4). Attribut som anger delkonstituenter har inga särskilda kännetecken man ges så långt möjlitgt namn som antyder konstituenternas grammatiska funktion. På satsnivå har vi således attribut som SUBJ, PRED, OBJ och ADV.LOC, på frasnivå DET, MOD, HEAD, etc och på ordnivå STEM och FLEX. Attribut som gäller ett och samma led skrivs under varandra i en kolumn och ett värde skrivs alltid till höger om sitt attribut.

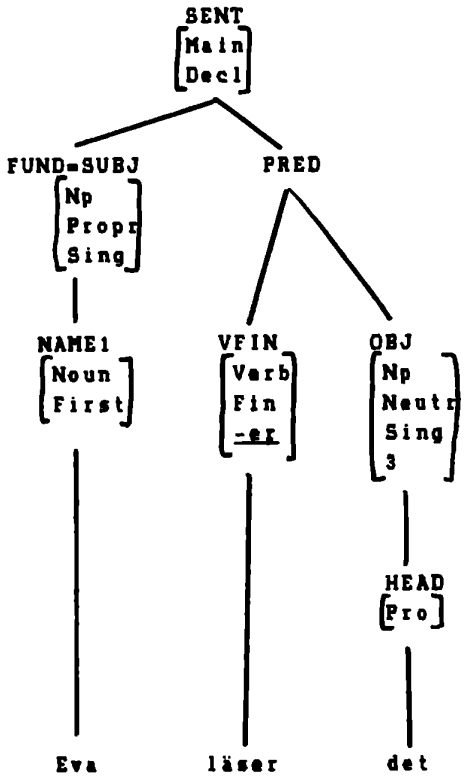
Formatet på SVE.UCP:s grammatiska beskrivningar representerar en utveckling av Kays (1977) "structures", som också är oordnade träd begränsade av unicitetsprincipen. Användningen av oordnade träd som representationsformat för grammatisk struktur har varit sparsam inom modern teoretisk linvistik, som i stället föredragit ordnade träd(5). Det ökade intresset för grammatiska funktioner har emellertid lett till att oordnade träd börjat användas alltmer. De s k f-strukturer som används inom LFG har precis samma format men något annorlunda restriktioner på vad som kan vara ett möjligt värde. Kays (1982)

Functional Descriptions är också väsentligen oordnade träd. Det kan sägas att SVE.UCP:s utskriftsformat inte är det mest transparenta för att återge trädstruktur. En lådstruktur av den typ som används för LFG:s f-strukturer, illustrerad i (5), eller en graf av typen (6) som för skillnad mellan olika typer av attribut är åskådligare, men helt ekvivalenta representationsformer.

(5)



(6)



3. Unicitetsprincipen. Unicitetsprincipen har en komputationell motivering men också en lingvistisk. Den komputationella motiveringen är att unifieringsoperationen som bygger upp grammatiska beskrivningar inte vore väldefinierad annars. Den lingvistiska är att det helt enkelt normalt verkar vara så att paradigmatiska val bara görs en gång inom en given överordnad struktur. Många villkor som formulerats för lingvistisk struktur på olika beskrivningsnivåer kan ses som specialfall av denna princip. En av de mera kända är väl kasusgrammatikens One-case-per-clause-principle (Fillmore, 1968:21). I s k Relationell Grammatik finns motsvarande villkor formulerat för grammatiska funktioner (Johnson, 1977:155, Perlmutter, 1980:211). Inom X-barteori anses varje fras ha ett entydigt bestämt huvud (Jackendoff, 1977:53) och ibland begränsas dess "specificerare" explicit till högst en av varje slag (ibid. 104). Men det finns många fall där man inte gärna kan, eller vill, upprätthålla den. Det gäller i första hand samordnade konstruktioner, som i (7) och (8),

för dels är det omotiverat att ge de olika leden olika funktion, dels finns det ingen gräns för hur många de kan vara. Det finns också andra fall där det är rimligt att säga att vi har att göra med flera element av samma funktion, t ex vid sekvenser av satsadverbial som i (9), eller vid sekvenser av namn, som i (10).Perifera adverbial, som i (11), kan också i princip vara obegränsat många och man undviker många problem om man låter bli att skilja på dem.

(7) Man säljer ägg, potatis, grönsaker, blommor och mycket annat där.
(8) Han har en fin, billig, liten lägenhet.

(9) De har nog tyvärr antagligen redan glömt det.
(10) Tager du, Karl Teodor Erik Jakob Olsson, denna Eva Margareta ...

(11) Igår åt jag middag med Svante på Wermlandskällaren.

Det finns olika sätt att komma runt unicitetsprincipen. I LFG tillåter man mängder av f-strukturer som värden. F-strukturen för (11) kommer således att innehålla ett attribut ADJUNCTS vars värde är en mängd av tre f-strukturer, var och en svarande mot ett av adverbialen. I UCP:n används en speciell variabel över attributnamn, kallad :NEW, vid uppbyggnaden av beskrivningar som kan innehålla ett uppräkneligt antal led av samma slag. :NEW instansieras som heltalet '1' första gången det ges ett värde i en viss regel och därefter som ett en enhet större heltal då det på nytt tilldelas ett värde av samma regel. Attributen under CHARS i (1) illustrerar en användning av :NEW-variabeln. En annan illustreras i (12), som förenklat visar hur samordningen i (7) beskrivs av SVE.UCP.

```
(12)   FUND = (PHR.CAT = NP
                NP.FEAT = (COORD = +)
                1 = ägg
                2 = potatis
                3 = grönsaker
                4 = blommor
              CONJ = och
                5 = mycket annat
```

Med detta sätt att beskriva samordningar och liknande sekvenser blir den linjära ordningen mellan de ingående leden också inkodad. Denna information är väsentlig om man använder beskrivningen som utgångspunkt för semantisk

tolkning (något som i LFG anses vara f-strukturens viktigaste uppgift eller om den används för generering. Anaforiska pronomen tolkas gärna olika beroende på var i en samordning de uppträder, vilket (13) och (14) illustrerar, och ordningsföljden mellan adverbial kan vara avgörande för deras relativa "scope". (15) har två tolkningar, men (16) har bara en.

(13) Olle och hans fru var här igår.
(14) Hans fru och Olle var här igår.

(15) På grund av kylan gick han inte hem.
(16) Inte gick han hem på grund av kylan (i alla fall).

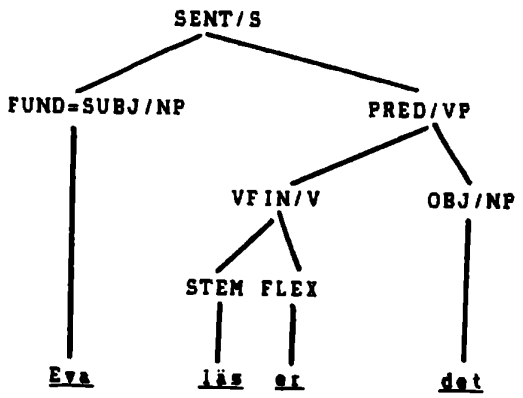
4. Frasstruktur. Eftersom SVE.UCP:s grammatiska beskrivningar inte återger linjär ordning strukturellt kan de naturligtvis inte heller återge frasstruktur strukturellt. Däremot kan frasstruktur beskrivas på annat sätt. Vi kan införa ett attribut ORDER vars värde ger information om ordningsföljden mellan delkonstituenter. Ett problem är då att vi inte vet vilka attribut som representerar delkonstituenter och vilka som representerar annan information om konstituenten. Men detta kan antingen specificeras globalt eller genom att vi känner igen delkonstituenter på att de har ett värde som anger morfosyntaktisk kategori.

Med införandet av ORDER måste vi också införa en ny typ av värden. I det enklaste fallet är värdet en ordnad lista av de attributnamn (vägar) som representerar delkonstituenter, svarande mot en viss permutation av dem. (1) kan t ex kompletteras med följande enkla beskrivningar för att vi ska ha en fullständig specifikation av frasstrukturen:

under SENT: ORDER = ⟨FUND, PRED⟩
under PRED: ORDER = ⟨VFIN, OBJ⟩
under VFIN: ORDER = ⟨STEM, FLEX⟩

Vi antar då att konstitutenter med multipla funktioner ordnas via det attribut som har deras grammatiska beskrivning till värde, så att t ex SUBJ i (1) inte behöver ordnas explicit eftersom dess värde är en referens till FUND. Den information som då finns under ORDER och CHARS svarar precis mot den som ges av frasstrukturträdet (17), som vi för tydlighets skull också kan etikettera med morfosyntaktiska kategorier.

(17)



I det allmänna fallet finns emellertid ingen perfekt korrespondens mellan funktionella enheter och fraser, i alla fall om de senare uppfattas som sammanhängande. Många funktionella enheter är diskontinuerliga. Vi har inte bara fall av extraherade led utan också många fall av mera lokala diskontinuiteter. (18)-(22) visar några olika typer. De extraherade leden ger egentligen minst problem eftersom de ordnas efter sina tematiska funktioner, dvs som fundament, fokus etc. För beskrivningen av de andra fallen kan vi införa variabler av olika slag (se Kay, 1981). Ta (19) som exempel. På satsnivå kan vi anta att vi analyserat (19) i ett tidsadverbial (då), ett predikat (gav upp), ett subjekt (han) och

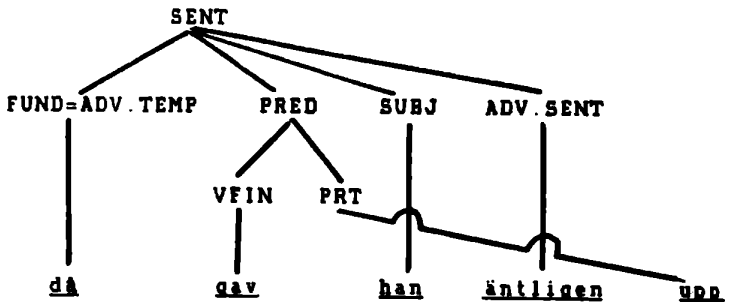
(18)	Glasögonen brukar hon glömma var hon har lagt.	(EXTRAHERADE LED)
(19)	Då gav han äntligen upp.	(VERB + PARTIKEL)
(20)	En bättre åkare än Stenmark finns inte.	(KOMPARATIVT ATTRIBUT)
(21)	Jag såg en duva igår som var alldeles vit.	(KORRELAT + RELATIV)
(22)	Nästa år, sägs det, kommer kungen att abdikera.	(PARENTETISKT INSKOTT)

ett satsadverbial (äntligen) och att predikatet analyserats i ett finit verb och partikel. De ordningsbeskrivningar vi behöver kan ges följande form, där C är en variabel över precis en konstituent, X en variabel över en godtycklig

under SENT: ORDER = ⟨# FUND C SUBJ ADV.SENT X⟩
under PRED: ORDER = ⟨# C VFIN X PRT⟩

sekvens av konstituenter, och # en satsgräns. Dessa två ordningsbeskrivningar tillsammans specificerar den givna ordningen mellan konstituenterna och ingen annan, men ger inte ett ordnat frasstrukturträd av gängse typ utan ett träd av utseendet (23). Det talar om för oss precis vilka funktionella enheter som identifieras och hur de fördelar sig på inputsträngen. Denna typ av träd är inte så vanlig som representationsformat, kanske för att de gör ett osnyggt intryck. men de kan väl motiveras. Deras formella egenskaper är heller inte svårare att specificera än ordnade träds[6].

(23)



Givet den stora användning som frasstrukturträd fått i modern lingvistik är det kanske frestande att anta att meningars inneboende struktur helt enkelt är "frasstrukturell" och att varje mening måste tilldelas (minst) en frasstruktur. Detta förefaller att vara den generativa grammatikens position, utan att man därför besvärat sig så mycket att argumentera för den[7]. Typiskt är då att man försöker återinföra beskrivningen av grammatiska relationer på frasstrukturkonfigurationer. Också LFG, som annars kraftigt tar avstånd från frasstrukturträd som generellt beskrivningsformat, behåller en nivå av kontextfritt genererbar frasstruktur, så att varje mening grammatisk beskrivs med två strukturer, en frasstruktur (eller c-struktur som man föredrar att kalla det) och en funktionell struktur, f-strukturen. Varför är det då enligt LFG nödvändigt med en c-struktur, som explicit sägs vara helt irrelevant för semantisk tolkning? Man ger två huvudskäl; det ena är att en sådan struktur behövs som input till den fonologiska komponenten, det andra är att den jämte lexikoninformation behövs som input vid härledningen av f-strukturer. Det första skälet ska jag inte beröra, men det andra som i princip är Aspectteorins ide att frasstrukturella konfigurationer är definierade för grammatiska relationer (Chomsky, 1965:68 f). Så associerar t ex Kaplan & Bresnan (1982:184) med frasstrukturrelgeln (24) en tilldelning av subjektsfunktionen till NP.

(24) S → NP VP

Dock skiljer sig LFG:s användning av dessa konfigurationella definitioner från Aspectsteorins på några väsentliga punkter:

1/ De är språkspecifika. Subjekt, objekt osv är universella primitiva begrepp som inte kan reduceras till frasstrukturkonfigurationer, men som i vissa språk uttrycks konfigurationellt, i andra kanske morfologiskt;
2/ De hänför sig direkt till det språkliga uttrycket (ytstrukturen);

3/ Varje instans av en grammatisk funktion behöver inte uttryckas på detta sätt. Den kan t ex vara given lexikalt.

I och med att den tredje möjligheten finns uppstår frågan om när vi har att göra med t ex ett konfigurationellt givet subjekt och när vi har ett subjekt som är givet på annat sätt. LFG:s svar tycks vara att vi har det förra fallet dels när subjektet står där det ska stå enligt definitionen (naturligt nog) eller då det extraherats ut ur sin sats som i en fråga eller en relativsats. I s k kontrollstrukturer får däremot komplementets verb sitt subjekt från det överordnade verbets lexikoninformation. Lova har således en specifikation som talar om att subjektet i komplementsbisatsen är identiskt med dess eget i en mening som (25).

(25) Pelle lovade komma.
(26) Det läser Eva.

Skälet till att extraherade led ges en grammatisk funktion via ett tomt element i en definierande position är att det finns strukturella villkor på när sådana extraktioner är möjliga (Ross, 1969). Men det finns också många andra villkor på dem, t ex av pragmatisk art (Allwood, 1982: Engdahl, 1982) utan att det anses nödvändigt att upprätta en särskild pragmatisk representation som i härledningen föregår f-strukturen. Överhuvudtaget gäller för bestämningen av grammatiska funktioner att annan information än den rent morfosyntaktiska måste komma med. Meningen (26) ges av SVE.UCP bara en beskrivning och de strukturellt möjliga alternativen slås ut på ett tidigt stadium när verbets lexikala egenskaper jämförs med de båda NP:nas lexikala egenskaper. En sekvens NP - V - NP kan åsättas flera olika c-strukturer, t ex de i (27), svarande mot olika f-strukturer, men det finns såvitt jag kan se ingen anledning att generera alla dessa i sin helhet för att först i efterhand slå ut flertalet av dem på grundval av lexikal information, vilket i princip är den procedur som LFG använder sig av. (28) är ett annat exempel som lätt leder till en explosion av c-strukturer och f-strukturer men som tillåter en (eller två) tolkningar i ett textsammanhang.

(27)a.	NP - VP(V - NP)	SUBJ - PRED (VERB - OBJ)	/Erik badar Lisa/
	alt	SUBJ - PRED (VERB - PF)	/Erik badar en glad gosse/
b.	NP - S(V - NP - e)	FUND=OBJ - VFIN - SUBJ	/Henne badar Erik/
c.	NP - VP(V) - NP	SUBJ - PRED (VERB) - ADV	/Erik badar varje dag/
d.	NP - S(V - NP)	FUND=ADV - PRED - SUBJ	/Varje dag badar Erik/

(28) Eva och Erik Lind bor en trappa upp i ett gult hus vid Stortorget i Nora.

En annan fråga är om strukturella villkor måste uttryckas i termer av frasstrukturkonfigurationer. Den enda funktion av dem som vi hittills diskuterat som kan ges en entydig frasstrukturdefinition är fundamentet medan däremot både subjekt och objekt kräver flera olika. Om vi däremot uttrycket strukturella positioner i termer av fält (Didrichsen, 1957) och fältgränser kan vi få mera användbara identifikationskriterier, förslagsvis

1/ SUBJEKT är ett NP, eller ett S i nexusfältet eller i fundamentställning, som står i nominativ eller i omarkerat kasus och som uppfyller de lexikala villkor som verbet ställer:
2/ OBJEKT är ett NP, eller ett S i verbalfältet, eller i fundamentställning, som har ackusativt eller omarkerat kasus som uppfyller de lexikala villkor som verbet ställer.

Dessa definitioner är inte så enkla som de frasstrukturella, men har fördelen att vara direkt tillämpbara på språkliga data. De är också möjliga att formalisera; i SVE.UCP finns två regler som bestämmer subjekt i en sats. Den ena, som utnyttjar kategoriell och morfologisk information, anropas i huvudsatsregeln efter det att det finita verbet och ett eventuellt satsadverb identifierats (i nexusfältet, om vi så vill; (Jf Sågvall Hein, 1983b), och plockar upp ett subjekt i den positionen eller från fundamentet. Den andra, som kontrollerar att det sålunda identifierade subjektet har vissa särdrag, anropas från den regel som söker efter verbets argument och som hämtas ur lexikonet. Denna verbregel svarar också för objektssökning och kan på samma sätt lägga semantiska villkor på detta.

NOTER

⟨1⟩ För information om UCP:n och UCP-formalismen se Sågvall Hein (1980, 1983a) samt Carlsson (1982).

⟨2⟩ (1) är inte den fullständiga beskrivningen utan en något förkortad version. Skillnaden är dock betydelselös för den här framställningens syften.

⟨3⟩ CASE = NIL innebär att ledets kasus är omarkerat. LEX = !DEN är en hänvisning till ett lexem, vars egenskaper är rudimentärt specificerade i en särskild lexembas.

⟨4⟩ Jf Kapland & Bresnan, 1982:181.

⟨5⟩ Syntaktiska analysträd av den typ som används i Montaguegrammatik är dock undantag.

⟨6⟩ Se t ex Karlgren, 1972 och McCawley, 1982, för argument och formaliseringar av sådana alternativa representationsformer.

⟨7⟩ Chomsky & Miller (1963:288-89) skriver: "We assume that such a tree graph must be a part of the structural description of any sentence; we refer to it as a phrase-marker (P-marker). A grammar must, for adequacy, provide a P-marker for each sentence." Se också Chomsky (1965:123f). För några fördelar med frasstrukturrepresentation, se Gazdar (1982).

REFERENSER

Allwood, J. (1982) The Complex NP Constraint in Swedish. I Engdahl & Ejerhed (1982) s 15-32.

Bresnan, J. (1982) (utg.) The Mental Representation of Grammatical Relations, Cambridge, Mass. The MIT Press.

Carlson, M. (1982) Uppsala Chart Parser 2, System Documentaion. Uppsala, UCDL.

Chomsky, N. (1965) Aspects of the Theory of Syntax. Cambridge, Mass. The MIT Press.

Chomsky, N. & Miller, G. (1963) Introduction to the Formal Analysis of Natural Language. I Luce, Bush & Galanter (utg.) Handbook of Mathematical Psychology s. 269-322.

Diderichsen, P. (1957) Elementaer dansk grammatik, 2 upp,. Köpenhamn, Gyldendal.

Engdahl, E. (1982) Restrictions on Unbounded Dependencies in Swedish. I Engdahl & Ejerhed (1982) s. 151-174.

Engdahl, E. & Ejerhed, E. (1982) (utg.) Readings on Unbounded Dependencies in Scandinavian Languages, Umeå, Almqvist & Wiksell international.

Gazdar, G. (1982) Phrase Structure Grammar, I P. Jacobson & G. K. Pullum, (utg.), The Nature of Syntactic Representaion, Reidel, Dordrecht.

Jackendoff, R. (1977) X-bar Syntax, Cambridge, Mass. The MIT Press.

Johnson, D. (1977) On Relational Constraints on Grammars. I Syntax & Semantics, Vol 8, s. 151-177.

Karlgren, H. (1972) Why Trees in Syntax? Interim Report No. 41, KVAL, Stockholm.

Kaplan, R. & Bresnan, J. (1982) Grammatical Representation. I Bresnan, 1982.

Kay, M. (1977) Reversible Grammar: Summary of the Formalism. Xerox Research Center, Palo Alto.

Kay, M. (1981) An Algorithm for Compiling Parsing Tables from a Grammar. Xerox Research Center, Palo Alto.

McCawley, J. D. (1982) Parentheticals and Discontinous Constituent Structure. Linguistic Inquiry, Vol 13, s. 91-106.

Perlmutter, D. M. (1980) Relational Grammar. I Syntax & Semantics, Vol 13, s. 195-230, New York, Academic Press.

Ross, J. R. (1967) Constraints on Variables in Syntax. Doctoral dissertation, MIT, Cambridge, Mass.

Sågvall-Hein, A-L. (1980) An Overview of the Uppsala Chart Parser Version 1. Uppsala, UCDL.

Sågvall-Hein, A-L. (1983a) A Parser for Swedish. Status Report for SVE.UCP, February, 1983. Uppsala, UCDL.

Sågvall-Hein, A-L. (1983b) Regelaktivering i en parser för svenska (SVE.UCP). I denna volym.

Teleman, U. (1974) Manual för grammatisk beskrivning av talad och skriven svenska. Lund, Studentlitteratur.

Sture Allén

INTE BARA IDIOM

Detta skall handla om kollokationer eller ordförbindelser. Vissa grundfrågor inom språkvetenskapen upphör inte att fängsla en, och kollokationerna har för mig varit en sådan grundfråga. Bland mycket annat har de ett intressant förhållande till parsning.

Jag minns med stor intensitet mina intryck, när jag såg de första resultaten av konkordanskörningar för ganska många år sedan. De öppnade nya språkliga vyer. Man såg i ett huj vilken betydelse kollokationerna måste ha i den språkliga aktiviteten. Detta har varit en viktig utgångspunkt för ständigt uppdaterade tankar kring kollokationer.

En av de första jag stötte på som över huvud taget hade tänkt i de här banorna med datamaskinen i perspektivet var John Sinclair (1970). Han menade att man skulle ta fasta på vad som är statistiskt signifikant för att få fram kollokationerna. De ord som uppträdde tillsammans oftare än man hade anledning att förvänta sig, sett mot en slumpmässig bakgrund, skulle ha kollokationell karaktär. Kriteriet gav både relevanta och irrelevanta (*if take* osv.) förbindelser. Resultatet av arbetet var således inte övertygande i detta avseende, inte heller för Sinclair själv. Också andra experiment har sedan visat att det naturligtvis inte är så enkelt, även om det är en del av sanningen.

De första tankarna med utgångspunkt i en allmän uppfattning om språket och om vad konkordanserna visade ledde till uppläggningen av arbetet på tredje delen av Nusvensk frekvensordbok (1975) och presentationen av resultaten. Jag talade under 1970-talets första hälft bland annat i Åbo, Pisa och London om kollokationerna och deras roll i språket (föredragen trycktes 1973, 1976 respektive 1977).

Det som var utgångspunkten för tredje delen av frekvensordboken var iakttagelsen att kollokationerna ständigt återkommer,

är rekurrenta i den mening som anges i inledningen till frekvensordboken. Villkoret i denna undersökning var att det skulle finnas minst två identiska belägg på en miljon löpande ord. Med det villkoret fick vi ut 660 000 belägg på ordkombinationer. Det var kanske den första överväldigande siffran för oss. Den vanligaste av alla förbindelserna var *det är*, sedan kom *och den*, *sig i*, *än att*, *på ett helt annat sätt* osv., alltså välformade och icke välformade ordkombinationer om vartannat.

Ur de 660 000 beläggen analyserade vi fram vad vi kallade för konstruktioner. På dem lade vi villkoren att de skulle vara grammatiskt styrda och lexikaliskt selekterade. De utgör sålunda ett urval ur kombinationerna. Vi fick 50 000 olika konstruktioner, som vi klassade i 17 olika huvudtyper: *den stora frågan*, *i linje med*, *ta form*, *sköta om*, *kommer att fortsätta*, *för att undersöka*, *är på väg*, *mycket ung*, *även om*, *om också*, *för att*, *in i*, *men låt oss inte gå händelserna i förväg*, *det är givet att*, *kort sagt*, *som människa*, *jo då*. Dessa exempel markerar de olika typerna.

Ur konstruktionerna gjorde vi i sin tur ett urval. Det gällde idiomen i ordets snäva bemärkelse, dvs. de kollokationer som har oförutsägbar betydelse sett från de ingående ordens betydelsers synpunkt. Vi tillämpade kriterierna ganska strängt och fick fram 300 olika idiom: *ge sig i kast med*, *i elfte timmen*, *lägga sista handen vid*, *det är inte utan att*, *göra avkall på*, *slå dövörat till* osv. Idiomen utgör alltså en mycket liten del av den stora mängden kollokationer.

I den nämnda inledningen påpekar jag, att det är ganska tydligt att resultaten bör få konsekvenser för språkteorin. Lexikonet ser antagligen inte ut på det sätt som man har tänkt sig tidigare, t. ex. inom den generativa inriktningen. Vad man kan kalla lexikaliska block av många olika slag bör finnas i språkmedvetandet.

Strax efter utgivningen av frekvensordbokens tredje del publicerade Jan Anward och Per Linell (1976) en uppsats om lexikaliserade fraser i svenskan. De tog fonologi, grammatiska egenskaper m.m. i betraktande och sammanfattade sitt resonemang så

här: varje enskild konstituent i en lexikalisk enhet (dvs. i en kollokation) kan inte exploatera sin betydelse fullt ut och kan inte syntaktiskt varieras i någon större utsträckning; kan inte ha egen referens (säger man *sparka boll*, syftar man inte på en speciell boll); kan inte böjas fritt; kan inte modifieras fritt; kan inte fritt befrågas, negeras eller affirmeras; kan inte varieras med avseende på ordföljd och prosodi i någon större utsträckning; kan å andra sidan utmärkas av morfologiska och andra oregelbundenheter.

Om man tittar lite närmare på detta, ser man att flera av villkoren är av sådan karaktär att de karakteriserar vad jag vill kalla idiom snarare än kollokationer i allmänhet. Men slutsatsen är i alla fall densamma som den jag hade dragit från andra utgångspunkter, nämligen att lexikonet spelar en större roll än man kanske trott och att grammatiken spelar en något mindre roll.

Charles Ruhl och Adam Makkai förde en diskussion om idiom i en volym från 1976. Utgångspunkten var att Makkai hade skrivit en bok om Idiom structure in English. Ruhl menade att ett fel med den boken var att Makkai inte hade tagit hänsyn till den omedvetna delen av språkmanifestationen. Han tog som exempel *take off* som ju på engelska betyder 'become airborne' men som i sin nominaliserade form också kan betyda 'parody'. Makkai menade att man här har två olika lexikaliska enheter medan Ruhl efter en genomgång av ett hundratal belägg på *take off* noterar hur man från den ena betydelsen gradvis kommer över i den andra.

På detta svarar Makkai genom att ta fram alla ord som i engelskan slutar på *-ic(k)* (*bic*, *brick*, *chick*, *dick* osv.) och tillsammans med sina studenter resonera sig fram till hur vart och ett kan anknytas till dels 'liv' eller 'död', dels 'skatter'. Med andra ord: från vad som helst kan man resonera sig fram till vad som helst. Det är ett sätt att ironiskt avfärda den tanke som Ruhl hade. Man måste försöka identifiera de olika lexikaliska enheterna, menar Makkai. Han använder rentav benämningen *institutions* för vissa av de här kollokationerna – de har så att säga en stadfäst roll i språket.

Ungefär samtidigt hade vi i Göteborg ett mindre projekt som hette Algoritmisk textanalys. Jag förbigår här det som gjordes på programsidan och i någon mån på syntaxsidan och nämner de lexikaliska delresultaten. Vi utarbetade ett paradigmmärkt baslexikon på ungefär 8000 enheter. Det är publicerat av Staffan Hellberg i en bok från 1978. Baslexikonet upptar stam, uppslagsform och paradigmnummer och täcker därmed i princip hela morfologin. Vidare upprättades ett speciallexikon över heterografer (ickehomografer) på ungefär 900 enheter. Ett tredje resultat var ett speciallexikon över disambiguerande kollokationer på omkring 1600 enheter. Det omfattar sådana kollokationer som innehåller homografer som blir disambiguerade genom att ingå i kollokationerna. Ett exempel är *komma hem*, där *komma* kan vara verb eller substantiv och *hem* kan vara substantiv eller adverb, medan *komma hem* är entydigt. Till detta kom ett ändelselexikon för sannolikhetsklassificering av ord som saknas i baslexikonet. Ett av de ständigt återkommande problemen är ju att man stöter på nya ord.

Bland det mest intressanta var att de två speciallexikonen på 1600 respektive 900 enheter vid körningar visade sig täcka 50 procent av en okänd text av normal typ. Det är tankeväckande från parsningens synpunkt. Jag kan tillägga att de 8000 enheterna i baslexikonet täcker i runda tal 90 % av en text. Då skall vi emellertid komma ihåg att de innefattar en mängd homografer. Det fina med de två speciallexikonen är att de ger stycken av fast mark som analysen kan bygga vidare på.

Låt mig nämna ytterligare några som på olika sätt har arbetat med kollokationer. En av dem är Harald Burger som har publicerat en intressant bok om idiom (1973), där han framför allt är inne på de teoretiska problem som knyter sig till begreppet. Maurice Gross (1982) har undersökt vad han kallar "frozen sentences", vilket också är ett slags kollokationsbegrepp. I hans fall har det gällt franska. Göran Kjellmer vid engelska institutionen i Göteborg är sysselsatt med en genomgång av hela Brown-korpusen för att ta fram kollokationsmaterialet ur den. Syftet är främst att utarbeta en frasordbok.

Så är vi framme vid Lexikalisk databas. Det är det största projektet vid Språkdata för närvarande. I det definierar vi omkring 75 000 lemman ur det moderna svenska språket och ger uppgifter av många olika slag, bl.a. just beträffande fraseologi och idiomatik.

Här är ett utdrag ur kollokationsuppgifterna rörande lemmat *land*, som representerar tre olika lexem (lexikaliska enheter baserade på kärnbetydelser). Jag ger inte deras definitioner, utan vi kan se på exemplen vad de avser. Till det första lexemet hör *inom landets gränser, flytta till ett annat land, de afrikanska länderna* och idiomen *det heliga landet* eller *det förlovade landet* och *skuggornas land*. Det andra omfattar *en sjöman går i land, land i sikte, på torra land, färdas till lands* och idiomen *förstå hur landet ligger, gå i land med något* och *nu går skam på torra land*. Det tredje har *hon var från landet, resa till landet under veckoslutet* och idiomet *ingen mans land*. En av tankarna med projektet Lexikalisk databas är just att man med utgångspunkt från i första hand kollokationerna och definitionerna skall arbeta vidare i riktning mot ett lexikon för parsning. I existerande system är de såvitt bekant, liksom lexikaliska uppgifter över huvud, svagt företrädda. Låt mig ändå i detta sammanhang nämna Kaplan & Bresnan (1980), Sager (1981) och Zimmermann, Kroupa & Keil (1983).

På senare tid har jag kommit att uppmärksamma en annan typ av indikationer som jag anser vara viktig från såväl teoretisk som praktisk och psykologisk synpunkt. Det gäller vad jag kallar för de metaspråkliga kommentarerna i texter. Det är alltså så att språkbrukarna själva i viss utsträckning talar om hur deras lexikon ser ut, något som man kanske inte har uppmärksammat tidigare. De metaspråkliga kommentarerna gäller rätt ofta enskilda ord men inte sällan just kollokationer. Man markerar dem med uttryck av typen *som det heter, som det så vackert heter* osv. Ett litet urval exempel följer: *ett rörligt intellekt, som det heter; karavanen rör sig trots att hundarna skäller, som det heter i ett gammalt arabiskt ordstäv; administrativ databehandling, som det heter; ett förslag till, som det så vackert heter, en förenklad deklarationsblankett; bidrag för att,*

som det heter, förbättra konkurrensmöjligheterna på den internationella marknaden; hon är väl död vid det här laget, om man så säger; vi nådde alltså, för att lätt travestera ett slitet uttryck, ända fram.

Själva de metaspråkliga uttrycken är rikt varierade och baserar sig på en rad olika verb: *för att använda en kliché, för att använda ett gammalt ordspråk, för att använda ett slitet uttryck, för att använda herr NNs egen formulering; för att citera NN; som det heter, som det numera heter, som det brukar heta, som det så vackert heter; som man brukar kalla det, som det kallas, så kallad; som frasen lyder, som uttrycket lyder; som man säger, så att säga, om jag så säger, som det brukar sägas; för att travestera ett gammalt uttryck; om man så vill; om uttrycket tillåts, om man så får uttrycka saken, som NN uttrycker det.* Alla de metaspråkliga kommentarerna syftar inte alltid på kollokationer i den mening jag avser här, men mycket ofta visar det sig vara fallet. Man kan dela in dem i några huvudtyper. *Som det heter* syftar ofta på en kurant kollokation. *Som X lyder* syftar gärna på ett ordspråk eller ordstäv. *För att citera* anger direkt källan: en författare, en lagtext, en förordning eller någonting sådant. *För att travestera* kan i princip syfta på alla de olika typerna. Som travesti åberopar den indirekt en kollokation av något slag.

De metaspråkliga kommentarer som man kan plocka fram på det här sättet ger ytterligare en inblick i människans lexikon. De fogar sig också till de tidigare typerna av kollokationella kriterier som har varit rekurrensen, de grammatiska konstruktionskriterierna, idiomkriteriet och de tillkommande lingvistiska kriterier som Anward och Linell har pekat på. De bidrar alltså till att ge en ny bild av lexikonet och följaktligen också en ny bild av hur vi fungerar i språkproduktionen och i perceptionen. Därmed är de av fundamentalt intresse vid utvecklingen av parsningsystem.

Litteratur

Allén, Sture, Om fraser i svenskan. (Svenskans beskrivning 7. Ed. Christer Hummelstedt. Åbo 1973, s. 24-31.)

Allén, Sture, On phraseology in lexicology. (Cahiers de lexicologie 29 (1976), s. 83-90.)

Allén, Sture, Text-based lexicography and algorithmic text analysis. (ALLC Bulletin 5: 2 (1977), s. 126-131.)

Allén, Sture, et al., Nusvensk frekvensordbok baserad på tidningstext. 3. Ordförbindelser. 1975.

Anward, Jan, & Linell, Per, Om lexikaliserade fraser i svenskan. (Nysvenska Studier 55-56 (1975-76), s. 77-119.)

Burger, Harald, Idiomatik des Deutschen. Tübingen 1973. (Germanistische Arbeitshefte. Ed. Otmar Werner & Franz Hundsnurscher. 16.)

Gross, Maurice, Simple sentences. (Text processing. Proceedings of Nobel Symposium 51. Ed. Sture Allén. 1982, s. 297-315.)

Hellberg, Staffan, The morphology of Present-Day Swedish. Word-inflection, word-formation, basic dictionary. 1978.

Kaplan, R.M., & Bresnan, J.W., Lexical-functional grammar: a formal system for grammatical representation. Occasional Paper 13. MIT Center for Cognitive Science. Cambridge, Mass. 1980.

Makkai, Adam, Idioms, psychology, and the lexemic principle. (The Third Lacus Forum. Ed. Robert J. Di Pietro & Edward L. Blansitt, Jr. Columbia, South Carol. 1976, s. 467-478.)

Ruhl, Charles, Idioms and data. (The Third Lacus Forum. Ed. Robert J. Di Pietro & Edward L. Blansitt, Jr. Columbia, South Carol. 1976, s. 456-466.)

Sager, Naomi, Natural language information processing: a computer grammar of English and its applications. Reading, MA 1981.

Sinclair, J. McH., Jones, S., & Daley, R., English lexical studies. Department of English. Birmingham 1970.

Zimmermann, Harald H., Kroupa, Edith, & Keil, Gerald, CTX. Ein Verfahren zur computergestützten Texterschliessung. Saarbrücken 1983.

Bengt Beckman
Vitklövervägen 38
16360 SPÅNGA

UTNYTTJANDE AV ORDKLASSER FÖR FÖRFATTARBESTÄMNING

Hösten 1974 dök det i Paris upp en sensationell skrift i vilken 1965 års nobelpristagare Michail Sjolochov anklagades för plagiat. Studien hade skrivits av en senare avliden sovjetisk kritiker D, och förordet hade skrivits av Alexander Solzjenitsyn, som helt stödde slutsatsen: det mesta av Stilla flyter Don har inte skrivits av Michail Sjolochov utan av en annan kosackförfattare, Fedor Krjukov.

Detta är bakgrunden till att det hösten 1975 bildades ett svenskt-norskt team som tog sig an frågan: "Vem har skrivit Stilla flyter Don?" Deltagare var Sven Gustavsson och Bengt Beckman från Sverige och Geir Kjetsaa och Steinar Gil från Norge. Avsikten med projektet var dels att lösa författarproblemet, dels att testa kvantitativa metoder för stilanalys och författarskapsundersökningar.

Undersökningen är nu - efter några avbrott - slutförd och resultaten redovisas i bokform.[1] Föredraget behandlar en av de studier som gjorts.

Studien utgör en utvidgning av en metod som beskrivits vid två tidigare tillfällen, dels vid ett låtsat författarbestämningsförfarande på material av de sovjetiska författarna K Simonov och K Paustovskij, dels vid de preliminära undersökningarna i det mera spektakulära fallet med Krjukov och Sjolochov som möjliga författare till Stilla flyter Don.

Det kriterium som användes i det första fallet var ordklassfördelning i meningsbörjan och meningsslut. I Geir Kjetsaas pilotstudie "Storms on the Quiet Don" undersöktes Sjolochov, Krjukov och "Stilla flyter Don" ur bl a denna aspekt. Resultaten, som klart talar för Sjolochov, har presenterats i andra sammanhang och skall ej beröras här. Emellertid bekräftade undersökningen ordklasskriteriets användbarhet. Jag har därför i den större, datorstödda undersökningen i samma ämne, vars resultat nu föreligger, använt ordklasskombinationer totalt och inom hela meningar som kriterium.

---

[1] Geir Kjetsaa, Sven Gustavsson, Bengt Beckman, Steinar Gil.
The Authorship of The Quiet Don. Slavica Norvegica II. Oslo 1983.

Ordklassindelningen i nämnda mindre undersökningar gjordes manuellt. Detta är - frånsett det besvärliga i arbetet - otillfredsställande ur den synpunkten att objektivitet och konsekvens ej kan garanteras. När Sven Gustavsson tog initiativet till projektet "Vem har skrivit Stilla flyter Don?" fanns ett automatiskt analyssystem uppbyggt vid Uppsala Datacentral under ledning av Anna Sågvall Hein. Systemet delar in orden i morfologiska klasser efter böjningsändelser och andra formella kännetecken. Detta resulterar i en modifierad ordklassindelning, som får fördelen att vara konsekvent, vilket ju i detta sammanhang är betydligt viktigare än att den är mänskligt rätt. Systemet har modifierats och utökats för att passa den aktuella undersökningen. Sålunda har klassindelningen genom att tillvarata underklasser ökats till att omfatta 25 klasser. Systemet utför en klassbestämning av varje ord och översätter varje mening till en svit klassbetecknande bokstäver. (Fig 1, Fig 2)

Sammanlagt har 176 500 ord fördelade på 12 400 meningar skrivits in till analyssystemet via terminaler. Till systemet hör ett stamlexikon. Varje nytt ord som tillförs lexikonet måste förses med en serie kriterier för att ordet när det nästa gång dyker upp i någon annan böjningsform skall kunna identifieras automatiskt. Denna uppdatering av lexikonet har blivit en omfattande uppgift. Som tidigare nämnts modifierades systemet och många tilläggsprogram skrevs. Detta arbete skedde vid Uppsala Datacentral och - på senare tid - vid Uppsala universitet, Centrum för datorlingvistik.

På materialet i klassbetecknad form har gjorts ett flertal statistiska undersökningar. Alla resultat tyder på samma sak. Ordklassfördelningen skiljer sig från Krjukovs men är lik Sjolochovs. Som exempel har tagits 4-klasskombinationer. I Fig 3 anges under H_0 de förväntade frekvenserna framräknade ur äkta Krjukov-material och under H_1 motsvarande ur Sjolochov-material. Under X_1 återfinns de i det omstridda verket observerade frekvenserna.

Den svaghet som vidlåder de flesta metoder är, att också det omstridda materialet måste vara ganska stort för att duga till statistiska jämförelser. Vad man eftersträvar är ett test, genom vilket man kan utläsa trolig författartillhörighet på en mindre mängd omstritt material, t ex 20-30 meningar. Detta test bör vara så beskaffat, att man - mening för mening - tillvaratar alla i meningen ingående syntaktiska två-kombinationer och jämför med sannolikheter framräknade ur resp författares oomstridda verk. Genom att multiplicera sannolikheterna i varje mening får man ett mått på hur bra den är som - i detta fallet - Sjolochovmening resp Krjukovmening. Två test enligt denna modell har prövats på det aktuella materialet.

Fig 1. Klassindelning och klassbeteckningar.

A	Nominatives of animates	Basic class 1
B	Other cases of animates	
C	Nominatives of inanimates	
D	Other cases of inanimates	
E	Nominatives	Basic class 2
F	Other forms	
G	Long forms, nominative	Basic class 3
H	Long forms, other cases	
I	Short forms, comparative forms like jasnee	
J	Nominatives of kto, čto, my, vy, ja, ty	Basic class 4
K	Other forms of pronouns mentioned in J	
L	Other members of basic class 4	
M	All forms	Basic class 5
N	Pure prepositions like bez, bezo, v, vo	Basic class 6
O	Prepositions which can also be adverbs, for example vblizi, vdol', vnutri	
P	Pure conjunctions like a, i, ili	
Q	Conjunctions which can also be adverbs, for example da, odnako, poka	
R	Other members of basic class 6	
S	Infinitives	Basic class 7
T	Gerunds	
U	Participles, long forms, nominative	
V	Participles, long forms, other cases	
W	Participles, short forms	
X	Other forms belonging to basic class 7	
Y	All punctuation marks other than sentence-dividing marks	

Fig 2. Meningar översatta till klassbetecknande bokstäver samt de ursprungliga meningarna.

```
 6  G C Y N H D C
15  C N F C X N C N D
45  G G C N H N C H D Y P R C Y G C D Y G
    G C V D C P I Y U N D H D C D
70  N C Y N D H D Y G C Y G C Y U H D G Y
    G C Y C N D Y N F Y U H D C
75  N D Y G C C
```

1. Melechovskij dvor - na samom kraju chutora.
2. Vorotca so skotin'ego baza vedut na sever k Donu.
3. Krutoj vos'misažennyj spusk meždu zamšelych v prozeleni melovych glyb, i vot bèreg: perlamutrovaja rossyp' rakušek, seraja izlomistaja kajma nacelovannoj volnami gal'ki i dal'še - perekipajuščee pod vetrom voronenoj rjab'ju strem'ja Dona.
4. Na vostok, za krasnotalom gumennych pletnej, - getmanskij šljach, polynnaja prosed', istoptannyj konskimi kopytami buryj, živuščoj pridorožnik, časovenka na razvilke; za nej - zadernutaja tekučim marevom step'.
5. S juga - melovaja chrebtina gory.

Fig 3. Tetragramfrekvenser.
H_0=förväntade frekvenser framräknade ur Krjukov-material.
H_1=förväntade frekvenser framräknade ur Sjolochov-material.
X_1=obseverade frekvenser i Stilla flyter Don (Tichij Don = TD)

TD total (n = 44,408)

TETRA	H_0 expected	H_1 expected	X_1 observed
CYXN	38.63	99.03	104
DYNH	83.04	45.74	50
DYXN	50.18	127.45	195
GYGC	81.27	19.54	23
GYGY	91.92	19.54	24
HDYN	71.50	48.40	50
HYHD	95.92	23.09	25
NCYX	45.30	92.37	76
NDYX	83.04	124.34	149
NFDY	63.94	35.52	38
NHDX	51.07	72.83	67
RXND	39.52	58.62	29
XNCY	59.95	133.22	111
XNDY	115.90	166.53	166
YGYG	82.15	18.65	26
YNHD	96.81	64.39	96
YXNC	41.74	129.23	118
YXND	64.84	130.12	196
YXYX	69.72	35.97	42

Test I. Vi antar att en text ordklassmässigt är en Markovkedja, d v s att ett ords ordklasstillhörighet har en sannolikhetsfördelning som enbart beror av föregående ords ordklass. Vi talar om övergångssannolikheter p(i, j), d v s sannolikheten att ett ord i ordklass i följs av ett ord i ordklass j. Dessa övergångssannolikheter är positionsberoende. Speciellt stort är positionsberoendet i början och slutet av meningen. Dessutom inför vi sannolikheten P_i att en mening börjar med ordklass i. Alla dessa sannolikheter bestäms genom statistik som upprättats på okänt material. Vi får då olika uppsättningar sannolikheter för olika författare. Våra testvariabler blir således (meningen antas vara $x_1\ x_2\ x_3 \ldots x_n$)

$$P_S = P^S_{x_1} \prod_{i=1}^{N-1} P^S_i (x_i, x_{i+1})$$

$$P_k = P^K_{x_1} \prod_{i=1}^{N-1} P^k_i (x_i, x_{i+1}) \qquad \text{TEST I}$$

Test II. Ett problem vid testvalet är: hur bra är approximationen att vi har Markovkedjor? I varje fall inte perfekt. Som alternativt test har därför använts följande. Låt $q^s_k(i, j)$ vara sannolikheten att bigrammet på plats k och k+1 är ett ord i ordklassen i och ett i ordklassen j (hos författaren S).

$$P_S = \prod_{i=1}^{N-1} q^S_i (x_i, x_{i+1})$$

$$P_K = \prod_{i=1}^{N-1} q^K_i (x_i, x_{i+1}) \qquad \text{TEST II}$$

Detta test (Test II) ger högre värden där författaren använt för sig vanliga ordklasser. Något överraskande ger test II bättre resultat än test I när det prövats på samma S- och K-texter.

Testen applicerades på 10-meningsavsnitt ur Stilla flyter Don (Tichij Don= TD). Resultaten exemplifieras av fig 4.

Fig 4.

TD,1

Number of sentences		Word number	Number of words	Test I Sentences of type S	Test I Sentences of type K	Test II Sentences of type S	Test II Sentences of type K	I Predominantly	II Predominantly
1	10	1-123	123	6	4	8	2	S	S
2	10	801-914	114	7	3	10	0	S	S
3	10	1384-1510	127	2	8	7	3	K	S
4	10	2001-2079	79	9	1	9	1	S	S
5	10	2573-2661	89	6	4	8	2	S	S
6	10	3148-3212	65	5	5	7	3	X	S
7	10	3671-3855	185	9	1	9	1	S	S
8	10	4398-4498	101	5	5	7	3	X	S
9	10	5012-5092	81	5	5	8	2	X	S
10	10	5630-5744	145	7	3	8	2	S	S
11	10	6415-6565	151	6	4	5	5	S	X
12	10	6916-6987	72	7	3	9	1	S	S
13	10	7363-7479	117	7	3	8	2	S	S
14	10	7991-8075	85	4	6	7	3	K	S
15	10	8654-8777	124	8	2	9	1	S	S
16	10	9231-9305	74	4	6	8	2	K	S
17	10	9911-10115	204	6	4	6	4	S	S
18	10	10915-11063	149	7	3	7	3	S	S
19	10	11573-11695	123	6	4	10	0	S	S
20	10	12320-12384	65	5	5	8	2	X	S

Sammanfattning av resultat

10-meningsavsnitt.

Stickprov ur äkta Sjolochov- och Krjukov-texter:

	Test I			Test II		
	S	K	X	S	K	X
Äkta S	12	5	6	22	1	0
Äkta K	0	22	1	3	17	3

Stickprov ur Stilla flyter Don (Tichij Don):

	Test I			Test II		
	S	K	X	S	K	X
TD 1	13	3	4	19	0	1
TD 2	13	4	3	19	0	1
TD 3	15	6	3	22	0	2
TDtot	41	13	10	60	0	4

Under S noteras antal 10-meningsavsnitt med dominans av S-meningar.
Under K noteras antal 10-meningsavsnitt med dominans av K-meningar.
Under X noteras antal 10-meningsavsnitt där 5 meningar är av S-typ och 5 är av K-typ.

Anm. Flera skäl kan tänkas till testets goda diskriminerande förmåga, t ex:
- de 25 klasserna är en lämplig abstraktionsnivå för ord
- kombinationer av två klasser är en nivå på vilken individuella vanor klart framträder
- meningskonstruktion, i det avseende som testet avspeglar, är en djupt rotad vana, som inte förändras vare sig av ämnesområde eller författarens utveckling.

Arendse Bernth
Datalogisk Inst. Københavns Univ.
Sigurdsgade 41, DK-2200 København N

LOGIK ANVENDT TIL OVERSÆTTELSE AF JAPANSK.

På Datalogisk Institut ved Københavns Universitet eksperimenteres for tiden med at anvende logikprogrammering som hjælpemiddel ved automatiseret oversættelse fra japansk til engelsk og dansk. Logikprogrammeringssproget Prolog anvendes til syntaksanalyse af japansk, opbygning af en prædikatlogisk repræsentation af teksten, fortolkning af denne og ved hjælp af ordbøger oversættelse til engelsk eller dansk.

1. Bemærkninger om det japanske sprog.

Japansk er ikke beslægtet med noget andet sprog (måske bortset fra koreansk), og det har sin egen måde at beskrive grammatik på.
Indo-europæiske begreber passer i virkeligheden ikke særligt godt til japansk. På trods af dette vil beskrivelsen af det japanske sprog anvende indo-europæisk terminologi, da det formodes, at læseren er mest fortrolig med denne. Der er altså tale om et vist misforhold mellem beskrive-måde og det beskrevne. Endvidere er der af fremstillingsmæsssige grunde foretaget nogle simplificeringer, f.eks. i omtalen af partikler.

1.1 Sætningskonstruktion.

Principielt er enhver ordrækkefølge tilladt, blot verbet kommer til sidst. I de allerfleste tilfælde vil man dog se denne rækkefølge:

subjekt indirekte-objekt objekt verbum.

Subjektet udelades som regel, hvis det fremgår af sammenhængen. Kasus angives ved efterstillede partikler, også kaldet postpositioner. Blandt de vigtigste partikler kan nævnes:

"wa" el. "ga" (subjektspartikel),
"ni" (indir. objektspartikel),
"o" (objektspartikel).

Der er dog en vis forskel på "wa" og "ga", som det vil føre for vidt at komme ind på her.
Enhver underordnet sætning skal komme før den tilhørende hovedsætning. Da verbet altid står sidst i en sætning, vil verbet i en relativ sætning stå umiddelbart foran det substantiv, som det er relativt til. Der findes ikke relative pronominer på japansk.

1.2 Verber og adjektiver.

På japansk er der kun ringe forskel på verber og adjektiver. Verber og adjektiver bøjes ikke i køn, tal eller kasus, derimod bøjes begge i tid.

Verber (og adjektiver) bruges ofte uden et eksplicit subjekt. Subjektet afgøres da af sammenhængen. Både et verbum og et adjektiv kan stå alene og danne en sætning.

Der findes reelt kun to tider: præsens og præteritum. Derimod findes der mange former, der angiver stemninger, troværdighed o. a. Endvidere findes der forskellige grader af høflighed. Foruden specielt høfligt sprog findes der til hvert verbum/adjektiv en kort (mindre høflig) og en længere (mere høflig) form. Brugen af de lange og korte former afhænger af omstændighederne, men en hovedregel er, at man i almindeligt, neutralt talesprog anvender den korte form i underordnede sætninger, og den lange i hovedsætninger. I skriftsprog anvendes for det meste udelukkende de korte former.

1.3 Eksempler på nogle japanske sætninger.

Eksempel I:	Eksempel II:
人は花を見る	花を見る人は笑う
i kunreisiki-transkription:	
hito wa hana o miru	hana o miru hito wa warau
'subj.par' obj.par	
menneske blomst ser	blomst ser menneske smiler
(et menneske ser en blomst)	(et menneske, der ser en blomst, smiler)

Prædikatlogisk repræsentation for:

Eksempel I:

```
exists(X,(hito(X) & exists(Y,(hana(Y) & present(miru(X,Y))))))
```

Eksempel II:

```
exists(X,(hito(X) & (exists(Y,(hana(Y) & present(miru(X,Y)))))
                  & present(warau(X)))).
```

2. Prolog og sprogbehandling.

Prolog er baseret på en delmængde af 1.ordens prædikatkalkule (såkaldte definitte klausuler).
En definit klausul er et udtryk på formen:

C0 <- C1,C2,...,Cn

hvor C0 er konklusionen og de øvrige Cer er betingelser. Ci har formen:

p(t1,t2,...,tk)

hvor p er et prædikat og ti termer.
Prolog kan dels opfattes som et problembeskrivelsesværktøj (det deklarative aspekt) og dels som et program (det procedurale aspekt).
Logikprogrammering er deskriptiv i modsætning til traditionelle programmeringssprog, der er preskriptive.
Dette betyder, at har man beskrevet et problem, har man -stort set- også løst det. Dette giver sig tydeligt udtryk i syntaksanalyseopgaver. En grammatik udtrykt i traditionel BNF-notation skrives meget nemt om til Prolog -og så får man i tilgift et program, der kan foretage syntaksanalyse.
Denne Prolog-grammatik kan ved indføjelse af nogle ekstra variable returnere et resultat af syntaksanalysen. Vi udfører forsøg med at lade en Prolog-grammatik danne en prædikatlogisk model af forskellige sætningstyper. Denne logiske model forsøges anvendt som mellemsprog ved automatiseret oversættelse.
I et Prolog-program lægger man sig ikke på forhånd fast på forudsætninger og resultater. Dette skyldes, at Prolog er deskriptiv: programmet beskriver relationen mellem ind- og uddata, men fortæller ikke noget om, hvad der er forudsætninger (inddata) og hvad der er resultater (uddata).
Dette betyder, at et program kan bruges "begge veje", f.eks. kan det samme program både differentiere og integrere. På denne måde kan man oversætte fra et sprog til et andet ved hjælp af et analyseprogram for hvert sprog, samt en ordbog.
Dette princip forsøger vi at anvende ved oversættelse fra japansk til

dansk.
F. eks. kan man som inddata til den japanske del give:

    hito wa hana o miru

Dette giver som resultat den logiske repræsentation som beskrevet i afsnit 1.3. Lader man den danske del (incl. ordbog) behandle dette, fås:

    et menneske ser en blomst

2.1 Eksempel på japansk grammatik.

Udtrykt i en BNF notation:

```
<japsentence>       ::= <subjectnounphrase><verbphrase>

<subjectnounphrase>::= <relclause><nounphrase>
                       <subjectparticle><determiner>

<objectnounphrase>  ::= <relclause><nounphrase>
                       <objectparticle><determiner>

<relclause>         ::= E | <verbphrase>

<nounphrase>        ::= <noun>
<noun>              ::= hito | hana
<subjectparticle>   ::= wa | ga
<objectparticle>    ::= o
<determiner>        ::= E | daredemo

<verbphrase>        ::= <intransitiveverb> |
                       <objectnounphrase><transitiveverb>

<intransitiveverb> ::= warau
<transitiveverb>    ::= miru
```

E angiver den tomme streng.

Den samme grammatik udtrykt i Prolog (variable begynder med stort):

```
japsentence(S,T)        <- subjectnounphrase(S,U),
                           verbphrase(U,T).

subjectnounphrase(S,T) <- relclause(S,U),
                           nounphrase(U,V),
                           subjectpar(V,W),
                           determiner(W,T).

objectnounphrase(S,T)  <- relclause(S,U),
                           nounphrase(U,V),
                           objectpar(V,W),
                           determiner(W,T).

relclause(S,S).
relclause(S,T)  <- verbphrase(S,T).
nounphrase(S,T) <- noun(S,T).

subjectpar(S,T) <- c(wa,S,T).
subjectpar(S,T) <- c(ga,S,T).
objectpar(S,T)  <- c(o,S,T).

determiner(S,S).
determiner(S,T) <- c(daredemo,S,T).

verbphrase(S,T) <- intransverb(S,T).
verbphrase(S,T) <- objectnounphrase(S,U),
                   transverb(U,T).

intransverb(S,T) <- c(warau,S,T).
transverb(S,T)   <- c(miru,S,T).

noun(S,T)        <- c(hito,S,T).
noun(S,T)        <- c(hana,S,T).

c(Word,[Word|S],S).
```

'S','T','U','V',og 'W' fungerer som en slags pegepinde i sætningen.
Den første Prolog-regel siger (fortolket deklarativt):
"en 'japsentence' består af en 'subjectnounphrase' efterfulgt af en 'verbphrase'"
eller (fortolket proceduralt):
"der er en 'japsentence' fra S til T, hvis der er en 'subjectnounphrase' fra S til U og en 'verbphrase' fra U til T".
c(hito,S,T) betyder "ordet 'hito' befinder sig i positionen mellem S og T".
Man kan nu spørge:

```
japsentence([hito,ga,warau],[]).
```

Programmet vil så undersøge, om der er tale om en lovlig japansk sætning.

2.2 Semantik.

Ovenstående program kan kun foretage syntaksanalyse og svare ja/nej. For at gemme et semantisk resultat er det nødvendigt at indføje yderligere variable. Som eksempel på denne udvidelse kan vi vise, hvorledes reglerne for 'japsentence', 'noun' og 'determiner' kommer til at se ud:

```
japsentence(X,Logic,S,T) <- subjectnounphrase(X,Vp,Subj,S,U),
                            verbphrase(X,Vp,Logic,S,T).

noun(X,hito(X),S,T) <- c(hito,S,T).

determiner(X,P1,P2,all(X,(P1=>P2)),S,T) <- c(daredemo,S,T).

determiner(X,P1,P2,exists(X,(P1&P2)),S,S).
```

Variablen 'X' angiver det, vi fokuserer på, og 'Logic' returnerer den logiske repræsentation af sætningen.
Reglen for substantiver knytter et substantiv til det fokus ('X'), vi har. Således er semantikken for 'hito' ('mand'):

```
hito(X)
```

Som det ses, opbygges selve den logiske struktur i reglerne for 'determiner'.

Første regel for 'determiner' tager sig af 'daredemo' ('enhver'). Semantikken for 'enhver' bliver et alkvantoriseret udtryk:

```
all(X,(a(X) => b(X)))
```

Anden regel tager sig af 'en'/'et'. Japansk har hverken bestemte eller ubestemte artikler, ejheller bøjes substantiver i tal. I dette eksempel regnes et japansk substantiv for singularis.
Semantikken for 'en'/'et' bliver et eksistenskvantoriseret udtryk:

```
exists(X,(a(X) & b(X)))
```

3. Konklusion.

Som det fremgår af ovenstående eksempel er det relativt enkelt at lave et analyse-program i Prolog, når man har gjort sig grammatikken klar. Et af formålene med dette skrift er at gøre opmærksom på dette. Endvidere kunne man tænke sig en analysator for hvert af mange forskellige sprog; man kunne så -i hvert fald i teorien- foretage oversættelse mellem vilkårlige sprog ved hjælp af disse programmer. Dette skyldes det fælles mellemsprog.
Det skal dog nævnes, at Prolog har visse begrænsninger f.eks. med hensyn til klausulerne og den rækkefølge, de bliver udført i. Dette medfører, at det deklarative aspekt ikke er 100% løsrevet fra det procedurale aspekt. Denne begrænsning ligger ikke i logikprogrammering som sådan, men i det på nuværende tidspunkt mest tilgængelige logikprogrammeringssprog, nemlig Prolog. I det japanske femte generationsprojekt arbejdes der på en højere grad af parallellitet for at afhjælpe problemet med udførelsesrækkefølgen.
Det egentlige problem består i at finde et godt mellemsprog. Her er anvendt en prædikatlogisk model baseret på hypoteserne fremstillet i Pereira(1980).
Det er klart, at det er nemmere at lave en god repræsentation af en tekst, jo mere præcist og entydigt sproget er. Japansk er et meget vagt og upræcist sprog; faktisk bryder man sig slet ikke om at udtrykke sig præcist for ikke at støde modparten. Som eksempel kan nævnes, at substantiver, verber og lign. ikke bøjes i tal, og at subjektet for det meste udelades.
Man må gætte sig til meningen udfra sammenhængen. Det er klart, at disse ting vanskeliggør en automatisk oversættelse betydeligt. Det har bestemt heller ikke været muligt at tage højde for alt; formålet er også snarere at afprøve nogle teknikker på nogle begrænsede dele

af sprogene.
Endvidere er det naturligvis essentielt for en god oversættelse, at systemet har adgang til en beskrivelse af et relevant verdensbillede, hvilket også ville hjælpe til at fastlægge meningen. Dette er et ikke uvæsentligt problem, som vi dog ikke beskæftiger os med.

4. Litteratur.

Clocksin,W.F. & C.S. Mellish(1981): "Programming In Prolog". Springer.

Colmerauer,A.(1982): "An Interesting Subset of Natural Language". i Clark,K. & S-A. Taernlund (eds.): "Logic Programming". Akad.Press.

Dahl,V.(1979): "Logical Design of Deductive Natural Language Consultable Data Bases." Proceedings of Very Large Data Bases.

Koch,G.(1981): "Grammars and Predicate Calculus". Diku Report 81/16, Datalogisk Inst. ved Københavns Universitet.

Koch,G.(1983): "Stepwise Development of Logic Programmed Software Development Methods". Diku Report 83/5, Datalogisk Inst. ved Københavns Universitet.

Kowalski,R.A.(1979): "Logic for Problem Solving". North-Holland.

Miller,R.A.(1967): "The Japanese Language". University of Chigago.

Pereira,F.C.N. & D.H.D Warren(1980): "Definite Clause Grammars for Language Analysis a Survey of the Formalism And a Comparison with Augmented Transition Networks". Artificial Intellingence 13,3.

Lars Borin
Centrum för datorlingvistik
Sturegatan 13B[V]
752 23 UPPSALA

ETT TEXTDATABASSYSTEM FÖR LINGVISTER

1 Bakgrund

Under en längre tid har Centrum för datorlingvistik utvecklat och tillhandahållit en rad programpaket för textbearbetning i olika former. Dessa programpaket har använts och används under samlingsnamnet TEXTPACK (Nurmi & Rosén 1983; Rosén 1983; Rosén & Sjöberg 1983) av språkvetare från olika språkinstitutioner, främst vid Uppsala Universitet.

Till de språk som i skrivande stund datorbearbetas med de olika TEXTPACK-rutinerna hör bl.a. engelska, finska, persiska, ryska, sanskrit och svenska, med större eller mindre problem vad gäller inmatning, teckenrepresentation, kollatering och utskrift.

De dataformat TEXTPACKsystemet arbetar med är i hög grad standardiserade, men de accessmetoder det använder är avhängiga av det operativsystem vi arbetar under (IBM MVS, se (PL181b)), och de accessoperatorer programmeringsspråket tillhandahåller (PL/1, se (PL181a)). För att komma åt detta maskinberoende har vi vid Centrum för datorlingvistik under ca. två och ett halvt års tid sysslat med att utveckla ett mer generellt textbearbetningssystem.

2 Förutsättningar och problem

Vid utvecklingsarbetet har vi utgått ifrån att datoranvändarna vid universiteten av ekonomiska skäl under ganska lång tid framåt kommer att vara hänvisade till de typer av datorer och in- och utmatningsutrustning vi har idag, med de problem detta innebär för den språkvetenskapliga databehandlingen. Några sådana problem är:

Vid inmatning kommer den normala utrustningen att bestå av en vanlig bildskärmsterminal med svenskt tangentbord[1]. För de västeuropeiska språkens del kan man räkna med att i högre grad än hittills kunna utnyttja sättband från datorsättningsutrustningar[2], men när det gäller andra språk, kan möjligheterna i det avseendet vara betydligt mer begränsade av olika skäl (låg datoriseringsgrad, trög byråkrati osv.).

Ett område där utvecklingen går snabbt är optisk läsning, och där kan

man säkert förvänta sig en hel del intressanta nyheter så småningom.

När det gäller presentationen av det språkliga materialet ser bilden ljusare ut, åtminstone vad gäller utskrift på papper. De skrivare man idag använder vid stora datoranläggningar (mest radskrivare) är visserligen rätt begränsade vad gäller teckenvariation, något som också gäller de flesta skönskrivare för ordbehandling, men nu börjar de s.k. laserskrivarna bli allt billigare och utgör idag ett ekonomiskt bärkraftigt alternativ till de traditionella skrivarna[3]. Med laserskrivarna har man i princip inte längre några begränsningar vad gäller teckenuppsättningar vid utskriften, och de skriver dessutom med en kvalitet som lämpar sig för publicering. Handlar det däremot om utskrift på bildskärm gäller vad som sagts om bildskärmsterminaler ovan.

Lagringen och bearbetningen är de kritiska stegen i den datoriserade textbehandlingen; inmatnings- och presentationsproblem är mest en fråga om att ha ett lämpligt gränssnitt mot den lagrade informationen. Att finna en lämplig form för lagring och bearbetning av text ur lingvistisk synvinkel hänger intimt samman med datorns teckenrepresentation.

Den enda mer vittgående förändring man kan förutspå för språkvetarnas del, är en ytterligare utbyggnad av den distribuerade databehandlingen, i och med att fler och fler universitetsinstitutioner, även språkvetenskapliga, skaffar sig mikrodatorer. Dessa har ju blivit allt billigare och kraftfullare sedan de först introducerades, och den utvecklingen kan förväntas fortgå ett tag till. Av den anledningen förutser vi att efterfrågan på portabel (maskinoberoende) programvara kommer att öka i framtiden.

Med dessa förutsättningar i åtanke siktar vi på att utveckla ett generellt system för lingvistisk textbearbetning.

3 Krav

Följande krav vill vi att detta system skall uppfylla:

1) Det skall vara **användarvänligt.** Man vill i så hög grad som möjligt automatisera de administrativa uppgifter användaren alltför ofta själv tvingas utföra, såsom att se till att de data systemet arbetar med alltid är konsistenta inbördes. Ju fler sådana rutiner systemet kan klara av, i desto högre grad kan man förskjuta användarens aktivitet in på de lingvistiskt relevanta problemens område. Om man ser det hela ur en annan synvinkel kan man säga, att en stor del av all den språkvetenskapliga databehandling som görs idag består av rent 'mekanisk' textbearbetning (grafordsstatistik, produktion av konkordanser, etc.), och att det är angeläget att göra sådana bearbetningar så enkla som möjligt att utföra för användaren.

2) Ett annat krav är att systemet skall vara i möjligaste mån **generellt,**

med en inbyggd beredskap att klara av de mest skilda särspråkliga problem som kan tänkas dyka upp.

En aspekt på generalitetsproblemet är frågan om vilken sorts information olika användare vill lagra om sina texter och textelement; ett idealiskt system bör ha en flexibel informationsstruktur, där det är lätt att finna plats för nya typer av information, och där man har förutsett att det kan komma att läggas till systemrutiner som bearbetar denna nya information. Systemet måste alltså vara öppet, både vad gäller informationsstruktur och mjukvara.

3) Systemet bör vara i mesta möjliga mån **portabelt**. Detta hänger ihop med den utveckling mot distribuerad databehandling vi förutsåg ovan; i och med att antalet datorer ökar, kommer också efterfrågan på portabel programvara att öka. Om man från början gör programmen portabla minskar man dessutom den möda som måste läggas ned på konvertering och eventuell omprogrammering.

4) Som nämndes i avsnitt 2 ovan, är den teckenrepresentation man använder helt avgörande för hur man kan lagra och bearbeta språkligt material. Det man önskar sig av teckenrepresentationen är följande:

1. Att varje tecken skall vara unikt.
2. Att ens tecken sorteras (kollateras) på det sätt man själv bestämmer.
3. Ordlängder måste stämma med språkets ortografi; digram, trigram osv. skall kunna räknas som en bokstav.
4. Att i vissa fall kunna behandla tecknen på (allo-)grafnivå ("Appearance" i Xerox83 (s. 4)), och i andra fall bara räkna med grafemen, tecknens "kanoniska" identitet.

5) Eftersom många tecken, främst skiljetecken, inte är entydiga, jfr. förkortningspunkt och meningsavslutande punkt, vill man kunna betrakta sådana flertydiga tecken som homografer, och få dem markerade som sådana i texten, för att man sedan skall kunna ta hand om dem med någon form av homografseparering.

3.1 Ungerskan som exempelproblem

För att i någon mån konkretisera de problem man kan stöta på vid språklig databehandling, valde jag ungerskan som typfall, mest av en slump (jag höll på att studera ungerska då jag började arbeta på det textbearbetningssystem som redovisas i nästa avsnitt), men den visade sig vara mycket intressant i detta sammanhang.

Ungerskan skrivs med ett modifierat latinskt alfabete, enligt följande:

(a á) b c cs d dz dzs (e é) f g gy h (i í) j k l ly m
n ny (o ó) (ö ő) p r s sz t ty (u ú) (ü ű) v z zs

Man lägger märke till den stora mängden digram (och trigrammet 'dzs'), som i sorteringshänseende uppför sig som enkla bokstäver, något som datorn inte är förberedd för, då den sorterar tecken för tecken. Långa vokaler skrivs med en akut accent över vokaltecknet (utom långt /ö/ och/ü/ som skrivs med två akuta accenter) och samsorteras med motsvarande korta vokal i ordböcker o.dyl. (markerat med parenteser ovan), men i övrigt betraktas som egna bokstäver, medan långa konsonanter dubbeltecknas, och betraktas som två separata bokstäver. En egenhet i den ungerska ortografin är dock att för de långa varianterna av de konsonanter som skrivs med digram eller trigram fördubblas endast det första tecknet (långt /gy/ skrivs <ggy>), men kombinationen skall sorteras som om den bestod av två separata bokstäver (<ggy> skall sorteras som <gygy>). Versaler samsorteras med gemena, men som i andra språk med både stora och små bokstäver, har de en inbördes sorteringsordning för sådana ord som skiljer sig åt enbart med avseende på dessa (som svenska 'sten' och 'Sten' (namnet)). Se vidare MHSz79.

Den ungerska ortografin orsakar ett annat problem, som egentligen inte hör hemma här, men som det ändå kan vara illustrativt att ta upp i det här sammanhanget: Det kan inträffa i en morfemgräns, att två tecken som ingår i ett bokstavsdigram kommer att stå intill varandra, utan att de för den skull utgör någon bokstav; orden **kőzség** 'kommun' och **kőzsák** 'stenpelare, rauk' delas upp i bokstäver som följer:

<k> <ö> <z> <s> <é> <g> - <k> <ő> <zs> <á> <k>[4].

Här följer ett exempel på hur man skulle sortera några ord enligt de ungerska ortografiska normerna, och hur ett normalt datorsorteringsprogram skulle sortera samma ord (vi förutsätter ett ungerskt tangentbord, för att kunna bortse från problemet med att representera de långa vokalerna[5]):

Ungersk sortering		Datorsortering
kása	'gröt'	kassza
kastély	'lustslott'	kastély
kasza	'lie'	kasza
kaszinó	'kasino'	kaszinó
kassza	'kassa'	kaszt
kaszt	'kast (samhällsklass)'	kása

Datorsorteringen går uppenbarligen inte att använda utan modifikationer.

Kommersiellt tillgängliga sorteringsprogram (ex.-vis SYNCSORT (SyncS80), som finns tillgängligt vid UDAC, där vi gör många av våra bearbetningar) brukar tillåta samsorteringar och ändringar av sorteringsordningen, men man brukar inte kunna komma åt problemet med bokstavsdigram (och längre "-gram") inom ramarna för dessa.

4 En lösning: Textdatabassystemet

Det stod ganska snart klart för oss, att ett system uppbyggt kring en databas vore ett bra sätt att uppfylla de krav och lösa de problem som jag redogjort för i de föregående två avsnitten. Med ett lämpligt valt (läs: portabelt) databassystem, får man flera av kraven uppfyllda nästan gratis:

- Man vinner en hel del i fråga om konsistens jämfört med vanliga, fil-orienterade bearbetningar: Med riktigt skrivna access- och uppdaterings-rutiner ser man till att ändringar i den lagrade informationen slår igenom överallt där det behövs, utan att man som användare behöver hålla reda på vilka filer man har ändrat i, och vilka som måste ändras som en konsekvens av detta.

- Man kan skriva en mer integrerad mjukvara: Eftersom allting händer i en och samma databasrymd, där all information är lagrad i ett enhetligt format, har man i varje ögonblick tillgång till denna information, som man annars skulle behöva hoppa mellan flera olika undersystem (specialskrivna program, operativ-systemrutiner, terminalsystemrutiner osv.) för att komma åt, något som inte kan göra annat än stjäla tid från ens lingvistiska problem, textbearbetningen.

- Man får en i hög grad portabel mjukvara: Genom att man arbetar med databas-systemets accessoperatorer, som förblir desamma oberoende av vilken dator man använder, så kan man flytta sina textbearbetningsrutiner mellan olika datorer utan att behöva skriva om dem.

Vårt textdatabassystem under utveckling består av två delar:

1) Ett generellt relationsdatabassystem, Mimer/DB (Mimer 1982a), som tillhandahåller rutiner för att skapa, ladda, ändra och tömma databanker, och som tar hand om alla operativsystemberoende dataförflyttningar och ersätter dessa med egna databasaccessrutiner, lika på alla datorer som har Mimer/DB[6]. Detta innebär naturligtvis att de program som använder Mimer/DB inte behöver skrivas om för att kunna flyttas mellan olika datorer.

2) En accessmetod, specialdesignad för att lagra och bearbeta text ur lingvistisk synvinkel, som vi själva utvecklat. I första hand har målet varit att implementera de funktioner som TEXTPACK tillhandahåller (se nedan), men det är meningen att man fortsättningsvis skall kunna utnyttja de speciella fördelar databasorganisationen ger för att kunna underlätta användarens arbete.

Databassystemet tillhandahåller ett "virtuellt" filhanteringssystem, och vår accessmetod ett slags operativsystem, speciellt ägnat för lingvistisk bearbetning av textmaterial. Båda dessa komponenter är i hög grad portabla. De textbearbetningsfunktioner som implementerats är följande:

- Möjlighet att definiera alfabeten/skriftsystem med upp till 65 535 (2^{16}-1) tecken i en användarbestämd sorteringsordning, samt att definiera textformatterande specialtecken och teckenkombinationer som skall specialbehandlas i kollateringshänseende.

- Inläsning av texter i en databank, där dessa lagras i ett kompakt internt format.

- Utskrift, på bildskärm eller papper, av grafordslistor, meningslistor och grafordskonkordanser.

Under implementering är:

- Rutiner för datorstödd lemmatisering och homografseparering av texter i databasen.

- Rutiner för att ändra den information som finns i databasen, speciellt för rättning av texter och ändring av alfabetesdefinitioner. Här kan man inte använda ett konventionellt databasfrågespråk (som Mimer/QL (Mimer 1982b)), då det interna format texterna lagras i kräver en rätt omfattande avkodning innan det blir läsbart. Dessutom måste man se till att all information som hör ihop förblir konsistent.

- Rutiner för att hämta in och lagra information om de texter som lagras i databasen, ss. texttyp, genre, författare osv..

Det användardefinierade alfabetet har följande egenskaper:

- Varje tecken har en unik 16-bitsrepresentation[7],

- som samtidigt utgör dess grundposition i sorteringsordningen. Dessutom kan man för varje tecken ange en samsorteringsposition, eller tala om att ett tecken (t.ex. bindestreck) skall ignoreras vid sorteringen. Allt detta görs vid alfabetesdefinitionen för den egna databasen.

- Digram, trigram osv. som räknas som egna bokstäver enligt den aktuella ortografin behandlas som odelbara enheter i den interna representationen (de tilldelas en egen 16-bitskod).

- Varje teckens 'utseende' lagras separat från själva grundtecknet, detta för att användaren skall ha största möjliga frihet när det gäller sådana saker som att kunna ta hänsyn till versaler eller ignorera dem vid olika slags bearbetningar eller olika bearbetningsfaser. För varje tecken har man (arbiträrt) möjlighet att definiera upp till fem gemena (ex.: grekiskt sigma har två varianter) och fem versala allografer; bokstavsdigrammen brukar ha två allografer, t.ex: 'Cs', 'CS' (/č/ i ungersk ortografi; se avsnitt 3 ovan).

Texternas interna format är delvis dikterat av lingvistiska hänsyn och delvis av praktiska begränsningar i MIMER/DB.

När en text läses in i databasen går den först igenom en konverteringsrutin som slår upp varje tecken i ett bokstavsträd. Bokstavsträdet innehåller information om hur tecknen skall sorteras, vad tecknen har för typ (alfabetiskt, siffra, skiljetecken, specialtecken osv.). Tecknen konverteras till det ovan nämnda 16-bits teckensetet och sparas undan i grafordssträngar. Grafordssträngarna i sin tur läggs in i en strängtabell, och representeras sedan i databasens andra tabeller som pekare till denna, detta på grund av att det använda databassystemet inte tillåter variabel fältlängd i tabellerna. Av utrymmesskäl vill man inte överallt reservera plats för det längsta ord man kan tänkas stöta på, eftersom databasen då till stor del skulle bestå av tomrum.[8]

5 Framtiden

Sammanfattningsvis kan man säga att vi strävar efter att utveckla ett flexibelt och portabelt system för lingvistisk textbearbetning, byggt på en databas, för

att därigenom kunna ge systemet mesta möjliga kontroll över datasambanden i den information som lagras.

Textdatabassystemet står ännu på experimentstadiet; man kan läsa in texter, och man kan skriva ut dem på olika sätt, men man har ännu ingen möjlighet att ändra i dem. Vi arbetar dock kontinuerligt med att utveckla och förfina detta redskap för lingvistiskt orienterad textbearbetning, och räknar med att det i framtiden kommer att bli vårt standardverktyg för sådana ändamål.

För det framtida utvecklingsarbetet på textdatabasen svarar i första hand Valentina Rosén vid Centrum för datorlingvistik.

Noter

1. Dock har det gjorts enstaka ansträngningar att konstruera inmatningsutrustning för specifika skriftsystem, ss. Nancarrows "Ideo-Matic Encoder" (Nancarrow 1980; 1981; 1982).

2. Språkdata i Göteborg är något av pionjärer i Sverige när det gäller att utnyttja sättband. Bland annat är Nusvensk frekvensordbok (Allén 1970; 1971; Allén et al. 1975; 1980) baserad på material framtaget från tidningssättband.

3. I BYTE:s marsnummer för 1984 kan man läsa om en ny japansk laserskrivarmekanism som säljs för 1000-2000 dollar (Willis 1984). Om man lägger till kostnaden för kontrollelektroniken, hamnar en färdig laserskrivare i samma prisklass som de dyrare typhjulsskrivarna.

4. Problem av denna typ får snarast lösas genom lemmatisering. Datorn har ingen möjlighet att lösa dem utan lingvistisk kunskap, och när man börjar laborera med sådan, har man redan förflyttat sig bort från den rent 'mekaniska' textbearbetningen, som Rolf Gavare mycket riktigt påpekade efter den muntliga presentationen.

5. Ungerska skrivmaskiner har separata tangenter för de långa vokalerna (á, é osv.), men inte för de bokstäver som skrivs med mer än ett tecken.

6. Mimer finns f.n. (våren 1984) implementerat på ett tjugotal dator - operativsystemkombinationer, däribland flera mikro- och minidatorer (Mimer 1983).

7. Alfabeten med maximalt 255 tecken får rutinmässigt en 8-bitsrepresentation för att man skall spara utrymme i databasen.

8. Rolf Gavare redogör i sin lilla skrift "Lexikografisk alfabetisering" (Gavare 1983) för många av de problem man har att brottas med vid sortering av språkligt material i datamaskinella sammanhang, och presenterar en sorteringsalgoritm som tar hänsyn till de faktorer jag har redogjort för, och en del andra dessutom. Hans resonemang gäller främst svenskan, men hans algoritm är mycket genomtänkt, och t.ex. hans sätt att behandla diakritika som inte är en integrerad del av en bokstav borde passas in någonstans i vårt databassystem.

Referenser

Allén, S. 1970. Nusvensk frekvensordbok baserad på tidningstext. 1. Graford, homografkomponenter. Stockholm.

---------- 1971. Nusvensk frekvensordbok baserad på tidningstext. 2. Lemman. Stockholm.

Allén, S. et al. 1975. Nusvensk frekvensordbok baserad på tidningstext. 3. Ordförbindelser. Stockholm.

Allén, S., S. Berg, J. Järborg, J. Löfström, B. Ralph, C. Sjögreen 1980. Nusvensk frekvensordbok baserad på tidningstext. 4. Ordled, betydelser. Stockholm.

Borin, L. 1983. Systemdokumentation för textdatabasen: Version 1.0. UCDL-R-83-3. Uppsala.

Teletex80 = Character Repertoire and Coded Character Sets for the International Teletex Service. CCITT Recommendation S.61. Geneva 1980.

Gavare, R. 1983. Lexikografisk alfabetisering. Rapporter från Språkdata 14. Göteborg.

JIS78 = JIS (Japanese Industrial Standard) C6226-1978. Code of the Japanese Graphic Character Set for Information Interchange. Japanese Standards Association. Tokyo 1978.

MHSz79 = A magyar helyesírás szabályai (De ungerska rättskrivningsreglerna). Tizedik kiadás. Budapest 1979.

Mimer 1982a. Mimer/DB Reference Manual. Mimer Information Systems. Uppsala

----- 1982b. Mimer/QL Reference Manual. Mimer Information Systems. Uppsala.

----- 1983. Mimer Newsletter, Nr. 1, Januari 1983.
Uppsala Datacentral. Uppsala.

Nancarrow, P. H. 1980. A System for Processing Tibetan Texts in their Original Orthography. i: ALLC Journal, 1980, no. 1.

---------- 1981. A Brief Account of the Development and First Major Installation of the Ideo-Matic Chinese Character Encoder. i: ALLC Bulletin, 1981, no. 3.

---------- 1982. Processing of Ancient Egyptian Hieroglyphic Texts by Computer. i: Richard W. Bailey (ed.): Computing in the Humanities. Amsterdam - New York - Oxford.

Nurmi, S. & V. Rosén 1983. TEXTPACK - Basprogram för bearbetning av text (Del 1). Centrum för datorlingvistik. Uppsala.

PL181a = OS and DOS PL/1 Language Reference Manual. GC26-3977-0. IBM 1981.

PL181b = OS PL/1 Optimizing Compiler: Programmer's Guide. SC33-0006-5. IBM 1981.

Rosén, V. 1983. TEXTPACK - Stegen "PRE" och "BAS" (Del 2). Centrum för datorlingvistik. Uppsala.

Rosén, V. & M. Sjöberg 1983. TEXTPACK - Stegen "RAD" och "MEN" (Del 3). Centrum för datorlingvistik. Uppsala.

SyncS80 = A Programmer's Guide to SyncSort: OS & OS/VS. WCS-0101-0. Whitlow Computer Systems 1980.

Willis, R. 1984. The Japan Shows. An Update on the Japanese Computing Scene. BYTE, Vol. 9, No. 3, March 1984.

Xerox83 = Xerox System Integration Standard XSIS 058305. Character Encoding Standard, May 1983. El Segundo, California 1983.

Benny Brodda
Institutionen för Lingvistik
Stockholms Universitet
S-106 91 Stockholm

RÄTTSINFORMATIK OCH LINGVISTIK

0. Inledning

I denna rapport skall ges en kort översikt över projektet Rättsinformatik och Lingvistik, vilket är ett samarbetsprojekt mellan Institutet för Rättsinformatik (IRI) och Institutionen för Lingvistik, båda vid Stockholms Universitet, och bekostat av medel från DFI (400.000:- för budgetåren 83/84-84/85). Huvudansvarig för projektet i sin helhet är prof Peter Seipel, IRI, och huvudansvarig från Lingvistiska Institutionen är författaren. Av naturliga skäl skall jag här i huvudsak beröra de (dator)lingvistiska delarna av projektet.

Men först litet bakgrundsinformation. Allsedan början av 70-talet har man under överinseende av Samarbetsorganet för Rättsväsendets Informationssystem (SARI) systematiskt byggt upp publikt tillgängliga databaser innehållande bl a allt nytt SFS-material (dvs väsentligen de nya lagarna), material från domstolsverket (sedan 1980 t ex de fulla texterna till rättsfallsreferaten HD och hovrätter, Regeringsrätt och kammarrätter samt från arbetsdomstolen och bostadsdomstolen). Rent fysiskt administreras dessa informationssystem av den statliga datamaskinscentralen DAFA. DAFA handhar också ett stort antal andra, icke publika, statliga informationssystem, men dem kan vi lämna därhän i detta sammanhang. De publikt tillgängliga informationssystemen omfattar i dag (vid årsskiftet 83/84) databaser om totalt c:a 30 miljoner ord löpande text, och de har för närvarande en tillväxttakt av flera miljoner ord per år. (Uppgifterna här ovan är hämtade ur RÄTTSDATA, utveckling, nuläge och framtid, Justitiedepartementet, Jan 1984.)

Denna tillväxttakt kan förväntas öka ganska kraftigt de närmaste åren, för man diskuterar nu möjligheterna att dels lägga in tidigare material än det som nu finns där, dvs SFS-material från tiden före 1970, domstolsmaterial från tiden före 1980, mer full-

ständigt riksdagsmaterial (fn lagras enbart register över riksdagstrycket), samt börja föra in material som i dag inte alls finns där: föreskrifter, anvisningar, material från andra domstolar än de ovan uppräknade, på sikt också lagförarbeten, relevant juridisk litteratur mm, mm. Det är långt ifrån orealistiskt att gissa att dessa informationssystem kan komma att omfatta någonting i storleksordningen en miljard löpord vid slutet av 90-talet. Till yttermera visso diskuterar man redan idag möjligheterna att på ett eller annat sätt också länka dessa system till andra informationssystem inom och utom landet, särskilt då motsvarande system i våra nordiska grannländer. Etc, etc. Det är alltså ganska gigantiska informationssystem som är under uppbyggnad.

1. Rättsinformatik och Lingvistik

Har nu detta något med lingvistik att skaffa? I högsta grad, ty den grundläggande frågeställningen här är naturligtvis exakt densamma som i alla IR-system (IR=Information Retrieval), nämligen problemet att finna alla dokument (i en given dokumentsamling) som handlar om en viss sak, och helst bara dem. Denna frågeställning är i sin tur mycket nära besläktad med den lingvistiska frågeställningen om hur man med formella och automatiska metoder skall beskriva och känna igen det semantiska innehållet i en given text, en frågeställning som också råkar vara en av de mest aktuella just nu inom den komputationella lingvistiken. Skillnaden här från grundforskning som den normalt bedrives inom lingvistiken är först och främst skalan, men också att behovet av fungerande lösningar är överhängande. (Insikten om att denna koppling finns mellan IR och lingvistik är naturligtvis inte ny; jfr - t ex - Brodda & Karlgren 1965.)

Det finns också skäl som gör just Rättsinformatiken intressant. Rättsinformationssystemen är förmodligen de informationssystem i Sverige som i dag är snabbast växande, vilket medför att behovet av bra lösningar är mer överhängande för detta område än något annat. I den internationella IR-forskningen diskuterar man vanligtvis informationsåtervinningssystem för vetenskaplig litteratur, ett självklart både intressant och viktigt område. Sådana system är dock vanligtvis inriktade på återvinning av engelskspråkiga texter (eller möjligtvis texter från något annat internationellt språk-

område), medan rättsinformationssystem med nödvändighet måste vara inriktade på språket i det aktuella landet, i Sverige alltså svenska. Här finns alltså ett direkt behov av utvecklandet av datorlingvistiska metoder för analys av svenska, ett språk som det av naturliga skäl inte forskas så väldigt intensivt om utanför Sveriges gränser.

Det finns också andra aspekter av detta. Det kan väl inte ha undgått någon att det pågått en mycket intensiv debatt om datoriseringen och ett förment därmed förknippat hot mot demokratin. Och det är klart, den som i framtiden behärskar dessa mycket stora informationssystem, han kommer att ha ett oerhört försteg före alla andra bara genom att veta mer.

Enligt mitt förmenande kan man i dag skönja två ganska klara och varandra motstridiga utvecklingstendenser i samhället i detta avseende. Den ena är just den som väl vanligen diskuteras, nämligen en hotande utvecklingen mot en teknokratiskt styrd oligarki, ett fåtalsvälde där just de mimarober som behärskar de enorma informationssystemen innehar nyckeln till makten. Den andra utvecklingstendensen är minst lika stark, tycker jag, men kanske inte lika mycket uppmärksammad i den allmänna debatten, nämligen tendensen mot en verkligt öppen demokrati, där i princip alla medborgare har tillgång till lika mycket information som vilken expert som helst. När kabel-TV-systemen är utbyggda kommer vi dessutom att ha dessa system tillgängliga hemifrån via höghastighetsterminaler.

Hur det går är väldigt avhängigt av hur vi agerar i dag, det är nu vi formar utvecklingslinjerna för det framtida samhället, och här tycker jag man måste diskutera forskningens - och forskarens - ansvar. Den humanistiska språkvetenskapen kan här hjälpa till att utveckla systemen till sant mänskliga och lättåtkomliga informationssystem för envar. Det här är alldeles för allvarliga problem för att lämnas till militären och datamaskinsfabrikanterna att göra upp om på stängda kontor.

2. Nya, okonventionalla angreppssätt efterlyses

Jag skall i det här avsnittet genom ett par enkla tillämpningar av vanlig, hederlig reguladetri visa att helt nya angreppssätt på informationsssökningsproblematiken är av nöden påkallade. När det

gäller existerande, i praktiskt bruk varande informationssöknings-system så arbetar dessa nästan undantagslöst enligt den booleska principen: sökfrågan utformas som ett booleskt uttryck på ett antal sökord eller deskriptorer, och systemets "svar" utgöres av de artiklar där orden eller deskriptorerna förekommer i den sökbara delen av artiklarna i enlighet med det angivna booleska villkoret.

Det finns ganska goda skäl till varför booleska söksystem är så spridda: sökmetodiken är konceptuellt enkel, och det behövs följaktligen väldigt litet av inlärning för att förstå den. De arbetar med en enkel, välutprovad och välfungerande "teknologi", som bl a erbjuder snabba svarstider samt, icke minst viktigt, erbjuder en enkel lösning på uppdateringsproblematiken. Allt detta gör nog att boolesk sökning för överskådlig tid kommer att utgöra en viktig komponent i framtida söksystem .

Varför duger då inte sådana system allt framgent? Ja, förenklat uttryckt kan man säga att de inte klarar uppskalning utan vidare. Ett enkelt räkneexempel kan få illustrera detta. Ett typiskt söksystem har, säg, 10.000 dokument i databasen. Antag sedan att systemet på en viss given sökfråga producerar 20 referenser som svar. Detta är en både typisk och rimlig situation. Om man söker med samma sökfråga i ett analogt system men nu med 10 miljoner dokument i databasen skulle man få 20.000 referenser som svar vid bibehållen precision i sökmekanismen. Detta är inte längre rimligt. Visserligen kan man foga till ytterligare booleska villkor i sin sökfråga, men mycket tyder på att booleska system erbjuder alltför trubbiga instrument för att tillåta en att mejsla ut tillräckligt finstrukturerade svarsmängder i dokumentrymden. (Jfr Walker & Karlgren & Kay 1973.)

Vad är då vägen ut ur detta dilemma? Ja, jag har ju ovan påpekat att IR-problematiken i sig är ekvivalent med att definiera (och känna igen) det semantiska innehållet i text, och på något sätt är det ju just det som den komputationella lingvistiken i dag explicit ser som ett av sina kanske mest centrala forskningsområden, alltså syntaktisk/semantisk parsing, och det ligger nära till hands att börja undersöka om metoder hämtade från detta forskningsområde kunde vara tillämpliga i det här sammanhanget. På sikt är detta säkert möjligt, men inom en mer överskådlig framtid är jag mer pessimistisk, åtminstone om man rör sig inom den mer storskaliga miljö jag diskuterat. Denna pessimism grundar jag på rena

performansöverväganden: algoritmer rapporterade i litteraturen anger ofta 1 sekund per ord och väl det som typiska värden för processning av löpande text, och det även med mycket begränsade lexika och mycket begränsade textkorpora; exempel på sådana algoritmer återfinns i denna volym.

Med en algoritm som tar en sekund per ord att processa löpande text, och med en miljard ord i textbasen tar det faktiskt drygt 30 år att ladda sökdatabasen (1 miljard sekunder = 31.7 år, för att vara exakt), och det säger sig självt att detta inte är intressant i något som helst praktiskt sammanhang. Till yttermera visso är det idag högst oklart vad man skulle göra med sina parsningsträd när man väl är klar; jfr Walker & Karlgren & Kay. (Det man efterlyser är en "metrik" över mängden av sådana här grafer, med vars hjälp man kan definiera semantiskt avstånd mellan i första hand meningar och i andra hand grupper av (textkonstituerande) meningar. Någon effektiv sådan metrik är mig veterligt inte beskriven någonstans. (Se vidare avsn 6, nedan.)

Å andra sidan är det inte så lätt att säga vad man skall göra i stället. I själva verket torde det inte finnas något enskilt "guldägg" som i ett enda slag löser alla problem, utan man får snarare rikta in sig på att tålmodigt pröva sig fram efter många olika vägar och ta vara på varje tänkbart uppslag. Det kommer att bli den samlade kören av åtgärder som (i bästa fall) ger en totalperformans som är den man skulle vilja ha eller har anledning att förvänta sig. En sak kan dock kanske vara värd att påpeka i det här sammanhanget, nämligen att den problemställning vi diskuterar är fullständigt resultatinriktad - ett bra söksystem är bra oavsett hur det fungerar inne i maskinen. Detta medför att det är fritt fram att komma med hur okonventionella analysmetoder som helst, fungerar de så fungerar de. Och den saken är klar, skall man åstadkomma något radikalt genombrott här, så måste man vara okonventionell; traditionella metoder inom den komputationella lingvistiken tycks alltid ha de inherenta performansbegränsningar jag ovan påpekade. (Därmed inte sagt att jag tror på ad hoc metoder - den som har friska och djärva och rimliga hypoteser om hur människan bär sig åt när hon avgör att två texter handlar om samma sak och har förmåga att fånga dessa hypotser i en algoritm har nog bättre chanser att lyckas än den som "bara" är duktig på att sätta ihop fiffiga hashtabeller eller vad det nu kan vara som han/hon är duktig på.)

I det följande skall jag ge en kort översikt över några av de delprojekt av datalingvistisk natur som är planerade eller påbörjade inom ramen för projektet Rättsinformatik och Lingvistik.

3. Heuristisk Parsning

De ursprungliga principerna för den variant av heuristisk analys som vi börjat tillämpa vid Institutionen för Lingvistik, SU, finns först beskrivna i Brodda, 1979, och där finns också redogjort för ett system enligt dessa principer tillämpat på morfologisk analys av svenska. Principerna är senare ytterligare utvecklade i Brodda, 1983, och där ges också en skiss till en tillämpning på syntaktisk nivå. Källgren har sedan ytterligare följt dessa riktlinjer, och bl a utvecklat ett mer fullständigt system för ytsyntaktisk parsning; se denna volym och Källgren 1984.

Heuristisk parsning enligt vår uppfattning av denna innebär att man inte följer en strikt algoritm, utan låter flera av varandra oberoende och parallella processer gripa in i analysen enligt ett ganska stokastiskt mönster. Hitintills har vi också envist hållit fast vid att så långt som möjligt enbart utnyttja information på språkets ytnivåer ("analys utan lexikon"), dock inte så mycket av princip utan fastmer av nyfikenhet på att se hur långt man kan driva analysen med enbart ytkriterier. Enligt vårt sätt att se det hela innebär problemet med att foga till olika typer av lexika bara att länka in ytterligare processer i schemat, parallellt med andra strukturigenkännande processer. Systemmässigt innebär det alltså ingen artskillnad i vårt heuristiska schema att använda lexikon (alternativt låta bli), utan skillnaden kommer bara att synas som en skillnad i performans. Härigenom kan vi alltså exakt se vad lexikonen bidrar med för slags information (i informationsteoretisk mening) - och vad de kostar (i körtid).

I nästa avsnitt presenteras en direkt tillämpning av den heuristiska metoden.

4. Automatisk indexering

Ovan antydde jag att det är/var mer eller mindre av vetenskaplig nyfikenhet som vi försökte hålla oss till strikt ytstrukturell analys. Detta är naturligvis en sanning med modifikation. I själva verket hade vi flera skäl att pröva den linjen. Ett var väl just

vetenskaplig nyfikenhet, men där hade vi också en bestämd hypotes, nämligen att ytstrukturen borde kunna tappas på bra mycket mer information än vad som hitintills antagits. Veterligt har man i praktiskt taget alla parsningssystem som diskuterats i litteraturen nästan axiomatiskt utgått från att man måste ha ett (stam)lexikon med som bas för analysen, och det var det axiomet vi ville utmana, åtminstone så långt analysen rör det rent syntaktiska. (Kommer man in på den semantiska analysen måste man ju rimligen ha ett riktigt lexikon med - lexikonet utgör ju basen för den semantiska komponenten.) Hitintills tycker jag nog att våra datorexperiment i vart fall inte jävat den hypotesen; jfr Källgren, denna volym. Jfr också Hornstrand 1983 där ett experiment redovisas för att utröna i vad mån försökspersoner känner igen satsers syntax med utgångspunkt från samma typ av information som våra parsningssystem utnyttjar, och inte heller denna undersökning jävar vår grundläggande hypotes. (Samma experiment håller just nu - VT 84 - på att upprepas med engelskt material, och det är ytterst intressant att notera, att de preliminära resultaten helt motsvarar de för svenska.)

Ett annat skäl till att pröva den ytstrukturella ansatsen var att i den mån den lyckades så bör man rimligen kunna få relativt snabba system. (Att slå i stora, skivminneslagrade lexika är en inte alldeles kostnadfri sysselsättning i datasammanhang.) De experiment i den riktning vi hitintills genomfört tyder på att den typ av parsing som redovisas i Källgren, denna volym, bör kunna drivas därhän att man får uppåt en 85-90 % korrekt genererade ytträd ur en godtycklig, opreparerad text med en performans av kanske 100 ord/s (på en DEC-10:a). Nu är det klart att i dagens läge har vi samma problem med dessa ytträd som man i IR-sammanhang har med varje form av syntaktiska/semantiska strukturer erhållna ur löpande text, nämligen: Vad skall man göra med dem?

En användning av dem har vi dock funnit i ett av delprojekten under projektet Rättsinformatik och Lingvistik, i ett experiment med automatisk indexering kallat Substantivjakten, närmare redovisat i Källgren, 1984. Ideen vi ville testa var att se i hur hög grad substantiv excerperade ur en löpande text kunde fungera som indikatorer på innehållet i texten i fråga. Detta experiment är fortfarande under utvärdering, men preliminärt törs vi nog påstå att substantiven tycks vara utomordentligt goda indikatorer på textens innehåll, åtminstone som i det här fallet juridiska texter

(närmare bestämt lagtexter), medan t ex verben excerperade på samma sätt ger mycket svaga ledtrådar om innehållet. (Det kan kanske också vara värt att påpeka att användare i mycket stor utsträckning utnyttjar just substantiv i sina sökuttryck.)

Låt mig kort antyda på vad sätt den heuristiska parsningen kan utnyttjas för att identifiera substantiv. Vissa ord signalerar själva att de är substantiv: MYNDIGHETEN, TIDNINGAR, LAGARNA. Ordsluten ("kadenserna", jfr Brodda, 1979, 1982) -(IG)HETEN, -NINGAR, -ARNA fungerar alla som ganska starka indikatorer på substantiv. Andra typer av kadenser är i och för sig också indikatorer på substantiv, men inte särskilt starka: jfr -ER i MAGER, NEGER, NIGER, SÖNDER och BÖNDER, vilket betyder att det är mycket vanskligt att utgå från att ord på -ER är substantiv. Tillsammans med andra ytelement i omgivningen kan de dock förvandlas till starka indikatorer: ALLA (SVERIGE)S (BÖND)ER. Ibland är det bara sådana kringliggande ytelement som indirekt pekar ut ett ord som ett substantiv: ETT (LITE)T (HUS) PÅ (LAND)ET. HUS och LANDET blir här utplockade som substantiv, LANDET enligt konfigurationen PÅ -ET, HUS helt på grundval av indirekta indikationer.

5. "Huru känna igen ord ute i texten fastän de är böjda?"

Som jag tidigare påpekade så är alla i dag i praktiskt bruk varande söksystem väsentligen sk booleska söksystem. En sökfråga till systemet utgöres av ett antal sökord hopkopplade med booleska villkor avseende ordens förekomster i de sökta texterna. Ett problem i det här sammanhanget är att orden ute i texterna inte ser ut på det sätt som det angavs i sökfrågan. Det naturliga är nämligen att man ger sökorden i sin grundform i sökfrågan (eller i vart fall i en enda form), men att det sedan är stor sannolikhet för att orden återfinns i texten i andra former än de man angav. Problemet är då att ändå känna igen orden som lika.

Det traditionella sättet att göra detta är att utnyttja någon form av trunkering. Antag att ett av sökorden är REGEL och att man vill finna alla instanser av det ordet, men också REGELN, REGLER, REGLERNAS... Man kan ju då i och för sig ange alla tänkbara böjningsvarianter själv, men detta kan vara ganska arbetsamt - i en faktisk söksession kan man behöva ange sökfrågor i flera omgångar, var och en innehållande flera sökord - och dessutom är det ju lätt

att glömma någon i hastighten. Trunkering innebär att man anger sökordet t ex som REG$, därmed menande alla ord som inleds med strängen REG (detta kallas högertrunkering); i det aktuella fallet får man då (bl a) de böjda formerna av REGEL som träff.

Trunkering är tyvärr ett ganska trubbigt instrument; i det ovan angivna fallet skulle vi också få REGERING, REGN, REGEMENTE m fl som träff förutom de sökta. Problemet är alltså hur man skall finna bara sökordet och dess böjningsvarianter men inga andra ord (förutom då eventuella homonymer; dessa kan man ju aldrig komma ifrån).

Nu finns det ett par, tre olika sätt att åstadkomma detta. Ett sätt är att gå igenom hela databasen och för alla ord i den löpande texten utföra en (manuell eller) automatisk morfologisk analys, resulterande i en (rot)lemmatisering, alltså en identifiering av de grundord som ordet i fråga är uppbyggt av. Dessa lemmatiserade ord är sedan de som ingår i dokumentindexet, alltså de strukturer som sökningen utförs på. Ett sådant förfarande beskrives av Fjeldvig & Golden, denna volym. Jfr också Brodda, 1982, där problemet att känna igen sammansättningar med utgångspunkt från heuristiska principer diskuteras.

Ett annat sätt är att man utifrån det angivna sökordet genererar de möjliga böjningsstammarna, alltså de former av ordet som ändelser hängs på. Lemmat REGEL har böjningsstammarna REGEL och REGLE. På den förra kan ändelserna 0, -N, -S, -NS hängas, på den senare -R, -RNA, -RS och -RNAS. På samma sätt har BONDE dels BONDE dels BÖNDE som böjningsstammar. Denna ansats har visat sig vara mycket effektiv för finskan (Karlsson & Koskenniemi, personlig ref) och torde också framgångsrikt kunna tillämpas på svenskan.

Problemet att generera böjningsstammarna är i stort sett ekvivalent med att identifiera böjningsklass. I och för sig kan man genom direkt lexikonslagning få reda på den (och för de oregelbundna orden är detta nästan den enda möjligheten, men dessa representerar ett marginellt problem i svenskan), men i viss utsträckning kan man också direkt av ordets form predicera dess böjningsklass; ord som slutar på -NING är 2:a deklinationen, ord på (IG)HET är 3:e etc. Värre är det när man på detta sätt inte har ett morfologiskt stöd, men där gäller ofta vissa "default"-förhållanden. Så t ex kan man defaultmässigt anta att tvåstaviga ord slutande på -E tillhör 2:a deklinationen, och undantagen (som måste testas mot ett lexikon) är ett femtiotal neutrer på -E (VÄRDE, VITTNE, SÄTE, ...).

Etc.

Det vi närmast tänkte testa är ett förfarande som man lite löst kunde kalla "ordsubtraktion". Om man bokstav för bokstav "subtraherar" sökordet REGEL från textordet REGLER får man en "slatt" kvar i vardera ordet, en residualsträng, nämligen EL i det första och LER i det andra. Nu råkar det vara så att bägge dessa två residualsträngar är typiska ordslut till ord i samma böjningskategori, och vi kan ta det som kriterium för att träff föreligger. Gentemot t ex ordet REGERING får vi residualsträngarna EL resp ERING, och dessa är inte typiska kadenser till ord i samma böjningsklass. REGEL och REGERING är alltså inte böjningsvarianter av varandra. (Tekniskt innebär ordsubtraktionsprincipen att ord lokaliseras till sin position i texten via hashtabeller utgående från ordens invarianta stam. I ordet REGEL är orddelen REG hela tiden oförändrad, REG är ordets invarianta stam. På samma sätt har ordet POJKE POJK som invariant stam. Etc.)

Jag skulle kanske kort beröra andra typer av trunkering också. Vid sidan av den typ jag ovan diskuterade (som alltså inte finns tillgänglig i söksystem i dag) utnyttjar man vanligtvis traditionell trunkering. Denna innebär att systemet uppsöker de ord som inleds med den angivna strängen. Detta betyder att också sammansättningar med den angivna strängen som första led hittas, och en sådan funktion är naturligtvis mycket användbar och måste rimligen också kunna erbjudas. Därför tänker jag mig att man skall kunna ange t ex REGEL$ för äkta högertrunkering (som ger ord av typ REGELBOK) och REGEL= för att ange att det enbart är böjningsvarianter jag önskar, alltså av den typ jag tidigare diskuterade.

Sedan har vi problemet med vänstertrunkering, alltså att man anger att det ord man söker på också får ingå som slutled i en sammansättning; ex: MYNDIGHETSREGEL som träff på REGEL. Denna typ av trunkering kan tekniskt sett vara mycket besvärlig att effektuera, beroende på att sökningen vanligtvis organiseras efter ordinitiala sekvenser av tecken, antingen via sk hashtabeller eller via successiva binära träd, bokstav för bokstav eller något liknande. (Det är sådana tekniker som möjliggjort den höga effektiviteten hos de booleska söksystemen.) Ett sätt att åstadkomma fri vänstertrunkering är att låta sökningen gå på rotlemmatiserade former (jfr ovan) av orden i den löpande texten. Ordet MYNDIGHETSREGEL skulle då få ett internt utseende ungefär som MYNDIG>HET´S-REGEL, där

bindestrecket skulle markera sammansättningsgräns. Vänstertrunkeringsuttryck av typ $REGEL skulle då uppfattas som att man sökte normalt men nu i den analyserade strukturen (obs REGEL blir med lämplig tolkning av bindestrecket ett självständigt ord i exemplet ovan). Om man ville ha ordet ovan också i sina böjda former skulle man alltså ange något i stil med $REGEL=.

Det finns också andra sätt att komma tillrätta med problematiken kring vänstertrunkeringen, t ex att man som alternativ till att organisera sökningen efter ordinitiala teckensekvenser gör det hela med utgångspunkt från ordfinala sekvenser. Dock måste man även här ha möjlighet att bortse från böjningsändelserna men exakt hur det skulle kunna gå till är inte alldeles klart. Det är dock klart att detta skall kunna gå att åstadkomma med en kolossalt mycket enklare morfologisk komponent än den som förutsätts i ett system med fullständig morfologisk analys, bl a bör man klara sig nästan helt utan stamlexikon.

6. "Gimme more o´ that!"

Det sista delprojektet jag här skall ta upp är kanske inte i den nu pågående fasen så där direkt tillämpad lingvistik, och därför tänker jag inte behandla det så utförligt. På sikt kommer det dock att bli så i högsta grad, ty det förutsätter att man faktiskt löst problemet med att definiera (semantiskt) "avstånd" mellan texter (jfr avsn 3, ovan). I Brodda 1970 införde jag en mycket enkel metrik för att mäta avståndet mellan deskriptorvektorer (ungefär "den procentuella överlappningen"), som där närmast användes för att mäta avståndet mellan sökfråga och dokumentindex. Ingenting hindrar emellertid att samma mått används för att mäta avståndet mellan (innehållet) i olika textavsnitt. I det aktuella projektet avser jag att tillämpa det hela på paragraf- eller styckenivå (paragrafer och/eller stycken är de naturliga grundenheterna i t ex lagar).

Det problem som diskuterades i Brodda 1970 tog sin utgångspunkt från följande enkla observation: antag att vi har ett sökuttryck med två sökord, A och B. Om vi kopplar ihop dem med OCH-operatorn innebär det att vi enbart betraktar sådana dokument som träff där både A och B ingår i dokumentet (eller rättare sagt dess index, dvs den sökbara delen av dokumentet). Om vi i stället väljer ELLER-

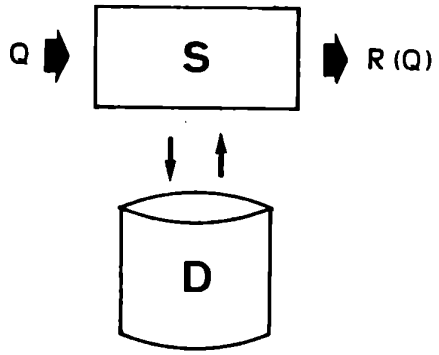
operatorn innebär det att det "räcker" med att endera A eller B ingår (och för all del, även bägge). Antag att vi nu i stället tillämpar följande metod att beräkna om träff föreligger, nämligen att vi åsätter vardera av orden i sökuttrycket ett poängtal, t ex poängtalet 1, och med utgångspunkt från dessa poängtal tilldelar vi sedan varje dokument i sin tur ett poängtal, nämligen totalsumman av de poäng för ord som vi återfinner i dokumentet, och sedan anger vi separat en tröskel, ovanför vilken vi kräver att ett dokument skall hamna för att träff skall föreligga. (Sådana sökmekanismer finns implementerade i många söksystem.) Låt oss nu se hur träffmängden varierar med olika val av tröskel. Sätter vi tröskeln = 1 räcker det uppenbarligen med att endast ett av orden skall ingå för att träff skall föreligga, medan om vi kräver totalpoängen 2 måste uppenbarligen båda orden vara närvarande. Vi ser alltså att tröskeln satt till 1 är ekvivalent med ett ELLER-villkor, medan tröskeln satt till 2 ger något som är ekvivalent med ett OCH-villkor.

Problemet jag diskuterar i ovan nämnda artikel är det generella hur booleska sökförfaranden förhåller sig till "poäng"-förfaranden, och det jag visar är kort följande: Varje poängförfarande är ekvivalent med ett booleskt sökuttryck. Omvänt kan däremot ett givet booleskt uttryck bara approximeras med ett pänguttryck, men lustigt nog så, att approximationen normalt faktiskt blir "bättre" (naturligare) än det ursprungliga booleska uttrycket. För att åstadkomma denna approximation utnyttjas en speciell teknik att väga samman deskriptorvektorer, lämpligen kallad (boolesk) faltning (hur den utföres är för komplicerat att här gå in på), och det är denna sammanvägningsteknik som jag avser utnyttja i det delprojekt som rubriken till detta avsnitt syftar på.

Bakgrunden är följande: Antag att man under en söksession "råkat" finna några dokument (paragrafer, stycken) som visar sig vara relevanta. (Man kan t ex ha startat sessionen med en sk "screening" - "browsing", översiktssökning. Man kan ju också ha fått tips av en kollega, mm.) En mycket naturlig frågeställning är nu om man inte borde kunna få söksystemet att med utgångspunkt från de funna dokumenten självt plocka fram alla liknande. Kort sagt, det man efterlyser är existensen av ett funktionskommando (kanske till och med en funktionstangent) just med innebörden "Gimme more o´ that!". (Jfr också Seipel 1976, där just frågeställningen hur

man skall kunna finna "liknande" lagparagrafer diskuteras.)

Matematiskt kan det nya i denna frågeställning formuleras i följande termer. Antag att vi har ett söksystem av befintlig typ givet, omfattande ett sökförfarande S och en databas D. För en given sökfråga Q ("Query") räknar så söksystemet ut "svaret", responsmängden R(Q) som (hänvisningar till) de dokument som enligt sökförfarandet S uppfyllde sökfrågan Q. Vi har alltså en situation som



Figuren ovan representerar den normala söksituationen. (Denna är dock inte så enkelriktad som figuren ovan låter antyda. Man har ju mycket starka feed back mekanismer, genom att man i en given söksituation vanligtvis börjar med en relativt vag formulering av sitt sökuttryck, och beroende på det svar man erhåller modifierar man successivt sökuttrycket tills responsmängden svarar mot ungefär det man tycker sig vilja ha.)

I "Gimme-more-o'-that"-situationen är frågeställningen den omvända. Där har man en hygglig mängd RO given, och det man frågar efter är om det finns ett sökuttryck Q sådant att R(Q) dels omfattar RO och dels alla liknande dokument. R(Q) skall alltså vara en i någon mening optimalt utvald supermängd till den givna mängden RO.

Nu visar det sig att detta problem rent matematiskt är väldigt underbestämt. Rent teoretiskt kan det finnas ett mycket stort antal supermängder till RO som kunde vara pretendenter på att vara den eftersökta supermängden och samtidigt svarande mot någon sökfråga. För att få någon ordning på det hela måste man då lägga till ytterligare kriterier på hur denna eftersökta sökfråga skall konstrueras. Det ena och mycket nödvändiga kriteriet är att man skall kunna bestämma Q på ett effektivt sätt (dvs Q skall gå att bestämma med en enkel algoritm) ur dokumenten ingående i RO. Ett andra kriteriet är att den erhållna supermängden R(Q) skall vara naturlig

och "bra". Nu visar det sig att den ovan omnämnda faltningsoperationen utförd på deskriptorvektorerna till de dokument som ingår i RO tycks uppfylla dessa villkor. Det första villkoret är mycket enkelt uppfyllt, och en del simuleringsexperiment jag utfört under senare tid tyder också på att de genererade sökfrågorna ger mycket naturliga responsmängder som resultat. Storskaliga "riktiga" test för att utröna det senare planeras så snart vi funnit något lämpligt söksystem som erbjuder en tillräckligt experimentbetonad miljö.

Referenser

Brodda, B. "Document Retrieval - a Topological Problem", SMIL 1970.

Brodda, B. "Något om de svenska orden fonotax och morfotax", i Papers from the Institute of Linguistics, Stockholm University, PILUS no. 38, Stockholm 1979.

Brodda, B. "Yttre kriterier för igenkänning av sammansättningar" i Förhandlingar rörande svenskans beskrivning, nr 13, Helsingfors 1982.

Brodda, B. "An Experiment with Heuristic Parsing", i Papers from the 7th Scand Conf of linguists, Helsinki University 1983.

Brodda, B. & Karlgren, H. "Informationssökningsmetodik", Rapport nr 1 till Kgl Statskontoret ang informationssökningsmetodik. Skriptor, Stockholm 1965.

Hornstrand, Ch."Något om de syntaktiska ytmönstren i svenskan", PILUS nr 50, 1983.

Källgren G. "Automatisk excerpering av substantiv ur löpande text", IRI-rapport 1984:1, Institutet för Rättsinformatik, Stockholms Universitet 1984.

RÄTTSDATA, utveckling, nuläge och framtid, Justitiedepartementet 1984 (ref: Ds Ju 1984:3).

Seipel, P. "Informationssystem för Rättsväsendet: Projekt Index", Stockholm 1976 (SARI).

Walker, D. & Karlgren, H. & Kay, M. (Editors): "Natural Language and Information Science - Perspecitvie and Direction for Research", FID, publ. 551, Stockholm 1973 (Skriptor).

Helge Dyvik
Institutt for fonetikk og lingvistikk
Universitetet i Bergen
Sydnesplass 9
5000 Bergen

Knut Hofland
NAVFs EDB-senter for humanistisk forskning
Postboks 53
5014 Bergen-Universitet

Parsing basert på LFG: Et MIT/Xerox-system applisert på norsk

Det vi har å legge frem her idag, er ikke egne forskningsresultater, men snarere en rapport om et parsing-prosjekt for norsk som vi er i ferd med å sette i gang i Bergen, og en presentasjon av hovedtrekkene i det språkanalysesystemet prosjektet anvender. «Vi» i denne sammenheng er NAVFs EDB-senter for humanistisk forskning ved Knut Hofland, og Institutt for fonetikk og lingvistikk ved Helge Dyvik. Vi regner også med å kunne knytte flere personer til dette prosjektet i tiden som kommer.

Grunnlaget for parsing-prosjektet er det utviklingsarbeid som ble utført ved NAVFs EDB-senter av Per-Kristian Halvorsen, som hadde et forskerstipendium der frem til august 1983. Halvorsen implementerte et universelt språkanalysesystem som er utviklet i samarbeid mellom forskere ved MIT og Xerox Parc i California - bl.a. Halvorsen selv - og begynte arbeidet med å bygge opp en norsk parser innenfor rammen av dette systemet. Halvorsen er nettopp begynt i en ny stilling hos Xerox i California, og kunne derfor ikke selv være til stede her for å presentere sitt arbeid.

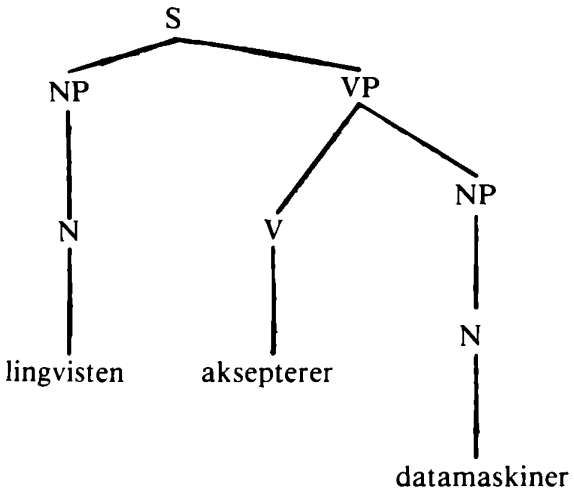
Det norsk-fragmentet som hittil er bygget opp, er meget begrenset. Det omfatter enkle aktive deklarative setninger som kan inneholde infinitivkomplementer kontrollert av subjekt eller objekt (f.eks. «Per lovet Kari å komme»), og et rudimentært leksikon. I det videre arbeidet vil vi i første omgang konsentrere oss om verbalsystemet, nærmere bestemt systemet av perifrastiske konstruksjoner og de modale og aspektuelle kategoriene de uttrykker, om utvidelse av leksikon, og dernest om langdistanse-avhengigheter av den typen vi finner i *hv*-spørsmål, relativsetninger og topikaliserte konstruksjoner. Siktemålet er å øke parserens dekningsgrad til å omfatte sentrale konstruksjonstyper og et større ordforråd, og etter hvert å aktivisere noen av fakultetets språkmiljøer i dette arbeidet. Analysesystemet burde ligge godt til rette for dette, som vi skal se. Vi håper også å kunne knytte arbeidet sammen med noe av den mer anvendelses-orienterte forskningen som skjer ved andre institutter ved Universitetet i Bergen. Det språkanalysesystemet som benyttes, kan karakteriseres som «lingvistvennlig». Det er utviklet under ledelse av Joan Bresnan ved MIT og Ronald M. Kaplan ved Xerox Parc. Den interne representasjonen av grammatikken er en nettverk-struktur, og parseralgoritmen bygger på Kaplans «General Syntactic Processor». Men systemet inneholder også en gramatikktolker som fritar brukeren fra å formulere sin grammatiske beskrivelse som transisjons-nettverk. Grammatiske beskrivelser kan skrives direkte inn i form av regler innenfor grammatikkmodellen «leksikalsk-funksjonell grammatikk» (LFG), og grammatikktolkeren oversetter så beskrivelsen til den mer maskin-motiverte nettverk-strukturen som parser-algoritmen refererer til. LFG er en lingvistisk motivert modell med formelle egenskaper som stort sett er kjent fra moderne lingvistisk tradisjon. Dette legger forholdene til rette for at språkforskere uten spesiell interesse for parsingteori kan knyttes til datalingvistiske prosjekter.

LFG er en transformasjonsfri grammatikk-modell. Modellen skiller dermed ikke mellom dypstruktur og overflatestruktur i konstituent-analysen. Fenomener som EQUI, eller PRO-kontroll i nyere versjoner av Chomskyansk syntaks - f.eks. identifikasjonen av subjektet i «Per lovet Kari å synge» som det underforståtte subjekt for infinitiven - beskrives i leksikon som en opplysning om kontrollegenskapene ved det overordnede verbet. (PRO-analysen i EST opererer riktignok heller ikke med noen transformasjon; men den antar et tomt PRO som det syntaktiske subjekt for infinitiven, noe LFG-analysen unngår.) På tilsvarende måte behandles *passiv* i leksikon som en redundansregel, som i praksis innebærer at aktiv og passiv form av samme verb blir to ulike leksikalske oppslag med identisk semantisk form, men med ulike valg av nominale ledd som argumenter. Langdistanse-avhengigheter som de vi finner i *hv*-spørsmål,

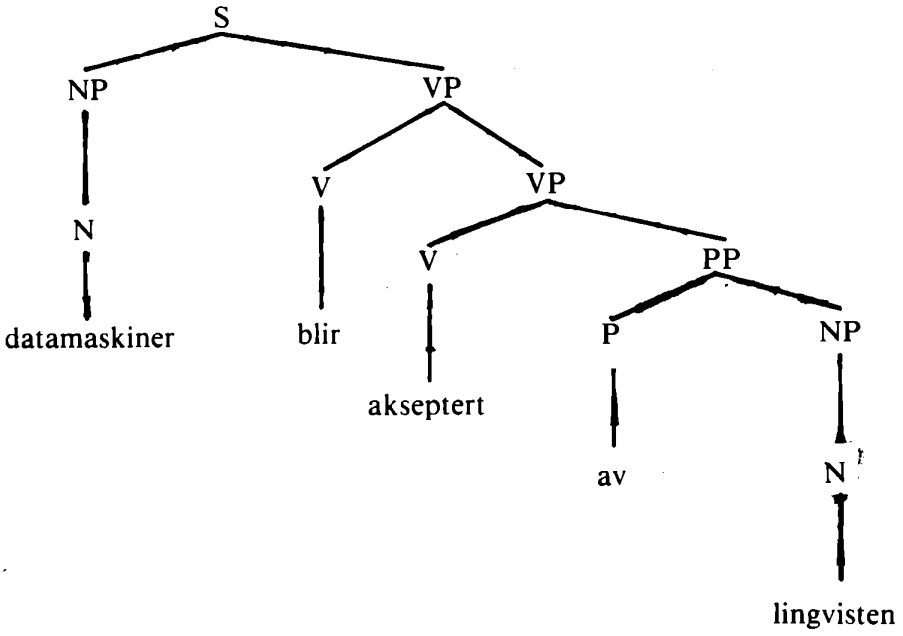
relativsetninger osv. (f.eks. «Hvem påstod Per at Kari ikke likte at han kjente?») lar seg neppe behandle leksikalsk; de ivaretas ved hjelp av en spesiell type korresponderende variabler på den kontrollerende konstituenten («hvem») og den kontrollerte tomme plassen (etter «kjente»).

Dermed kan grammatikkens kontekstfrie frasestrukturregler generere konstituentstrukturer som direkte korresponderer med den observerte streng av former. Analysesystemet tillater at grammatikken skrives direkte inn i form av slike regler. Vi får da konstituentstrukturer av vanlig type:

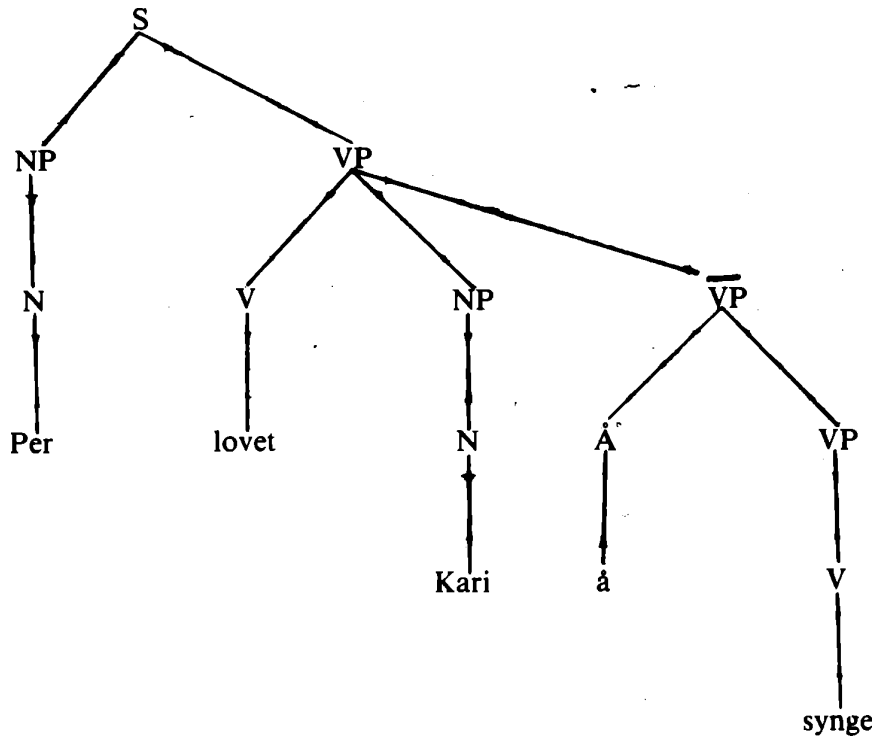
(1)



(2)



(3)

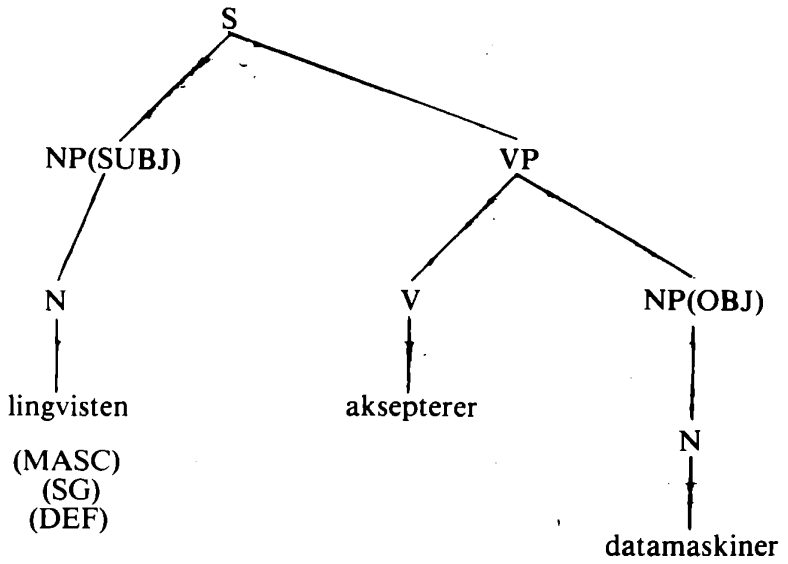


Disse strukturene er da alle generert av kontekstfrie frasestrukturregler. Det innebærer f.eks. at (1) og (2) ikke er relatert gjennom den syntaktiske derivasjonen, og at de syntaktiske reglene heller ikke relaterer «Per» i (3) til noen tom subjektplass foran «å synge». For å uttrykke disse relasjonene, og dermed få et brukbart utgangspunkt for en semantisk fortolkning, må disse enkle strukturrepresentasjonene suppleres med ytterligere uttrykksmidler. Dette er også nødvendig for å eliminere ugrammatikalske setninger som f.eks. *«Per aksepterer Kari å synge», en setning som frasestrukturreglene alene vil generere hvis de først genererer (3). Disse ytterligere uttrykksmidlene er *grammatiske funksjoner* og *grammatiske trekk*.

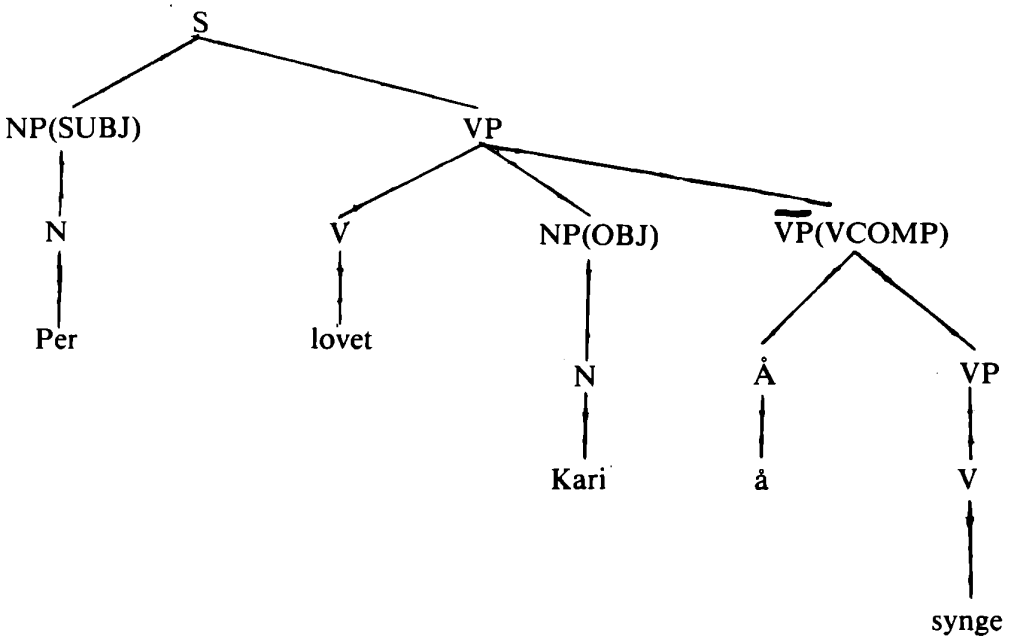
LFG behandler grammatiske funksjoner som SUBJEKT, OBJEKT osv. som primitiver, og ikke som størrelser som nødvendigvis skal være konfigurasjonelt definerbare, slik tilfellet er innenfor EST. De funksjonelle termene SUBJEKT, OBJEKT, osv. i LFG har ikke noen selvstendig interpretasjon, men fungerer bare som et grunnlag for oversettelsen fra syntaktisk til semantisk representasjon - det vil si, de bidrar til å knytte forbindelsen mellom syntaktiske konstituenter og semantiske argumentposisjoner. Dessuten har de en viktig funksjon i å filtrere bort ugrammatikalske frasestrukturer, som vi skal se.

Dels gjennom leksikon og den morfologiske analysen og dels gjennom frasestrukturreglene blir da de syntaktiske trærne supplert med funksjoner og trekk som f.eks. opplyser om at «lingvisten» er SUBJEKT i (1) mens «datamaskiner» er OBJEKT, og videre at «lingvisten» er MASKULINUM, SINGULARIS, DEFINITT. Denne informasjonen kunne vi føye inn i trærne:

(1')

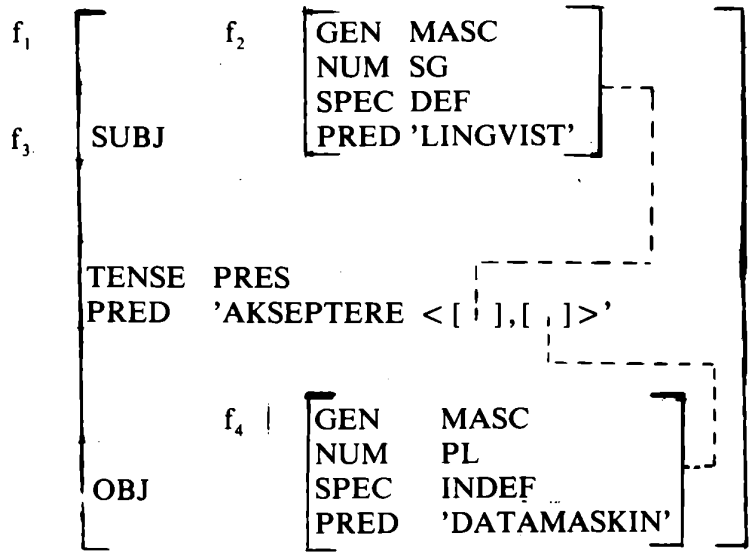

S
NP(SUBJ)
VP
N
V
NP(OBJ)
lingvisten
(MASC)
(SG)
(DEF)
aksepterer
N
datamaskiner

(3')


S
NP(SUBJ)
VP
N
V
NP(OBJ)
VP(VCOMP)
Per
lovet
N
Å
VP
Kari
å
V
synge

På grunnlag av disse funksjonene og trekkene kan man bygge opp en representasjon av setningens *funksjonelle struktur*, der man abstraherer bort fra den syntaktiske tre-konfigurasjonen. En funksjonell struktur er et hierarkisk arrangement av attributter og verdier, der attributtene er grammatiske funksjoner som SUBJEKT osv. og trekk-dimensjoner som NUMERUS osv., og verdiene er enkle symboler som SINGULARIS, semantiske former, eller nye funksjonelle strukturer med sine egne attributter. Den funksjonelle struktur til (1') kan da se slik ut:

(4)



Her har f.eks. attributtet SUBJEKT en ny funksjonell struktur som verdi, nemlig dem som tilsvarer NPen «lingvisten», og den har i sin tur sine egne attributter GEN, NUM osv., med verdier. Hvert attributt har bare en verdi. Hvis den funksjonelle strukturen benevnes f_1, tillater den hierarkiske strukturen oss da å referere konsist til dens enkelte elementer: f_1 (SUBJ) «f_1's SUBJEKT» blir da en funksjon med verdi lik den funksjonelle strukturen til «lingvisten» (f_2), og f_1's (SUBJ)(NUM) «f_1's SUBJEKTs NUMERUS» blir en funksjon med verdien SG. Attributtet PRED har en semantisk form som verdi, og den danner grunnlag for den videre oversettelse til en semantisk representasjon. For substantiver representeres verdien bare med substantivets grunnform. For verb angis argumentstrukturen og hvilke elementer i den funksjonelle strukturen som fyller argumentplassene. I diagrammet er dette angitt ved stiplede linjer fra de strukturene som fyller argumentplassen. Dette er gjort for å understreke at den funksjonelle strukturen som fyller SUBJEKT-attributtet og den som fyller første argumentplass til 'akseptere' er *en og den samme*, og ikke bare to strukturer med samme form. Det samme kunne man oppnå gjennom ko-indeksering av strukturene og argumentplassene.

Som nevnt innføres de grammatiske funksjonene og trekkene dels gjennom frasestrukturreglene og dels gjennom leksikon. I reglene skjer dette ved at de ulike konstituentene får *funksjonelle ligninger* knyttet til seg:

(5)

S → NP VP

(↑SUBJ) = ↓ (↑ = ↓)

VP → V (NP)

(↑OBJ) = ↓

I leksikon er bl.a. følgende informasjon knyttet til formen «aksepterer»:

(6)

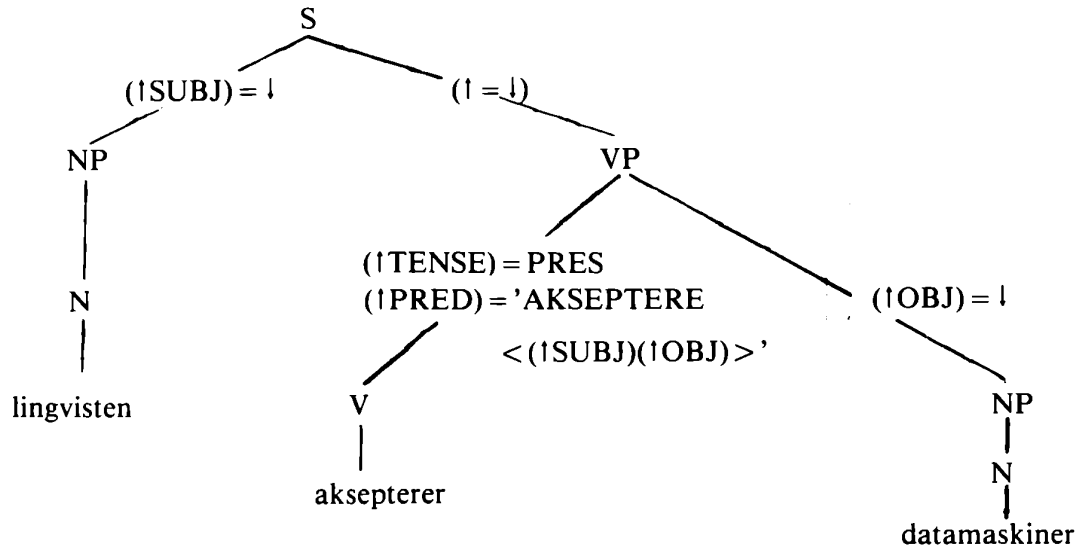
aksepterer: V

(↑TENSE) = PRES

(↑PRED) = 'AKSEPTERE < (↑SUBJ)(↑OBJ) >'

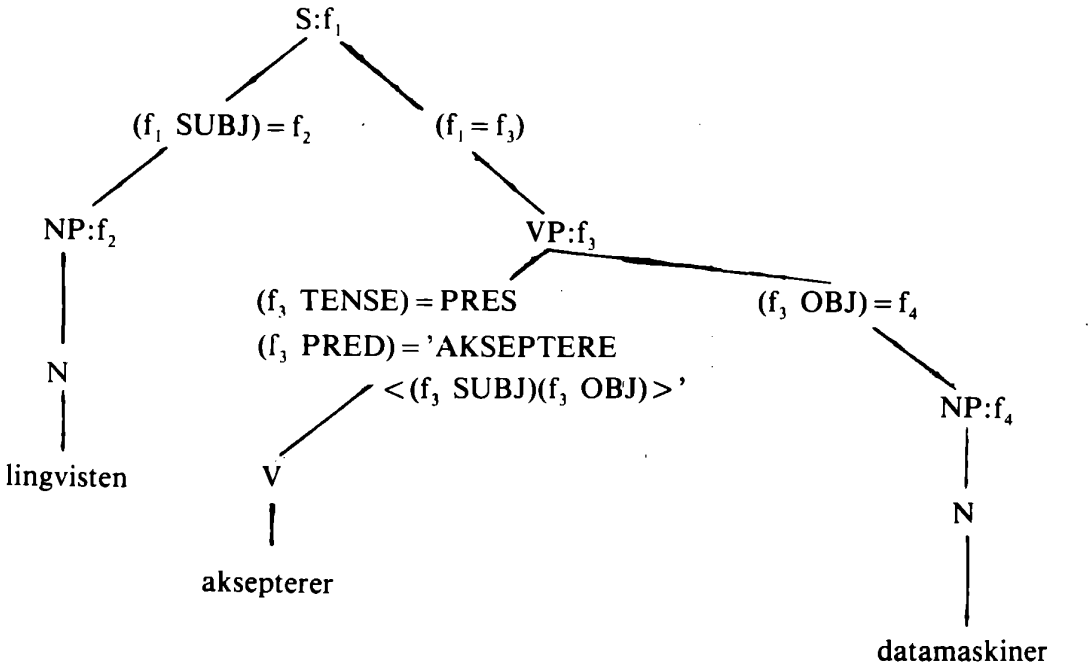
For intuitivt å forstå hva pilene innebærer, kan vi tenke oss en slik ligning plassert i treet på den grenen som fører ned til den konstituenten ligningen tilhører:

(7)



Pilene er *metavariabler* som tar som verdier indekserte variabler som blir knyttet til de enkelte knutene i treet. Piler som peker opp, får som verdi den variabelen som tilhører knuten over, og piler som peker ned, får som verdi den variabelen som tilhører knuten under. Disse variablene (f_1, f_2, f_3 osv.) tar i sin tur som verdier de funksjonelle strukturene som tilsvarer de respektive knutene. Etter at de aktuelle knutene har fått hver sin variabel, kan vi dermed erstatte pilene (metavariablene) med variabler slik at resultatet blir som følger:

(8)



«(f_1 SUBJ)» osv. kan vi lese «f_1's SUBJEKT» osv. Disse ligningene kan så løses med en funksjonell struktur som (4) som resultat. Legg merke til at ligningen under VP identifiserer den funksjonelle struktur for VP med den for S, slik at f-strukturen får færre nivåer enn treet.

På grunn av f-strukturens formelle egenskaper, og det forhold at syntaktiske regler aldri *endrer* tilordningen av funksjoner, kan de bygges opp additivt, det vil si, det spiller ingen rolle i hvilken rekkefølge ligningene løses. Dette er av betydning for løsningsalgoritmen, som da blir enklere enn den ville ha vært hvis en viss rekkefølge hadde måttet bli observert.

De funksjonelle strukturene inneholder da dels informasjon som er nødvendig ved den semantiske fortolkning, og dels sørger de for å filtrere ut som ugrammatikalske visse strukturer som tillates av frasestrukturreglene. Som et eksempel på det første kan vi betrakte hvordan subjektkontrollen i (3) («Per lovet Kari å synge») blir uttrykt. Under 'love' i leksikon finner vi blant annet disse ligningene:

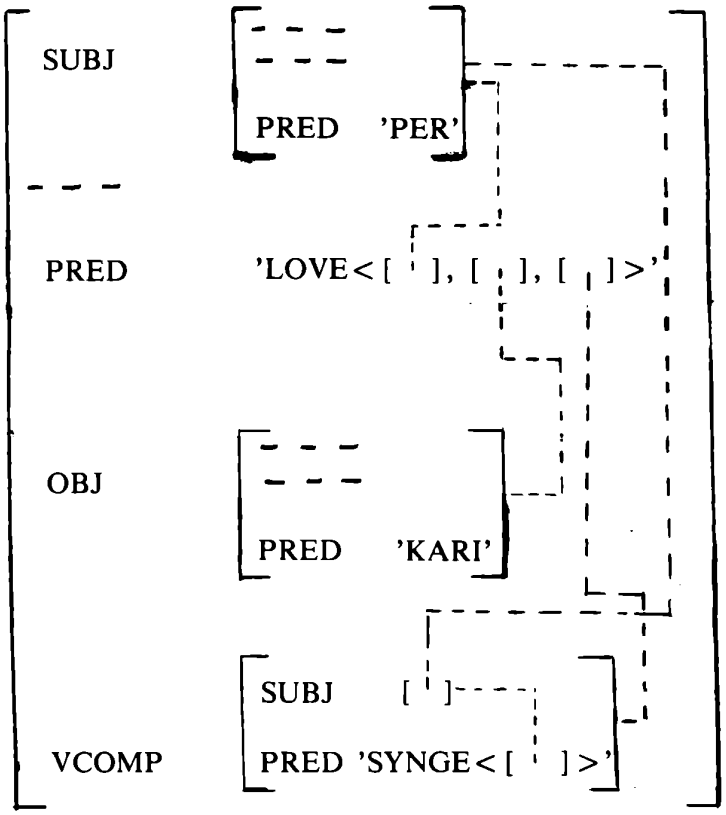
(9)

love: (↑PRED) = 'love<(↑SUBJ),(↑OBJ),(↑VCOMP)>'

(↑VCOMP SUBJ) = (↑SUBJ)

Den siste av disse ligningene stipulerer da at verbalkomplementets subjekt er identisk med setningens subjekt, på samme måte som første ligning stipulerer at første argument for 'love' er identisk med setningens subjekt. Resultatet av at denne ligningen er med, blir en funksjonell struktur som denne:

(10)



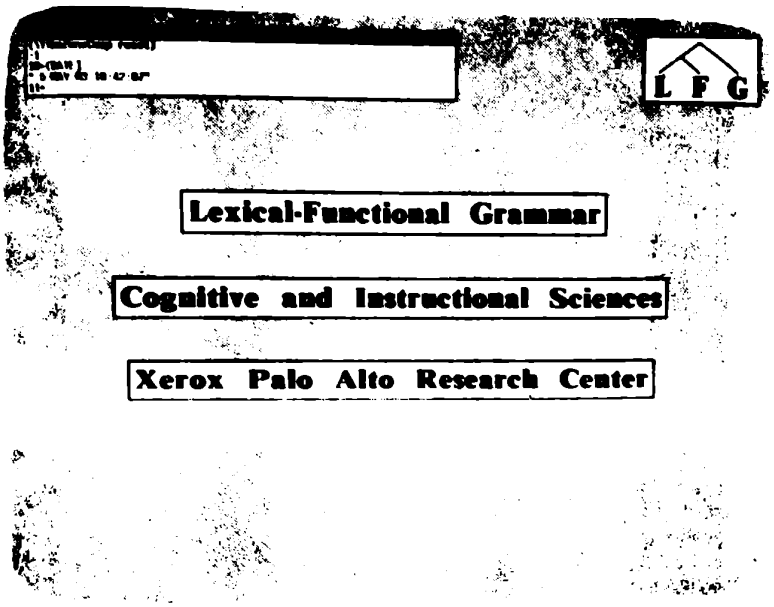
Her fremgår det at setningens subjekt 'Per' også er å oppfatte som subjekt for verbalkomplementet 'å synge', og dermed også som argument til predikatet 'synge'.

Den filtrerende effekt ser vi f.eks. hvis vi tenker oss at vi sløyfet infinitivfrasen og bare genererte «Per lovet Kari.». Da ville det ikke være noe til å fylle tredje argumentplass i den semantiske formen til 'love', og f-strukturen ville være ufullstendig. Omvendt, hvis vi skiftet verbet *lovet* ut med *akseptere* («Per aksepterte Kari å synge»), ville verbets semantiske form bare ha to argumentplasser, og funksjonen VCOMP ville ikke finne noen plass i setningens semantiske form. Resultatet ville altså være en usammenhengende f-struktur, og dette markerer setningen som ugrammatikalsk. Noe lignende gjelder behandlingen av utillatelige langdistanseavhengigheter, f.eks. at ett kontrollerende ledd korreleres med mer enn en tom plass: Frasestrukturreglene muliggjør det, men den funksjonelle strukturen filtrerer ut resultatet.

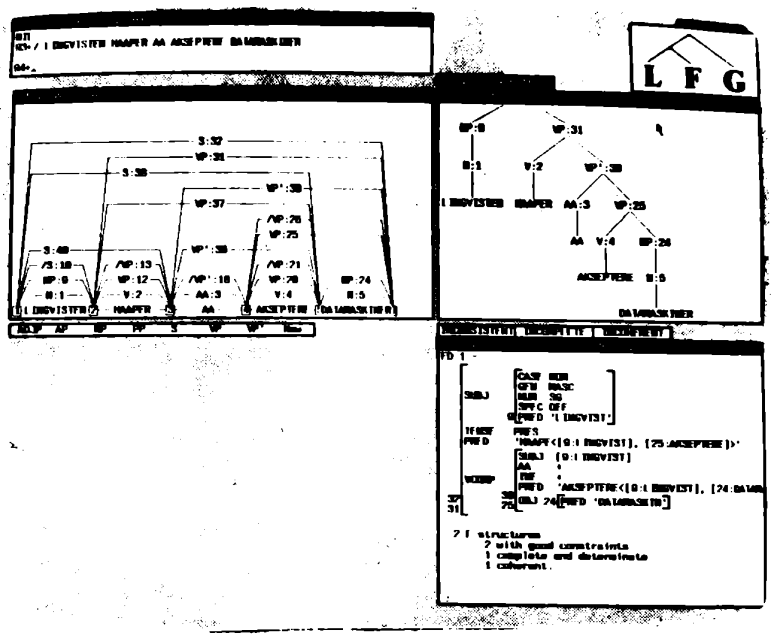
Fra et parsingsynspunkt har denne grammatikkmodellen interessante egenskaper. Ved at den funksjonelle strukturen med sine ulike avhengigheter kan bygges opp på etterskudd, så å si, på grunnlag av de funksjonelle ligningene, kan selve parsingen skje uavhengig av dem. Det innebærer at algoritmen kan forholde seg til en kontekst-fri frasestrukturgrammatikk, dvs. et rekursivt transisjonsnettverk, med de fordeler det innebærer. Resultatet er at algoritmen vil finne trær også for ugrammatikalske setninger, og

også eventuelle uakseptable alternativer som reglene måtte tillate; men disse gir da opphav til inkonsistente, ufullstendige eller usammenhengende f-strukturer, og filtreres dermed ut. En ulempe er at algoritmen da kan tenkes å bruke tid på å regne ut ubrukelige analyser, men i praksis er det naturligvis mulig å interkalere løsning av ligninger i selve parseralgoritmen, i den grad det er ønskelig. I denne sammenheng er det en fordel at man ikke er bundet til noen bestemt rekkefølge i løsningen av ligningene.

Systemet har også implementert en algoritme som oversetter f-strukturer til semantiske strukturer, dvs. en representasjon av setningen i høyere ordens intensjonal predikatslogikk. Denne algoritmen er utarbeidet av Per-Kristian Halvorsen. I de semantiske strukturene uttrykkes f.eks. mulighetene for kvantor-rekkevidde.

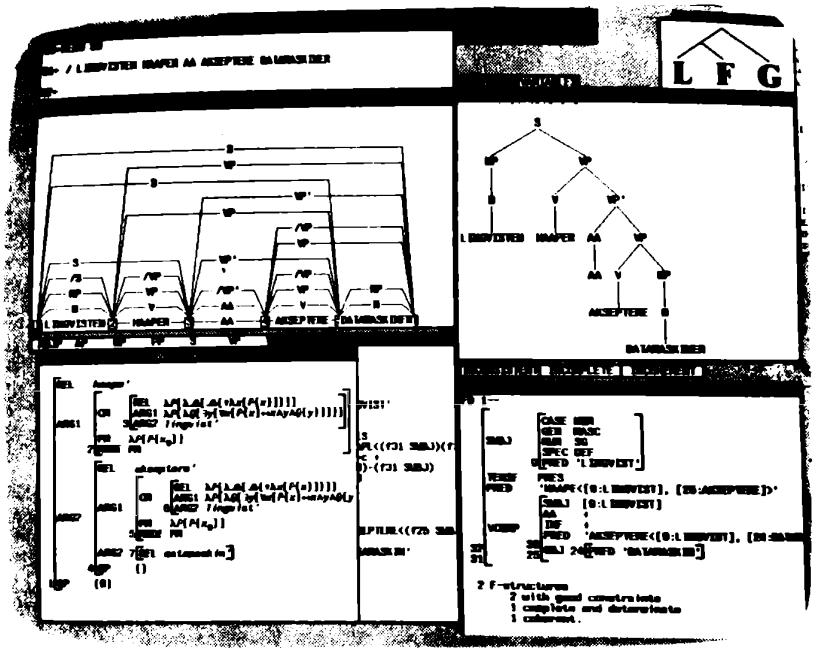


Analysesystemet er implementert i INTERLISP, først på DEC-20 ved MIT og Xerox og senere på LISP-maskiner ved Xerox. En LISP-maskin er en en-bruker maskin med stort primærlager (1.5 Mb) og 10-30 Mb masselager. Denne kan stå tilknyttet et større nett (Ethernet). Maskinen har en stor grafisk skjerm (bitmap display 1000 × 1000 punkter) og en «mus» som pekeinnretning til skjerm. Mye av interaksjonen mellom maskin og bruker skjer via denne. «Musen» har to knapper, en for å kalle frem en meny og en for å utføre en ordre som blir pekt på. Videre har systemet en avansert vinduspakke som gjør det mulig å skrive og redigere data i forskjellige områder på skjermen. Eksemplene i fortsettelsen er hentet fra kjøringer på en slik maskin.



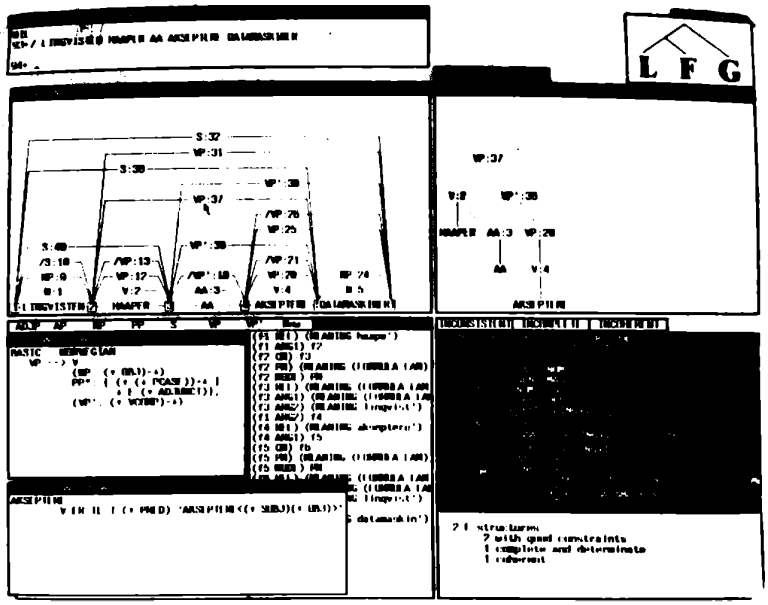
Eksempel 1.

Vi har skrevet inn setningen øverst til venstre og får ut et antall frasestrukturtrær (c-structure) og funksjonelle strukturer (f-structure). For de funksjonelle strukturer får vi angitt hvor mange som er konsistente, sammenhengende og fullstendige. Ved å peke på merkelappen CHART får vi ut et vindu som viser hvoledes analysen har blitt foretatt.



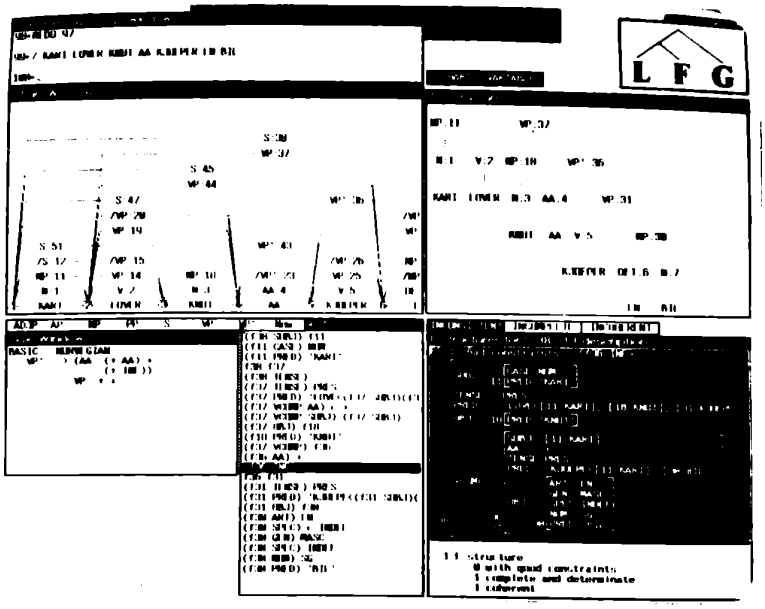
Eksempel 2.

I eksempel 2 har vi også fått ut den semantiske struktur. Ved å peke på forskjellige nivåer i frasetrukturtreet og den funksjonelle struktur vil vi få ut deler av den funksjonelle struktur og den semantiske struktur. Dersom vi peker på et ord vil vi få ut dette ordets innførsel i leksikon. Denne kan vi eventuelt rette og deretter kjøre analysen på ny. Tilsvarende kan vi få ut og rette de grammatiske reglene ved å peke på regelnavnet midt på venstre skjermhalvdel. Dette er vist i eksempel 3 (rule window og lexicon window).



Eksempel 3.

I eksempel 4 ser vi hvorledes det går med en ugrammatisk setning. Vi får her et frasestrukturtre, men ingen konsistent f-struktur. Ved å peke på merkelappen INCONSISTENT viser systemet den inkonsistente f-struktur og angir hvilken funksjonell ligning som ikke er oppfylt (f36 INF). Utsnitt av de funksjonelle ligninger vises i midterste vindu nederst på skjermen. Vi har også kalt opp regelen VP' og ser hvor den aktuelle ligningen står i grammatikken.



Eksempel 4.

LITTERATUR

Bresnan, J. (red.): *The mental reprensentation of grammatical relations*. Cambridge, Mass.: The MIT Press (1982). (Inneholder en rekke artikler om LFG, bl.a. en introduksjon til systemet av R.M. Kaplan og J. Bresnan.)

Halvorsen, P.-K.: Semantics for Lexical-Functional Grammar. *Linguistic Inquiry* 14(4).567-615 (1983).

Jens Erlandsen
IAML
Njalsgade 96
DK 2300 kbh. S.

GESA, et GEnerelt System til Analyse af naturlige sprog, udformet som et oversætter-fortolker system med virtuel mellemkode.

Parsingsystemer til automatisk analyse af naturlige sprog kan udformes på utallige måder - og det gælder næsten uanset hvilken metode, man vælger for selve parsing-processen.

Det er et større projekt at udforme en parser til et rimelig stort udsnit af naturligt sprog. Som regel vil udformningen af så store systemer ske i flere tempi: Primært sker den i kravspecifikationsfasen, hvor der med udgangspunkt i bl.a. brugerbehov og ind- og uddata opstilles en række krav til systemet; Sekundært i designfasen, hvor systemets struktur, algoritmer og datastrukturer endeligt fastlægges. I designfasen vil det typisk være sådan, at der i forhold til de opstillede krav findes flere løsninger.

I denne artikel vil jeg kort skitsere fire system-modeller, og derefter opridse nogle af de overvejelser i designfasen, der førte til at GESA-systemet blev udformet som et oversætter-fortolker system med virtuel mellemkode. Jeg har altså her anlagt en typisk "system-designer synsvinkel" på udformningsproblematikken, idet jeg har set bort fra alle overvejelser angående systemets sproglige kapacitet.

En mere udførlig beskrivelse af GESA, hvor også disse overvejelser er taget med, er givet i SAML nr. 10.

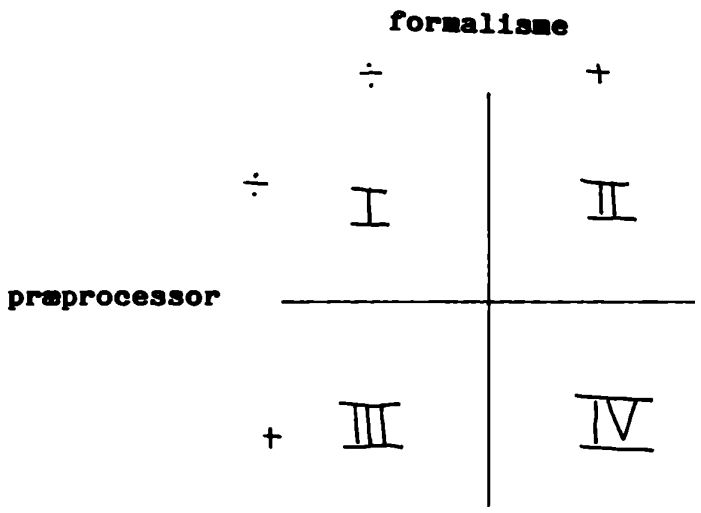
Skemaet med de fire modeller.

De fire udformningsmåder - eller rettere system-modeller - jeg har valgt at tage med her, kan opstilles i et skema, hvor der er taget hensyn til to forhold:

For det første om det samlede parsing-system indeholder en præprocessor i en eller anden form, der behandler sprogbeskrivelsen, inden den anvendes af parseren, eller om det ikke indeholder en sådan processor.

For det andet om grammatikken skal formuleres i en særlig formalisme eller om den formuleres (mere eller mindre) direkte i et allerede kendt programmeringssprog.

Kombineres de to træk, fåes et skema med 4 felter:



MODEL I.

Den første model kan skematisk fremstilles på denne måde:

I denne og i de følgende skitser repræsenterer kasser processorer (dvs. programmer/system-dele) og cirklerne data (fx I: inddata, O: uddata).

I denne model er parser og grammatik bygget sammen i en uadskillelig helhed. Der findes altså ikke nogen særlig præprocessor (programmeringssprogets eventuelle oversætter er ikke medregnet), der behandler en grammatik; og hvis man overhovedet kan tale om en grammatik, er den i hvert fald ikke formuleret i en særlig formalisme (programmeringssproget tæller altså ikke som særlig formalisme i denne sammenhæng).

Er der tale om et eksperimental-system, hvor sprogbeskrivelsen hyppigt ændres og udvides, bliver sådanne systemer let kaotiske og uoverskuelige, hvis ikke man anvender nogle gennemgående principper for data- og process-strukturen.

Et sådant princip kunne fx være recursive-descent (se Aho & Ullman 1977), der kort beskrevet går ud på, at hvert non-terminal i grammatikken (fx på EBNF-form) får sin egen procedure.

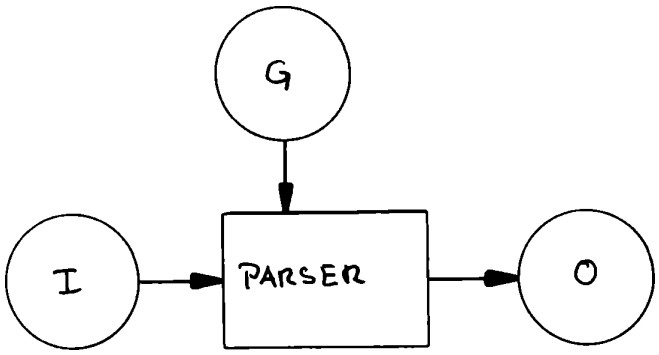
Men selv med anvendelse af sådanne principper er det som regel ikke nogen enkel sag, at ændre sprogbeskrivelsen, fordi det betyder at selve programmet skal ændres. Alle med erfaring i system-arbejde og programmering ved, hvor problematisk det er at ændre større systemer, fx et halvt år efter de er "færdiggjorte".

Har man anvendt et fornuftigt struktureringsprincip, fx det nævnte recursive descent-princip, kan parsere udformet efter denne model blive uhyre effektive.

Så er man i en situation, hvor effektiviteten spiller en stor rolle, fx fordi parseren skal analysere terminal-input fra en bruger i et interaktivt system (fx et undervisningsprogram) og/eller ligger sprogbeskrivelsen helt fast, har modellen måske alligevel en vis berettigelse.

MODEL II.

Denne model, der altså er karakteriseret ved, at grammatikken skrives i en særlig formalisme og direkte i denne form anvendes af parseren under processen, kan skematisk fremstilles på denne måde:



Sådanne systemer kaldes ofte fortolkere, fordi grammatikken fortolkes under processen. Grammatikken betragtes her som data (cirkel G), parseren indlæser før eller under selve parsingprocessen, og den er altså adskilt fra selve parserprogrammet.

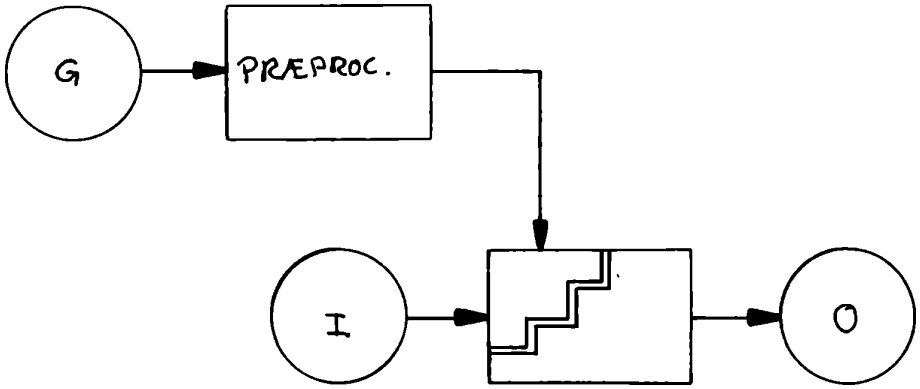
Dette skulle gøre det enklere at ændre og udvide grammatikken, og forandringerne kan måske ligefrem foretages af personer uden kendskab til programmering.

På den anden side er brugervenlige formalismer, og det vil her sige formalismer, hvori beskrivelse af sprog udtrykkes i en for beskriveren naturlig form, ofte et ineffektivt arbejdsgrundlag for parseren.

Da dette forhold er et velkendt problem ved fortolkere, udformes formalismerne ofte således, at der tages mere hensyn til en effektiv fortolkning end til dens naturlighed for brugeren.

MODEL III.

Model III er modellen, hvor der nok er en præprocessor til behandling af grammatikken inden den anvendes af parseren, men hvor grammatikken altså stadig formuleres direkte i et programmeringssprog eller i en slags formalisme, der er tæt på et kendt programmeringssprog. Skematisk ser den således ud:

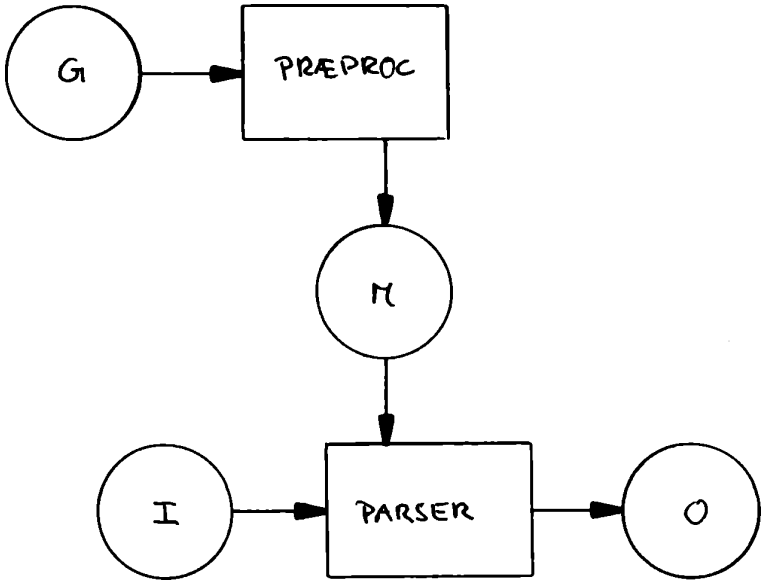


Denne model vælges ofte i systemer, hvor selve parseren skrives i et programmeringssprog, der i sig selv kan blive fortolket, fx LISP.

Præprocessorens primære opgave kan her være at undersøge, om grammatikeren har overholdt programmeringssprogets syntaktiske og semantiske regler i sin formulering af grammatikken; og måske foretager præprocessoren en mindre omformning af grammatikken. Det, der især skiller præprocessoren i denne model fra præprocessoren i næste model, er, at den her hvis den foretager en omformning af grammatikken, altid afleverer den i en programmeringssprogsform, så den direkte kan indgå i parsersystemet.

MODEL IV.

I denne model omformer præprocessoren grammatikken til en form, der af parseren anvendes som data. Denne model vælges i situationer, hvor man ønsker at være friere stillet med hensyn til grammatik-formalismens udformning, dvs. hvor man ønsker en særlig formalisme. Den kan skitseres således:



Denne udformning har sammenlignet med model II en række fordele. Her skal blot nævnes, at den ofte vil kunne give en mere effektiv parsing-proces, og at fejl opdages på et tidligere tidspunkt, og med større sikkerhed. Det sidste giver modellen et stort plus i brugervenlighed.

Præprocessorens uddata kan være på en af to former. Hvis uddata er på tabelform, kaldes præprocessoren en tabel-generator. Er uddata derimod en særlig mellemkode, der fungerer som instruktioner, kaldes præprocessoren en oversætter. I skitsen ovenfor er koden instruktioner til en maskine, der ikke eksisterer som hardware, men som simuleres af et program. En sådan maskine kaldes en virtuel maskine, og mellemkoden til den for virtuel mellemkode.

Udformningen af GESA-systemet.

I det følgende vil jeg kort opridse nogle af de overvejelser i design-fasen, der førte til, at GESA-systemet blev udformet som et oversætter-fortolker system med virtuel mellemkode.

Et af målene med GESA har været at lave et system, der gør det muligt for studerende og en en bredere kreds af lingvister at eksperimentere med lingvistiske beskrivelser og strategier.

Da der ikke kan forudsættes kendskab til programmering og da systemet primært er eksperimentelt, vil det være hensigtsmæssigt at vælge en model for systemets udformning, hvor grammatikken er adskilt fra selve parser-programmet. Model I ovenfor er hermed ude af billedet.

Det må i et eksperimental-system anses for vigtigt, at grammatikeren overfor systemet kan formulere sprogbeskrivelsen i en form som er overskuelig og gennemskuelig og samtidig føles naturlig. Det må derfor være mest hensigtsmæssigt at lave en særlig formalisme til grammatikkerne.

Parsing-systemer til naturlige sprog, vil altid være meget ressourcekrævende, dvs. processen vil kræve meget plads i maskinen og vil tage meget lang tid. Så selv om der er tale om et eksperimental-system, hvor en bruger formodentlig ikke vil prioritere effektivitet særlig højt, bør alt andet lige den mest effektive model vælges. Hermed er model II ude af billedet.

Alt i alt må model IV nok anses som den mest hensigtsmæssige udformning for et system som GESA. Til gengæld er det den vanskeligste og mest omfangsrige model at implementere. Dette kunne tale for at formalismen blev tilpasset, så systemet blev enklere, men det ville sikkert føre til det resultat, at den ikke længere opfyldte de oprindelige krav.

Det kan selvfølgelig ikke accepteres at de "lingvistiske" krav til formalismen og krav om brugervenlighed nedprioriteres på denne måde; på den anden side skal beskrivelsen jo omskrives.

Ved at betragte den formalisme sprogbeskrivelsen skrives i som et højniveau programmeringssprog og præprocessoren som en oversætter, kunne man måske komme ud over denne problemstilling.

For en ikke-datalog virkede tanken om at skulle lave en rigtig oversætter overvældende, men en nøjere gennemgang af litteratur om design af programmeringssprog og oversættere viste, at man er nået meget langt med at forene krav til programmerings-

sprog og deres oversættere med et krav om, at de sidste skulle være nemme at lave og vedligeholde.

Inden for dette forskningsområde har man især koncentreret sig om oversætterens parserdel, idet man har søgt efter metoder, der kunne klare flest af de eksisterende programmeringssprogs konstruktioner på den mest effektive måde.

Længst er man nok nået med en metode kaldet LR-parsing, der oprindelig er udviklet af Knuth (Knuth 1965) (se også Aho and Ullman 1977). Metoden kaldes LR, fordi parseren skanner inddata fra "Left-to-right" og konstruerer "a Rightmost derivation in reverse". En LR-parser har tre fordele frem for de hidtil anvendte metoder:

1. Den kan udformes, så den kan genkende en hvilken som helst sprogkonstruktion, der kan beskrives med en kontekstfri grammatik.

2. Den er mere generel end tidligere anvendte parsere og tilmed lige så effektiv.

3. Den finder syntaksfejl så hurtigt, det er muligt med strikt venstre-højre skanning af inddata.

Desværre er en LR-parser/compiler til et bestemt programmeringssprog vanskelig at implementere, hvis man starter fra bar bund. Teknikken forudsætter næsten, at man har en LR-parser/compiler-generator: dvs. at selve parseren er tabeldrevet og at der findes en præprocessor, der kan indlæse en kontekstfri grammatik og generere den tilsvarende tabel.

Har man en sådan LR-parser-generator, mangler man "kun" at lave symboltabel-behandlingen, typecheck o.lign. og kodegenerering i at have en fuldstændig oversætter. At lave en oversætter til et PASCAL-subset kan derfor gives som en 4-ugers opgave på datalogi 1. del.

LR-parsere findes i forskellige varianter. Den mest udbredte

kaldes Look-A-head LR; LALR(1)-parseren er den foretrukne, fordi den er næsten lige så effektiv som en LR-parser, men kun behøver 1/10 så store tabeller.

Implementering af en PASCAL-oversætter kan som nævnt betragtes som en overkommelig opgave, og da PASCAL og PASCAL-oversættere findes grundigt beskrevet, var der grundlag for at undersøge om de til PASCAL stillede krav kunne forenes med de krav, der var blevet stillet til formalismen i GESA.

Det viste sig at være tilfældet, og da der tilmed fandtes en LALR(1)-parser/compiler-generator, BOBS (Eriksen a.o. 1982) på vores maskine, var der kun et alvorligt problem tilbage: Hvordan skulle oversætterens uddata se ud.

Normalt er uddata fra en PASCAL-oversætter ikke en parsertabel, men derimod maskinkode, assembler eller den såkaldte P-kode. P(ASCAL)-kode er et instruktionssæt til en virtuel maskine, der som nævnt er en maskine, der ikke eksisterer som hardware. En virtuel maskine kan via fortolkning af instruktionerne simuleres af et program, der er skrevet i en eksisterende maskines sprog. Så ved at lade oversætteren generere en virtuel mellemkode, opnår man altså at gøre hele systemet mere uafhængigt af en enkelt maskine. Hvor portabelt det i sidste ende bliver, afhænger selvfølgelig også af, hvilket sprog det er skrevet i.

En anden fordel ved virtuel mellemkode er, at den kan udformes efter behov, idet instruktionssæt og maskine definerer hinanden. Instruktionssættet kan altså tilpasses, at den virtuelle maskine er en parser til naturlige sprog.

Da P-kode alene ikke hensigtsmæssigt definerer en sådan parser, var det nødvendigt at definere et nyt instruktionssæt: G-kode, hvis oversætteren skulle generere virtuel mellemkode.

Ved bestemmelsen af instruktionssættet skulle der tages hensyn til, at sættet ikke måtte blive for stort, at den enkelte instruktion ikke måtte blive for kompleks, at koden skulle

kunne genereres direkte ved første gennemlæsning af inddata (et-passage-oversætter), og endelig som nævnt, at den virtuelle maskine var en parser til naturlige sprog.

Det lykkedes at udforme et instruktionssæt, der opfyldte disse krav, og GESA-systemet blev derfor udformet som et oversætter-fortolker system med virtuel mellemkode.

Aho and Ullman 1977

A. V. Aho and J. D. Ullmann: Principles of compiler design (addison-Wesley Publishing Company 1977).

Eriksen a.o. 1982

S. H. Eriksen, B. Bæk Jensen, B. Bruun Kristensen, and O. Lehrmann Madsen: The BOBS-system, 3rd ed. (Aarhus University 1982)

Erlandsen 1983

J. Erlandsen: En introduktion til parsing og parseren GESA, in Skrifter om Anvendt og Matematisk Lingvistik 10, (Københavns Universitet 1983).

Knuth 1965

D. E. Knuth: On the translation of languages from left to right, in Information and control 8, 1965, pp. 607-639

Forsk.stip. Tove Fjeldvig og cand.philol. Anne Golden
Institutt for rettsinformatikk
Universitetet i Oslo
Niels Juelsgt. 16
Oslo 2

Oslo, 1. juni 1983

AUTOMATISK ROTLEMMATISERING

1. Prosjekt for automatisk rotlemmatisering

Institutt for rettsinformatikk (IRI) (tidligere Institutt for privatretts avdeling for EDB-spørsmål) har i mange år drevet forskning omkring tekstsøkesystemer. I ett av disse prosjektene har vi spesielt tatt opp ulike lingvistiske aspekter knyttet til denne type systemer. Foreløpig har arbeidet vært konsentrert omkring utvikling av en metode for gruppering av ord med felles rot på tvers av ordklassene. Prosessen har fått navnet "automatisk rotlemmatisering", bl.a. for å skille den fra den mer vanlige lemmatiseringsprosessen som opererer innenfor de tradisjonelle ordklassegrensene.

> Et lemma kan sammenlignes med et slags stikkord eller et oppslagsord i en ordbok. At to ord tilhører samme lemma, betyr enten de er ulike bøyningsformer av samme grunnform (leksem) eller at de er to ulike skriftvarianter av samme leksikalske ord (f.eks. fram og frem). Et rotlemma vil derfor være en betegnelse på ord som har samme rot og samme semantiske betydningen når man ser bort fra den informasjon som ligger i selve bøynings- og avledningsendelsen.

2. Bakgrunn

Arbeidet med gruppering av ord med felles rot ble allerede påbegynt i 1979 som en aktivitet under prosjekt NORIS (34) ved IRI. Dette prosjektet, som var finansiert av Norges Teknisk-Naturvitenskapelige Forskningsråd og ledet av cand.mag. Tove Fjeldvig, tok sikte på å undersøke muligheten for enkle strategier for tekstsøking basert på argumenter i naturlig språk.

Blant de problemer man ønsket å belyse i dette prosjektet, var muligheten for automatisk utvidelse av søkeargumentet med alle aktuelle bøyningsformer til søkeordene. Også avledningsformene var aktuelle forutsatt at ordene representerte det samme innholdet (grunnidéen).

> I et tekstsøkesystem vil i prinsippet alle ordene i dokumentene være søkbare. Det vil bl.a. si at en bruker må selv definere alle mulige bøyninger og avledninger av aktuelle søkeord. Hvis man for eksempel bare angir søkeordet "BIL", vil man ikke finne de dokumenter som inneholder ordene "BILEN", "BILER" eller "BILENE".

Man fant det også interessant å undersøke om en slik rutine representerte et alternativ til manuell høyre-trunkering, eller om den kunne inngå som et ledd i en rutine for automatisk trunkering.

> Trunkering er en måte å spesifisere søkeord på ved å definere en viss følge av tegn som søkeordet skal inneholde. Alle ord som inneholder den definerte tegnstrengen anses kvalifisert som søkeord. Den mest vanlige form for trunkering er høyre-trunkering, hvor tegnstrengens høyre del er uspesifisert. Søkeargumentet BIL* (hvor * er brukt som trunkeringstegn) vil omfatte alle ord som begynner med bokstavene "bil", f.eks. BILER, BILEN, BILENE, BILHOLD, BILDE, BILLION. Ulempen med trunkering er at den medfører en del støy og ikke inkluderer vokalvekslinger (f.eks. sterke verb og uregelmessige substantiv).

Dessuten ville en slik rutine gjøre analyse av et søkeargument i naturlig språk lettere, og dermed også øke muligheten for et bedre søkegrunnlag.

I NORIS (34) ble det etterhvert et spørsmål om automatisk rotlemmatisering. Ved prosjektets opphør i 1981 fant vi resultatene såpass interessante, at vi ønsket å fortsette studiene.

Samtidig med prosjekt NORIS (34) pågikk det også et prosjekt LÆREBOKSPRÅK ved Nordisk institutt (UIO) hvor man var opptatt av lemmatiseringsproblematikken. Prosjektet var ledet av amanuensis Anne Hvenekilde og cand. philol. Anne Golden og finansiert av Kirke- og undervisningsdepartementet. Formålet med prosjektet var å kartlegge de høyfrekvente ordene i en del fagbøker for grunnskolen, slik at det kunne lages støttemateriell i norsk for innvandrerelever. Støttemateriellet skulle i første omgang konsentrere seg om vokabularet i fagbøkene, og man ønsket derfor å finne fram til de ordene som de fremmedspråklige elevene fikk mest nytte av å ha lært ved lesing av fagbøker i skolen. I dette arbeidet var det ikke tilstrekkelig å ta utgangspunkt i grafordenes frekvens - og heller ikke lemmaets frekvens (dvs. den samlede frekvens for de ulike bøyningsformer av samme grunnord). Det riktigste bilde av ordforådet fikk man ved å beregne frekvensen til et grunnord med alle dets avledninger. Med andre ord: det var nødvendig å rotlemmatisere ordene.

> F.eks. hvis en fremmedspråklig elev har lært ordet "ANVENDE", vil hun (evt. han) også forstå ordene "ANVENDELSE" og "ANVENDELIG" så snart vedkommende også har lært noen enkle regler for ordlaging i norsk.

I dette prosjektet ble grupperingen av ordene foretatt manuelt, og det utviklet seg mange diskusjoner omkring hvilke ord som tilhørte samme semantiske rotlemma.

Ved de Nordiske datalingvistdager 1981 ble vi oppmerksom på vår felles interesse for automatisk rotlemmatisering, og et samarbeid ble etablert. Fordi den ene hadde kompetanse i edb og den andre i lingvistikk, kunne vi nå ta fatt på en rekke av de uløste problemer som vi hver for oss hadde stått ovenfor.

Arbeidet med automatisk rotlemmatisering ble derfor intensivert, og i dag utgjør det et eget delprosjekt under NORIS (58). Dette prosjektet er finansiert av NTNF og tar sikte på å utvikle en "intelligent forsats" til tekstsøkesystemer.

3. Målsetning

3.1 Programsystem

Målsetningen for arbeidet med den automatiske rotlemmatiseringen var å utvikle et programsystem for rotlemmatisering som ikke var for ressurskrevende. Dette var spesielt viktig med tanke på implementering av et slikt program i tekstsøkesystemer, da responstiden i slike systemer spiller en meget viktig rolle.

Det var derfor utelukket å basere programsystemet på et manuelt utviklet leksikon. I steden valgte vi å basere oss på et sett med generelle regler for rotlemmatisering som var uavhengig av datamaterialet. Dette regelsettet kunne forsåvidt også inneholde ord, men disse måtte i tilfelle tilhøre lukkede ordklasser (f.eks. funksjonsord, sterke verb etc.) slik at det ikke oppsto behov for endring av regelsettet ved oppdatering av datamaterialet.

3.2 Rotlemmatisering

Rotlemmatiseringen skulle bidra til at ord med felles grunnform ble gruppert. Dette gjelder ikke ord som i vesentlig grad har fått endret sin betydning ved at de har fått lagt til endelser eller avledninger (f.eks. BEHOLDE og BEHOLDNING, KOMMUNE og KOMMUNIST, OPPDRAG og OPPDRAGELSE, STAT og STATISK.) Eksempel på ord som kan sies å tilhøre samme rotlemma er:

AMERIKA	ANTA
AMERIKAS	ANTAS
AMERIKANSK	ANTOK
AMERIKANSKE	ANTATT
AMERIKANER	ANTATTE
AMERIKANERE	ANTAKELSE
AMERIKANERNE	ANTAGELSE
AMERIKANISERE	ANTAGELSEN
AMERIKANISERT	ANTAKELIG
:	:
:	:

3.3 Homografseparering

Målsettingen omfattet ikke kartlegging av homografer. Dette problemområdet ble ansett for å være for omfattende innenfor rammen av prosjektet. Imidlertid vil en langt større del av homografene være interne homografer (dvs. homografer som har samme rotlemmatilhørighet) enn tilfellet er ved vanlig lemmatisering.

> ARBEIDER (nomen agentis) og ARBEIDER (verb, presens) er eksempler på interne homografer i en rotlemmatiserings-prosess. De skal grupperes sammen og skaper derfor ikke noe problem. I en vanlig lemmatiseringsprosess ville disse regnes som eksterne homografer fordi de tilhører hvert sitt lemma.

Derimot vil eksterne homografer virke forstyrrende (f.eks. ordet HELT som både kan være et substantiv, et adverb og et verb i perfektum partisipp). Retningslinjen for grupperingen var at vi skulle forsøke å plassere ordet i det rotlemmaet som vi regnet med hadde høyest frekvens, men generelt skulle rotlemmatiseringen aksepteres så sant ordet ble plassert i ett av de riktige rotlemmaene.

4. Gjennomføring

Prosjektet har følgende hovedaktiviteter:

1) Tilretteleggelse av datamateriale
2) Utvikling av et regelsett
3) Utvikling av ett programsystem
4) Testing

I utviklingen av regelsettet var det nødvendig med et eksperimentmateriale. I vårt tilfelle var det naturlig å ta utgangspunkt i det materialet som allerede var tilgjengelig, og eksperimentmaterialet ble derfor sammensatt av ulike fagbøker for grunnskolen (geografi, fysikk og historie) og 2 juridiske dokumentsamlinger (tinglysingsavgjørelser og sammendrag av lagmannsrettsavgjørelser i familie-, skifte- og arverett). Tilsammen besto korpuset av ca. 1/2 mill. løpende ord, hvorav de juridiske samlinger utgjorde ca. 2/3. Vi fant det hensiktsmessig å fjerne skrivefeil, utenlandske ord, nynorske ord, forkortelser og noen ord med gammel skriveform. Når det gjaldt navn, beholdt vi bare egennavn som hadde substantiv som siste ledd og navn på land og verdensdeler. Antall ulike ord i korpuset ble som følge av dette redusert fra ca. 29.000 til ca. 25.000.

Tyngden i prosjektet har helt opplagt ligget i spesifiseringen av regelsettet. Gjennomføringen kan deles i tre stadier:

a) Forslag til hovedregler ble satt opp på bakgrunn av en systematisering av formverket i norsk.
b) Hovedregler ble testet og spesialregler ble innført.
c) Reglene ble vurdert ut fra deres hyppighet.

Arbeidet har nærmest tatt form av en "feedback-prosess". Vi startet med et sett med hovedregler. Disse ble så testet på

eksperimentmaterialet, og ut fra en vurdering av feilene ble spesialregler satt opp. Vi gjentok så prosessen med det nye regelsettet, og fortsatte inntil vi sto igjen med en fullstendig systematisering og kategorisering av alle ordene i eksperiment-materialet. Før det endelige regelsettet ble fastsatt, ble det foretatt en vurdering av de enkelte reglers "eksistensberet-tigelse" på grunnlag av hvor hyppig de ble brukt - dvs. hvor mange ulike og løpende ord de dekket. Særlig gjaldt dette spesialreglene.

I overensstemmelse med målsettingen ble det lagt vekt på at regelsettet skulle være så uavhengig av korpuset som mulig. Det var allikevel vanskelig å unngå at enkelte regler ble noe "preget" av vårt materiale, f.eks. spesialreglene. For å få en generell bedømmelse av regelsettet til slutt, ble det testet mot et helt tilfeldig valgt materiale (barneboken "Ole Brumm").

5. Regelsettet

Den ordbehandlingen som regelsettet dekker, kan deles i 7 kategorier:

1) fjerning av bøyningsendelser
2) fjerning av avledningsendelser
3) nøytralisering av vokalvekslinger
4) nøytralisering av konsonantforenklinger og -fordoblinger
5) nøytralisering av stavelsessammentrekninger
6) nøytralisering av skriftvarianter av samme leksikalske ord
7) markering av ord som får felles oppslagsform som følge av vår behandling uten at de tilhører samme rotlemma.

En del ord må behandles med hensyn til flere av disse kategoriene samtidig. Det finnes derfor forskjellige typer regler. En type er "enkel" og tar bare for seg en kategori av gangen. Disse reglene er i noen tilfeller temporære, dvs. de er

kodet slik at de sender ordet til viderebehandling i motsetning til de endelige reglene som avslutter behandlingen av ordet. Andre regler er sammensatte, de dekker flere av kategoriene på ren gang.

Etter en systematisk gjennomgåelse av de forskjellige ordklassers paradigmer, satte vi opp et forslag til fjerning av bøyningsendelsene (pkt. 1). Likeledes valgte vi ut en "normalform" når det gjaldt paradigmer som inneholdt vokalvekslinger (pkt. 3), konsonantforenklinger/-fordoblinger (pkt. 4) og stavelsessammentrekninger (pkt. 5). Ord som forekom hyppig i forskjellige skriftvarianter (f.eks. fram/frem, nå/nu) ble lagt inn spesielt (pkt. 6).

> De sterke verbene FINNE og VINNE har vokalvekslingene i-a-u. Vi regner infinitivs-/presensformen som normal form og erstatter preteritums- og perfektumsformen med denne, dvs. -ANT og -UNNET strykes og -INN settes i stedet. (For å forenkle reglene strykes alltid e'en når den er utlyd).

Avledningsendelsene ble vurdert i forhold til hvor stor grad de endret det semantiske innholdet av ordet, og i første omgang ble de aller fleste lagt inn i regelsettet med beskjed om at de skulle fjernes (pkt.2). Vi undersøkte også hvilke prefikser (preposisjoner og adverb) som forekom hyppig i sammensetninger med sterke verb (f.eks. INNGÅ) og funksjonsord (f.eks. DERPÅ). Disse ble samlet i 2 forskjellige grupper og skulle hjelpe til å bestemme hvorvidt et ord var et sterkt verb (som måtte nøytralisere vokalvekslingen) eller et funksjonsord (som i de fleste tilfeller skulle beholde den formen det hadde).

De reglene vi så kom fram til, kjørte vi ut på vårt eksperimentmateriale for å kartlegge omfanget av følgende problemer:

1) ord med "falske" endelser
2) ord med større uregelmessighet i rotlemmakomponentene enn hovedreglene dekket
3) uheldige grupperinger

1) En falsk endelse

En falsk endelse er en bokstav eller en bokstavkombinasjon som er en del av stammen, men som har en form som er identisk med en bøynings- eller avledningsendelse. I prinsippet kan alle bøynings- og avledningsendelser ha en "tvilling" som er falsk, men det er stor forskjell på hyppigheten av forekomstene av disse falske endelsene. Eksempler på falske endelser er -EN i LAKEN, -ER i METER, -S i PRIS og -A i KOLLEGA. Hvis målsettingen hadde vært å finne fram til stammen i ordene, måtte disse falske endelsene beholdes. Men med vår målsetting kan vi tillate oss å la disse bli fjernet så lenge oppslagsordet for rotlemmaet (dvs. det som blir igjen av stammen) blir entydig.

> -A'en i utlyd kan være en bøyningsendelse (bestemt form entall hunkjønn eller bestemt form flertall intetkjønn) og skal da fjernes. Men den kan være en del av stammen (eks. KOLLEGA) eller den kan tilhøre et sterkt verb i preteritum (INNLA). Når A'en er del av stammen, burde den beholdes, når den er utlyd i forbindelse med et sterkt verb, burde det sterke verbet forandres til "normalformen" (se over). Dette har vi løst på følgende måte: Hovedregelen er at A'en fjernes i utlyd. At den dermed blir fjernet i ord som KOLLEGA, løser vi ved også å fjerne den i de andre bøyningsformene i ord med -A som siste bokstav i stammen (-AEN, -AER, -AENE i eksemplet med kollega). Så lenge oppslagsordet er entydig, er vårt krav oppfylt. Ved endelser som har en bokstavkombinasjon som kunne tilsi at de er sterke verb, kaller vi opp prefikslisten for sterke verb og sjekker ordets begynnelse mot denne. Det viser seg nemlig at svært mange av de sammensatte sterke verbene nettopp består av en prefiks + verbet (det viktigste unntaket er LEGGE som også danner forholdsvis mange sammensetninger med substantiv). Hvis vi får tilslag på den aktuelle prefikslisten, blir ordet oppfattet som et sammensatt sterkt verb og behandlet deretter. I eksempelet med INNLA ville INN være å finne blant prefiksene for verb, og ordet ville forandres til INNLEGG, mens f.eks. KULA ikke ville få tilslag på prefikslisten og ville følge hovedregelen.

Hvis derimot oppslagsordet for et rotlemma faller sammen med oppslagsord for andre rotlemma, kan ikke de falske endelsene fjernes. I disse tilfellene må vi legge inn spesialregler for å kunne skille mellom de ekte og de falske endelsene.

I noen tilfeller sløyfet vi hovedregelen, og de ordene som hadde denne bokstavkombinasjonen som en virkelig avlednings- eller bøyningsendelse, fikk spesialregler. I andre tilfeller var det ordene med falske endelser som fikk spesialreglene. Metoden vi valgte var alltid den som gav færrest regler totalt sett.

2) **Ord med større uregelmessighet i rotlemmakomponentene enn hovedreglene dekket**

En del ord som klart tilhører samme rotlemma, viser større uregelmessigehet enn våre hovedregler tilsier. Dette gjelder i første omgang fremmedord, lånt fra latin og gresk. I slike tilfeller måtte vi legge inn noen spesialregler som gikk forbi suffiksgrensen og behandlet stammen i ordet.

PRODUSERE og PRODUKSJON tilhører samme rotlemma og skal derfor grupperes sammen. Ved å fjerne -ERE og -SJON ville vi stå igjen med oppslagsordene PRODUS og PRODUK som må tillempes hverandre. I dette tilfellet vil PRODU være en entydig oppslagsform, og vi la inn regler som fjernet S'en og K'en i forbindelse med disse suffiksene.

3) **Uheldige grupperinger**

Uheldige grupperinger er ord som tilhører forskjellige rotlemmaer, men som får felles oppslagsform uten at de i utgangspunktet er homografer. I mange tilfeller dreier det seg om innholdsord som får samme form som et funksjonsord. I slike tilfeller har vi lagt inn og markert de aktuelle funksjonsordene (de tilhører jo en del av ordforrådet som ikke ekspanderer), slik at vi kan skille gruppene fra hverandre.

- E'en i utlyd blir alltid fjernet, og ordet MENE vil derfor få rotlemmaet MEN. For at det ikke skal bli gruppert sammen med konjuksjonen MEN, har vi lagt inn konjunksjonen og markert denne. Dermed får vi skilt de to rotlemmaene fra hverandre.

6. Programsystem for automatisk rotlemmatisering

6.1 Oversikt

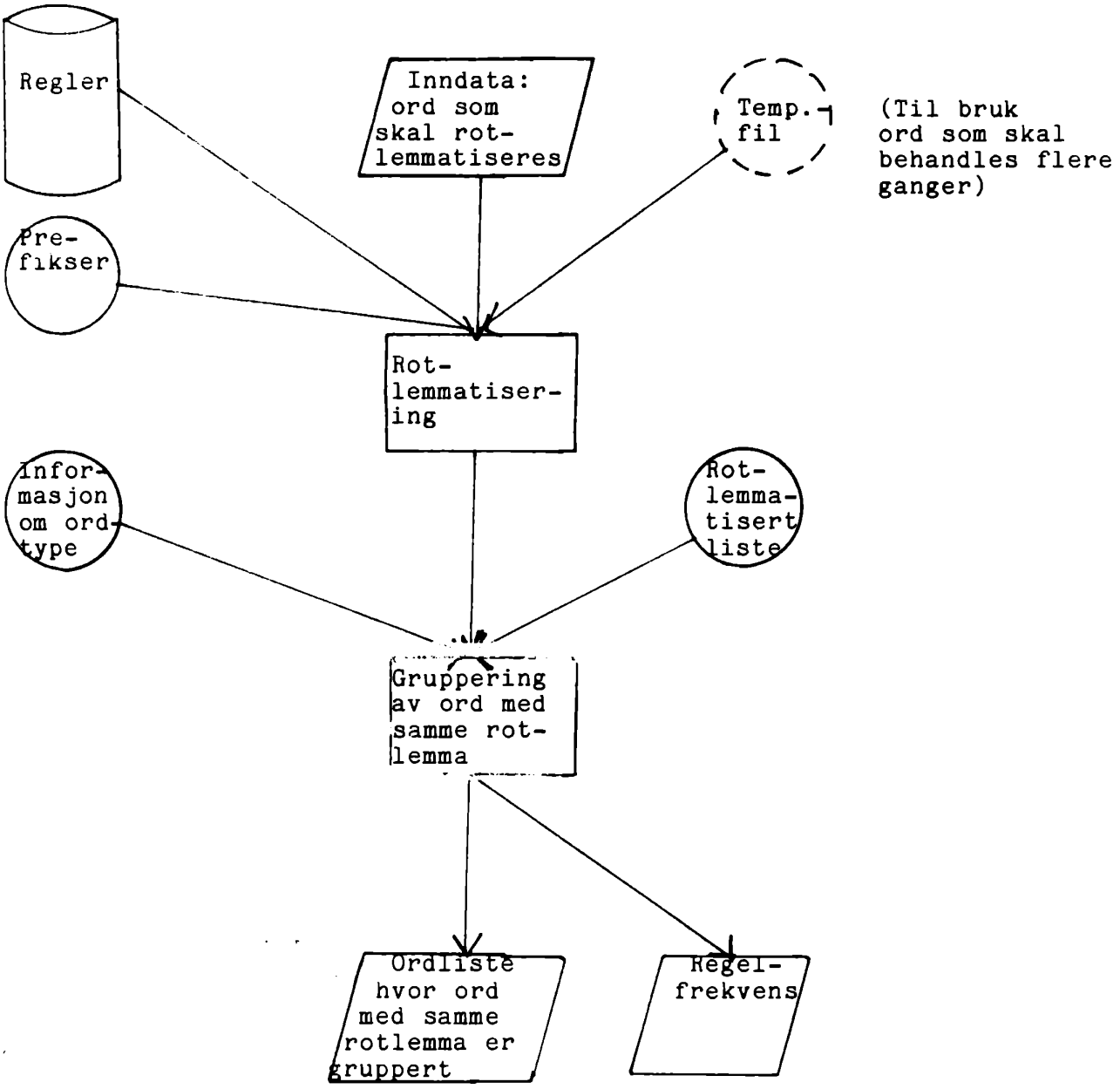
Figuren på neste side gir en oversikt over programsystemet for automatisk rotlemmatisering.

Inndata til programmet er ett eller flere ord, f.eks. en frekvensordliste som i vårt tilfelle. Resultat er en rotlemmatisert liste hvor hvert ord er angitt med rotlemma og en del tilleggsinformasjon, f.eks. hvilke regler som er brukt og om ordet er et funksjonsord. Det siste er nyttig informasjon ved gruppering av ordene fordi vi ikke ønsker å gruppere funksjonsord sammen med andre ord, (jfr. eksemplet med MEN og MENE).

Som del av resultatet får man også en oversikt over hvor hyppig den enkelte regel er brukt. F.eks. endelsen -EN er brukt 3399 ganger, mens -ENE er brukt bare 1681 ganger. Denne informasjon var til stor nytte for oss ved spesifisering av regellisten, og den kan også være interessant for dem som ønsker å studere suffiksene i et datamateriale.

Den rotlemmatiserte listen blir til slutt gitt som "input" til et program som grupperer alle ord med samme rotlemma. Dette programmet beregner også den samlede frekvens for rotlemmaeene.

Oversikt over programsystemet for automatisk rotlemmatisering.



6.2. Spesifisering av reglene

Hver regel inneholder en tegnstreng, en typebetegnelse, en ordre og et krav.

Tegnstrengen består av ett eller flere tegn som skal sjekkes mot ordets høyre del. I enkelte tilfeller kan eller må tegnstrengen utgjøre hele ordet.

Typebetegnelsen angir om ordet f.eks. er et funksjonsord eller et sterkt verb. Denne informasjonen hindrer at bestemte typer ord (f.eks. funksjonsord) blir gruppert sammen med andre ord.

Ordren gir informasjon om hvor stor del av ordet som eventuelt skal fjernes og hvilken tegnstreng som eventuelt skal legges til. Dessuten gir ordren beskjed om ordet skal behandles på nytt etter at ordren er utført (f.eks. ord med genitivs s).

Et generelt krav til rotlemmaet er at det minst må bestå av to tegn, hvorav det ene må være en vokal. I tillegg kan hver regel stille krav til

a) hvor stor del av ordet tegnstrengen skal utgjøre (f.eks. hele ordet eller bare en del av det),
b) ordets begynnelse (f.eks. at det skal være en prefiks),
c) at tegnstrengen ikke er etterfulgt av andre teng (dvs. at ordet ikke har vært behandlet tidligere). I slike tilfeller kaller vi regelen "lukket".

Eksempler på regler og bruken av dem er gitt på neste side.

6.3 Algoritme

Programmet behandler ordet bakfra. Etterhvert som det beveger seg et tegn mot venstre, sjekkes ordets høyre del mot tegnstrengen i regelen. Dette gjentas for hver regel inntil

ordets høyre del finner en regel som passer, eller at alle reglene er sjekket. I det siste tilfellet får ordet et rotlemma som er lik ordet.

Hvis regelen passer - dvs. at tegnstrengen er lik ordets høyre del eller hele ordet - sjekkes først regelens krav. Er disse også tilfredsstilt, utføres ordren. Hvis ikke, fortsetter søkeprosessen nedover regellisten.

I de tilfeller et ord skal behandles på nytt igjen, merkes ordet og legges ut på en temporær fil. Denne filen behandles til slutt som om den var en vanlig inputfil. Det er ingen grenser for hvor mange ganger et ord kan behandles før rotlemmaet er bestemt.

Regellisten er organisert som en kjedet fil for å gjøre søkingen mer effektiv. Dessuten er både inndata og regellisten (tegnstrengene) sortert bakfra i samme rekkefølge, slik at programmet ikke starter øverst på resultatlisten for hvert nytt ord. Det finnes mange alternative søkeprosesser som er mer effektive, og programsystemet BETA ville f.eks. ha vært velegnet i dette tilfellet (jfr. Benny Brodda 1982 "The BETA system", foredrag holdt på COLING i 1982).

6.4 Eksempel på automatisk rotlemmatisering

Vi ønsker å rotlemmatisere VARE og GÅRDEIERNES og forutsetter at reglene på neste side gjelder.

Behandling av VARE:

1. gang passer regel 6 og -E blir fjernet. Resultat: VAR.
   Ordren gir beskjed om at ordet skal behandles på nytt.
2. gang stemmer ordet overens med tegnstrengen i regel 3, men kravene til regelen er ikke tilfredsstilt (jfr. krav c). Søkingen fortsetter og regel 4 passer. Ordet tilfredsstiller kravene og får rotlemmaet VAR.

Regel nr.	Tegn-streng	Type-betegn.	Ordre			Krav a): Tegn-strengen må ut-gjøre ...	Krav b): Sjekk prefiks-liste	Krav c Lukket
			Slett	Legg	Behandl.			
1	S		1		Ja	høyre del av ordet		
2	ER		2		Nei	høyre del av ordet		
3	VAR	sterkt verb	2	ÆR	Nei	hele ordet		Ja
4	R		0		Nei	høyre del av ordet		
5	ERNE	subst. eller verb	2		Ja	høyre del av ordet		
6	E		1		Ja	høyre del av ordet		
7	LA	sterkt verb	1	EGG	Nei	høyre del av ordet el. hele ordet	Ja	Ja
8	A	subst. eller verb	1		Nei	høyre del av ordet		

Behandling av GÅRDEIERNES:

1. gang passer regel 1 og endelsen -S blir fjernet.
   Resultat:GÅRDEIERNE.
   Regelen inneholder ikke typebetegnelse, men gir beskjed om at ordet skal behandles på nytt.
2. gang stopper prosessen ved regel 5 og endelsen -NE fjernes.
   Resultat:GÅRDEIER og typebetegnelse "verb eller substantiv".
   Også denne regelen gir beskjed om at ordet skal behandles på nytt.
3. gang passer regel 2 og -ER fjernes. Ordet får rotlemmaet GÅRDEI og typebetegnelsen beholdes.

7. Resultat

7.1 Oversikt

Eksperimentmaterialet besto av 489.382 løpende ord og 23.890 ulike graford. Av disse ble 685 (2,8 %) ulike graford gruppert feil. Tilsammen utgjorde disse 6.740 løpende ord - dvs. 1,4 % av det totale antall ord i korpuset.

7.2 Typer feil.

Feilene er inndelt i 3 hovedkategorier.

a) "Tunge" feil - (utgjør 2,3 %)
b) "Lette" feil - (" 0,3 %)
c) "Vriene" feil- (" 0,3 %)

Tunge feil har vi valgt som betegnelse på ord som ikke er behandlet i samsvar med de 7 kategoriene i avsnitt 5. Disse kan inndeles i 3 undergrupper.

a1) Rotlemmaet inneholder endelser som burde ha vært fjernet - eller at rotlemmaet ikke er fullstendig. Det siste tilfellet gjelder spesielt avledningsendelser som fører til at ordet får endret sitt semantisk innholdet i forhold til ordstammen, for eksempel:

DEFINISJON	får	rotlemma	DEFINISJON,	mens	ønsket	rotlemma	er	DEFI
SYKT	"	"	SYKT	"	"	"	"	SYK
SMERTE	"	"	SMER	"	"	"	"	SMERT

a2) Rotlemmaet blir for lite og derfor tvetydig, for eksempel:

FETTER får rotlemma FETT, mens ønsket rotlemma er FETTER

SETET	"	"	SE	"	"	"	"	SET
LEVERE	"	"	LEV	"	"	"	"	LEVER

a3) Ordet inneholder stavelsessammensetninger som ikke blir nøytralisert, for eksempel:

USSEL	får	rotlemma	USSL,	mens	ønsket	rotlemma	er	USL
SYKKEL	"	"	SYKKL	"	"	"	"	SYKL

Lette feil omfatter ord hvor avledningsendelsen ikke er fjernet, men hvor endelsen i en viss grad endrer det semantiske innholdet til ordet i forhold til ordstammen. Endringen er allikevel ikke så stor at vi vil beholde endelsen, eksempelvis:

ADRESSAT	er	ikke	gruppert	sammen	med ADRESSE
BILIST	"	"	"	"	" BIL
GARANTIST	"	"	"	"	" GARANTI, GARANTERE

Vriene feil gjelder ord som har så avvikende skrivemåte i de ulike former, at det kreves spesialbehandling i hvert enkelt tilfelle, eksempelvis:

KONTO og KONTI
EPILEPSI og EPILEPTISKE
FØDSELSDATO og FØDSELSDATUM

7.3 Homografer

Målsetningen for den automatiske rotlemmatiseringen omfattet ikke homografseparering (jfr. også pkt. 3.3). Ord med felles rot er derfor plassert i samme gruppe, selv om de har avvikende innhold. Dette har vi ikke regnet som feil. Det samme gjelder selv om ordet i sin bøyde form er entydig, f.eks.:

RETTSLIG som har fått rotlemmaet RETT (homograf en (varm)rett)

IRI

MUNNING " " " " MUNN (" " munn)

7.4 Rotlemmatisering av et tilfeldig valgt materiale

For å få en så generell bedømmelse av den automatiske rotlemmatiseringen som mulig, ble regelsettet helt til slutt utprøvd på et tilfeldig valgt materiale. Vi valgte barneboken Ole Brumm som består av 19725 løpende ord og 2.369 ulike graford.

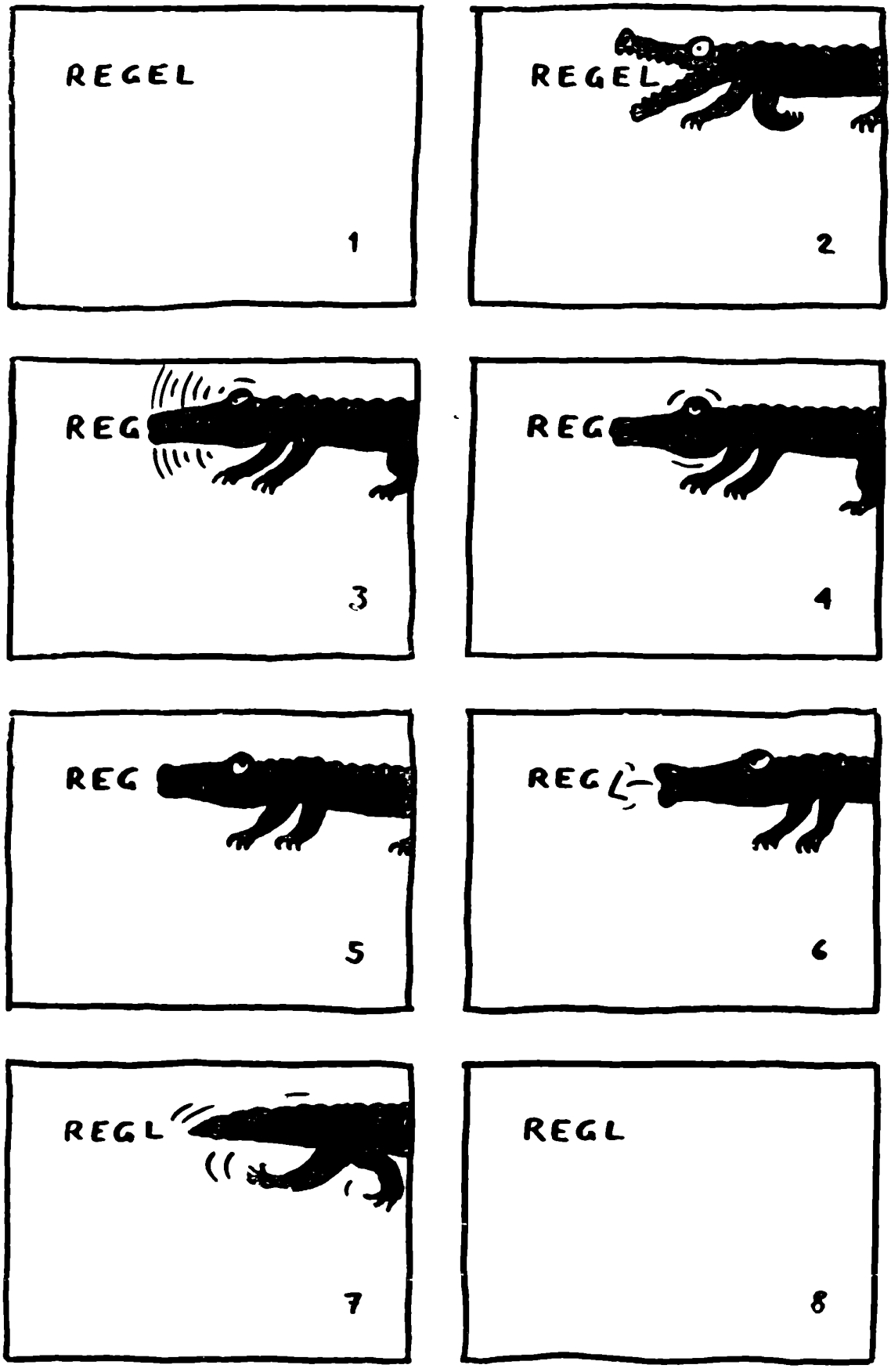
Resultatet viste at ca. 97,6 % av de ulike ordene ble plassert i riktig gruppe.

8. Videreføring

Språkbruken i tekster fra forskjellige fagområder kan variere sterkt, og et regelsett som har som mål å være tekstuavhengig må nødvendigvis bli ganske omfattende. Dessuten vil kravet til korrekthet variere alt etter hva rotlemmatiseringen skal brukes til. I de to konkrete problemstilligene vi tok utgangspunkt i, var korrekthetskravet forskjellig. I det førstnevnte - tekstsøkingsproblemet - er det viktig at de ordene som blir vurdert som søkeord blir gruppert riktig, mens det ikke gjør noe om det forekommer feilgrupperinger blant støyordene. Den andre problemstillingen - utskillingen av høyfrekvente ord - krever derimot at alle typer ord blir gruppert riktig. Hvis ikke ville frekvensen forskyve seg, og man vil ikke finne fram til de gruppene man ønsket. Man kunne derfor tenke seg at et regelsett ble delt inn i forskjellige lag eller pakker. Utgangspunktet kunne være en bunke med basisregler, og man kunne tilføre lag med spesialregler som gav større korrekthet. Dessuten kunne man tenke seg spesielle fagpakker eller genre-pakker, f.eks. en juss-pakke, en fysikk-pakke, en medisin-pakke eller en språkkonservativ-pakke og en språkradikal-pakke. På denne måten kunne man få et regelsett som var tilpasset både teksten og bruken.

En del av reglene er markert med hensyn til ordklassetilhørighet. Alle reglene er markert enten som funksjonsord eller som innholdsord. Denne markeringen er først og fremst lagt inn for å skille mellom ord som ellers ville bli gruppert sammen. Dessuten har det vært en hjelp under arbeidet å vite hvilke ord en regel er ment å dekke. Dette arbeidet kunne lett utvides, og en automatisk markering av ordklassetilhørighet skulle være innen rekkevidde.

ALLIGATORISK ROT-LEMMATISERING



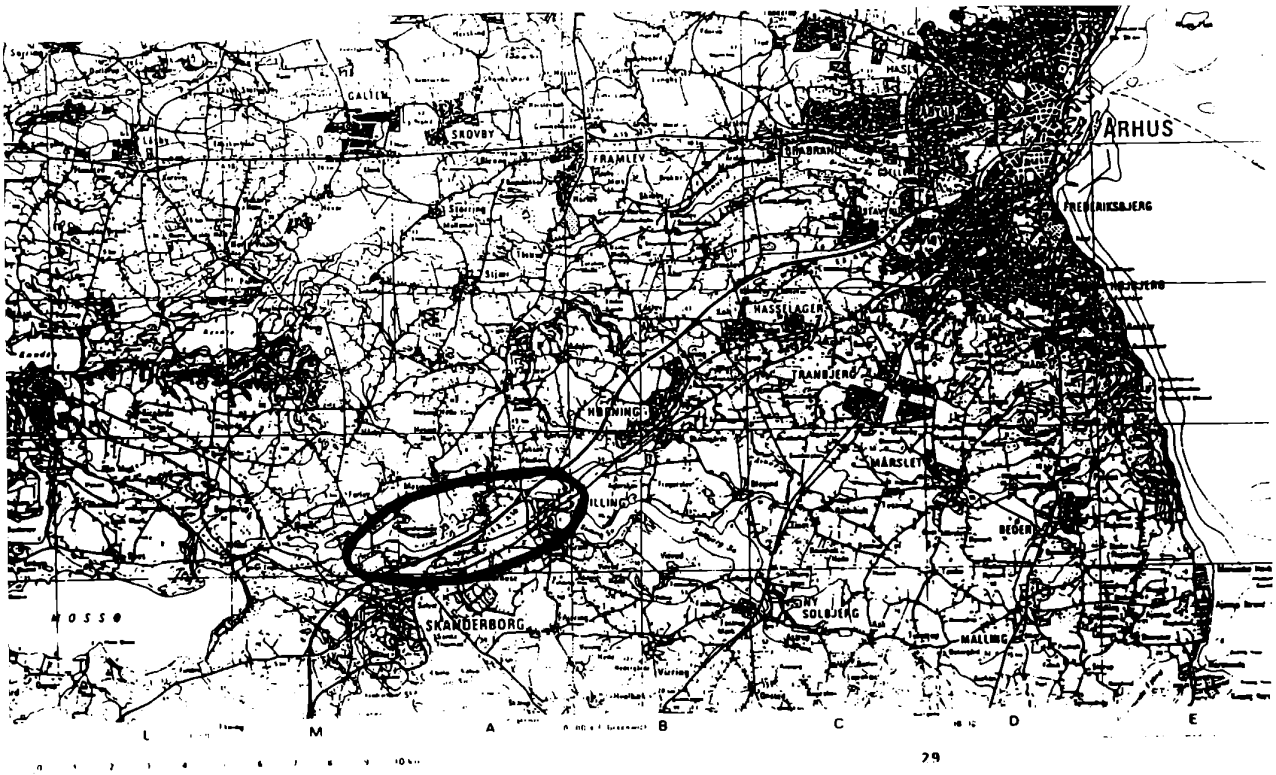
Henrik Holmboe
Institut for Lingvistik
Aarhus Universitet
Otto Rudsgade 67-69
DK-8200 Århus N

KONKORDANS OVER DE DANSKE RUNEINDSKRIFTER

De danske runeindskrifter rummer en meget rig mængde oplysninger af både sproglig og historisk art. Den sproglige form bevirker imidlertid, at runeindskrifterne kan være vanskelige at bruge som kildemateriale for historikere eller arkæologer. Det, der kan være et nyttigt eller endog et tilstrækkeligt hjælpemateriale for filologer, kan for den sprogligt mindre trænede historiker være utilstrækkeligt. Man kan altså nærme sig runeindskrifterne enten med henblik på runologiske studier eller med henblik på vikingetidsstudier, og det viser sig, at man får brug for redskaber af lidt forskelligt tilsnit afhængigt af, om man vælger den ene eller den anden tilgang.

Størstedelen af indskriftmaterialet, der udgør henved 700 indskrifter dækkende tiden fra ca. 200 til ca. 1500 forefindes i meget velpubliceret form. Hovedværket er stadigvæk Lis Jacobsen & Erik Moltke "Danmarks Runeindskrifter" fra 1941-42, i hvilket samtlige frem til 1941 fundne indskrifter er medtaget. Dette værk blev i 1976 fulgt op af Erik Moltke med bogen "Runerne i Danmark og deres oprindelse", som medtager alle runeindskrifter, der er fundet indtil da.

Omend sjældent finder man stadigvæk nye runeindskrifter, f.eks. på genstande, som fremdrages ved arkæologiske udgravninger. De omfattende udgravninger af moseofferfundet i Illerup Ådal syd for Århus har bragt nogle genstande for dagen; Illerup Ådal rummer stadig ifølge arkæologerne umådelige mængder af genstande, så det vil ikke være overraskende, om der også i fremtiden skulle vise sig nye runeindskrifter derfra.



Illerup Ådal

De runeindskrifter, der er fundet efter 1976 - altså efter Moltkes bog - er publiceret i diverse tidsskrifter eller henligger evt. upublicerede, men det er stadigvæk kun meget få indskrifter, der ikke er medtaget i en samlet og ensartet udgivelse.

Hvad angår den leksikografiske behandling af runematerialet, er afsnittet om ord- og navneforråd i Danmarks Runeindskrifter stadig eneherskende; her finder man en ordbog af næsten klassisk tilsnit med opslagsord, morfologisk karakteristik, oversættelse, enkelte belægsteder anført og henvisninger til den videnskabelige litteratur, dersom det pågældende ord har været genstand for særskilt behandling. Forud for Danmarks Runeindskrifter bør også nævnes Ludv. F.A. Wimmer: De danske Runemindesmærker, Håndudgave ved Lis Jacobsen fra 1914, som rummer en ordbog med henvisning til belægsteder. Imidlertid er Wimmers udgave ikke komplet, hvorfor ordbogen heller ikke er det.

***?driugR**, adj.* se triu. . . .

***drotning**, f. 1*) (til *drottin* „herre") 'dronning', husfrue, frue[1]. **rhafnukatufi hinu runaR þasi aft þurui trutnik sina** 26 Læborg. *2*) kvinde af fornem stand, gift med eller ætling af *drottin*[2]; som titel gengives ordet bedst ved frue. **qsur lathirþiR . . raist runar þasi at qsbuþ trunik** 134 Ravnkilde 1 (sv. påvirket). — *Akk. sg.* **trutnik** 26, **trunik**[3] 134.

Anm. 1. Se sp.52 samt *Wimmer*.DRM.II p.51f.; *JohSteenstrup*.FestskrErslev (1927) p.70–75 antager, at ordet betegner landets dronning (se også bibl. til 26 Læborg); herimod *RegnarKnudsen*.AarbAarhus (1932) p.235–37.

Anm. 2. Jfr. Heimskringla, hvor ordet anvendes om 'kvinde af kongeslægt' (*Fritzner* bet.4). Se også *Heinertz*.Drottning und Käring, APhS.X (1935–36) p.145–58. Formelt kunde ordet på Ravnkilde-st. tolkes som landets dronning (således *JohSteenstrup* l. c.), men nogen dronning Asbod kendes ikke, og der savnes holdepunkt for at antage, at Asser 'landbestyrer' i Ravnkilde skulde have ristet runer over landets dronning.

Anm. 3. **trunik** snarest ristefejl, muligvis dog assimilation, jfr. *Brøndum-Nielsen*.GldaGr.II p.227.

***drottin**, m. 1*) „drot", herre. **qsur sati stin þqnsi aft ualtuka trutin sin** 131[1], 209[1], **tuka kurms sun saR hulan trutin** 295[1]. *2*) om jætternes hersker, fyrste. **þur uigi þik (þ)orsa trutin** 419[a]. *3*) [jfr. *dominus*] herren, om Kristus. **kuþ tr(u)tin**[a] 380, **uar drotin** 373 (gotl.). — *Nom. sg.* **tr(u)tin** 380 NyLars 2, **drotin** 373 Åker (gotl.); *akk. sg.* **trutin** 131 Års, 209 Glavendrup, 295 Hällestad 1, 419 Canterbury, **drotin** 373 Åker (gotl.); *dat. sg.* **drotni** smst. (2 g.).

Anm. 1. Ordet er i disse tre indskr. — svarende til brugen i samtidige skjaldekvad om konger — anvendt om en høvding (fyrste) i forhold til hans mænd, men ikke som titel, se *JohSteenstrup*.FestskrErslev (1927) p.70–76. Anvendelsen på 209 stemmer med, at hovgoder ifig. Snorre kaldtes *drótnar*, jfr. også m. h. t. Alles rang *Moltke*.APhS.VII (1932–33) p.92. Se endvidere *Heinertz*.APhS.X (1935–36) p.161.

Anm. 2. Samme udtryk forekommer også i Þrymskviþa og i Sigtuna-indskriften, se *ILindquist*.Religiösa runtexter I (1932) p.26f., 31f.; *HSperber*.BeitrDSprLit. XXXVII (1912) p.155 opfatter **þorsa trutin** som nom., apposition til **þur**.

Anm. 3. Guþ drottin findes på runestene fra Uppland: *Liljegren* no.28, *Brate*.Svenska runristare (1925) p.61, 91, fra Vester Götland: *Torin* no.32, jfr. no.11.

Eksempel fra ordbogen i Danmarks Runeindskrifter

dróttinn Masc. (Nom. trutin 149, Acc. trutin 19, 80, 109), „Drot", Herre; om Høvdingen: 19, 109, om Husets Herre: 80, om Gud Herren: 149.

dróttning Fem. (Acc. trutnik 7, trunik 22), Herskerinde, Frue. I Betydning „Dronning" forekommer Ordet ikke i danske Runeindskrifter.

Eksempel fra ordbogen i Wimmer

Moltkes bog fra 1976 bidrager ikke med noget leksikografisk. Stikordsregistre med udtømmende henvisninger ordnet emnevis er der ingen af publikationerne, der har. Bedst også i denne henseende er Danmarks Runeindskrifter, der rummer et realleksikon.

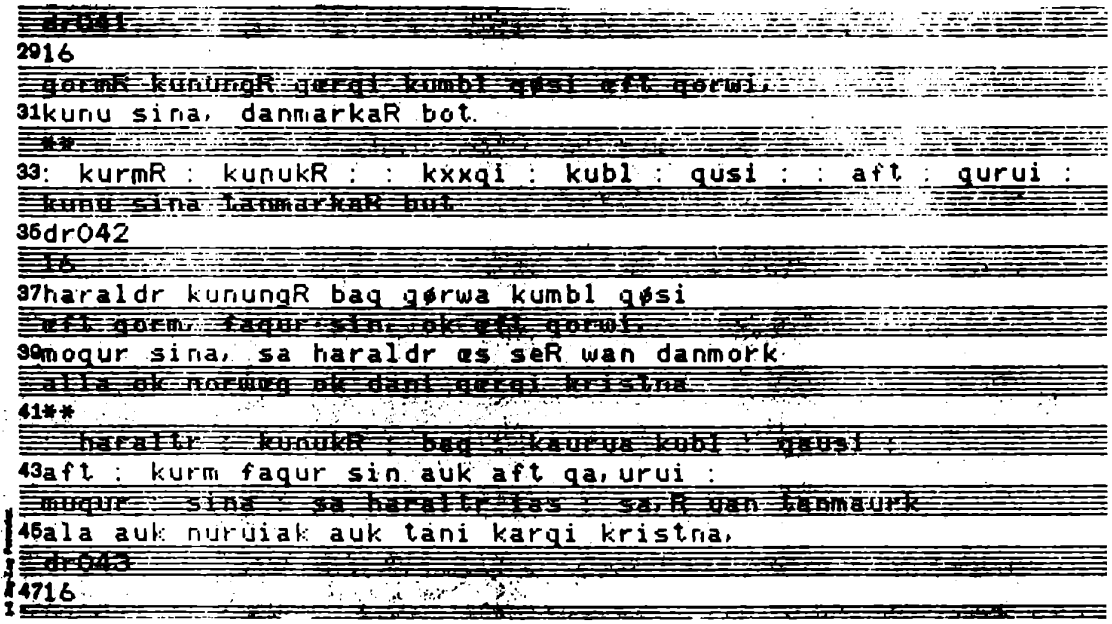
På denne baggrund blev planen om at fremstille konkordanser over de danske runeindskrifter til. Konkordanser er efterhånden blevet så velkendte hjælpemidler, at det må være unødvendigt at gentage, hvilke fordele de rummer. Det er heller ikke overraskende at fremstille konkordanser ved hjælp af

elektronisk databehandling; men dermed er ikke sagt, at det er trivielt at gøre det. I forbindelse med behandlingen af runeindskrifterne dukkede der således problemer op, hvis løsning ikke blot kunne overføres fra andre tilsvarende leksikografiske arbejder. Et af dem var, at der ikke eksisterer en simpel nummerering af og dermed referencemulighed til de danske runeindskrifter. De indskrifter, der er udgivet i Danmarks Runeindskrifter, refereres ofte med numrene i dette værk. Men de indskrifter, som Moltke har publiceret, og som ikke findes i Danmarks Runeindskrifter, har ikke noget nummer. I det hele taget møder man ofte en helt speciel, meget indforstået referencemåde til runeindskrifterne såsom: Ribe lyfstav, Torsbjerg skjoldbule, Kragelund døroverligger, Gallehus guldhorn 2.

I konkordanserne har vi valgt at referere til de indskrifter, der findes i Danmarks Runeindskrifter, med DR - efterfulgt af det pågældende nummer, og til Moltkes indskrifter med M efterfulgt af sidetallet i Moltkes bog ganget med 10 plus et étcifret tal, der angiver, hvilken runeindskrift på den pågældende side, der refereres til.

Overførslen af runeindskrifterne til maskinlæselig form er foregået ved en terminal og i det såkaldte BIB-SYS-system på RECAU, et system, der umiddelbart muliggør strukturering af de enkelte runedokumenter i forskellige poster. Hver enkelt indskrift er indskrevet med referencenummer, klassifikation efter alfabettype (16 eller 24 tegns futhark) transskription, hvis det har været muligt at etablere en sådan, samt translitteration. Ved translitterationen tilstræbes en én-til-én-overensstemmelse mellem de originale runeindskrifters tegn og gengivelse i publikationen. I den transskriberede form gengives indskrifterne i det, som Erik Moltke har kaldt "en slags normaliseret vikingetidsoldnordisk". Herved normaliseres indskrifterne, forskelle i ortografi og tegninventar sløres, men dette normaliserede niveau har vi fundet var et nyttigt udgangspunkt for en konkordans, som først og fremmest skulle tjene realdisciplinerne som et emne- og sagregister. Translitterationsniveauet vil derimod snarere henvende sig til de

runologisk interesserede. Fordelen i denne sammenhæng ved at benytte edb er, at dette gør det praktisk muligt og overkommeligt at vælge en både-og-løsning, altså tilvejebringelse af både en transskriptionskonkordans og en translitterationskonkordans udgivet i ét og samme år.



Eksempel på indskrift

Den maskinlæsbare version af runeindskrifterne, vi har lavet, følger Lis Jacobsens og Erik Moltkes udgave meget nøje; vi har ikke set det som vores opgave i denne forbindelse at præstere nye læsninger eller nye tolkninger. Den eneste gennemgående afvigelse fra udgaverne i de trykte konkordanser er, at udgavernes større eller mindre antal prikker til markering af større eller mindre ulæselige områder af indskriften er erstattet af en standardlacune på tre prikker.

Med hensyn til formatet stod vi overfor valget mellem en såkaldt KWIC-konkordans (keyword in context) eller en KWOC-konkordans (keyword out of context). Efter at have studeret udskrifter i såvel KWIC som KWOC stod det klart, at en KWIC-konkordans var den mest velegnede til nærværende formål. KWIC-konkordansens store fordel ligger netop deri, at man umiddelbart kan læse søgeordet i den rette tekstmæssige sammenhæng. Dette mere en opvejer KWOC-formatets måske lidt større overskuelighed.

Konkordansen er opbygget på den måde, at søgeordet skal findes midt på siden, til venstre for dette læses den forudgående og til højre den tekst, der følger efter søgeordet. Yderst til venstre er indskriftens identifikation angivet. DR efterfulgt af et nummer markerer, at indskriften stammer fra Lis Jacobsen og Erik Moltke "Danmarks Runeindskrifter". DRbr og DRm ligeledes efterfulgt af en nummerangivelse markerer, at der er tale om henholdsvis en brakteat og en mønt publiceret i "Danmarks Runeindskrifter". M efterfulgt af et femcifret tal angiver, at indskriften er hentet fra Erik Moltke "Runerne i Danmark og deres oprindelse". Denne nummerering, der er forfatternes egen, gør det muligt via nummeret at finde frem til den relevante indskrift i Moltkes bog. F.eks. findes M0780 på s.78 i bogen, nullet til sidst i nummeret fortæller, at det er den eneste indskrift på denne side. Dertil kommer et fåtal af indskrifter hentet fra andre publikationer, disse har numrene M001 - M007.

DR379	16		sazur	
		let resa sten aftiR aiwarb, faþur sin,		
		druknaþi han uti með alla skipara. andgi		e kristr hialpi sioiu hans. sten þessi stai aftiR.
DR305	16		•	eadrinc.
DR318	16		brændr	
		garþi kumbi þesi aftiR gubba, faþur sin, ok		efi, felagi gubba.
DR354	16			efi ok þorgotr þeR lagþu sten yftiR þorlaks gub hialpi sol hans.
DR367	16			ego sum lapis.
DR012	24			ek hlewagastiR holtijaR horna tawido.
DR196	24			ek erilaR asugisias ... haite ... ginnu ... ... wigiu ...
DR261	24			ek erilaR sa wilagaR haiteka ... alu
DRBR45	24			ek fakaR fahido.
DRBR64	24			ek fakaR fahido.
M0990	24			ek unwod ...
DR202	16		soti satti sten þansi aft	elef, broþur sin, sun asgots røþumskialda.
DR373	16	þitta iR santi gabrel ok segþi santa		
		maria, at han skuldi barn fyþa. þitta iR		elizabeþ ok maria ok hailsas. þiar hwilis maria, sum han barn fyddi, skapera himinz ok iorþaR, sum os leysti. þitta iRu þaiR briR kununga R, sur fyrsti giarþu offr warum drotni. þiar tok han wiþr kununga offri, war drottin. þiar riþu þaiR burt briR kununga R, siþan þaiR offrat hafa orum drotni. ... þet þiar fraa ... loþaR toku warn drottin ok .....n wiþr fre ok gettu. siþan faddu þaiR han burt þiaþan bundin, ok negidu þiar loþaR iesus a krus. si fraa a þitta. sigrafR mesteri.
DR398	16		broþiR ok	emundr þeR letu resa sten þanna aftiR sigmund, faþur sin. kristr hialpi siaiu hans ok santa michael ok santa maria.
DR411	16		steinn sasi	
		es sattr eftir sibba ... goþa, son foldars.		en hans liþi sandi at ey ... folginn liggr hinns fylgþu - flestr vissi þat - mestar dąþir dolga brubar draugr i þelmsi haugi munat reiþ-viþurr raþa rogstarkr i danmarku endlis larmungrundar grgrandarl landi.
DR353	16		hiar liggar	enar arnbiornar sur. guþ.

Eksempel på transskriptionskonkordans

Til højre for identifikationen og altså umiddelbart inden selve indskriften angives, hvorvidt den pågældende indskrift tilhører 24-tegns eller 16-tegns futharken. Enkelte indskrifter forekommer i dobbeltfortolkning, altså både i en a- og en b-udgave. Denne fremgangsmåde er valgt, når der har været tvivl om, hvorvidt en indskrift hører hjemme i 24- eller 16-tegns futharken.

De videre perspektiver for edb-arbejdet med runeindskrifterne går i to retninger: dels er det nærliggende at fortsætte med at indskrive og lave konkordanser også for de runeindskrifter, der hører til uden for Danmark, dels vil det nu være muligt at underkaste det danske materiale en række kvantitative undersøgelser og sammenligne dem med tilsvarende data fra senere sprogtrin. Hvornår disse planer kan omsættes til virkelighed, afhænger af bevillinger af mandskab og penge; jeg håber, at ressourcerne viser sig, men nogen tidsplan tør jeg ikke angive.

Den fuldstændige reference til de nævnte runekonkordanser er:

Andersen, Ingeborg & Henrik Holmboe:
Konkordans over de danske runeindskrifter
-Transskription ISBN 87-981240-8-0
-Translitteration ISBN 87-981240-1-3
Forlaget Arkona, Århus, 1983.

Bibliografi:

Jacobsen, Lis og Erik Moltke: Danmarks Runeindskrifter. København 1941-42.

Moltke, Erik: Runerne i Danmark og deres oprindelse. København 1976.

Wimmer, Ludvig F.A.: De danske Runemindesmærker. Haandudgave ved Lis Jacobsen. København 1914.

Harri Jäppinen, Aarno Lehtola,
Esa Nelimarkka, and Matti Ylilammi
Helsinki University of Technology
Espoo, Finland

KNOWLEDGE ENGINEERING APPLIED TO MORPHOLOGICAL ANALYSIS[1]

Introduction

We are currently designing a data-base interface for queries in natural Finnish. As is well known, Finnish is a highly inflectional language. Consequently, in data base applications as well as in other natural Finnish processing systems morphological analysis of word forms constitutes a fundamental computational subproblem. This paper describes our solution to the problem.

Our model is intended for the analysis of Finnish word forms. The system performs all meaningful morphotactic segmentations for a given surface word form, transforms alternated stems into the basic form (sg nominative or 1st infinitive for nominals and verbs, respectively), and matches the stems against lexical entries in order to find the meaninful words. The present version of the system does not analyze compound word forms into their constituents, nor does it analyze derivational word forms. We are building a new version which will have some of these characteristics. Otherwise model is complete: it has been fully implemented, and tests indicate its correctness so far to lie in the neighborhood of 99.5 % (Jäppinen et al., 1983).

Other models have been reported. Brodda and Karlsson (1980) attempted to find the most probable morphotactic segmentations for Finnish word forms without a reference to a lexicon. They report close to 90 % accuracy. Sågvall-Hein (1978) reports an attempt to apply the Reversible Grammar System to a subclass of Finnish morphology. Karttunen et al. (1981) and Koskenniemi (1983) report two distinct and complete models. Both systems first search in a lexicon all words whose roots match with a given input word form and then prune the ones which could result in the input word form. The latter model is symmetric: it analyzes as well as generates Finnish word forms.

---

[1] This research is supported by SITRA (Finnish National Fund for Research and Development), PL 329, 00121 Helsinki 12, Finland

Due to the computational environment of our model, data-base interface, we set forth the following three design objectives for our analyser: 1) analysis should be efficient, 2) all valid interpretations of an input word form should be found (in the context of a given lexicon), and 3) the addition of new lexical entries should be easy. The last goal, a human engineering viewpoint, suggested us to minimize morphological information in lexical entries and, instead, store morphological data maximally into active knowledge sources.

Heuristic morphology

We approached the problem of morphological analysis from the vantage point of heuristic search. Heuristics has been used for many years in artificial intelligence problem solving situations. In more recent research heuristics has been expanded into multi-level search, and novel control strategies have been developed to govern the process. Examples are speech understanding (Erman et al., 1980) and many so called expert systems.

Morphology is a much more constrained task domain than speech understanding, mass spectrometry, medical diagnosis, and other typical expert system applications. We, however, decided to study how multi-dimensional heuristic search applies to morphology. We do not use heuristic search because algorithmic methods would not apply (they do, as Karttunen et al. (1981) and Koskenniemi (1983) have demonstrated). Our argument for using heuristic rules was to see if we could get a faster method which would distribute most morphological knowledge in active rules.

We knew that we must be on guard against risks involved: heuristics might sometimes erroneously generate wrong interpretations. Such dangers, however, did not materialize.

Finnish morphology briefly visited

A brief informal and incomplete statement of Finnish morphology is described below. We have presented a bit more detailed statement elsewhere (Jäppinen et al., 1983). An interested reader is adviced to consult Karlsson (1981) for a thorough exposure.

The surface form of a Finnish nominal is composed of the following constituents (parentheses denote optionality):

(1) root + $ + number + case + (possession) + (clitics).

Root denotes the unvarying head part of a word stem, and the stem ending ($) its alternating tail. The root may, however, vary under consonant gradation process. Finnish nominals appear in 14 cases: nominative, genetive, partitive, essive, translative, instructive, abessive, ablative, and allative. Genetive, partitive and illative are more irregular than the others in that they realize in more than one allomorphs. Plural is indicated by a 'i', 'j', 't' or a null string (ø) depending on a context.

To visualize, the Finnish word forms 'takissaniko' (= in my coat?) and 'takkeihisihan' (= into your coats!) are segmented as (notice consonant gradation k - kk):

(2) tak + i + ø + ssa + ni + ko
takk + e + i + hi + si + han

The constituent structure for verbs is

(3) root + $ + conjugation + person + (clitics) .

Verbs in Finnish have nominal forms: 5 infinitives and 2 participles in the active voice, and 1 infinitive and 2 participles in the passive voice. These nominal verb forms may receive some but not all cases, and the 1st participle may participate in the adjective comparation process. Most nominal forms may receive a person and a clitic segment in the standard way.

A comparative adjective in Finnish is indicated by the suffix 'mPA' and superlative by 'in' or 'imPA' where 'P' participates in the consonant gradation process (P -> p or P -> m) and 'A' may realize as an 'a', 'ä', 'i' or a null string depending on number and case. The stem ending between a root and a comparation segment is identical to that of the singular genetive case ($$_{gen}$$) for the comparative forms and the plural essive case ($$_{ESS}$$) for the superlative forms.

(4) root + $$_{gen}$$ + mPA + number + case + ...

root + $$_{ESS}$$ + $$\begin{matrix} imPA \\ in \end{matrix}$$ + number + case + ...

The heuristic model

The basic control structure of MORFIN, as we call the model, employs the hypothesis-and-test paradigm as follows. A global data base is divided into four levels: surface word form (SWF), morphotactic (MT), basic word form (BWT) and confirmation (C) levels. Between these levels active knowledge sources, production rules, progressively generate and test hypotheses.

Between the surface word form level and the morphotactic level morphotactic productions (MPs) produce hypotheses of possible segmentations and interpretations of an input word. They leave alternated stems untouched. Stem productions (SPs) produce inverse transformations of the variant stems into canonical basic word forms. For nominals we use the singular nominative case and for verbs the 1st infinitive as the basic form. Some stems get rejected in this second phase as impossible alternations. Dictionary look-up finally tests the proliferated hypothetical basic word forms against the existing lexicon. Morphologically ambiguous words result in multiple confirmation level entries, if the words exist in the lexicon.

Figure 1 portrays the levels and an example analysis of an ambiguous surface word form. The numbers in parentheses are pointers to the previous level. '*' indicates a confirmed hypothesis; 'VA' and 'HA' are postulates for the strong (or neutral) and weak (or neutral) grades, respectively.

SWF-level:	VOIMINA	
MT-level:	1. VOIMINA=	N SG nom
	2. VOIMINA=	V akt imper pr y 2 p
	3. VOIMI= NA=	N SG ess
	4. VOIM= I=NA=	N PL ess
	5. VOI= M= I=NA=	N akt III inf PL ess
BWF-level:	1. VOIMIN A	VA (1)
	2. VOIMIN AA	HA (2)
	3. VOIM I	VA (3)
	* 4. VOIM A	VA (4)
	5. VOI N	VA (4)
	6. VOI MI	VA (4)
	7. VO IDA	VA (5)
	* 8. VOI DA	VA (5)
C-level:	VOIMA - TR FORCE, N PL ess	
	VOIDA - TR CAN, N akt III inf PL ess	

Figure 1. The analysis of "voimina".

Morphotactic knowledge

Morphotactic knowledge in MORFIN is embedded in the MPs. The morpheme productions recognize legal morphological surface-segment configurations in a word and slice the word accordingly. The recognition proceeds from right to left up to the stem boundary as in Brodda and Karlsson (1981). That is, for nominals, verbs and adjectival comparation forms morphotactic segmentations are done from clitics up to (but excluding) the stem ending.

We use directly the allomorphic variants of the morphemes. Since possible segment configurations overlap, several mutually exclusive hypotheses are usually produced on the morphotactic level.

We dressed the morphotactical knowledge of MORFIN into context-sensitive rules. The condition part in a production recognizes a single valid morphotactic segment. To prune search we attached up to two left contextual graphemes in a condition:

(5) name: $(\&_{i2})(\&_{i1})$ $segment_i$
--> POSTULATE([$interpretation_i$],[next])

Here $\&_{i1}$ and $\&_{i2}$ describe sets of allowed graphemes in a word to the left of the segment. A recognition leads into the removal of the segment. The (partial) interpretation is recorded and control proceeds to look for the subsequent consistent segmentations (indicated by 'next').

As an example the production

(6) Rpp14: (ALL-[i:,O:])(V+Ý)n
--> POSTULATE([verb,active,ind,sg1,no_clitics],[ten1]),

recognizes the 1st person singular verb suffix if a word ends with an 'n', has an ordinary or stressed vowel next to the left and any letter except a long 'i', 'o' or 'ö' second to the left. The partial interpretation of a 1st person singular verb in active voice with no clitics is recorded. Control proceeds then to the collection of rules named 'ten1' which recognizes modal and temporal morphemes.

In specifying MPs we found Brodda and Karlsson (1981) a useful source. The morphotactic knowledge comprises currently 201 distinct MPs. To facilitate efficient processing we compiled the MPs into 32 distinct state transition automata (3 for clitics, 1 for person, 5 for tense, 3 for case, 2 for

number, 3 for passive, 5 for participle, 5 for comparation, and 5 for infinitive segments). 'next' in (5) hence indicates legal successor automata for the separated morpheme. Only segmentations and partial interpretations which comprise consistent total interpretations, which end up with an expected stem boundary, get postulated on the MT-level.

To define an abstract morphotactic automata (MTA) let {...}, [...] and <...> denote context, continuation and interpretation formulas, respectively. A concatenated expression {...} [..] <...> is a termination formula. Let § stand for a termination formula or sequence of them. A sequence of termination symbols, preceded by an optional word from the alphabet £={a, b, ... , A, B, ...} is a valid morphotactic automaton $:

(7) $ --> §
$ --> £+§ .

Complex MTAs are generated by the dotted pair

(8) $ --> ($. $) .

To illustrate MTAs below is the automaton PP for possessive and person in list notation (including (6)):

```
(9)  (n{(V+V́)}(ALL-[i:,0:]) [TEN1<VERB,ACT,IND,S1P>
       (V{(V-[U:])(d,k,l,n,r,s,t,)} [obl,TEN3,PAS2]<VERB,ACT,IND,S3P>
          {(0:)(k)} [obl,TEN3]<VERB,ACT,IND,S3P>
          {(A:)(ALL)}[obl,CA3]<3P>)
        t{(V+V́)(ALL-[i:,0:])} [TEN1]<VERB,ACT,IND,S2P>
        (A(v{(V+V́)(ALL-[i:,0:])} [TEN1]<VERB,ACT,IND,P3P>
         V{(0:)(k)} [obl,TEN3]<VERB,ACT,IMPER,P3P>)
        V{(V:-[e:])(ALL)} []<VERB,ACT,IND,S3P>
        A(s(n{(V+V́)(ALL)} [PAR1,PAR4,INF3,INF4,KOMP2]<GENOM,3P>
             {(V+V́)(ALL)} [obl,CA1]<3P>))
        e(m(m{(V+V́)(ALL)} [obl,CA1]<P1P>
             {(V+V́)(ALL)} [PAR1,PAR4,INF3,INF4,KOMP2]<GENOM,P1P>
             {(V+V́)(ALL-[i:,0:])} [TEN1,TEN4]<VERB,ACT,IND,P1P>)
          n(n{(V+V́)(ALL)} [obl,CA1]<P2P>
             {(V+V́)(ALL)} [PAR1,PAR4,INF3,INF4,KOMP2]<GENOM,P2P>)
          t(t{(V+V́)(ALL)-[i:,0:])} [TEN1]<VERB,ACT,IND,P2P>))
        i(n{(V+V́)(ALL)} [obl,CA1]<S1P>
```

$$\left\{(V+\acute{V})(ALL)\right\}[PAR1,PAR4,INF3,INF4,KOMP2]<GENOM,S1P)$$
$$s\left\{(V+\acute{V})(ALL)\right\}[obl,CA1]<S2P>$$
$$\left\{(V+\acute{V})(ALL)\right\}[PAR1,PAR4,INF3,INF4,KOMP2]<GENOM,S2P>))$$

'V' stands for vowels, 'Ý' for the stressed vowels, ':' a long vowel, 'A' for an 'a' or 'ä', 'O' for an 'o' or 'ö', 'U' for a 'u' or 'y', and 'ALL' for any letter.

Knowledge of stem behavior

Stem productions, SPs, describe inverse transformations of stems to their canonical basic forms (the nominative singular case or the 1st infinitive). Stem productions are case-, number-, genus-, mood- and tense-specific heuristic rules which postulate canonical basic forms which in the context of the given morphotactic segmentations might have resulted in the observed variant stem forms. The rules may reject a candidate stem form as an impossible transformation, or produce one or more basic form hypotheses.

Heuristic knowledge of stem behavior is dressed into a set of productions of the following form:

(10) condition --> POSTULATE(cut,string,shift,grade)

If the condition in a production is satisfied, a hypothesis in the canonical form is postulated on the BWF-level by cutting the variant stem ('cut'), adding a new string ('string') as a canonical ending (separated by a blank), and possibly shifting the blank ('shift'). 'Grade' postulates gradation for the stem: 'HA' signals the weak (or neutral) and 'VA' the strong (or neutral) grade.

A well-formed condition (WFC) and its truth value is defined recursively as follows. Any lower case letter in the Finnish alphabet is a WFC and such a condition is true, if the last letter of a stem is identical to that letter. If $\&_1$, $\&_2$,...,$\&_n$ are WFCs, then the following constructions are also WFCs:

(11) (i) $\&_n...\&_2\&_1$
(ii) $<\&_1,\&_2,...,\&_n>$.

(i) is true if $\&_1$ and $\&_2$ and ... and $\&_n$ are true, in that order, under the stipulation that the recognized letters in a stem are consumed (the next condition tests the next letter).

(ii) is true if $\&_1$ or $\&_2$ or ... or $\&_n$ is true. The testing of $\&_i$'s proceeds from left to right and halts if a recognition occurs. Each $\&_i$ starts afresh; only the recognizing $\&_i$ consumes letters.

To enhance compact notation we stipulate that a capital letter can be used as a macro name for a WFC. As an example production, consider the following:

(12) <Ka,y>hde --> POSTULATE(3,'ksi',0,HA)

('K' is an abbreviation for <d, f, g, h, k,...>, i.e. the set of Finnish consonants). This production recognizes, among others, the genetive stems 'kahde' (=of two) or 'yhde' (=of one) and generates basic word form hypotheses 'ka ksi' (=two) and 'y ksi' (=one), respectively.

The SPs were collected into 11 distinct sets of productions for nominals and 6 sets for verbs. On average a set has about 25 rules. These sets were compiled into state transition automata to yield efficient processing.

Confirmation of postulates

In addition to an input word itself and its partial interpretations the lexicon is the only static data structure in MORFIN; all other morphological knowledge is dressed in active rules. For example, the word 'pur| si^{42}' is stored in a single entry as

(13) pur si S NE <..semantic information..>

'takk| i^{4}' is stored in two separate entries as

(14)

takk i S VA <..semantic information..>
tak i S HA <..semantic information..>

where 'S' stands for a noun, 'VA' for strong, 'HA' for weak, and 'NE' for neutral grade.

The confirmation of a basic word form hypothesis corresponds to a match against lexical entries. An entry matches with a hypothesis if the words match and the grades are not of the opposite strength. If the hypothesis is an adjective comparation form, the lexical entry must furthermore be marked as an adjective.

Performance of the model

MORFIN covers the whole Finnish morphology with the provision below. The singular instructive and the old plural genetive IN-cases are exceedingly rare and are left out for the sake of efficiency. They could be easily added. Currently compound nouns are not analyzed into their constituent parts but we are in the process of designing a new version of MORFIN which will analyze also compound nouns.

There is a precompiled version of MORFIN written in standard Pascal. It has been tested both in DEC 20 and VAX 11/780 configurations.

The analysis of a random word in a newspaper text takes between 4 to 60 ms of DEC 2060 CPU time, about 15 ms on average. On average about 4 postulates were generated on the basic word form level.

The basic approach of MORFIN applies well, we believe, also to analysis of derivational word forms. All one has to do is to add proper MPs and (sometimes) SPs (if needed). Thus, for instance, to account 'iloinen' --> 'iloisesti' derivation we have to add only a single MP to recognize 'sti' and can use the already implemented SPs: the stem alternation is similar to the singular elative case: 'iloinen' (=glad) --> 'iloisesta' (=from glad).

One of the main objectives in our design was to store minimal amount of morphological data in a lexicon and dress it maximally in active rules. We feel success in this regard. The only morphological knowledge words in a lexicon carry is the boundary between the root and the stem ending of a word and the grade of the stem. entries. Consequently, MORFIN has a convenient functional feature: words not existing in a lexicon get analyzed as well as those that do exist. A user can add new entries simply by indicating which of the postulated forms are right. It seems that the introduction of new lexical entries is not as straightforward for a casual user in the other systems.

It seems implausible to us that a native speaker of an inflectional language tags morphological data in individual words. If we take a grammatical but meaningless (non-existent) word, say, 'ventukoissa' and test native Finns, they probably all would agree that it represents the plural inessive form of a non-existent word 'ventukka'. Our model covers such phenomena. 'Ventukoissa' is analyzed as well as meaningful words. Only the dictionary look-up process rejects the word as meaningless.

Conclusion

We have designed and implemented a system for the computational morphological analysis of Finnish word forms. Our analysis is based on multi-level heuristic search in which modular active knowledge sources postulate and evaluate partial morphological interpretations. On the first level morphotactic productions postulate and interprete morphotactic segmentations of an input word. The second phase converts the postulated variant stems into their basic forms. The third phase matches the proliferated basic word form hypotheses against a lexicon.

All morphological knowledge other than the root boundary and the grade of a lexeme is dressed in procedural form, which yields efficient analysis. Grammatical but meaningless word forms become analyzed in the model as well as meaningful ones.

References

Brodda, B., Karlsson, F., An experiment with automatic morphological analysis of Finnish, Papers from the Institute of Linguistics, University of Stockholm, Stockholm 1980.

Erman, L.D. et al., The Hearsay-II speech-understanding system: integrating knowledge to resolve uncertainty. Computing Surveys (June, 1980), 213-253.

Jäppinen, H., Lehtola, A., Nelimarkka, E. and Ylilammi, M., Morphological Analysis of Finnish - A heuristic approach. Helsingi University of Technology, Digital Systems Laboratory, Report B 26, 1983.

Karlsson, F., Finsk Grammatik. Suomalaisen Kirjallisuuden Seura, Helsinki, 1981.

Karttunen, L., Root, R., and Uszkoreit, Hl, TEXFIN: Morphological analysis of Finnish by computer. Proc. of the 71st Ann. Meeting of the SASS. Albuquerque, 1981.

Koskenniemi, K., Two-level model for morphological analysis. IJCAI-83, 1983, 683-685.

Sågvall-Hein, A-L., Finnish morphological analysis in the reversible grammar system, COLING 78, 1978.

Fred Karlsson
Department of General Linguistics
University of Helsinki
Hallituskatu 11-13
SF-00100 HELSINKI 10
FINLAND

TAGGING AND PARSING FINNISH

1. Introductory remarks

Current parsing theory has an obvious English bias. Many of the problems discussed originate in typical features of English such as the scarcity of surface morphosyntactic markers, the scarcity of word forms, and the occurrence of certain types of ambiguous syntactic structures. E.g. Winograd (1983, 544-5) argues that morphological phenomena do not lend themselves well to the methodology of generative grammar because of their high degree of irregularity and idiosyncracy. Furthermore, he opines that any analysis seeking to find morphological regularities must examine words in terms of their history. He even goes so far as to argue that, in contrast to what holds for syntax, native speakers do not utilize grammatical knowledge in the production and understanding of morphological structures. He concedes, though, that there are a few highly productive morphological phenomena that cannot be handled by lexical look-up. But the bulk of morphology is anyway to be deposited in the lexicon just by listing individual forms.

This view does, perhaps, descriptive justice to large portions of English morphology even though the demarcation line between syntax and morphology seems unnecessarily strict even here. But a **general model** of morphological competence and processing must surely provide stronger means for dealing with e.g. the plethora of word forms found in more synthetic languages.

This paper sketches some basic problems to be dealt with in tagging and parsing synthetic, morphologically rich languages such as Finnish. We see tagging and parsing as closely interdependent (cf. Brodda 1982). A good tagging program is equivalent to a parser in important respects. It is also a methodological prerequisite to doing large-scale parsing, which cannot succeed without the possibility of checking hypotheses on easily accessible, large, grammatically coded corpora.

It is self-evident that the wealth of overt ending morphs is beneficial e.g. in determining constituent structure, dependencies, and syntactic functions. A sufficiently general morphological analyzer conclusively solves most local syntactic problems and provides valuable clues to long-distance dependencies as well. Syntactic parsing strategies of the types discussed e.g. by Bever (1970) and Kimball (1973) certainly have to be widely employed in parsing synthetic languages. But from a general point of view, we see as particularly important the make-up of the **lexicon** that emerges when one tries to model intricate systems of inflexion and derivation.

Specifically, this paper deals with derivational morphology and morphological tagging as initial phases of a project with the aim of constructing a morphosyntactic parser for unrestricted Finnish text. The final system should i.a. be able to cope with the **whole vocabulary** including exceptions, loanwords, neologisms, potential productive derivatives, etc. We see it as important not to compromise in this respect since initial restrictions to micro-worlds or vocabularies consisting of only a few hundred words tend to severely misrepresent the problems encountered when the whole lexicon is faced. The system should optimally be **bidirectional**, i.e. able both to analyze and to synthesize (produce) word forms and sentences. It should also, as far as possible, have an interpretation as a model of the "real" morphosyntactic processes. No semantics has yet been included. We take one minimal requirement of an adequate semantics to be the ability of the system to treat all morphosyntactic forms.

2. A model of derivational morphology

Koskenniemi (1983; also cf. this volume) has designed a general model for word-form recognition and production. The model has been applied to Finnish and yielded a full description of inflexional morphology satisfying the requirements outlined above. The current running lexicon contains the top 3,000 entries of the Finnish frequency dictionary. The model analyzes and produces all the (inflexional and cliticized) 2,000 forms of Finnish nominal paradigms and the 18,000 forms of verb paradigms. The ultimate lexicon will contain some 10,000 entries which suffices for analyzing ordinary running text (excluding, of course, specialized vocabulary).

Koskenniemi's model also analyzes **compounds** provided the (base forms of their) constituent parts are in the lexicon. Compounding is such a central morphological means in synthetic languages that it must be easily tractable also in computational models aspiring general applicability.

The third morphological domain to be covered is **derivation.** The total number of derivational morphemes in Finnish is 150-200 depending upon how the most opaque and unfrequent ones are interpreted. Some 50-60 are highly productive and these are to be discussed here. The maximal productive Finnish derivational system comprises nine morphotactic positions with the following contents (Karlsson 1983).

1	2	3	4	5	6	7	8	9		
(root)	V_1	V_2	V_3	PASS	N_1	N_2	A_1	A_2	N_3	(infl.)	(clit.)

Positions 1,2,3 contain endings deriving verbs from either nouns or verbs. These endings are mainly causative, reflexive, frequentative, or momentaneous. Combinations of up to three of these occur (e.g. lue/t/utt/ele 'read/caus/caus/freq', i.e. 'habitually make somebody to read'). Position 4 has only one member, the so-called passive (better: indefinite). Position 5 contains the majority of (denominal or deverbal) noun endings, position 6 a few noun endings appending to those in position 5 (e.g. asu/nto/la 'hostel'). Position 7 contains the majority of

adjectival endings, position 8 a few adjectival endings appending to position 7 (e.g. kiihty/vä/mpi 'more accelerating'). Position 9, finally, contains only one ending deriving nouns from derived adjectives (kiihty/vä/mm/yys 'the property of being more accelerating').

The derivational system is further complicated by the possibility of repeated recursion from positions 7 and 9, i.e. some derived adjectives and nouns permit further derivation of verbs etc during a second and even third pass through the morphotactic flow chart. This provides for derivatives such as oike/ude/llis/ta/minen (approximately:) 'causing to be legal', where ude is N_3 (first pass), llis is A_1 (second pass), ta is V_1 (third pass) and minen is N_1 (third pass).

As a first step, the derivational system has been modelled as a BETA rule-system (cf. Brodda (forthcoming), Brodda & Karlsson 1981) that generates all productive derivatives for any given input root. The morphophonological dependencies between roots and endings, and between endings in sequences, have been precisely described and implemented as BETA-rules in order to prevent at least morphological overgeneralizations. This descriptive task alone is a considerable venture. Semantic restrictions, however, have not yet been considered.

We pick an example. Appendix 1 contains the near-maximal set of 161 derivatives, including some awkward ones, that the BETA rule system generates for the verb hakkaa 'hew'. The first six forms (group a) are derived verbs with flow chart positions 1,2,3 and some combinations of these filled. Then follow (group b) 57 nominal derivatives based on the underived root, including combinations of up to three nominal endings. Then follow minimal sets (c-h, 16-17 members in each) of nominal derivatives of the derived verbs (a). The sets (c-h) are conservatively composed. If made maximal, they would contain more than 50 members each.

Now recall that each of the six derived verbs has some 18,000 inflexional and cliticized forms, and all the 155 derived nominals some 2,000 such forms. This would give an astounding sum total of more than 400,000 potential productive derived, inflected, and cliticized forms of the single verb

hakkaa.

Such figures clearly show the untenability of any attempt to treat the bulk of synthetic morphological systems, be they agglutinative as Turkish or semi-agglutinative as Finnish, by way of mere lexical listing. The vast majority of the 400,000 forms under scrutiny certainly are both morphologically and semantically regular in the strongest sense of the word, i.e. their morphophonological behaviour and compositional meaning is predictable. These forms must be generated by rules in autonomous grammars as well as in models of morphological performance. Of course, this is not to draw the demarcation line between rules and lexicon where it was drawn by SPE-type phonology. We just stress that the huge majority of forms in synthetic languages must be described as products of rules. Several recent anglocentric theories of the lexicon have gone too far in the opposite direction by including all or most forms in the lexicon (cf. Lieber 1981 for references). Also note that it would make no psycholinguistic sense to claim that the Finnish lexicon is tens, or even hundreds, of thousands of times larger than the English one.

The BETA rule system provides an initial formalization in the process mode of Finnish derivational morphology. The following step will be to incorporate these productive derivational mechanisms in the Finnish implementation of Koskenniemi's two-level model. This makes it possible to reduce the size of the lexicon by eliminating all morphologically and semantically predictable derivatives. It also vastly improves the possibilities of the system to deal with running text. Note e.g. that the system then is able to analyze also occasional compounds with (any number of) derived constituents. It will, as a case in point, have the full power of coping with all the 400,000 forms of the verb hakkaa, both as independent words and as constituents in compounds.

This is the kind of full coverage needed in any reliable practical application such as information retrieval or spelling correction.

3. Morphological tagging: FINTAG

Readily available facilities for testing hypotheses and alternatives on large natural corpora are mandatory when one tries to design a parser for running text. For this purpose we have devised a morphological tagging program, FINTAG, that consists of thirteen BETA rule modules geared to apply in sequence to any input text. The output is the original text so analyzed that (a) all word forms have been tagged with a part of speech label, and (b) all inflexional endings and clitics have been segmented. The tagging format is intagging in the sense of Brodda (1982), i.e. the part of speech labels are prefixed to the word forms (separated by a colon). The output format is thus e.g. PR:TÄMÄ=N N:VUODE=N N:ALKU VF:ON A:KYLMÄ=Ä N:AIKA=A 'the beginning of this year is a cold time', where the tags have standard meanings (VF = finite verb), =N is the genitive sg. ending, and =Ä, =A are partitive sg. endings. The equation mark (=) serves as ending juncture except in illatives (%) and possessives (").

The thirteen BETA rule modules (containing a total of some 7000 lines of substitution rules) have been mutually sequenced both on linguistic and strategic grounds. Some of the rule modules check for endings, some are lexicons looking for specific stems. The part of speech tags are assigned according to the following **tagging strategies** (the invocation order is mostly the one listed):

- whenever a word form is conclusively tagged and segmented, prefix a plus-sign to it marking that the remaining rule modules will not analyse this form; the final module removes all plus-signs;
- initially mark all potentially monosyllabic first syllables with a temporary diacritic (for facilitating ending identification; the final module removes all remaining diacritics);
- the top 200 word forms of the Finnish frequency dictionary are tagged and segmented as wholes;
- those 600 most common adverbs are segmented as wholes

that would otherwise be oversegmented by the ending procedures (note: lots of contemporary adverbs are petrified, etymologically discernible nominals);
- (the stems of) closed lexical classes are identified by lexical lists;
- segment all endings in a specified order dispersed over most of the 13 modules (this is the bulk of the morphology and largely equivalent to the work reported in Brodda & Karlsson 1981); those comparatively few unitary words that are potentially homonymous to endings are mostly listed and thereby exempted from segmentation;
- whenever possible, predict the part of speech labels on the basis of segmented inflexional or derivational endings;
- use (inconclusive) frequency-based stem-lexicons for adjectives and verbs to assign part of speech labels;
- by default, predict that the remaining untagged word forms are nouns.

Appendix 2 shows the intermediate phases in the process of tagging two sentences. Only six of the thirteen rule modules have been active. It is clearly seen how most of the "tagging load" resides with module 4 (the frequent word forms), module 6 (most verb endings; note: VI3 = 3rd infinitive, VPA2 = 2nd active participle, VPP2 = 2nd passive participle), module 9 (most nominal and some verb endings), module 12 (certain pronominal, numeral, and adjectival stems), and module 13 (by default, nouns).

The output contains two errors. PARI=A is, here, a numeral and not a noun. ESITELMINÄ is undersegmented, the correct analysis would be N:ESITELM=I=NÄ, an essive pl. left unsegmented on purpose due to surface homonymy with several frequent nouns with a base form ending in -ina, -inä. Such errors are, of course, due to the necessary but fallible heuristics inherent in any model not utilizing a total lexicon. Where conclusive decisions cannot be made, the most likely route is picked. In such istances, the heuristic choices have been made on the basis of large corpus studies.

So far, FINTAG has been applied to a text consisting of 66,000 word forms. It has an average success rate of some **85 %** (in regard to word form tokens). A tagged form is taken to be correct only if no changes have to be made, i.e. the part of speech label is proper and disambiguated in context, all endings (if any) are properly segmented, and base forms are left unsegmented. Over- and undersegmentations are counted as errors, as are unresolved homonymies (e.g. N/A:SUOMALAINEN) and, of course, (eventual other) improperly placed boundaries and improper part of speech labels.

It deserves to be stressed that the success rate 85 % has been achieved on the **level of word forms** alone, without (a) active syntactic checking of the properties of neighbouring words or the whole clause, and (b) "memory" of previous decisions. This is a further proof of the **high information load of overt morphological markers** (also cf. Brodda & Karlsson 1981). Tentative experiments indicate that the success rate of FINTAG may fairly easily be raised to 93-95 % by invoking simple syntactic environment tests. In particular, this would provide effective means for disambiguating part of speech homonymies such as N/A:SUOMALAINEN, which, in the current version, constitute a large share of the "errors". Here we are approaching procedures akin to genuine parsing.

Note, for the sake of comparison, that Leech, Garside & Atwell (1983, 13) report a success rate of 96.7 % in tagging the LOB corpus. This figure is only minimally ahead of the morphologically dominated (revised) FINTAG.

The final version of FINTAG will be expanded to cover grammatical functions (subject, object, etc) and head-modifier relations, cf. the work done by Svartvik, Eeg-Olofsson, Forsheden, Oreström & Thavenius (1982) in syntactically tagging the Survey of Spoken English corpus. Even such information is, in Finnish, to a very large extent (over 90 %) inferrable from surface morph configurations. This is work in progress, based on LISP. Here, the aims and problems of tagging and parsing converge. Sophisticated "tagging" is nothing but applying theoretical "parsing" models.

REFERENCES

Bever, T.G. 1970. The cognitive basis for linguistic structures. In J.R. Hayes (ed.), **Cognition and the development of language**, John Wiley & Sons, N.Y., 279-352.

Brodda, B. 1982. Problems with tagging - and a solution. **Nordic Journal of Linguistics** 5:2, 93-116.

--- (forthcoming) The BETA system. Data Linguistica, Gothenburg.

Brodda, B. & Karlsson, F. 1981. An experiment with automatic morphological analysis of Finnish. Department of General Linguistics, University of Helsinki, Publications No. 7.

Karlsson, F. 1983. **Suomen yleiskielen äänne- ja muotorakenne.** WSOY, Porvoo.

Kimball, J. 1973. Seven principles of surface structure parsing in natural language. **Cognition** 2, 15-47.

Koskenniemi, K. 1983. Two-level morphology. A general computational model for word-form recognition and production. Department of General Linguistics, University of Helsinki, Publications No. 11.

Leech, G., Garside, R. & Atwell, E. 1983. The automatic grammatical tagging of the LOB corpus. ICAME News No. 7, 13-33.

Lieber, R. 1981. On the organization of the lexicon. IULC.

Svartvik, J. & Eeg-Olofsson, M. & Forsheden, O. & Oreström, B. & Thavenius, C. 1982. **Survey of Spoken English.** Report on Research 1975-1981. CWK Gleerup, Lund.

Winograd, T. 1983. **Language as a cognitive process.** Vol.1: Syntax. Addison Wesley, N.Y.

APPENDIX 1. Maximal set of derivatives for the verb hakkaa 'hew'

hakkaile (a)
hakkailutta
hakkautta
hakkaise
hakkautu
hakkaantu

hakkaaminen
hakkaaja (b)
hakkaajuus
hakkaajatar
hakkaajattaruus
hakkaajamainen
hakkaajamaisuus
hakkaajatarmainen
hakkaajatarmaisuus
hakkaajamaisempi
hakkaajamaisemmuus
hakkaajamaisin
hakkaajamaisimmuus
hakkaava
hakkaavuus
hakkaavampi
hakkaavammuus
hakkaavin
hakkaavimmuus
hakannut
hakanneisuus
hakanneempi
hakanneemmuus
hakannein
hakanneimmuus
hakattava
hakattavuus
hakattavampi
hakattavammuus
hakattavin
hakattavimmuus
hakattu
hakattuus
hakatumpi
hakatummuus
hakatuin
hakatuimmuus
hakkaamaton
hakkaamattomuus
hakkaamattomampi
hakkaamattomammuus
hakkaamattomin
hakkaamattomimmuus
hakkaavainen
hakkaavaisuus
hakkaavaisempi
hakkaavaisemmuus
hakkaavaisin
hakkaavaisimmuus
hakkaus
hakkauksellinen
hakkauksellisuus
hakkauksellisempi
hakkauksellisemmuus
hakkauksellisin
hakkauksellisimmuus

hakkaileminen
hakkailija (c)
hakkailijuus
hakkailijamainen
hakkailijamaisuus
hakkaileva
hakkailevuus
hakkaillut
hakkailleisuus
hakkailtava
hakkailtavuus
hakkailtu
hakkailematon
hakkailemattomuus
hakkailevainen
hakkailevaisuus
hakkailu

hakkailuttaminen
hakkailuttaja (d)
hakkailuttajuus
hakkailuttajamainen
hakkailuttajamaisuus
hakkailuttava
hakkailuttavuus
hakkailuttanut
hakkailuttaneisuus
hakkailutettava
hakkailutettavuus
hakkailutettu
hakkailuttamaton
hakkailuttamattomuus
hakkailuttavainen
hakkailuttavaisuus

hakkauttaminen
hakkauttaja (e)
hakkauttajuus
hakkauttajamainen
hakkauttajamaisuus
hakkauttava
hakkauttavuus
hakkauttanut
hakkauttaneisuus
hakkautettava
hakkautettavuus
hakkautettu
hakkauttamaton
hakkauttamattomuus
hakkauttavainen
hakkauttavaisuus

hakkaiseminen
hakkaisija (f)
hakkaisijuus
hakkaisijamainen
hakkaisijamaisuus
hakkaiseva
hakkaisevuus
hakkaissut
hakkaisseisuus
hakkaistava
hakkaistavuus
hakkaistu
hakkaisematon
hakkaisemattomuus
hakkaisevainen
hakkaisevaisuus
hakkaisu

hakkautuminen
hakkautuja (g)
hakkautujuus
hakkautujamainen
hakkautujamaisuus
hakkautuva
hakkautuvuus
hakkautunut
hakkautuneisuus
hakkauduttava
hakkauduttavuus
hakkauduttu
hakkautumaton
hakkautumattomuus
hakkautuvainen
hakkautuvaisuus

hakkaantuminen
hakkaantuja (h)
hakkaantujuus
hakkaantujamainen
hakkaantujamaisuus
hakkaantuva
hakkaantuvuus
hakkaantunut
hakkaantuneisuus
hakkaannuttava
hakkaannuttavuus
hakkaannuttu
hakkaantumaton
hakkaantumattomuus
hakkaantuvainen
hakkaantuvaisuus

(a) = root → der. V
(b) = root → der. N, A
(c) = hakka/ile → der. N, A
(d) = hakka/il/utta → der. N, A
(e) = hakka/utta → der. N, A
(f) = hakka/ise → der. N, A
(g) = hakka/utu → der. N, A
(h) = hakka/antu → der. N, A

APPENDIX 2. Intermediate phases of the tagging process. Numbers refer to the output of the respective active module. The effects of each module are underlined. Two incorrect outputs are starred. Cf. the text.

1

TÄ2MÄN
KO'KOELMAN
KI'RJOITUKSET
O'VAT
PA'RIA
KO'LMEA
LU'KUUN
O'TTAMATTA
SY'NTYNEET
VII'DEN
VII'ME
VUOlDEN
AI'KANA.
E'RÄÄT
NII2STÄ
+VF:ON
JU'LKISTETTU
LE'HDISTÖSSÄ,
E'RÄÄT
RA'DIOSSA,
E'RÄÄT
E'SITELMINÄ.

4

+PR:TÄMÄ=N
KO'KOELMAN
KI'RJOITUKSET
+VF:O=VAT
PA'RIA
KO'LMEA
LU'KUUN
O'TTAMATTA
SY'NTYNEET
VII'DEN
+A:VIIME
+N:VUODE=N
+N:AIKA=NA.
E'RÄÄT
+PR:NI=I=STÄ
+VF:ON
JU'LKISTETTU
LE'HDISTÖSSÄ,
E'RÄÄT
RA'DIOSSA,
E'RÄÄT
E'SITELMINÄ.

6

+PR:TÄMÄ=N
KO'KOELMAN
KI'RJOITUKSET
+VF:O=VAT
PA'RIA
KO'LMEA
LU'KUUN
+VI3:OTTA=MA=TTA
+VPA2:SYNTY=NEE=T
VII'DEN
+A:VIIME
+N:VUODE=N
+N:AIKA=NA.
E'RÄÄT
+PR:NI=I=STÄ
+VF:ON
+VPP2:JULKISTE=TTU
LE'HDISTÖSSÄ,
E'RÄÄT
RA'DIOSSA,
E'RÄÄT
E'SITELMINÄ.

9

+PR:TÄMÄ=N
KOKOELMA=N
KIRJOITUKSE=T
+VF:O=VAT
PARI=A
KOLMEA
LUKU%UN
+VI3:OTTA=MA=TTA
+VPA2:SYNTY=NEE=T
VIIDE=N
+A:VIIME
+N:VUODE=N
+N:AIKA=NA.
ERÄÄ=T
+PR:NI=I=STÄ
+VF:ON
+VPP2:JULKISTE=TTU
LEHDISTÖ=SSÄ,
ERÄÄ=T
RADIO=SSA,
ERÄÄ=T
ESITELMINÄ.

12

+PR:TÄMÄ=N
KOKOELMA=N
KIRJOITUKSE=T
+VF:O=VAT
PARI=A
+NUM:KOLME=A
LUKU%UN
+VI3:OTTA=MA=TTA
+VPA2:SYNTY=NEE=T
+NUM:VIIDE=N
+A:VIIME
+N:VUODE=N
+N:AIKA=NA.
+PR:ERÄÄ=T
+PR:NI=I=STÄ
+VF:ON
+VPP2:JULKISTE=TTU
LEHDISTÖ=SSÄ,
+PR:ERÄÄ=T
RADIO=SSA,
+PR:ERÄÄ=T
ESITELMINÄ.

Final output

13

PR:TÄMÄ=N
N:KOKOELMA=N
N:KIRJOITUKSE=T
VF:O=VAT
★N:PARI=A
NUM:KOLME=A
N:LUKU%UN
VI3:OTTA=MA=TTA
VPA2:SYNTY=NEE=T
NUM:VIIDE=N
A:VIIME
N:VUODE=N
N:AIKA=NA.
PR:ERÄÄ=T
PR:NI=I=STÄ
VF:ON
VPP2:JULKISTE=TTU
N:LEHDISTÖ=SSÄ,
PR:ERÄÄ=T
N:RADIO=SSA,
PR:ERÄÄ=T
★N:ESITELMINÄ.

Gregers Koch
Datalogisk Institut Københavns Universitet
Sigurdsgade 41
DK-2000 København N

NATURAL LANGUAGE PROGRAMMING

1. Aims

This is a study of automated programming. We are aiming at the stepwise development of a programming environmemt consisting of an automatic translation system to translate texts in natural language (e.g. software requirement specifications) into certain logical formulae according to some semantic theory, as well as into executable programs. The only semantic theories considered here are logical theories, whether they be related to the lambda calculus or to the predicate calculus, and we shall henceforth talk about the level of logical representation (rather than semantic representation).

The paradigm of the method may be guessed from the figure 1. The horisontal axis displays the gap between the human user and the computer (that is the so-called man-machine communication problem). The vertical axis indicates the level of abstraction, from low levels of abstraction up to higher ones. The areas of natural language, programming language, and the predicate calculus are indicated with an overlap between the latter two, as some predicate calculus expressions are executable and other are not.

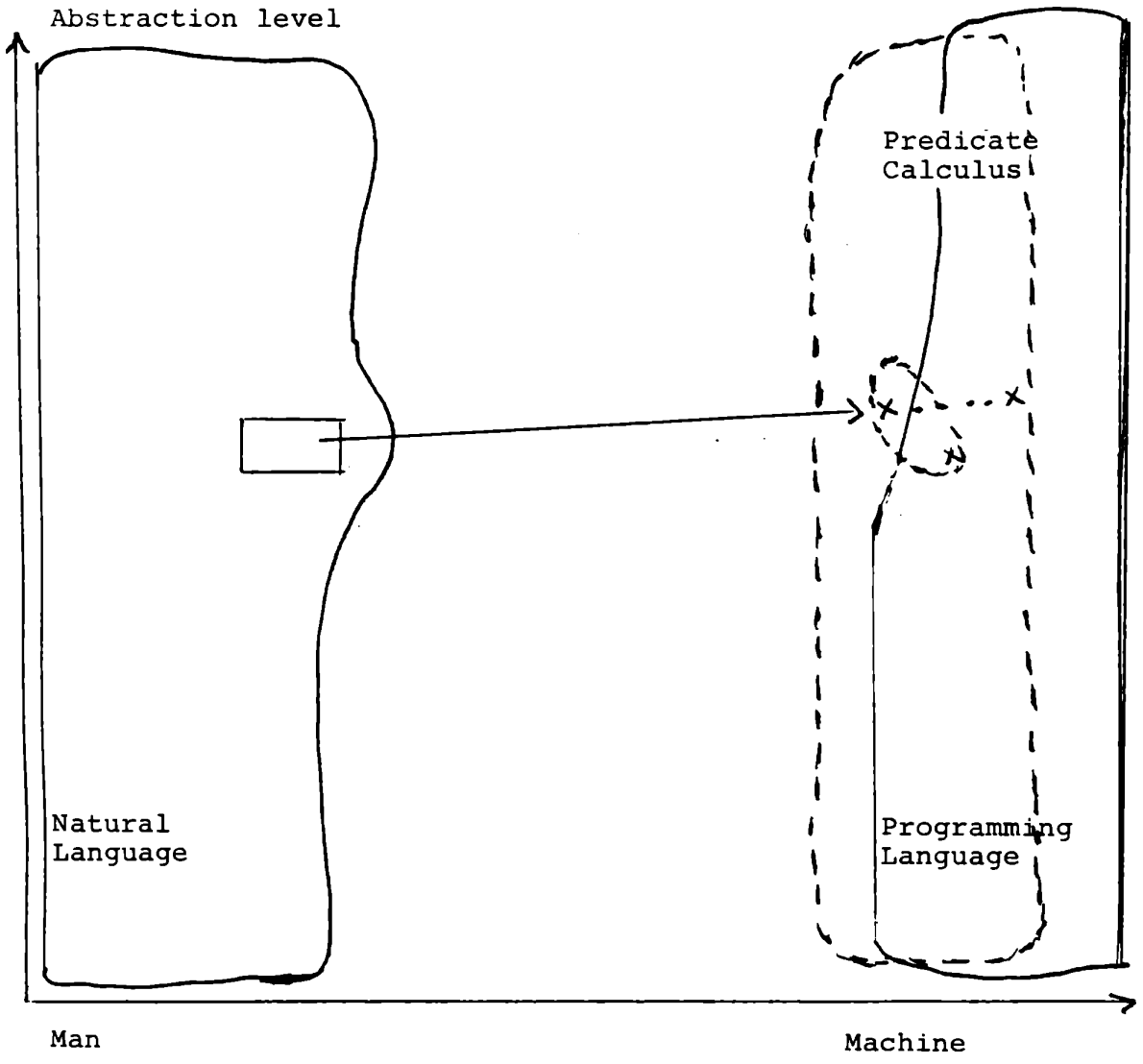


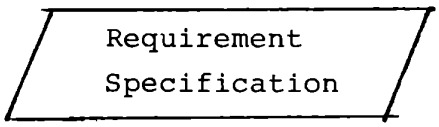
Figure 1.

The aim of this project is to develop a rigorously defined "square" sublanguage of natural language with a corresponding system performing automated translation into the predicate calculus.

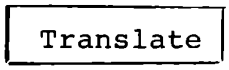
2. Developing natural sublanguages

Starting with whatever semantic theories are available in the literature the aim here is to develop step by step a computational sublanguage of natural language. This development is performed on the basis of carefully selected examples (and it will be illustrated by examples). The textual formulation of the requirement specification will thus be translated automatically into a logical representation.

The figure 2 shows the same thing in a diagrammatic form. Here are indicated the documents occuring during the process, for instance



and a few processes to be discussed, for instance



Several aspects of the figure 2 will be commented upon in the rest of the paper.

3. Reverse translation

Generating i.e. reverse translation back into natural language, makes it possible to check the quality of the textual translation process.
It is a characteristic property of logic programming languages that (at least in principle) the user may apply the same program for translation and for generation.

4. Logical alternatives

The particular choice of logical representation is essentially arbitrary, but the scientific literature heavily

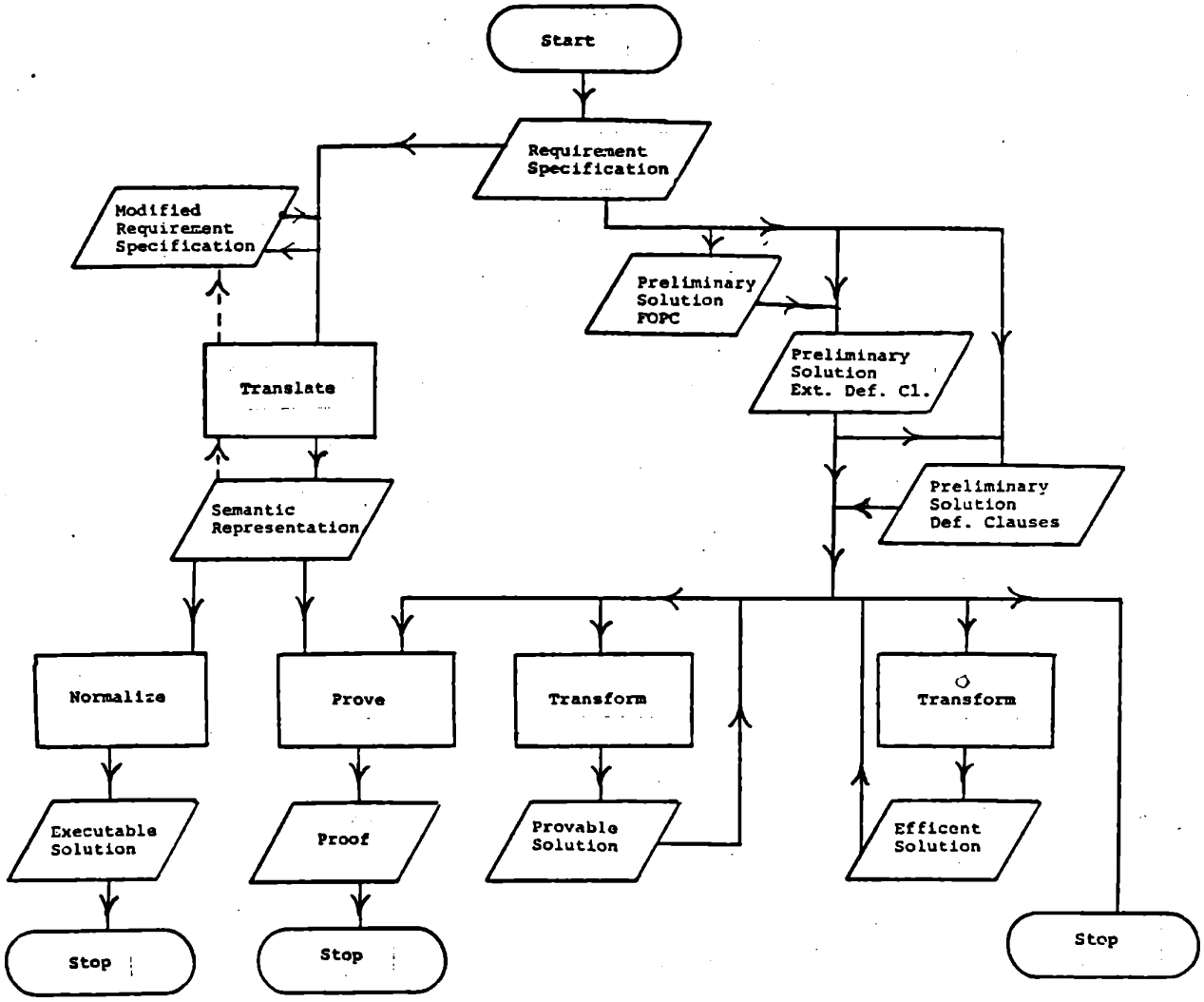


Figure 2

supports two kinds of representations. Here we chose a logic grammar (like [4]). An interesting alternative is a intensional grammar á la PHLIQA1 [3,7,14,16] . Conceivably the latter alternative may be characterized as a prototypical lambda calculus method and the former alternative as a prototypical predicate calculus method.

Also in a more narrow context of parsing (instead of full translation) the following method is probably preferable to that of the Uppsala Chart Parser [17] as far as modifiability, extensibility, portability, and experimentation with regard to grammatical descriptions are concerned.

The stepwise development of the computational sublanguage of ordinary English used a number of carefully selected benchmark problems.

5. An example

The first benchmark problem is simple enough to allow a fairly thorough presentation.
By the example of the fallible Greek we also demonstrate the idea of expressing a grammar as a logic program.

The problem consists in having the computer respond in a sensible way to the following ordinary English text (or natural language requirement specification):

Turing is human.
Socrates is human.
Socrates is Greek.
every human is fallible.
which human is Greek and is fallible ?

i.e. the program should answer the query in the last sentence.
We select the following simple context-free grammar

```
<Sentence>        ::=<Term><Verbphrase>
<Term>            ::=<Propername>|<Determiner><Nounphrase>
<Nounphrase>      ::=<Noun>|<Noun><Relativeclause>
<Relativeclause>::= that <Verbphrase>
<Verbphrase>      ::=<Transitiveverb><Term>|<Verbphrase>and <Verbphrase>
<Determiner>      ::= every | which
<Noun>            ::= human
<Transitiveverb>::= is | isn't
<Propername>      ::= Turing | Socrates | Greek | fallible |  human
```

If we just want a parser (to accept or reject the input sentence) we may simply change the grammar into the logic program of figure 3, where the variables function as pointers to the input string.
To solve our problem of the fallible Greek we need a somewhat more sophisticated translation of the accepted input sentence, as shown in figure 4. For the sake of clarity we have here omitted the variables functioning as pointers (as in figure 3). We have only given the variables designating the focus and result(s).

Translate:

```
Sentence(x,z) if Term(x,y) & Verbphrase(y,z).
Term(x,y) if Propername(x,y).
Term(x,z) if Determiner(x,y) & Nounphrase(y,z).
Nounphrase(x,y) if Noun(x,y).
Nounphrase(x,z) if Noun(x,y)&Relativeclause(y,z).
Relativeclause(x,z)if Check(that,x,y)&Verbphrase(y,z)
Verbphrase(x,z)if Transitiveverb(x,y)&Term(y,z).
Verbphrase(x,w) if Verbphrase(x,y)&Check(and,y,z)
   & Verbphrase(z,w).
Determiner(x,y)if Check(every,x,y).
Determiner(x,y)if Check(which,x,y).
Noun(x,y)if Check(human,x,y).
Transitiveverb(x,y)if Check(x,y)
Transitiveverb(x,y)if Check(isn't,x,y).
Propername(x,y)if Check(Turing,x,y).
Propername(x,y)if Check(Socrates,x,y).
Propername(x,y)if Check(Greek,x,y).
Propername(x,y)if Check(fallible,x,y).
Propername(x,y)if Check(human),x,y).
```

Figure 3.

Translate:

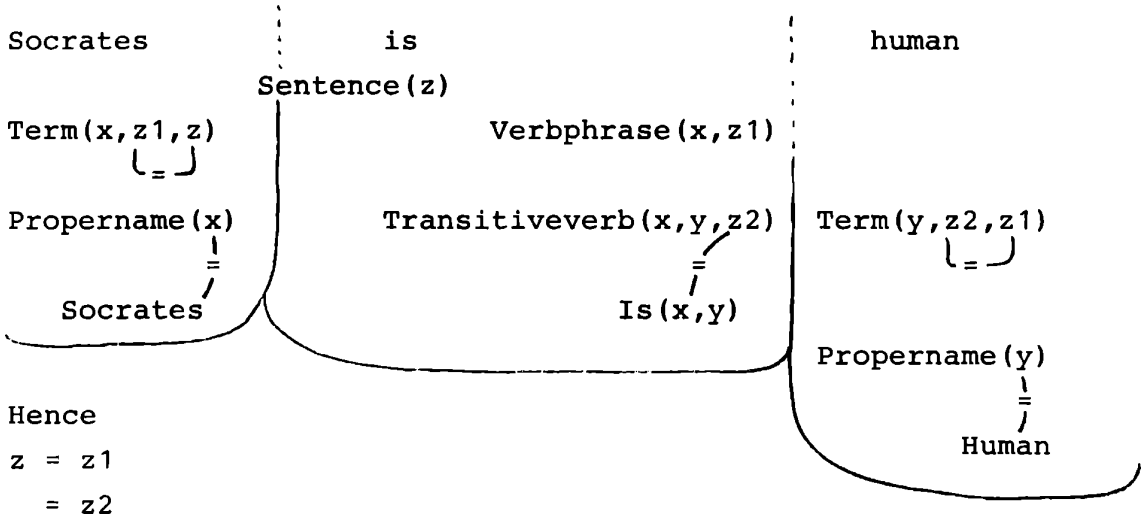
```
Sentence(z)if Term(x,z1,z)&Verbphrase(x,z1).
Term(x,z,z)if Propername(x).
Term(x,z1,z)if Determiner(x,z2,z1,z)&Nounphrase(x,z2).
Nounphrase(x,z)if Noun(x,z).
Nounphrase(x,z1&z2)if Noun(x,z1)&Relativeclause(x,z2)
Relativeclause(x,z)if Check(that)&Verbphrase(x,z).
Verbphrase(x,z)if Transitiveverb(x,y,z1)&Term(y,z1,z)
Verbphrase(x,z1&z2)if Verbphrase(x,z1)&Check(and)
   & Verbphrase(x,z2).
Determiner(x,z1,z2,∀x[z1⇒z2])if Check(every).
Determiner(x,z1,z2,Which(x,z1&z2))if Check(which).
Noun(x,Is(x,Human)) if Check(human).
Transitiveverb(x,y,Is(x,y)) if Check(is).
Transitiveverb(x,y,¬Is(x,y))if Check (isn't).
Propername(Turing)if Check(Turing).
Propername(Socrates)(if Check(Socrates).
Propername(Greek) if Check(Greek).
Propername(Fallible)if Check(fallible).
Propername(Human)if Check(human).
```

Figure 4.

The computational processing of the second and the second last sentences are displayed in the figures 5 and 6, respectively.

In conclusion the output from the translation program Translate is shown in figure 7.

Example

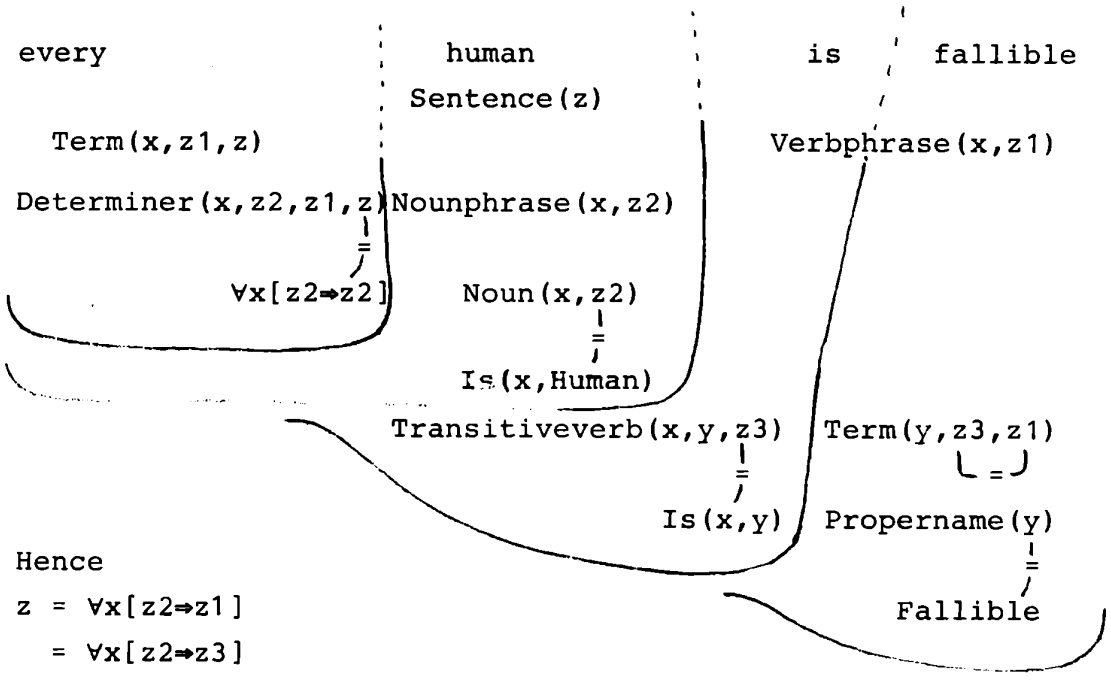


Hence

z = z1
  = z2
  = Is(x,y)
  = Is(Socrates, Human)

Figure 5

Example



Hence

z = ∀x[z2⇒z1]
  = ∀x[z2⇒z3]
  = ∀x[z2⇒Is(x,y)]
  = ∀x[Is(x,Human)⇒Is(x,Fallible)]

Figure 6

Translation output:

```
Is (Turing, Human).
Is (Socrates, Human).
Is (Socrates, Greek).
∀x[Is(x,Human)⇒Is(x,Fallible)].

Which(x,Is(x,Greek)&Is(x,Fallible)&Is(x,Human)).
```

Figure 7.

6. Normalization

The further transformation of formulae in the logical representation depends heavily upon the particular choice of logic programming language. In case of a dialect of Prolog the transformation should constitute a normalization into conjunctive normal form (or clausal form). Here the question is raised whether or not the sublanguage actually will generate logical formulae in a clausel form that are definite (Horn clauses). (An interesting problem would be to characterize the sublanguages satisfying this requirement).

In the first benchmark problem we get the normalized formulae of figure 8.

Clausal form (conjunctive normal form):

```
Is(Turing,Human).
Is(Socrates,Human).
Is(Socrates,Greek).
Is(x,Fallible)if Is(x,Human).
Print(x) if Is(x,Greek)&Is(x,Fallible)&Is(x,Human).
```

Figure 8.

7. Verifying handwritten programs

The logical representation may be used in the context of verifying handwritten logic programs, as indicated in figure 2. As an example of verification we have the second benchmark problem which is a variant of the Alpine Club problem from the artificial intelligence literature [15].

8. Natural language programming

The logical representation allows direct execution on the computer, which means that we are really developing an automated programming system based on natural language. It seems likely that this level of logical representation should be

considered as yet another level (a fifth) in the context of the four levels dealt with in the EUROTRA project [13]. Actually it is an obvious possibility to extract this kind of information from the third, so-called logico-semantic level and build the recommended logical representations from that information. Unfortunately, I tend to be very pessimistic as to whether this task will actually be realised by the EUROTRA participants.

It is relatively easy to supplement this automated programming system with specific rules concerning the problem domain to make it a knowledge-based system, whether the rules are formulated in a notation akin to the chosen logical representation, or in the form of additional texts in natural language.

So this system may constitute the kernel of a knowledge based automated programming system, where frame information specific to the universe of discourse are to be added. This kind of programming may very well be termed "programming in natural language" or "natural language programming"(hence the title of this paper).

The third benchmark problem is a prototypical database query language problem [19]. The fourth benchmark problem is a small computer aided design problem in architectural design. These benchmark problems fall into the class of natural language programming. Further details may be found in the report [12].

9. Status remarks

The system here was written for English to develop an English computational sublanguage. An obvious alternative could be to develop a similar system in some (or every) Scandinavian language (and it might include the surface structures of the logic programming language). A related experiment using the same method in the automated translation from Japanese will be reported on (in [2]).

When developing the appropriate natural sublanguage certain difficulties showed up in connection with pronouns. They may be examplified by the sentence:

A man takes an apple and he eats it.

The difficulties concerned the scope rules of the quantification and they could certainly be overcome by extending the logical connectives into two-dimensional operators in a systematic manner [12].

As far as the plural of nouns and quantification are concerned, they were needed in the fourth benchmark problem. There seems to be essentially six different ways to extend the computational sublanguage with quantification, as illustrated in figure 9.

There is a need for gaining experience with the use of systems like this one. Virtually nothing is available in the scientific literature.

Such a system should not be considered completely trivial to use, although its potentiality for popularization should be recognised (Virtually everybody who is not an analphabet might learn to use it).

Decisions:

1: Should the translation be one-pass or two-pass (many-pass) ?
2: Should the result be expressed in higher order functions (or cardinality) ?
3. Should there be two truth-values or three (many) ?

1: Passes	2: Higher-order	3: Values	
1	No	2	(here)
1	No	3	-
1	Yes	2	(Intensional grammars and Lexical-Functional Grammars)
1	Yes	3	
2	Yes	2	(here)
2	Yes	3	(Logic grammars).

Figure 9.

10. References

A few related contributions from my institute are included in the list though not referred to in the paper.

[1] F. Als et al: Compiling in Prolog (in Danish), DIKU report 83/16, Institute of Datalogy. Copenhagen University, 1983.

[2] A. Bernth: Logics applied to the translation of Japanese (in Danish), these proceedings.

[3] W.Bronnenberg et al.: The question answering system PHLIQA1, in L. Bolc (ed.) Natural communication with computers Vol. 2, 1980.

[4] A. Colmerauer: An interesting subset of natural language, in Clark and Tärnlund (eds.) Logic programming, 1982.

[5] N.D.Jones and A.Mycroft: Stepwise development of denotational semantics for Prolog, DIKU report 83/1, Institute of Datalogy, Copenhagen University, 1983.

[6] P.H. Jørgensen and G. Koch: Two new methods of natural language database queries (in Danish), Proc. NordDATA Conf., Copenhagen 1981, 2, 227-232.

[7] G. Koch: Experimental formalization of Danish . Institute of Datalogy, Copenhagen University, 1979. DIKU report 79/19 (in Danish).

[8] G. Koch: A Prolog way of representing natural language fragments, DIKU report 80/16, Institute of Datalogy, Copenhagen University, 1980.

[9] G. Koch: A problem oriented software development method in computational linguistics (in Danish), Proc. De Nordiske Datalingvistikdagene, E. Lien (red.), Trondheim University, 1981, 47-63.

[10] G. Koch: Grammars and predicate calculus, DIKU report 81/16, Institute of Datalogy, Copenhagen University, 1981.

[11] G. Koch and K.B. Larsen: Logical prototyping in system development (in Danish), Proc. NordDATA Conf.,Göteborg, 1982, 1, 270-273.

[12] G. Koch: Stepwise development of logic programmed software development methods, DIKU report 83/5, Institute of Datalogy, Copenhagen University, 1983.

[13] B. Maegaard and H.Ruus; Multilingual syntax and morphology for machine translation, in K. Hyldgaard-Jensen and B.Maegaard (eds.): Machine translation and comptutational lexicography, Copenhagen 1982, 26-34.

[14] R. Montague: Formal philosophy, 1974.

[15] N.J.Nilsson: Principles of artificial intelligence, Springer-Verlag 1982.

[16] P.S.Olsen: Computational philosophy, future DIKU report, Institute of Datalogy, Copenhagen University, 1984.

[17] A.-L. Sågvall-Hein: A parser for Swedish, Uppsala University, UCDL-R-83-2.

[18] E. Upfal: Programming in predicate logic, DIKU report 81/9, Institute of Datalogy, Copenhagen University, 1981

[19] J.E. Ullman:Principles of database systems, 1980.

Kimmo Koskenniemi
Instit för allmän språkvetenskap
Helsingfors universitetet
Regeringsgatan 11-13
SF-00100 Helsingfors 10
Finland

A GENERAL COMPUTATIONAL MODEL FOR WORD-FORM RECOGNITION AND PRODUCTION

1. Generative phonology as a general formalism

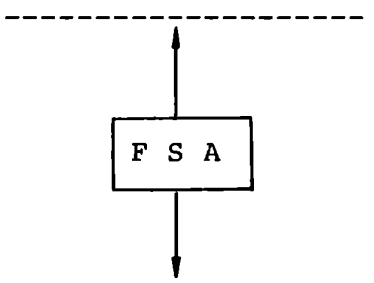
The formalism of generative phonology has been widely used since its introduction in the 1960's. The generative formalism is general enough to be applied to the morphology of any language, and its rules are stated in linguistically relevant terms. The morphology of a language is described by a set of rules which start from a underlying lexical representation, and transform it step by step until the surface representation is reached. So-called abstract phonology insists on invariant lexical representations for morphemes, and thus all variations among distinct surface forms must be accounted for by rules. This has led to a need for regulating the order in which the rules may be applied.

The generative formalism is conceptually unidirectional, because only the production of word-forms is guaranteed to be straight forward. As the rules are applied, they sometimes deform the context of other rules. Backwards application of rules would require either foresight or extensive trials of tentative rule applications.

The generative formalism has proven to be computationally difficult, and therefore it has found little use in morphological programs. Until recently only simulators of rules have been written and used for testing phonological descriptions or in the teaching of phonology.

2. The computational model of Kay and Kaplan

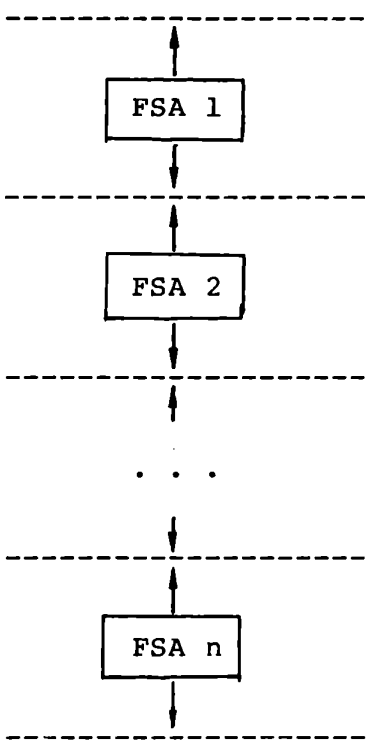
Martin Kay and Ron Kaplan from Xerox PARC noticed that each of the generative rewriting rules can be represented by a finite state automaton (or transducer) (Kay 1982). Such an automaton would compare two successive levels of the generative framework: the level immediately before application of the rule, and the level after application of the rule:



the string before application of the rule

the string after application of the rule

The whole morphological grammar would then be a cascade of such levels and automata:



lexical representation

after 1st rule

after 2nd rule

after (n-1)st rule

surface representation

A cascade of automata is not operational as such, but Kay and Kaplan noted that the automata could be merged into a single, larger automaton. Merging is possible by using the techniques of automata theory. The large automaton would be functionally identical to the cascade, although single rules could no more be identified within it. The merged automaton would be both operational, efficient and bidirectional. Given a lexical representation, it would produce the surface form, and, vice versa, given a surface form it would guide lexical search and locate the appropriate endings in the lexicon.

In principle, the approach seems ideal. But there is one vital problem: the size of the merged automaton. Descriptions of languages with complex morphology, such as Finnish, seem to result in very large merged automata. Although there are no conclusive numerical estimates yet it seems probable that the size may grow prohibitively large.

3. The two-level approach

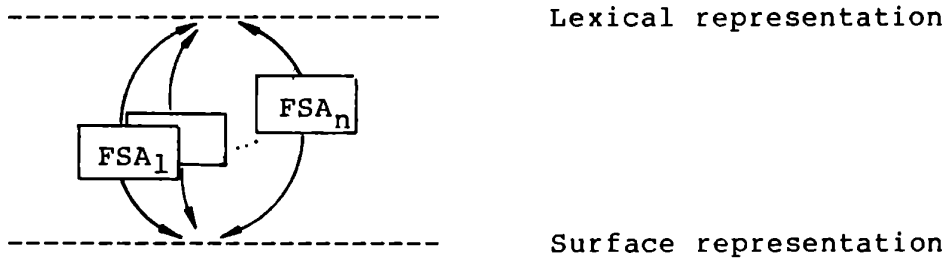
My approach is computationally very close to that of Kay and Kaplan, but it is based on a different morphological theory. It avoids the problems of their model. Instead of abstract phonology, I follow the lines of concrete or natural morphology (e.g. Linell, Jackendoff, Zager, Dressler, Wurzel).

The two-level model rejects abstract lexical representations, i.e. there need not always be a single invariant underlying representation. Some variations are considered suppletion-like and are not described with rules. The role of rules is restricted to one segment variations, which are fairly natural. Alternations which affect more than one segment, or where the alternating segments are unrelated, are considered suppletion-like and handled by the lexicon system.

The rules are completely parallel in the two-level model. The principle leads to a different theory of morphology. The new theory is incompatible with abstract phonology, but in agreement with (at least certain branches of) concrete/natural morphology.

4. Two-level rules and automata

There are only two representations in the two-level model: the lexical representation and the surface representation. No intermediate stages "exist", even in principle. The rules correspond to automata, as in the Kay and Kaplan model, but they operate in parallel instead of being cascaded:



The rule-automata compare the two representations directly, and a configuration must be accepted by each of them in order to be valid.

The two-level model (and the program) can operate in both directions: the same description can be utilized as such for producing surface word-forms from lexical representations, and for analyzing surface forms by finding the appropriate lexical entries and endings. In this sense the two-level model is bidirectional just as the Kay and Kaplan model.

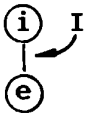
In addition to processual bidirectionality, the two-level model achieves a certain kind of conceptual nondirectionality. This means that the descriptions are not built on the concept of producing surface forms from underlying lexical representations. Instead, the rules describe the morphology and morphophonology in terms of correspondences. Thus, the two-level rules do not produce anything, they only relate two levels of representation to each other.

We take an example from Finnish morphology. The noun **lasi** 'glass' represents the productive and most common type of nouns ending in **i**. The lexical representation of the partitive plural form consists of the stem **lasi**, the plural morpheme **I**, and the partitive ending **A**. In the two-level framework we write the lexical representation **lasiIA** above the surface form **laseja**:

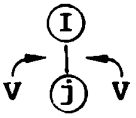
Lexical representation:	l a s i I A
Surface representation:	l a s e j a

This configuration exhibits three morphophonological variations:

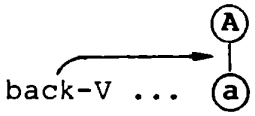
(1) Stem final **i** is realized as **e** in front of typical plural forms, i.e. when **I** follows on the lexical level, schematically:



(2) The plural **I** itself is realized as **j** if it occurs between vowels on the surface, schematically:



(3) The partitive ending, like other endings, agrees with the stem with respect to vowel harmony. An archiphoneme **A** is used instead of two distinct partitive endings. It is realized as **ä** or **a** according to the harmonic value of the stem, schematically:

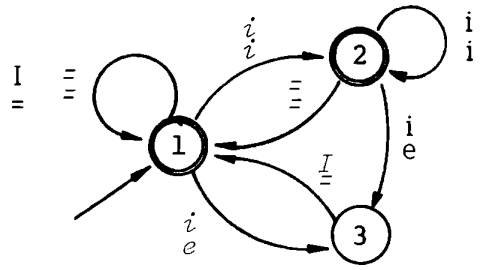


As we can see, the two-level rules may refer both to the lexical and to the surface representations.

In order to see how finite state automata can be used for describing such morphophonological alternations, we construct the three automata which perform the checking needed for the three alternations mentioned above. Instead of single characters, the automata accept character pairs. Each of the three automata must accept the following sequence of pairs:

l	a	s	i	I	A
l	a	s	e	j	a

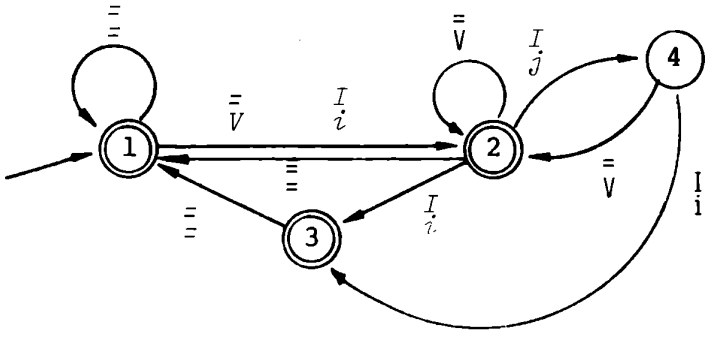
The task of the first rule-automaton is to permit the pair $\frac{i}{e}$ if and only if the plural I follows. The following automaton with three states (1, 2, 3) performs this:



State 1 is the initial state of the automaton. If the automaton receives pairs other than $\frac{i}{e}$ or $\frac{i}{i}$ it will remain in state 1 (the symbol $\frac{=}{=}$ denotes "any other pair"). Receiving a pair $\frac{i}{e}$ causes a transition to state 3. States 1 and 2 are final states (denoted by double circles), i.e. if the automaton is in one of them at the end of the input, the automaton accepts the input. State 3 is, however, a nonfinal state, and the automaton should leave it before the input ends (or else the input is rejected). If the next character pair has plural I as its lexical character (which is denoted by $\frac{I}{=}$), the automaton returns to state 1. Any other pair will cause the input to be rejected because there is no appropriate transition arc. This part of the automaton accomplishes the "only if" part of the correspondence: the pair $\frac{i}{e}$ is allowed only if it is followed by the plural I.

The remaining state 2 is needed for the converse statement. If a lexical i is followed by plural I, we must have the correspondence $\frac{i}{e}$ and nothing else on the surface. Thus, if we encounter a correspondence of lexical i other than $\frac{i}{e}$ (these are denoted by $\frac{i}{=}$) it must not be followed by the plural I. Anything else ($\frac{=}{=}$) will return the automaton to state 1.

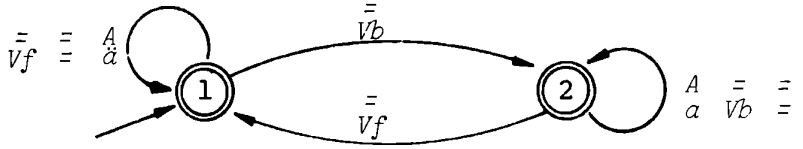
The automaton for plural I between vowels consists of four states:



The automaton remains in the initial state 1 until it enounters a surface vowel. It advances to state 2 after a surface vowel ($\bar{\bar{V}}$), but returns to state 1 if something else (i.e. a consonant) follows. The correspondence $\frac{I}{j}$ is allowed only in state 2 (the only transition arc labelled with $\frac{I}{j}$ starts from 2). Receiving this pair leads to state 4, which is nonfinal because the right context must also be satisfied. The only escape from state 4 is via a surface vowel, anything else terminates the execution because of a missing transition arc. This part of the automaton (1, 2, 4) checks that $\frac{I}{j}$ occurs only in the correct context.

The plural I is usually realized as i on the surface, but between vowels this possibility is excluded. This is accomplished by state 3. The default correspondence $\frac{I}{i}$ may be preceded or followed by surface vowels, but not at the same time. Thus, there is an appropriate transition from 1 to 1, but from state 2 (where a vowel has been detected at the left) we go to state 3. There anything else but a surface vowel will return us to state 1. The absence of an arc for $\bar{\bar{V}}$ excludes the default occurrence between vowels.

The automaton for vowel harmony is a simple one. It has only two states, both of which are final states:



The automaton starts from state 1 and remains there if the word-form is front harmonic. Detection of a back harmonic vowel (Vb = **a, o, u**) will cause a transition to state 2. In the front harmonic state 1 only $\frac{A}{ä}$ is allowed, and in the back harmonic state 2 only $\frac{A}{a}$.

As it stands now, the two-level program accepts the rules as tabular automata, e.g. the first rule automaton is coded as:

```
"i - e in front of plural I"  3  4
                 i   i   I   =
                 i   e   =   =
           1:    2   3   1   1
           2:    2   3   0   1
           3.    0   0   1   0
```

This entry format is, in fact, more practical than the state transition diagrams. The tabular representation remains more readable even when there are half a dozen states or more. It has also proven to be quite useful even for those who are linguists rather than computer professionals, and it has been applied in descriptions for several languages (Karttunen & Wittenburg 1983, Alam 1983, Khan 1983, Lun 1983).

There is, however, an obvious need for using a more rule-like formalism instead of the automata. There is a notation for two-level rules (Koskenniemi 1983), according to which the above automaton could be written as:

$$\underset{e}{i} \quad <=> \quad -- \quad \underset{=}{I}$$

This formalism is useful in schetching two-level descriptions with pencil and paper. A compiler accepting rules in such a formalism and automatically transforming them into finite state automata has been planned.

5. Two-level lexicon system

Single two-level rules are at least as powerful as single rules of generative phonology. The rule component of the two-level model as a whole (at least in practical descriptions) appears to be less powerful, because of the lack of extrinsic rule ordering. Within the two-level model, certain types of variations are described in the lexicon rather than by using rules.

Variations affecting longer sequences of phonemes or where the relation between the alternatives is phonologically otherwise nonnatural, are described by giving distinct lexical representations. Generalizations are not lost since insofar as the variation pertains to many lexemes, the alternatives are given as a minilexicon referred to by all entries possessing the same alternation.

The alternation in words of the following types are described using the minilexicon method:

hevonen	-	hevosen		'horse'
vapaus	-	vapautena	- vapauksia	'freedom'

The lexical entries of such words gives only the nonvarying part of the stem and refers to a common alternation pattern nen/S or s-t-ks/S:

```
hevo      nen/S     "Horse S";
vapau     s-t-ks/S  "Freedom S";
```

The minilexicons for the alternation patterns list the alternative lexical representations and associate them with the appropriate sets of endings:

```
LEXICON nen/S        nen   S0    "";
                     sE    S123  ""
LEXICON s-t-ks/S     s     S0    "";
                     TE    S13   "";
                     ksE   S2    ""
```

6. Current status of the two-level program

The two-level program has been implemented first in PASCAL language and can be run at least on the Burroughs B7800, DEC-20, and large IBM systems. The program is fully operational and reasonably fast (about 0.1 CPU seconds per word although hardly any effort has been spent to optimize the execution speed). Lauri Karttunen and his students at the University of Texas have implemented the model in INTERLISP (Karttunen 1983, Gajek & al. 1983, Khan & al. 1983). The execution speed of their version is comparable to that of the PASCAL version. Nothing would prevent the use of the PASCAL version on micro-computeres with e.g. 128 kB memory.

The model has been tested by making a comprehensive description of Finnish morphology covering all types of nominal and verbal inflection including compounding. Karttunen and his students have made two-level descriptions of Japanese, Ruma-

nian, English and French. At the University of Helsinki, two descriptions are reaching completion: one of Swedish and one of Old Church Slavonic.

The two-level model could be part of any natural language processing system. Especially the ability both to analyze and to generate is useful. Systems dealing with many languages, such as machine translation systems, could benefit from the uniform language-independent formalism.

The accuracy of information retrieval systems can be enhanced by using the two-level model for discarding hits which are not true inflected forms of the search key. The algorithm could be also used for detecting spelling errors.

References

Alam, Y., 1983. A Two-Level Morphological Analysis of Japanese. TLF 22, pp. 229-253.

Gajek, O., H. Beck, D. Elder, and G. Whittemore, 1983. KIMMO: LISP Implementation. TLF 22, pp. 187-202.

Karttunen, L., 1983. KIMMO: A General Morphological Processor. TLF 22, pp. 165-186.

Karttunen, L., and K. Wittenburg, 1983. A Two-Level Morphological Description of English. TLF 22, pp. 217-228.

Kay, M., 1982. When meta-rules are not meta-rules. In: Sparck-Jones, K. and Y. Wilks, 1982. Automatic natural language Parsing. University of Essex, Cognitive Studies Centre. (CSCM-10.)

Khan, R., 1983. A Two-Level Morphological Analysis of Rumanian. TLF 22, pp. 253-270.

Khan, R., J. Liu, T. Ito, and K. Shuldberg, 1983. KIMMO User's Manual. TLF 22, pp. 203-216.

Koskenniemi, K., 1983a. Two-level Model for Morphological Analysis. Proceedings of IJCAI-83, pp. 683-685.

---, 1983b. Two-level Morphology: A General Computational Model for Word-Form Recognition and Production. University of Helsinki, Dept. of General Linguistics, Publications, 11.

Lun, S., 1983. A Two-Level Analysis of French. TLF 22, pp. 271-278.

TLF: Texas Linguistic Forum. Department of Linguistics, University of Texas, Austin, TX 78712.

Gunnel Källgren
Institutionen för lingvistik
Stockholms universitet
106 91 Stockholm

HP - A HEURISTIC FINITE STATE PARSER BASED ON MORPHOLOGY

In the project "Satsanalys utifrån morfologiska kriterier", financed by HSFR, I work with a computerized model for parsing of Swedish sentences. (A more detailed description of this in Swedish is Källgren 1983.) It is connected with other work carried out by Benny Brodda, Stockholm, so we use the general label Heuristic Parsing (HP) for our joint efforts.

To be able to explain the ability of human beings to perceive and interpret language as quickly and as correctly as is actually done, we have to make certain assumptions. We must e g assume that humans have a highly developed skill for parallel processing, not only in the sense that one can take part in a conversation while driving a car and rubbing one´s nose at the same time. More important in this connection is that language perception and language understanding can be fruitfully regarded as a system of parallel processes. Another important assumption is that most of these processes work in a heuristic manner, in the sense that they strive for optimal probability rather than absolute correctness, that in choice situations they take a fair guess or a short-cut rather than a toilsome testing of every alternative.

In our HP-project, we try to draw some consequences of these assumptions. For a start, we have picked out what we believe to be one of the many processes involved and furthermore we treat it as consisting in its turn of several parallel subprocesses. We also strive to make it work "heuristically" in the way sketched above, i e more by trial and error than by careful stepwise analysis. In this way we will try to build and study a model that will perform a few of the many tasks a human mind can manage.

The major process we have chosen has to do with perception and interpretation of surface morphological features. This, we think, plays an important role in word class assessment and deciding of constituent structure. To be able to really interpret the full text, enormous amounts of semantic as well as pragmatic information and general knowledge of the world is necessary. This, we do not even try to do. Instead we want to stick strictly to a purely morphological surface level in order to investigate how much information about language structure can be gathered from that level. It may not be possible to say precisely HOW much, but we have already found that it is surprisingly much, so much that it has to be payed more heed to it also in more lexically based models.

To give a technical overview of the system, it can be described as a series of Finite State Pattern Matching machines (FS-PM), where the superordinate system governing the subprocesses can also be regarded as a Finite State machine. This will make the system equivalent to an Augmented Transitory Network.

Each of the subordinate FS-PM machines corresponds to a separate computer program. For computational reasons the programs are run sequentially on texts, but are meant to mimic parallel processing. Each program delivers its result as an updated version of the original text. The analysis is entered directly into the text file in the form of a marked bracket notation (but without the brackets to save space, cf below).

Normally the programs are used for ordinary running text in computer readable form. Before and after the analysis programs proper, programs for standardizing and cleaning up input and output are run, but apart from that, no pre-processing is needed. A full lexicon is not needed either. Literally any Swedish text can be taken and processed in a short time and to a low cost, but the results will improve somewhat if some rules are designed to handle the peculiarities of a specific text type, e g rules for legal texts as opposed to fictional prose.

The analyzing programs are all based on pattern matching procedures on different levels. Strings and configurations in the text file are matched against the rules of the programs that scan the text in the manner of a Turing machine. The first three analysis programs have single words and parts of words as their range. The next three build up different kinds of constituents and the last one takes in larger sentence patterns.

The description to follow covers the state of the system in the autumn of 1983. It is important to remember that the system is still only in the first part of its development. There is much left to be improved and refined, but even so, the present results are quite interesting.

The total number of lexicon rules in the system is at present about 300. Most of them are in the first program, MK (Mark). MK marks the words for word class according to the code given in Figure 1, which is used throughout the analysis. The code letter is put before and after the identified word, thus making up bracket and mark in one.

(Fig. 1)

a = adverb(ial)	n = noun (phrase)
b = particle	o = possessive
c = clause	p = prepositional phrase
d = determiner	q = quantifier
e = preposition	r = pronoun
f = copula	s =
g = auxiliary	t =
h = form of 'have'	u = supine verb
i = infinite verb	v = finite verb
j = adjective	w = finite verb or noun
k = conjunction	x = finite verb or adjective
l = proper name	y = noun or adjective
m = infinite marker	z = adverb or adjective

MK contains function words from closed word classes, e g prepositions, pronouns and conjunctions. The number of words in the lexicon will probably have to be increased with several more adverbs, as they are often difficult to identify on purely morphological and distributional criteria. Still, the lexicon will never grow very big, it will in general be common to all text types and it will mostly contain words with a grammatical function rather than real content words.

Some typical results from MK are shown in (2).

(2) ePÅe ´on´, kOCHk ´and´, fÄRf ´is´

MK does not perform a real analysis, just an identification of entire word forms. The next program, SM (Swedish Morphology), carries out an advanced morphological analysis of every word. Identified prefixes and derivational and flexional suffixes are segmented and marked, as are some segmentable final letters. If information about word class can be extracted from the internal structure of a word, "bracket" marking according to the code in Figure 1 is also entered.

Words ending in e g NING or SION/TION (plus possible flexional endings) are always nouns in Swedish and get marked as such by SM, while words ending in LIG/LIG´T/LIG´A are likely to be adjectives and marked as such, etc.

Some heuristic devices come to play in this program. Some word forms can e g be either nouns or adjectives, like VAKEN, which is either the definite form of the noun VAK "hole in the ice", or the n-gender singular form of the adjective "awake". In cases like this, the definite decision is simply postponed until more information is available. The ending EN is segmented and the word is marked as ambiguous between noun and adjective. The same method is used for several other word class ambiguities.

Some output from SM is shown in (3). It can be a word that is segmented and marked, either as unambiguous or as ambiguous, or segmented and not marked, where the morphological information is not sufficient for identification at this stage. Words that have no recognizable inner analysis are left unaffected by the program.

(3) nFÖR>KORT<NING=ENn ´the abbreviation´
yVAK=ENy ´awake´/´hole in ice´
FÖR>KLAR´A ´explain´

The next program, PF (prefix), builds on the analysis done by SM. PF contains a lexicon with irregular verb forms that cannot be identified as verbs on morphological basis. They are listed here rather than in MK as they often appear as parts of conjoined words and MK only can work on unanalyzed words. PF will recognize words from its lexikon both as single words and as word stems, marking as verbs both SÅG "saw" and FÖR>SÅG "provided". Furthermore, the program has some heuristic default values. Words with a prefix and ending in vowel + R are also supposed to be verbs, this time in the present tense, as FÖR>SER "provides". Words with a prefix and ending in a vowel are too ambiguous to be marked yet, like FÖR>KLAR´A infinite

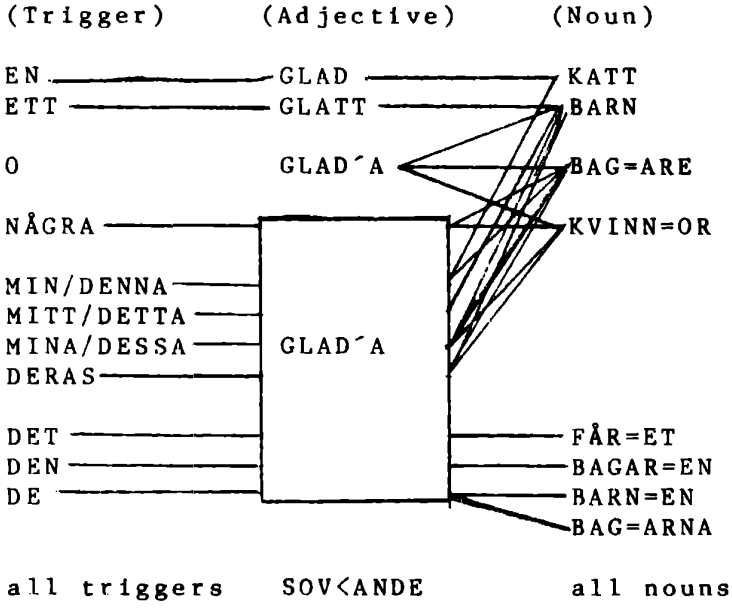
verb "explain", FÖR>MÅG´A indefinite noun "ability", FÖR>SYNT´A adjective "timid". Words with a recognizable adjective ending were identified already in SM, FÖR>KLAR<LIG´A "explicable", as were preterite verbs, FÖR>KLAR=ADE "explained". All other words with a prefix are simply assumed to be nouns. This holds to a remarkably high degree but is a point where we have to swallow some errors in the hope that they will be amended further on.

PF also contains some other types of possible word stems that are treated in a similar way. Some outputs from PF:

(4) vFÖR>SÅGv ´provided´
vFÖR>SERv ´provides´
nFÖR>SLAGn ´suggestion´

This is as far as one can get on the level of single words. Next come programs for building up constituents; noun phrases, prepositional phrases, and infinite phrases in turn. The monstrous diagram in (5) is an attempt at drawing a flow chart for the combinability patterns in Swedish noun phrases. The diagram is not exhaustive but it covers a fair amount of the noun phrases encountered in running text and gives a picture of the complexities involved.

(Fig. 5)



The implementation of diagram (5) in the program NP provides a very clear illustration of the finite state character of the programs. Every type of trigger is given its unique internal state and the different forms of modifiers and nouns have conditions on what internal states they accept. A condition can contain one single state, as that associated with nouns ending in =ORNA, which can only be preceded by the plural definite article DE, possibly followed by adjectives ending in ´A or NDE. A condition can also contain a whole set of states, like that for words ending in =ARE that have a very high combinability. Even so, the intervening adjectives must always be checked EN GLAD BAG=ARE "a happy baker" is OK, as is NÅGRA

GLAD´A BAG=ARE "some happy bakers", but neither *EN GLAD´A BAG=ARE nor *NÅGRA GLAD BAG=ARE, which are both excluded by the rules.

The NP-rules also dissolves several of the ambiguities noted in earlier programs. To return to the word VAK=EN that SM marked as ambiguous, it might appear in constructions like (6) and (7).

(6) qENq yVAK=ENy FLICK´A ´an alert girl´

(7) dDENd MÖRK´A yVAK=ENy ´the dark hole in the ice´

After the indefinite trigger EN in (6) the ending =EN is allowed to appear on an adjective but not on a noun while the unmarked word ending in ´A is a possible noun but excluded as an adjective. In (7) however, ´A is one of the two possible adjective endings after the definite article and =EN as a noun ending is congruent with both DEN and ´A. The noun phrases will then be:

(6´) nEN+VAK=EN+FLICK´An

(7´) nDEN+MÖRK´A+VAKENn

Had the article instead been DET, =EN would have been excluded as either noun ending or adjective ending, and the noun phrase would have terminated after MÖRK´A, treating the adjective as a nominalization, which is also quite possible and not uncommon in Swedish.

The program PP builds up prepositional phrases by connecting a preposition (marked ´e´ by MK) to a following noun phrase. It also identifies nouns by connecting prepositions and words that were formerly unmarked or marked as ambiguous between noun and something else. PP also builds up conjoined noun phrases within the range of a preposition. Some results are shown in (8).

(8) pMED+GLÄDJEp ´with joy´

pTILL+FLICK=AN+,+POJK=EN+OCH+DERAS+MORp
´to the girl, the boy and their mother´

IF (for InFinite) identifies infinite constructions. Those are often discontinuous. When they are triggered by the infinite marker ATT (which can also be a subordinating conjunction) adverbials are allowed to intervene between the trigger and the infinite verb. When the trigger is an auxiliary verb, adverbials and at most one noun phrase or one pronoun are allowed to intervene.

The IF-rules are alerted by one of the above triggers and then scans the string, only allowing constituents of the types mentioned to pass, until either a disallowed constituent breaks the scan or a possible infinite is encountered. Possible infinites are unmarked words with one of three possible surface forms: ending in a segmentable A, HOPP´A "jump", FÖR>LOR´A "loose"; monosyllables ending in any vowel, STÅ "stand"; or words with a segmented prefix and a monosyllabic stem as above,

FÖR>STÅ "understand". This is again based on probabilities, but the pattern "possible trigger (+ allowed constituent) + word of the described form" is strong enough to pick out almost every infinite verb with very few overgenerations. Chains of auxiliary verbs can also be managed. Illustrations:

(9) nDETn gKUNDEg rHANr aINTEa iFÖR>KLAR´Ai
´that, he could not explain´

mATTm aTROLIGENa aINTEa aBARAa iGÅi
´to probably not just go´

The last of the analyzing programs, DA (DisAmbiguation), works on the level of sentence syntax. The development of this program has only just started. Much more can be done on this level, e g identification of clausal structure, but at present it is mainly used for disambiguating words that are marked as ambiguous.

A word like FÅNG=AR will have been marked with ´w´by SM, as it can be either a verb in present tense, "catches", or a plural noun, "prisoners". If nothing else has changed the marking during the processing, it is likely that DA will.

(10) wFÅNG=ARw vFLYDDEv . ´prisoners escaped´
(11) aDÄRIFRÅNa vFLYDDEv wFÅNG=ARw . ´from there escaped prisoner
(12) nFLICK=ANn wFÅNG=ARw nHUND=ENn . ´the girl catches the dog´

In (10) and (11) ´w´ appears before or after ´v´, e g a finite verb. Two finite verbs are not allowed in the same clause and there is nothing to signal clause boundary, so the w-word must be a noun. In (12) ´w´ both precedes and follows a noun. Either condition would be sufficient to turn ´w´ into ´v´ for finite verb, as noun phrases are very rarely given on a row with nothing between them. DA contains several similar mechanisms for changing earlier marks.

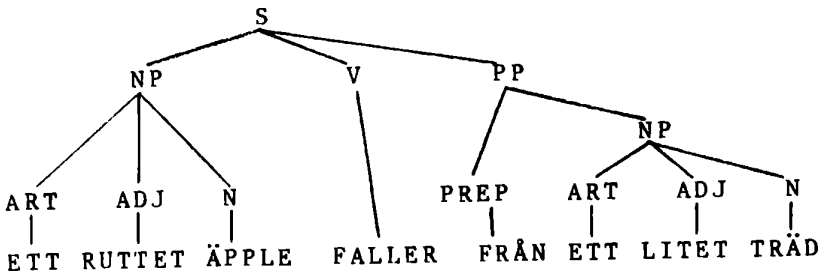
(10´) nFÅNG=ARn vFLYDDEv .
(11´) aDÄRIFRÅNa vFLYDDEv nFÅNG=ARn .
(12´) nFLICK=ANn vFÅNG=ARv nHUND=ENn .

This is the end of the analysis part of the system. Some analyzed sentences are shown in (13). When buildin constituents I take away their inner analysis and just put in +-signs. This is a deliberate choice. If the information was kept throughout the processing the output would look even messier than in (13), but it would contain literally all information, except the dominating S but including labels, that is needed to construct the tree structures in (14). (The underlined words are the only ones that have appeared in a lexicon.)

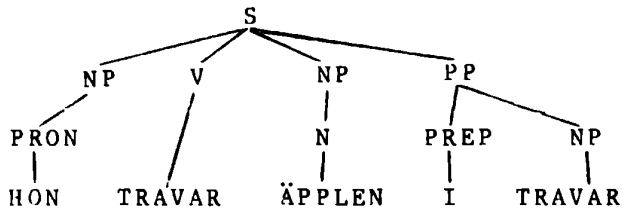
(13) a, nETT+RUTT=ET+ÄPPLEn vFALL=ERv pFRÅN+ETT+LIT=ET+TRÄDp .
´a rotten apple falls from a small tree´

b, nHONn vTRAV=ARv nÄPPL=ENn pI+TRAV=ARp .
´she piles apples in piles´

(14) a,



b,



All this shows clearly that this heuristic system can be used as a parser. It can parse the sentences of any Swedish text quickly and cheaply and with a result that is quite good, even when it is not perfect. This is fair enough, I am surprised myself that it works so well with so simple means, but what makes it interesting is not the degree of success or failure, but the fact that it is a truly linguistic system. The results are not reached by ad hoc solutions and smart programming in general, but by implementing linguistic patterns and probabilities based on linguistic intuitions.

We must ask ourselves why this model works so well. How can the information gained from morphology and a closed set of function words be sufficient to build up the whole structure of a sentence? I want to claim that morphology has an impact on language perception that has been grossly underestimated. Language is an economical system, we would not carry around so much morphology in language if we did not use it for something important. It seems likely that a "morphological interpreter" plays an important part as one of the several parallel processes at work in natural language understanding. Its results are of course always checked against the results from different semantic and pragmatic interpreters. In case of discrepancies, the semantics will probably always win, but where the results are in accordance, which they mostly are, the morphological interpreter may well be the process that gives sentences their grammatical structure.

If we regard the HP-system as a first attempt at modelling this morphological interpreter it means that the system also has a psycholinguistic relevance that deserves to be seriously considered and investigated.

Källgren, Gunnel (1983): Substantivjakten.
To be published as IRI-PM, Institutet för Rättsinformatik, Stockholms universitet.

Bente Maegaard,

Københavns Universitet,
Institut for anvendt og
matematisk lingvistik,
Njalsgade 96
2300 København S

Regelformalismer til brug ved datamatisk lingvistik.

Når man i 'gamle dage' lavede et system til sproglig analyse, gjorde man det oftest på den måde, at man skrev et program, i hvilket man udtrykte al den viden, der skulle bruges. Det, der især er interessant her, er at den grammatiske viden der skulle bruges, var udtrykt i selve programmet. Programmøren og sprogforskeren var måske en og samme person, men selv i det tilfælde er det uhensigtsmæssigt - af mange kendte grunde.

Derfor er man da også mere og mere gået over til at skrive systemer, hvor program og data holdes adskilt - og grammatikker altså opfattes som data. Disse grammatikker skrives i et bestemt format; det er det jeg ovenfor har kaldt regelformalisme.

Der findes efterhånden en række sådanne parsere med forskellige regelformalismer. Disse formalismer afviger fra hinanden på forskellige måder. Jeg vil her især interessere mig for, hvordan det ser ud fra brugerens - lingvistens - synspunkt, og mindre for, hvordan det er implementeret.

De oplagte krav, som en lingvist stiller til en regelformalisme, er at den er rimeligt naturlig - man skal kunne udtrykke sproglige fakta på en rimeligt intuitiv måde - og at den er klar og overskuelig.

EUROTRA.

Det projekt, som jeg i det følgende vil referere til, er EUROTRA, EF's maskinoversættelsesprojekt. Projektet blev vedtaget i november 1982 og har en løbetid på 5,5 år. Efter 5,5 år skal en prototype af systemet være færdig; den skal kunne oversætte mellem de 7 EF-sprog (dansk, engelsk, fransk, græsk, italiensk, nederlandsk og tysk). De tekster, der skal

kunne oversættes skal ligge inden for et bestemt emneområde (f.eks. informationsteknologi), med et ordforråd på ca. 20.000 ord.

Projektperioden er inddelt i 3 faser, á 2, 2 og 1.5 år. I den første fase, som vi nu er i gang med, skal både den lingvistiske og den programmelmæssige side - og koblingen mellem dem - defineres.

EUROTRAs regelformalisme.

Regelformalismen er en meget væsentlig del af grænsefladen mellem det sproglige og det programmelmæssige: man skal kunne udtrykke de sproglige fakta, som den lingvistiske model lægger op til, og samtidig bestemmer den valgte type af programmel en række egenskaber ved formalismen.

En hovedfilosofi i EUROTRA's programmelsystem er, at det skal være deklarativt. Hermed menes, at den lingvistiske viden om, hvad der er et sprogligt faktum, er adskilt fra den procedurelle viden om hvordan og hvornår denne viden skal udnyttes. I den yderste konsekvens betyder dette, at lingvisten skriver sine regler og at han ikke ved, i hvilken rækkefølge, de bliver anvendt. Det er dog næppe muligt at forestille sig f.eks. hele analysen af dansk skrevet i et stort ustruktureret deklarativt system. Derfor er EUROTRA's programmel bygget som et såkaldt 'Controlled Production System', dvs. et produktionssystem udstyret med et kontrolsprog. Kontrolsproget giver mulighed for at samle regler i 'grammatikker' og endvidere for at bestemme, om en grammatik skal anvendes kun én gang eller gentages, om grammatikker på samme niveau skal anvendes parallelt eller sekventielt osv. Dette er den mest procedurale del af formalismen.

Der gælder to hovedprincipper for den deklarative del af EUROTRA's regelformalisme: den skal være generel og den skal kunne beskrive de relevante data.

Kravet om generalitet skal forstås således: EUROTRA har 3 hovedmoduler: analyse, overførsel og generering, og det er hensigten, at samme formalisme skal kunne bruges i alle moduler. Denne formalisme skal yderligere kunne anvendes til

at udtrykke forskellige lingvistiske strategier, idet de deltagende grupper fra de forskellige lande nogenlunde frit skal kunne vælge strategi (inden for de rammer, som det valgte produktionssystem sætter).

Udover at formalismen skal kunne beskrive forskellige typer af lingvistisk strategi, skal den som nævnt kunne bruges på alle niveauer af oversættelsesprocessen: til grammatikregler såvel som ordbogsregler, til morfologi såvel som kasusgrammatik og syntaks, til regler med stor kompleksitet såvel som til helt enkle regler.

Dette hovedkrav om, at formalismen skal være generel er af to grunde delvis modstridende med kravet om naturlighed og klarhed. For det første må man tage hensyn til de mest komplekse regler, når man udformer formalismen, og det betyder, at de enkle regler kan blive unødigt komplicerede at udtrykke. For det andet er det jo sådan, at jo mere skræddersyet en formalisme er til et bestemt formål, jo nemmere er det at bruge den. Der er således gode argumenter for at udvikle særlige formalismer (og særlige fortolkere) til specielle velafgrænsede delopgaver. Man vil dog udarbejde en generel formalisme, der kan bruges overalt, således at eventuelle specialformalismer kun er et supplement.

Jeg har ovenfor nævnt to hovedkrav til formalismen:

1) den skal være deklarativ - og jeg har nævnt, at det ikke helt er opfyldt, og at det næppe heller er nogen god idé at hævde kravet rigoristisk,
2) den skal være generel. Dette krav kan opfyldes, men det er formentlig heller ikke her hensigtsmæssigt at overholde kravet strengt.

Det tredie hovedkrav er, at formalismen skal kunne håndtere de data, vi arbejder med i EUROTRA. Dette krav er der ingen mulighed for at slække på.

De data, der skal behandles, er træstrukturer med komplekse oplysninger (dekorationer) på kunderne. Træstrukturerne skal kunne se ud på alle mulige måder; men dekorationerne kan kun indeholde bestemte oplysninger i bestemte mønstre. Man kan derfor lade brugeren erklære, hvilke dekorationer, der

er mulige. Det er praktisk både for det fortolkende program og for brugeren.

Det generelle regelformat.

Det generelle format for en regel i EUROTRA-formalismen er den velkendte genskrivningsregel:
venstre side → højre side
Her består såvel venstre side som højre side af træstrukturer med knudeoplysninger. Formatet ser således ud:

geometry	< specifikation af et træ > ➡ < specifikation af et træ >
conditions	< betingelser på dekorationerne >
assignments	< tilskrivning af værdier til højresiden>

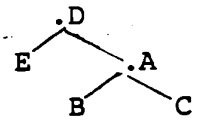
F.eks. vil

geometry	A + B ➡ C(A' + B')
conditions	MS of A = ADJ and
	MS of B = NOUN and
	GENDER of A = GENDER of B and
	NUMBER of A = NUMBER of B
assignments	A':=A;
	B':=B;
	MS of C:=NP
	GENDER of C:=GENDER of B;
	NUMBER of C:=NUMBER of B.

danne et substantivsyntagme af et adjektiv og et substantiv, der stemmer overens i køn og tal. (MS betyder morpho-syntactic class, resten skulle være umiddelbart forståeligt).
Det kan måske føles en lille smule omstændeligt at skrive regler på denne måde; men det kan næppe gøres meget nemmere.

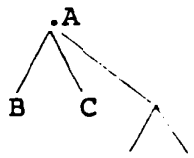
En af de ting, vi har diskuteret, er om træstrukturering i geometrien altid skal være et træ med rod eller om det godt kan være et deltræ af et større træ, altså om træet

også skal kunne findes i denne datastruktur



Sommetider vil lingvisten gerne have at træet bliver fundet, uanset hvor det er placeret, sommetider er han kun interesseret, hvis det har en rod. Det skal derfor være muligt at specificere dette. Her adskiller systemet sig fra f.eks. Q-systemet, hvor man kun kan få adgang til kunder i et træ ved at specificere hele vejen fra roden. Det at specificere hele denne vej er besværligt, men meget værre er det, at man må lave lige så mange regler, som der findes mulige træstrukturer, som deltræet kan indgå i.

Et andet særligt tilfælde, er noget svarende til følgende



hvor B og C ikke er de eneste datterknuder til A. Det er relativt nemt at klare at beskrive i en formalisme: i EUROTRA skriver man A(B+C +#X), hvor #X kan være tom eller bestå af et eller flere træer, hvis der kan være grene til højre for C. Hvis der også kan være grene til venstre for B og mellem B og C, må man skrive A(#Y+B+#Z+C+#X).

En sidste ting, jeg vil nævne, omkring træstrukturerne, er, at de selvfølgelig normalt er ordnede, dvs. i træet A(B+C) står B til venstre for C. Orden er som regel en relevant egenskab ved et træ. Men netop i oversættelsesprocessen er der ét tilfælde, hvor man evt. kan være uinteresseret i orden. Det er i genereringsfasen. Her har man ved udgangen

fra overførselsfasen fået en træstruktur, hvor ordene står i en eller anden rækkefølge, som ikke nødvendigvis er den rigtige på målsproget. Her vil det kunne være praktisk, at man kan skrive konstituenterne op, meddele at de må betragtes som uordnede, og fortælle, hvilken orden man ønsker, de skal stå i. Alternativet er, at man må lave lige så mange regler, som der er 'forkerte' rækkefølger for ordene; dette er for det første besværligt, for det andet betyder det, at man skal forestille sig alle de mulige rækkefølger af konstituenter, hvilket ofte vil føles helt irrelevant.

Jeg har her nævnt nogle af de muligheder, vi mener der er for at lette lingvistens arbejde med at skrive regler vedrørende træstrukturer. Selv om implementeringen af dem gør systemet lidt mindre effektivt, er der tale om en god investering.

Brugergrænsefladen.
Brugergrænsefladen består dels af den/de regelformalismer, systemet tilbyder og dels af de editeringsfaciliteter, der stilles til rådighed.

Brugerens arbejde kan lettes meget ved at specielle editorer stilles til rådighed. F.eks kan en regeleditor automatisk bede om de 3 hovedelementer i en regel, og en ordbogseditor kan automatisk bede om at få udfyldt relevante felter (afhængig af allerede indtastet information, således f.eks. at man beder om køn for substantiver, men ikke for verber). Sådanne editorer vil blive udarbejdet.

Når spørgsmålet om brugergrænseflade føles så vigtigt i dette projekt, er det fordi 100-150 mennesker fordelt på 10 lande og endnu flere universiteter, skal arbejde intensivt med den, når projektet er i gang i fuld skala.

Alt hvad der kan gøres for at lette lingvisternes arbejde skal gøres, for det første fordi det hurtigt vil have tjent sig ind, for det andet - og ikke mindst - fordi det giver større sikkerhed mod fejl.

Litteratur.
Alain Colmerauer: Les systèmes-Q, TAUM, Montréal, 1970.
Anna Sågvall Hein: A parser for Swedish, UCDL, Uppsala, 1983.
Dieter Maas and Bente Maegaard: Syntax and Semantics of the EUROTRA Formalism, EEC, 1984 (ikke frit tilgængelig).

Esa Nelimarkka, Harri Jäppinen,
and Aarno Lehtola,
Helsinki University of Technology
Espoo, Finland

A COMPUTATIONAL MODEL OF FINNISH SENTENCE STRUCTURE[1]

Introduction

The present paper propounds an outline of a computational model of Finnish sentence structures. Although we focus on Finnish we feel that the ideas behind the model might be applicable to other languages as well, in particular to other inflectional free word order languages.

A parser based on this model is being implemented as a component of a larger system, namely a natural language data base interface. There it will follow a component of morphological analysis (see Jäppinen et al [8]); hence, throughout the present paper it is assumed that all relevant morphological and lexical information is computationally available for all words in a sentence. Even though we have a data base application in mind, sentence analysis will be based on general linguistic knowledge. All application dependent inferences are left to subsequent modules which are not discussed here.

The linguistic foundations

We shall freely borrow ideas of Anderson [1], [2], Tarvainen [11] and Pajunen [9], and Fillmore [3], [4] and Siro [10] concerning dependency and case grammars. We shall use the latter grammar to introduce semantics into syntacs and the former to give a basis for a functional syntax where the subordinate dependency relations are specified with formal binary relations. The structure which these grammars will impose on sentences is a dependency-constituency hybrid structure with labelled dependants, similar to the sister-dependency structure of Hudson [7]. We shall briefly explicate and reason our choices.

---

1 This research is supported by SITRA (Finnish National Fund for Research and Development), PL 329, 00120 Helsinki 12, Finland

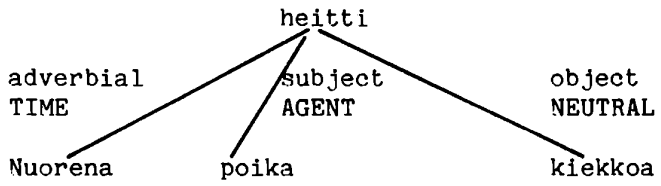
Firstly, Finnish is a "free word order" language in the sense that the order of the main constituents of a sentence is relatively free. Variations of word order configurations convey thematical and discursional information. Hence, we must be ready to meet sentences with seemingly odd word orders which in a given context, however, are quite natural.

The computational model should acknowledge this state of affairs and be adjusted to cope efficiently with it. This demands a structure within which word order variations can be conveniently described. An important case in point is to avoid discontinuities in the structure caused by transformations.

We argue that a dependency-constituency hybrid structure induced by a dependency grammar meets the requirements. This structure consists of part-of-whole relations of constituents and labelled binary dependency relations within a constituent.

The sentence "Nuorena poika heitti kiekkoa" ("As young, the boy (used to) throw the discus"), for example, will be given the structure

(1)



or, a linearized equivalent with labelled parentheses,

(2) $((\text{Nuorena})_{\substack{\text{advl}\\\text{TIME}}}\ (\text{poika})_{\substack{\text{subj}\\\text{AGENT}}}\ \text{heitti}\ (\text{kiekkoa})_{\substack{\text{obj}\\\text{NEUTRAL}}}\ )$,

where parentheses indicate constituent boundaries and, within each constituent, the word without parentheses is the head. (To be more precise, an inflected word appears as a complex of all its syntactic, morphological and semantic properties. Hence, our structure representation is a tree with nodes which are labelled with complex expressions.)

The advantage of such structures lies in the fact that many word order varying transformations can now be localized to a permutation of the head and its dependants in a constituent. As an example we have the permutations

(2') ((Poika)$_{subj}$ heitti (nuorena)$_{advl}$ (kiekkoa)$_{obj}$)

(2'') (Heittikö (poika)$_{subj}$ (nuorena)$_{advl}$ (kiekkoa)$_{obj}$)

(2''') ((Kiekkoako)$_{obj}$ (poika)$_{subj}$ heitti (nuorena)$_{advl}$)

Having reduced the depth of stuructures (by having a verb and its subject, object, adverbials etc. on the same level) we bypass many discontinuities that would have appeared in a deeper structure as a result of rising transformations.

The second argument for our choices is the well acknowledged prominent role of a finite verb in regard to the form and meaning of a sentence. The meaning of a verb (or a word of other categories as well) includes knowledge of its deep cases, and the choice of a particular verb to express this meaning determines to a great extent what deep cases are present on the surface level and in what functions.

Moreover, due to the free word order of Finnish, the main means to indicate the function of a word in a sentence is the use of surface case suffixes, and very often the actual case depends not only on the intended function or role but on the verb as well.

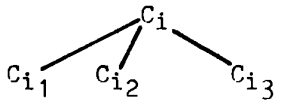
Both of these arguments speak well for a combination of dependency and case grammars. We claim that such a combination can be put into a computational form, and that the resulting model is one which efficiently takes advantage of the central role of the constituent head in the actual parsing process. We shall outline below how this can be done with finite 2-way tree automata.

Specification of dependency structures

In a dependency approach the description of a constituent associates the head with a specification of its dependants and their order.

Hays [5] used the notation

(3)



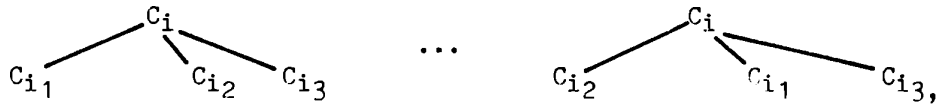
or its linear equivalent,

(3') $C_i(C_{i_1} * C_{i_2}\ C_{i_3})$,

to describe the dependants of a word in a category C_i.

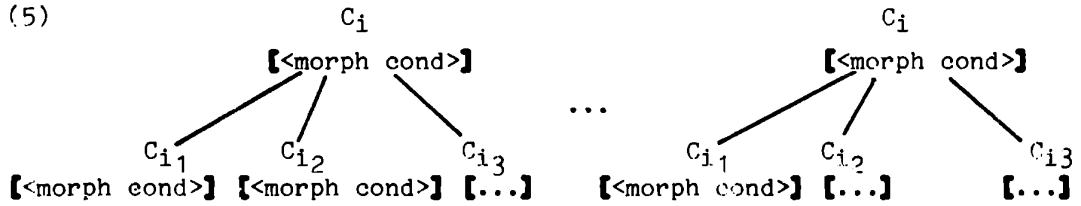
Such a formalism is not suitable for our purposes. Firstly, due to free word order, one would either have to postulate permutation transformations or directly proliferate the "frame" (3) to the different word order configurations, to

(4)



for example. Secondly, in case of an inflectional language, the definition of dependants with restrictions on mere categories would not suffice. Morphological conditions on the head and its dependants are needed, too. This would lead to another kind of frame multiplication:

(5)

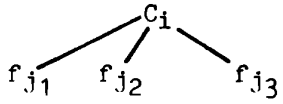


where the different frames describe mutually exclusive morhological situations. Thirdly, we wish to know the syntactic function of a dependant, not only its place, category, number, case etc. We shall now show how to overcome these difficulties.

A 2-way finite tree automaton -model

We begin with a unification of the frames (5) by abstracting the laws that govern the restrictions on the head and its dependants in such a sequence. These abstractions, which we call functions, are defined as formal binary relations on the set of inflected words (which are considered as complexies of all their relevant properties). It is no surprise that these abstract functions will correspond to — and are named after — the traditional parsing categories like subject, object, adverbial, genetive attribute etc. After the adoption of these abstract functions the sequence (5) reduces to a single "relation frame"

(6)

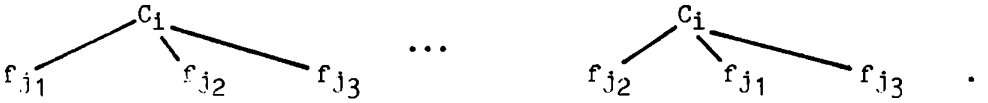


or be a linear expression

(6') $$C_i(f_{j_1} * f_{j_2}\ f_{j_3}) \quad ,$$

where the f_j's stand for functions.

We are still left with the permutational variants

(7) 

Our final step is to combine these relation frames with a structure building 2-way finite tree automaton. At this point we also give a computational form to the idea of the dominance of the head of the constituent with respect to its form and meaning.

Recall that a standard 2-way finite automaton consists of a set of states, one of which is a starting state and some of which are final states, and of a set of transition arcs between the states. Each arc recognizes a word, changes the state of the automaton and moves the reading head either to the left or right. (Cf. Hopcroft-Ullman [6].)

We modify this standard notion to recognize left and right side dependants of a word - obligatory, facultative valencial dependants and free ones - starting from the most immediate neighbour.

Instead of recognizing words (or a word categories) we make these automata recognize functions, i.e. occurances of abstract relations between the postulated head and its left or right neighbour. Secondly, in addition to recognition we make the "arcs" build the structure determined by the observed function, e.g. attach the neighbour as a dependant, label it in agreement with the function and its interpretation etc.

An illustration

We approximate the syntactic function "object" of a finite verb with the two productions

(8)

```
RELATION:Object
        (RecognObj -> (D:=Object)(C:-D)(DELETE D))

RELATION:RecognObj
        ((R=+transitive -nominal)(D=+nominal -sentence)
          -> ((D=Partitive) -> (D=Plural);
                               ((D=Singular) -> (D=-countable);
                                                ((D=+countable)(R=<Negative
                                                                   +pobj>)))));
             ((D=Accusative)(R=Positive));
             ((D=Nominative)(R=Positive)
                            -> (D=Plural);
                               ((D=Singular)(R=Active
                                              <Indicative
                                               Conditional
                                               Potential
                                               (Imperative 3P)>)));
             ((D=Genetive Singular)(R=Positive
                                     <Passive
                                      (Active Imperative <1P 2P>)>)))
$
```

The relation "RecognObj" in the above production form is a kind of boolean expression of the morphological restrictions on a finite verb and its nominal object. The relation "Object", on the other hand, after a succesfull match of these conditions, labels the postulated dependant (D) as "object" and attaches it to the postulated regent (R).

The fragment

(9)

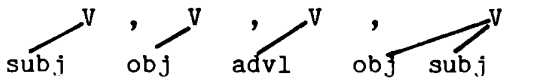
```
AUTOMATON:Verb

STATE:?V
      ((D=+phrase) -> (Subject -> (C:=?VS LEFT));
                      (Object -> (C:=?VO LEFT));
                      (Adverbial -> (C:=?V LEFT));
                      (GenSubj -> (C:=VS? RIGHT));
                      (SentAdvl -> (C:=?V RIGHT));
                      (T => (C:=V? RIGHT)));
      ((D=-phrase) -> (C:=V? RIGHT))

STATE:?VS
      ((D=+phrase) -> (Object -> (C:=?VSO LEFT));
                      (Adverbial -> (C:=?VS LEFT));
                      (SentAdvl -> (C:=VS? RIGHT));
                      (T -> (C:=VS? RIGHT)));
      ((D=-phrase) -> (C:=VS? RIGHT)(BuildPhraseOn RIGHT));
```

of a verb automaton recognizes and builds, for example, partial structures like

(10)



Parsing with a sequence of 2-way automata

So far we have shown how to associate a 2-way automaton to a word via its syntactic category. This gives a local description of the language. We argue that with a few simple control instructions we can make these local automata activate each other and actually parse an input sentence.

An unfinished parse consists of a sequence of constituents, which may be complete or incomplete. Further, each such constituent is associated with an automaton in some state and reading position. Now, the question is how to activate these automata. At any time, exactly one of the automata is to be active, i.e. trying to find dependants to the constituent head in the immediate neighbourhood.

Only a completed constituent (one featured as +phrase) is to be matched as a dependant. To start the completion of an uncomplete constituent the control has to be moved to this constituent. This is done with a push-like control operation "BuildPhraseOnRight" which deactivates the current automaton and activates the neighbour next to the right (see the illustration above).

On the other hand, a tree in a final state will be labelled as a +phrase (along with other relevant labels such as ±sentence, ±nominal, ±main etc.). Pop-like control operations "FindRegOnLeft" and FindRegOnRight" deactivate the current constituent (i.e. the corresponding automaton) and activate the leftmost or rightmost constituent, respectively.

We claim that such simple "local control" yields a strongly data driven bottom-up and left-to-right strategy which has also top-down features in the form of expectations on lacking dependants.

We use heuristic rules to reduce the number of alternative postulates in the course of parsing. For example, we might include the production

```
((R1 = ',')(R2 = 'että')(C = +trans +cogn) -> (C:=SV?Sent0 RIGHT)
                                               (BuildPhraseOnRIGHT))
```

in the state VS? of the verb automaton to recognize an evident forthcoming sentence object of a cognitive verb and to set the verb to the state SV?Sent0 to wait for this sentence.

Comparison

Our model, as we have shown, consists of a collection of finite transition networks which activate each other with pop- and push-like control operations. How does our approach then differ from, say, Woods' ATN-formalism, which seems to have similar characteristics?

One difference is the use of 2-way automata instead of 1-way automata. There are also other major differences. ATN-parsers seem to use pure constituent structures containing non-terminal auxiliary categories (VP, NP, AP...) without explicit use of dependency relations within a constituent. In our dependency oriented model non-terminal categories are not needed, and a constituent is not postulated until its head is found. In fact, each word collects actively its dependants to make up a consitutent where it is the head.

A further characteristic of our model is the late postulation of a function or a semantic role. Trees are built blindly without any predecided purpose. The function or semantic role of a constituent is not postulated until some earlier or forthcoming neighbour is activated to recognize dependants of its own. Thus, a constituent just waits to be chosen into some function.

The above feature explains why no registers are needed in our approach.

Conclusions

We have outlined a model of Finnish which is based on 2-way structure building transition networks. We have, as the above illustration exhibits, specified our model with a kind of production-rule formalism.

A compiler which compiles such descriptions into LISP is under construction. This LISP-code is further compiled into a directly executable code so that no interpretation of the productions or production packets of the grammar is necessary. That is, most of the linguistic knowledge is put into active form. We hope to get implementational results in early spring 1984.

References

[1] Anderson, J.: The Grammar of Case: Towards a Localistic Theory. Cambridge University Press, London & New York, 1971.

[2] Anderson, J.: On Case Grammar. Croom Helm, London 1977.

[3] Fillmore, C.: The case for a case. In Universals in Linguistic Theory (eds. Bach, E., and Harms, T.), Holt, Rinehart & Winston, New York 1968, 1–88.

[4] Fillmore, C.: Types of lexical information. In Semantics: An Interdisciplinary Reader (eds. Steinberg, D., and Jacobovitz, L.), Cambridge University Press, 1971, 109–137.

[5] Hays, D.: Dependency theory: A formalism and some observations. Language 40, 1964, 511–525.

[6] Hopcroft, J., and Ullman, J: Introduction to Automata Theory, Languages and Computation. Addison-Wesley, Reading, Mass., 1979.

[7] Hudson, R.: Arguments for a Non-transformational Grammar. The University of Chicago Press, 1976.

[8] Jäppinen, H., et al.: Morphological Analysis of Finnish: A Heuristic Approach. Helsinki Univ. of Tech., Digital Systems Laboratory, Report B26, 1983.

[9] Pajunen, A.: Suomen kielen verbien leksikaalinen kuvaus. Lisensiaattityö. Turun yliopiston suomalaisen ja yleisen kielitieteen laitos, 1982.

[10] Siro, P.: Sijakielioppi (2., korjattu painos). Oy Gaudeamus Ab, Helsinki, 1977.

[11] Tarvainen, K.: Dependenssikielioppi. Oy Gaudeamus Ab, Helsinki, 1977.

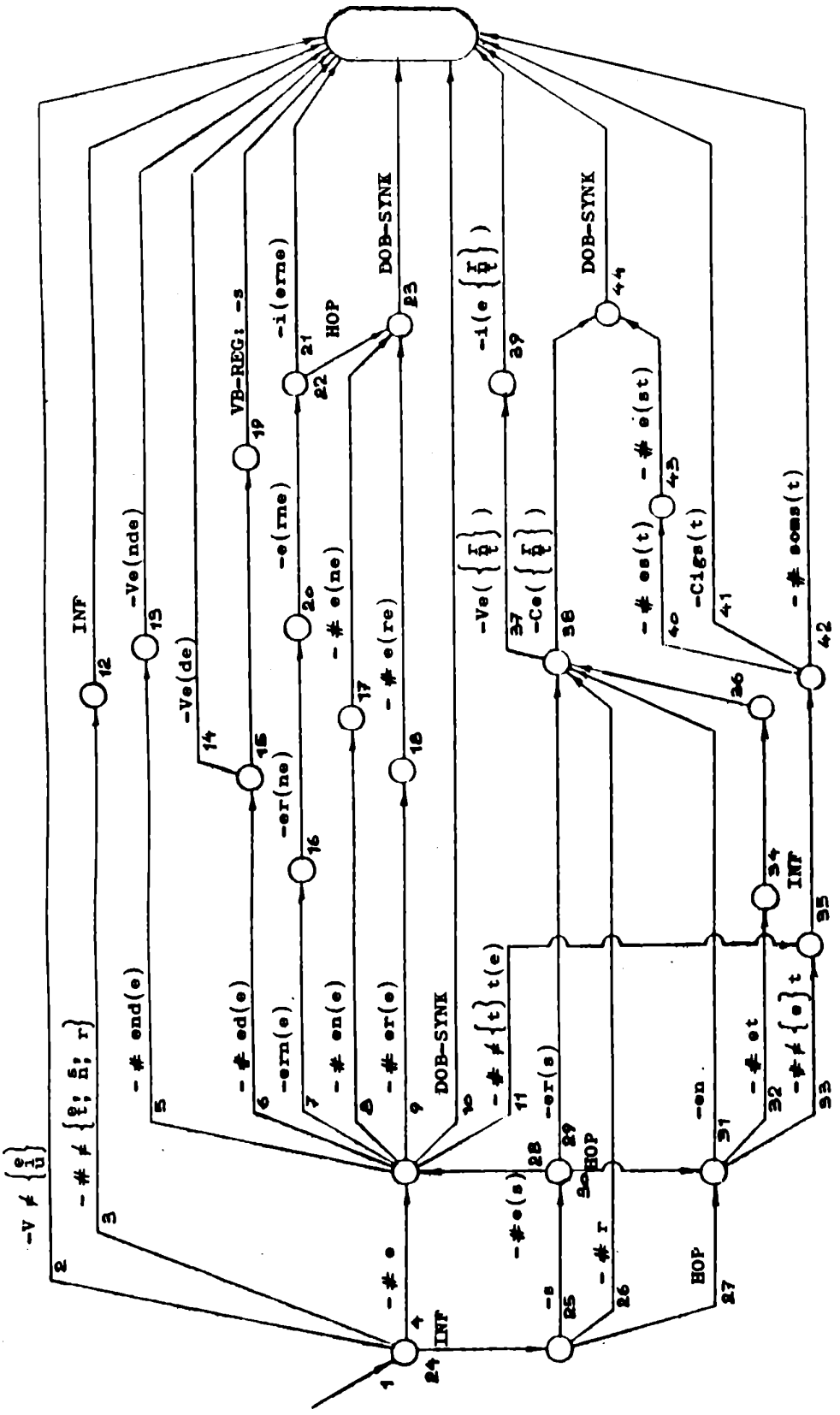
[12] Woods, W.: Transition network grammar for natural language analysis. Communications of the ACM 13, 1970, 591–606.

Hanne Ruus
Institut for nordisk filologi
Københavns Universitet
Njalsgade 80
DK-2300 København S.

DANSK MORFOLOGI TIL AUTOMATISK ANALYSE.

Den datamatiske analyse af naturlige sprog kan have de mest forskelligartede formål. Man kan have et sprogvidenskabeligt formål: Ved at formulere dele af et naturligt sprogs beskrivelse i grammatikker beregnet til at anvendes af en parser (jf. andre bidrag i denne publikation) og udvælge passende varierede data, kan man kontrollere, om den lingvistiske beskrivelse er udtømmende og sammenhængende. Man kan have et matematisk formål: Man kan afprøve, hvilken type formelt system og dertil svarende automat der egner sig til at beskrive en given del af et naturligt sprogs grammatik. Endvidere kan man have et formål, der er defineret af en bestemt anvendelse af den datamatiske analyse af naturlige sprog.

Når man udarbejder et stykke sprogbeskrivelse, vil anvendelsen af denne beskrivelse være medbestemmende for, hvordan beskrivelsen vil komme til at se ud; i øvrigt vil beskrivelsen selvfølgelig afhænge af det sprog og det fænomen, man beskriver. De nordiske sprog har en relativt kompliceret suffigerende fleksion, sammenlignet med andre vesteuropæiske sprog som engelsk, tysk og fransk. Alle fleksionskategorier kan udtrykkes med suffikser. Det betyder, at de store fleksionsordklasser substantiver, verber og adjektiver alle har ordformer med mere end ét suffigeret fleksiv. En maksimal substantivform er f.eks. pigernes, som er pige bøjet i numerus (pluralis: -r), i bekendthed (-ne) og i kasus (genitiv: -s). En maksimal verbalform er ventedes, der er bøjet i tempus (præteritum: -de) og i diatese (passiv: -s). En maksimal adjektivform vil være den yngstes bøjet i komparation (superlativ: -ste) og kasus (genitiv: -s). Før et eller flere fleksiver kan et ords stamme kræve forskellige morfografematiske ændringer som konsonantfordobling eller -forenkling og/eller synkope. Formen gamle er f.eks. stammen gammel ændret ved tilføjelse af fleksivet -e, synkope (tab af e'et foran l) og konsonantforenkling.



Figur 1. (fra Ruus 1977).

Den nordiske fleksion er også karakteristisk ved sit ringe inventar af endelser, som så til gengæld hver især har flere forskellige betydninger. I dansk bruges f.eks. -r både af substantiver og verber, -e af både substantiver, verber og adjektiver. I norrønt sprog bruges -a, -ar, -ir i både substantiv-, verbal- og adjektivbøjningen.

Figur 1. viser en udtømmende beskrivelse af den regelmæssige danske substantiv-, verbal- og adjektivbøjning. Man går ind i netværket til venstre. De bøjede ord undersøges bagfra. Når en del af en endelse passer med betingelsen på en pil, fjernes denne del og man fortsætter i den knude, som pilen fører til. DOB-SYNK refererer til et delnetværk, hvor de morfografematiske ændringer behandles (jf. Ruus 1977, Ruus 1978). Denne grammatik giver en udtømmende beskrivelse af de mulige endelser, deres indbyrdes kombinationsmuligheder og deres optræden sammen med morfografematiske ændringer. Der er ikke i denne grammatik taget stilling til, hvilken eller hvilke fortolkninger de enkelte endelser og endelseskombinationer skal have. Når man bruger grammatikken over dansk fleksion til at vise, at den er en udtømmende sproglig beskrivelse, vil man udfolde den enkelte endelses potentiale i overensstemmelse med almindelig sprogvidenskabelig praksis. I adjektivbøjningen vil -e f.eks. blive beskrevet som enten bestemt form singularis eller ubestemt form pluralis eller bestemt form pluralis. Hvis man skal bruge de morfologiske oplysninger i en praktisk anvendelse, kan det være hensigtsmæssigt at udskyde fortolkningen af en endelse til et senere trin i analysen.

Blandt de mange områder, hvor man arbejder med datamatisk analyse af naturlige sprog til bestemte praktiske formål, er maskinoversættelse et af de mest ambitiøse (Ruus 1984). I maskinoversættelsessystemer beregner man en syntaktisk-semantisk struktur for sætningerne i kildeteksten og bruger denne strukturs informationer til at finde de rigtige målsprogsækvivalenter (se f.eks. om Eurotra Maegaard og Ruus 1980, Ruus og Maegaard 1982). Hvordan man beregner den syntaktisk-semantiske struktur, afhænger af, hvilken lingvistisk strategi man vælger. For at kunne beregne den må man dog have nogle oplysninger om de sekvenser af tegn, som inddata består af. Her kan man inddrage den morfologiske analyse.

Man kunne spørge, om man ikke kan klare sig med en fuldformsordbog. I en sådan vil alle bøjede ordformer forekomme som selvstæn-

dige opslag ledsaget af de relevante oplysninger om bøjningsform og betydning. Et maskinoversættelsessystem er et generelt system. Det skal kunne genkende og behandle alle ord i det analyserede sprog. Til denne anvendelse vil det være umuligt for sprog som de nordiske at fremstille en komplet fuldformsordbog, da nye sammensatte og afledte ord dannes løbende.Selv med en større fuldformsordbog måtte man sørge for at have et system til at behandle ukendte ordformer bl.a. ved at fjerne bøjningsendelser. Når man skal have et sådant system, kan man lige så godt bruge det til alle ordformer. Så kan man nøjes med at lagre lemmaer i ordbogen. Hvert lemma skal så forsynes med oplysninger om bøjning, syntaktisk klasse og betydninger.

Man skal altså bruge en fleksionsanalyse. Hvilke oplysninger kan fleksionsanalysen give? Fleksiver kan levere to slags oplysninger: De fleksiver, der har en betydning, repræsenterer den del af lingvistisk semantik, der er bedst udforsket. Det gælder f.eks. numerus- og tempusfleksiver. De fleksiver, der ikke har betydning, kan ofte bruges som vejvisere i den syntaktiske analyse. Det gælder f.eks. genusfleksiver.

Blandt andet som følge af de multianvendelige endelser er de nordiske ord ofte flertydige. Med Allén (Allén 1970 p. XIX) kan man skelne mellem ekstern homografi, dvs. at en ordform kan tilhøre forskellige syntaktiske klasser f.eks. murer sb. og murer vb. i præsens, og intern homografi, dvs. at en ordform kan være forskellige former af samme lemma f.eks. ord, som kan være singularis eller pluralis. Selv ved en ganske enkel fleksivanalyse med ordbogsopslag vil man altså ofte få to og flere løsningsforslag til de enkelte ordformer i den tekst, der skal analyseres.

I det følgende skal jeg skitsere en metode til at behandle interne homografer, sådan at de får én og kun én beskrivelse af fleksionsanalysen og dernæst fortolkes i den syntaktiske analyse. Som eksempel vælges den danske adjektivbøjning i positiv i kombination med neutrale substantiver.

Ud fra bøjningen i positiv kan man udskille fire slags danske adjektiver:

	endelser	eksempel
type I	-Ø -t -e	sød
type II	-Ø -t	grå
type III	-Ø -e	sort frisk
type IV	-Ø	ringe

Som man ser, optræder Ø-endelsen ved alle fire typer adjektiver. Det giver i alle tilfælde anledning til intern homografi, men af forskellig art:

artikel	type I	type II	type III	type IV	substantiv
	sød-Ø	grå-Ø	sort-Ø	ringe-Ø	(fælleskøn)
	sød-t	grå-t	sort-Ø	ringe-Ø	øl, brød
et	sød-t	grå-t	sort-Ø	ringe-Ø	brød
det	sød-e	grå-Ø	sort-e	ringe-Ø	brød
	sød-e	grå-Ø	sort-e	ringe-Ø	brød
de	sød-e	grå-Ø	sort-e	ringe-Ø	brød

Man ser af ovenstående oversigt, at Ø-endelsen bruges i alle kontekster ved type-IV-adjektiver, ved type-II og type-III bruges Ø-endelsen, henholdsvis hvor type-I har Ø- og e-endelse og hvor type-I har Ø- og t-endelse. Det er derfor nødvendigt at operere med 4 forskellige Ø-endelser ved adjektiver i positiv. Hvis vi nummererer dem efter typerne ovenfor, kan de fire adjektiver have følgende oplysninger i ordbogen om deres bøjning i positiv:

sød	adj-Ø1, adj-t, adj-e
grå	adj-Ø2, adj-t
sort	adj-Ø3, adj-e
ringe	adj-Ø4

For de forskellige former af de fire adjektiver leverer fleksionsanalysen følgende oplysninger:

	inddata	analyseresultat
type I	sød*	sød+adj-Ø1
	søde*	sød+adj-e
	sødt	sød+adj-t

type II	grå	grå+adj-Ø2
	gråt	grå+adj-t
type III	sort*	sort+adj-Ø3
	sorte	sort+adj-e
type IV	ringe*	ringe+adj-Ø4

En analyse af pluralisbøjningen af neutrale substantiver vil vise, at man her har 2 forskellige Ø-endelser:

artikel	type I	type II	type III
det	brød-Ø	hus-Ø	æble-Ø
de	brød-Ø	hus-e	æble-r

Alle tre typer substantiver har Ø-endelse i singularis, type-I har også Ø-endelse i pluralis, hvor de andre typer har e- og r-endelse. En del af fleksionsoplysningerne ved disse ord i ordbogen kunne da se således ud:

brød	sbneu-Ø1
hus	sbneu-Ø2, sb-e
æble	sbneu-Ø2, sb-r

Fra fleksionsanalyse med ordbogsopslag får vi da følgende resultater for former af disse ord:

	inddata	analyseresultat
type I	brød*	brød+sbneu-Ø1
type II	hus*	hus+sbneu-Ø2
	huse*	hus+sb-e
type III	æble	æble+sbneu-Ø2
	æbler	æble+sb-r

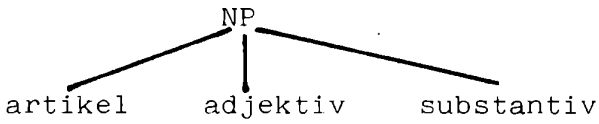
Man ser, at den enkelte ordform kun får et analyseresultat med den omhandlede ordklasse. * ved ordformerne angiver, at de har ekstern homografi og derfor også vil blive analyseret som former af andre ordklasser.

Det står nu tilbage at skitsere, hvordan den syntaktiske analyse kan fortolke fleksivoplysningerne, når den danner nominalhypotagmer. Vi antager, at den syntaktiske analyse bl.a. indeholder en grammatik, der bygger nominalhypotagmer, og at den er opdelt i undergrammatikker, der bygger forskellige slags nominalhypotagmer. Én undergrammatik bygger hypotagmer med neutrale ord i singularis som kerne

(1) en regel med betingelsen

$$\underline{et} + \text{adjektiv} + \begin{Bmatrix} \text{adj-t} \\ \text{adj-Ø3} \\ \text{adj-Ø4} \end{Bmatrix} + \text{substantiv} + \begin{Bmatrix} \text{sbneu-Ø1} \\ \text{sbneu-Ø2} \end{Bmatrix}$$

vil bygge et hypotagme:



I reglen vil man så også sørge for, at egenskaberne ubestemt og singularis bliver knyttet til hypotagmet.

(2) en regel med betingelsen

$$\underline{\text{det}} + \text{adjektiv} + \left\{\begin{array}{l}\text{adj-e}\\ \text{adj-Ø2}\\ \text{adj-Ø4}\end{array}\right\} + \text{substantiv} + \left\{\begin{array}{l}\text{sbneu-Ø1}\\ \text{sbneu-Ø2}\end{array}\right\}$$

vil bygge et hypotagme med samme struktur som regel (1), men her vil egenskaberne bestemt og singularis blive knyttet til hypotagmet.

Hvis man sammenholder de syntaktiske reglers betingelser med fleksionsanalyseresultaterne ovenfor, vil man se, at analysen af adjektivformerne sødt, gråt, sort, ringe passer med betingelsen for adjektivet i regel (1) og analysen af formerne søde, grå, sorte, ringe med betingelsen i regel (2). De flertydige former søde, grå, sorte, ringe får således en rigtig tolkning, uden at det har været nødvendigt at søge mellem alternative tolkningsforslag. Det samme gælder for den flertydige substantivform brød, der her bliver tolket som singularis, fordi den står i en singulariskontekst.

Da genus ikke markeres i pluralis på dansk, vil reglerne for konstruktion af nominalhypotagmer i pluralis skulle behandle både neutrale substantiver og fælleskønssubstantiver. Grammatikregler for hypotagmer i pluralis kan derfor først formuleres, når bøjningen af fælleskønssubstantiver er analyseret for flertydige endelser.

Den her skitserede behandling af interne homografer skulle da have følgende fordele: der gives kun ét fleksionsanalyseresultat for de forskellige tolkningsmuligheder af en intern homograf. Den rigtige tolkning vælges i den syntaktiske analyse, hvor man bygger på konteksten. I de sjældne tilfælde, hvor den nærmeste kontekst er flertydig på samme måde for både substantiv

og adjektiv f.eks. ringe brød med analysen ringe+adj-Ø4+brød+ sbneu-Ø1, vil analysen selvfølgelig passe både til betingelsen på en regel, der danner nøgne nominalhypotagmer i singularis, og på én, der danner nøgne nominalhypotagmer i pluralis. Hvis man i en given tekst skal vælge den rigtige af disse to løsninger, må man inddrage en større kontekst.

Ved behandling af store tekstmængder vil det være en tidsmæssig fordel, at interne homografer kun får én analyse i fleksionsanalysen, der skal gennemløbes for alle sætninger. De omtalte interne homografier hører nemlig ligesom de interne homografier i verbalbøjningen til blandt de almindeligste bøjningsformer (Noesgaard 1960).

Den beskrevne strategi kan muligvis også anvendes ved ekstern homografi som den forekommer ved mange funktionsord f.eks. den, det, de artikler eller pronominer. Hvis disse ords hele betydning fremgår af deres syntaktiske placering, kan de optræde eksplicit i de syntaktiske regler som eksemplificeret ovenfor og vil derfor ikke kræve ordbogsoplysninger og morfologisk analyse. I et maskinoversættelsessystem, der jo skal analysere alle slags tekster og mange af dem, vil det være en tidsmæssig gevinst, hvis sådanne ord kan nøjes med en meget simpel eller ingen indledende analyse. Disse ord hører til blandt de allerhyppigste ord, hvor en beregning har vist, at de 40 hyppigste homografer udgør 31% af en tekst (Maegaard og Ruus 1980b).Om den betydningsbeskrivelse, der kan deduceres i analysen, er tilstrækkelig for disse ord til maskinel oversættelse, vil vise sig, når man kommer til praktiske eksperimenter med at producere oversættelser til og fra dansk, forhåbentlig i løbet af det næste par år.

Litteraturhenvisninger.

Allén, Sture 1970: Nusvensk Frekvensordbok bd.I
Maegaard, Bente and Ruus, Hanne 1980: Structuring Linguistic Information for Machine Translation, The EUROTRA Interface Structure, i Human Translation, Machine Translation ed. Suzanne Hanon and Viggo Hjørnager Pedersen = NOK 39, Romansk Institut, Odense Universitet, p. 159-169.
Maegaard, Bente og Hanne Ruus: Danske almindelige ord. Rangfrekvenslister og deres brug, i SAML 7, p. 5-22.
Noesgaard, A. 1960: Hyppighedsundersøgelser III.

Ruus, Hanne 1977: Ordmekanik, i SAML III, p.79-106.
Ruus, Hanne 1978: Suffix-stripping. Automatisk lemmatisering af regelmæssigt bøjede danske ord, i Selskab for nordisk filologi, Årsberetning for 1974-76, p.14-21.
Ruus, Hanne and Maegaard, Bente: Multilingual Syntax and Morphology for Machine Translation, i Machine Translation and Computational Lexicography,ed. Karl Hyldgaard-Jensen and Bente Maegaard, p. 26-34.
Ruus, Hanne 1984: Investigating the Nordic Languages for the Information Society, udkommer i Proceedings from V. International Conference of Nordic Languages and Modern Linguistics, Århus.

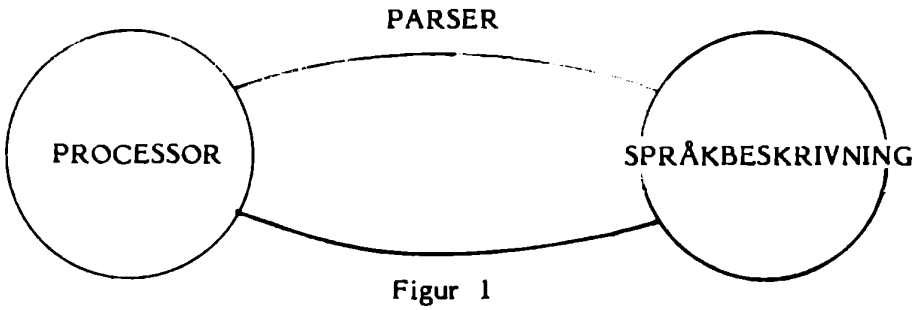
Øvrige bidrag om parsing i nærværende publikation.

Forkortelse:
SAML = Skrifter om Anvendt og Matematisk Lingvistik, udgivet af Institut for anvendt og matematisk lingvistik, Københavns Universitet.

Anna Sågvall Hein
Centrum för datorlingvistik
Sturegatan 13 B, 5 tr
752 23 UPPSALA

REGELAKTIVERING I EN PARSER FÖR SVENSKA (SVE.UCP)

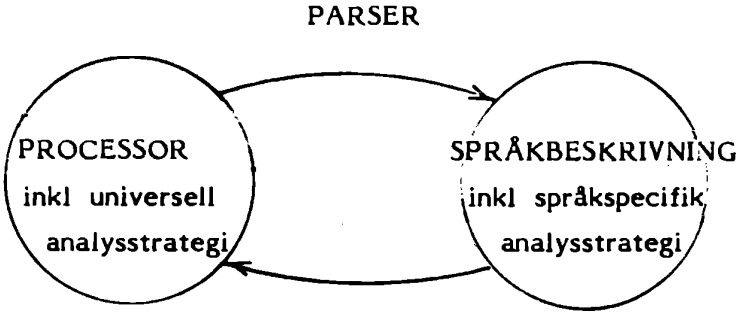
1. Inledning. Parsing är den process i vilken ett språkligt uttryck (ord, fras, sats, mening) tilldelas en lingvistisk beskrivning enligt en given språkbeskrivning. För det processerande maskineri, som sätter språkbeskrivningen i arbete i relation till det uttryck som skall analyseras, analysuttrycket, använder jag termen 'processor'. Termen introducerades i datorlingvistiska sammanhang av R Kaplan i hans artikel 'A General Syntactic Processor' (Kaplan 73). Utformningen av processorn är ur datorlingvistiskt perspektiv inte mindre intressant än den av språkbeskrivningen.



Figur 1

GSP inkluderar ett format för att representera analysuttrycket i dess olika bearbetningsstadier, dvs dess initiala form såväl som intermediära och finala resultat, 'the chart', och ett antal processprimitiver för uppbyggande av en nätverksgrammatik. Analysen förlöper som en följd av enkla bearbetningssteg, 'tasks'. Administrationen av de olika bearbetningsstegen handhas av en agenda. Vidare innefattar GSP en styrande algoritm, samt ytterligare ett antal kontrollstrukturer i algoritmens tjänst. I en av Kay föreslagen vidareutveckling av GSP (Kay 75), omvandlas charten från en passiv datastruktur för lagring av lingvistiska strukturer till en aktiv struktur, vilken övertar många

av de kontrollfunktioner som i den tidigare versionen vilade på särskilda hjälpstrukturer. Uppsala Chart Processor, som utvecklats och implementerats vid Centrum för datorlingvistik (se Sågvall Hein 80, 81, 82 och 84, under utgivning, samt Carlsson 81), bygger vidare på GSP-traditionen, med utnyttjande av iden om 'den aktiva charten'. Försedd med en språkbeskrivning för ett givet språk, fungerar UCP som en parser för språket i fråga. Sålunda förfogar vi redan över en parser för svenska, SVE.UCP, med begränsad kompetens (se Sågvall Hein 83). Pågående arbete koncentreras till en systematisk utbyggnad av den svenska kompetensen. Från sina föregångare skiljer sig UCP främst därigenom, att den utför både morfologisk och syntaktisk analys. Den morfologiska analysen förlöper i analogi med den syntaktiska. Såväl lexikonsökning som regeltillämpning (fonologisk, morfologisk och syntaktisk) realiseras som en följd av konceptuellt homogena bearbetningssteg. Att kunna integrera den morfologiska analysen med den syntaktiska är en förutsättning för korrekt ordigenkänning i de fall de lexikaliska enheterna är diskontinuerligt presenterade i texten. Ett annat av de utmärkande dragen för en UCP-parser har att göra med förhållandet mellan processorn och språkbeskrivningen. Sålunda inkluderar den särspråkliga kompetensen inte enbart de lingvistiska enheter och regler, som beskriver språket i fråga, utan även regelaktiverande mekanismer. Ansatsen bygger på det antagandet, att man kan skilja mellan en universell och en språkspecifik analysstrategi. Den universella strategin ligger inom processorns domän, och den språkspecifika inom den särspråkliga kompetensens område (fig 2).



Figur 2

När det gäller frågan hur analysstrategin skall inordnas i språkbeskrivningen har jag valt att integrera den i de enskilda lingvistiska reglerna som det alternativ som erbjuder störst flexibilitet och därmed också den bästa experimentmiljön.[1] Sålunda följer vi i UCP-formalismen upp den ursprungliga GSP-tanken på en procedural lingvistisk formalism. Språkbeskrivningen i en UCP-parser för-

delar sig på tre typer av konstrukter, nämligen grammatik, lexika och baser. Innehållet i de olika konstrukterna uttrycks i en och samma formalism, dvs den procedurala UCP-formalismen. UCP-formalismen är sålunda generell. Konstrukterna skiljer sig åt med avseende på organisationsform och accessmetod.

I sin nuvarande utformning inkluderar SVE.UCP: en grammatik innehållande morfologiska och syntaktiska regler (SVE.GRAM), ett huvudlexikon över rötter, stammar och oböjliga ord (SVE.DIC), ett antal affixlexikon och ett lexikon över de skiljetecken vilka betraktas som särskilda ord (SEP). Vidare omfattar den svenska språkbeskrivningen en lexembas och en karaktärbas. I lexembasen finns semantisk information avseende substantiv och verb. Karaktärbasen reflekterar en klassificering av karaktärerna i bokstäver, siffror och skiljetecken. Via särdragsmarkering anges för bokstäverna huruvida de svarar mot vokal eller konsonant. Dessutom finns ett antal fonetiska särdrag upptagna.

En av de stora fördelarna med en generell lingvistisk formalism, och en procedural sådan, är den att den underlättar samspelet mellan de olika lingvistiska kunskapskällorna under parsingprocessen. I SVE.UCP har vi sålunda utformat en analysgång för svenska meningar, som grovt ser ut som följer. Meningen presenteras för SVE.UCP som en följd av karaktärer exakt som den står i texten, dvs med skiljetecken och stora och små bokstäver. Analysen inleds med att UCP anropar begynnelseregeln i grammatiken. Dess effekt är att initiera två olika processer, nämligen sökning i huvudlexikonet, resp tillämpning av huvudsatsregeln. Lexikonsökningen, som avser karaktärer, igångsätts omedelbart, medan huvudsatsregeln måste invänta igenkänning av den typ av lingvistiska enheter, för vilken den är definierad, dvs ord och fraser. Grammatiska beskrivningar av dessa presenteras som resultat av det analysarbete som lexikonsökningen ger upphov till. Då t ex en substantivisk stam återfunnits i lexikonet anropas sålunda bl a den morfologiska regel som känner igen substantiv och bygger upp motsvarande morfologiska beskrivning. (Vidare initieras sökning i de substantiviska affixlexikonen efter påföljande affix.) Substantivregeln i sin tur anropar en NP-regel i grammatiken, som känner igen och bygger upp beskrivningar av nominalfraser. Därigenom blir lämpligt arbetsmaterial tillgängligt för huvudsatsregeln, som kan sätta i gång sitt arbete.

Föreliggande presentation syftar inte till att beskriva enskildheterna i samspelet mellan grammatik, lexika och baser i vår svenska parser, utan till att visa på de faciliteter som UCP-formalismen erbjuder för att uttrycka ett sådant. Helt kort kan dock konstateras att den analysstrategi som vi byggt in i den

svenska kompetensen i vår parser, ligger i linje med den perceptuella strategi som Kimball förespråkar (Kimball 73).

2. UCP-formalismen. UCP-formalismen inkluderar, förutom ett format för att uttrycka lingvistiska regler, också ett format för att uttrycka grammatiska beskrivningar av lingvistiska enheter. De uttrycks som hierarkiska mängder av attribut-värde-par (Sågvall Hein 83 och Ahrenberg, denna volym). Grammatiken upptar ett antal regler, vilka var och en består av ett unikt namn, en regelkropp, en domänspecifikation och, optionellt, angivande av ett kontextuellt villkor för dess tillämpning (fig 3).

```
Regelnamn
    Regelkropp
    Domän
    (Villkor för dess tillämpning)
```

Figur 3

Regelkroppen är uppbyggd som en följd av enkla operationer, vilka exekveras i sekvens, till dess någon operation misslyckas (returnerar NIL), varvid exekveringen avbryts. Operationerna utförs i relation till analysuttrycket i dess chartrepresentation. I domänen anges på vilken typ av lingvistiska enheter regeln är tillämplig. Som första exempel på en grammatisk regel anförs den regel med vilken analysen av en svensk mening inleds (fig 4).

```
START.RULE
        PROCESS(SVE.DIC),
        PROCESS(MAIN);
          Domain: CHAR;
```

Figur 4

Regelns namn är START.RULE. Regelkroppen upptar två operationer, nämligen PROCESS(SVE.DIC) och PROCESS(MAIN). I det ögonblick startregeln anropas föreligger analysuttrycket som en följd av karaktärer, varför regelns domän specificeras till karaktärer (CHAR). SVE.DIC är namnet på det svenska huvudlexikonet. Det upptar rötter, stammar och ordformer. Exekvering av operationen PROCESS (SVE.DIC) har som effekt att sökning i huvudlexikonet initieras. MAIN är namnet

på huvudsatsregeln i grammatiken. Via operationen (MAIN) anropas denna regel. Operationen uttrycker sålunda en prediktion om en huvudsats. Startregeln anropas av UCP. Övriga regelanrop sker via operationer i grammatik och lexika. Fortsättningsvis skall jag koncentrera mig på vilka möjligheter som finns för detta. En förenklad version av vår svenska huvudsatsregel (fig 5) får fungera som illustration.

2.1 Regelaktivering i den svenska huvudsatsregeln. Som framgår av domänspecifikationen i fig 5, opererar huvudsatsregeln på fraser (PHRase.CAT) och ord (WORD.CAT). Huvudsatsens regelkropp består till stor del av s k subregelanrop. V.FIN, ADVERBIALS1, SUBJECT, ADVERBIALS2, FUNDCHECK och END.OF.SENT är mnemoniska namn på grammatiska lägen, subregler, i vilka faktiska operationer specificeras. Så snart processorn under exekveringen av en regel träffar på ett subregelnamn, så utförs de operationer som upptas under detta regelnamn. Först i huvudsatsregeln ligger dock ett antal operationer, som inleder uppbyggandet av en grammatisk beskrivning av satsen. Bland dessa finns en, som ansätter den första konstituenten i satsen som fundament.[2]

```
MAIN
  "ansätt aktuell konstituent som fundament,
   ... ",
  ADVANCE,
  V.FIN,
  ADVANCE,
  ADVERBIALS1,
  SUBJECT,
  ADVERBIALS1,
  DO(<& PRED VFIN VERB.FEAT LEX :VERBACTION>),
  ADVERBIALS2,
  FUNDCHECK,
  (END.OF.SENT, MINORSTORE, MAJORPROCESS(SENTENCE)/
   "test på konjunktion och el skiljetecken",
   MAJORPROCESS(SENTENCE.COORD), MINORSTORE, ADVANCE,
   PROCESS(MAIN));
    Domain: PHR.CAT / WORD.CAT;
```

Figur 5

Efter utförandet av de initiala ansättningsoperationerna flyttas processorns

uppmärksamhet ett steg framåt i satsen, dvs till slutet av den första konstituenten (fundamentet). Detta sker med hjälp av operatorn ADVANCE. Subregeln V.FIN predicerar ett finit verb i detta läge och tilldelar ev återfunnet sådant till predikatet, varpå ett avancemang sker till subregeln ADVERBIALS1.[3] ADVERBIALS1 beaktar optionella satsadverbial och gör motsvarande tilldelningar till den grammatiska beskrivningen. SUBJECT (fig 6) illustrerar möjligheten till parallell analys.

```
SUBJECT
     ("aktuell konstituent är en NP",
      "kasustest",
      "ansätt aktuell konstituent som subjekt",
      ADVANCE //
      "fundamentet är en NP",
      "kasustest",
      "ansätt fundamentet som subjekt");
```

Figur 6

När subjektsregeln tillämpas, har processorn sin uppmärksamhet riktad på den första konstituenten efter det finita verbet, såvida denna inte tolkats som ett satsadverbial, i vilket fall processorn fokuserar på den första konstituenten efter detta satsadverbial. Subjektsregeln upptar två av varandra oberoende parallella alternativ, åtskilda av '//'. I det första alternativet inspekteras aktuell konstituent, och i det andra fundamentet.[4] Om aktuell konstituent är en NP i grundformskasus, så ansätts (beskrivningen av) den som subjekt, och ett avancemang sker till nästa konstituent. I annat fall avbryts exekveringen av denna analysgren. Oberoende av dess utfall, exekveras den alternativa analysgrenen enl följande. Om fundamentet är en NP i grundformskasus, så ansätts den som subjekt. I annat fall avbryts exekveringen. Om vi sålunda har en sats av typen 'NP V NP', så förgrenar sig analysen via subjektsregeln i två av varandra oberoende alternativ, med olika värden på subjektsattributet. Vi talar här om parallella processer och om operationen 'independent disjunktion' ('//'). Båda processerna löper samman i sökandet efter ytterligare optionella satsadverbial (ADVERBIALS1). Därpå följer DO-operationen, som utifrån syntaktiska och semantiska egenskaper hos det finita verbet svarar för igenkänning av övriga grammatiska funktioner på satsnivå.

Med hjälp av DO-operatorn kan vi utföra s k variabelt subregelanrop. Då en

DO-operation exekveras, evalueras först dess argument till ett subregelnamn, varpå denna subregel i sin tur exekveras. Som Do-operationens argument är utformat i huvudsatsregeln, evalueras det till namn på vad jag kallar en 'verbaktionsregel', i vilken det finita verbets argumentstruktur definieras. DO-argumentet har den i UCP-formalismen gängse formen av ett s k path-uttryck, '<...>', (se Kay 77). Om vi bortser från dess sista med kolon (':') prefigerade element, så uttrycker det en referens till det finita verbets lexembeteckning i den grammatiska beskrivning av satsen som är under uppbyggnad. Om det finita verbet sålunda är 'bor', så returnerar den första delen av path-uttrycket lexembeteckningen '!bo'. Det med kolon prefigerade elementet (':verbaction') tolkas av processorn som en anmodan att söka efter värdet på egenskapen 'verbaction' under lexembeteckningen '!bo' i lexembasen. För exemplet i fråga returnerar processorn 'type.bo', som är namn på aktuell verbaktionsregel. Med hjälp av kolon-faciliteten kan vi sålunda signalera till processorn, att den skall hämta information ur någon av baserna. (Lexembasen inkluderar f n förutom verb med associerade verbaktionsregler även substantiv med associerade semantiska särdrag.) Fig 7 visar verbaktionsregeln för verb av typen 'bo'.

```
TYPE.BO
    PREDSUBJ('ANIM),
    ADVLOC;
```

Figur 7

Regel upptar två subregelanrop, varav det första illustrerar hur en subregel kan anropas med ett argument, s k subregelanrop med variabel. PREDSUBJ ställer krav på subjektet; i detta fall krävs att subjektet är animat. ADVLOC kräver ett lokationsadverbial och gör motsvarande ansättning till den grammatiska beskrivningen av satsen.

Låt oss gå återvända till huvudsatsregeln (fig 5). ADVERBIALS2 känner igen efterställda satsadverbial och FUNDCHECK kontrollerar att fundamentet fått en grammatisk tolkning. Därpå uttrycker huvudsatsregeln två alternativ, åtskilda av '/', ett uttryck för operationen 'dependent disjunktion' (jfr OR). Enligt det första alternativet är satsen (och meningen) slut, i vilket fall huvudsatskonstituenten skall avslutas. Detta sker via operationen MINORSTORE. Den fungerar på så sätt, att den avslutar, stänger, konstituenten före det element som processorn just inspekterar, i detta fall det skiljetecken som markerar slut på meningen. Den låter oss sålunda se ett element framåt, in-

nan vi bestämmer oss för att avsluta en konstituent. Då vi känt igen en huvudsats och påföljande stora skiljetecken, vet vi också att vi har att göra med en mening. I regeln SENTENCE byggs den grammatiska beskrivningen av meningen upp, i vilken beskrivningen av huvudsatsen ingår som en del.

Anropet till SENTENCE sker via operatorn MAJORPROCESS. Den har som effekt, att den regel vars namn står som dess argument (SENTENCE), initieras från samma punkt i analysuttrycket som den regel från vilken anropet görs (MAIN). Sålunda kommer sentence-regeln att tillämpas från meningens början. Därigenom skiljer den sig från den andra regelaktiverande operatorn PROCESS, som initierar en process från den punkt på vilken processorn fokuserar i det ögonblick anropet görs. Med andra ord, så svarar MAJORPROCESS för regelaktivering bottom-up och PROCESS för regelaktivering top-down.

Enligt huvudsatsregelns återstående alternativ är huvudsatsen men inte meningen slut. COORD.TEST söker på små skiljetecken och konjunktioner. Om testet ger ett negativt utfall, så avbryts exekveringen av regeln. I annat fall sker följande: Beskrivningen av huvudsatsen avslutas. Regeln för koordinerade meningar (SENTENCE.COORD) anropas bottom-up och huvudsatsregeln anropas top-down. Som framgår av det sistnämnda anropet tillåter formalismen, att en regel anropas rekursivt.

Huvudsatsregeln arbetar på fraser och ord. Igenkänning av dessa och uppbyggande av motsvarande grammatiska beskrivningar sker som en följd av den sökning i huvudlexikonet som startregeln initierar (fig 4). Låt oss titta närmare på hur detta realiseras.

2.2 Lexikonsökning och igenkänning av ord och fraser. Uppslagsorden i lexikonen i en UCP-parser ligger lagrade som följder av teckenträd, och sökningen går stegvis, tecken för tecken. Varje sådant söksteg är analogt med tillämpning av en grammatisk regel. En sökregel inkluderar en obligatorisk operation, nämligen jämförelse mellan aktuellt tecken i analysuttrycket och i lexikonträdet. Operationen i fråga betraktas som universellt giltig, varför den är inbyggd i processorn. Däremot får vi för varje enskild applikation bestämma vilken typ av tecken sökningen avser, dvs definiera sökregelns domän. Vidare kan vi specificera ytterligare operationer för de fall jämförelsen utfaller positivt. Denna information ges i ett antal meta-uppslagsord för varje lexikon. Fig 8 visar metauppslagsorden i SVE.DIC.

```
DOMAIN
   CHAR;

GATHERING-RULE
   "tilldelning av aktuell karaktär till attributet CHARS";

FORWARD-ACTION
   ADVANCE;

...
```

Figur 8

Domänen specificeras till karaktärer (bokstäver, skiljetecken, specialtecken, mellanslag och siffror). (Som alternativ kan man tänka sig att sökningen skall utföras på en fonematisk representation av analysuttrycket.) Under GATHERING-RULE anger vi, att uppgift om vilken karaktär som jämförts skall tilldelas till den grammatiska beskrivningen. Det är med andra ord ett sätt att spara uppgift om den initiala strängen. Under FORWARD-ACTION flyttar vi processorns uppmärksamhet ett steg (i detta fall karaktär) framåt. Då denna framflyttning sker oberoende av om jämförelseoperationen gäller den sista karaktären i ett uppslagsord eller inte, blir sökningen icke-deterministisk och alla i lexikonet förutsedda segmenteringar beaktas.

Då sökprocessen lett fram till ett uppslagsord, dvs vid positivt utfall av jämförelsen mellan en karaktär i analysuttrycket och uppslagsordets sista karaktär, ansluts de operationer som bildar uppslagsordets regelkropp till operationen under GATHERING-RULE. Fig 9 visar exempel på en lexikonartikel.

```
EVA
   "tilldelning av värdet NOUNSTEM till attributet MORPH.CAT",
   "tilldelning av värdet !EVA till attributet LEX",
   PATTERN.FIRST.NAME;
```

Figur 9

Regelkroppen består av tre operationer, dvs ansättning av morfologisk kategori (NOUNSTEM), ansättning av lexembeteckning (!EVA) och ett subregelanrop (PATTERN.FIRST.NAME).

```
PATTERN.FIRST.NAME
    SPECIFY.SEX,
    "tilldelning av värdet + till attributet PROPR",
    "tilldelning av värdet + till attributet FIRST (förnamn)",
    "tilldelning av värdet SING till attributet NUMB",
    "tilldelning av värdet UTR till attributet GENDER",
    STORE.PROPR;
```

Figur 10

Anrop till PATTERN.FIRST.NAME (fig 10) görs från alla förnamn i lexikonet. Utgående från sista bokstaven i stammen och med utnyttjande av de uppgifter om karaktärerna som finns lagrade i karaktärbasen, fastställs via SPECIFY.SEX naturligt genus.[5] Sist återfinns ett anrop till subregeln STORE.PROPRium (fig 11).

```
STORE.PROPR
    MAJORPROCESS(NOUN),
    ADVANCE,
    (SPACE.OR.PUNCT / MINORSTORE, PROCESS(CASE));
```

Figur 11

STORE.PROPR inleds med ett anrop till substantivregeln, som känner igen substantivet och bygger upp dess morfologiska beskrivning. Från substantivregeln anropas vidare den NP-regel, som känner igen nominalfrasen och bygger upp dess syntaktiska beskrivning. STORE.PROPR avslutas med en disjunktiv (alternativ) operation. Det första alternativet, SPACE.OR.PUNCT, avser det fall då stammen omedelbart följs av en separator (ett mellanslag eller ett skiljetecken). Härvid avslutas beskrivningen av stamkonstituenten på erforderligt sätt, dvs inklusive mellanslaget och exklusive skiljetecknet. Därpå initieras sökning i huvudlexikonet (efter nästa ord), respektive i separatorlexikonet (efter aktuellt skiljetecken). I annat fall avslutas beskrivningen av stamkonstituenten via MINORSTORE, och sökning i kasuslexikonet igångsätts.

2.3 Sammanfattning av de processorienterade faciliteterna i UCP-formalismen. Sammanfattningsvis erbjuder UCP-formalismen följande möjligheter vad det gäller att uttrycka en språkspecifik analysstrategi:

- möjlighet att initiera sökning i visst lexikon från en regel i grammatiken eller från en lexikonartikel (PROCESS dic.name),

- möjlighet att anropa en regel i grammatiken top-down från en annan grammatisk regel eller från en lexikonartikel (PROCESS rule.name),

- möjlighet att anropa en regel i grammatiken bottom-up från en annan grammatisk regel eller från en lexikonartikel (MAJORPROCESS rule.name),

- möjlighet till enkel subregelaktivering från en grammatisk regel eller från en lexikonartikel (t ex SUBJECT, PATTERN.FIRST.NAME),

- möjlighet till variabel subregelaktivering från en grammatisk regel eller från en lexikonartikel (DO(<...>)), samt

- möjlighet till subregelaktivering med variabel från en grammatisk regel eller från en lexikonartikel (t ex (PREDSUBJ 'ANIM)).

3. Slutsats. Den informella presentationen av UCP-formalismen ovan, utifrån dess tillämpning i vår svenska parser, har inriktats mot att visa formalismens processorienterade faciliteter.[6] Genom förekomsten av denna typ av faciliteter skiljer sig en procedural formalism från en deklarativ. Det är genom tillgång till dessa, som språkvetaren i utarbetandet av en språkbeskrivning för ett givet språk har möjlighet att uttrycka en språkspecifik analysstrategi. UCP-formalismen erbjuder stor frihet i den konkreta utformningen av språkbeskrivningen, vad gäller dess strukturering i grammatik, lexika och baser samt vid utformningen av de enskilda reglerna i grammatik och lexika. I kraft av sin processorienterade utformning öppnar den sig mot interaktion med andra komponenter i en språkförståelsemodell.

NOTER.

(1) Ett annat alternativ vore att utforma analysstrategin som en egen komponent i språkbeskrivningen. Ett sådant företag bedöms dock som prematurt, då det bör vila på fördjupade insikter om olika analysstrategier. Vi räknar med att vi med vår ansats kommer att kunna experimentera oss fram till resultat

av värde i det sammanhanget.

(2) För att inte belasta framställningen i onödan med de tekniska enskildheterna i UCP-formalismen, har jag valt att uttrycka ansättnings- och testoperationer informellt. Det informella uttryckssättet markeras via anföringstecken.

(3) Om inget finit verb återfinns, så avbryts exekveringen av detta alternativ; en förklaring till misslyckandet kan vara att endast en delkonstituent av det egentliga fundamentet uppfattats som sådant; UCP svarar för att alla tänkbara alternativ, som grammatiken upptar beaktas.

(4) Trots att processorn i detta analysmoment fokuserar på konstituenten efter det finita verbet (ev föregången av optionellt satsadverbial), så är fundamentet tillgängligt för inspektion i och med att det ansatts i den grammatiska beskrivning av satsen som är under uppbyggnad. - Aktuell grammatisk beskrivning under uppbyggnad är ständigt tillgänglig under analysen i form av en aktiv båge i charten.

(5) För förnamn, som bryter mot de allmänna reglerna, anges genus individuellt i lexikonartikeln.

(6) De av formalismens operatorer som svarar mot tilldelning, tester etc har avsiktligt förbigåtts. För en informell presentation av UCP-formalismen i sin helhet hänvisas till Sågvall Hein 84, under utgivning, och för en teknisk presentation till Carlsson 81.

REFERENSER

Ahrenberg, L (1984), De grammatiska beskrivningarna i SVE.UCP (denna volym)

Carlsson, M (1981), Uppsala Chart Parser 2. System Documentation. UCDL-R-81-1. Center for Computational Linguistics. Uppsala University.

Kaplan, R M (1973), A General Syntactic Processor in Rustin, ed, Natural Language Processing. Englewood Cliffs, N J: Prentice Hall.

Kay, M (1975), Syntactic Processing and the Functional Sentence Perspective. I: Schank, R & Nash-Webber, B L (utg), Theoretical Issues in Natural Language Processing, Cambridge, Mass

--- (1977), Reversible Grammar. Summary of the Formalism. Xerox Research Center. Palo Alto.

Kimball, J (1973), Seven Principles of Surface Structure Parsing in Natural Language. Cognition, vol 2, s 15-47

Sågvall Hein, A (1980), An Overview of the Uppsala Chart Parser Version 1 (UCP-1). UCDL-R-80-1. Center for Computational Linguistics. Uppsala University.

--- (1981), Uppsala Chart Parser, Version 2 (UCP-2) - En översikt. I: Lien, E (red), De Nordiske Datalingvistikkdagene 1981. Foredrag fra en konferanse på Universitetssenteret på Dragvoll 22-23 oktober 1981.

--- (1982), An Experimental Parser. I: Proceedings of COLING 1982. Prag s 121-126

--- (1983), A Parser for Swedish. Status Report for SVE.UCP, February 1983. UCDL-R-83-1. Centrum för datorlingvistik. Uppsala Universitet.

--- (1984), Parsing by means of Uppsala Chart Processor. I: Bolc, L (ed), Natural Language Parsing Systems. Springer-Verlag. (Planerad utgivning 1984).