PAPERS FROM THE FIFTH SCANDINAVIAN CONFERENCE OF COMPUTATIONAL LINGUISTICS

Helsinki, December 11—12, 1985

Edited by Fred KARLSSON

1986

CONTENTS

Lars Ahrenberg
Institutionen för datavetenskap
Universitetet i Linköping
S-581 83 LINKÖPING

LEXIKALISK-FUNKTIONELL GRAMMATIK PÅ SVENSKA

1. Inledning: LFG och GWB.

LFG (Lexical-Functional Grammar, här försvenskat till Lexikalisk-funktionell grammatik) är en teori för den universella grammatiken i Chomskys mening, dvs en teori för den mänskliga språkförmågan, betraktad som en förmåga att associera språkliga uttryck med språkliga betydelser. LFG lägger stor vikt vid att den grammatiska teorin ska kunna inordnas i en realistisk modell för språkinlärning, språkförståelse och språkproduktion och antar att det kan ske på rakast möjliga sätt, dvs så att grammatiken ingår som en komponent i performansmodeller.

Det som främst skiljer LFG från traditionell transformationsgrammatik är följande:

(1) Grammatiska funktioner av typen subjekt, objekt, komplement betraktas som primitiva begrepp, som inte generellt kan definieras i termer av frasstrukturkonfigurationer;

(2) Inga syntaktiska regler tillåts förändra grammatisk funktion hos en konstituent. Detta utesluter många av de klassiska transformationerna i TG, t ex passivtransformationen. Istället finns den information som behövs för att tolka konstituenterna i en sats som uttryck för semantiska argument angiven explicit i lexikon. De regelmässiga variationerna i inkodningen av argument beskrivs med redundansregler i lexikon, som är formulerade i termer av grammatiska funktioner;

(3) Transformationer används överhuvudtaget inte och därmed inte heller distinktionen djupstruktur/ytstruktur. Den syntaktiska beskrivningen har ändå två klart åtskilda nivåer: *konstituentstruktur* (förkortat *c-struktur*) och *funktionell struktur* (förkortat *f-struktur*), där c-strukturen är ett frasstrukturträd genererat av en kontextfri frasstrukturgrammatik, medan f-strukturen, som representerar de grammatiska relationerna i en sats och konstituenternas grammatiska egenskaper, är bestämd både av den kontextfria grammatiken och lexikal information.

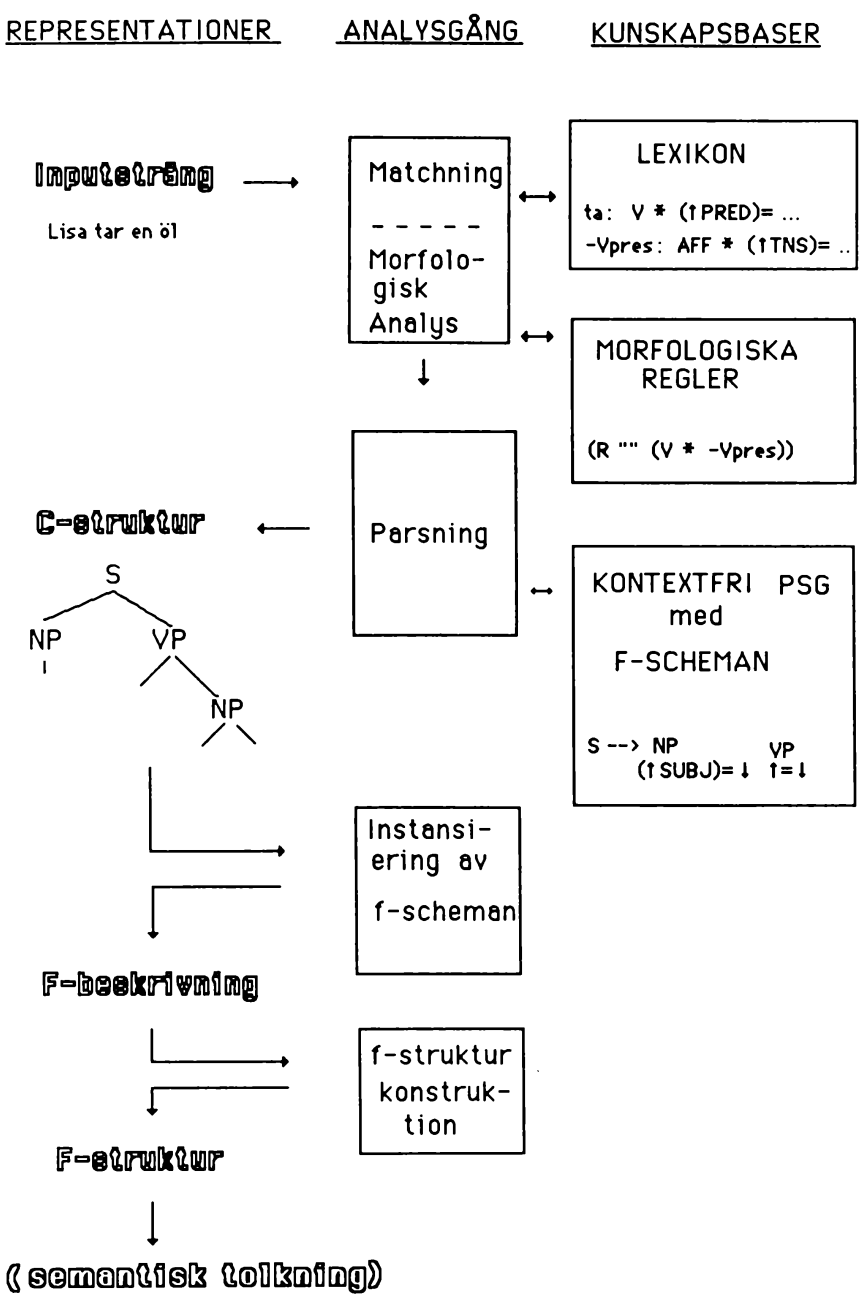
I figur 1 visas en schematisk framställning av de olika komponenter och representationer som en LFG använder sig av tillsammans med en beskrivning av hur en f-struktur härleds. För mer detaljerade redogörelser av teorin hänvisas i övrigt till referenserna i slutet på artikeln.

GWB, eller "The LFG Grammar Writer's Workbench", är en implementering i Interlisp-D med vars hjälp man kan skriva och testa LFG-grammatikor. Den är konstruerad vid Xerox Park och är ett utomordentligt användarvänligt redskap. Den tillåter samtidig display av c-struktur, f-struktur, chart, använda regler och lexikoningångar. Den tillåter också att man arbetar med flera olika grammatikor och lexika på samma gång och att man lånar regler från en grammatik till en annan. Alla delar av LFG-formalismen är dock inte implementerade.

2. En LFG för svenska?

På IDA i Linköping har vi haft tillgång till GWB sedan våren 1985. Vårt intresse för LFG grundar sig främst på vår strävan att utveckla/anpassa en lingvistiskt motiverad, komputationellt effektiv parser för användning i ett gränssnitt mot expertsystem eller databassystem. Ur denna synpunkt är det två saker som är väsentliga: att parsern klarar av att hantera ett tillräckligt brett register av svenska konstruktioner och uttryck och att den gör det effektivt. Min presentation av LFG och GWB här ska ses i detta perspektiv. Någon större svensk LFG har vi inte tagit fram ännu, bl a på grund av begränsningarna i den nuvarande parsern, men ett par mindre svenska fragment föreligger, varav ett är resultatet av ett examensarbete (Månsbacka, u. arb.).

Fig. 1: LFG: komponenter och analysgång.



När det gäller konstruktionen av en större LFG-grammatik för svenska blir behovet av en väl fungerande morfologi uppenbar. Morfologins roll för inkodning av grammatiska funktioner har betonats i LFG, men det finns ingen utarbetad morfologisk teori inom LFG's ram. Vad man kan notera är att den morfologiska informationen endast representeras i f-strukturen. Detta får som konsekvens, åtminstone med nuvarande parsningsmetod, att den morfologiska informationen blir mindre värd än den kategoriella när det gäller att generera hypoteser om meningens struktur. Jag skall ge några exempel på detta i fortsättningen.

En annan företeelse som är vanlig i svenskan är verbpartiklar i lös sammansättning med verbet. Sådana sammansättningar måste i de flesta fall betraktas som särskilda predikat. Deras subkategoriseringsegenskaper skiljer sig i allmänhet från det enkla verbets egna. På grund av att svenskan alltid placerar det finita verbet i en bestämd position blir dessa predikat dessutom ofta diskontinuerliga. Detta ger upphov till vissa beskrivningssvårigheter i LFG, som arbetar med ett snävt, främst komputationellt motiverat lexembegrepp.

Svenskan är speciell också vad gäller s k långdistansberoenden. Det är välkänt att svenskan och de andra skandinaviska språken är mer liberala än t ex engelskan. Den teori för dessa som presenterats (Kaplan&Bresnan, 1982) motiveras delvis utifrån de skandinaviska språkens egenskaper härvidlag. Den formella apparat man använder är betydande: icke-lokala metavariabler, kategoriindex för dessa, kontrolldomäner, begränsande domäner m.m. Hela denna apparat gör ett oskönt intryck, i synnerhet om man jämför med den övriga teorin (jmf. Steedman, 1985). Den är f ö inte implementerad i GWB ännu och jag kommer inte mera att beröra den delen av teorin här.

3. Ett parsningsexempel.

Den lingvistiska kompetensen i en LFG är uppdelad på tre komponenter: ett lexikon, en morfologisk komponent och en frasstrukturgrammatik. Lexikoningångarna är enkla ord eller morfem och för varje ingång ges uppgift om kategoritillhörighet, särdrag och, i förekommande fall, semantisk form och kontrollegenskaper. Det kan se ut så här:

(1) *öl* N D3 (↑ GEND)=UTR, (↑ PERS)=3, (↑ PRED)='öl'

Lisa N PN (↑ GEND)=UTR, (↑ PERS)=3, (↑ NUM)=SING, (↑ PRED)='Lisa'.

en DET * (↑GEND)=UTR, (↑NUM)=SING, (↑SPEC)=INDEF.

ta V * (↑PRED)='ta<(↑SUBJ)(↑OBJ)>'.

-VPres AFF * (↑TENSE)=PRES.

Den första informationen som ges om ordet/morfemet gäller dess kategoritillhörighet och den andra gäller subkategori ('*' betyder att ingen subkategori anges). Därefter anges ordets *funktionella scheman* (kortare, *f-scheman*), som hänför sig till konstituenter i f-strukturen. Så till exempel anger schemat (↑ GEND)=UTR för *Lisa* att den konstituent i f-strukturen till vilken *Lisa* hör ges värdet UTR(um) för attributet GEND(er). Uppåtpilen (↑) är en s k lokal metavariabel varom mera nedan.

Attributet PRED är speciellt. Dess värde är en *semantisk form*, dvs ett uttryck som används i den semantiska tolkningen av satsen. En sådan semantisk form kan innehålla en struktur som talar om hur kringliggande konstituenter ska tolkas i förhållande till ordet självt. Den semantiska formen för *ta* anger således att subjektet i satsen ska tolkas som första argument till 'ta', och att objektet ska tolkas som andra argument. För *Lisa* finns ingen motsvarande struktur i PRED-värdet eftersom detta ord inte har några argument. Grammatiska ord och morfem tilldelas inget PRED-värde överhuvudtaget. Den semantiska tolkningen baseras på f-strukturen (se Halvorsen,1983). Den kommer inte vidare att beröras i det följande.

-VPres är ett morfem. Det hittas i en ordform med hjälp av morfologiska regler. Den morfologiska regelkomponenten i GWB är en slags halvmesyr som utan teoretiska anspråk är inlagd för att underlätta lexikonskrivningen. Den tillåter oss dock utan vidare att analysera en verbform som *tar* i stam och ändelse, som i (2):

(2) tar -> ta + -VPres

Den sammantagna lexikoninformationen för båda dessa element kommer att associeras med den givna ordformen. Effekten av den morfologiska analysen blir i detta fall densamma som om vi hade haft ordet *tar* direkt i lexikon med den information som anges i (3):

(3) *tar* V * (↑ PRED)='ta<(↑ SUBJ)(↑ OBJ)>',
(↑ TENSE)=PRES.

Låt oss nu pröva att parsa meningen (4). För att göra detta krävs förutom det lilla lexikon vi just givit också en uppsättning frasstrukturregler. Dessa finns angivna i (5).

(4) Lisa tar en öl.

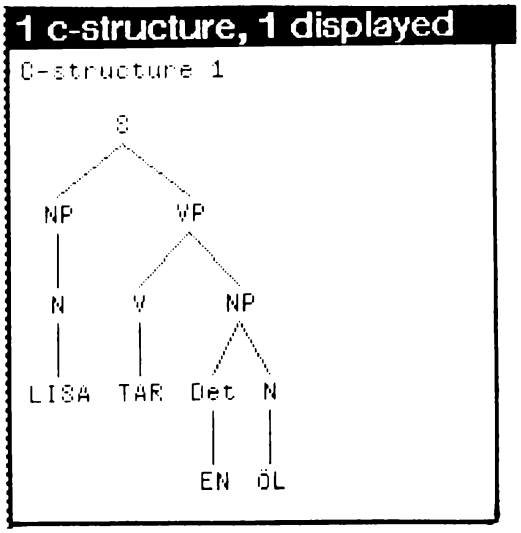
(5)a. S --> NP VP
(↑ SUBJ)=↓ ↑=↓

b. NP --> (DET) N
↑=↓

c. VP --> V (NP)
↑=↓ (↑OBJ)=↓

Frasstrukturreglerna i (5) är något mer komplicerade än ordinära frasstrukturregler därigenom att de är försedda med f-scheman. Den kategoriella informationen i reglerna är dock tillräcklig för att generera en konstituentstruktur för (4), nämligen (6).

(6)



C-strukturen är syntaktiskt sett en slags hjälpstruktur, ett steg på vägen mot den viktigare f-strukturen. F-strukturen bestäms ytterst av f-schemana, både de som stammar från de tillämpade frasstrukturreglerna och de som stammar från lexikon. Låt oss först titta på vad f-schemana i (5)

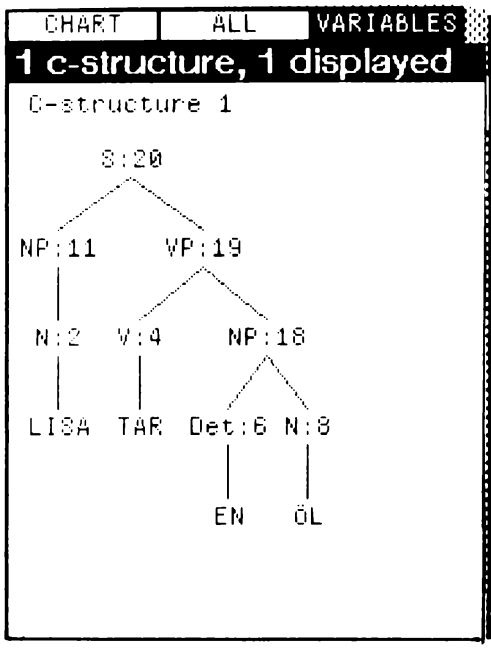
innebär.

F-schemat (↑ SUBJ)=↓, associerat med NP i den första regeln säger kortfattat att satsen S har ett subjekt som utgörs av detta NP. Eller, mer korrekt: den f-struktur som associeras med S har ett attribut SUBJ, vars värde utgörs av den f-struktur som associeras med NP. Uppåtpilen är således en slags pekare mot vänsterledets f-struktur, och nedåtpilen, en annan lokal metavariabel, är en pekare mot f-strukturen för den del av högerledet som den står vid.

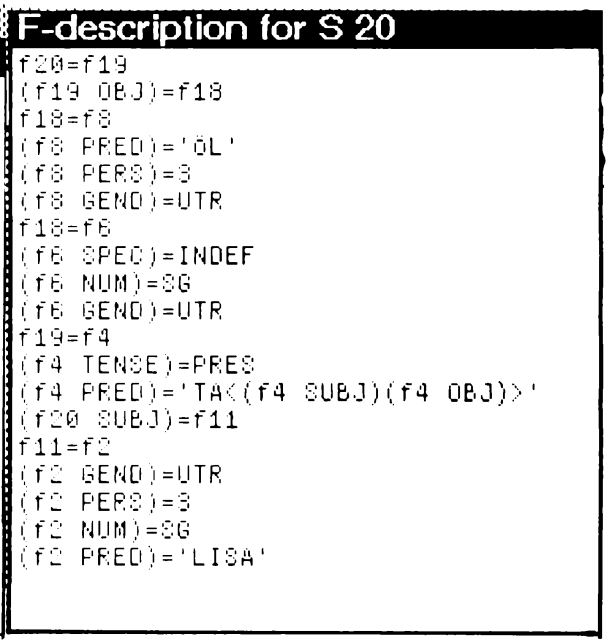
Schemat ↑=↓ unifierar två f-strukturer, den som hör till vänsterledet och den som hör till den konstituenten själv. För VP i den första regeln sägs alltså att dess f-struktur ska unifieras med f-strukturen för S. Detsamma gäller substantivet N i en NP och allmänt för alla konstituenter som kan betraktas som huvudled i sina respektive satser/fraser.

F-strukturen härleds i flera steg. Noderna i c-strukturen numreras först i identifieringssyfte med en särskild algoritm. Resultatet av denna procedur visas i (7). C-strukturnodernas antagna motsvarigheter i f-strukturen anges med samma tal fastän prefigerat med ett 'f'. Dessa symboler utgör faktiska variabler och kommer att ersätta metavariablerna i f-schemana. Schemat (↑ SUBJ)=↓ från grammatikregeln (5a) kommer alltså att associeras med noden 11 i c-strukturen och bli instantierat som (f20 SUBJ)= f11. De övriga schemana från grammatiken och lexikon ger på samma sätt upphov till andra instantierade scheman. Den totala mängden av instantierade scheman kallas en *funktionell beskrivning* (kortare: *f-beskrivning*). F-beskrivningen för vårt exempel visas i (8):

(7)

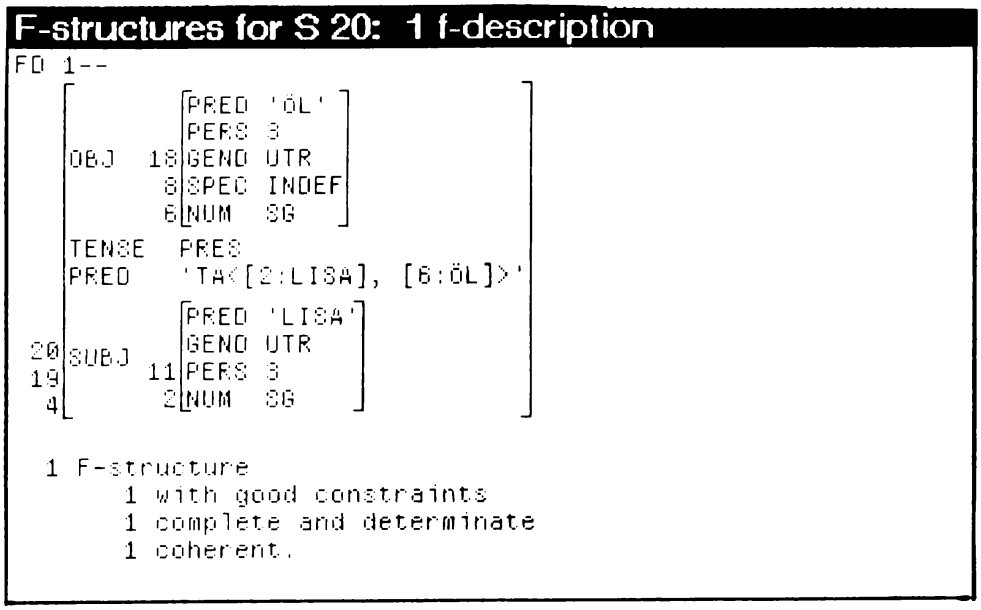


(8)



C-strukturen har nu helt spelat ut sin roll. F-beskrivningen kan betraktas som ett slags ekvationssystem och avgörande för om meningen kan tillordnas en f-struktur är om detta system har en entydig, minimal lösning. Detta är fallet under vissa, specificerbara villkor och det existerar minst en lösningsalgoritm som i så fall konstruerar lösningen. Och lösningen är ingenting annat än meningens f-struktur. I (9) visas f-strukturen för (4), som direkt kan härledas ur (8).

(9)



De villkor som f-beskrivningen måste uppfylla är avgörande för om det analyserade uttrycket klassas som grammatiskt eller ogrammatiskt. Ett sådant villkor gäller *bestämdhet* och *konsistens*: varje attribut måste ha ett, och precis ett, värde. Ett annat villkor gäller *fullständighet* och *koherens*: en f-struktur måste innehålla alla de grammatiska funktioner som dess predikat anger, och får inte innehålla någon styrbar grammatisk funktion (subjekt, objekt eller sekundärt objekt) som inte anges av dess predikat. Den syntaktiska beskrivningen i LFG sker således till lika stor del med den algoritm som kontrollerar f-beskrivningar och konstruerar f-strukturer, som med frasstrukturregler.

4. Parsningsmetoder för LFG.

Den parsningsmetod som används i GWB, och som finns beskriven i Kaplan & Bresnan (1982), bygger till stor del på den ovan refererade lösningsalgoritmen för f-beskrivningar. Härledningen sker här stegvis, från c-struktur via f-beskrivning till f-struktur. De högre nivåerna interagerar inte på något sätt med de lägre i parsningen. Det påpekas också av författarna att syftet med den föreslagna algoritmen inte varit att eftersträva psykologisk rimlighet, eller ens effektivitet, utan istället att leda i bevis att LFG har vissa formellt tilltalande egenskaper såsom avgörbarhet.

De ganska starka hypoteser som ställs för representationsformernas universella giltighet gäller således inte parsningsalgoritmen. Den övergenererar kraftigt i nuvarande skick eftersom den konstruerar alla möjliga c-strukturer som är kompatibla med frasstrukturreglerna och de ingående ordens kategoritillhörighet. Detta antal kan bli stort eftersom den kategoriella informationen i lexikon är ytterligt fattig. Enligt Bresnan (1982b: 294f) skall de lexikala kategorierna inskränkas till att omfatta huvudkategorierna N, A, V, P och S, vilka i sin tur är baserade på två stycken kategoriella särdrag, samt ett fåtal mindre kategorier av typen DET och COMP. Frasstrukturreglerna kan förutom dessa lexikala kategorier endast referera till projektioner av dessa av högst grad 2, såsom NP, PP, VP, AP och S'. Avsteg från dessa begränsningar på syntaktiska kategorier görs dock på andra ställen i litteraturen (se nedan).

Alternativa, mer performansinriktade parsningsmodeller diskuteras på olika ställen i Bresnan (1982a), t ex i artikeln Ford, Bresnan & Kaplan (1982), där man också anstränger sig att visa att en kompetensgrammatik av LFG's typ har ett förklaringsvärde för många performansfenomen. Ingen av dessa har dock implementerats.

5. Morfologins roll i LFG.

Det är ett välbekant faktum att morfologiska signaler är goda indikatorer på frasstruktur, något som för svenskans del tydligast demonstrerats av möjligheterna till korrekt morfologibaserad heuristisk parsning (se t ex Källgren (1984)). Det kan då vara intressant att konstatera att man inom LFG inte alls utnyttjar morfologiska signaler på detta sätt. Tar man hänsyn till denna egenskap hos morfologiska signaler vinner man också en hel del i parsningseffektivitet, i och med att antalet felaktiga genererade c-strukturer minskas betydligt. Den viktiga konsekvensen av detta är inte i och för sig det minskade antalet c-strukturer, utan det minskade antalet hypoteser rörande f-struktur. Det förutsätter dock en mindre förändring av grundkonceptet i LFG.

De funktionella schemana är väsentligen av tre olika slag: semantiska, relationella och morfosyntaktiska. I (10) illustreras ett exempel av vardera slaget:

(10)a. (↑PRED)='ta<(SUBJ)(OBJ)>'
b. (↑SUBJ)=↓
c. (↑CASE)=GEN

De semantiska gäller attributet PRED och är viktiga i syntaxen framförallt genom den argumentstruktur de specificerar. De relationella anger att en viss grammatisk funktion fylls av en viss konstituent och de morfosyntaktiska anger en morfosyntaktisk egenskap i formen av ett attribut-värdepar. Den senare typen av scheman är det enda sättet som LFG använder får att representera morfosyntaktiska särdrag. En konsekvens av detta är att sådana särdrag endast representeras i f-strukturen. En annan är att de inte kommer att utnyttjas i härledningen av c-struktur utan blir "synliga" för parsningsalgoritmen först i steget från f-beskrivning till f-struktur.

Låt oss titta på ett exempel. Regeln för genitiva NP:n kan se ut som i (11) (e. Kaplan&Bresnan, s. 243). Här förekommer dels ett morfosyntaktiskt schema, dels ett relationellt:

(11) NP --> NP N
(↓CASE)=c GEN ↑=↓
(↑POSS)=↓

Eftersom inget av dessa scheman utnyttjas i härledningen av c-struktur, kommer vi att generera hypoteser om närvaron av ett genitivattribut i fall som (12)-(14). Alla dessa hypoteser kommer elimineras i den fortsatta bearbetningen men det förefaller onödigt att generera dem överhuvudtaget.

(12) Lisa skickade Erik(NP) paketet(N).
(13) Pappa(NP) mamma(N) och jag ...
(14) Lisa drack ett glas(NP) mjölk(N).

Alternativt skulle vi kunna tillåta vissa morfologiska egenskaper att vara avgörande för kategoritillhörighet, exempelvis så att ett genitivt substantiv, och en genitiv NP, behandlas på ett annat sätt av frasstrukturreglerna än omarkerade substantiv och NP. I GWB är det nu inte möjligt att behandla syntaktiska kategorier som särdragsknippen, och således heller inte möjligt att låta syntaktiska regler referera till enstaka särdrag. Däremot är det möjligt att ändra kategoribeteckning via en morfologisk regel. En vanlig regel för s-genitiven i svenska kan formuleras på följande sätt:

(15) (s (N -GSg N(Gen)))

Regeln säger att om ett textord kan analyseras i ett 's' och en stam av kategori N, så kan det betraktas som morfologiskt sammansatt av denna stam och suffixet -GSg (för Genitiv Singularis) och tillordnas kategorin N(Gen), vilken alltså skiljer sig från N. Ord av denna kategorin ingår då som huvudordet i genitiva NP, som kan tilldelas kategoribeteckningen NP(Gen). Om vi på detta sätt tillåter särdraget genitiv att vara kategoribestämmande kommer ingen av de felaktiga hypoteserna för satserna (12)-(14) att uppstå.

Vilka morfologiska egenskaper kan man då anse vara kategoribestämmande? Ett svar, vilket är det svar som ges av GPSG (Gazdar et al. 1985) är att alla är det, dvs syntaktiska kategorier är att betrakta som bestämda särdragsstrukturer och alla särdrag har någon roll att spela i någon frasstrukturregel. Ett sådant svar är dock knappast möjligt inom ramen för LFG-modellen, som i så fall skulle förlora det mesta av sin syntaktiska särart, utan bli en GPSG med lite annorlunda semantik. Ett annat, restriktivare svar är att sådana morfologiska egenskaper bör räknas som kategoribestämmande som dels medför specifika hypoteser för den lokala frasstrukturkonfigurationen, och dels är semantiskt redundanta, dvs obehövliga för den semantiska tolkningen. Några exempel:

(1) Genitiva NP skiljer sig från omarkerade NP genom att uppträda i konfigurationen (NP (__ N)). Genitiv är en semantiskt redundant kategori eftersom dess betydelse finns inkodad i funktionen POSS. Någon motsvarande skillnad finns t ex inte för maskulina och feminina NP, vilka däremot kan ha betydelse för anaforisk referens.

(2) Finita verb skiljer sig från infinita genom att förekomma i satser med subjekt och genom att inte kunna stå efter *att*. Finithet i sig ger inget bidrag till den semantiska tolkningen, vilket däremot tempusangivelser som presens och preteritum gör.

(3) Interrogativa och relativa konstituenter har en speciell syntax, vilket motiverar att de ges egna kategorier. Detta tycks också vara Kaplan&Bresnans mening (ibid. s. 243).

Den här distinktionen mellan "kategoriella" och "funktionella" morfosyntaktiska särdrag kan lätt göras formellt i en LFG. Konfigurationella särdrag får räknas som kategoridefinierande och kan därmed spela en roll i frasstrukturgrammatiken, medan de andra definieras med hjälp av f-scheman. Med genitiven som exempel skulle vi alltså kunna ersätta regel (11) ovan med (11').

(11') NP --> NP(+GEN) N
(↑POSS)=↓ ↑=↓

Ett alternativ är naturligtvis att försöka använda f-scheman i ett tidigt skede av parsningsprocessen för att eliminera felaktiga hypoteser. Det är dock oklart hur det i så fall ska gå till.

6. Lexem i LFG och svenskans sammansatta verb.

I GWB är lexikonets uppslagsord en bokstavssträng, eller ett abstrakt morfem som kan hittas via en morfologisk regel. I teorin tänker man sig dock att lexem kan ha mera struktur än så i sina uppslagsord. Bresnan (1982c) beskriver prepositionella passiver i engelskan med hjälp av en regel som inkorporerar prepositioner i verbet, varvid ett sammansatt verb uppstår:

(16) *V-P Incorporation*
Operation on lexical form: (P OBJ) -> (OBJ)
Morphological change: V <--> V: [V P]

Det sista uttrycket bildar ett komplex, som i konkreta instanser av regeln ger upphov till sammansatta lexikonord:

(17) [(speak) (of)] V * (↑ PRED)='speak-of<(↑SUBJ)(↑ OBJ)>'

På samma sätt beskrivs vissa idiomatiska uttryck av typen *take advantage of, find fault with*, dvs som i (18):

(18) [(take) (advantage)] V * (↑ PRED)='t.a.of<(↑SUBJ)(↑ OF OBJ)>'

Vid sidan av dessa uppslagsord finns också under *speak* och *take* hänvisningar till samma semantiska predikat, men med en annan uppsättning grammatiska funktioner:

(19) *speak* V * (↑ PRED)='speak-of<(↑ SUBJ)(↑ OF OBJ)>'

(20) *take* V * (↑ PRED)='t.a.of<(↑SUBJ)(↑ OF OBJ)>(↑OBJ)'
(↑ OBJ FORM)=c ADVANTAGE

Skälen till att olika lexem antas är flera. Det viktigaste är att alla former av lexikal inkorporering i LFG kräver syntaktisk närhet (Bresnan, ibid. s. 37), dvs. orden måste stå intill varandra i satsen. Den komplexa uppslagsformen kan därför inte användas för fall som (21) och (22) där de olika delarna är separerade från varandra.

(21) Advantage was taken of John.
(22) It is of John that I am speaking.

Den definitivt vanligaste beskrivningen av idiomatiska uttryck i LFG-litteraturen är som i (21). Detta innebär att den semantiska formen knyts till en del av det idiomatiska uttrycket, nämligen det som syntaktiskt sett är huvudled. För idiomatiska verb är detta huvudled naturligtvis verbet. Det är lätt att inse att följden av detta blir att ett verb som *ta*, med sin rikedom i användningar, måste bli associerat med hundratals olika semantiska former i lexikon. Antalet lexikonuppgifter blir sedan ytterligare mångfaldigat genom att varje semantisk form har olika möjligheter att inkoda sina respektive argument. Den parsningsteknik som används i GWB aktiverar alla dessa möjligheter varje gång verbet förekommer utan hänsyn till syntaktisk eller semantisk kontext, något som resulterar i en explosion av olösliga f-beskrivningar.

Även ur teoretisk synpunkt leder denna beskrivning av idiomatiska och andra diskontinuerliga verb till besvärliga konsekvenser. Jag vill avslutningsvis visa på detta med en illustration från svenska. Betrakta följande meningar:

(25) Lisa tog en öl.
(26) Lisa tog sig en öl.
(27) Lisa tog med en present.
(28) Lisa tog med sig en present.
(29) Svenskarna tog sig med till finalen.
(30) *Lisa tog sig med en present. (med *med* som betonad partikel)
(31) *Svenskarna tog med sig till finalen.
(32) *Svenskarna tog sig med sig till finalen.
(33) *Lisa tog med sig bort en present

I (25) har vi ett enkelt verb med grundformen *ta*. I (26) har vi ett reflexivt verb, *ta sig* och i (27) ett partikelverb, *ta med*. Ett sammansatt verb kan innehålla både reflexiv och partikel, vilket visas av (28) och (29). Ordningen mellan reflexiven och partikeln är väsentlig. Båda ordningarna kan visserligen förekomma men ger i regel upphov till olika verb (såvida inte den ena ordningen ger upphov till nonsens). *Ta med sig* och *ta sig med* har olika betydelse och olika argumentram. Detta illustreras av (30) och (31). Trots att reflexiven kan förekomma både före och efter en partikel, finns det inga verb som har två reflexiver, en före och en efter partikeln. (32) är uppenbart omöjlig. Det får heller inte förekomma fler än en partikel, vilket illustreras av (33).

Dessa konstruktioner kan vara diskontinuerliga. Mellan verb och partikel kan subjektet stå liksom satsadverben. Mellan verb och reflexiv kan likaledes stå ett subjekt, och åtminstone vissa satsadverb. Då reflexiven står efter partikeln kan dock ingenting komma emellan:

(34) Därför tog Lisa med sig en present.
(35) Svenskarna tog sig inte med till finalen.
(36) Svenskarna tog inte sig med till finalen.
(37) *Lisa tog med inte sig någon present.
(38) *Därför tog med Lisa sig en present.

I LFG är det naturligt att behandla de regelmässiga sammansättningarna med reflexiv resp. partikel med lexikala redundansregler. De sammansatta verb som inte är regelmässiga får behandlas som idiom. I bägge fallen uppstår problemet hur deras uppslagsformer skall se ut. Om vi håller fast vid kravet att inkorporeringar endast kan beröra syntaktiskt närstående led, så följer det att endast partikel + reflexiv konsekvent kan behandlas som en lexikal enhet. Detta innebär då också att "idiommodellen" måste följas i stor utsträckning, dvs den argumentstruktur som hör till ett visst sammansatt verb, kopplas till dess första del, det enkla verbet. Det finns emellertid ett problem med det tillvägagångssättet, nämligen att matchningen av verbets argumentstruktur mot faktiskt funna argument i satsen sker i f-strukturen, där ordningsrelationer inte är representerade.

Om vi vill kunna beskriva skillnaden mellan verb som *ta med sig* och *ta sig med*, *ställa upp sig* och *ställa sig upp*, liksom skillnaden mellan de förekommande *ta igen sig* och *ta sig igenom* och de icke förekommande **ta sig igen*, **ta igenom sig*, så måste vi i LFG göra det med hänvisning till antingen en skillnad i grammatisk funktion eller en skillnad i förekomsten av ett morfosyntaktiskt särdrag.

De två positioner där reflexiven *sig* uppträder är båda möjliga objektspositioner i svenska. Det finns ingen anledning att analysera en NP som står efter partikel som någonting annat än objekt. Verb med och utan partikel kan samordnas med avseende på ett sådant NP, som i (39), och ett NP efter partikel kan utgöra subjekt till ett efterkommande infinitivt komplement, som i (40). (Det antas i LFG att inga oblika objekt kan kontrollera subjektet till ett sådant komplement.)

(39) Lisa köpte och åt upp två glassar.
(40) Lisa sa till Erik att hämta en öl. (med *till* som betonad partikel)

Tittar vi sedan på partikeln så vore det tänkbart att i det ena fallet säga att den har partikelfunktion och i det andra fallet säga att den har någon slags adverbiell funktion. Detta hjälper dock inte särskilt mycket om vi tar hänsyn till den uppsättning grammatiska funktioner som LFG förutsätter. I båda fallen är det funktionen XCOMP, vilken anger en slags generell predikativ funktion, som ligger närmast. Inga oberoende argument talar heller för att man skulle skilja på två olika funktioner, snarare tvärtom. Det är precis samma adverb som förekommer i båda fallen och många transitiva partikelverb kan ha sitt objekt antingen före eller efter partikeln, utan att man därför vill säga att partikeln bytt funktion:

(41) Lisa tog med en vän.
(42) Lisa tog en vän med.
(43) Lisa tog med sig en vän.
(44) Lisa tog en vän med sig.

De båda hypotetiska funktionerna vi talar om kan inte heller förekomma samtidigt, vilket starkt talar för att de inte är olika. Vi kan visserligen ha meningar som (45), men här är *ut* definitivt ett adverbial som inte alls är knutet till verbet, vilket visas av det likaledes möjliga (45).

(45) Lisa tog med en stol ut.
(46) Lisa tog en stol med ut.

Det verkar alltså fel att skilja de båda fallen åt funktionellt. Det förefaller rimligare att säga att den skillnad vi har att göra med är en skillnad i hur sammansatta lexikala enheter uttrycks. I motsats till vad som vanligen är fallet spelar här ordningsföljden mellan olika delar en viktig roll, inte bara formen. Ett sätt att komma runt problemet i LFG vore att i f-beskrivningar inte bara använda attributet FORM utan också ett attribut ORDER som på något sätt representerar ordningsföljd. Att göra detta vore dock att överge den antagna arbetsfördelningen mellan c-struktur och f-beskrivning. För, om det visar sig nödvändigt att använda attribut som representerar uttrycksform och ordning i f-strukturen för vissa typer av uttryck, så kan man fråga sig varför man inte bör använda dem generellt.

Litteratur.

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

Bresnan, J. (1982b): *Control and Complementation*, i Bresnan (1982a): 282-390.

Bresnan, J. (1982c): *The Passive in Lexical Theory*, i Bresnan (1982a): 3-86.

Ford, M., Bresnan, J. & Kaplan, R.M. (1982): *A Competence-Based Theory of Syntactic Closure*, i Bresnan (1982a): 727-796.

Gazdar, G., Klein, E., Pullum, G.K. and Sag, I. (1985): *Generalized Phrase Structure Grammar*, Basil Blackwell.

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

Kaplan, R.M. & Bresnan, J. (1982): *Lexical-Functional Grammar: A Formal System for Grammatical Representation*, i Bresnan (1982a): 173-281.

Källgren, G. (1984): *HP - A Heuristic Finite State Parser Based on Morphology*, i Sågvall Hein (utg.): *Föredrag vid de nordiska datalingvistikdagarna 1983*, Uppsala Centrum för Datorlingvistik, 155-161.

Levin L., Rappaport, M. & Zaenen A. (utg.) (1983): *Papers in Lexical-Functional Grammar*, IULC, Bloominton Ind.

Månsbacka, C. (u. arb.): *En svensk minigrammatik i LFG*. Linköping, IDA.

Steedman, M. (1985): *LFG and Psychological Explanation*, Linguistics and Philosophy 8:3, 359-385.

Olli Blåberg
Forskningsenheten för datalingvistik
Helsingfors universitet

BÖJDA SVENSKA SUBSTANTIV I BETA

1. Inledning

Svensk substantivböjning har oftast beskrivits med deklinationer. Många enskilda deklinationsmarkeringar är emellertid redundanta. Böjningsparadigmet syns ofta i vissa mönster i grundformen då genus är given. Denna artikel rapporterar om en undersökning vars syfte var att undersöka om de avvikande typerna kan räknas upp så att resultatet blir en beskrivning av den produktiva böjningstypen. Relevanta ytmönster hittas bl a i Svensk Baklängesordbok (1981). Som testmaterial har använts ett sampel på ca 2.500 substantiv ur Svenska Akademiens Ordlista (1973).

Beskrivningen är gjord i BETA, definierad och implementerad av Benny Brodda. BETA påminner om de universalmaskiner som utvecklades av matematikerna Post respektive Turing. Med universalmaskin förstås en abstrakt apparat som kan utföra alla de operationer som överhuvudtaget kan utföras algoritmiskt. Således går det att uttrycka de mest olika lingvistiska regelsystem i BETA, som inte är någon morfologisk teori i sig. Det förblir en uppgift för språkforskaren att motivera reglerna.

Regelfilen SWEPARAD i sin nuvarande omfattning består av ca 1.000 BETA regler (ca 7.000 tecken). För att tolka den har det använts en BETA implementering som är skriven i PASCAL av Kimmo Koskenniemi. Denna något mindre version får plats i en PC mikrodator. I inmatningen till SWEPARAD ges det önskade ordets grundform och genus. Eventuella sammansättningsjunkturer förutsätts vara markerade (med "="). I övrigt arbetar SWEPARAD med den ortografiska teckensträngen.

IN: EN+HÄST

UT: HÄST HÄSTS HÄSTEN HÄSTENS HÄSTAR HÄSTARS HÄSTARNA HÄSTARNAS

Många språkvetare har velat se analogier mellan datalingvistiska program och hur den mänskliga hjärnan antas processera språkliga data. Jag jämför datorer helst med pålitliga räknedosor som utför väldefinierade operationer snabbt och ofelbart.

2. Beskrivningsobjektet

Många lingvistiskt sett rent normativa beslut måste fattas innan regler kan skrivas. Oftast gäller det val mellan högspråkliga scheman och faktisk variation. Typisk är allomorfen -ena 'pluralis definit'. Den uppträder s g s aldrig i text, ?husena, fast suffixet är en del av (de flesta) svenskars muntliga språkbruk.

I några fall bildas böjningsparadigmet av flera stammar, t ex amerikan vs. amerikanare. Typen ansökan, ansökningar är tydligt heterotematisk. Skriftspråkets vacklan skapar ytterligare variation. T ex schejk, shejk och sjejk kan tolkas som olika "böjningsstammar" av ett lexem. Dessutom finns t ex etablerade kortformer som stan (jfr staden).

Det finns inget självklart svar på huruvida typerna arbetarn respektive arbetarena systematiskt skall ingå i paradigmet för nomen agentis suffixet -are utöver de stilistiskt neutrala arbetaren och arbetarna. Många nyare lånord saknar en stabil böjningsnorm, t ex baby (babyer / babies / bebisar) och schlager (schlagrarna / schlagersen). I princip är det omöjligt att gardera sig mot alla upptänkliga typer av framtida pluralisbildning.

I Elerts (1970, 142f.) förteckning ingår flera typer som här bedömts som främmande. Den engelska ortografins alternation y-ies (hobby, party, sherry) är jämförelsevis stabil i språket, vid sidan av den av Akademien rekommenderade typen hobbyer.

Några formella adjektiv, t ex bekant, används som vanliga substantiv. Adjektiven och verben kunde givetvis återges i BETA analogt med substantiven.

I det ortografiska alfabetet har tecknet é grafematisk status (jfr arme vs armé eller ide vs idé). Likväl finns vacklan, speciellt då formerna är entydiga, t ex armeerna och ideerna, jfr arméerna, idéerna.

3. Allmänt om strategin

SWEPARAD påminner till sin allmänna struktur mycket om systemet WGEN skriven av Jouko Lindstedt. WGEN genererar böjningsparadigm för finska nomen (Koskenniemi 1985). I synnerhet Hellbergs (1974, 1978) och Broddas (1979, 1982) arbeten kring svensk morfologi har också givit viktiga impulser.

Många stammar innehåller i sin form information om vilka suffix de tar. Men inte alla ord underkastar sig normalfallet, t ex omljudsorden och ord med vissa avledningssuffix. Reglerna för de avvikande orden utgör ett filter som befinner sig före normalfallets regler och släpper igenom de stammar som inte innehåller markerade mönster.

Sedan de markerade mönstren känts igen under analysens gång, blir de korrekta suffixen insatta. T ex i stammen sko hittas inga relevanta mönster och paradigmet bestäms till det omarkerade för utrumgenus på vokal, d v s skon respektive skor. Pluralis definit skorna samt genitivformerna följer av beslutet. Ordet profil däremot har mönstret -VC(*)VC, och böjs profilen, profiler, *profiln, *profilar.

I implementeringen genereras det flera ut-strängar av en enda in-sträng, vilket förutsätter förgreningsställen. Om förgreningen placeras s a s tidigt, slösas det med resurserna, vilket leder till längre exekveringstider och sämre överblick. En regel bör optimalt inte tillämpas mer än en gång.

4. Allomorfin i substantivböjningen

Allomorfin i böjningen är inte obetydlig. Man måste fastställa vilka allomorfer som härleds med regler och vilka som postuleras som lexikala. I SWEPARAD härleds inga ändelser från några andra med regler. Som lingvistiskt argument kan användas morfologiska "minimala" eller "subminimala par", med analogier i fonologin. Tanken har utvecklats från Karlssons (1983) begrepp kvasimorfem.

Pluralissuffixen -r, -ar och -er förekommer i s g s identisk fonologisk omgivning, i enstavingar på -ö, öar, köer samt mör. Även för suffixen -ar och -or hittas ett "minimalt par", vågar, vågor. Analogt finns nollsuffixet också i den grammatiska kontexten utrumgenus, tekniker, *teknikrar. T ex Svensk deskriptiv

grammatik (1981) förutsätter fem deklinationer med grundallomorferna -or, -ar, -er, -Ø samt -n. I SWEPARAD inkluderas även -r bland pluralissuffixen. För singularis definit finner man "minimala par" som akademin och akademien samt knät och knäet. Lexikalt förutsätts således alla allomorferna -n, -en, -t samt -et.

En regel kunde härleda t ex -r efter vokal i stamslut från lexikalt -er: "stryk e i suffix efter vokal och morfemgräns". Antalet lexikala enheter reducerades, vilket inom vissa teorier uppfattats som målet för språkbeskrivningen. T ex mö+er skulle härledas rätt: mör, men kö+er skulle bli *kör utan markering av stammen med någon "konsonant", deleterad senare av någon annan regel. Man kan tänka sig analoga regler för neutrum pluralis. Pluralis indefinit av segel kunde postuleras som lexikalt segel+n där suffixet -n deleterades efter konsonant (jfr häfte-n). Dock bör man inte låta typen regeln framstå som ogrammatisk.

Även en rik uppsättning systematiska allomorfer tillåter generaliseringar. Distributionen av suffixen -r, -ar och -er styrs ofta av den fonologiska kontexten, men inte alltid. Det finns en tendens enligt vilken utrum ord tar pluralissuffix på -r (-ar, -er, -r, -or) och neutrum ord suffix utan -r, men inte heller denna princip finns genomförd. Av dessa fakta dras den slutsatsen att suffixen inte bör genereras uteslutande varken på basis av den fonologiska eller grammatiska kontexten.

5. Grundmönstret

Det omarkerade mönstret kombinerar genusinformationen och den fonologiska informationen om ordet, t ex stavelseantalet det sista tecknets fonologiska egenskaper. I grundmönstret i SWEPARAD ingår ord med tryckaccenten på annan än sista stavelse. Tryckaccenten markeras ju inte systematiskt i ortografin.

I singularis definit får enstaviga utra suffixet -n efter vokal samt -en efter konsonant utom flerstaviga på -Vl eller -Vr, hästen, tröskeln, dollarn. Neutrum får -t efter vokal samt -et efter konsonant. Enstavingar på får vokal suffixet -et, häftet, huset, knät samt knäet.

I pluralis tilldelas alla utra på konsonant ändelsen -ar samt på vokal -r, hästar, skor. Neutra böjs med suffixen -n respektive -na efter vokaler samt nollsuffixet respektive -en

efter konsonanter, häften, häftena; tak, taken. Ytterligare har neutrum tvåstavingar på stamslutande -er det definita pluralissuffixet -na, fönsterna utöver fönstren (Hellberg 1976).

Genitivsuffixet -s tillkommer efter stammar på andra tecken än s, x och z. Övriga får det ortografiska suffixet ´. Undantagsvis används apostrofen i ord som i uttal slutar på ett sje-ljud, t ex Bangladesh´ (jfr Bangladeshs). Bangladesh uppfattas ogärna som en genitivform. I SWEPARAD får dock inte ens de som slutar på en entydig sje-ljudssekvens någon apostrof som genitivmarkör.

I grundmönstret ingår också en morfofonologisk alternation. Senare än i första stavelse stryks e:et i sekvenserna -el, -er eller -en, ifall en suffixvokal följer omedelbart, trösklar, fönstren. Produktiviteten hos denna alternation är ett av de starkare argumenten för att både de korta och långa allomorferna för singularis definit postuleras lexikalt, d v s e:et i suffixen -n/-en respektive -t/-et varken stryks eller sätts in under derivationen. I annat fall skulle det bli avsevärda problem med regelordningen i derivationer av typen tröskeln, jfr *trösklen.

6. Den ordfinala tryckaccenten

Undantagslöst (?) får utrum substantiv med final tryckaccent pluralissuffixet -er samt den definita singularisallomorfen -en, optionellt för stammar på vokal, jfr akademien, akademin. Ordet budget kan ta både -ar och -er då betoningen ligger på första stavelse. Om tryckaccenten placeras på andra stavelse, är endast -er möjligt. Detta visar sambandet mellan tryckaccenten och paradigmtillhörigheten. Neutra har den längre allomorfen för singularis definit optionellt för alla stammar, partiet och partit. Ingen ändelse -na finns för typen tvåstavingar på -er: klaver, klaveren, *klavren, *klaverna, jfr fönstren, fönster, fönsterna.

Ytterligare blockeras grundmönstrets morfofonologiska alternation: kamelen, kameler, *kameln, *kamler, *kamlar; fenomenet, fenomenen, *fenomnet, *fenomnen. Den ordfinala tryckaccenten är ett tillräckligt villkor för avsaknaden av alternationen, men inte ett nödvändigt, siden, sidenet, *sidnet.

Då final tryckaccent inte markeras direkt i ortografin måste man uppnå samma effekt med andra medel. T ex tvåstavingar på dubbelkonsonant markeras för detta paradigm. Mönstret -VC(*)VCC

räknas från slutet, och de två sista konsonanterna är identiska, t ex metall, metaller. Ett relativt informativt mönster är också -VC(*)VC, t ex profilen, *profiln och kanyler, *kanylar.

Några mönster är systematiskt flertydiga. Schemat -VC(*)an räcker inte till för entydig paradigmbestämning, afrikan men predikan. Samma gäller för -VC(*)is: polis, poliser men tjockis, tjockisar. Lösningen är att räkna upp de ord som tillhör den numerärt mindre eller mindre produktiva gruppen. Lösningen är otillfredsställande då bägge typerna är stora och produktiva. De är olika till den fonologiska formen och sammanfaller endast i ortografin. Det finns också en typ där tryckaccenten alternerar, på första stavelse i singularis och på sista i pluralis: kansler, kanslern, *kansleren, kanslerer, *kanslrer.

7. -ar eller - er för utrumgenus på konsonant

Relativt många substantiv böjs mot ovan presenterade regler utan några till formen reducerbara principer. Främst denna egenskap i böjningen motiverar begreppet deklination i svenskan. För flera fonologiska typer är valet mellan -ar och -er fixerat skilt för varje lexem. Det heter vild, vildar men bild, bilder och hund, hundar men kund, kunder. Ytterligare kan t ex stammar på vokal ta typiska konsonantsuffix, t ex öar och köer, eller ord på konsonant kan ta suffixet -or, t ex ros, våg, trots att suffixet i övrigt är förbehållet stamtypen utra på -a, flicka.

Framförallt -ar och -er efter konsonant i stammar med tryckaccenten på första stavelse är relevanta. De övriga typerna är små och det kan knappast hittas några regler. Några homografer skiljs åt av deklinationen, t ex bal, bank, bas, cykel, fil, form, gång, mask, regel, slav, vals o s v. Några sådana formella kriterier finns inte som kunde ge en uttömmande beskrivning av distributionen mellan -ar och -er. Dessutom finns ord där -ar och -er är i s g s fri variation, t ex checker och checkar.

Grova mönster kan dock stipuleras. Dessa verkar vara mera eller mindre godtyckliga, men nyttiga tendenser i lexikonet. Vad som synkront ligger bakom vilket suffix som fixeras för vilket lexem torde sakna ett enkelt svar. Linell (1972, 32f.) noterar att det ofta är ordets innehåll (och härstamning) som motiverar dess böjning. Abstrakta och vissa inlånade substantiv tenderar

att ha -er böjning. Bl a paret balar eller baler ger en intuition om detta. En beskrivning är alltså endast en återgivning av en lista på undantag, inte en förklaring av fenomenet.

Några tecken är så gott som irrelevanta då de av fonotaktiska eller ortografiskt-historiska skäl sällan förekommer i svenska ordslut, typiskt c, h och w. Uppenbarligen av språkhistoriska skäl verkar -er dominera i stammar på vissa dentalkonsonanter, t ex -d (kod, skillnad), -s (vas, klass) och -t (helhet, stat).

Det förutsätts undantagslistor och submönster. Det finns t ex ord på -d som tar -ar, sked, värld, tråd m m. Tillvägagångssättet är analogt i normalfallet: orden på -g tilldelas pluralis -ar, dag. Men bl a de som slutar på -rg får suffixet -er, färg, utom undantagen, korg. Dels skall undantagen till normalfallet räknas upp, d v s sak, saker trots normalfallets burk, burkar, dels undantagen till undantagslistorna, hytt, hytter men hatt, hattar. Systemet fungerar inom ramen för det vokabulär som utgjort utgångspunkten. Men då det inte blivit klart vad som egentligen styr valet av antingen -ar eller -er, finns inga garantier för att alla nya ord kommer att gå enligt samma mönster.

8. Typerna flicka och pojke

De språkhistoriskt jämförbara typerna flicka och pojke beter sig olika i datorbeskrivningar, jfr Hellberg (1978, 18). Typen flicka kan hanteras med ett fåtal triviala regler. Varje utrum stam på -a bildar singularisformerna med a:et kvar i stammen och pluralisformerna utan a, flickan, flickor. Enda komplikationen kommer från det faktum att de intermediära representationerna för typen kamera inte deltar i den morfofonologiska alternationen, således kameror, *kamror. De egentliga undantagen till reglerna för stamslutande -a är få, t ex historia, kollega.

Typen pojke kan inte ges någon uttömmande beskrivning. Ordfinalt -e i singularis kan signalera tre olika paradigm: pojke, pojkar; linje, linjer (normalfallet) eller idé, idéer. Då är idiosynkratiska (avlednings-) suffix, t ex -are, -ande samt -ge, garage, redan borträknade. Enda alternativet är att gå igenom alla mönster som kan utnyttjas. Det kan tyckas att resultatet saknar elegans, men det fungerar.

9. Några komplikationer

Singularis definit är oftast regelbunden. Dock saknar t ex substantiven med -an efter verbrötter definita singularissuffix, t ex ansökan, *ansökanen. Utra med final tryckaccent på -n får ofta ingen markör för definit singularis i tal (Thorell 1977, 31f.). Formen telefon (även enstaviga, scen) kan tolkas som antingen indefinit eller definit. Pragmatiskt kan man motivera några paradigm som saknar markörer för definit singularis, bland dem månadsnamnen, t ex januari. Undvikande av homonymi verkar i vissa fall vara den bästa tillgängliga motiveringen, t ex bit är inte singularis definit av bi, men knät är en form av knä.

Ett ords genus syns i singularis definit. De ord som alternativt har båda definita suffixen har två genus, t ex paraply. Valet av pluralissuffix predestineras däremot inte av genus. Visserligen finns det en stark tendens till det grundmönster som diskuterades ovan, men ofta är det antigen ett genomskinligt avledningssuffix eller rentav fonologiska mönster som är starkare indicier för val av pluralisändelserna. Jag ser ingen orsak att tala om "genusbytare" (jfr Teleman 1969, 172; Linell 1972, 33).

Bland utrumgenus är det i synnerhet avledda substantiv med -are, -ande och -(ik)er som tar oväntade suffix, arbetare, handlande, tekniker, inte *arbetarar, *handlander eller *teknikrar.

Bland neutrum är iögonenfallande t ex flerstaviga neutra på betonat -i som oftast har pluralis -er, t ex kansli. Dock böjs t ex staffli i pluralis definit stafflien eller stafflierna. Även orden på -um är speciella: antingen enligt typen faktum, fakta eller museum, museer. Analysen görs mera problematisk av tendensen att böja även dessa enligt grundmönstret. Formen fakta skall normativt ersättas med grundmönstrets pluralis indefinit faktum. Ibland träffas även typen museumet vid sidan av museet.

I några synkront genomskinliga sammansättningar böjs efterleden annorlunda än som fristående stam. Som exempel kan nämnas jungfru, jungfrur enligt grundmönstret, inte *jungfruar. Efterleden kan också böjas mera markerat än den fristående formen: bokstav, inte *bokstavar utan bokstäver.

Massorden verkar ofta inte ha ens potentiella pluralissuffix. T ex substantivet mat kan heta varken *matar, *mater eller *mator i pluralis. Detta visar att det inte enbart är

formen som betingar böjningen hos alla ord utan det är i innehållet man hittar motiveringen till denna morfologiska egenhet.

I vissa ortografiska ställningar dubbleras respektive förenklas ordfinala -m och -n: program, programmet, *programet; mun, munnen, *munen. I den mån det finns s k intern böjning i svenskan, täcks den inte av SWEPARAD, t ex prins-gemål, prinsen-gemålen samt kronprins-regent, kronprinsen-regenten.

10. Avvikande paradigm

I svenskan återfinns tendensen att det ofta är högfrekventa, centrala begrepp i kulturen som har ett avvikande böjningsmönster. Ord som man, moder, son, finger, öra, fot, huvud, sommar, morgon, afton, djävul, bok, grej, bryter alla mot grundmönstret. Några egentliga formella eller semantiska kriterier verkar inte finnas. Givetvis implicerar tendensen inte att alla benämningar på "centrala begrepp" måste ha avvikande böjning. T ex kroppsdelsbetecknande ansikte, hår, arm, knä, tå, släktskapsbetecknande syster, faster och tidsbetecknande dag, dygn, vinter, vår, höst m m böjs enligt grundmönstret.

Karaktäristiskt är singularisformerna oftast regelbundna. Men även avvikande singularis definit förekommer, t ex sommaren, djävulen, papperet, himmelen utöver sommarn, djävuln, pappret, himmeln, som går enligt grundmönstret (jfr dollarn, konsuln, numret, muskeln). I pluralis kan avvikelserna däremot vara relativt stora och idiosynkratiska. Exempelvis paradigmet man, män, männen, även mannar, är ett unikum.

Det verkar finnas delgrupper också bland de avvikande orden. Det finns t ex många omljudsord på -and, and, brand, hand, rand, strand, tand, land (neutrum !). Få talare skulle dock vara villiga att acceptera *sänder som pluralis av massordet sand.

Enda möjligheten att återge dessa avvikande paradigm är att ge en förteckning över dem på ett eller annat sätt. Det skulle emellertid innebära en stor belastning för regelapparaten och därmed också för den exekverande implementeringen om man tvingades räkna upp varje enskild böjd form. Därför har det visat sig vara förnuftigt att markera vissa lexikala ingångar med "morfofonem" som möjliggör endast en lexikal representation för t ex omljudsorden av typen stad, städer.

11. Sammanfattning

Modern svensk substantivböjning motiveras av många faktorer. Dels finns grammatisk genus, häst-en, hus-et. Dels väljs suffixen på basis av stammens fonologiska form, häst-ar, sko-r, idé-er. Dels finns fonologiskt omotiverade paradigm, bal-ar, bal-er, och paradigm där godtycklig fonologiskt betingad paradigmtillhörighet är starkare än genus, kansli-et, kansli-er, *kansli-n. De sistnämnda typerna motiverar egenskapen deklination hos substantiven. Det förekommer också fonologiskt motiverad alternation i stammarna, muskel, muskler, som saknas i regel då de fonologiska villkoren inte är uppfyllda, kamel, kameler. Ytterligare upprätthålls ett stort antal idiosynkratiska paradigm, bok, böcker, där ingen motivation står att finna i stammens form.

Den svenska böjnigen är ett språkligt delsystem som måste motstå tryck, av vilka t ex främmande språk och inhemsk variation inte är de minst betydelsefulla. Således finns optionalitet också i böjningen, betingad av många olika faktorer. Denna variation accentuerar språkbeskrivningens allmänna normativa karaktär.

Erfarenheterna från SWEPARAD visar att det inte finns någon enkel lösning på datoriserad svensk paradigmgenerering. Men det är faktiskt möjligt att konstruera ett tillfredsställande system som kan utvidgas utan svårighet och som fungerar väl på en mikrodator. BETA har återigen visat sig vara ett utmärkt lingvistiskt verktyg för morfologisk datorbeskrivning.

Datalingvistiken tvingar fram explicita beskrivningar. Den syn på svensk substantivböjning som tillämpats här avviker på flera punkter från gängse uppfattning, men kan likväl försvaras. Å andra sidan är det klart att flera olika beskrivningar kan alla fylla sin funktion. T ex pedagogiska målsättningar behöver inte nödvändigtvis sammanfalla med datalingvistiska syften.

Litteratur

Brodda, B. (1979), Något om de svenska ordens fonotax och morfotax: iakttagelser med utgångspunkt med experiment med automatisk analys. Papers from the Institute for Linguistics, University of Stockholm 38.

- - - (1982), Yttre kriterier för igenkänning av sammansättningar. Svenskans beskrivning 13. Meddelanden från institutionen för nordiska språk och nordisk litteratur vid Helsingfors universitet, serie B nr 6. 102-114.

Deskriptiv svensk grammatik (1981), Red. Holm, B. & Nylund, E. Trelleborg. 10:e tryckningen.

Elert, C. (1970), Ljud och ord i svenskan. Uppsala.

Hellberg, S. (1974), Graphonomic Rules in Phonology. Nordistica Gothoburgensia 7.

- - - (1976), Bestämd form pluralis. Nysvenska studier 55-56:227-250.

- - - (1978), The Morphology of Present-Day Swedish. Stockholm.

Karlsson, F. (1983), Suomen kielen äänne- ja muotorakenne. Juva.

Koskenniemi, K. (1985). A system for generating Finnish inflected word-forms. I: Computational Morphosyntax. Report on Research 1981-1984. Red. Karlsson, F. Publications of the Department of General Linguistics, University of Helsinki, no. 13. 63-80.

Linell, P. (1972), Remarks on Swedish Morphology. Reports from Uppsala University Department of Linguistics nr 1.

Svensk Baklängesordbok (1981), Allén, S. & Eeg-Olofsson, M. & Gavare, R. & Sjögreen, C.. Nacka.

Svenska Akademiens Ordlista över svenska språket (1973), 10:e upplagan. Stockholm.

Teleman, U. (1969), Böjningssuffixens form i nusvenskan. Arkiv för nordisk filologi. 84:e bandet. Lund.

Thorell, O. (1977), Svensk grammatik. 2:a upplagan. Stockholm.

Lars Borin
Uppsala University
Center for Computational Linguistics
Box 513
S-751 20 UPPSALA
Sweden

WHAT IS A LEXICAL REPRESENTATION?

1. Introduction

In this paper, I will discuss one aspect of the lexicon, namely its morphological organization.

For about two years I have been working with Koskenniemi's two-level model (Koskenniemi 1983), on a Polish two-level description. In this work, I have become more and more interested in the formalism itself, something that has tended to push work on the language description into the background. It seems to be the case that in most concrete two-level descriptions, the rule component is forced to carry too heavy a burden in comparison with the lexicon, perhaps because the lexicon is very simple as to its implementation. Like many other lexical systems in computer applications it is implemented as a tree, with a root node and leaves, from which the lexical entries are retrieved when the analysis routine has traversed the tree. The two-level lexicon is a bit more sophisticated than this, however, in that there is not only one, but several lexicon trees, the so called mini-lexicons. The user links the minilexicons into a whole, most often into a root lexicon and a number of suffix lexicons. In Hockett's terminology, we could speak of an Item-and-Arrangement (IA) model (Hockett 1958, pp 386ff). Elsewhere I have characterized this kind of lexicon system as most suited for describing suffixing languages with a comparatively high degree of agglutination (Borin 1985, p 35); This is mainly due to the fact that the system works according to what Blåberg (1984, p 61) aptly has termed the "forget-where-you-came-from"

principle. It makes it very hard to describe discontinuous dependencies within word forms; one is forced either to duplicate a considerable amount of information in the lexicon or, at least in the two-level model, let the two-level rules take care of some of the morphotax description, which is not their primary purpose (cf. Karttunen 1984, pp 178-181).

In the minilexicons, stems and affixes are grouped into conceptually motivated collections and the links between them describe the morphotax of the language in question. It is not only morphotax, however, that is described by the lexicon linkages: alternations that comprise more than a single segment are often taken care of in separate minilexicons, and in this way one tends to mix linguistically motivated categories with categories that are introduced to ease the work of the lexicon writer; cf. the "technical stems" in Hellberg's system for Swedish morphology (Hellberg 1978, p 13ff; Doherty et al 1986).

2. The problem

It seems that one would need a more sophisticated lexicon system to take care of a number of important morphological phenomena that, for different reasons, should not fall in the domain of morphophonological rules, be they of the generative kind or two-level rules. Here is a representative, even if not exhaustive, list of the kind of phenomena I have in mind:

- Discontinuous morphs, like in Sw. förstora 'enlarge'; Ger. gesagt 'said'; Po. najstarszy 'oldest'.

- Inflection of certain compound types, like in Fi. kolmekymmentäviisi 'thirty-five', Adessive kolmellakymmenelläviidellä; Ru. vagon-restoran 'restaurant car', Genitive vagona-restorana.

- Reduplication, like in Gr. leipō 'leave', Perfect leloipa.

- Suprasegmental features, like accent, as far as they are reflected in the morphology, like in Ru. bol'šój 'big', ból'šij 'bigger'.

I am currently working on a lexicon system for the two-level model that should be capable of dealing with these phenomena, in a formalism that is intentionally in line with traditional linguistic taxonomy. In my view, one should aim at a clear separation of the conceptual organization of the lexicon description from implementational details. The former is important from a theoretical linguistic point of view, while the status of the latter is at best uncertain.

3. The Lexicon Formalism

Fig. 1 shows a small sample lexicon in the format I propose to use<1>. As can be seen, morphotax is explicitly specified, in the form of regular expressions. The first morphotax specification in fig. 1 states that the category Adj (for adjective) has the constituents Stem, Comp (comparative suffix) and Final (gender/number/case portmanteau morphs), in the order they are given. The constituent Comp is optional, which is signalled by the parentheses around it. The category Adj has two alternative constituent structures, which are separated by a comma in the specification. The other possible structure given in fig. 1 is: Sup (superlative prefix), Stem, Comp and Final, where none of the parts are optional. The constituents in the morphotax specifications correspond to (groups of) minilexicons, where the minilexicons have names of the form 'category.constituent', e.g. things that can fill the Stem position in category Adj are found in

---

<1> Since this particular lexicon is used only to illustrate the lexicon formalism, I have taken the liberty to mix Polish (adjectives) and Russian (noun-noun compounds) in it.

```
MORPHOTAX
     Adj = Stem (Comp:(Degree:Pos)):(Degree:Comp) Final,
           Sup Stem Comp Final;
     Noun = Stem (Case:(Case:Nom)),
            Stem (Case:(Case:Nom)) Hyphen Stem (Case:(Case:Nom));
END

LEXICON Adj.Stem :(Cat:Adj) =
     ENTRIES
        star (Type:Qual),
        now  (Type:Qual),
        rad  (Type:Short);
LEXICON Adj.Sup :(Degree:Sup) =
     ENTRIES
        naj+ ;
LEXICON Adj.Comp :(Type:Qual) =
     ENTRIES
        +sz ;
LEXICON Adj.Final :(Type:Qual) =
     ENTRIES
        +y  (Numb:Sg Gend:Masc Case:Nom),
        +a  (Numb:Sg Gend:Fem Case:Nom),
        +e  (Numb:Sg Gend:Neut Case:Nom);
LEXICON Adj.Final :(Type:Short) =
     ENTRIES
        0  (Numb:Sg Gend:Masc Case:Nom),
        +o (Numb:Sg Gend:Neut Case:Nom),
        +a (Numb:Sg Gend:Fem Case:Nom);
LEXICON Noun.Stem :(Cat:Noun) =
     SUBLEX Dim :(Form:Dim) =
        Ek ;
     ENTRIES
        vagon ,
        restoran ,
        zegar Dim,
        ogon Dim,
        dub ,
        velikan ;
LEXICON Noun.Case =
     ENTRIES
        +a (Case:Gen),
        +u (Case:Dat),
        +e (Case:Loc);
LEXICON Noun.Hyphen =
     ENTRIES
        - ;
END
```

Figure 1.

the lexicon(s) Adj.Stem<2>. Thus, the minilexicons are intended to group morph(eme)s of similar morphological categories. In addition, there is the concept of sublexicons. These were introduced for the sake of space and work economy. In an inflectional morphology, the sublexicons can be used to collect e.g. word endings (like derivational suffixes) with special inflections, like in the lexicon Noun.Stem in fig. 1, where the sublexicon Dim contains a diminutive suffix.

To account for phenomena like discontinuous morphs and agreement phenomena within words there is another mechanism apart from the morphotax specifications, namely feature-value graphs (directed acyclic graphs, or DAG:s), that are checked for mutual consistency and added on to as the analysis routine moves through the lexicon. The DAG:s can appear at various points in the lexicon:

1) On a constituent in the morphotax specifications.
2) In a minilexicon or sublexicon as a whole.
3) In an individual lexical entry.

The adding-on procedure is a kind of unification<3>, as described in e.g. Karttunen 1984. This ensures that word forms like Ru. *vagon-restorana will not get any analysis, since (Case:Nom) in the first part will not unify with (Case:Gen) in the second part. The final DAG of a successful analysis is produced as part of the output.

The morphotax specifications and the DAG:s together succeed very nicely in capturing the traditional linguistic concept of markedness:

<2> I.e., there may be arbitrarily many lexicons with the label Adj.Stem, grouped according to e.g. declensions.
<3> Actually, it is unification split up into two separate steps: a compatibility check and, if this succeeds, unification with copying.

with optional constituents one may specify a DAG that is added to the structure being built only if the constituent is not included in the analysis. This mechanism is used for assigning the positive degree to adjectives in the lexicon in fig. 1; if no material is taken from the lexicon Adj.Comp during analysis, the analysed adjective gets the positive degree by default.

Like the variuos versions we have of the two-level system, this extension to its lexicon is written in PASCAL.

4. Future Plans

Tne preceding section gave a brief overview over the current status of the lexicon system. Among the things that have not yet been implemented, but are due for inclusion in the system in the near future are:

- Negative value specifications in the DAG:s, e.g. (Case:-Dat).
- Disjunction of values in the DAG:s, e.g. (Case:(Ack OR Gen)).
- Full regular expression capability in the morphotax specifications.

For handling the last two problems in the list in section 2, I am currently exploring the possibility of using a formalism that is reminiscent of and largely inspired by the ones used in autosegmental phonology (Goldsmith 1976; McCarthy 1981; 1982), or Aronoff's (1976) word formation rules (see fig. 2). The main idea in both these approaches (even though the details differ) could be interpreted as a kind of constraint equation for lexical entries. Reduplication would be handled in something like the following manner: With the minilexicon for stems that reduplicate would follow a template specifying the kind of reduplication and the conditions under which it applies:

```
TEMPLATE :(Tense:Perfect) =
    C V <C V #>
    1 2  1 2
```

The notation is a cross between McCarthy's and Aronoff's. The part within angle brackets is a condition on the stem: it should begin with a CV sequence. The expression as a whole states that this initial CV sequence may be reduplicated if the word form can be interpreted as perfect tense. These templates would act as filters, or constraint mechanisms, between the lexical and surface representations, quite independent of and in parallel to any morphophonological rules in the description. What the exact form and power of these templates would be is not entirely clear at the moment, but material on linguistic universals, like the data on the possible forms of reduplicative constructions collected by Moravcsik (1978), obviously has an important role to play here.

(46) Adjective Reduplication Rule (Aronoff 1976:77)

```
C V C V X§
1 2 3 4 5  -> 1 2 3 4 1 3 4 5
```

(17) a. CVCCVC (kattab) b. (McCarthy 1981:388)

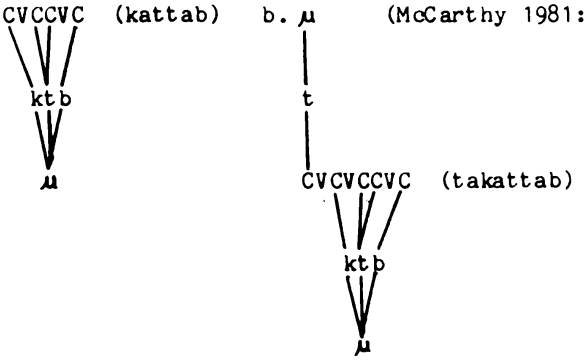


Figure 2.

The original motivation behind the development of autosegmental phonology was to account for phenomena like tone and accent, which are

relevant for the problem of accentually conditioned morphological alternations of the type cited last in the list in section 2.

5. New Problems

While the approach with feature-value graphs that are unified together solves some of the original problems in the list in section 2, they unfortunately introduce some new ones that, furthermore, are principally interesting.

One problem concerns word formation. Compounds of the type represented by the Finnish numerals can be handled very nicely with the proposed method, since the case agreement is enforced automatically in the lexicon. Some other kinds of word formation become very hard to handle in this framework, however. Category shifting affixes are prohibited at present, since a stem with e.g. (Cat:Noun) will not unify with a suffix that tries to assign (Cat:Verb) to the derived word form. If one subscribes to the view of e.g. Aronoff, that word formation is a process that operates on words to form new words, where the category of the word in question may or may not change in the process, there must be some further mechanism to account for this.

Another problem is more specifically connected with the framework this lexicon system was developed in, viz. two-level morphology. It is the problem of how the two-level rules should interact with the feature-value graphs. It is a characteristic trait of the two-level model that morphological features are segmentalized in the lexical representations (Nyman 1984, p 473), because this is the only way in which the two-level rules can access features that trigger some alternation, e.g. Tense:Perfect in the template in section 4 above. One could imagine that the morphological features, or rather, the DAG:s that contain them, be treated as lexical segments by the rules,

i.e. be accessible only at specific points in the lexical representation. On the other hand, one could try to restrict the domain of the rules, something that is needed on independent grounds: some alternation types comprise only specific word categories, e.g. Swedish apophony, which concerns only verbs.

6. Conclusion

In conclusion, I would like to stress that even if the things mentioned above are an important part of a lexical representation, they are still only a part of what is needed in a lexicon system for full-fledged natural language processing. In addition, the lexicon should reflect our knowledge as language users about productivity and prototypes in morphology. Certainly, the lexicon must also contain some structured semantic information.

References

Aronoff, M. 1976. Word Formation in Generative Grammar. Linguistic Inquiry Monograph 1. Cambridge, Mass. and London.

Blåberg, O. 1984. Svensk böjningsmorfologi - En tvånivåbeskrivning. Unpublished Master's Thesis. University of Helsinki, Dept. of General Linguistics.

Borin, L. 1985. Tvånivåmorfologi - Introduktion och användarhandledning. UCDL-L-3. Uppsala University, Center for Computational Linguistics.

Doherty, P., Rankin, I. & Wirén, M. 1986. Erfarenheter av en implementering av Hellbergs system för svensk morfologi. In this volume.

Goldsmith, J.A. 1976. Autosegmental Phonology. Indiana University Linguistics Club. Bloomington.

Hellberg, S. 1978. The Morphology of Present-Day Swedish. Stockholm.

Hockett, C.F. 1958 (1954). Two Models of Grammatical Description. In: Joos, M. (ed.). Readings in Linguistics. New York.

Karttunen, L. 1983. KIMMO: A General Morphological Processor. Texas Linguistic Forum 22. University of Texas at Austin, Dept. of Linguistics.

---------- 1984. Features and Values. Chapter 2 of Shieber, S., Karttunen, L. and Pereira, F.C.N. Notes from the Unification Underground. SRI International Technical Note 327. Menlo Park, Calif.

Koskenniemi, K. 1983. Two-level Morphology - A General Computational Model for Word-Form Recognition and Production. Publications No. 11. University of Helsinki, Dept. of General Linguistics.

McCarthy, J.J. 1981. A Prosodic Theory of Nonconcatenative Morphology. Linguistic Inquiry, Vol. 12, No. 3.

---------- 1982. Prosodic Templates, Morphemic Templates and Morphemic Tiers. In: Hulst/Smith (eds.). The Structure of Phonological Representations, Part I. Dordrecht - Cinnaminson.

Moravcsik, E. 1978. Reduplicative Constructions. In: Greenberg, J. (ed.). Universals of Human Language, Vol. 3, Word Structure.

Nyman, M. 1984. Sananmuotojen tunnistuksen ja tuoton mallintamista (Review of Koskenniemi 1983). Virittäjä 4/1984. Helsinki.

Lauri Carlson
University of Helsinki
Research Unit for Computational Linguistics

LP RULES IN UNIFICATION GRAMMAR

The purpose of this paper is to consider extension of unification based context free grammar with (a suitable generalization of) the IDLP formalism of Gazdar et al. (1985).

We define unification based context free grammar (UCFG) as a generalization of CF grammar. The shared property is the CF form of productions (a single nonterminal on the LHS of a rule). This allows use of appropriately modified CFG parsing algorithms. The HUG grammar formalism of Lauri Karttunen (paper presented in this conference) can be regarded as an instance of UCFG.

The structure of the present paper is as follows. In section 1, definitions are given to relevant types of grammar. Section 2 discusses generalization of IDLP grammar to unification. A parsing problem bound up with the generalization is described in section 2.2, which closes with a definition of unification IDLP grammar. Direct parsing of UIDLP grammar is discussed in section 2.3. The last section 2.4 proposes a generalization of the notion of a unification LP rule, aimed to avoid the cost of parsing full UIDLP grammar while retaining enough expressive power for the statement of common types of word order constraints.

1. Definitions

Standard notations of formal grammar theory are used (see e.g Hopcroft and Ullman (1979) for definitions). **in, +, and iff** are used for set theoretic membership, union and definitional

equivalence, respectively.

1.1. Definition of standard CF grammar

We recapitulate the definition of standard CF grammar to serve as a point of comparison.

G = <N,T,P,S>, N, T disjoint and finite, **P** a finite subset of **NxV*.**

u => w iff u = xAy and w = xUy for some A -> U in P, xy in V*.

w in L(A) iff A =>* w. L(G) = L(S).

Then the definition of UCFG can be approached as follows.

1.2. Definition of unification based CF grammar (UCFG)

G = (N,T,P,S) where **N = L(G')** for the following CFG:

G' = (N',T',P',Cat), T' = F + C + =),(,l,v,fail.

P' =
Cat -> fail
Cat -> Var, Var -> v; Varl
Cat -> c, c in C
Cat -> (f1=Cat f2=Cat ... fn=Cat), where f1,f2,...,fn = F

P a finite subset of **N x V*, S in N.**

The foremost difference here is that the set of nonterminal symbols of a UCFG is infinite (it is the language of another context free grammar).

Although only a finite number of nonterminals can occur in rules, a rule can match (unify with) an infinity of different nonterminals in the course of different derivations. In other

words, a UCFG rule can have an infinite number of rule instances.

To define the yield relation, we need to introduce a number of auxiliary concepts.

1.2.1. Substitutions

An **substitution** is a partial function **s** from **L(Var)** to **L(Cat)**. A substitution **s** is extended to V* by the clause **s(xy) = s(x)s(y)**.

A substitution which is one-one in **L(Var)** is a called a renaming of variables. We define a notational equivalence relation == among alphabetic variants so that **x==y** if **x = s(y)** for a renaming of variables **s**. == is extended to productions in the obvious way: **p = A -> U == B -> W** iff **AU == BW**.

A substitution **s** with values in **L(Var)** can be iterated. To define eventual values of **s**, we define **s*(x)** so that **s*(x) = x** if **s(x)** is undefined or $s^n(x) = x$ for some **n>0**; else **s*(x) = s*(s(x))**.

The smallest substitution is the empty substitution **0**. The inconsistent substitution **1** is such that **1(x) = fail** for any **x**.

1.2.2. Unification

A **unifier** for categories **A,B** is a substitution **s** s.t. **s(A) = s(B)**. **s** is a **maximally general unifier (mgu)** for **A,B**, or **s = mgu(A,B)**, if **u(s(AB)) = u(AB)** for any unifier **u** for **A,B**. It can be determined up to alphabetic variance using the following schema.

mgu(A,B) = u(A,B,0), where

u(c,c,0) = 0, c in C

u(v,B,0) = <v,B> if **v** in L(Var) does not occur in **B.**

u((f1=c1...fn=cn),(f1=d1...fn=dn),0) = u(c1,d1,u((f2=c2...fn=cn),(f2=d2...fn=dn),0)

u(A,B,s) = s + u(s*(A),s*(B),0)

otherwise **u(A,B,s) = 1.**

A unifies with B iff **mgu(A,B)** is not **1.** The **unification** of **A** and **B** is **mgu(A,B)'(A) = mgu(A,B)'(B). A subsumes B** iff **mgu(A,B)(A) == A.** That is, unification of **B** into **A** does not change **A.**

1.2.3. Derivability

With these basic concepts, we can define the UCFG yield relation as follows.

u => w iff **u = xAy** and **w = s(xWy)** where **B -> W == p** for some production **p in P, xy in V*** and substitution **s = mgu(A,B).**

The main differences compared to CF are that the matching relation between **A** and the left hand side of **p** is not identity but unification (a rule can match an infinite number of nonterminals consistent with it) and that the expansion of **A** with **p** can instantiate (rewrite) nonterminals in **u** outside **A.**

The == relation in the definition allows renaming of variables between repeated applications of a rule. We have chosen to rename the production **p.** Equivalently, we could rename **u.**

Example UCFG:

(cat=S num=v0) -> (cat=A num=v1) (cat=B num=v1) (cat=C num=v1)

(cat=v1 num=(num=v2)) -> (cat=v1 num=v2) (cat=v1 num=0)

(cat=A num=0) -> a

(cat=B num=0) -> b

(cat=C num=0) -> c

This UCFG generates the non-CF language $a^n b^n c^n$.

1.3. CF IDLP grammar

Standard rewriting rules contain dominance and precedence information connected together. The idea of IDLP grammars is to separate dominance from precedence. There are two types of rules: ID rules of form

A -> B1, ... , Bn

where **A** is a nonterminal and **B1, ... Bn** form a multiset (set with possible repetitions) of symbols in **V**, and LP rules of form

A < B

where **A** and **B** are nonterminals in **N**.

A CFIDLP grammar **H** can be defined as a pair **(G,<)** where **G** is a standard CF grammar and < is a strict partial ordering in **NxN**. For simplicity, we fix a standard ordering of the RHS's of ID rules which contains < as a subset. Henceforth we assume ID rules are normalized into standard order.

To define the yield relation, we proceed as follows. Let **L** be a

subset of **V***. We define **Sat(L,<)** as the set of those words **w** in **L** that are not of form **xByAz** where **A < B**. If **w** is in **Sat(V*,<)** we say that **w** satisfies <. A production **p** satisfies < iff its RHS satisfies <. We define a relation --> in **V*** so that **x --> y** iff **y** is a permutation of **x** and **y** satisfies <.

Then **u => w** in **H = (G,<) iff u = xAy, w = xUy,** and **A -> W** is in **P** st.t **W --> U.**

The novelty here is that a given production actually defines a set of ordered productions obtained by permuting its right hand side in LP acceptable ways.

Thus any IDLP grammar has an equivalent ordered grammar. Conversely, any ordered G has a trivial equivalent ILDPG. The conversions affect the nonterminal vocabulary and therewith the parse trees generated by the grammars.

In Gazdar et al. (1985:49) the strong generative capacity of IDLP grammars is characterized in terms of the ECPO (Equivalent Constant Partial Ordering) property. In an IDLPG with fixed nonterminal vocabulary, if A < B in the RHS of one production, A < B in the RHS of all productions. If and only if a ordered grammar has this property, it can be rewritten as an IDLP grammar without changing the nonterminal vocabulary.

Since it is easy to move from the extended vocabulary of a covering IDLP grammar to the original vocabulary of the ordered grammar in any given derivation (cf. Aho and Ullman72:275), the ECPO property is of slight practical interest. This is all the more true in UCFG, where the feature composition of the top category provides a more flexible description of sentence structure than derivation history.

1.3.2. Parsing of IDLP grammar

This section assumes standard notions of Earley parsing (see e.g. Aho and Ullman 1972).

Barton (1985) shows parsing of IDLP grammars NP complete in the worst case (when < is empty). The basic fact is that the number n! of permutations of a string of length n grows exponentially with n. The combinatorial explosion arises in cases of lexical ambiguity, where (say) a string of form **a1...an** can be parsed in n! ways by an IDLP grammar **H = (G,<)** with **G = (S -> A1...An, Ai -> aj for all i,j)** and < empty.

Though exponential in the worst case, direct parsing of IDLP grammars can show savings compared to parsing corresponding ordered grammars. The source of the savings is that the number of items in the parse table can be kept small by keeping together items that are represented separately in the ordered grammar. In the worst case, this gets the number of items down from O(/G/!) to $O(2^{/G/})$ (one item per subset of RHS symbols instead of one per permutation).

An Earley parse item **A -> x.y** is <--> to item **A -> z.w** iff **x** <--> **z** and **y** <--> **w**. To simplify the identification of equivalent items, they can be normalized to a fixed alphabetic order.

The test **yB** <--> **By** involves checking LP-acceptability of **By** This can be done by looping through all LP rules and pairs **B,C**, **C** in **y**. Since the result of the test depends only on **G**, it can be precomputed.

2. UIDLP grammar

2.1. Definition

Analogy with the CF case suggests the following definitions.

A UIDLP grammar is a pair **H = (G,<)** where **G** is a UCFG and **<** is a strict partial order on **V**.

Let **L** be a subset of **V***. **Sat(L,<) = (w in L: w** is not of form **xCyDz** where **A < B** and **CD** subsumes **BA)**. If **w** is in **Sat(V*,<)** we say that **w** satisfies **<**. A production **p** satisfies **<** iff its **RHS** satisfies **<**. **x --> y** iff **y** is a permutation of **x** and **y** satisfies **<**.

Then **u => w** in **H = (G,<)** iff **u = xAy** and **w = s(xWy)** where **s(W) <--> U** and **B -> U == p** for some production **p in P, xy in V*** and substitution **s = mgu(A,B)**.

This definition combines the UCFG and IDLP yield relations in the straightforward way. LP rules are used as local tests checking a permutation of the RHS of a rule when the rule is applied. Category identity as the matching relation is generalized to subsumption.

2.2. Nonlocal subsumption problem

It follows from these definitions that the LP-acceptable permutations of an instance of a UID rule can be a proper subset of the permutations of the original rule. The reason is that in general, **Sat(L,<)** is a subset of **Sat(s(L),<)** but not vice versa. Instantiation can make LP rules applicable which do not apply to the uninstantiated rule.

In other words, in UIDLP grammar, word order can be sensitive to context. For example, the order of a V and its complements may depend on the character of the clause they belong to (German, Finnish). This cannot be decided locally by looking at the verb and its complements. This situation is exemplified in the following UIDLP grammar. (CF category symbols indexed with feature equations serve as shorthands for UCFG category symbols.)

S' -> S(type=main)
S' -> Comp S(type=sub)
S(type=x) -> NP VP(type=x)
VP(type=x) -> V(type=x) NP
Comp -> dass
NP -> Jungen
V -> sind
V(type = main) < NP
NP < V(type=sub)

This grammar left generates the sentences Jungen sind Jungen and dass Jungen Jungen sind (but not the illicit orders).

However, the same grammar parsed bottom up right to left accepts the illicit sentence dass Jungen sind Jungen:

```
S'
Comp          S(type=sub)
              S(type=x)
              NP      VP(type=x)          - passes LP rules
                      V(type=x) NP
dass          Jungen  sind      Jungen
```

A similar paradox can be constructed between left and right top down derivations of a variant grammar with rules **S' -> Comp(type=x) S (type=x), Comp(type=main) -> e, Comp(type=sub) -> dass.**

This result is undesirable in that it imports an order dependent, procedural feature to an otherwise declarative formalism. The LP-acceptability of a sentence can depend on the order applying the rules of grammar, so that certain parsing strategies can pass sentences which fail LP-rules on the surface.

As far as the definition of derivability is concerned, what we should have said to begin with is fairly clear. We want LP rules to regulate the order of sister constituents at all stages of

derivation (in particular, at the end of derivations). This can be stated as a global condition on derivations, or, as is actually done in GPSG (Gazdar et al. 1985:46), by interpreting LP conditions as node admissibility conditions. In this interpretation, an LP-rule **A < B** reads:

(LP) A pair of nodes **B'A'** subsuming **BA** cannot appear as sisters in a derivation tree.

Given that LP-acceptability is taken care of by (LP), we can simplify the UIDLPG yield relation as follows:

u => w in **H = (G,<)** iff **u = xAy** and **w = s(xWy)** where **s(W)** is a permutation of **U** and **B -> U == p** for some production **p in P, xy in V*** and substitution **s = mgu(A,B)**.

2.3. Parsing of UIDLP grammars

One way to parse UIDLP as revised above is to simply apply LP rules in the completed parse so as to filter out LP-inconsistent parses.

This is conceptually straightforward but inefficient. All permutations would be considered only to be discarded at the final step. We should bring LP constraints to bear as soon as possible, i.e. apply LP rules as global constraints on derivations.

Consider again the illicit parse above. Although **V** and its complement **NP** do not subsume the LP rule **NP < V(type=sub)**, they do unify with the rule. (If they did not unify with the rule, they could not subsume it later either.) Such undecided applications of LP rules should remain active until a decision can be made. (Again, the decision should be done as early as possible to cut off parses.) Let us say that in such cases, the LP rule **properly unifies** with the pair of categories.

Assume **B < C** properly unifies with **B', C'** when item **A -> xC'.yB'z** is formed. We need to save the active constraint with the undecided item in such a way that it will be reapplied to instances of the original undecided pair.

To do so, we associate with each undecided item with a list of constraints of the following kind. A constraint **c** is of form **"A < B to B'A'"**. It is **matched** if **A'B'** subsumes **AB**, **irrelevant** if **A'B'** does not unify with **AB**, and **active** otherwise. If **s** is an substitution, **s(c) = "A < B to s(B'A')"**.

Whenever a new item **i = s(A -> xC'.yz)** is to be combined from items **j = A -> x.yC'w** and **k = C" -> u.** using mgu **s**, we check each constraint on the constraint list **(s(c): c is on the constraint list of i or j or is of form "B < C to C'B'", B' in y and B < C a LP rule.)** If **c** is matched, reject the combination. If **c** is irrelevant, delete the constraint from the constraint list. Finally, if the combination is not rejected, assign the remaining constraints as the constraint list of the new item.

The above procedure is rather cumbersome. What is more, it is difficult to find cases in actual languages where it is really needed. The types of context sensitive word order constraints I have found allow for a simpler fix which is sketched in the following section.

2.4. Generalized LP rules

A UCFG category can be represented by a set of feature equations, specifying values of features instantiated in each category. Conversely, the set constitutes the least solution of its representing equations in the domain of UCFG categories.

A pair of UCFG categories can be similary represented as a higher order category with attributes 0,1 for the members of the pair. Feature equations can then be used to describe category pairs. (The idea is adapted from Karttunen (this volume).)

The standard interpretation of a unification LP rule **A < B** can be restated as follows.

(1) If **A'B'** subsumes **AB**, then **A' < B'**.

The contraposition of this constraint (given **A'** is not **B'**) is

(2) If **A' > B'**, then not: **A'B'** subsumes **AB**.

If the complement **c(AB)** of the specification **AB** can be expressed, the contrapositive constraint can be further rewritten as

(3) If **A' > B'** then **A'B'** unifies with **c(AB)**.

Then (3) could be directly verified by unifying **c(AB)** into A'B' whenever the antecedent of (3) is taken. This suffices to guarantee satisfaction of the consequent of (3), in fact strengthens it to subsumption.

In our example grammar, since **(type=sub) = c(type=main)**, application of **NP < V(type=sub)** to the pair **V(type=x) NP** could simply instantiate **V** into **V(type=main)**.

These observations suggest the following generalization of the notion of a LP rule.

(4) Assume **x > y**. Then if **xy** subsumes **AB** then **xy** subsumes **CD**.

Or equivalently,

(5) **x < y** if **xy** subsumes **AB**, else **xy** subsumes **CD**.

Let us consider some special cases of this general form. The original rule **A < B** is obtained as

(6) **x < y** if **xy** subsumes **AB** else **xy** subsumes **fail.**

Our example of nonlocal subsumption is taken care of by the pair of rules

(7) Assume **x > y.** Then if **xy** subsumes **((x cat) = V)(y cat) = NP))** then **xy** subsumes **((x type) = sub)**

(8) Assume **x > y.** Then if **xy** subsumes **((x cat) = NP)(y cat) = V))** then **xy** subsumes **((x type) = main)**

These rules instantiate the main and subordinate clause features at the time when the VP order is fixed. (An abbreviation of complementary rules such as (7)-(8) into one rule seems appropriate.)

The next example is a rule of functional word order interpretation.

(9) **x < y** if **xy** subsumes **(x = (y subject))** else **xy** subsumes **((x function) = rheme)**

This rule says that a subject following the main verb is rhematic.

As suggested before, generalized LP rules of form (4) (respectively, (5)) can be implemented in parsing so that the consequent (else) condition of the rule is actually unified in when the order condition of the rule is applicable (violated).

An implementation of generalized LP rules of this kind into HUG is in progress.

It should be kept in mind that generalized LP rules do not solve the nonlocal subsumption problem. At best, they make it possible to avoid the problem by allowing statement of some context sensitive word order rules without reference to nonlocal

conditions.

References

Aho, A. and J. Ullman (1972), **The Theory of Parsing, Translation, and Compiling. Volume 1: Parsing.** Prentice-Hall.

Barton, E. (1985), "The Computational Difficulty of ID/LP Parsing", in **Proceedings of the 23rd Annual Meeting of the ACL,** Chicago.

Gazdar, G, E. Klein, G. Pullum and I. Sag (1985), **Generalized Phrase Structure Grammar.** Harvard University Press.

Hopcroft, J. and J. Ullman (1979), **Introduction to Automata Theory, Languages, and Computation.** Addison-Wesley.

Shieber, S. (1984), "Direct Parsing of ID/LP Grammars", **Linguistics and Philosophy 7,** 135-154.

Östen Dahl
Univ of Stockholm

THE INTERPRETATION OF BOUND PRONOUNS

This paper is a report on work in progress with the aim of simulating on a computer some aspects of the process of understanding sentences, more specifically the interpretation of so-called **bound pronouns.** The work connects directly to some earlier papers of mine (Dahl 1983a and 1983b), where those problems were discussed from a theoretical point of view.

A bound pronoun is, roughly speaking, a pronoun which behaves analogously to a bound variable in logic. There are at least two kinds of criteria for regarding a pronoun as bound: (i) syntactic criteria - some kinds of pronouns, such as reflexives, reciprocals and so-called logophorics, must find their antecedents in syntactically defined domains, (ii) semantic criteria - some pronouns cannot be assigned referents 'in the world' but can be understood only if regarded as referentially dependent on their antecedents: this concerns e.g. pronouns bound by quantified NPs and wh-phrases (e.g. himself in Nobody likes himself). Although the classes of pronouns delimited by these criteria are not quite identical, they overlap to such an extent that in most cases, they can be regarded as equivalent.

In my earlier papers, I have discussed some cases of bound pronouns which are troublesome for the current theories that account for bound pronouns by translating them into some kind of logical notation using bound variable or equivalent devices. Those cases include:

(i) 'sloppy identity' cases (as in John loves his wife and so does Bill, where Bill may be understood to love either his own or John's wife)

(ii) 'dislocated bound pronouns', that is pronouns that have

been 'moved' (presupposing a transformational analysis) out of the scope of their binders, e.g. Himself, everyone despises, and in particular among those

(iii) pronouns with 'relational' (Engdahl 1985) or 'second-order' readings, as in a sentence such as The only woman every Englishman admires is his mother, the interpretation of which cannot be rendered without having recourse to second-order logic

The main idea put forward in my papers was that the troublesome cases could be accounted for if the location of the antecedent in the syntactic structure were considered an integral part of a bound pronoun's interpretation.

So far, two main versions of the pronoun interpretation program have been developed. In Dahl 1985, I report an attempt to construct a syntactic parser to be used on an 8-bit microcomputer, written in the LISP dialect muLISP. The first version of the pronoun interpretation program (henceforth 'Version 1') used a somewhat more elaborate version of this parser as its base, that is, the semantic part of the program took syntactically analysed sentences as its input and assigned referents to the noun phrases in them. In addition, the program had what can be called rudimentary conversational competence: the sentences processed were compared with a database and depending on the type of sentence, the appropriate action was taken: in the case of interrogative sentences, an answer was given, in the case of a declarative sentence, the proposition was added to the database. The reference assignment process in Version 1 worked in an top-down fashion, assigning referents first to the highest NPs in the syntactic structure. Extensive use was made of temporary registers, where processed NPs (with identified referents) were stored so as to be retrieved later on when needed as antecedents of pronouns. When an antecedent was found for a pronoun, both the referent and the location (that is, where it was found in the registers) of the antecedent was stored on the property list of the pronoun. This information was then exploited by the mechanisms used in the

'troublesome cases' listed above. Version 1 was thus able to handle both sloppy identity and at least some 'relational questions', e.g. the following:

(1) Whom does every man love, his wife or Mary?

However, Version 1 was rather slow, with processing times up to half a minute for processing some sentences (this would include both syntactic parsing, reference assignment, comparison with the database and appropriate reaction). There were several reasons for that, including inherent limitations in the hardware and software used. Of a more direct linguistic relevance, however, were the following circumstances: The syntactic and semantic components of the systems were wholly autonomous from each other, and indeed worked in rather different fashions: the syntactic parser was strictly bottom-up, systematically taking into considerations all possible analyses of the sentences, whereas the semantics, as has already been pointed out, worked from the top down, and with the principle of always choosing the first possible alternative. When I considered the slowness of Version 1 and also realized that what the semantic part of it did was largely repeating the syntactic analysis of the sentence, it appeared to me that it might be fruitful to try and build a system where syntactic and semantic analysis would be done in an integrated fashion. This, however, put stronger demands on the parsing mechanism, since it required a more intelligent way of handling structural ambiguities.

Version 2, then, has been designed to meet these demands. It is written in the MS-DOS version of muLISP and has been run on several kinds of IBM compatible computers. It has not yet been developed as fully as Version 1 (the 'conversational' part has not been implemented, for instance) but its performance is significantly better than that of Version 1, partly due to better hardware but also due to a more efficient structure of the parsing mechanism. Thus, the parsing time (including reference assignment) is about 20 milliseconds per word on an IBM AT computer. Perhaps the main advantage is that this time

is more or less linear, whereas the parsing time per word in Version 1 grew very rapidly with the length of sentences.

The syntactic analysis in Version 2 is done according to the following principles:

(i) the output is a LISP structure which can be characterized as an 'almost unlabelled bracketing', that is, with very few exceptions, the syntactic category of a constituent (which has the form of a list) is not explicitly marked but has to be deduced from the lexical category of its 'head', that is the first member (CAR) of the list

(ii) parsing is done from left to right in a more or less deterministic way

(iii) the fact that the category of a constituent is in general known when you have identified its head or its first word (which is often the same thing) is systematically exploited in predicting what comes next

(iv) backtracking is made by a systematic use of local parameters of LISP functions: every time a new word is parsed a call is made to a function and the partial structure built so far is passed to that function as a parameter - if the continued parse does not succeed, one automatically returns to the previous state

(v) at any point in the parsing process, the partial analysis arrived at so far is represented as a single stack of 'active constituents' (called the ACTIVESTACK), that is, constituents that have not yet been finished. To show what the parsing of a sentence may look like, we show the successive stages of the parsing of (2) in (3).

(2) John believes that Mary loves Bill

(3)

(expression to be parsed:)	(ACTIVESTACK:)
1: **John believes that Mary loves Bill**	NIL
2: **believes that Mary loves Bill**	((John) VP (S))
3: **that Mary loves Bill**	(NP (believe -s) (S (John)))
4: **Mary loves Bill**	(S (that) (believe -s) (S (John)))
5: **loves Bill**	((Mary) VP (S) (that)(believe -s) (S (John)))
6: **Bill**	(NP (love -s) (S (Mary)) (that) (believe -s) (S (John)))
7: **NIL**	((Bill) (love -s) (S (Mary)) (that) (believe -s) (S (John)))

(close all constituents)

(S (John) (believe -s (that (S (Mary) (love -s (Bill))))))

The assignment of referents to NPs is done during the syntactic analysis, more specifically, when the noun phrase in question is 'closed', i.e. moved off the stack of active constituents. When a referent is assigned to a noun phrase, a 'dotted pair' representing the referent is added to the list which represents the constituent in the structure. At present, the system can handle three kinds of NPs: proper names, bound pronouns, and NPs with a possessive in the determiner slot. For proper nouns, the assignment process is trivial: the proper name itself is used as a reference indicator. Thus, the LISP expression to the left of the arrow is converted into the one to the right of the arrow:

(4) (John) ----> (John (REF. John))

Some people may be disturbed by this rather vacuous process: the point here is that since we are not directly concerned with how proper names are interpreted we do not want to introduce any complications here. Of course, we could easily plug in a

routine that puts in a referential index or the like.

For NPs with a possessive determiner, the principle is also slightly ad hoc: the referent of the possessive expression is first determined, then the property list of that referent is examined to see if there is some property which coincides with the head noun of the NP: in that case, the value of that property becomes the referent of the whole NP. For instance, if we have the NP John's wife and we find the item (wife.Mary) on John's property list, then the referent of John's wife is taken to be Mary.

The most interesting part of the referent assignment procedure is that which assigns antecedents and referents to bound pronouns. The assumption is that the antecedent of a bound pronoun is to be found among the NPs that c-command it. According to the current definition, a node x c-commands a node y if and only if the node that immediately dominates x also dominates y. In the present system, the c-commanders of an NP that is being 'closed' are always precisely those NPs that are immediate constituents of the members of the ACTIVESTACK. For instance, when the NP Bill in (2) above is closed, the ACTIVESTACK looks as follows:

(5) (love -s) (S (Mary)) (that) (believe -s) (S (John)))

The c-commanders in (5) are thus Mary and John.

This makes it possible to formulate a relatively simple algorithm for finding the possible antecedents. In addition to assigning a referent to a pronoun, the algorithm also stores the distance (in nodes) between the pronoun and its antecedent. The point of this will become clear later.

In muLISP formalism the main antecendent-finding function looks as follows (some irrelevant details have been left out):

(6)

```
(DEFUN ANTECEDENT (LAMBDA (X Y XNP XNODE DIST)
  (SETQ Y (CDR ACTIVESTACK))   Define Y as the ACTIVESTACK minus the NP
                                                    under consideration.
  (SETQ DIST 0)                                Set the variable DIST to 0.
  (LOOP                     Repeat until Y is empty or antecedent is
                                                                  found:
    ((NULL Y) NIL)
    (SETQ XNODE (POP Y))                  Set XNODE to next member of Y.
    (SETQ XNP (FIRSTNP XNODE)) Find the first NP in XNODE: call it XNP.
    ((AND
        (MEMBER (CAR X) REFLPROLIST)    If the pronoun is reflexive and
        (EQ (CAT XNODE) S) )                  XNODE is a sentence, then
      (PUT X 'ANTEC-DIST DIST)  set the antecedent-distance to DIST and
      (AGREE X XNP) )        the antecedent to XNP, if it agrees with
                                                    the pronoun, else to NIL,
    ((AND                              (if the pronoun is non-reflexive:)
        (AGREE X XNP)                 if XNP agrees with the pronoun then
        (NOT (AND                   unless the pronoun is non-possessive
         (NOT (POSSESSIVE X))                                        and
         (EQ (GET XNP REF) (GET (SUBJECT) REF)) ) )             XNP is
                                                     coreferent with the
                                                   subject of the sentence,
      (PUT X 'ANTEC-DIST DIST)then set the antecedent-distance to DIST
      XNP )                                        and the antecedent to XNP.
      XNP )
        (SETQ DIST (ADD1 DIST))  ) ) ))                  Add 1 to DIST.
```

This is certainly a simplified rule: for instance, it assumes that the antecedent of a reflexive is always the subject. However, in most simple cases, it assigns the closest possible antecedent to any bound pronoun.

Let us now have a closer look at the antecedent distance parameter. Its function is to define the location of the antecedent of a pronoun: this information is above all useful when the stored interpretation of the constituent which

contains the pronoun is retrieved later on. We shall illustrate what this means by looking at the way in which the program handles 'sloppy identity'. Consider the again the example from the beginning of the paper:

(7) John loves his wife and so does Bill

At present, the program is only able to handle a somewhat unidiomatic paraphrase of (7):

(8) John loves his wife and Bill too.

Basically, the following is what happens when (8) is interpreted by the system: First, the clause John loves his wife is parsed. Assuming that the system knows that Mary is John's wife, it will assign Mary as a referent to his wife. Then, the reduced clause Bill too is parsed. After the subject NP Bill the system expects a verb phrase: it takes the particle too as a signal of an elliptical VP. Every time a VP is parsed, it becomes the value of the variable LASTVP: in this case, LASTVP is loves his wife. The parsed version of this expression is now copied into the place where the VP should occur in the elliptical sentence. When this happens, the NP his wife is again subjected to the reference assignment process - however, since it is the second time, the antecedent is found not by the function ANTECEDENT but by another called FIND-ANTECEDENT-AGAIN. This function looks at the antecedent distance associated with the pronoun his and tries to find the NP at the corresponding place in the tree. In this case, it is Bill, so the referent of his wife is now taken to be Bill's wife.

Version 2 has not yet been developed so far that it can take care of the other problematic cases of bound pronouns, but in principle similar mechanisms as the one mentioned should be sufficient to solve the problems, as was demonstrated by Version 1. The point is that the antecedent distance parameter approach is inherently more powerful than the common way of displaying coreference relations, viz. by referential indices or multiple occurrences of the same variable letter, in that it

has a meaningful interpretation also out of context.

The above account has been lacking in explicitness in various ways. There are two reasons for this: the rather early stage of development of the program and the limited space available. The long-range aim of the undertaking is to provide a small yet powerful 'module' for processing natural languages sentences and texts, where the pronoun interpretation mechanism will only be a small part. Hopefully, the work on the 'module' will be possible to shed light on some questions of general theoretical interest.

REFERENCES

Dahl, Ö. 1983a. On the nature of bound pronouns. PILUS 48, Dept. of Linguistics, Univ. of Stockholm.

Dahl, Ö. 1983b. Bound pronouns in an integrated process model. In F. Karlsson, ed., Papers from the Seventh Scandinavian Conference of Linguistics. University of Helsinki, Dept. of General Linguistics.

Dahl, Ö. 1985. Syntactic Parsing on a Microcomputer. In S. Bäckman and G. Kjellmer, eds., Papers on Language and Literature presented to Alvar Ellegård and Erik Frykman. Gothenburg Studies in English 60. Göteborg: Acta Universitatis Gothoburgensis.

Engdahl, E. 1985. The Syntax and Semantics of Questions with Special Reference to Swedish. Dordrecht: Reidel.

Eva Ejerhed and Hank Bromley

A SELF-EXTENDING LEXICON: DESCRIPTION OF A WORD LEARNING PROGRAM

1. Introduction.

This paper describes the current implementation of the lexical analyzer (MORPH), of a finite state parser for Swedish (SWEDISH-SYNTAX). The MORPH program was developed by the authors in August 1985, and it is implemented in ZLISP on an LMI CADR Lisp machine. MORPH is a computational model of the recognition and analysis of written words, with the following brief characteristics:

MORPH is a word analyzer only, it does not synthesize or generate word forms.

- MORPH works left to right through a word, without pre-processing it in the form of suffix or prefix stripping.
- MORPH models both the process of word recognition/analysis and the process of word acquisition.
- MORPH works by combining the advantages of the two general methods for word recognition that are available: matching input words with words, and parsing input morphemes into words. By combining these methods, the disadvantages of using either method alone are avoided.

2. Considerations in designing MORPH.

As stated above, MORPH is a component of a larger system that is a parser for Swedish. At the Fourth Meeting of Scandinavian Computational Linguistics 1983 in Uppsala, I gave an overview of SWEDISH-SYNTAX, its organization and mode of operation (se also Church 1982, Ejerhed & Church 1983, and Ejerhed 1985). Since then, the work in Umeå on this system has been devoted to three things: (i) increasing the range of the syntactic constructions parsable by SWEDISH-SYNTAX by extending the grammar; (ii) adding a semantics component that assigns function-argument structures to a surface sentence on a given syntactic analysis of it; and (iii) increasing the number of words parsable by SWEDISH-SYNTAX.

The version of SWEDISH-SYNTAX that I described in 1983 had a full form lexicon which listed each inflected form' of a word along with its part of speech and morpho-syntactic features, e.g.

```
(defslex flicka noun +utr -pl -def)
(defslex flickan noun +utr -pl +def)
(defslex flickor noun +utr +pl -def)
(defslex flickorna noun +utr +pl +def)
```

The parser uses the part of speech information of a word in deciding constitutent structure for an input string, and it uses feature information when checking that agreement constraints are satisfied. Agreement in gender, number and definiteness between adjacent words in a noun phrase was and still is handled in SWEDISH-SYNTAX by a mechanism that filters out any combinations of non-agreeing words,like

```
* ett      vacker     flickan
 -utr      +utr       +utr
 -pl       -pl        -pl
 -def      -def       +def
```

This ensures that all noun phrases that are well formed with respect to constituent structure constraints (e.g. NP -- DET ADJECTIVE* NOUN), are also well formed with respect to agreement constraints. The way this works in SWEDISH-SYNTAX is not technically by unification, but by imposing the requirement that for immediate constituents of a noun phrase in Swedish no conflicts of values are allowed for the features of gender, number and definiteness. In the example above, there is both a gender conflict and a definiteness conflict For a more detailed technical description of the general properties of the agreement mechanism, see Church 1983, and for its application to Swedish syntax, see Ejerhed 1983.

We wanted to increase by a large amount the number of words that could be successfully dealt with by SWEDISH-SYNTAX, because a very large, and preferably self-extending, lexicon is necessary for most imaginable applications of natural language processing systems, such as natural language interfaces to data bases, or automatic information gathering (knowledge acquisition).

In designing MORPH, there were considerations of a theoretical as well as a practical nature. They were the following.

First, we wanted to test the hypothesis, that the proper role, and the only role, of morphology in word perception is in the processing and learning of unknown words. Unknown word processing proceeds by "decomposition first, retrieval second", to use the theoretical terminology of Job & Sartori 1984 in characterizing a possible model of word processing. The processing of known words, by contrast and on our hypothesis", proceeds by retrieval of the entire word and its associated properties (part of speech, features, and morphological decomposition) from a list ("retrieval first, decomposition second"). The source of this hypothesis, and the hybrid model of word processing, was research on speech synthesis (Byrd & Chodorow 1985. Church 1985). Associating pronunciations with written words is generally considered to be mediated by two processes, one that retrieves the stored pronunciation of a word, another that derives it from general letter to sound rules. Church 1985 gives a good argument why the task of associating pronunciations with words cannot be reduced entirely to either listing them or deriving them:

"Both approaches have their advantages and disadvantages; the dictionary approach fails for unknown words (e.g. proper names) and the letter to sound approach fails when the word does not follow the rules, which happens all too often in English. Most speech synthesizers adopt a hybrid strategy, using the dictionary when appropriate and letter to sound for the rest."

Translating this to the domain of syntactic and semantic analysis, we observed that the approach of listing full words will also fail for unknown words (e.g. the very large numbers of Swedish compounds, which are written as single graph words), and the approach of parsing all words will fail because of overrecognition (cf. the contribution of Doherty, Rankin, & Wirén to this conference). A hybrid model therefore seemed worth investigating, as an alternative to the currently popular full listing models in psycholinguistics and

computational linguistics (e.g. Wehrli 1985). For an overview of psycholinguistic work on morphology, see Henderson 1985, and for a set of recent studies of morphological constraints on word recognition, see Jarvella, Job, Sandström & Schreuder (forthcoming).

Second, we wanted a psycholinguistically and computationally plausible model of how words are entered into the lexicon of known words of a language user, as a function of language use. Most existing computational models of natural language deal with the processing of sentences relative to a fixed body of language knowledge, rather than with the acquisition of that knowledge. People acquire new knowledge about a language as part of the process of using it, and we have to try to simulate at least some aspects of language learning behavior on computers, it we want to shed further light on learning by people, and if we want to make computers more useful, languagewise.

Third, we wanted a unidirectional model of word recognition only, and not a bidirectional model of both word recognition and generation. There was a performance theoretic reason for this, as well as a practical one. In studies of language processing (e.g. Deutsch & Jarvella 1984), tasks involving perception and tasks involving production of the same linguistic structures have shown interesting differences, which undermines the idea that it is exactly the same knowledge of language (linguistic competence) that is at work in both performance processes. The practical consideration in modeling only word analysis was that it minimized the problem.

Fourth, and another practical consideration, was the limited time available for designing, implementing and testing the new lexical component, a total of four weeks.

Fifth, and last, the new lexical component had to provide information about words in a way that other components of SWEDISH-SYNTAX could use. In order for the lexical lookup routines to be of any use to the parser when presented with a written word, they have to serve the parser with the word plus its part of speech and features. In the case of ambiguous words, they have to hand the parser all of the analyses of the ambiguous word plus the information associated with each analysis of it. This meant that the names for parts of speech

and morpho syntactic features were already established, and prevented us from using directly any pre-exisiting morphological analyzer, such as Blaberg's two-level morphology for Swedish (Bläberg 1984), or Hellberg's Swedish morphology in the implementation of Doherty, Rankin & Wirén (this volume).

3. Description of the morphological analyzer.

3.1. Overview.

The two lexicon model of word processing that we arrived at has the following technical characteristics. There are two lexica, one of full words, called **word-lexicon**, and one of morphemes, called **morpheme-lexicon**. We will describe the morpheme-lexicon first.

3.2. The morpheme-lexicon.

The morpheme-lexicon is a lexicon of known morphemes, corresponding to the morphological decompositions of words given in the word-lexicon. The morpheme lexicon is a **hash table** (association list) with morphemes as **keys** and features as **values**. The advantage of this representation over other representations (discrimination networks, or treating words as Lisp atoms) is that the hash table enables a group of values to be associated with a single key and retrieved all at once. Since morpheme and word ambiguity is common, that is useful. Further, the morpheme-lexicon is sorted by values, meaning that morphemes with the same features are grouped together. The following is an example of the structure of entries in the morpheme-lexicon (note that "flick" here is a morpheme, not what Hellberg 1978 and others (Källgren 1986) have called a technical stem):

KEY	VALUE			
"flick"	(*	N	-WF	+UTR)
	1	2	3	4
	COMBINATORIAL FEATURES		BINARY FEATURES	

We will now describe the combinatorial features and the binary features of a morpheme. The former encode the combinatorial possibilities of the morpheme, and the latter are strictly binary features.

Position 1. The first combinatorial feature encodes what part of speech the morpheme combines with on its left, what it takes. The options are:

*	morpheme takes any part of speech on its left
NIL	morpheme takes nothing on its left
NOUN	morpheme takes specified part of speech on its left
VERB	
ADJ	
PREP	
ADV	
...	

Position 2. The second combinatorial feature encodes what part of speech a morpheme makes. The options are:

NOUN
VERB
ADJ
PREP
ADV
...

Position 3. The third combinatorial feature encodes whether or not the morpheme can occur in word final position. There are just two options:

-WF	morpheme combines with something on the right, i.e. it is non word final
+WF	morpheme combines with nothing on the right, i.e. it is word final

The word-finality feature enabled us to do morphological analysis without zero morphemes, which seemed desirable. An example of how it is used is the following:
sko / __ #(w b) has features (* NOUN +WF +UTR -PL -DEF)
sko / __ +(m b) has features (* NOUN -WF +UTR) and no more. It will get the features for number and definiteness from what it combines with, e.g. **sko+n, sko+r, sko+r+na.**

Position 4. Binary features. For the noun, adjective and verb system, they are:

+UTR	+AUX
-UTR	-AUX
+PL	+REAL
-PL	-REAL
+DEF	+FIN
-DEF	-FIN
	+PRS
	-PRS

The binary features are strictly binary, i.e. the complement of +UTR is -UTR, etc. Binary features are always fully specified, thus there are no redundancy rules that mediate between partially and fully specified feature values (the absence of a specification for a feature means that the item can pass the requirement of both the value + and the value - for that feature). There were two reasons for this, one being that it is doubtful that redundancy rules are of any practical use in a processing model, the other being that the absence of redundancy rules facilitates debugging and extending the lexicon. At any point of working with it, the feature specifications you see is what the processor gets, too. Below are samples of groups of entries of lexical morphemes and partial entries of some grammatical morphemes.

```
"flick"   ( *  N   -WF   +UTR)
"gryt"    ( *  N   -WF   +UTR)
"klock"   ( *  N   -WF   +UTR)
"pojk"    ( *  N   -WF   +UTR)
"drull"   ( *  N   -WF   +UTR)
"mån"     ( *  N   -WF   +UTR)
"himl"    ( *  N   -WF   +UTR)
```

```
"a" ( (A  A  +WF  +UTR  +PL  +DEF)   a / de söt+a tå+r+na
      (A  A  +WF  +UTR  +PL  -DEF)   a / söt+a tå+r
      (A  A  +WF  +UTR  -PL  +DEF)   a / den söt+a tå+n
      (A  A  +WF  -UTR  +PL  +DEF)   a / de söt+a bi+n+a
      (A  A  +WF  -UTR  +PL  -DEF)   a / söt+a bi+n
      (A  A  +WF  -UTR  -PL  +DEF)   a / det söt+a bi+et
      (N  N  -WF  +UTR  -PL)         a / flick+a+n
```

```
      (N  N  +WF  +UTR  -PL  -DEF)    a / flick+a
      (N  N  +WF  -UTR  +PL  +DEF)    a / bi+n+a
      (V  V  +WF  -AUX +REAL -FIN +PRS)  a / verk+a (inf)
      (V  V  +WF  -AUX -REAL) )       a / verk+a (impv)

"n" ( (N  N  +WF  +UTR  -PL  +DEF)    n / flick+a+n
      (N  N  +WF  -UTR  +PL  -DEF)    n / bi+n

"or" ( (N  N  -WF  +UTR  +PL)         or / flick+or+na
       (N  N  +WF  +UTR  +PL -DEF) ) or / flick+or

"na" ((V  A  +WF  +UTR  +PL  +DEF)    na / de druck+na ko+r+na
      (V  A  +WF  +UTR  +PL  -DEF)    na / druck+na ko+r
      (V  A  +WF  +UTR  -PL  +DEF)    na / den druck+na ko+n
      (V  A  +WF  -UTR  +PL  +DEF)    na / de druck+na bi+n+a
      (V  A  +WF  -UTR  +PL  -DEF)    na / druck+na bi+n
      (V  A  +WF  -UTR  -PL  +DEF)    na / det druck+na bi+et
      (N  N  +WF  +UTR  +PL  +DEF))   na / flick+or+na
```

The general approach to specifying word structure in MORPH is exceedingly simple. Word = morpheme* minus those sequences of morphemes ruled out by the morphotactic constraints encoded in the combinatorial features. That those features go a long way towards specifying admissible sequences in real cases is illustrated by the following example of three morphemes and their specifications:

```
"av"   ( (NIL PREP -WF)          av / av+led+a
         (NIL PREP +WF) )        av / av flick+a+n
"led"  ( (*   V   -WF)           led / led+er
                                       av+led+er
                                       bransch+led+ande
         (*   V   +WF -REAL) )   led / led!
"ning"( (V   N   -WF)            ning / led+ning+en
        (V   N   +WF +UTR -PL -DEF) ) ning / led+ning
```

Given this set of three morphemes **av, led, ning,** the maximal number of theoretically possible words (up to trimorphemic words) made out of them are: $3 + 3^2 + 3^3 = 39$.

3 monomorphemic words:

OK av, OK led, *ning

9 bimorphemic words:

*avav	*ledav	*ningav
OK avled	*ledled	*ningled
*avning	OK ledning	*ningning

27 trimorphemic words:

*avavav	*avledav	*avningav
*avavled	*avledled	*avningled
*avavning	OK avledning	*avningning
*ledavav	*ledledav	*ledningav
*ledavled	*ledledled	*ledningled
*ledavning	*ledledning	*ledningning
*ningavav	*ningledav	*ningningav
*ningavled	*ningledled	*ningningled
*ningavning	*ningledning	*ningningning

The specifications given for the three morphemes in each instance makes the correct prediction whether the word is admissible or inadmissible, which suggests that the framework is adequate for capturing such classical notions of morphology as prefix, root, and suffix, and the distinction between free and bound morphemes. However, we do not regard the framework for specifying admissible sequences as a complete and definitive proposal. It will have to undergo revisions in the amount and nature of morphological structure that it attributes to words.

3.3 The word-lexicon.

The word-lexicon is a lexicon of known words. It is also a hash table, with words as keys and their parts of speech, feature properties and morphological decomposition as values. It is sorted alphabetically. In the current version there is no semantics attached to either morphemes or words, but we plan to introduce this in the following way. Each morpheme will always have semantic information associated with it. Each word that has an idiosyncratic and non- compositional semantics will have that semantics associated with the word as a whole. However, most composite words have a straightforward compositional semantics, and for those there will be no listing in the word-lexicon of the semantics for the word as a whole. For such words, the semantics will be retrieved from each constituent morpheme via the morphological decomposition

(in accordance with a hypothesis of word perception of Caramazza et al 1985).
Below are examples of the structure of entries in the word-lexicon.

```
"flicka" (((NOUN +WF -PL -DEF +UTR)) ("flick" "a"))
"flickan" (((NOUN +WF +DEF -PL +UTR)) ("flick" "a" "n"))
"flickor" (((NOUN +WF +PL -DEF +UTR)) ("flick" "or"))
"flickorna" (((NOUN +WF +DEF +PL +UTR)) ("flick" "or" "na"))
```

These are copied with one minor change from the current entries in the word-lexicon, and they reveal an unnecessary feature of each entry, +WF. Since every word, by definition is word final, this feature should be removed. The entries above are very simple and do not illustrate what the entries of words with ambiguous analyses look like. Below are two examples.

```
"betalade" (((VERB +WF -AUX -PRS +FIN +REAL)
             ("be" "tal" "ade"))
            ((ADJECTIVE +WF +UTR +PL +DEF)
             (ADJECTIVE +WF +UTR +PL -DEF)
             (ADJECTIVE +WF +UTR -PL +DEF)
             (ADJECTIVE +WF -UTR +PL +DEF)
             (ADJECTIVE +WF -UTR +PL -DEF)
             (ADJECTIVE +WF -UTR -PL +DEF)
             ("be" "tal" "a" "d" "e")))

"bildrulle" (((NOUN +WF -PL -DEF +UTR) ("bil" "drull" "e"))
             ((NOUN +WF -PL -DEF +UTR) ("bild" "rull" "e")))
```

4. Using MORPH.

The central function in the MORPH program is called lookup-word, and it has the following easy to understand definition:

```
(defun lookup-word  (word)
 (cond ((lookup-word-in-lexicon word))
       ((analyze-word-and-add-to-lexicon word))
       ((ask-about-word-and-add-to-lexicon word)) ))
```

The function lookup-word, when applied to a word, first tries to find it in the word-lexicon, and if successful, returns its values as its value. If unsuccessful, it invokes the function analyze-word-and-add-to-lexicon. This function analyzes the word character by character from left to right and finds all

internall, consistent morphological decompositions of it, relative to the current morpheme-lexicon. These analyses are returned as the value of the function, and the result incorporated in the word-lexicon. If the function analyze-word-... is unsuccessful, the program asks the user to supply information about the word, as a last resort. Several utilities for the manual extension of the lexica are included in MORPH, in order to minimize the amount of typing done. In that sense it is a good example of a lexicon writer's workbench. For example, if a new morpheme is exactly like a previously known one, with respect to part of speech and features (and degree and nature of lexical ambiguity), then this information is automatically copied when typing the morpheme it is like, at the appropriate point in the interactive sequence.

As intended, MORPH has replaced the old lexicon of SWEDISH-SYNTAX, and the interaction works out well. For example, in the case of an ambigous word like "betalade" above, the new lexicon hands the parser all of the analyses, and the parser picks the appropriate one according to the context in which the word occurs. The following tree diagrams produced by SWEDISH-SYNTAX illustrate the interaction between the lexicon and the parser.

Fig. 1 Analys av
Hon betalade boken.

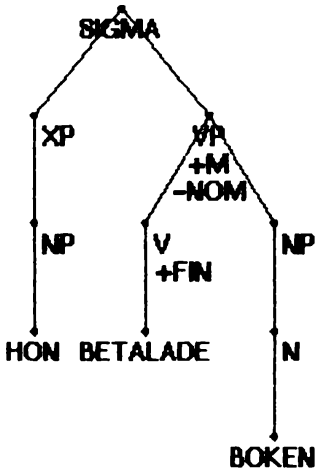
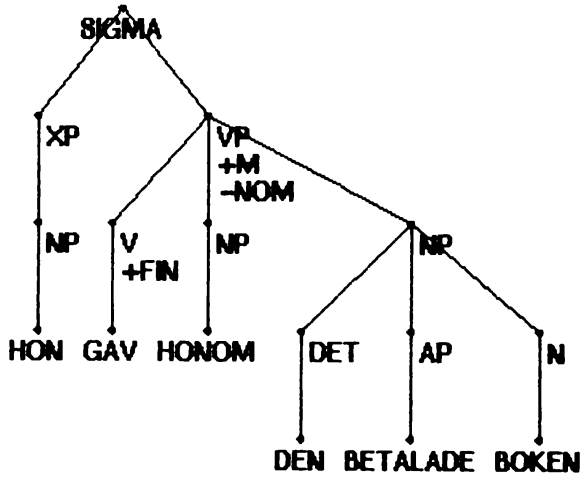


Fig. 2 Analys av
Hon gav honom den betalade boken.



Finally, we would like to present some preliminary figures that show how long it takes for the program to do lookup-word by lookup-word-in-lexicon (R = retrieval) on the one hand, and by analyze-word-and-add-to-lexicon (A = analysis) on the other hand. The words we tested are presented in descending order of morphological complexity, and the last word is an example of an ambiguously decomposable word.

#kyrk+klock+s+in+vig+ning#	A	0.219 sec
	R	0.004 sec
#bibliotek+s+in+vig+ning#	A	0.215 sec
	R	0.004 sec
#industri+av+veckl+ing#	A	0.313 sec
	R	0.004 sec
#ut+veckl+ing#	A	0.089 sec
	R	0.005 sec
#barn+cykel+nyckel#	A	0.223 sec
	R	0.004 sec
#till+tal+ande#	A	0.227 sec
#från+tag+ning#	A	0.065 sec
#kyrk+tag+ning#	A	0.079 sec
#bil+drull+e# #bild+rull+e#	A	0.068 sec
	R	0.003 sec

ACKNOWLEDGEMENTS

The work reported here was supported by a grant from the Tercentenary Foundation of the Bank of Sweden. We are grateful to the following persons for discussing issues of computational morphology with us, and for making contributions to the improvement of MORPH: Olli Bläberg, Robert Jarvella, Remo Job, Fred Karlsson, Lauri Karttunen, Kimmo Koskenniemi, Görel Sandström and Rob Schreuder.

NOTES

1. The old lexical analyzer was actually more intelligent than this description suggests, in that it enabled a grouping together of inflected forms of the same lexical item.

2. The position that morphology only plays a role in the processing of unknown words, and no role at all in the processing on known words is extreme, and it was deliberately chosen in order to investigate its consequences for a language processing system as a whole. In view of the fact that morphological structure appears to play a role in the processing of known (=high frequency) words, as well as unknown (=low frequency) words, we would now like to qualify our original hypothesis in one respect that concerns semantics. With respect to processing unknown words, the hypothesis stays the same. But with respect to the processing of unknown words, it is revised so that retrieval yields the entire word, its part of speech, its syntactic features and its morphological decomposition, but not the semantics of the word as a whole, in the case of normal, semantically compositional words. Only in the case of semantically non-compositional words (and phrases) does the word-lexicon carry the semantics of the word as a whole. This would imply that morphological structure is used as a control structure for semantic integration, in the case of compositional words.

REFERENCES

Bläberg, O., 1984, Svensk böjningsmorfologi - en tvånivåbeskrivning, Helsingfors universitet.

Byrd, R. & Chodorow, M., 1985, Using an on-line dictionary to find rhyming words and pronunciations for unknown words, Proceedings of the 23rd Annual Meeting of the Association for Computational Linguistics, pp 277-283.

Caramazza, A., Miceli, G., Silveri, M. & Laudanna, A., 1985, Reading mechanisms and the organisation of the lexicon: evidence from acquired dyslexia, Cognitive Neuropsychology 2(1), pp 81-114.

Church, K., 1982, A framework for processing finite state grammars of Swedish and English syntax, Report from the Department of General Linguistics, University of Umeå.

Church, K., 1983, Phrase-structure parsing: a method for taking advantage of allophonic constraints, Indiana University Linguistics Club.

Church, K., 1985, Stress assignment in letter to sound rules for speech synthesis, Proceedings of the 23rd Annual Meeting of the Association for Computational Linguistics, pp 246-253.

Deutsch, W. & Jarvella, R., 1984, Asymmetrien zwischen Sprachproduktion und Sprachverstehen, in C.F. Graumann & T. Herrmann (eds), Karl Bühlers Axiomatik - Fünfzig Jahre Axiomatik der Sprachwissenschaften, Frankfurt an Main, Klostermann, pp 173-199.

Doherty, P., Rankin, I. & Wirén, M., 1986, Erfarenheter av en implementering av Hellbergs system för svensk morfologi, i F. Karlsson (utg), Föredrag från V Nordiska Datalingvistikdagarna, 11-12 december, 1985, Helsingfors.

Ejerhed, E., 1983, Kongruens i en svensk finite state parser, föredrag vid 4:e svenska kollokviet i språklig databehandling - datorsimulering av verbalt beteende, 26 maj 1983, Lund.

Ejerhed, E., 1985, En ytstrukturgrammatik för svenska, i S. Allén et al (utg), Svenskans beskrivning 15, Institutionen för nordiska språk, Göteborgs universitet, s 175-192.

Ejerhed, E. & Church, K., 1983, Finite State Parsing, i F. Karlsson (ed), Papers from the Seventh Scnadinavian Conference of Linguistics, University of Helsinki, Department of General Linguistics, Publication No. 10 (II), pp 410-432.

Hellberg, S., 1978, The morphology of present day Swedish, Stockholm, Almqvist & Wiksell International.

Henderson, L., 1985, Toward a psychology of morphemes, in A.W. Ellis (ed), Progress in the psychology of language, Vol. 1, London, Lawrence Erlbaum Associates.

Jarvella, R., Job, R., Sandström, G. & Schreuder,R. (forthcoming), Morphological constraints on word recognition, Department of General Linguistics, University of Umeå.

Job, R. & Sartori, G., 1984, Morphological decomposition: evidence from crossed phonological dyslexia, The Quarterly Journal of Experimental Psychology 36A, pp 435-458.

Källgren, G., 1986, The role of the lexicon in heuristic parsing, paper presented at the 9th Scandinavian Conference of Linguistics, University of Stockholm, January 9-11, 1986.

Wehrli, E., 1985, Design and implementation of a lexical data base, Proceedings of the Second Conference of the European Chapter of the Association for Computational Linguistics, University of Geneva, pp 146-153.

Tove Fjeldvig og Anne Golden

AUTOMATISK SPLITTING AV SAMMENSATTE ORD - et lingvistisk hjelpemiddel for tekstsøking

SAMMENDRAG

Sammensatte ord skaper problemer ved ulike former for automatisk analyse av vokabularet i en tekst, f.eks. ved frekvensstudier. Problemet består i at meningsinnholdet i et sammensatt ord i mange tilfeller også kan beskrives i et uttrykk med de tilsvarende usammensatte ordene. I tekstsøking kan f.eks. de sammensatte ordene føre til at man ikke finner de dokumentene man søker etter fordi det ikke er samsvar i ordbruken mellom søkeargumentet og dokumentene. Hvis man f.eks. bare søker på et sammensatt ord uten å dele det opp i de enkelte ledd, vil man ikke finne de tekstene hvor alle leddene i det sammensatte ordet er nevnt, men løsrevet fra hverandre.

På denne bakgrunnen ble det utviklet en metode for automatisk splitting av sammensatte ord. Metoden er basert på et sett med ca. 1000 regler - og ikke et leksikon.

Prosjektet er et samarbeidsprosjekt mellom Institutt for rettsinformatikk (Universitetet i Oslo) og NAVF's EDB-senter for humanistisk forskning. Det er finansiert av NORDINFO med 9 månedsverk.

1. SAMMENSATTE ORD OG TEKSTSØKING

Et spesielt trekk ved de nordiske språkene er den hyppige bruken av sammensatte ord. I en hovedfagsoppgave ved Nordisk Institutt i Oslo ble det vist at ca. 25% av de ulike grafordene i en tekst var sammensatte ord (Munthe 1972). Undersøkelsen var basert både på et skjønnlitterært verk (ca. 127000 løpende ord) og en samling sakprosatekster (ca. 72000 løpende ord).

Med et sammensatt ord menes ethvert ord som er bygd opp på en slik måte at det kan deles i mindre enheter og at disse enhetene selv er ord som kan forekomme isolert. Et eksempel på et sammensatt ord er LESELAMPE som kan deles i enhetene LESE og LAMPE. Begge disse enhetene er selvstendige ord som kan opptre alene.

I tekstsøking skaper de sammensatte ordene to typer problemer. Det ene problemet oppstår når man under selve søkingen bare bruker sammensatte søkeord og det i tekstene (dokumentene) bare er brukt usammensatte ord til å beskrive det aktuelle meningsinnholdet. Disse dokumentene vil da ikke bli funnet. Det samme skjer hvis det i dokumentene

bare er brukt sammensatte ord og i søkeargumentet bare de tilsvarende usammensatte ordene.

For å få nærmere innsikt i effekten av å supplere søkeargumentet med de tilsvarende usammensatte ordene, ble det gjennomført en undersøkelse. Denne var basert på søkeargumenter som var stilt til LOVDATA - et system for søking i juridisk kildemateriale. Undersøkelsen viste at det var få av de sammensatte søkeordene som var supplert med andre uttrykk. Dessuten viste den at av ca. 80 søkeargumenter som bare inneholdt sammensatte ord, ga ca. 70 flere relevante dokumenter når de sammensatte ordene også ble splittet. Resultatet styrker med andre ord antagelsen om at man kan øke søkeeffektiviten ved å supplere et sammensatt søkeord med de tilsvarende usammensatte ordene.

Det andre problemet med sammensatte ord i tekstsøking møter man ved rangering av de funne dokumenene. Her benyttes gjerne kriterier som er basert på søkeordfrekvensen - slik at jo flere søkeord et dokument inneholder, jo høyere opp på resultatlista kommer dokumentet. Hvis man i frekvensberegningen ikke tar hensyn til forekomsten av de usammensatte ordene som de sammensatte ordene består av, vil søkeordfrekvensen gi et galt inntrykk av ordets hyppighet i dokumentet.

I dagens tekstsøkesystemer har man mulighet for å trunkere et søkeord - dvs. at det ikke søkes på hele ordet, men bare et nærmere spesifisert antall tegn i begynnelsen av ordet (høyretrunkering) eller i slutten av ordet (venstretrunkering). Søker man f.eks. på ordet ARV* - hvor "*" er trunkeringssymbolet - vil man også finne de dokumentene som inneholder ARVEAVGIFT, ARVERETT, ARVELOVEN osv. Tilsvarende for *TRYGD hvor BARETRYGD, SYKETRYGD, ALDERSTRYGD osv. vil bli funnet. Trunkering vil derfor kunne avhjelpe problemet med manglende bruk av sammensatte søkeord, men erfaring viser at det er mange som ikke benytter seg av dette hjelpemiddelet (jfr. Fjeldvig 1986).

2. SAMMENSATTE ORD I NORSK

Som nevnt innledningsvis, består et sammensatt ord av minst to usammensatte ord. En splitting av et sammensatt ord vil derfor si å finne fram til de usammensatte ordene som ordet er bygd opp av.

Vanligvis deles et sammensatt ord i to ledd, forleddet og etterleddet. Begge disse leddene kan selv være sammensatt og kan følgelig deles i nye ledd. I ordet LASTEBILSJÅFØR er f.eks. det sammensatte ordet LASTEBIL forleddet og SJÅFØR etterleddet. Betingelsen for at et ledd kan deles videre i underledd, er at det betår av frie morfemer, enten frie, semantiske morfemer (f.eks. KJØRE og BIL) eller frie, grammatiske morfemer (f.eks. OG, SÅ og TIL). Ved siden av de frie morfemene kan leddene inneholde bundne morfemer, dvs. bøyningsmorfemer (f.eks. -ENE og -TE) eller avledningsmorfemer (f.eks. -ING og U-). De forskjellige

bundne morfemene har faste plasser i forhold til det frie morfemet.

Forskjellen på frie, grammatiske morfemer og frie, semantiske morfemer er at de semantiske morfemene har både bøynings- og avledningsmuligheter (f.eks. KJØRE, KJØRTE og KJØRING), mens de grammatiske ikke har noen av delene. Begge kan imidlertid settes sammen med andre frie morfemer og danne sammensatte ord.

Verken de bundne morfemene eller de frie, grammatiske morfemene får tilført nye morfemer ved lån eller nydannelser. Vi kan derfor si at de er endelige grupper.

Leddene i et sammensatt ord vil inneholde følgende morfemtyper:

a) et fritt, semantisk morfem (f.eks LESE i LESELAMPE)
b) et fritt, grammatisk morfem (f.eks INN i INNGANG)
c) en avledning som bare inneholder ett fritt morfem (f.eks. ANBEFALING i ANBEFALINGSBREV),
d) en sammensetning av frie, semantiske morfemer (f.eks. LASTEBIL i LASTEBILSJÅFØR),
e) en sammensetning med kombinasjonen frie, grammatiske og frie, semantiske morfemer (f.eks. INNGANG i INNGANGSBILLETT),
f) en sammensetning av frie, grammatiske morfemer (f.eks. INNTIL i INNTILLIGGENDE),
g) en avledning som inneholder en sammensetning uten frie, grammatiske morfemer (f.eks BOKFØRING i BOKFØRINGSKURS),
h) en avledning som inneholder en sammensetning med kombinasjonen frie, grammatiske morfemer og frie, semantiske morfemer (f.eks UTDANNELSE i LÆRERUTDANNELSE),

I de sammensatte ordene er det etterleddet som er hovedleddet og som angir ordklassen til ordet. Vi finner sammensatte ord i de fleste ordklasser, men enkelte ordklasser har langt høyere frekvens av sammensatte ord enn andre. I Munthe (1972) framkommer det at ca. 75% av alle de sammensatte ordene var substantiv, ca. 15% var verb og ca. 6% var adjektiv. Når det gjaldt ordklassetilhørigheten til forleddet, var ca. 55% substantiv, ca. 26% adverb/preposisjoner og ca. 8% adjektiv. Den hyppigste kombinasjonen var "substantiv + substantiv" som utgjorde ca. 50% av de sammensatte ordene, mens kombinasjonen "adverb/preposisjon + verb" var den nest hyppigste og utgjorde ca. 11%.

Sett fra et nåtidsperspektiv er forleddet i en sammensetning nesten alltid ubøyd, men vi finner noen unntak.

Leddene i sammensetningene kan settes sammen på tre ulike måter:

1) direkte, f.eks. FOTFESTE og INNSETTE
2) med fuge-s, f.eks. DAGSREISE
3) med fuge-e, f.eks. BARNEHAGE

Det er i forbindelser med substantiv som forledd at fugebokstaven settes inn, og det er lydlige forhold som i første rekke bestemmer dette.

Det som står igjen av et ord når bøynings- og avledningsmorfemene er fjernet, kalles gjerne rotmorfemet. Vi vil i det følgende bruke denne betegnelsen i stedet for fritt morfem, fordi enkelte rotmorfemer må ha en endelse for å stå alene og følgelig ikke alltid er identiske med de frie morfemene.

Stavelsesstrukturen i usammensatte ord.
Ethvert ord i norsk og liknende språk er en sekvens av konsonanter og vokaler. Hvis et kluster defineres som 0, 1 eller flere konsonanter, vil alle ord passe inn i formelen:

ini + vok+ (med + vok + med + vok ....) + fin

hvor
ini: initialkluster
vok: vokal eller diftong
med: medialkluster
fin: finalkluster

(se Sigurd 1965 og Brodda 1979).

De aller fleste hjemlige, usammensatte ordene er enstavede når vi fjerner bøynings- og avledningsmorfemene, dvs. roten er enstavet (f.eks. ARV, BÅT og MANN). Et unntak er de såkalte lette tostavelesesordene som har en trykklett E i utlyd, f.eks. HANSKE og JENTE. Disse ordene mister imidlertid E'en når de får lagt til en endelse og de får da samme struktur som enstavelsesordene. Et annet unntak er tostavelsesord som ender på -EL, -EN, -ER, eks. SIRKEL, LAKEN, SKULDER. Disse ordene vil i noen tilfelle kaste E'en når de blir knyttet sammen med et bøynings- eller avledningsmorfem, f.eks SIRK'LER, SKULD'RE (tegne "'" markerer E'ens fjerning). Vi velger heretter å reservere uttrykket hjemlige ord til ord som bare inneholder norske bokstaver og i grunnformen er:

a) enstavet
b) enstavet + E
c) tostavet og ender på -EL, -EN eller -ER

Det er et begrenset antall klustre som kan stå i de forskjellige posisjonene på norsk. At et kluster kan bestå av 0 konsonanter, vil si at ordet kan innledes eller avsluttes med en vokal. Vi får følgende stavelsesstruktur for de tre typene hjemlige ord:

1) **ini + vok + fin**
2) **ini + vok + med** + E
3) **ini + vok + med** + EL/EN/ER

Hvis vi sammenholder disse strukturene, kan vi gi en fellesformel for alle hjemlige, usammensatte ord:

```
                fin
ini + vok +              (L)
          med + E +      (N)
                         (R)
```

Fremmede ord kaller vi ord som har

a) en annen struktur enn de hjemlige
b) unorske bokstaver (f.eks. C, Q, X eller Z)
c) fremmede avledningsmorfemer (f.eks. KON- og -ITET)

De fremmede ordene har ofte en flerstavet rot. Imidlertid er det mange av stavelsene som er svært produktive i utviklingen av nye ord innen teknologi og vitenskap (f.eks. TELE-, BIO-, HOMO-, -LOG, -FON og -GRAF). Disse stavelsene fungerer som avledningsmorfemer ved at de ikke kan stå alene (unntatt i fagsjargongen), men de skiller seg fra dem ved at de kan føyes direkte til hverandre f.eks. BIOLOG og TELEFON, i motsetning til avledningsmorfemene som må inneholde et fritt morfem,.

At de fremmede ordene ofte har flere enn én stavelse, vil si at de kan ha flere medialklustre i samme ord. De fremmede ordene vil altså ha følgende struktur:

ini + vok (+ **med** + **vok** + ...) + **fin**

Selv om alle klustrene som forekommer i hjemlige ord, kan forekomme i fremmede ord, er det langt sjeldnere med konsonantkombinasjoner i de fremmede ordene. De har ofte en veksling mellom én konsonant og én vokal.

Stavelsesstrukturen i sammensatte ord
I et sammensatt ord som består av 2 ledd, vil morfemene ha følgende rekkefølge:

(**pre** (+ **pre** ..)) + **rot** (+ **avl** (+ **avl** ..) (+E/S) +
(**pre** (+ **pre** ..)) + **rot** (+ **avl** (+ **avl** ..) (+ **bøyn** (+ **bøyn**..))

hvor

pre : prefiks
avl : avledningsendelse
E/S : fuge E eller S
bøyn : bøyningsendelse
rot : rotmorfem

Røttene vil ha den samme strukturen som i usammensatte ord (se ovenfor).

3. EN METODE FOR AUTOMATISK SPLITTING AV SAMMENSATTE ORD

Metoden for automatisk splitting av sammensatte ord vil i bruk bestå av et datamaskinsprogram og et sett med regler. Den er selvstendig og generell i den betydning at den for det første kan kobles til et hvilket som helst system - ikke bare tekstsøkesystemer, og for det andre kan anvendes på et hvilket som helst språk som følger overnevnte teori.

Ved bruk i et tekstsøkesystem vil metoden kunne fungere bedre hvis man koblet den til ordboka (søkefilen) i søkesystemet. På den måten kan man sjekke om de foreslåtte leddene forekommer i noen av dokumentene og dermed være bedre i stand til å velge ut den riktige løsningen. Anta f.eks. at vi har BOKSALG som søkeord. Ut fra et teoretisk grunnlag vil dette kunne deles slik:

1) BOK-SALG
2) BOKS-ALG

I ordboka til tekstsøkesystemet finner vi imidlertid ordene BOK, SALG og BOKS, men ikke ordet ALG. Dette taler for at den første løsningen er den riktige.

Resultatet som metoden gir, er bestemt av regelsettet. Dette grunnlagsmaterialet er språkavhengig, og resultatet vil være bestemt av i hvor stor grad det lar seg gjøre å formulere regler av denne type og hvor godt reglene dekker den informasjonen som det er forutsatt at de skal gjøre. Reglene skal gi informasjon om de ulike typer "byggeklosser" som et ord kan være satt sammen av, f.eks. initialkluster, prefikser, suffikser, morfemkombinasjoner, spesialord osv. En regel blir satt i kraft (dvs. at en "byggekloss" er identifisert") når alle betingelsene som knytter seg til regelen, er oppfylt. En avledningsendelse må f.eks. alltid forekomme etter et rotmorfem eller en annen avledningsendelse - og ikke bak et prefiks.

I dette prosjektet har vi utviklet et regelsett for norsk bokmål. En nærmere beskrivelse av regelsettet er gitt i avsnitt 4.

Fremgangsmåte
Metoden starter med de minste enhetene, nemlig vokalene og diftongene som er kjernen i morfemet. Dernest følger konsonantklustrene som omkranser vokalene. Metoden forsøker så å gjenkjenne de morfemene som ordet er bygd opp av, og deretter leddene. Den ender opp med ett eller flere forslag til løsninger. Disse forslagene rangeres ut fra kriterier som vi antar indikerer hvilken løsning som er mest sannsynlig.

Man kan betrakte metoden som bestående av 4 faser:

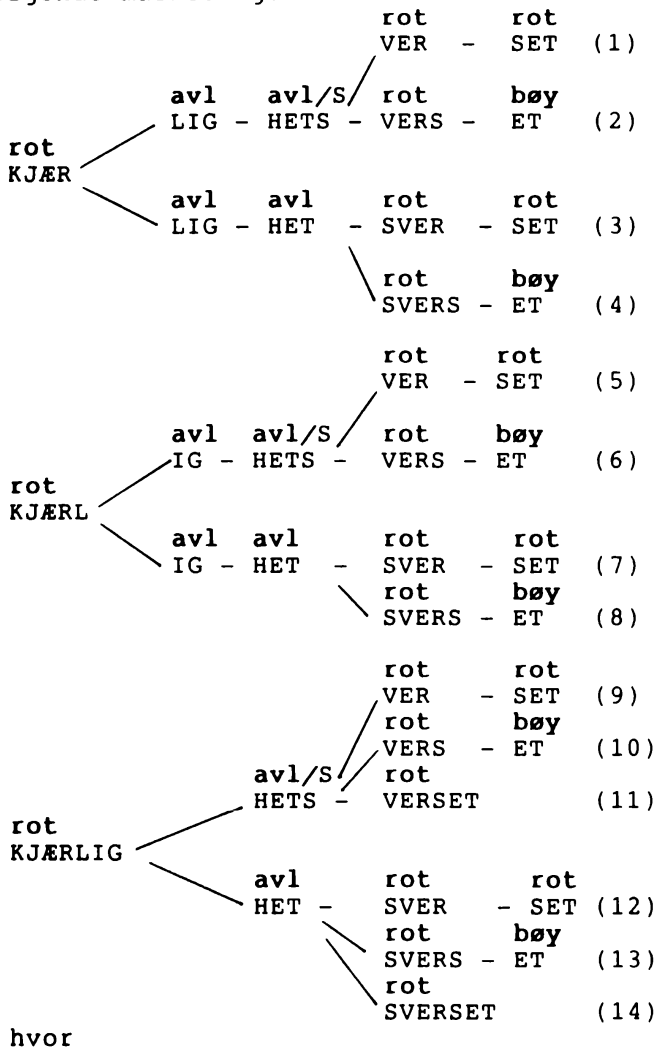
1) Kartlegge mulige morfemgrenser
2) Gjenkjenne registrerte morfemer (f.eks. bundne morfemer og frie grammatiske morfemer)
3) Gjenkjenne leddene på bakgrunn av mulige morfemkombinasjoner
4) Rangere forslagene

Den første fasen undersøker hvor morfemgrensene kan gå ut fra kjennskap til hvordan klustrene i et morfem er bygd opp. I ordet KJÆRLIGHETSVERSET vil vi finne følgende forslag til morfemgrenser:

KJÆR - L - IG - HET - S - VER - S - ET

Den neste fasen sjekker de mulige morfemene mot en liste med registrerte morfemer. Denne lista består primært av bunde morfer, men vi har også lagt inn de frie, grammatiske morfemene og en begrenset mengde med frie, semantiske morfemer (se avsnitt 4). Antall frie morfemer er begrenset fordi de er relatert til metoden og ikke til tekstsmaterialet. Det vil derfor ikke være aktuelt å supplere mengden eller eliminere elementer i den ved skifte av tekstmateriale.

I KJÆRLIGHETSVERSET vil de ulike morfemforslagene få følgende markering:



hvor

avl/S betyr avledningsendelse med fuge S.

Den tredje fasen fokuserer på morfemkombinasjonene. I eksempelet ovenfor vil vi få følgende forslag til leddgrenser:

KJÆRLIGHETS - VER - SET
KJÆRLIGHETS - VERSET
KJÆRLIGHET - SVER - SET
KJÆRLIGHET - SVERSET

Kunnskapen om hvilke avledningsendelser som må ha fuge-s når de står til forleddet, vil imidlertid forkaste to av forslagene og vi står igjen med:

KJÆRLIGHETS - VER - SET
KJÆRLIGHETS - VERSET

Den fjerde fasen stiller de alternative løsningene opp mot hverandre, og rangerer dem. For å avgjøre hvilken løsning som skal foreslås - eller i hvilken rekkefølge løsningene skal presenteres - må vi ta i bruk kriterier som er hentet fra empirien.

Ett av kriteriene vi bruker, er å preferere visse ordklasser framfor andre ut fra kjenskap de ulike ord-klassenes hyppighet som ledd i sammensetninger (se avsnitt 2). Avledningsendelsene bestemmer leddenes ordklasse, og regellista inneholder derfor informasjon om hvilke ord-klasser de forskjellige avledningsendelsene tilhører. Alle unntaksordene har også fått ordklassemarkering. Et annet kriterium for prioritering er antall ledd i et sammensatt ord. I eksempelet ovenfor vil man f.eks. legge vekt på at det er større sannsynlighet for at et sammensatt ord består av to ledd enn av tre ledd. Dessuten kan antall registrerte morfemer og rekkefølgen av dem (f.eks. hjemlige og fremmede morfemer) også kunne brukes i denne fasen. Andre bakgrunnsopplysninger kan f.eks. være:

- statistikk over klustrenes hyppighet
- " prefiksenes "
- " suffiksenes "

4. MER OM REGELSETTET

Regelsettets oppgave er å gjenkjenne morfemets oppbygning og morfemtype. Det kan grovt deles i:

1) morfemregler
2) bokstavregler

Morfemreglene består av tilsammen av ca. 600 regler, og de inneholder:

a) alle bundne morfemer, dvs. prefikser og suffikser
b) alle frie, grammatiske morfemer som forekommer i sammensetninger med frie, semantiske morfemer. Vi behandler dem som om de var bundne morfemer fordi vi pga. tekstsøkingsinteressen ikke ønsker å dele slike ord.

c) de vanligste av de frie, semantiske, hjemlige morfemene som har en struktur som tilsvarer enten

- et prefiks + et initialkluster (f.eks. PREST) eller
- et finalkluster + et suffiks (f.eks. RING)

d) de vanligste av de frie, semantiske morfemene på to bokstaver som kan forekomme i sammensetninger (f.eks. BY, IS og TE).

Bokstavreglene består av tilsammen ca. 400 regler og de inneholder oversikter over:

a) initialklustre
b) finalklustre
c) medialklustre
d) diftonger

Reglene kan inneholde

- betingelser for å tre i kraft
- restriksjoner mot å tre i kraft
- informasjoner om morfemer/klustre
- informasjon om fugebokstaver

Som eksempeler på betingelser, kan vi nevne at noen prefiks krever at rotmorfemet som følger etter, innledes med en bestemt bokstav, f.eks. det fremmed prefikset KOM som må etterfølges av B, M eller P.

Tilsvarende har enkelte prefiks restriksjoner mot enkelte påfølgende bokstaver, f.eks. det fremmede prefikset OB- kan ikke stå foran K, F eller P. Tegnsekvensen OB vil derfor ikke tolkes som prefiks hvis en av disse bokstavene følger etter. Noen suffiks vil forutsette at det ikke kommer flere suffiks foran, f.eks. -ELSE.

Reglene inneholder informasjon om:

a) morfemet er hjemlig, fremmed eventuelt begge deler.
b) det kan følge en affiks (prefiks eller suffiks) av samme type foran eller bak.
c) morfemet egentlig er en fremmed stavelse slik at det ikke nødvendigvis må komme et rotmorfem foran eller bak.
d) suffikset kan tilhøre forleddet.
e) hvilken ordklasse morfemet tilhører.
f) en fuge kan/må forekomme i forbindelse med suffikset når det er del av forleddet.
g) hvilke finalklustre som får fuge-s.

6. RESULTAT

I skrivende stund er prosjektet i avslutningsfasen. Det gjenstår å analysere og systematisere resultatene og kor-

rigere fase 4 (rangeringen) i henhold til disse. Vi har likevel sett nærmere på et mindre antall sammensatte ord (ca. 150 stk.), og resultatene ser lovende ut. De største problemene knytter seg, ikke uventet, til fuge-s (f.eks. BOKSALG som kan gi delingspunktene BOK-SALG og BOKS-ALG). De fremmede ordene skaper også problemer, men mindre enn ventet. Metoden er blitt testet mot et materiale på ca. 50 000 graford, og resultatene vil bli publisert i bl.a. Humanistisk Data i 1986.

Litteratur:

Brodda, Benny: Något om de svenska ordens fonotax och morofotax. Papers from the Institute of Linguistic, University of Stockholm (PILUS) nr 38, dec 1979.

Fjeldvig, Tove: Automatisk trunkering. Vil bli publisert i CompLex-serien (Universitetsforlaget) i 1986.

Fjeldvig, Tove og Anne Golden: Automatisk rotlemmatisering - et lingvistisk hjelpemiddel for tekstsøking. CompLex nr. 9, Universitetsforlaget. Oslo 1984.

Munthe, Synneve Kjuus: Sammensatte ord - En kvantitativ undersøkelse av norsk litteratur- og sakprosa. Hovedoppgave i nordisk, Universitetet i Oslo 1972.

Sigurd, Bengt: Phonotactic Structures in Swedish, Scandinavian University Books, Lund 1965.

Henrik Holmboe

Grammatisk beredskab

Hvor stærkt et grammatisk beredskab skal man være i besiddelse af, når man skal lave en morfologisk eller syntaktisk analyse af en tekst? Ideelt set er kravet, at det grammatiske beredskab forstået som viden om grammatiske fænomener og kategorier skal være så stærkt, at man er i stand til korrekt at analysere en hvilken som helst korrekt sætning eller ytring af sproget. Dette ideelle krav gælder, hvad enten man konstruerer en morfologisk eller syntaktisk parser til sproget, eller man lærer et menneske, f.eks. en gymnasieelev eller en studerende, at analysere et givet sprog. Når man vil være i stand til at analysere en hvilken som helst sætning i sproget, betyder det altså, at for et hvilket som helst morfologisk eller syntaktisk fænomen, skal den for analysen nødvendige viden indgå i det grammatiske beredskab.

En anden formulering af spørgsmålet om grammatisk beredskab kunne være: Hvor lille et beredskab kan man klare sig med? Denne formulering forudsætter, at man er villig til at acceptere, at analysen ikke altid lykkes. Man skal altså overveje, hvor ofte man kan acceptere en utilstrækkelig analyse. Har man en rimelig komplet grammatik, vil det i regelen kræve en ganske stor indsats at gøre den helt komplet, og udbyttet af denne indsats vil være marginal for analyseresultatet; hvis man omvendt har en komplet grammatik, der af én eller anden grund er uhåndterlig, vil den sandsynligvis kunne reduceres betydeligt uden nævneværdig skade for det endelige analyseresultat.

Kravet til det reducerede grammatiske beredskab må blot være, at det er umiddelbart ekstenderbart i retning af den komplette grammatik, således at en ekstension ikke bliver en modsigelse af den reducerede grammatik.

Det ideelle grammatiske beredskab er i god overensstemmelse med Hjelmslevs empiriprincip, som han formulerede det i "Omkring Sprogteoriens Grundlæggelse", Kbh. 1943. Ifølge dette princip skal sprogbeskrivelsen være modsigelsesfri, udtømmende og den simplest mulige, dog således, at kravet om modsigelsesfrihed er overordnet kravet om udtømmende beskrivelse og kravet om udtømmende beskrivelse overordnet kravet om simpelhed. Det lille eller reducerede grammatiske beredskab er udtryk for, at simpelhed prioriteres op, således at simpelheden får lov at dominere, når blot beskrivelsen er tilnærmelsesvis udtømmende.

Som udgangspunkt for opstillingen af en reduceret grammatik har jeg valgt 3. bog af Caesars Gallerkrig. Antagelsen er nu, at det for den grammatiske analyse af denne tekst ikke er nødvendigt at have et morfologisk beredskab, der ville være tilstrækkeligt stærkt til at parse en hvilken som helst latinsk tekst fra den klassiske periode og til at konstruere alle mulige former af et hvilket som helst latinsk ord fra samme periode. Målet er altså at foretage reduktioner, der er tilnærmelsesvis omkostningsfrie for analysens slutresultat.

Ikke overraskende viser det sig, at nominalmorfologien ikke kan reduceres, men at verbalmorfologien kan. På grundlag af en

retrogradliste med frekvensangivelse af tekstens ord er formerne optalt og opstillet i skema 1 og skema 2. Skema 1 er de absolutte frekvenser, skema 2 er formernes relative frekvens i % i forhold til samtlige verbalformer. De 788 verbalformer udgør 21.5% af tekstens ord. Foruden de i skemaet medtagne former optræder der 2 eksempler på 1. pl.præs.indik.akt., hvilket er 0.25%. Kolonnen Akt.+ Dep. angiver, hvor mange former i alt der har aktiv betydning, kolonnen Dep.+Pass., hvor mange der har passiv form. Tallene viser herefter, at de finitte former er 3. person (sg. el. plur.), at aktive former er 8 gange så hyppige som passive former, at infinitiv og perf.partc. deler de fleste af de infinitte former, samt at der kun er ca. 30 forskellige former i corpus. Ved finitte verbalformer skal her forstås usammensatte verbalformer. Sammensatte verbalformer skal opfattes som en kombination af perf.partc. og finitte former. Da der i alt er 69 former af verbet SUM, betyder det, at man ved den maximale justering får sammensatte former for 63.3% finitte former og 36.7% infinitte former.

Skema 3 viser formerne ordnet efter rang. Den stiplede linie mellem rang 11 og 12 viser, at de 11 former, hvis frekvens er over gennemsnitsfrekvensen 3.3, dækker 79.3% af verbalformerne. Linien over rang 20 viser, at de former, hvis frekvens er mindre end 1%, dækker 6.8% af verbalformerne.

Den endelige opstilling af den reducerede verbalmorfologi i skema 4 bygger på det princip, at det ikke er tilfældigt, at de hyppigste former er de hyppigste, men at det til gengæld er

ABS.	AKT.	+	DEP.	+	PASS.	I ALT
PRÆS.IND. 3.sg.	34	34		2	2	36
3.pl.	41	42	1	6	5	47
- KONJ.	11	12	1	2	1	13
	7	9	2	4	2	11
IMPF.IND.	43	43		5	5	48
	38	41	3	10	7	48
- KONJ.	22	25	3	16	13	38
	46	48	2	6	4	52
PERF. IND.	49					49
	33					33
- KONJ./FUT:EX.						
	3					3
PLUSQ.PF.IND.	12					12
	17					17
- KONJ.	10					10
	13					13
FIN. I ALT	181	185	4	25	21	206
	198	206	8	26	18	430
						224
INF. PRÆS.	88	105	17	44	27	132
- PERF.	7					7
PARTC.PRÆS.	4					4
- FUT.	6					6
- PERF.					167	167
GERUNDIV						42
INFIN. I ALT						358
VB. I ALT						788

%		AKT.	+	DEP.	+	PASS.	I ALT
PRÆS.IND.	3.sg.	4.3	4.3		0.3	0.3	4.6
	3.pl.	5.2	5.3	0.1	0.8	0.6	6.0
- KONJ.		1.4	1.5	0.1	0.3	0.1	1.6
		0.9	1.1	0.3	0.5	0.3	1.4
IMPF.IND.		5.5	5.5		0.6	0.6	6.1
		4.8	5.2	0.4	1.3	0.9	6.1
- KONJ.		2.8	3.1	0.4	2.0	1.6	4.8
		5.8	6.1	0.3	0.8	0.5	6.6
PERF. IND.		6.2					6.2
		4.2					4.2
- KONJ./FUT:EX.		0.4					0.4
PLUSQ.PF.IND.		1.5					1.5
		2.2					2.2
- KONJ.		1.3					1.3
		1.6					1.6
FIN. I ALT		23.0	23.5	0.5	3.2	2.7	26.1
		48.1			6.5		54.6
		25.1	26.1	1.0	3.3	2.3	28.4
INF. PRÆS.		11.2	13.3	2.2	5.6	3.4	16.8
- PERF.		0.9					0.9
PARTC.PRÆS.		0.5					0.5
- FUT.		0.8					0.8
- PERF.						21.2	21.2
GERUNDIV							5.3
INFIN. I ALT							45.4
VB. I ALT							100.0

RANG		%	ACCUMUL.
1:	perf.partc.pass.	21.2	
2:	inf.præs.akt.	11.2	
3:	perf.ind.akt.sg.	6.2	
4:	impf.konj.akt.pl.	5.8	
5:	inf.præs.pass.	5.6	50
6:	impf.ind.akt.sg.	5.5	
7:	gerundiv	5.3	
8:	præs.ind.akt.pl.	5.2	
9:	impf.ind.akt.pl.	4.8	
10:	præs.ind.akt.sg.	4.3	
11:	perf.ind.akt.pl.	4.2	79.3
			3.3
12:	impf.konj.akt.sg.	2.8	
13:	plusq.pf.ind.akt.pl.	2.2	
14:	impf.konj.pass.sg.	2.0	
15:	plusq.pf.konj.akt.pl.	1.6	
16:	plusq.pf.ind.akt.sg.	1.5	
17:	præs.konj.akt.sg.	1.4	
18:	impf.ind.pass.pl.	1.3	
19:	plusq.pf.konj.akt.sg.	1.3	
20:	inf.perf.akt.	0.9	6.8
21:	præs.konj.akt.pl.	0.9	
22:	præs.ind.pass.pl.	0.8	
23:	impf.konj.pass.pl.	0.8	
24:	partc.fut.akt.	0.8	
25:	impf.ind.pass.sg.	0.6	
26:	partc.præs.akt.	0.5	
27:	præs.konj.pass.pl.	0.5	
28:	perf.konj./fut.ex.akt.pl.	0.4	
29:	præs.ind.pass.sg.	0.3	
30:	præs.konj.pass.sg.	0.3	

	Ho. IND.		INF.		BI. KONJ.	
	AKT.	PASS.	A.	P.	A.	P.
P R Æ S.	-T -NT				-T	
I M P F.	-BAT -BANT	-BANTUR	-RE		-RET -RENT	-RETUR
P E R F.	-T -ERUNT		-ISSE			
P L U S Q. P F.	-ERAT -ERANT				-ISSET -ISSENT	
P.P.P.	-T- -(S)S-					
G E R U N D I V	-ND-					

tilfældigt, at de sjældneste former overhovedet er med i vores corpus. Eller formuleret lidt anderledes: Vi kan antage, at de hyppigste former i dette corpus også vil være de hyppigste i andre lignende, men at vi rimeligvis vil møde andre former end de sjældneste i stedet for dem, der -tilfældigvis - forekommer i dette corpus; men disse andre vil så nok være tilsvarende sjældne, - så sjældne, at en henvisning af dem til en punktkommentar ligger inden for rammerne af betegnelsen tilnærmelsesvis omkostningsfri for analysens slutresultat.

Ligesom det er omstændeligt at lære et menneske en udtømmende morfologi, kræver det også mange regler og mange regelniveauer at lære en datamat en morfologi, således som det er sket i f.eks. lemmatiseringsprogrammel, der er opbygget over den almindelige morfologis regler. Et lemmatiseringsprogram til analyseformål skal bl.a. kunne afgøre, om en given kontextform kan henføres til et bestemt lemma, om kontextformen "a" er en bøjningsform af lemmaet "b". Formålet kunne være, at man ønsker udskrevet alle belæg af et ord i dets forskellige bøjningsformer fra en tekst. Igen kan vi stille spørgsmålet: hvor mange fejl er vi villige til at acceptere i forbindelse med en metode, der er simplere end den klassiske lemmatisering? Mit datamateriale har igen været latin, dvs. et sprog af en overvejende eksternt flekterende, suffigerende type, eller "simpelt" udtrykt, hvor bøjning er noget, der foregår i slutningen af ordet. Udgangspunktet var endvidere, at metoden hellere måtte medtage for meget i første approximation end udelade ord, der burde have været medtaget.

En almindelig morfologisk analyse skal kunne identificere en stamme og det sæt af endelser, denne stamme kan være forsynet med. Endelserne kan være simple eller komplekse; man kan evt. diskutere, om man vil tale om en udvidet stamme med en simpel endelse eller en konstant stamme med komplekse endelser. Både stamme og endelse er ret konstante, dog kan specielt vokalalternationer forekomme, og ved overgangen mellem stamme og endelse kan sandhifænomener sløre stammens udlyd eller endelsens forlyd. Den udtømmende viden om alt dette er ganske kompleks.

Den simplere fremgangsmåde er at definere stammen som en konstant søgestreng, som altid skal forekomme som den initiale delstreng, hvis et ord skal betragtes som en mulig bøjningsform af et lemma. Hvis man søger med stammen som udgangspunkt, bliver resultatet af søgningen for upræcis; alle ønskede former kommer med, men resultatet indeholder for meget "støj". For at eliminere denne støj må man gennemføre en mere eller mindre detaljeret undersøgelse af det som følger efter stammen, hvilket ikke nødvendigvis er det samme som det, som morfologien forstår ved endelser. Jo mere detaljeret, desto nærmere vil analysen komme den egentlige morfologi i kompleksitet og viden.

Det har til min egen overraskelse vist sig, at følgende meget simple metode giver et resultat med meget lidt støj:

> for at et ord skal være en mulig bøjningsform, skal det 1) indeholde en bestemt stamme 2) have en total længde på et antal bogstaver, som ligger mellem stammen + den korteste mulige endelse og stammen + den længste mulige endelse

- dette forsynet med en -que og -ve-test.

Man kan sagtens forestille sig argumenterne for, at denne metode skulle give et upræcist resultat; derfor min overraskelse over, at resultatet blev meget "støjfrit" - på ca. 11.000 løbende ord (Plinius naturhistorie bog 36).

Efterskrift:

Spørgsmålet om, hvorvidt den reducerede grammatiks inventar svarer til det morfologiske inventar, man møder tidligt i modersmålstilegnelsen, - om, hvorvidt dette evt. genspejler en universel rækkefølge i erhvervelsen af morfologiske kategorier, er vanskeligt at besvare, fordi undersøgelser af primært materiale, dvs. børns spontane sprogproduktion, er så sjældne. Man har heller ikke mig bekendt undersøgelser af den mothertalk eller caretaker-talk, som børn i vid udstrækning er henvist til at lære ders modersmål af.

Jeg har derimod foretaget en lille undersøgelse af et par børnebøger for helt små børn og kunnet konstatere, at dette sprog, som man læser op for små børn, har følgende karakteristika:

genetiv forekommer ikke

indefinitte former er meget hyppigere end definitte.

definitte nominalsyntagmer får deres definithed ikke ved morforlogisk men ved syntaktisk markering, dvs. ikke ved efterhængt artikel, men ved rækkefølgen Det Adj N.

Jeg har så undersøgt nogle børnestile fra 3. - 7. klasse, og det ser ud til, at den skriftlige sprogproduktion i 3 kl. har meget færre genetiver og også færre definitte former end i de højere klasser.

Materialet er som sagt spinkelt og kan ikke bære vidtgående konklusioner. Min egen hypothese vil gå ud på, at man ikke skal vente at finde en universel rækkefølge i tilegnelsen af morfologiske kategorier analogt med den, Roman Jakobsen introducerer for fonologien. Spørgsmålet om, hvad der er morfologiens /a/ og /m/ og hvad der er dens /f/ og /s/, er forkert stillet; den rækkefølge, der nok kan konstateres, vil efter min mening skulle forklares conceptuelt eller semantisk, og ikke morfologisk.

Fred Karlsson
Department of General Linguistics
University of Helsinki

A PARADIGM-BASED MORPHOLOGICAL ANALYZER

1. Introduction

Computational morphology has advanced by leaps in the past few years. Since the pioneering work of Kay (e.g. Kay 1977), major contributions have been submitted especially by Karttunen (Karttunen & al. 1981) and Koskenniemi (1983). A common linguistic trait of this line of work has been a fairly strict adherence to the basic principles of generative phonology and morphology (especially of the IP type). The theories and models proposed have been decisively based on the notion of rules relating different levels of representation. Typically, the rules describe morphophonological alternations by which surface-level word-forms deviate from postulated lexical or underlying forms. Central concepts have also been the representation of lexicons as tree structures, minilexicons for describing morphotactic structure in terms of pointers to subsequent classes of allowed morphological categories (e.g. Karttunen & al. 1981), and the implementation of IP rules as finite-state transducers. A major achievement was Koskenniemi's (1983) truly bidirectional language-independent formalism for word-form production and recognition. Notions such as intraparadigmatic dependencies between subsets of endings and/or stems, as well as productivity and the mechanisms of lexical extension, have so far played only a minor role (however, cf. Ejerhed and Church's paper in the present volume).

This paper discusses a morphological analyzer called PARMORF that was designed for simulating not IP rules but **paradigmatic relationships.** One of the most notable recent trends in morphological theory has been the natural morphology advocated especially by Dressler, Mayerthaler, and Wurzel (e.g. Dressler 1985, Wurzel

1984). One of its key concepts is the notion of **paradigmatic dependency** that has been elaborated especially by Wurzel (also cf. Bybee 1985). This body of work has provided important impetus for the present effort. In particular, it is my intention to explore how feasible a paradigm view of morphology is in building computational models of word-form recognition. Another point of interest is how easily such a model can be designed to incorporate morphological productivity and lexical extension.

An important feature of PARMORF is that it renounces the use of morphophonemic symbols on the lexical level, and also does away with the corresponding phonological rules. Diacritics are used only for the purpose of singling out members of truly non-productive and closed inflectional types. Whatever morphophonological alternations there are will be expressed by stating intra-paradigmatic dependencies between stems and ending classes.

The central property of PARMORF is that the lexicon tree operational in word-form analysis is based on **stems** that are derived by **paradigmatic pattern rules** from base forms which may be either entries in the main lexicon or new words that are about to be integrated in the lexicon. The base forms of the lexical entries as such are not directly involved in word-form recognition. The PARMORF main lexicon for Finnish thus contains i.a. the noun lexeme kauppa 'shop' (N.B. in straightforward phonological shape without morphophonemes). For this lexeme, general pattern rules determine four stems with their appropriate morphotactic information (here omitted), viz. kauppa, kaupa, kauppo, and kaupo. These stems are inserted in the tree used for word-form recognition.

It is my hypothesis that once the inflectional behavior of a word is known, recognition of individual instances of it takes place in relation to the concrete stems in the lexicon tree. No (analogues of) IP/IA rules are invoked in the actual process of word-form recognition.

PARMORF embodies the hypothesis that morphological processing in the sense of "applying rules" consists primarily in determining how words so far unknown to the language user are inflected. For any word, this piece of knowledge should be supplied by a working theory of morphological productivity (here

formalized as pattern rules). Supposing that all words belonging to unproductive and closed inflectional subclasses are marked in the lexicon, the pattern rules will derive appropriate stem sets for them, and productive default stem sets for all unmarked words (whether in the lexicon or not).

This approach to morphological productivity makes the process of lexical extension fall out from entities already in the grammar. Since word-forms are recognized just by scanning concrete stems and concrete endings, PARMORF should lend itself to psycholinguistic interpretation more directly than models invoking generative rules. These models face the problem of determining how, precisely, phonological rules and their implementation as finite-state automata should be related to real behavior.

2. Lexical representations

There are at least eight ways in which the lexical forms of words may be construed:

(1) Minimal listing of the SPE-type where even distantly related word-forms are derived from a shared lexical source whose composition is claimed to be (systematic-)phonological. This underlying form lexically represents all word-forms (the whole inflectional paradigm). A central goal is to minimize the number of lexemes and to maximize the statement of morphophonolocial alternations as IP rules. Word-forms are indirectly related to the lexical representations (i.e. derived by rules).

(2) Constrained minimal listing where remotely related (especially morphophonemically irregular) word-forms are not derived from a common source. The number of lexemes postulated is therefore somewhat larger than under (1). The vast majority of words is represented by a unique lexical form as in (1). However, these base-forms as well as the rules are subject to more restricted (naturalness) conditions than are SPE-type rules. This is a modified SPE-position advocated by several variants of natural generative phonology (e.g. Hooper 1976).

(3) Unique lexical forms allowing diacritics and morphopho-

nemes. This position is embodied in most two-level implementations based on Koskenniemi's (1983) model. A lexical form may contain several morphophonemic and diacritic (e.g. juncture) symbols. Otherwise, it resembles (1,2), especially in the use of phonological rules (to be compiled as finite-state automata). Paradigms are represented by a single base-form, as in (1,2).

(4) Stems. This solution is advocated here. I regard all phonologically distinct bound variants of a base-form as separate stems. A stem-based lexicon is bound to be somewhat larger than a lexicon containing unique base forms for most words. One of the present purposes is to explore whether the amount of repetition will be prohibitively large so as to render this approach unfeasible. It deserves to be stressed that common initial substrings, meanings, category information, syntactic features, etc., in a set of stems manifesting one lexeme will not be repeated but shared in the stem tree. We are thus not heading for a theory involving whole-sale listing. - No comprehensive stem-based theory of morphology has so far been advanced, apart from the "technical stem" stem approach (5), and some general mention of (full) stems as a theoretical possibility for lexical representation (e.g. Linell 1979).

(5) Technical stems. This concept refers to the minimal invariant phonological substance occurring in all (full) stems, e.g. kaup in Fi. kauppa. Such technical stems have been used by Hellberg (1978) in his description of Swedish morphology, and by Karttunen & al. (1981) in their Finnish morphology (TEXFIN). In this approach, stem alternations are described e.g. by postulating minilexicons pointed to by the relevant technical stems.

(6) Full listing hypothesis (FLH). FLH claims that all word-forms are listed in the lexicon. This view is widely entertained in psycholinguistic research on word-form recognition (cf. Butterworth 1983). We shall discard this possibility since it leads to implausible consequences for highly inflected languages such as Finnish. Given that a Finnish verb has some 15,000 forms and an English verb less than five, FLH entails that learning Finnish verbal morphology would be thousands of times more cumbersome than learning English, and that a Finn would need much more neural space to internalize his verbs than would an Englishman.

Furthermore, according to FLH, upon learning a new verb a Finn should have to internally generate all the 15,000 forms - most of which he would never use. All this seems implausible. In face of these remarks, FLH without precisions and amendments is not acceptable as a general (psycholinguistic) theory of lexical organization. Stems provide a more uniform crosslinguistic characterization of the lexicon. E.g. English and Finnish don't differ decisively in regard to how many stems a word may have. Finnish verbs and nouns have maximally five or six stems.

(7) Semantically feasible word-forms. This would be a more realistic reduced version of FLH (to my knowledge, not yet elaborated). It would claim that the lexicon contains word-forms, but only those that are semantically feasible. Thus, the English lexicon would not (normally) contain e.g. plural forms for proper names or mass words, or personal forms for meteorological verbs.

(8) Prototypical word-forms. Given that most words, due to obvious semantic reasons, favour certain forms (e.g Fi. local nouns favour the local cases, mass nouns the partitive case, countables the nominative), it is more reasonable to suppose that the core lexicon of a language user contains the very word-forms that he/she has learnt, especially those that are in frequent active use, i.e. the prototypical ones (cf. Karlsson 1985).

All of (1-8) are not mutually exclusive. Any "realistic" model (i.e. striving not only for system description but also for isomorphy with psycholinguistic facts) must be able to account at least for frequency effects which often manifest themselves on the level of individual word-forms (cf. Garnham 1985:45 for an overview). This would presuppose special treatment (e.g. separate listing) of the most frequent and deeply engraved word-forms, regardless of whether the bulk of the lexemes are represented according to one of the alternatiuns (1-5). However, in what follows we shall only consider the feasibility of (4).

In approaches (1-3), the basic set-up of word-form processing is this:

LEXICON(S) (often compiled as tree structures)
RULES (often implemented as finite-state transducers)
SURFACE WORD-FORMS

In computational models, the (main and ending) lexicons are normally implemented as trees. These trees are direct operational analogues of the respective lexicons and are therefore the only processually relevant lexical structure. The lexicon list is an epiphenomenon helpful in inspecting the existing stock of words.

The present approach is sligthly different. I postulate a main lexicon (list) containing the stock of lexemes. Here, each lexeme is represented as a quintuple:

<base-form nextLexicon meaning syntFeatures cat>

Each lexeme has a unique base-form consisting of phonemes only. No morphological markings are needed when all stems of a base-form are predictable by general pattern rules. E.g., all Swedish nouns ending in -el, -en, -er lose their -e- in certain morphological environments and therefore no individual base-forms need diacritics. However, predicting the morphophonological behavior of the Finnish inflectional types vesi (nom.) : vede+n (gen.) and lasi (nom.) : lasi+n (gen.) presupposes that the members of the former closed, unproductive, complex class are marked (say, vesi>). Pattern rules tell what special stems -si> -nouns have. Unmarked -si -nouns constitute the unmarked default pattern.

The Finnish main lexicon thus contains nominal and verbal entries such as the following ones. nextLex will be specified for each stem by the pattern rules, the meaning is here just represented by a translation into English, and the syntactic features occur in bare outline.

```
(talo NIL house (Countable ...) N)
(vesi> NIL water (Mass ...) N)
(hullu NIL mad NIL A)
(suuri> NIL big NIL A)
(raskas> NIL heavy NIL A)
(kannas NIL isthmus (Concrete ...) N)
(anta NIL give (Trans AllRection) V)
(asu NIL live (Intrans IneRection) V)
...
```

Given the information supplied by each lexical entry, pattern rules compile the **stem lexicon tree** active in word-form recognition. The stem lexicon is crucially different from the main lexicon list since it contains full sets of stems. The stems of each lexeme share initial substrings, meaning, syntactic features, and part of speech, i.e. all lexical information apart from alternating stem segments is given just once. The core of PARMORF is thus:

PATTERN RULES (predicting stems)
STEM TREE
WORD-FORMS

3. Pattern rules

Pattern rules embody the predictive power of morphology. They are in active use only when **a new word is added to the stem tree.** Given appropriate information, the stems of a base-form are predicted and inserted in the stem tree. Once integrated, PARMORF presupposes no more (IP or IA type) processing for recognizing forms of a word. In many respects, this model is equally applicable to children's acquisition of morphology and to an adult's adding words to his/her lexicon. Note that this model embodies the core of FLH without endless listing of concrete word-forms, but also without rule processing.

The pattern rules also explicate one aspect of paradigm constitution. They determine what stems belong together and also what morphophonological alternations belong together. Such clusterings are at the heart of traditional paradigms.

Pattern rules are IF-THEN -rules obeying the following format where parentheses indicate elements not necessarily used in all pattern rules:

IF	THEN
base form coda	stem-coda$_1$ + nextLex$_1$
part of speech	(stem-coda$_2$ + nextLex$_2$)
(number of syllables)	(stem-coda$_n$ + nextLex$_n$)
(morphosyntactic feature(s))	

The core of the IF-part is the **base-form coda** (closely related to Bybee and Slobin's (1982) notion "schema"), i.e. the shortest segment string extracted from the **end** of the base form that suffices for predicting the stems. The coda is expressed as a sequence of phonemes (plus a diacritic, where needed). The part of speech is also needed by the IF-part. Syllable number is often required, as might be specific morphosyntactic features (e.g. Swedish stem prediction often needs gender). Apart from the number of syllables (which is determined by a separate algorithm) the IF-part information is given in the main lexicon entries.

For new words, this information must be made available by context of use. Evidently, inflectional behavior cannot be predicted without knowledge of part of speech, etc.

The THEN-part provides a set of pairs (at least one) each consisting of a **stem-coda** and a reference to the appropriate ending lexicon.

The inserted full stems are formed by appending the residue of the base form (i.e. what is to the left of the base-form coda) to each stem-coda. Typical Finnish pattern rules look as follows (by convention, names of ending trees are prefixed by a slash):

IF	THEN
kko, N, 2	kko /huppu
	ko /hupu

IF	THEN
ppa, N, 2	ppa /nom/sg/str
	pa /nom/sg/w
	ppo /j/pl
	po /nom/pl/w

IF	THEN
ppa, N, 3	ppa /nom/sg/str
	pa /nom/sg/w
	ppo /j/pl
	po /ammate

IF	THEN
ppää, V	ppää /loukkaa
	ppä /loukka
	pä /louka

A disyllabic gradable noun ending in -kko thus has two stems and the appropriate ending trees are /huppu and /hupu, respectively. A disyllabic gradable noun in -ppa has four stems. A trisyllabic noun in -ppa has the same four stems but a difference in what endings are allowed in weak grade plurals (ulapoita vs. *kaupoita). A verb ending in -ppää has three stems, etc.

Pattern rules are normally differentiated at least for nouns and verbs, often also for nouns and adjectives (not so in Finnish). All base-form codas generated by the pattern rules for a certain part of speech are inserted into a **pattern tree.** There is one pattern tree for each distinct part of speech. The segments of each base-form coda are inserted in reverted order, prefixed by an integer indicating the number of syllables where needed. Thus, the strings inserted in the nominal pattern tree for the first three pattern rules just mentioned are okk, app, 3app.

THEN-parts are entries under the last node of each identifiable coda in this tree. Once this base-form pattern tree exists for a given part of speech, the stem set for any such base-form is found by **picking the longest match in the pattern tree** for the search key consisting of the base-form segments in reverted order.

Thus, when the stem set for the noun ulappa is to be determined, a match for the string 3appalu is sought in the pattern tree (the integer "3" having been prefixed by the syllable counting algorithm). The longest match found will be 3app and the corresponding entry is retrieved. The four stems thus determined are inserted in the stem tree and then used in the recognition

process. For nongradable trisyllabic nouns in -a the longest match found will be 3a providing only two stems (-a, -o).

The base-form codas of the pattern rules are expressed as strings of phonemes and eventual inflectional diacritics. This leads to repetition especially for words subject to consonant gradation and mutation of the final vowel (both exemplified by ulappa). E.g. up to 15 individual instances of consonant gradation will be separately stated for the paradigms where they actually occur. There are thus some 15 pattern rules for disyllabic nouns ending in -o, viz. -kko, -ppo, -nto, etc.

Deviating from generative practice, I have deliberately chosen not to generalize consonant gradation, vowel mutation, and similar morphophonological alternations across paradigms. At first sight, this seems to lead to prohibitively unilluminating repetition. However, there are positive linguistic arguments in favour of this solution. **Particular paradigms might contain morphophonological gaps** that should somehow be accounted for. Thus, trisyllabic Fi. nouns allow only a few gradable stops at the final syllable boundary: pp, tt, kk, nt, nk. Nouns of the kaikki> -type disallow i.a. the gradable combinations lt, nt, rt at the syllable boundary. Paradigms like *karampa : karamman, *kanti : kannen are not just accidentally lacking but morphophonologically ungrammatical. That is, individual pattern rules explicitly state the allowed possibilities up to systematic gaps but exclude the latter, thereby accounting for systematic restrictions.

The principle of longest match used in searching the pattern tree gives a convenient and uniform way of handling exceptions. If the inverted form of a whole word is found in the pattern tree, it will by definition be the longest match. Thus, exception features for individual words are generally not needed.

The total number of pattern rules with the above concrete properties invoked in my full description of Finnish nominal and verbal morphology is some 1,130 (600 for nouns, 530 for verbs; some 250 exceptional pronominal forms are not included in the first figure). This number includes all idiosyncracies (roughly half of these rules concern one item only). Considering that the power of the pattern rule system is such as to predict the inflection of all nouns, adjectives, and verbs in the lexicon,

including all exceptions, **and** the default inflection of any such word not in the lexicon, and furthermore excluding many types of impossible paradigms, we would not regard the number as "prohibitively large", especially when one takes into account that no further morphophonological rules or processing is invoked in word-form recognition. I.e., full productive mastery of Finnish morpho(no)logy presupposes learning some 1,100 concrete phoneme-level rules.

4. Ending lexicons

Similarly behaving endings are grouped into ending lexicons which are triples with the following structure:

<name, otherLex, endings>

Each ending lexicon has a name (conventionally prefixed by a slash) normally chosen so as to give a mnemonic hint of what kinds of stems or words it is normally appended to. The component "otherLex" provides a (possibly null) list of **other ending lexicons paradigmatically included in the present one.** This facility provides a convenient opportunity of stating paradigmatic relationships between distributionally related subsets of endings. Finally, the compartment "endings" is a (possibly empty) set of endings belonging to the current ending lexicon (i.e. possibly empty because an ending lexicon may consist exclusive of references to other ending lexicons under otherLex). Each ending, in turn, is a triple:

<item, nextLex, entry>

where "item" is the ending in phonemic shape, "nextLex" a reference to the next morphotactic position, and "entry" contains a list of morphological categories. Vowel harmony is an exception to the phonemic principle of item structure, i.e. suffix vowel harmony pairs are lexically represented as the archiphonemes A,

O, U, which are spelled out as a-ä, o-ö, u-y when the ending lexicons are compiled into trees used in actual processing.

The endings and entries are often listed as wholes, especially in close-knit combinations of e.g. number and case for nouns. Such combinations are often subject to bidirectional dependencies that are hard to capture otherwise. The /j/pl lexicon below contains good examples of this dependence. The plural allomorph j occurs only if the following ptv. or gen. case morph starts with a vowel, and the latter occur only if pl. j precedes. Furthermore, for gradable nouns the -jA, -jen -combinations are tied to strong-grade stems only (koivikkojen vs. *koivikojen). This complex paradigmatic interdependence between a certain stem, a certain number morph, and a certain case morph has proven laborious to capture by (morpho)phonological rules. Under the present approach, it suffices to point from one stem to one lexicon.

A psycholinguistic argument for treating (some) ending sequences as wholes comes from the observation that children acquiring inflectional languages seldom make errors involving the order of morphemes in a word (cf. Bybee 1985:114ff. for an overview).

The following are typical examples of ending lexicons. The name is given on the first line, otherLex on the second, and the endings, if any, are indented.

```
(/nom/sg/str
(/clit/nom /ill/Vn /poss3)
     (A /poss4 (PTV SG))
     (nA /poss4 (ESS SG)))

(/nom/sg/w
NIL
     (n /clit (GEN SG))
     (llA /poss4 (ADE SG))
     (ltA /poss4 (ABL SG))
     (lle /poss5 (ALL SG))
     (ssA /poss4 (INE SG))
     (stA /poss4 (ELA SG))
```

```
     (ksi /clit (TRA SG))
     (kse /poss6 (TRA SG))
     (ttA /poss4 (ABE SG))
     (t /clit (NOM PL)))

(/nom/pl/w
NIL
     (illA /poss4 (ADE PL))
     (iltA /poss4 (ABL PL))
     (ille /poss5 (ALL PL))
     (issA /poss4 (INE PL))
     (istA /poss4 (ELA PL))
     (iksi /clit (TRA PL))
     (ikse /poss6 (TRA PL))
     (ittA /poss4 (ABE PL))
     (in /clit (INS PL))
     (i /poss2 (INS PL)))

(/i/pl
NIL
     (iA /poss4 (PTV PL))
     (ien /clit (GEN PL))
     (ie /poss2 (GEN PL))
     (iin /clit (ILL PL))
     (ii /poss2 (ILL PL))
     (inA /poss4 (ESS PL))
     (ine /poss6 (COM SG/PL)))

(/j/pl
NIL
     (jA /poss4 (PTV PL))
     (jen /clit (GEN PL))
     (je /poss2 (GEN PL))
     (ihin /clit (ILL PL))
     (ihi /poss2 (ILL PL))
     (inA /poss4 (ESS PL))
     (ine /poss6 (COM SG/PL)))
```

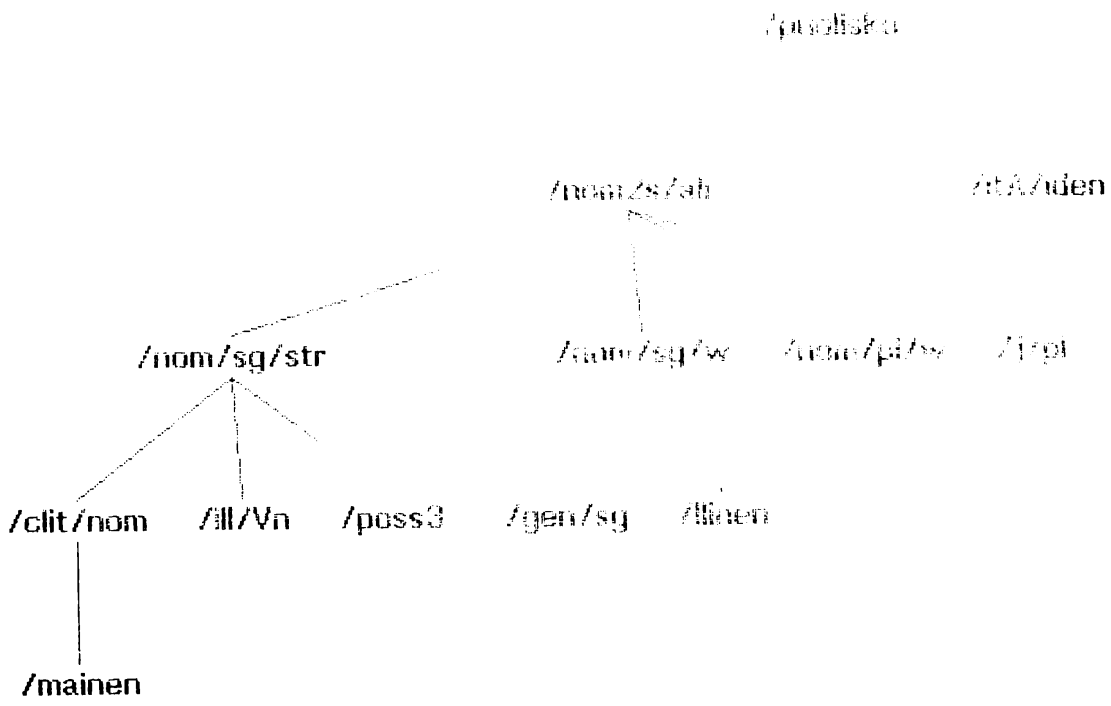
```
(/huppu
(/nom/sg/str /j/pl))

(/hupu
(/nom/sg/w /nom/pl/w))

(/nom/2s/all
(/huppu /hupu))

(/puolisko
(/nom/2s/all /itA/iden))

(/itA/iden
NIL
     (itA /poss4 (PTV PL))
     (iden /clit (GEN PL))
     (itten /clit (GEN PL))
     (ide /poss2 (GEN PL))
     (itte /poss2 (GEN PL)))
```

Endings in the same ending lexicon behave alike. An ending lexicon constitutes a kind of "paradigmatic natural class". Thus, /nom/sg/str contains endings occurring after strong-grade sg. stems of (certain) gradable nouns. These endings are ptv. -A and ess. -nA, plus certain clitics, possessives, and illatives included via the specifications in otherLex. /nom/sg/w contains the corresponding weak-grade sg. endings, /nom/pl/w the weak-grade pl. endings.

The paradigm formalism enables us to capture complex **intersecting paradigmatic networks** by way of otherLex references. Thus, the lexicon /huppu (covering strong-grade sg. and pl. stems like huppu, lakko) contains the members of /nom/sg/str and /j/pl but no endings of its own. /hupu (covering the corresponding sg. and pl. weak-grade stems) contains the members of /nom/sg/w, /nom/pl/w. Then one may continue: /nom/2s/all covers the corresponding non-gradable stems (words like talo, hullu) and is described by referring via otherLex to /huppu, /hupu. Yet another layer may be added by describing trisyllabic non-gradable nouns

(e.g.puolisko) as /puolisko consisting of /nom/2s/all and /itA/iden. This captures the generalization that these nouns depart from the disyllabic ones only in having some more alternative plural endings.

In other words, references via otherLex are recursively broken down by tracing all the lexicons invoked. The hierarchical paradigmatic lexicon network may be displayed as follows:



The full description of Finnish contains 134 ending lexicons. At run-time, two options are available for compiling ending lexicons to trees. In the minimal version, each ending tree contains only the endings listed in the respective lexicon, and when a word-form is to be analyzed, eventual otherLex references are all checked separately and recursively by jumping from tree to tree. E.g. when the /puolisko tree is consulted, all 13 trees

in the display above are run through. The 134 minimal ending trees require some 1,000 nodes. The maximal option lumps together into one tree the endings of the current lexicon plus all endings found by recursively checking the otherLex references (e.g. all 13 trees under /puolisko). In this mode, the lexicon trees require some 8,000 nodes. Of course, using maximal ending trees speeds up the recognition process (roughly by a factor of three).

This kind of paradigmatic description does capture significant generalizations. It also makes interesting predictions, e.g. that paradigm levelling or extension is likely to concern **all** the members of a given ending lexicons (in due course).

5. Implementation and evaluation

The formalism for expressing pattern rules, stems, and ending lexicons is language-independent. The pattern rules must, of course, be determined by the linguist before they can be read by the program, i.e. before the pattern tree is constructed. The program reads lexical entries of the specified type upon constructing the stem tree.

So far, I have only tested the model on Finnish. The current size of the Finnish main lexicon is roughly 9,000 items (of which 4,300 are nouns and 2,000 verbs). On the average, a Fi. noun has 2,5 stems and a verb 3,2 stems (in the sense of phonologically distinct from the base-form). When all stems of these 9,000 items are compiled into the stem tree, its size is roughly 41,000 nodes. A rough comparison to Koskenniemi's (1983; personal communication) Fi. lexicon shows that a full stem-approach **less than doubles the number of nodes in the main lexicon tree.** I find this rough ratio interesting as it proves that a stem-based lexicon is **not** prohibitively much larger than a lexicon based on unique lexical forms. For IE languages stem-based lexicons would be even more manageable than in Finnish.

The "cost" of the stem-based approach is thus a doubling of the number of main lexicon nodes. This is counterbalanced by an elimination of morphophonological processing at run-time which of

course streamlines and speeds up the actual process of word-form recognition. Using maximal ending trees, word-form recognition over the 9,000-item stem tree takes 30 ms on the average (including multiple analyses of homonyms). Short unambiguous words are analyzed in 10-15 ms.

The program provides for productive morphological analysis of any compound just by turning a switch. In normal mode, all analyses are produced. Another switch constrains the analyzer to producing one analysis only. The given efficiency figures pertain to this non-compound mode.

Acknowledgement

I am indebted to Kimmo Koskenniemi and Martti Nyman for insightful comments.

References

Butterworth, Brian 1983. "Lexical Representation". In B. Butterworth, ed., Language Production, Vol. 2, New York: Academic Press, 257-294.

Bybee, J.L. 1985. Morphology. A Study of the Relation between Meaning and Form. Amsterdam: Benjamins.

Bybee, J.L. and Slobin, D.I. 1982. "Rules and schemas in the development and use of the English past tense". Language 58, 265-289.

Dressler, Wolfgang U. 1985. Morphonology. Ann Arbor: Karoma Publishers.

Garnham, Alan 1985. Psycholinguistics. Central Topics. Methuen: London and New York.

Hellberg, Staffan 1978. The Morphology of Present-Day Swedish. Stockholm: Almqvist & Wiksell International.

Hooper, J.B. 1976. An Introduction to Natural Generative Phonolo-

gy. New York: Academic Press.

Karlsson, Fred 1985. "Paradigms and Word-Forms". Studia gramatyczne 7, 135-154.

Karlsson, Fred & Koskenniemi, Kimmo 1985. "A Process Model of Morphology and Lexicon". Folia Linguistica 19:1/2, 207-229.

Karttunen, Lauri, Root, Rebecca, & Uszkoreit, Hans 1981. "TEXFIN: Morphological Analysis of Finnish by Computer". Paper read at the 71st Annual Meeting of the SASS, Albuquerque, New Mexico.

Kay, Martin 1977. "Morphological and Syntactic Analysis". In A. Zampolli, ed., Linguistic Structures Processing, Amsterdam: North-Holland, 131-234.

Koskenniemi, Kimmo 1983. Two-level Morphology: A Theory for Automatic Word-Form Recognition and Production. University of Helsinki, Department of General Linguistics, Publications No. 11.

Linell, Per 1979. Psychological Reality in Phonology. Cambridge University Press.

Mayerthaler, Willi 1981. Morphologische Natürlichkeit. Wiesbaden: Athenaion.

Wurzel, W.U. 1984. Flexionsmorphologie und Natürlichkeit. Berlin: Akademie-Verlag.

Kimmo Kettunen

SITRA Foundation
P.O. Box 329
SF-00121 HELSINKI

ON MODELLING DEPENDENCY-ORIENTED PARSING

1. Introduction

Dependency grammars have not been in the mainstream of grammars whether we consider "traditional" grammars or grammars used in computational linguistics. In recent years, however, interest in dependency-oriented grammars has grown considerably. Text book writers and researchers even outside the dependency grammar tradition have laid more interest in dependency relations (cf. e.g. Lyons 1977, Matthews 1981, Miller 1985, Somers 1984).

Much work in dependency analysis seems to have been done in Slavonic and German languages and Japanese, which are all (highly) inflectional. Quite recently researchers of non-inflectional languages have also become interested in dependency-oriented models. It is probable that dependency analysis is suitable for both kinds of languages. Currently there is active work going on in some sort of automatic dependency analysis or formal modelling of dependency analysis at least in Czechoslovakia (e.g. Sgall, ed. 1984), Japan (Muraki & Shunji & Fukumochi 1985), BRD (Hellwig 1985), GDR (Kunze 1982), Great Britain (Hudson 1985, Fraser 1985), in the USA (Starosta 1985), Soviet Union (Urutyan & Simonyan 1983). EC´s machine translation project, EUROTRA, uses dependency relations in its sentence analysis (Johnson & King & des Tombe 1985).

This paper describes some basic linguistical characteristics of the parser, DADA, that has been implemented as a part of a database interface system for written Finnish queries (Nelimarkka et al 1983, 1984, Lehtola et al 1985). The parser is general and it is by now capable of analyzing a nontrivial subset of Finnish clauses. The basic idea of the parser is to provide analyzed sentences with syntactico-semantic structure. The structure that is given to an input clause is a functional case-labeled dependency structure. Dependency is stated and interpreted in functional labels which are then further interpreted using semantic roles. Therefore a superficial semantic representation is given to the analyzed sentence.

The following set lists salient features of our parser:

1) Strength in grasping word-order variations of an inflectional language. This is due to the dependency grammar and to the implementation that employs two-way automata (cf. Levelt 1974).

2) Multidimensional analysis: full advantage of the rich inflectional morphology of Finnish is obtained. Rules of grammar are stated so that morphological knowledge as well as knowledge from higher strata may appear in them.

3) Parallel syntactic and case-semantic analysis or only syntactic analysis may be obtained as one wishes.

4) Semi-strict separation of linguistic knowledge and parsing mechanism. This is due to the high-level grammar description language, DPL, in which the grammar is written. The grammar and the parser in their present version have some 30 pages of DPL-description. That makes about 5500 lines of compiled elementary LISP-code.

5) The parser is strongly data-driven. Parsing proceeds bottom-up and is lexicon-based. Structures are built from words to larger constituents (cf. Winograd 1983).

6) All the time the parser has only a few rules that must be checked. The only hypothesis made are those, which come with expectations of the current stack. When rules are activated like this, the size of grammar will not affect the efficiency of the parser.

2. A sketch of the parsing process

Firstly we shall briefly sketch the overall parsing process. A morphological analysis precedes the parser. We have a morphological processor that analyzes the inflected words and gives the basic word forms and inflectional categories (cf. Jäppinen et al 1983). This is, of course, a prerequisite for parsing a highly inflected language, such as Finnish. The parser gets morphologically analyzed words with their lexical information. Lexical descriptions come from the parser´s own lexicon.

A disambiguator for ambiguous morphological output should also exist somewhere. One place for it could be after morphological analysis (cf. Karlsson 1985b) or the disambiguation could be done during the parse. So far we have none. For each morphologically ambiguous token of a word form the parser tries to find dependents. This leads to multiple work and possible misparses.

The DADA parser proceeds word-by-word in the input clause. The basic method of the parser is that it tries to recognize possible dependents for each token of a word category moving out from the nearest neighbour. As the parser is modelled with two-way automata, it can recognize subordinates both from the left and right context of the current input word. For each word category possible dependents are described in automaton networks.

We may generalize that during the parse local dependencies are sought first: each word category gathers dependents of its own type. When each non-verbal category has finished gathering its dependents and it is labeled as +Phrase (i.e. it governs > 0 dependents), it may be matched to the global structure of the clause. In non-elliptic sentences (sentences containing a finite verb) the parsing is always finished when some global dependent (dependent of the main verb) is attached and the working stack is empty.

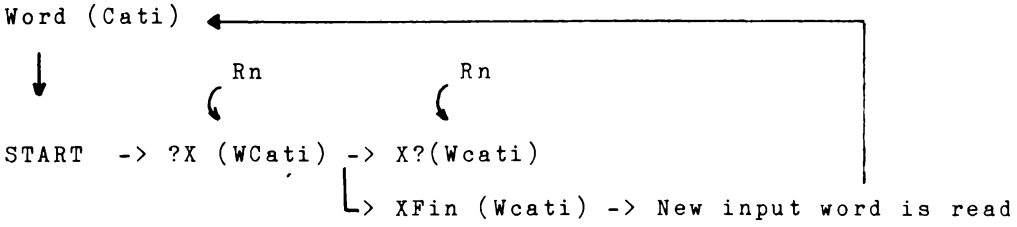


Fig.1: a simplified description of the flow of parsing

Here ?X and X? refer to left and right-hand states, respectively. START is the initial state that sends the analysis to the proper automaton network. Rn´s are dependency relations.

Dependency is stated in two steps: for each word-class a set of possible dependents is determined by function names in states. Possible orderings of dependent and regent are stated in left and right-side automaton descriptions.

Dependency relations are the only rules that are used. A dependency relation concerns a pair of words (C, D) which have to fullfill the requirements stated in the rule. Rules are written as conditions or constraints that have to hold between C and D. Schematically rules are stated as follows:

```
((C = < MorphC SyntCat ConstFeat SyntFeat SemCat SemFeat
                                  FrameCat FrameFeat >)

    ->

((D = < MorphC  SyntCat ConstFeat SyntFeat SemCat SemFeat
                                  FrameCat FrameFeat >)))
```

where

MorphC = inflectional knowledge
SyntCat = syntactic category
SyntFeat = a bundle of syntactic features
SemCat = name of the semantic class
SemFeat = a bundle of distinctive semantic features
ConstFeat = knowledge that is figured out during the parse and
is stated in binary features
FrameCat = frame category of the verb
FrameFeatc= frame features of the verb
C = regent candidate
D = dependent candidate

Features and categories can be written in disjunctive and conjunctive statements to form rules of grammar. When a match between C and D succeeds D is attached to C and labeled with the proper syntactic function label and (possibly) with semantic case role label.

Our kind of description is rather far from the Haysian classical dependency notation. Whereas Hays describes order and restrictions in the same formula (Hays 1964), we have different descriptions for each. Especially the word order restrictions are currently described rather clumsily. Possible word orders are blurred into paths of the automaton network. As a linguistic description this is not satisfying and a new way for describing the word order should be found. The second major problem is that lexical knowledge is stated in two places. At present automaton descriptions work as a kind of valency lexicons which give the possible dependents of each word-category. But it is obvious that this information should be given separately in the lexicon.

3. Assigned structures

The parser builds labeled dependency trees which have the following characteristics:

- the linear order of the surface clause is preserved in the trees

- heads and their modifiers are attached directly without any non-terminal symbols

- dependency trees have labels of two kinds: syntactic function labels and case role labels. Syntactic functions are the main objects that are created by the dependency relations. Case role labels are further interpretations of these functional relations.

4. Dependency

As we know two elements of a sentence are directly related in a dependency characterization if one DEPENDS on (conversely, is GOVERNED by) the other. The relationship is transitive, irreflexive and antisymmetric. A dependency relation is said to hold between two elements in certain circumstances. One member of this relation is called the GOVERNOR (or head or regent), the other the DEPENDENT (or modifier) (cf. e.g. Hudson 1980a, 1984, Mel´cuk 1979, Kunze 1975).

Two intuitive ideas determine the existence of a dependency relation between the governor and the dependent:

i) The governor expects to find certain types of dependents in its neighbourhood. This expectation is inherent to the element that constitutes the governor, and may be a piece of dictionary knowledge.

ii) The governor is felt to be modified by the dependent (not vice versa).

Johnson & King & des Tombe (1985) have discussed some basic problems concerning the dependency construction. It is useful to briefly state their points and consider our formalism in those respects.

The basic representational principles may be stated followingly (cf. also Robinson 1970):

i) There is one-to-one correspondence between leaves of the tree and primitive elements of the construction represented by the tree.

ii) There is one-to-one correspondence between nonterminal nodes of the tree and constructions of the text.

iii) Labellings on the nodes express the categories of primitive elements, constructions, and dependency relations.

As Johnson & King & des Tombe point out, this is elegant but empirically wrong and must be augmented. There are two classes of problems to the basic representational theory. Firstly it implies that no unit can be a member of more than one construction. Secondly it implies that every unit must be a member of at least one construction (except the text itself). But this is not the case. In a sentence like "John tried to swim" "John" is member of two constructions (Hudson calls this modifier-sharing) and this cannot be represented in a tree form. Accordingly in the sentence "Tom went to Paris and Hanna to London" "went" is a governor for two constructions (head-sharing). The Eurotra formalism has introduced a speacial notion of EMPTY ELEMENTS to handle these phenomena. These are shadow elements of their antecedents, i.e. the elements that participate in more than one construction. The empty elements in trees are leaves that do not correspond to anything at all in the text (the one-to-one correspondence is no longer valid).

There are some further problems. In some constructions there exist elements that are not dependent on anything in the clause. In "Frankly, I do not care a bit" "frankly" does not seem to be dependent on any word in the clause. For these situations a notion of TRANSCONSTRUCTIONALS is introduced in the Eurotra formalism. These are handled in a way that makes them as if they were dependents in the construction they are related to intuitively. A special label, pseudodependency, is attached to them.

Such problems are of course existent also in Finnish. Especially the problem of modifier-sharing is common in rather simple clauses already. Different kinds of infinitive constructions are typical examples of the phenomenon. Clauses such as "Poika haluaa analysoida kiviä" ("The boy wants to analyze stones") cannot be properly handled in our parser at present. Some new methods for handling these phenomena should be added either to the parser or at least in post parser analysis of sentences. At present the constructions of the parser are based only on the naively elegant one-to-one correspondence principle.

REFERENCES

Fraser, Norman 1985. A Word Grammar Parser. M.Sc. thesis, Department of Computer Science. University College, London.

Hays, David G. 1964. Dependency Theory: a Formalism and Some Observations. Language 40: 511 - 525.

---------- 1966. Parsing. In Hays, D.G. (ed.). Readings in Automatic Language Processing. American Elsevier Publishing Company, New York: 73 - 82.

Hellwig, Peter 1985. Program System PLAIN. "Programs for Language Analysis and Inference. Examples of Application." Computing Unit and Linguistics and International Studies Department. University of Surrey, Guildford.

Heringer, Hans Jürgen & Strecker, Bruno & Wimmer, Rainer 1980. Syntax. Fragen - Lösungen - Alternativen. Wilhelm Fink Verlag, München.

Hudson, Richard 1976. Arguments for a Non-transformational Grammar. The University of Chicago Press, Chicago.

---------- 1980a. Constituency and Dependency. Linguistics 18: 179 - 198.

---------- 1980b. Daughter-dependency Grammar. In Hans-Heinrich Lieb (ed.). Oberflächensyntax und Semantik. Linguistische Arbeiten 93. Max Niemeyer Verlag, Tübingen.

---------- 1983. Word Grammar. In Hattori, Shiro & Inoue, Kazuko 1983 (ed.). Proceedings of the XIIIth International Congress of Linguists, Tokyo: 89 - 101.

---------- 1984. Word Grammar. Basil Blackwell, Oxford.

---------- 1985. A Prolog Implementation of Word Grammar. Mimeo, October 85, University College, London.

Johnson, Rod & King, Maghi & des Tombe, Louis 1985. A Multilingual System under Development. Computational Linguistics 11: 155 - 169.

Jäppinen, Harri & Lehtola, Aarno & Nelimarkka, Esa & Ylilammi, Matti 1983. Morphological Analysis of Finnish: a Heuristic Approach. Helsinki University of Technology, Digital Systems Laboratory, report B26.

Karlsson, Fred 1985a (ed.) Computational Morphosyntax. Report on Research 1981 - 1984. Publications of the Department of General Linguistics. No. 13. University of Helsinki.

---------- 1985b. Parsing Finnish in Terms of Process Grammar. In Karlsson 1985a (ed.): 137 - 176.

Kunze, Jürgen 1975. Abhängigkeitsgrammatik. Studia Grammatica XII. Akademie-Verlag, Berlin.

---------- 1982a (Hrg.). Automatische Analyse des Deutschen. Akademie-Verlag, Berlin.

---------- 1982b. Einführung. In Kunze 1982a: 17 - 34.

Lehtola, Aarno & Jäppinen, Harri & Nelimarkka, Esa 1985. Language-based Environment for Natural Language Parsing. Proceedings of the Second Conference of the European Chapter of the Association for Computational Linguistics, USA: 98 - 106.

Levelt, W.J.M. 1974. Formal Grammars in Linguistics and Psycholinguistics. Volume II: Applications in Linguistic Theory. Mouton, The Hague.

Lyons, John 1977. Semantics 2. Cambridge University Press, Cambridge.

Matthews, P.M. 1981. Syntax. Cambridge University Press, Cambridge.

Mel´cuk, Igor 1979. Studies in Dependency Syntax. Linguistica Extranea, Studia 2. Karoma Publishers, Ann Arbor.
Miller, J. 1985. Semantics and Syntax: Parallels and Connections. Cambridge University Press, Cambridge.
Muraki, Kazunori & Ichiyama, Shunji & Fukumochi, Yasumoto 1985. Augmented Dependency Grammar: a Simple Interface between the Grammar Rule and the Knowledge. Proceedings of the Second Conference of the European Chapter of the Association for Computational Linguistics, USA: 198 - 204.
Nelimarkka, Esa & Lehtola, Aarno & Jäppinen, Harri 1984a. A Computational Model of Finnish Sentence Structure. In Anna Sågvall Hein (ed.). Föredrag vid De Nordiska Datalingvistik-dagarna 1983, Uppsala: 169 - 177.
---------- 1984b. Parsing an Inflectional Free Word Order Language with Two-way Finite Automata. In Shea, Tim O. (ed.). ECAI-84: Advances in Artificial Intelligence. Elsevier Science Publishers: 167 - 176.
Platek, M. & Sgall, J. 1984. A Dependency Base for Linguistic Description. In Sgall 1984 (ed.): 63 - 98.
Robinson, Jane J. 1970. Dependency Structures and Transformational Rules. Language 46: 259 - 285.
Sgall, Petr 1984 (ed.). Contributions to Functional Syntax, Semantics and Language Comprehension. Linguistic & Literary Studies in Eastern Europe vol. 16. John Benjamins, Amsterdam.
Somers, Harald 1984. On the Validity of the Complement-adjunct Distinction in Valency Grammar. Linguistics 22: 507 - 530.
Starosta, Stanley 1985. The End of Phrase Structure as We Know It. Series A, Paper no. 147. L.A.U.D.T., Linguistic Agency of Duisburg.
Urutyan, R.L. & Simonyan, S.L. 1983. Analysis of Equivalence in Language by Means of D-grammars. In Tiits, M. (ed.). Symposium on Grammars of Analysis and Synthesis and Their Representation in Computational Structures. Academy of Sciences of the Estonian S.S.R., Tallinn: 108 - 109.
Winograd, Terry 1983. Language as a Cognitive Process. Volume I, Syntax. Addison Wesley, Massachusetts.

Poul Søren Kjærsgaard
Institut for erhvervssprog
Odense Universitet
Campusvej 55
DK-5230 Odense M

REFTEX
ET DATAMATSTØTTET OVERSÆTTELSESSYSTEM

1. Indledning

Der er flere måder, hvorpå datamaskiner kan inddrages i oversættelsesarbejdet, hvorved forstås samtlige de trin, der foretages fra, at der foreligger en tekst på et kildesprog til, at denne tekst foreligger oversat til et målsprog.

Inden for datamatisk oversættelse sondres der mellem to hovedfremgangsmåder, automatisk oversættelse (machine translation) og datamatstøttet oversættelse (machine-aided translation). Den første er den ældste og den, som Warren Weaver tænkte på i sit berømte memorandum fra 1949.[1] Formålet er her at lade maskiner udføre den væsentligste del (kvantitativt som kvalitativt) af oversættelsesprocessen, dog i de fleste tilfælde suppleret med menneskelig efterredigering af maskinens output. Denne fremgangsmåde er skiftevis blevet mødt med optimisme og pessimisme, men oplever her i 80'erne en renæssance gennem det europæiske EUROTRA-projekt.

Heroverfor står den datamatstøttede oversættelse, hvor mennesker fortsat udfører den væsentligste del af oversættelsesarbejdet, men bistået af maskiner til de rutineprægede dele af dette arbejde. Sådanne systemer er blevet udviklet siden 60'erne. Der findes mange forskellige systemer, som ikke nærmere skal beskrives her. Fælles for dem er, at de bygger på en ordbog/termbank, indeholdende to eller flere sprog, og at de derudover indeholder nogle søge- og redigeringsfunktioner til søgning af ækvivalenter i målsproget og til udskrivning af lister eller til automatisk indsætning af disse ækvivalenter i kildesprogsteksten. Lidt forenklet kan man beskrive datamatstøttede oversættelser med det følgende citat:

> All known methods are essentially simple search procedures based on the comparison of characters but with more or less sophisticated features ... They do not by any means take the place of the translator's thought process, it rather relieves him of routine search operations.[2]

Denne karakteristik gælder også for det system, der beskrives i det følgende.

Men i modsætning til de fleste datamatstøttede systemer der indeholder en ordbog, bygger REFTEX på en tosproget (kontrastiv) konkordans, altså en fortegnelse over alle et ords forekomster i forbindelse med deres respektive kontekster i kildesproget og for hver eneste af disse det tilsvarende tekstafsnit (konkordans) i målsproget.

2. Princippet i REFTEX

REFTEX (som er et akronym for reference text) er et datamatstøttet oversættelsessystem, der adskiller sig fra såvel de automatiske som fra flertallet af de datamatstøttede systemer.

Systemet er interaktivt (konversationelt), og det kan beskrives på følgende måde:

> En oversætter, der i forbindelse med en oversættelse af en tekst er stødt på nogle problemer (i form af ord eller udtryk, der enten ikke findes eller ikke findes tilfredsstillende oversat i traditionelle hjælpemidler som glossarer og ordbøger) indtaster et efter et disse ord og udtryk på sin dataskærm. Datamaten forsøger herefter at finde disse ord/udtryk i referencetekster (tekstpar af original og oversættelse på hhv. kildesprog og målsprog), der på forhånd er indlæst og lagret i en adækvat datastruktur. Såfremt ordet findes, udskriver datamaten én efter én den/de passage/r (konkordansen), der indeholder ordet og de tilsvarende passager i et eller flere sproglige målsprog. Oversætterens opgave er herefter - i kraft af sin sproglige kompetence - at afgøre, om det/de oversættelsesforslag, konkordanserne indeholder, kan anvendes til det konkrete problem eller i givet fald at forkaste dem alle.

Det skal tilføjes for en ordens skyld, at fremgangsmåden i modsætning til almindelige konkordanser ikke sigter mod et færdigt produkt, fremstillet én gang for alle, og som kan opbevares på mikrofiche eller trykkes, men derimod skal opfattes som et ad hoc-værktøj som en oversætter i påkommende tilfælde kan benytte sig af. Den tosprogede konkordans fremstår altså som et øjebliksbillede på skærmen eller fra printeren.

REFTEX bygger på en kombination af to fremgangsmåder, dels anvendelse af referencetekster, dvs. tidligere oversættelser, som hjælpemiddel ved oversættelsesarbejde, dels anvendelse af en datamatbaseret tosproget konkordans.

Begge principper er velkendte. Således anvendte munke i middelalderen Biblen som referencetekst for at lære ordenes betydning at kende. Idag anvendes princippet af bl.a. oversættelsestjenesterne ved De europæiske Fællesskaber.

Anvendelse af tosprogede konkordanser (og inden for de seneste årtier ved hjælp af edb, elektronisk databehandling) sker i forbindelse med filologiske arbejder

med henblik på f.eks. syntaktiske undersøgelser.[3]

Kombinationen af de to fremgangsmåder som i REFTEX findes omtalt et par steder som et muligt arbejdsområde.[4]

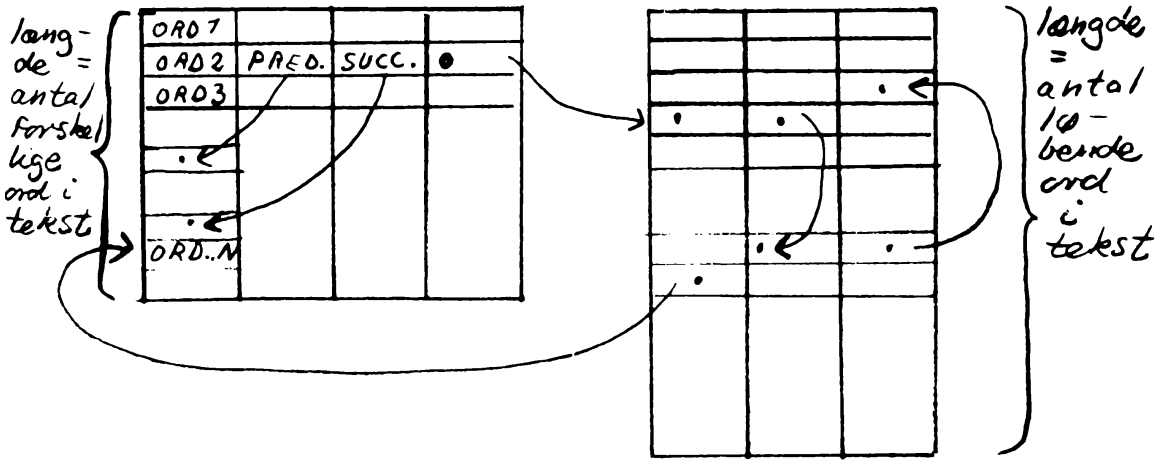
I forhold til ordbøger repræsenterer brugen af konkordanser en anden lingvistisk beskrivelsesmåde. I stedet for undertiden ukommenterede oversættelsesækvivalenter (tosprogede ordbøger)[5] eller forklaringer og definitioner, som kan være perifrastiske eller cirkelprægede, benytter oversætteren sig af parallelle og ækvivalente kontekster (tekstudsnit), hvori de pågældende ord forekommer. Det karakteristiske herved er, at oversætteren gennem sin sproglige kompetence foretager et induktivt ræsonnement ved at slutte (inducere) fra de enkelte forekomster til reglen (oversættelsen). Man kan også udtrykke det på den måde, at mens ordbogen er konceptualistisk/definitorisk (langue-plan), er konkordansen perceptiv (parole-plan).

3. REFTEX-systemet (programmer)

REFTEX består af to programpakker.

I den første, som anvendes efter indskrivning af referenceteksterne på en fil, opbygges der en binær datastruktur for hver enkelt tekst. Den færdige datastruktur består af to poster (records). I den ene registreres tekstens forskellige ord, hvert ords successor og predecessor i træet/datastrukturen samt den første forekomst (occurrence) i den løbende tekst af hvert enkelt ord. I den anden registreres hvert løbende ord i teksten. For det første dets placering i træet/datastrukturen (i post 1), for det andet, hvor eventuelt næste forekomst af ordet findes, og for det tredje en registrering af begyndelsesposition for det afsnit, hvori det pågældende ord findes.

De enkelte posters indbyrdes relation fremgår af figur 1:



Den færdige træstruktur gemmes én gang for alle på en fil. Formålet med at opbygge denne træstruktur er at forkorte søgetiden (højere 'retrievability') samt at give yderligere information om teksten (uden samtidig at miste eksisterende), f.eks. om, hvor i teksten et ord forekommer, dets absolutte og/eller relative frekvens samt hvor det eventuelt findes næste gang.

Den anden del, som er den del af systemet, som en oversætter/bruger normalt har kendskab til, omfatter tre dele:

a) oversætteren finder de referencetekst-par, som anses for relevante for den pågældende oversættelse, indlæser dem i datamaten samt oplyser, hvilket sprog der er tale om.
b) oversætteren indtaster det ord eller udtryk, hvis kontekst og oversættelse han ønsker at kende.
c) systemet reagerer med udskrift på skærmen af den første forekomst samt den tilsvarende oversættelse, hvis brugbarhed for den aktuelle oversættelse herefter vurderes. Hvis forslaget er acceptabelt, fortsættes til det næste problem (ord/udtryk). Hvis den ikke forekommer anvendelig, fortsættes til næste forekomst og således videre, til der findes en anvendelig eller til alle forekomster (og deres tilhørende oversættelser) er fundet irrelevante og dermed forkastet. I dette tilfælde gemmes søgeordet med henblik på en efterfølgende søgning i et andet referencetekst-par.

Der gives i afsnit 6 nogle eksempler til illustration af systemets funktion.

4. Konkordanser og datamater

Et grundlæggende princip ved REFTEX er som nævnt ovenfor anvendelsen af konkordanser. Som bekendt giver udarbejdelse af konkordanser ved hjælp af datamater nogle problemer. Det er hverken stedet eller hensigten her at give en oversigt over sådanne konkordansteoretiske problemer,[6] men alene at pege på, hvordan problemerne er søgt løst i REFTEX. Der skal her peges på tre problemer:

Det første er kontekstens længde. Så længe der er tale om ensprogede konkordanser over poesi eller versificeret prosa, kan problemet stort set ignoreres, idet man nøjes med udskrift af den linje, hvori ordet forekommer. Anderledes forholder det sig her, hvor der er tale om (sag)prosatekst. Problemet er løst i REFTEX ved en simpel forudredigering, der består i indsætning af såkaldt nummererede afsnits-(el. konkordans) markører, der ligeledes muliggør en entydig identifikation af den tilsvarende passage i målsprogsreferenceteksten.

Det andet problem er, hvad man har kaldt "word-form diffusion", dvs. det

forhold, at et ord kan have flere bøjningsformer udover grundformen, lemmaet. Den oftest anvendte løsning består i trunkering af endelser i datastrukturen og efterfølgende sammenligning af søgeord og trunkeret ord. I REFTEX er det konventionelt grundformen, der indtastes, hvorefter regelmæssige bøjningsendelser (desinenser) tilføjes de produktive ordklasser. De på den måde genererede bøjningsformer søges herefter i referenceteksten. Et sådant system er udarbejdet for fransk og spansk (for substantiver, adjektiver, verber og participier).

Det tredje og sidste problem, der skal omtales her, er det, der kaldes "homoform insentivity", hvormed der tænkes på det forhold, at ét og samme ord kan have flere betydninger. Der er med andre ord homografi- og polysemiproblemet. Dette problem kan kun løses helt gennem en forudgående præ-editering, som ofte viser sig ufuldkommen, eller ved en form for parsing. Ingen af disse løsninger er anvendt i REFTEX. I stedet bruges der en pragmatisk løsning, som i det mindste i en række tilfælde gør det muligt at komme uden om problemet. Den består i muligheden af at søge et andet ord samtidig med det første altså en slags kookkurrens. Det skal dog tilføjes, at en sådan løsning formentlig vil være anvendelig alene ved en ad hoc anvendelse af konkordanser som her og ikke, hvis endemålet er udarbejdelse af en færdig konkordans.

5. REFTEX's anvendelsesområder

Som det vil være fremgået af det foregående, er REFTEX særlig udviklet som et supplement til eksisterende hjælpemidler for en oversætter.

Som sådan foreligger det i en færdig og anvendelig form. Alligevel kan man her forestille sig systemet udbygget i retning af, hvad jeg her vil betegne som en tekstbank. En tekstbank er et bibliotek af referencetekster (i den her anvendte betydning) om et antal forskellige emner og på et givet antal sprog. Desuden kan man forestille sig REFTEX integreret i et større system, hvor det ved hjælp af avanceret tekstbehandling og specielle skærme vil være muligt at foretage samtlige aspekter af oversættelsesarbejdet fra skærmterminalen, herunder, at skærmen opdeles i felter med tekst, der skal oversættes, oversat tekst samt diverse informationer fra termbank, tekstbank osv.

Et af de principper, som REFTEX bygger på, er som nævnt tosprogede konkordanser. Jeg vil afslutningsvis pege på et par områder, hvor dette princip efter min opfattelse også skulle kunne finde anvendelse.

Man kan således undersøge, om tosprogede konkordanser kan anvendes til at bestemme en oversættelses egalitet, dvs. hvorvidt et givet ord eller udtryk i kildesproget konsekvent er oversat til et bestemt ord/udtryk i målsproget. Det

antages her, at det f.eks. i tekniske tekster er ønskeligt at opnå en relativt høj egalitetsgrad.

Det kan endvidere undersøges, i hvilket omfang tosprogede konkordanser vil kunne anvendes ved oversættelsesundersøgelser. Ved hjælp af en opstilling af sammenlignelige fakta, hvormed menes en parallelisering af korrespondenser for et bestemt syntaktisk område, i to sprog antages det, at man kan udarbejde oversættelsesregler, dvs. regler for, hvilken/hvilke ækvivalent(er) i et målsprog der svarer til et bestemt syntaktisk fænomen i et kildesprog. Endemålet skulle her være en nærmere bestemmelse af, hvornår en bestemt oversættelsesmulighed foretrækkes frem for potentielle andre og således føre til uddragelse af regler herfor.

En sådan hypotese forudsætter naturligvis, at konteksten, som er det, der især kan studeres ved konkordanser, er om ikke afgørende for - så dog i det mindste har stor indflydelse på valget af denne eller hin oversættelse af et bestemt sprogligt fænomen. Konteksten er jo - udover menneskets sunde fornuft og situationsfornemmelse/viden om verden - det hjælpemiddel, vi bruger for at løse de flertydigheder, der findes i ethvert sprog.

6. Eksempler på kørsler:

De følgende eksempler, der er taget fra en kørsel med en mindre tekst til testning af programmerne, er skrevet ud på papir.

Store bogstaver: Maskinens spørgsmål til bruger (u/kursiv) samt udskrift af svar/konkordanser (m/kursiv)

Små bogstaver: Brugers svar til maskinen (u/kursiv) samt kommentarer (kursiv)

6.1 Kørsel med program ARBOR

Det antages, at der alene opbygges to binære træstrukturer, én for en kildesprogstekst, en anden for en målsprogstekst (oversættelse af den første):

I DETTE PROGRAM ETABLERES TRÆSTRUKTURER PÅ GRUNDLAG AF TEKSTFILER
HVOR MANGE TEKSTFILER SKAL INDLÆSES I EN BINÆR TRÆSTRUKTUR?
2
HVAD HEDDER DEN 1. TEKSTFIL, DER SKAL INDLÆSES?
tex44dk *(opbygning af denne træstruktur)*
HVAD HEDDER DEN 2. TEKSTFIL, der skal indlæses?
tex44fr *(opbygning af træstruktur for tex44fr)*
NU FORELIGGER SAMTLIGE TEKSTFILER SOM BINÆRE TRÆSTRUKTURER PÅ HVER SIN FIL
DISSE FILER ARKIVERES NU UNDER DERES RESPEKTIVE NAVNE
TEX44DK ARCHIVED

TEX44FR ARCHIVED
SLUT PÅ PROGRAM ARBOR

6.2 Kørsel med program REFTEX

6.2.1 Søgning af et ord med efterfølgende udskrift

6.2.2 Søgning af et ord med suffiksgenerering og arkivering

6.2.3 Søgning af et ord med kookkurrens (samt suffiksgenerering)

6.2.4 Søgning af et ord med kookkurrens (uden suffiksgenerering)

6.2.5 Søgning af arkiveret ord i en alternativ referencetekst

PROGRAM REFTEX

DER BEGYNDER HER ER ET HJÆLPEMIDDEL TIL BRUG VED OVERSÆTTELSE
PROGRAMMET ER INTERAKTIVT, DVS. AT DE SKAL SVARE PÅ DE SPØRGS-
MÅL, SOM STILLES UNDER PROGRAMAFVIKLINGEN(HUSK AT SKRIVE KOMMA
MELLEM DE ENKELTE SVAR).
HVAD HEDDER DE REFERENCETEKSTER, SOM SKAL ANVENDES NU?
SKRIV FØRST NAVN PÅ KILDESPROGSFIL, OG DERNÆST PÅ MÅLSPROGSFIL
tex25fr, tex25dk
HVILKET SPROG ER KILDESPROGSTEKSTEN SKREVET PÅ? (FRANSK, SPANSK)
fransk
6.2.1 SKRIV SØGEORD OG ORDTYPE (SUBSTANTIV, ADJEKTIV, VERBUM,
PARTICIPIUM ELLER ANDET):
activité, substantiv
ACTIVITE, DER FINDES 4 GANGE I TEKST TEX44FR, ER NU FUNDET.
OPTRÆDER *ACTIVITE* SAMMEN MED ET ANDET ORD?
SKRIV JA ELLER NEJ
nej
NU UDSKRIVES DEN 1. FOREKOMST AF *ACTIVITE*:
DEPUIS 1962, CETTE BRANCHE D'ACTIVITE A PERDU 20% DE SES SALARIES
NU UDSKRIVES PARALLELT AFSNIT I MÅLSPROGSTEKSTEN:
DENNE INDUSTRIGREN HAR MISTET 20% AF SINE ANSATTE SIDEN 1962.
ER RESULTATET TILFREDSSTILLENDE? SKRIV JA ELLER NEJ
ja
FINDES DER ANDRE SØGEORD? (SKRIV JA ELLER NEJ)
ja
6.2.2 SKRIV SØGEORD OG ORDTYPE (SUBSTANTIV, ADJEKTIV, ... se ovf.):
note, substantiv *(udtryk, der søges: note explicative)*
NOTE FOREKOMMER DESVÆRRE IKKE I REFERENCETEKSTEN
SKAL DER FORETAGES SUFFIKSGENERERING? (SKRIV JA ELLER NEJ)
ja
ER SUFFIKSDANNELSEN REGELMÆSSIG? (SKRIV JA ELLER NEJ)
ja
SØGEORDET ER *NOTES*
NOTES FOREKOMMER DESVÆRRE IKKE I REFERENCETEKSTEN
SUFFIKSGENERERING HAR FUNDET STED - DESVÆRRE UDEN RESULTAT
VIL DE SELV INDTASTE EN ANDEN FORM? *(med henblik på eventuelt*
nej *uregelmæssige former)*
SKRIV DEN FORM, DER SKAL GEMMES TIL SENERE SØGNING:
note
NU GEMMES *NOTE* MED HENBLIK PÅ SENERE SØGNING I EN ANDEN TEKST
FINDES DER ANDRE SØGEORD?
ja

6.2.3 SKRIV SØGEORD OG ORDTYPE:
effet, substantiv *(udtryk, der søges: effets secondaires)*
EFFET, DER FINDES 25 GANGE I TEKST TEX25FR, ER NU FUNDET
OPTRÆDER *EFFET* SAMMEN MED ET BESTEMT ANDET ORD?
ja
SKRIV DETTE ORD:
secondaire
SVAR NU PÅ FØLGENDE SPØRGSMÅL (HUSK KOMMA MELLEM SVARENE):
FINDES *SECONDAIRE* FORAN ELLER BAGVED *EFFET*?
HVOR MANGE POSITIONER?
KONGRUERER *SECONDAIRE* MED *EFFET* ?
bagved, 1, ja
DET UNDERSØGES NU, OM *EFFET* OG *SECONDAIRE* FINDES SAMMEN
EFFET OG *SECONDAIRE* OPTRÆDER DESVÆRRE IKKE SAMMEN I TEKSTEN
SKAL DER FORETAGES SUFFIKSGENERERING?
ja
ER SUFFIKSDANNELSEN REGELMÆSSIG?
ja
SØGEORDET ER *EFFETS*
SKRIV DEN KONGRUEREDE FORM AF FØLGEORDET:
secondaires
EFFETS, DER FINDES 13 GANGE I TEKST TEX25FR, ER NU FUNDET
DET UNDERSØGES NU, OM *EFFETS* OG *SECONDAIRES* FINDES SAMMEN
NU UDSKRIVES DEN 6. FOREKOMST AF *EFFETS*:
IL EST DIFFICILE D'ABOUTIR A UNE HARMONISATION COMPLETE DES INDICATIONS THERAPEUTIQUES, CONTRE-INDICATIONS, AVERTISSEMENTS ET EFFETS SECONDAIRES, ETANT DONNE QUE DES POSITIONS DIFFERENCIEES ONT ETE PRISES POUR DES PRODUITS SIMILAIRES
NU UDSKRIVES PARALLELT AFSNIT I MÅLSPROGSTEKSTEN:
DET ER VANSKELIGT AT NÅ FREM TIL EN FULDKOMMEN HARMONISERING AF TERAPEUTISKE INDIKATIONER, ADVARSLER OG BIVIRKNINGER, EFTERSOM DER ER TAGET FORSKELLIGE BESLUTNINGER OM PRODUKTER, DER LIGNER HINANDEN
ER RESULTATET TILFREDSSTILLENDE?
nej
NU UDSKRIVES DEN 9. FOREKOMST AF *EFFETS*:
ON ENTEND PAR EFFETS SECONDAIRES LES RESULTATS D'UN PRODUIT PHARMACEUTIQUE QUI NE SONT VISES DIRECTEMENT
NU UDSKRIVES PARALLELT AFSNIT I MÅLSPROGSTEKSTEN:
VED BIVIRKNINGER FORSTÅS SÅDANNE VIRKNINGER AF ET LÆGEMIDDEL, SOM IKKE ER DIREKTE TILSIGTET
ER RESULTATET TILFREDSSTILLENDE?
ja
FINDES DER ANDRE SØGEORD?
ja
6.2.4 SKRIV SØGEORD OG ORDTYPE:
application, substantiv *(udtryk, der søges: en application de)*
APPLICATION, DER FINDES 5 GANGE I TEKSTEN, ER NU FUNDET
OPTRÆDER *APPLICATION* SAMMEN MED ET ANDET ORD?
ja
SKRIV DETTE ORD:
en
SVAR NU PÅ FØLGENDE SPØRGSMÅL (HUSK KOMMA MELLEM SVARENE):
FINDES *EN* FORAN ELLER BAGVED *APPLICATION*
HVOR MANGE POSITIONER?
KONGRUERER *EN* MED *APPLICATION*
foran, 1, nej

DET UNDERSØGES NU, OM *EN* OG *APPLICATION* FINDES SAMMEN
NU UDSKRIVES DEN 3. FOREKOMST AF *APPLICATION*:
CE MEME ARTICLE PRESCRIT A LA COMMISSION DE FAIRE RAPPORT AUX CONSEIL SUR LE FONCTIONNEMENT DU COMITE, POUR LA PREMIERE FOIS DEUX ANS APRES L'ENTREE EN APPLICATION DE LA DIRECTIVE
NU UDSKRIVES PARALLELT AFSNIT I MÅLSPROGSTEKST:
I SAMME ARTIKEL FASTSÆTTES, AT KOMMISSIONEN SKAL AFLÆGGE BERETNING FOR RÅDET OM, UDVALGETS VIRKSOMHED, FØRSTE GANG TO ÅR EFTER DIREKTIVETS IKRAFTTRÆDEN
ER RESULTATET TILFREDSSTILLLENDE?
nej *(eksemplet viser, at skønt mængden af information begrænses kan der alligevel optræde homografer. For at modvirke det, skal der laves søgning af tosidig kookkurrens.)*
NU UDSKRIVES DEN 4. FOREKOMST AF *APPLICATION*:
LE COMITE A RECU DEUX DEMANDES D'AUTORISATION DE MISE SUR LE MARCHE EN APPLICATION DE L'ARTICLE 9.
NU UDSKRIVES PARALLELT AFSNIT FRA MÅLSPROGSTEKSTEN:
UDVALGET HAR MODTAGET TO ANSØGNINGER OM MARKEDSFØRINGSTILLADELSE I MEDFØR AF ARTIKEL 9 I DIREKTIVET.
ER RESULTATET TILFREDSSTILLENDE?
ja
FINDES DER ANDRE SØGEORD?
nej
FØLGENDE SØGEORD HAR IKKE KUNNET FINDES ELLER IKKE KUNNET UDSKRIVES MED EN TILFREDSSTILLENDE KONTEKST FRA REFERENCETEKST TEX44FR: NOTE
FINDES DER ANDRE REFERENCETEKSTER?
ja
DE RESTERENDE SØGEORD SØGES FUNDET I EN ANDEN REFERENCETEKST.
HVAD HEDDER DE REFERENCETEKSTER, DER SKAL ANVENDES NU?
SKRIV FØRST NAVN PÅ KILDESPROGSFIL OG DERNÆST PÅ MÅLSPROGSFIL
tex44fr, tex44dk
6.2.5 DET NÆSTE SØGEORD ER *NOTE*, SOM ER ET SUBSTANTIV
NOTE, DER FOREKOMMER 3 GANGE I TEKST TEX44FR, ER NU FUNDET:
OPTRÆDER *NOTE* SAMMEN MED ET ANDET ORD?
ja
SKRIV DETTE ORD
explicative
svar nu på følgende spørgsmål (HUSK KOMMA MELLEM SVARENE):
FINDES *EXPLICATIVE* FORAN ELLER BAGVED *NOTE*?
HVOR MANGE POSITIONER?
KONGRUERER *EXPLICATIVE* MED *NOTE*?
bagved, 1, ja
DET UNDERSØGES NU, OM *NOTE* OG *EXPLICATIVE* FINDES SAMMEN
NOTE OG *EXPLICATIVE* OPTRÆDER DESVÆRRE IKKE SAMMEN I TEKSTEN
SKAL DER FORETAGES SUFFIKSGENERERING?
ja
ER SUFFIKSDANNELSEN REGELMÆSSIG?
ja
SØGEORDET ER *NOTES*
SKRIV DEN KONGRUEREDE FORM AF FØLGEORDET:
explicatives
NOTES, DER FINDES 5 GANGE I TEKST TEX44FR, ER NU FUNDET
DET UNDERSØGES NU, OM *NOTES* OG EXPLICATIVES* FINDES SAMMEN
NU UDSKRIVES DEN 3. FOREKOMST AF *NOTES*:
LA PROCEDURE SUIVANTE A JUSQU'A MAINTENANT ETE SUIVIE AFIN DE PROCEDER AUX CONSULTATIONS LES PLUS ETENDUES: LES GROUPES D'EX-

PERTS ELABORENT DES NOTES EXPLICATIVES, DE CARACTERE SCIENTIFIQUE TENANT COMPTE DE TOUTES LES DONNEES DISPONIBLES QUI PEUVENT ETRE RECUEILLIES
NU UDSKRIVES PARALLELT AFSNIT I MÅLSPROGSTEKSTEN:
HIDTIL HAR MAN FULGT FØLGENDE FREMGANGSMÅDE MED HENBLIK PÅ AT HØRE FLEST MULIGT: EKSPERTGRUPPERNE UDARBEJDER FORKLARENDE BEMÆRKNINGER AF VIDENSKABELIG ART UNDER HENSYNTAGEN TIL ALLE FORELIGGENDE OPLYSNINGER
ER RESULTATET TILFREDSSTILLENDE?
ja
NU ER SAMTLIGE SØGEORD FUNDET OG UDSKREVET MED EN TILFREDSSTILLENDE KONTEKST.
SLUT PÅ PROGRAM REFTEX.

Noter:

1. Warren Weaver (pp. 15-23) in Locke and Booth (eds.): Machine Translation of Languages. N.Y. 1965.
2. Krollmann in Overcoming the Language Barrier. München 1978.
3. Se f.eks. John Dawson: A multi-Language Multi-text Concordance as an Aid in Manuscript Study (pp. 21-8) in Computers and the Humanities 14.1 (1980) Fluching N.Y.
   og Susan Hockey, V. Shibayev: The Bilingual Literary and Linguistic Concordance Balcon pp. 133-9 in ALLC Bulletin 3.2 (1978). Stockport.
4. Se f.eks. Peter Arthern: Machine Translation and Computerized Terminology Systems; a Translators viewpoint pp. 77-109 in Barbara Snell (ed.): Translation and the Computer. Den Haag 1979.
   Concetta Carestia-Greenfield et Daniel Serain: La traduction assisté par ordinateur: Des banques de terminologie aux systèmes interactifs de traduction. Paris 1976.
5. Jvf. Jean Rey: Préface au Petit Robert, dernière édition.
6. Se f.eks.: T.H. Howard-Hill: Literary Concordances, A Complete Handbook for the Preparation of Manual and Computer Concordances. Oxford 1979.
   Suzanne Hanon: Mots dans le texte, mots hors du texte: réflexions méthodologiques sur quelques index et concordances appliqués à des œvres française, italienne et espagnole (pp. 272-96) in Revue Romane 12 (1977) Copenhague.
   id.: Ordbøger, konkordanser og lemmatisering, pp. 89-102 in Nordiske Datalingvistikdage i København 1979.
   Dimitri J. Kourbourlis: From a word-form concordance to a dictionary-form concordance. pp. 225-33 in J.L. Mitchell (ed.): Computers in the Humanities. Edinburgh 1974.

Reference:

Poul Søren Kjærsgaard: Oversættelse og edb. Specialeafhandling. Romansk Institut Aarhus Universitet 1981.

Gregers Koch
KøbenHavns Universitet

COMPUTATIONAL LINGUISTICS AND MATHEMATICAL LOGIC FROM A COMPUTER SCIENCE POINT OF VIEW

Let us look at a special type of context sensitive grammars, the so-called Definite Clause Grammars, or DCG, as described by Pereira and Warren (1980). The following question seems natural here:

- Can every context sensitive language be described by means of a definite clause grammar ?

The question must be answered in the affirmative, as can be seen by simulating Turing machines or better Markov algorithms by means of definite clause grammars. This means that the definite clause grammars have at least the same computational power as the grammars of Chomsky type 0. And hence theoretically speaking they can also handle languages of Chomsky type 1.

If we focus on the question which practical methods have been developed to handle context sensitive languages by means of definite clause grammars or similar formalisms then the situation is much more unclear. It seems to be a problem with a considerable actual research effort. We may see the occuring of several new grammar types as the results of exactly this tendency, for instance Gapping Grammars in (Dahl & Abramson 1984, Dahl 1984) and Puzzle Grammars (Sabatier 1984).

The next natural and fundamental question seems to be the following

- Does there exist any known problems within the languages that can be described by means of definite clause grammars ?

It should be answered in the affirmative, actually there exist many problems. As an example we may look at possibly the most famous problem, the coordination problem.

The text that we want to examine consists simply of the Danish sentence

- Peter ønsker ikke at spise et æble.
  (Peter does not want to eat an apple)

What should be the output or the result of the process corresponding to this sentence as input? It should be a sort of semantic representation providing a reasonable picture of the meaning of the statement. We tend to argue for the use of logical representations, and also here in the choice of semantic representation, so the notion logico-semantic representation seems to be justified.

Are we able to construct an automatic translation of an appropriate sublanguage into such a logico-semantic representation ? Yes, for example it can be done by a Prolog program like the one described in the appendix. (Presumably we wanted a DCG describing the same translation, but in the used version of Micro-Prolog there is no support of DCGs).

During such an execution the result variable obtains a value to be output like this:

```
(Not
  (Oensker Peter (Exist
      x
      ((x
          Is
          Aeble) And (Spise
          Peter
          x)))))
```

which corresponds to the following logical formula in a more traditional notation:

$\neg$ Ønsker(Peter,$\exists$ x(Æble(x) & Spiser(Peter,x))).

We notice that this seems to be a socalled intensional reading, and that Montague grammars here show themselves (Montague 1974). Perhaps we rather wanted a socalled extensional reading like the following formula

$\forall$ x(Æble(x) $\rightarrow$ $\neg$ Ønsker(Peter,Spiser(Peter,x))).

The problem here seems to be due to scope problems, and this is exactly the wanted example of the coordination problem.

Another reasonable question is: what do the simplest logico-semantic assignments (or representation transformations) look like ? That question seems to have a trivial answer, namely the constant assignment functions mapping everything into e.g. false. So we should probably modify the question a little:

- How can we characterize the simplest non-constant logico-semantic assignments ?

To give a reasonable answer to this question we should discuss further what does the term "simplest" mean in this context.

Let us suppose that simplicity involves Frege's principle for referential transparency.

Under the asumption of the principle of referential transparency we can argue that a particular assignment is the simplest possible one, viz. an assignment giving a mixture of lambda calculus expressions and predicate calculus expressions as the logico-semantic representation.

It seems justified to call this representation a denotational semantic representation, in the style of (Milne & Strachey 1976, Gordon 1979).

Before presenting this denotational semantic assignment function called Sem, it seems appropriate to look at yet another little grammar G2 and its treatment by means of definite clause grammars. (Figure 3)

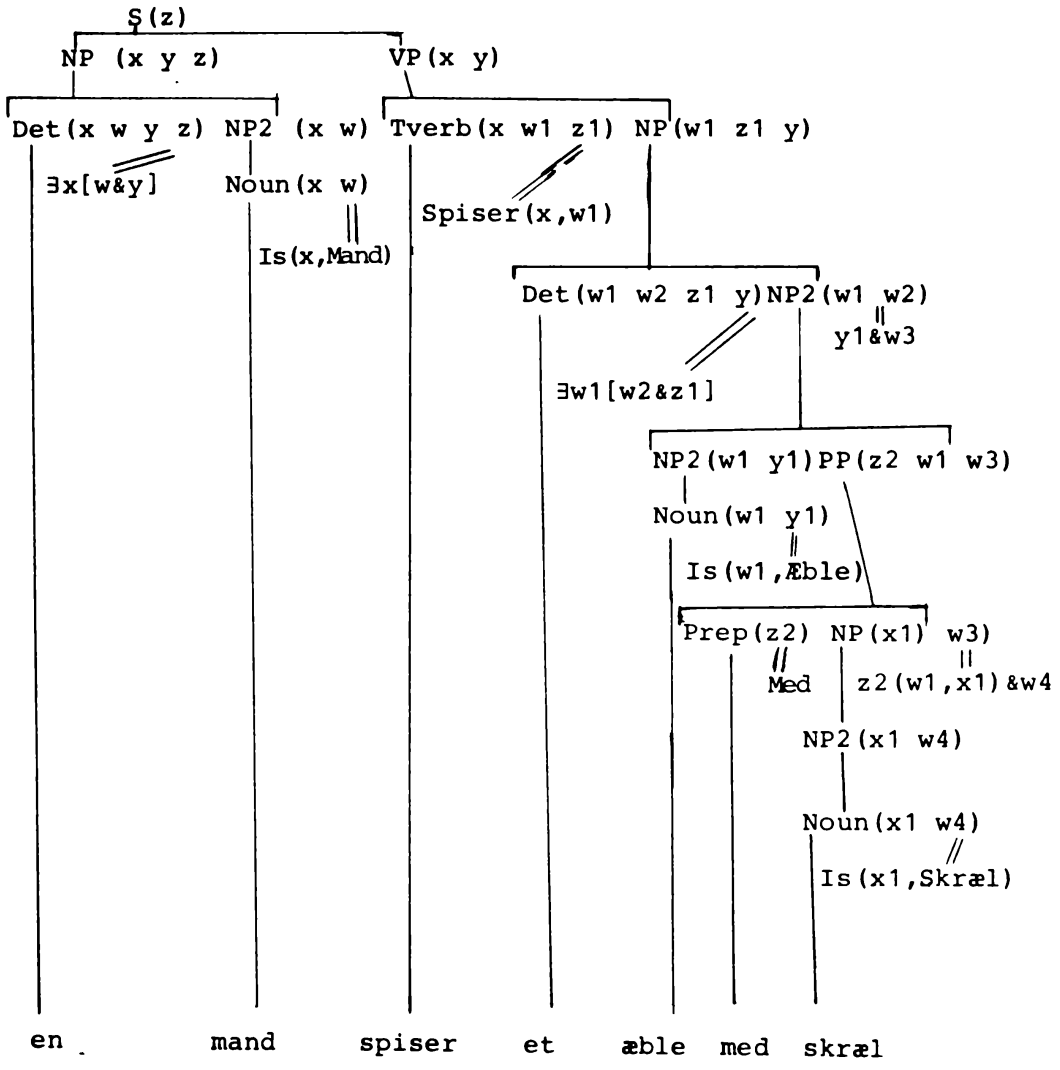
Perhaps we ought to show the system's treatment of a few selected sentences. The sentences are the following:

- En mand spiser et æble med skræl
  (a man eats an apple with the peel)

- Enhver ung mand elsker en smuk kvinde.
  (every young man loves a beatiful woman)

See the figures 1 and 2.

en mand spiser et æble med skræl



dvs.

$z = \exists x[w\&y]$

$= \exists x[w\&\exists w1[w2\&z1]]$

$= \exists x[w\&\exists w1[y1\&w3\&z1]]$

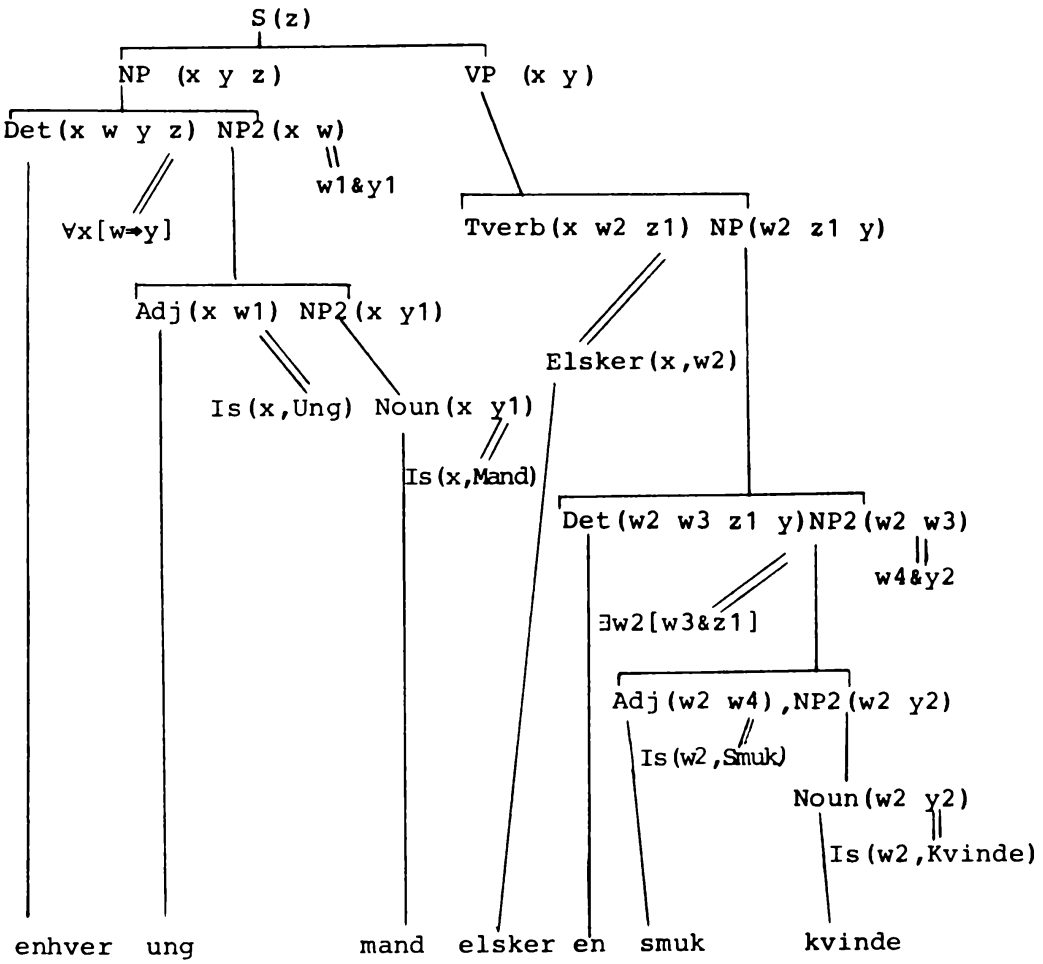
$= \exists x[Is(x,Mand)\&\exists w1[Is(w1,Æble)\&w3\&Spiser(x,w1)]]$

$= \exists x[Is(x,Mand)\&\exists w1[Is(w1,Æble)$
$\& z2(w1,x1)\&w4 \& Spiser(x,w1)]]$

$= \exists x[Is(x,Mand)\&\exists w1$
$[Is(w1,Æble)\&Med(w1,x1)\&Is(x1,Skræl)\&Spiser(x,w1)]]$

Figure 1

enhver ung mand elsker en smuk kvinde



dvs.

$z = \forall x[w \Rightarrow y]$

$= \forall x[w1 \& y1 \Rightarrow y]$

$= \forall x[w1 \& y1 \Rightarrow \exists w2[w3 \& z1]]$

$= \forall x[w1 \& y1 \Rightarrow \exists w2[w4 \& y2 \& z1]]$

$= \forall x[Is(x,Ung) \& Is(x,Mand) \Rightarrow \exists w2[Is(w2,Smuk) \& Is(w2,Kvinde) \& Elsker(x,w2)]]$

$= \forall x[Is(x,Ung) \& Is(x,Mand) \Rightarrow \exists [Is(y,Smuk) \& Is(y,Kvinde) \& Elsker(x,y)]]$

Figure 2

Now let us return to the problem to construct the simplest possible logico-semantic assignment function. We can make two assertions here:

Assertion A:

The following logico-semantic assignment function Sem in the grammar G2 is the simplest non-constant assignment function according to the mentioned criteria.

Assertion B:

The logico-semantic representation of a text by means of this denotational semantic assignment function Sem is identical to the corresponding logico-semantic representation of the text in the grammar G2 (which is the output corresponding to the result variable z when executing the definite clause grammar corresponding to G2).

The assertion A may be demonstrated to hold by always choosing the simplest possible alternatives.

The assertion B may be proven formally (though not possible within the limits of this paper), but the assertion may also be accepted by looking at a few examples. Let us try, using the example grammar G2 and the small texts above.

```
S    → NP      VP
NP   → Det     NP2
NP   → NP2
NP2  → Noun
NP2  → Adj     NP2
NP2  → NP2     PP
PP   → Prep    NP
VP   → IVerb
NP   → TVerb   NP
VP   → VP      PP
Det  → enhver
Det  → ethvert
Det  → en
Det  → et
Det  → hvilken
Det  → hvilket
```

Figure 3

$$
\begin{aligned}
&\text{Sem("en mand spiser et æble med skræl")}\\
=\ &\text{Sem(en}\hat{}\text{mand}\hat{}\text{spiser}\hat{}\text{et}\hat{}\text{æble}\hat{}\text{med}\hat{}\text{skræl)}\\
=\ &\text{Sem(D1}\hat{}\text{N1}\hat{}\text{Tv}\hat{}\text{D2}\hat{}\text{N2}\hat{}\text{P}\hat{}\text{Nart)}\\
=\ &\text{Combine}_{[D,N,Tv,D,N,P,Nart]}(\text{Sem(D1),Sem(N1)Sem(Tv),Sem(D2),}\\
&\qquad\qquad\text{Sem(N2),Sem(P),Sem(Nart))}\\
=\ &\text{Sem(D1)(Sem(N1),}\lambda\text{t.[Sem(D2)(Sem(N2)\&\&(Sem(P)oSem(Nart)),}\\
&\qquad\qquad\text{First(Sem(Tv),t)])}\\
=\ &\text{Sem(D1)(Sem(N1),}\lambda\text{t.[Sem(D2)(Sem(N2)\&\&(Sem(P)oSem(Nart)),}\\
&\qquad\qquad\text{First(}\lambda\text{(u,v).Tv(u,v),t))])}\\
=\ &\text{Sem(D1)(Sem(N1),}\lambda\text{t.[Sem(D2)(Sem(N2)\&\&(Sem(P)oSem(Nart)),}\\
&\qquad\qquad\lambda\text{v.Tv(t,v))])}\\
=\ &\text{Sem(D1)(Sem(N1),}\lambda\text{t.[(}\lambda\text{(u,w).}\exists\text{y[u(y)\&w(y)])}\\
&\qquad\text{((}\lambda\text{v.Is(v,N2))\&\&((}\lambda\text{(s,t).P(s,t))o(Nart)),}\lambda\text{v.Tv(t,v))])}\\
=\ &\text{Sem(D1)(Sem(N1)}\lambda\text{t.[(}\lambda\text{(u,w).}\exists\text{y[u(y)\&w(y)])}\\
&\qquad\text{((}\lambda\text{v.Is(v,N2))\&\&(}\lambda\text{s.P(s,Nart)),}\lambda\text{v.Tv(t,v))])}\\
=\ &\text{Sem(D1)(Sem(N1),}\lambda\text{t.[(}\lambda\text{(u,w).}\exists\text{y[u(y)\&w(y)])}\\
&\qquad\text{(}\lambda\text{s.(Is(s,N2)\&P(s,Nart)),}\lambda\text{v.Tv(t,v))])}\\
=\ &\text{Sem(D1)(Sem(N1),}\\
&\qquad\lambda\text{t.}\exists\text{y[(}\lambda\text{s.(Is(s,N2)\&P(s,Nart)))(y)\&(}\lambda\text{v.Tv(t,v))(y)])}\\
=\ &\text{Sem(D1)(Sem(N1),}\\
&\qquad\lambda\text{t.}\exists\text{y[Is(y,N2)\&P(y,Nart)\&Tv(t,y)])}\\
=\ &\text{(}\lambda\text{(u,v).}\exists\text{x[u(x)\&v(x)])}\\
&\qquad\text{(}\lambda\text{s.Is(s,N1),}\lambda\text{t.}\exists\text{y[Is(y,N2)\&P(y,Nart)\&Tv(t,y)])}\\
=\ &\exists\text{x[(}\lambda\text{s.Is(s,N1))(x)}\\
&\qquad\&\text{(}\lambda\text{t.}\exists\text{y[Is(y,N2)\&P(y,Nart)\&Tv(t,y)])(x)]}\\
=\ &\exists\text{x(Is(x,N1)\&}\exists\text{y[Is(y,N2)\&P(y,Nart)\&Tv(x,y)]]}\\
=\ &\exists\text{x(Is(x,Mand)\&}\exists\text{y[Is(y,Æble)\&Med(y,Skræl)\&Spiser(x,y)]].}
\end{aligned}
$$

```
  Sem("enhver ung mand elsker en smuk kvinde")
= Sem(enhver^ung^mand^elsker^en^smuk^kvinde)
= Sem(D1^A1^N1^Tv^D2^A2^N2)
= Combine[D,A,N,Tv,D,A,N](Sem(D1),Sem(A1),Sem(N1),Sem(Tv),
                          Sem(D2),Sem(A2),Sem(N2))

= Sem(D1)(Sem(A1)&&Sem(N1),
        λt.(Sem(D2)(Sem(A2)&&Sem(N2),First(Sem(Tv),t))))
= Sem(D1)(Sem(A1)&&Sem(N1),
        λt.(Sem(D2)(λu.Is(u,A2)&&λv.Is(v,N2),
                                First(λ(x,y).Tv(x,y),t))))
= Sem(D1)(Sem(A1)&&Sem(N1),
        λt.(Sem(D2)(λu.(Is(u,A2)&Is(u,N2)),λy.Tv(t,y))))
= Sem(D1)(Sem(A1)&&Sem(N1),
        λt.((λ(x,z).∃v[x(v)&z(v)])(λu.(Is(u,A2)&Is(u,N2)),
                                          λy.Tv(t,y))))

= Sem(D1)(Sem(A1)&&Sem(N1),
        λt.(∃v[(λu.(Is(u,A2)&Is(u,N2)))(v)&(λy.Tv(t,y))(v)]))
= (λ(y,z).∀x[y(x)➡z(x)])(λu.Is(u,A1))&&(λv.Is(v,N1))),
        λt.(∃v[Is(v,A2)&Is(v,N2)&Tv(t,v)]))
= (λ(y,z).∀x[y(x)➡z(x)])(λu.(Is(u,A1)&Is(u,N1)),
        λt.(∃v[Is(v,A2)&Is(v,N2)&Tv(t,v)]))
= ∀x[(λu.(Is(u,A1)&Is(u,N1)))(x)
        ➡(λt.(∃v[Is(v,A2)&Is(v,N2)&Tv(t,v)]))(x)]
= ∀x[Is(x,A1)&Is(x,N1)➡∃v[Is(v,A2)&Is(v,N2)&Tv(x,v)]]


= ∀x[Is(x,Ung)&Is(x,Mand)
        ➡ ∃v[Is(v,Smuk)&Is(v,Kvinde)&Elsker(x,v)]]
```

References

Allwood, J. et al. "Logic in Linguistics", Cambridge University Press, 1977.

Cresswell, M. "Logics and Languages", Methuen, 1973.

Montague, R. "Formal Philosophy: Selected Papers of Richard Montague", ed. R. Thomason, Yale University Press, 1974.

Bernth, A. "Logic Programming and Translation Between Natural Languages", NordDATA Proceedings 3, 727-731, 1984, Helsinki.

Bernth, A. "Indføring i Prolog for Lingvister", noter, Datalogisk Institut ved Københavns Universitet, 1985.

Blikle, A. "Applied Denotational Semantics", lecture notes, Polish Academy of Sciences, Warsaw, 1985.

Blikle, A. & Tarlecki, A. "Naive Denotational Semantics", Proc. IFIP 1983, Paris, 345-356.

Charniak,E. & Wilks, Y.(eds.), "Computational Semantics", North-Holland, 1976.

Clocksin, W.F. & Mellish, C.S. "Programming in Prolog", Springer-Verlag, 1981.

Dahl, V. "More on Gapping Grammars", Proc.Fifth Gener. Comp. Systems 1984, Tokyo, p. 669 ff.

Dahl, V. & McCord, M. "Treating Coordination in Logic Grammars", Am. Journ. Comp. Ling. 1983.

Dahl, V. & Abramson, H. "On Gapping Grammars", Proc. Intern. Conf. Logic Programming, 1984, 77-88.

Gordon, M. J. C. "The Denotational Description of Programming Languages: An Introduction", Springer, 1979.

Hopcroft J.E. & Ullman, J.D. "Formal Languages and Their Relation to Automata", Addison-Wesley, 1969.

Koch, G. "Stepwise Development of Logic Programmed Software Development Methods", DIKU report 83/5, 1983.

Koch, G.(ed.), "Fifth Generation Programming vol. 1: Logic Programming in Natural Language Analysis", Proceedings of Workshop in Copenhagen, Dec. 1984, DIKU report 85/2.

Koch, G. "Who is a Fallible Greek in Logic Grammars", p.54-77 in (Koch 85 a), 1985 b.

Milne, R.E. & Strachey, C."A Theory of Programming Language Semantics", Chapman and Hall, 1976.

Pereira F. & Warren, D. "Definite Clause Grammars for Language Analysis - A Survey of the Formalism and a Comparison with Augmented Transition Networks", Artificial Intelligence, 13,3 (1980), 231-278.

Sabatier, P. "Puzzle Grammars", Proc. Natural Language Understanding and Logic Programming, Rennes, 1984, 121 - 136.

Sabatier, P. "Des Grammaires de Metamophose aux Grammaires de Puzzle", Copenhagen Workshop on Logic Programming in Natural Language Analysis, 1984.

Saint-Dizier, P. "On Syntax and Semantics of Modifiers in Natural Language Sentences" in (Koch 85 a), 1985.

Appendix

```
Dt (X Y Z) if
          Ds (X Y Z)
Dt ((Which X (Y And Z)) x y) if
          c (hvilken x z) and
          Dn (X Y z X1) and
          Dvp (X Z X1 y)
Dt ((Which X (Y And Z)) x y) if
          c (hvilket x z) and
          Dn (X Y z X1) and
          Dvp (X Z X1 y)
Ds1 (X Y Z) if
          Dnp (x y X Y z) and
          Dvp (x y z Z)
Ds ((X And Y) Z x) if
          Ds1 (X Z y) and
          c (og y z) and
          Ds (Y z x)
Ds (X Y Z) if
          Ds1 (X Y Z)
Ds ((X Imp Y) Z x) if
          c (hvis Z y) and
          Ds (X y z) and
          c (saa z X1) and
          c (gaelder X1 Y1) and
          c (at Y1 Z1) and
          Ds (Y Z1 x)
Dnp (X Y Z x y) if
          Dd (X z Y Z x X1) and
          Dn (X Y1 X1 Z1) and
          Drc (X Y1 z Z1 y)
Dnp (X Y Y Z x) if
          Dprop (X Z x)
Drc (X Y Y Z Z)
Drc (X Y (Y And Z) x y) if
          c (der x z) and
          Dvp (X Z z y)
Drc (X Y (Y And (Not Z)) x y) if
          c (der x z) and
          c (ikke z X1) and
          Dvp (X Z X1 y)
Drc (X Y (Y And Z) x y) if
          c (som x z) and
          Dnp (X1 Y1 Z z Z1) and
          Dtv (X1 X Y1 Z1 y)
Drc (X Y (Y And (Not Z)) x y) if
          c (som x z) and
          Dnp (X1 Y1 Z z Z1) and
          c (ikke Z1 x1) and
          Dtv (X1 X Y1 x1 y)
Dvp1 (X (Not Y) Z x) if
          Dnegvp (X Y Z x)
Dvp1 (X Y Z x) if
          Dposvp (X Y Z x)
```

```
Dvp (X (Y And Z) x y) if
          Dvp1 (X Y x z) and
          c (og z X1) and
          Dvp (X Z X1 y)
Dvp (X Y Z x) if
          Dvp1 (X Y Z x)
Dnegvp (X Y Z x) if
          Dvp-vp (X y Y Z z) and
          c (ikke z X1) and
          c (at X1 Y1) and
          Di (X y Y1 x)
Dnegvp (X Y Z x) if
          Dtv (X y z Z X1) and
          c (ikke X1 Y1) and
          Dnp (y z Y Y1 x)
Dnegvp (X Y Z x) if
          Div (X Y Z y) and
          c (ikke y x)
Dposvp (X Y Z x) if
          Dvp-vp (X y Y Z z) and
          c (at z X1) and
          Di (X y X1 x)
Dposvp (X Y Z x) if
          Dtv (X y z Z X1) and
          Dnp (y z Y X1 x)
Dposvp (X Y Z x) if
          Div (X Y Z x)
Di (X Y Z x) if
          Dti (X y z Z X1) and
          Dnp (y z Y X1 x)
Di (X Y Z x) if
          Dii (X Y Z x)
Di (X Y Z x) if
          Dvp-vpi (X y Y Z z) and
          c (at z X1) and
          Di (X y X1 x)
Dd (X Y Z (All X (Y Imp Z)) x y) if
          c (enhver x y)
Dd (X Y Z (All X (Y Imp Z)) x y) if
          c (ethvert x y)
Dd (X Y Z (Exist X (Y And Z)) x y) if
          c (en x y)
Dd (X Y Z (Exist X (Y And Z)) x y) if
          c (et x y)
Div (X (Y X) Z x) if
          c (Y Z x) and
          Diverbs (y) and
          Y Member y
Dtv (X Y (Z X Y) x y) if
          c (Z x y) and
          Dtverbs (z) and
          Z Member z
```

```
Dvp-vp (X Y (Z X Y) x y) if
         c (Z x y) and
         Dvp-vps (z) and
         Z Member z
Dii (X (Y X) Z x) if
         c (Y Z x) and
         Diinfs (y) and
         Y Member y
Dti (X Y (Z X Y) x y) if
         c (Z x y) and
         Dtinfs (z) and
         Z Member z
Dvp-vpi (X Y (Z X Y) x y) if
         c (Z x y) and
         Dvp-vpinfs (z) and
         Z Member z
Dn (X (X Is Y) Z x) if
         c (Y Z x) and
         Dnouns (y) and
         Y Member y
Dprop (X Y Z) if
         c (X Y Z) and
         Dprops (x) and
         X Member x
Diverbs ((lever blomstrer smiler flyver))
Dtverbs ((elsker spiser er har holder-af bestiger))
Dvp-vps ((oensker haaber))
Diinfs ((leve blomstre smile flyve))
Dtínfs ((elske spise vaere have holde-af bestige))
Dvp-vpinfs ((oenske haabe))
Dnouns ((kvinde aeble mand soft-ice fugl trae medlem skisportsmand...
 menneske ting))
Dprops ((Mary John Peter Prolog Tony Mike sne regn))
```

Kimmo Koskenniemi
Research Unit for Computational Linguistics
University of Helsinki, Hallituskatu 11
SF-00100 Helsinki, Finland

COMPILATION OF AUTOMATA FROM MORPHOLOGICAL TWO-LEVEL RULES

1. INTRODUCTION

The two-level model is a framework for describing word inflection. The model consists of a lexicon system and a formalism for two-level rules. The lexicon system defines all possible lexical representations of word-forms whereas the rules express the permitted relations between lexical and surface representations. Word recognition is thus reduced into the question of finding a permissible lexical representation which is in a proper relation to the surface form. Similarly, generation is the inverse where the lexical representation is known and the task is to find a surface representation which is in a proper relation to it.

Within the two-level model these relations pertaining to the rule component have been expressed in two ways. A rule formalism has been used for communicating the idea of the rules, whereas the actual implementations have been accomplished by hand-coding the rules as finite state automata. The close connection between rules and finite state machines has facilitated this hand-coding.

Expressing rules as numbers in a transition matrix is, of course, not optimal. Although it has proven to be feasible, it is tedious. It also tends to distract the linguist's thoughts from morphophonological variations to technical matters. Furthermore, hand compiled automata are often not quite consistent with intended rules. Discrepancies arise because the design of rule automata is often affected by assumptions

on the regularity of actual word-forms. Thus, such automata usually function correctly with most of the data but they have a less clear relation with original intended rules.

The rule compiler described below rectifies this problem by letting the linguist write rules in a true rule formalism while the computer produces the automata mechanically. These can then be used in conventional two-level programs, both for testing and and for production use. Several two-level descriptions are now on their way towards completion using the compiler.

2. THE FORMALISM OF TWO-LEVEL RULES

The actual rule formalism supported by the two-level compiler differs only slightly from the original formalism proposed in Koskenniemi (1983). One of the differences is the use of linear representation for pairs, thus **a:o** is used instead of $\frac{a}{o}$. Furthermore, the equal sign is no longer used for denoting the full lexical or surface alphabet. A surface vowel is written simply as **:V** and a lexical **a** as **a:**. Other basic elements are as they used to be:

(1) A sequence of elements are written one after another, thus **:V :V** stands for two successive surface vowels.
(2) Alternative elements are separated by a vertical bar and enclosed in square brackets, e.g. **[:i | :j]** stands for either a surface **i** or a surface **j**.
(3) Iteration is indicated with a superscript asterisk or plus sign, e.g. **:C*** stands for zero or more surface consonants whereas **:C+** requires at least one surface consonant.

Rules with operators => <= and <=> exist as before and they are interpreted as before. A rule:

I:j => :V __ :V

states that every occurrence of a pair **I:j** must be in a context of **.. :V __ :V ..** i.e. between two surface vowels. A rule:

I:j <= :V __ :V

states that between surface vowels a lexical **I** has no other possible realizations than a **j**. Rules with operators <=> are combinations of those with <= and =>.

There is one new type of rules with an operator **~<=**. This rule forbids any occurrences of **LC CP RC**, i.e. it forbids **CP** in the context **LC __ RC** .

Another difference is in the formalism for collapsing several similar rules into one rule. The initial formalism used angle brackets, but this has been replaced by equivalent means using so called "where" clauses. If W denotes a morphophoneme for vowel doubling then a rule for vowel doubling is expressed in the present formalism as:

W:x <=> :x __ ; where x in V;

The definition of the rule component of two-level descriptions consists of six sections:

- a surface alphabet as a list of surface characters
- subsets of the surface alphabet which are used in the rules
- a lexical alphabet
- subsets of the lexical alphabet
- definitions for abbreviations or subexpressions used in the rules
- two-level rules.

A sample two-level description is given below:

```
Lexical Alphabet
  a b c d e f g h i j k l m n o p q r s t u v w
  x y z å ä ö al a2 E I = W;
Lexical Sets
  V = a e i o u y å ä ö al a2 E I W;
  C = b c d f g h j k l m n p q r s t v w x z;
  Diacritics = / ^;
Surface Alphabet
  a b c d e f g h i j k l m n o p q r s t u v w
  x y x å ä ö;
Surface Sets
  V = a e i o u y å ä ö;
  C = b c d f g h j k l m n p q r s t v w x z;
Definitions
  Defaults = al:a a2:a E:e I:i;
Rules
  "Vowel doubling"        W:X  =>   :X __;
                          where X in V;
  "Suppressed doubling"   W:0  <=>  __ I:;
                                    I: __;
                                    W:V __;
  "Stem final V"          [al:0 | a2:o | E:0 | i:e]
                               <=>  __ I:;
  "Plural I"              I:j  <=>  :V __ :V;
```

Note that surface and lexical alphabets are declared separately. This guarantees that the role of each segment is uniquely determined. The sets are separate also because it is not always evident e.g. which segments are considered to be vowels on the lexical level.

The first rule represents a set of several rules:

```
W:a  =>  :a __;
W:e  =>  :e __;
...
W:ö  =>  :ö __;
```

The "where" clause is interpreted in this way because the dummy variable **X** occurs on both sides of the rule. If it would

occur only on the context side then the abbreviation denotes one single rule with many context parts (one for each possibility of **X**).

Another effect of the expansion of "where" clauses is the introduction of some new character pairs. Pairs **W:a, W:e, ..., W:ö** do not occur anywhere in the description but they are implicitly included.

3. STEPS OF THE COMPILATION

The compilation of the two-level description into finite state automata proceeds in several steps. The computation relies essentially on Ron Kaplan's program packages (FSM, FST) for manipulating finite state machines and transducers. The collection of rules has to be treated as a whole because the set of character pairs (CPS) might be changed if some rules are altered thus changing the interpretation of some other rules. The steps of the compilation are:

(1) Transformation of the two-level rule description into a recursive list expression where rule components and pairs are identified.
(2) Expansion of "where" clauses in the rules.
(3) Collecting all pairs explicitly mentioned in rules and definitions in addition to the default set of all **x:x** where **x** is both a lexical and a surface character.
(4) Computing the exact interpretation of pairs which are not concrete pairs (where both the lexical and the lexical components are single characters). Some pairs like **:V** leave one level fully open and others may use the defined subsets such as **W:V** (character **W** corresponding to any of the vowels but not to a zero **0**). All such (partially) unspecified pairs **X:Y** denote the subset of CPS consisting of pairs **x:y** where **x** is **X** or is in **X** and **y** is **Y** or is in **Y**.
(5) Expand each abbreviation of the above type into an alternation. Insert the defined expression in place of the name of the expression.

(6) Compile the components of the rules (correspondence parts, left contexts and right contexts) into finite state machines.
(7) Split rules with the operator <=> into one rule with operator => and another with <=.
(8) Expand rules with operator <= and with multiple contexts into distinct rules with one context each.
(9) Compile the individual component rules separately.
(10) Merge the automata resulting from a single original rule into one rule automaton by intersecting them.

In the present version of the compiler each rule as defined by the user is compiled into a single automaton. If the expansion or compilation splits the rule into subparts these are finally combined into a single machine by the compiler.

The compilation is done on a Xerox 1108 Lisp machine with programs written in Interlisp-D. The resulting automata can then be used either on the Lisp Machine or transported to other systems. In order to be used by the present version of the Pascal two-level program the automata are converted into a tabular format which can be readily used. The format is slightly different from the original one given in Koskenniemi (1983) but it is significantly faster to read in. Such automata have been successfully used on MS-DOS micro computers such as IBM PC and Olivetti M24. Martti Nyman at the University of Helsinki working on one description for Modern Greek and another for Classical Greek and Jorma Luutonen at the University of Turku is working on one for Cheremis. Olli Blåberg has reformulated his Swedish description in terms of the present compiler.

The compiler was written during the summer 1985 at the Center for Studies on Language and Information at Stanford University. In addition to the finite state package written by Ron Kaplan the compiler utilizes Kaplan's concept of compiling complex rules with operator => and several context parts. The compiler was presented at a symposium on finite state morphology on July, 29-30 1985 at CSLI. The compiler has also stimulated some parallel efforts (Bear, in press, Ritchie et. al. 1985, Kinnunen, in preparation).

REFERENCES

Bear, John, (in press). A morphological recognizer with syntactic and phonological rules. Proceedings of COLING-86, Bonn.

Kinnunen, Maarit, (in preparation). Morfologisten sääntöjen kääntäminen äärellisiksi automaateiksi. (The compilation of morphological rules into finite state automata) Masters thesis, Department of Computer Science, University of Helsinki.

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

---, 1984. A general computational model for word-form recognition and production. Proceedings of COLING-84. pp. 178-181.

Ritchie, G.D., S.G. Pulman, and G.J. Russel, 1985. Dictionary and morphological analyzer (Prototype), User guide: Version 1.12. Department of Artificial Intelligence, University of Edinburgh.

Aarno Lehtola
Sitra Foundation
P.O.Box 329, SF-00121 Helsinki

DPL - A COMPUTATIONAL METHOD FOR DESCRIBING GRAMMARS AND MODELLING PARSERS

ABSTRACT DPL, Dependency Parser Language, is a special definition language for describing natural language grammars. It is based on functional dependency. A DPL-grammar consists of (1) definition of used metrics ie. property names and values (2) definition of binary dependency relations and grammatical functions between constituent-pairs (ie. words or recognized phrase constructs) and (3) description of constituent surroundings in the form of two-way automata. The compilation of DPL-grammars results in executable codes of corresponding parsers.

To ease the modelling of grammars there exists a linguistically oriented programming environment, which contains e.g. tracing facility for the parsing process, grammar-sensitive lexical maintenance programs, and routines for the interactive graphic display of parse trees and grammar definitions. Translator routines are also available for the transport of compiled code between various LISP-dialects. The DPL-compiler and associated tools can be used under INTERLISP and FRANZLISP. This paper focuses on knowledge engineering issues. Linguistic argumentation we have presented in /3/ and /4/. The detailed syntax of DPL with examples can be found in /2/.

I INTRODUCTION

Our initial objective was to build a parser for Finnish to work in real production applications. We were faced with both

linguistic and computational problems: (1) so far there was no formal description of the Finnish grammar, (2) there was no parser formalism that seemed specially suitable for highly inflectional and relatively free word order languages, (3) our computational solutions should be portable, efficient and general. In addition we wanted to have linguistic knowledge and processing mechanisms separated in our system. It is important that linguists may ignore the computational details of the parsing processes while the computer professionals may purely concentrate on computational issues.

The parser system we have developped is based on functional dependency. Grammar is specified by a family of two-way finite automata and by dependency function and relation definitions. These are expressed via DPL and compiled automatically to executable parsers. The flexible programming environment makes it easy to tune up parsers. The architecture of DPL-environment is described schematically in Figure 1. The main parts are highlighted by heavy lines. Single arrows represent data transfer; double arrows indicae the production of data structures. The realisations do not rely on specifics of underlying LISP-environments.

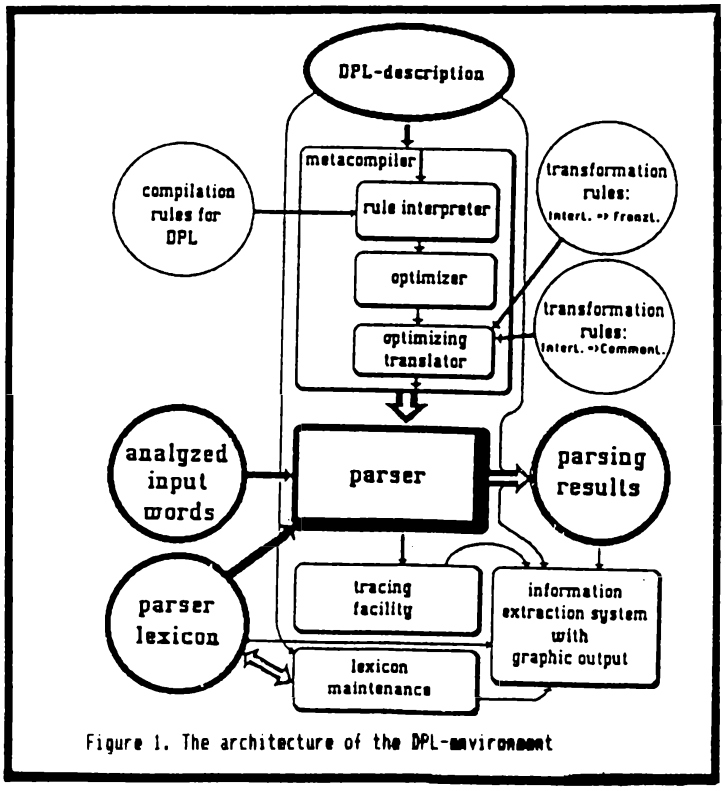


Figure 1. The architecture of the DPL-environment

II DPL-DESCRIPTIONS

The main data object in DPL is a constituent. A grammar specification opens with the structural descriptions of constituents and the allowed property names and property values. User may specify simple properties, features or categories. The structures of the lexical entries are also defined at the beginning. All properties of constituents may be referred in a uniform manner using their values straight. The system automatically takes into account the computational details associated to property types. For example, the system is automatically tuned to notice the inheritance of properties in their hierarchies. Extensive support to multidimensional analysis has been one of the central objectives in the design of the DPL-formalism. Patterning can be done in multiple dimensions and the property set associated to constituents can easily be extended.

The binary grammatical functions and relations are defined as special and-or-expressions which contain both property predications and search directing information. A DPL-function returns as its value the binary construct built from the so called current constituent and its dependent candidate, or it returns NIL. DPL-relations return as their values the pairs of constituents that have passed the predicate filter. A user may vary a predication between simple equality and equality with ambiguity elimination. As their side effects predications may also replace and insert properties.

In the level of two-way automata the working storage consists of two constituent stacks and of a register which holds the current constituent. The two stacks hold the right and left contexts of the current constituent. The basic decision for the automaton associated with the current constituent is to accept or reject a neighbor via a valid syntactico-semantic subordinate relation. Successfully called DPL-function subordinates the neighbor, and it disappears from the stack. The structure of an input sentence will be the outcome of a series of such binary constructions. Dynamic local control is realized by permitting the automata activate one another.

III THE DPL-COMPILER

A compilation results in executable code of a parser. The compiler produces highly optimized lisp code /1/. In the generated code only a small set of basic lisp functions is used. In bench marking was found that specialized higher level functions often consume more time than corresponding functions composed of open compilable basic functions. For instance many type checks can be avoided when the actual situation of use is known. In addition the chosen set makes it easier to transfer parsers to other computers. The low-level lisp code may be compiled to machine language level by normal lisp compilers.

Internally data structures are only partly dynamic for the reason of fast information fetch. Ambiguities are expressed locally to minimize redundant search. The principle of structure sharing is followed whenever new data structures are built. In the manipulation of constituent structures there exists a special service routine for each combination of property and predication types. These routines take special care of time and memory consumption. For instance with regard to replacements and insertions the copying includes physically only the path from the root of the list structure to the changed sublist. The logically shared parts will be shared also physically. This principle of structure sharing minimizes memory usage.

In the state transition network level the search is done depth first. To handle ambiguities DPL-functions and -relations process all alternative interpretations in parallel. In fact the alternatives are stored in the stacks and in the current constituent register as trees of alternants.

As a general time consuming strategy iteration is preferred to recursion whenever possible. Boolean expressions are optimized to avoid unnecessary nesting. The same affects also nested 'conds' and 'ifs'. Local memory reservation is minimized by taking into account control paths.

In the first version of the DPL-compiler the generation rules were intermixed with the compiler code. The maintenance of the compiler grew harder when we experimented with new computational characteristics. We therefore developed a metacompiler in which compilation is defined by rules.

Our parsers were aimed to be practical tools in real production applications. It was hence important to make the produced programs transferable. As of now we have a rule-based translator which converts parsers between LISP-dialects. The translator accepts currently INTERLISP, FRANZLISP and COMMON LISP.

IV LEXICON AND ITS MAINTENANCE

The environment has a special maintenance program for lexicons. The program uses video graphichs to ease updating and it performs various checks to guarantee the consistency of the lexical entries. It also co-operates with the information extraction facility to help the user in the selection of properties.

V THE TRACING FACILITY

The tracing facility is a convenient tool for grammar debugging. For example, in Figure 2 appears the trace of the parsing of the sentence "Poikani tuli illalla kentältä heittämästä kiekkoa." (i.e. "My son came back in the evening from the stadium where he had been throwing the discus."). Each row represents a state of the parser before the control enters the state mentioned on the right-hand column. The thus-far found constituents are shown by the parenthesis. An arrow head points from a dependent candidate (one which is subjected to dependency tests) towards the current constituent.

```
_(T POIKANI TULI ILLALLA KENTÄLTÄ HEITTÄMÄSTÄ KIEKKOA .)

383 conses
.03 seconds
0.0 seconds, garbage collection time
PARSED
_PATH()

=>  (POIKA)  (TULLA)  (ILTA)  (KENTTÄ)  (HEITTÄÄ)  (KIEKKO)  ?N
(POIKA)  <=  (TULLA)  (ILTA)  (KENTTÄ)  (HEITTÄÄ)  (KIEKKO)  N?
=>  (POIKA)  (TULLA)  (ILTA)  (KENTTÄ)  (HEITTÄÄ)  (KIEKKO)  ?NFinal
(##)  (POIKA)  (TULLA)  (ILTA)  (KENTTÄ)  (HEITTÄÄ)  (KIEKKO)  NIL
(POIKA)  =>  (TULLA)  (ILTA)  (KENTTÄ)  (HEITTÄÄ)  (KIEKKO)  ?V
=>  ((POIKA) TULLA)  (ILTA)  (KENTTÄ)  (HEITTÄÄ)  (KIEKKO)  ?VS
((POIKA) TULLA)  <=  (ILTA)  (KENTTÄ)  (HEITTÄÄ)  (KIEKKO)  VS?
((POIKA) TULLA)  =>  (ILTA)  (KENTTÄ)  (HEITTÄÄ)  (KIEKKO)  ?N
((POIKA) TULLA)  (ILTA)  <=  (KENTTÄ)  (HEITTÄÄ)  (KIEKKO)  N?
((POIKA) TULLA)  =>  (ILTA)  (KENTTÄ)  (HEITTÄÄ)  (KIEKKO)  ?NFinal
((POIKA) TULLA)  <=  (ILTA)  (KENTTÄ)  (HEITTÄÄ)  (KIEKKO)  VS?
((POIKA) TULLA (ILTA))  <=  (KENTTÄ)  (HEITTÄÄ)  (KIEKKO)  VS?
((POIKA) TULLA (ILTA))  =>  (KENTTÄ)  (HEITTÄÄ)  (KIEKKO)  ?N
((POIKA) TULLA (ILTA))  (KENTTÄ)  <=  (HEITTÄÄ)  (KIEKKO)  N?
((POIKA) TULLA (ILTA))  =>  (KENTTÄ)  (HEITTÄÄ)  (KIEKKO)  ?NFinal
((POIKA) TULLA (ILTA))  <=  (KENTTÄ)  (HEITTÄÄ)  (KIEKKO)  VS?
((POIKA) TULLA (ILTA) (KENTTÄ))  <=  (HEITTÄÄ)  (KIEKKO)  VS?
((POIKA) TULLA (ILTA) (KENTTÄ))  =>  (HEITTÄÄ)  (KIEKKO)  ?V
((POIKA) TULLA (ILTA) (KENTTÄ))  (HEITTÄÄ)  <=  (KIEKKO)  V?
((POIKA) TULLA (ILTA) (KENTTÄ))  (HEITTÄÄ)  =>  (KIEKKO)  ?N
((POIKA) TULLA (ILTA) (KENTTÄ))  (HEITTÄÄ)  (KIEKKO)  <=  N?
((POIKA) TULLA (ILTA) (KENTTÄ))  (HEITTÄÄ)  =>  (KIEKKO)  ?NFinal
((POIKA) TULLA (ILTA) (KENTTÄ))  (HEITTÄÄ)  <=  (KIEKKO)  V?
((POIKA) TULLA (ILTA) (KENTTÄ))  (HEITTÄÄ (KIEKKO))  <=  VO?
((POIKA) TULLA (ILTA) (KENTTÄ))  =>  (HEITTÄÄ (KIEKKO))  ?VFinal
((POIKA) TULLA (ILTA) (KENTTÄ))  <=  (HEITTÄÄ (KIEKKO))  VS?
((POIKA) TULLA (ILTA) (KENTTÄ) (HEITTÄÄ (KIEKKO)))  <=  VS?
=>  ((POIKA) TULLA (ILTA) (KENTTÄ) (HEITTÄÄ (KIEKKO)))  ?VFinal
((POIKA) TULLA (ILTA) (KENTTÄ) (HEITTÄÄ (KIEKKO)))  <=  MainSent?
((POIKA) TULLA (ILTA) (KENTTÄ) (HEITTÄÄ (KIEKKO)))  <=  MainSent? OK
DONE
```

Figure 2. A trace of parsing process

The tracing facility gives also the consumed CPU-time and two quality indicators: search efficiency and connection efficiency. Search efficiency is 100%, if no useless state transitions took place in the search. This figure is meaningless when the system is parameterized to full search because then all transitions are tried. Connection efficiency is the ratio of the number connections remaining in a result to the total number of connections attempted for it during the search.

There exists also automatic book-keeping of all sentences input to the system. These are divided to into two groups: parsed and not parsed. The first group constitutes growing test material to ensure monotonic improvement of grammars.

VI THE INFORMATION EXTRACTION FACILITY

In an actual working situation there may be thousands of linguistic symbols in the work space. To make such a complex manageable, we have implemented an information system that for a

given symbol pretty-prints all information associated with it.

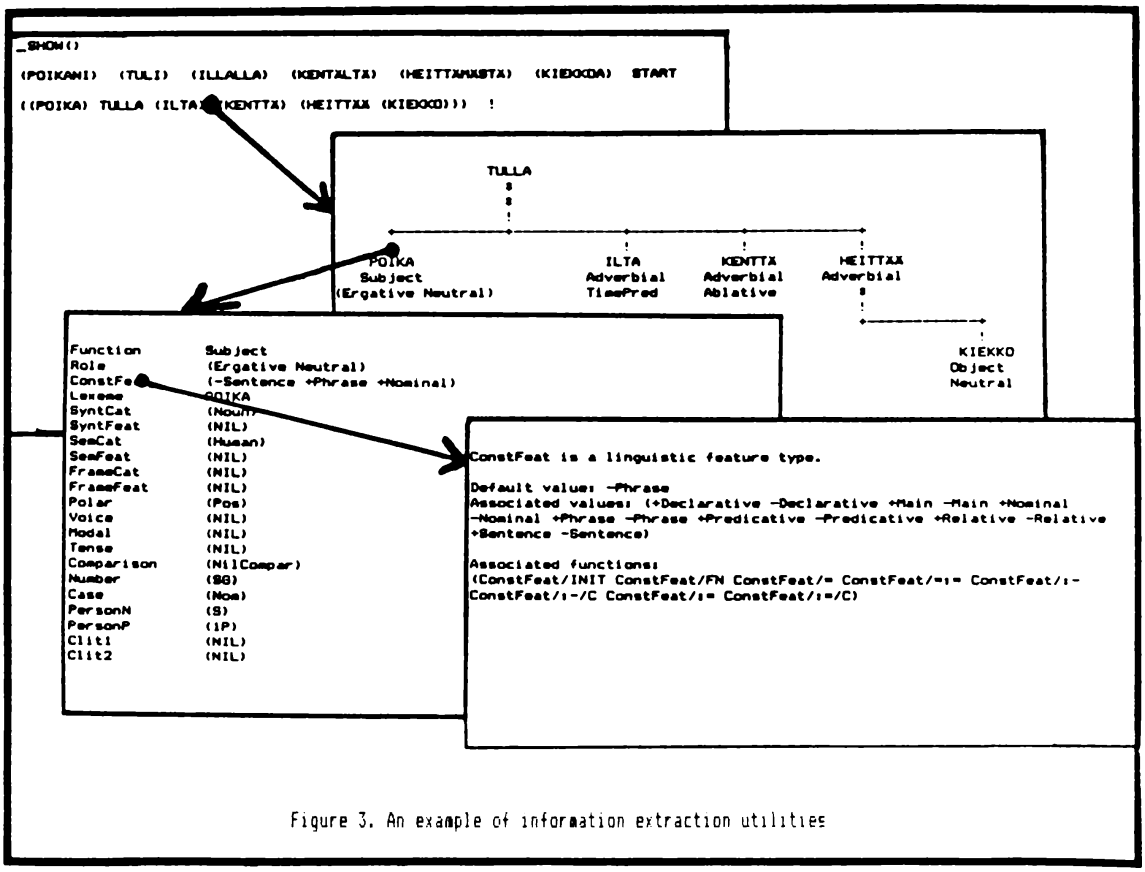


Figure 3. An example of information extraction utilities

The environment has routines for the graphic display of parsing results. A user can select information by pointing with the cursor. The example in Figure 3 demonstrates the use of this facility. The command SHOW() inquires the results of the parsing process described in Figure 2. The system replies by first printing the start state and then the found result(s) in compressed form. The cursor has been moved on top of this parse and CTRL-G has been typed. The system now draws the picture of the tree structure. Subsequently one of the nodes has been opened. The properties of the node POIKA appear pretty-printed. The user has furthermore asked information about the property type ConstFeat.

VII CONCLUSION

The parsing strategy applied for the DPL-formalism was originally viewed as a cognitive model . It has proved to result practical and efficient parsers as well. Experiments with a non-trivial set of Finnish sentence structures have been performed both on DEC-2060 and on VAX-11/780 systems. Experiments with a non-trivial set of Finnish sentence structures have been performed both on DEC-2060 and on VAX-11/780 systems. The time behaviour on DEC-2060 has been described in Figure 4. In those test runs only main sentences were used. The analysis of an eight word sentence, for instance, takes between 20 to 600 ms of DEC CPU-time in the INTERLISP-version depending on whether one wants only the first or, through complete search, all parses for structurally ambiguous sentences. The MACLISP-version of the parser runs about 20% faster on the same computer. The NIL-version (COMMON LISP compatible) is about 5 times slower on VAX.

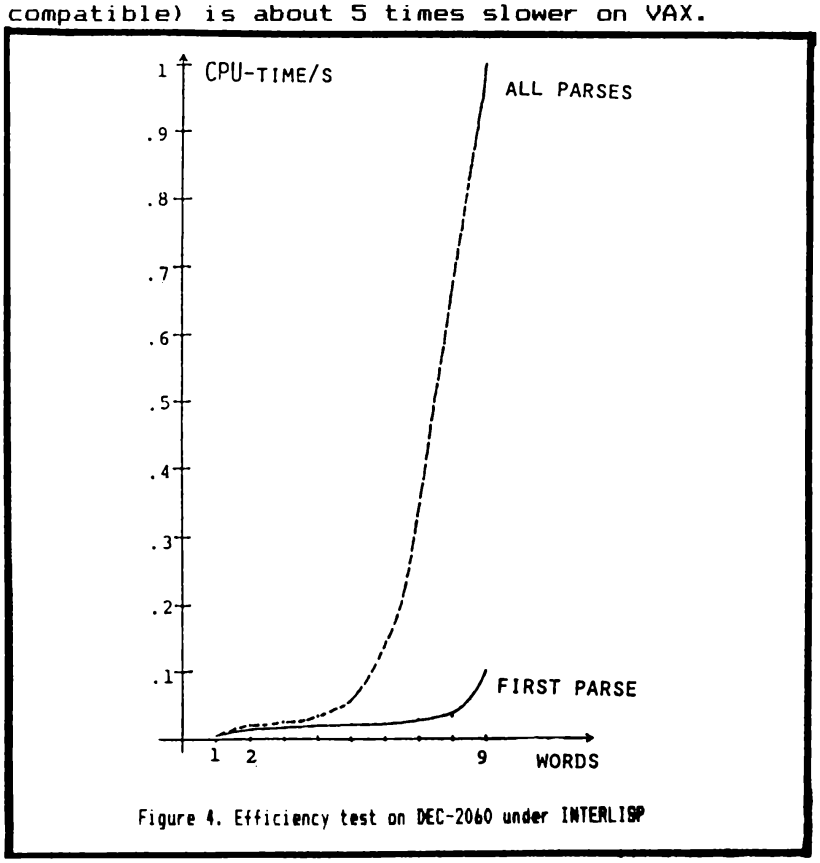


Figure 4. Efficiency test on DEC-2060 under INTERLISP

The whole environment has been transferred also to FRANZLISP on VAX. We have not yet focused on optimality issues in grammar descriptions. We believe that by reordering expectations in the

automata and by introducing more heuristics to reduce parallelism improvement in efficiency ensues.

REFERENCES

1. Lehtola, A., Compilation and Implementation of 2-way Tree Automata for the Parsing of Finnish. M.S. Thesis, Helsinki University of Technology, Department of Technical Physics, 1984, 120 p. (in Finnish)

2. Lehtola, A., Jäppinen, H. and Nelimarkka, E., Language-based Environment for Natural Language Parsing. Proc. of the 2nd Conf. of the European Chapter of the Association for Computational Linguistics, Geneva, 1985, pp. 98-106.

3. Nelimarkka, E., Jäppinen, H. and Lehtola, A., Two-way Finite Automata and Dependency Theory: A Parsing Method for Inflectional Free Word Order Languages. Proc. COLING84/ACL, Stanford, 1984, pp. 389-392.

4. Nelimarkka, E., Jäppinen, H. and Lehtola, A., Parsing an Inflectional Free Word Order Language with Two-way Finite Automata. Proc. of the 6th European Conference on Artificial Intelligence, Pisa, 1984, pp. 167-176.

5. Winograd, T., Language as a Cognitive Process. Volume I: Syntax. Addison-Wesley Publishing Company, Reading, 1983, 640 p.

IVAN RANKIN,
DEPT. of COMPUTER and INFORMATION SCIENCE,
LINKÖPING UNIVERSITY,
SWEDEN.

SMORF - an implementation of Hellberg's morphology system

Abstract: *A brief account of Hellberg's morphology system [Hellberg, 78] is presented - its aims and structure, and how it deals with inflection, derivation and compounding. Then there follows a discussion of our experience when implementing and running the program and evaluating the success of the system. Discussion points are accompanied by run-time examples. Some improvements on the original system have been made and are illustrated in the text. A summary of our evaluation is given.*
***Key words**: inflection, derivation, compounding, explicitness, exhaustiveness, overproduction of analyses, filters, linguistic transparency, semantics.*

1. AIMS AND STRUCTURE

1.1. AIMS

Hellberg's intention is to provide a "detailed account of Swedish morphology" and he adds "A system of paradigms has been set up and a basic dictionary compiled, for the primary purpose of being used in algorithmic text analysis." [Hellberg, 78].
Hellberg makes two specific claims for his system - **explicitness** and **exhaustiveness**:

a) "As to **explicitness**, this means that the system accounts for all types of variations, limitations and extensions, including forms which may seem self-evident to speakers of the language."
eg. the unsettled use of the plural form of *meddelande* with both *meddelanden* or *meddelande* as acceptable forms.

b) "**Exhaustiveness** implies for one thing that attempts have been made to cover paradigms with just a few members, perhaps only three or four, ..."
eg. the paradigm for *prestanda* is an example of a paradigm with only a few members, as opposed to the one covering words like *flicka* with many members.

"... and for another that not only inflectional forms are taken into account, but also stem modifications and linking elements which may occur in derivations and compounding."
eg. the paradigm for a word such as *gata* will account for compounds with no linking element - *gatsopare*, for example, as well as compounds with the linking element - *gatubelysning*.

The borderline between what is an acceptable form of a word and what is not acceptable is not always clear in reality and Hellberg comments: "Where the limits

should be drawn...cannot be decided once and for all, but gradually, on the basis of experience gained in the use of the system." op cit.

No claims are made to the effect that the system reproduces the way a human reader processes the text. Hellberg admits that the system fails to capture some of the morphological similarities between words or between groups of words, while at the same time this technical approach brings out similarities not adequately described in the traditional grammatical framework - such as "the distinction between strong and weak verb conjugation, where the system displays a multitude of transitional and mixed types between second, third and fourth conjugations". op cit.

1.2. STRUCTURE - THE DICTIONARY, THE PARADIGMS and THE SUBROUTINES.

1.2.1. The dictionary.

Words are stored in the dictionary in the form of a technical stem and a reference to a paradigm number (amongst other things). The technical stem is that part of the word which is common to all inflectional forms, eg.

Word	Technical stem	Paradigm nr.
flicka	*flick*	101
klaga	*klag*	715
nyckel	*nyck*	231
seger	*seg*	232
krypa	*kryp*	742
krupit	*krup*	744

We shall return to the consequences of the concept of the 'technical stem' later in the discussion of the implementation.

1.2.2. Paradigms and Subroutines.

To show how the paradigms and subroutines work, it may be clearer to follow an example. Imagine that the word *flickorna* is to be analyzed and assume for the time being its technical stem *flick-* has been located in the dictionary. The dictionary entry provides a reference to its related paradigm, 101.

Paradigm 101

nn utr

	lo # ----> 92	*flicka(s)*
	----> *e* ----> 96	*flickan(s)*
>5		*flickor(s)*
		flickorna(s)
	sh ----> 11	*flick(s)-*
	----> (lo)	*flicke-*

(Note that the possible forms of *flicka* have been added here for illustrative purposes; they are not part of the paradigm.)

a) First a check is made on the length of the rest of the word after the technical stem to determine whether it is so long that it must be a compound or derivative rather than an inflectional form. If so, it will be unnecessary to run through all the tests for inflectional endings as they will be bound to fail. The threshold value for this length is given in the paradigm (note >5 in the schema above). In the case of *flickorna* the length of the part of the word after the technical stem, *-orna* is not greater than 5 so the search is continued in the short (sh) branch. (Checking the length of the rest of the word and selecting a particular branch has no theoretical importance, it is a strategy designed purely to speed up the search.)

b) Within the long/short branches the tests are run in a top-to- bottom order; this order is based on text frequency-counts.

c) The search continues through the paradigm; the path chosen depends on the next character to be matched. The branches of the paradigms lead into subroutines. A paradigm is entered only once for each technical stem. All numbers in a paradigm refer to subroutines.

d) While traversing the paradigms and subroutines the morphological features are picked up; in this example **nn utr** (**noun non-neuter**) are picked up on entering the paradigm.

e) In this example the short branch of the paradigm refers first to Subroutine 11 (and later to the long form of the same paradigm) and the first letter to be matched is the *o* of *-orna*.

Subroutine 11

---->*a* ---->91 **obe sin**
---->*n* ---->91 **bes sin**
---->*#* *o* ---->10

The ---->*a* branch is not followed up, but the ---->*o* branch is, (although there are no features to be picked up). The # sign is of no theoretical importance. It simply marks the end of the linguistic stem. The search is directed to Subroutine 10.

Subroutine 10

---->*r* ---->91 **obe plu**
---->*n* ---->*a* ---->91 **bes plu**

Here the *r* in *-rna* is matched and then the second of the two options (either absence or presence of *na* to be matched) is chosen. The features **bes plu** (**definite plural**) are picked up and the search is referred to Subroutine 91.

Subroutine 91

```
If position 0 = s then
  if position 1 = blank then accept and label gru/gen
  else reject
Else if position 1 = blank then accept and label gru
   else if position 1 = s then
                  if position 2 = blank then accept and label gen
                  else reject
        else reject
```

Position 0 is the position of the last character matched. Position 1 is the position of the next character to be matched. At this stage in the example it is blank as the ending *-orna* has been successfully matched already and no further characters remain. The feature **gru (base form)** is picked up.
The analysis returns: ***flickorna: flicka*** **nn utr bes plu gru**
(noun non-neuter definite plural base-form)

Note that the paradigms reflect the morphological differences between words - if two words are similar but display a morphological difference, they will belong to different paradigms. The subroutines reflect similarities; several groups of words may be referred to a single subroutine, take the *-rna* plural ending of Subroutine 10, for example.

In passing it may be observed that there are 235 paradigms in all.

1.3. INFLECTION, DERIVATION and COMPOUNDING.

1.3.1 Inflection.

The way in which inflection is dealt with was exemplified in case of *flickorna* above. The paradigms and subroutines contain all the possible inflectional forms, and the features of each word are picked up as the paradigms and subroutines are traversed.

1.3.2 Derivation.

Hellberg was faced with a choice: either a) include derivational endings in the paradigms and subroutines, or b) include them in the dictionary where they are treated as entries in their own right alongside *flick, kryp* etc., and then view them as compound elements as far as the paradigms and subroutines are concerned.

For a number of reasons he chose the latter, mainly because choice a) would cause an "enormous expansion" of the paradigms and also restrictions in the use of productive suffixes are semantic rather than morphological in nature (and therefore a morphology system need not account for such restrictions). This means that a word like *storhet* will be treated as a "compound" of *stor* + *het* The same is true of prefixes

such as *o, sam* or *bi*: *omöjlig* will be treated as the first part of a compound such as *o* + *möjlig*.

1.3.3 Compounds.

The fact that the paradigm system is adequately equipped to handle compounds is a theoretical necessity considering the seemingly unlimited possibilities of compounding in Swedish, and a practical necessity in that it drastically reduces the number of words stored in the dictionary. If *hand* and *duk* are entered in the dictionary, there is no need to add *handduk*.

There are some exceptions to this rule. An example will suffice to clarify the issue. The word *blick* is in the dictionary. So are the prefixes *in-* and *över-*. This means that *inblick* and *överblick* do not need to be entered separately. However, although *ögon* is also in the dictionary, *ögonblick* must be entered in its own right as it is not derivable from its component parts - it is of a different gender from the above examples. This point is discussed further in the second section.

Note that the paradigms reflect the morphological differences between words - if two words are similar but display a morphological difference, they will belong to different paradigms. The subroutines reflect similarities; several groups of words may be referred to a single subroutine, take the *-rna* plural ending of Subroutine 10, for example.

In passing it may be observed that there are 235 paradigms in all.

2. ON IMPLEMENTING THE SYSTEM

The system has been implemented at Linköping Tekniska Högskola in two versions - in Interlisp-10 on DECSYSTEM-20 and in Interlisp-D on a Xerox 1108/1109. The basic dictionary was kindly supplied by Språkdata, Gothenburg. The system has been partially running and tested since summer 1984 and finally developed and tested since summer 1985.

2.1 INFLECTION, DERIVATION AND COMPOUNDING. THE OVERPRODUCTION OF ANALYSES.

The analysis of inflected forms has proved very accurate and complete. Examples such as *flickorna* and *drack* provide clear, unambiguous analyses.

```
SMORF Window
54-SMORF: FLICKORNA
(FLICKORNA (("FLICKA" (noun non-neut def plur base))))
55-SMORF: DRACK
(DRACK ((("DRICKA")
         (verb past))))
56-SMORF:
```

However, even seemingly uncomplicated word forms can display unexpected side effects, eg. *bil*.

```
57-SMORF: BIL
(BIL (("BIL" (noun non-neut indef sing base))
      ("BI" "L" (abbrev base))
      ("BI" "L" (abbrev))))
58-SMORF:
```

This problem stems from the fact that, in allowing for the almost unlimited ways of forming compounds - and in the formation of derivatives in this system - a profusion, if not to say confusion, of interpretations is possible. In the analyses of *bil* the correct interpretation comes first; the second and third interpretations have analyzed *bil* as a compound of *bi* and the abbreviation *l*. *Frukosten* illustrates the difficulties even more clearly.

```
SMORF Window
60-SMORF: FRUKOSTEN
(FRUKOSTEN (("FRUKOST" (noun non-neut def sing base)
                       )
            ("FRU" "KO" "STEN" (prop-name base))
            ("FRU" "KO" "STEN"
                   (noun non-neut indef sing base))
            ("FRU" "KO" "TE" "N" (abbrev base))
            ("FRU" "KO" "TE" "N" (abbrev))
            ("FRU" "KOSTA" "EN"
                   (art indef sing non-neut base))
            ("FRU" "KOSTA" "EN" (non-Swed))
            ("FRU" "KOSTA" "EN" (adv))
            ("FRU" "KOSTA" "EN" (pron nominal-fn
                                       base))
            ("FRUKOST" "EN"
                       (art indef sing non-neut
                            base))
            ("FRUKOST" "EN" (non-Swed))
            ("FRUKOST" "EN" (adv))
            ("FRUKOST" "EN" (pron nominal-fn base)))
           )
```

Filtering out the more unlikely analyses characterized a new stage of development. By preventing foreign words and abbreviations from forming compound elements it was possible to rule out the ubiquitous unlikely analysis of words such as *flicka* being interpreted as a compound of *flicka* and the English article *a* or **non-neuter definite singular** forms such as *bilen* being analyzed as a compound of the Swedish *bil* and the French *en* on the one hand, and *flicka* being analyzed as a compound of *flicka* + *a*, the latter part of the abbreviation of *bl a* (*bland annat*) on the other. These two filters alone reduced the number of "wrong" interpretations by about 30%. The following example shows the stepwise layering of filters on *flicka* the first version with no filters implemented, the last version with three filters implemented:

```
27-SMORF: FLICKA
(FLICKA (("FLICKA" (noun non-neut indef sing base))
         ("FLICKA" "A" (non-Swed))
         ("FLICKA" "A" (abbrev base))
         ("FLICKA" "A" (abbrev))))
28-SMORF:
```

SMORF Window

```
48-SMORF: FLICKA
(FLICKA (("FLICKA" (noun non-neut indef sing base))
         ("FLICKA" "A" (non-Swed))
         ("FLICKA" "A" (abbrev))))
49-SMORF:
```

SMORF Window

```
51-SMORF: FLICKA
(FLICKA (("FLICKA" (noun non-neut indef sing base))
         ("FLICKA" "A" (non-Swed))))
52-SMORF:
```

SMORF Window

```
54-SMORF: FLICKA
(FLICKA (("FLICKA" (noun non-neut indef sing base))))
55-SMORF:
```

So far filters have been put on in an ad hoc fashion. This is mainly because it has not yet been decided what use the system will be put to.

Another troublespot, also caused by the inclusion of derivative endings in the dictionary as items in their own right, was the common collocation *ska*.It was found that all adjectives of the *-sk* type in plural and singular definite attributive forms were interpreted not only as adjectives (ie. correctly) but also as verbs, eg. *ironi + ska*. To make matters worse *-ska* is included in the dictionary as a noun ending, eg. *ilska*. Thus *ironiska* also became a **noun**. As there is only a small number of nouns ending in *-ska* in Swedish, it would perhaps be a better choice to list them separately in the dictionary.

Before leaving this brief glimpse at the problem of the overproduction of analyses, it is in place to emphasize that the "correct" interpretation is always provided; the difficulty has been in reducing the number of erroneous alternatives.

2.2 WHAT THE SYSTEM LACKS

2.2.1. Re-insertion of deleted consonants.

One feature that the system lacks is the ability to deal with deleted consonants in compounds as in *ful(l)-lärd* or *glas(s)-skål*. What happens is that *full* is recognized first and then *ärd* is checked (and found not to exist in the dictionary), then *ful* is recognized and *lärd* is then checked (and found) which explains the two interpretations shown below. A similar explanation goes for *glasskål*.

```
SMORF Window
79-SMORF: FULLÄRD
(FULLÄRD (("FUL" "LÄRA"
                 (verb past-prt indef sing
                       non-neut base))
          ("FUL" "LÄRD"
                 (adj indef sing non-neut base)
                 )))
80-SMORF: GLASSKÅL
(GLASSKÅL (("GLAS" "SKÅL"
                  (noun non-neut indef sing
                        base))
           ("GLASS" "KÅL"
                   (noun non-neut indef sing
                         base))))
81-SMORF:
```

2.2.2 Dictionary search.

When using Hellberg's description of the system as a basis for our implementation, we found that no algorithm or even general criteria for organizing the dictionary search were provided. The solution we decided best fulfilled the requirements of the system was to conduct an exhaustive backward search on the string being analyzed such that for *bildrulle*, for example, the following technical stems would be sought in the dictionary:

bildrulle, bildrull, bildrul ... bild, bil, ... b

In this way all the possible technical stems of the initial string are found and the rest of the string is subsequently analyzed in the same way.

2.2.3 Linguistic transparency

One of Hellberg's claims was to provide "a detailed account of Swedish morphology". The system is not immediately comprehensible to alinguist on account of the large number of paradigms (235), many with only marginal differences (see [Brandt, 1985]). Only brief descriptions of each paradigm have been provided (which has not made it easy to add new words to the dictionary along with the correct paradigm

reference). Hellberg mentions in his introduction that some linguistic generalizations have given way to technical considerations. I would agree with Brandt in his criticism of the fact that the regular phenomenon of e-syncope is spread out over several paradigms, but not in his preference for another model with fewer paradigms that can capture broader linguistic generalizations. This point of view ignores the use the system was designed for - which was not just a theoretical model but the design for a practical implementation which is required to handle the individual idiosyncrasies of a word or group of words in authentic texts.
Also clouding the linguistic transparency is the fact that while most of the derivative endings have been included in the dictionary, a small subset have been tucked away in a subroutine of their own (93) as they differ from the others in that they demand the lack of an unstable vowel in the stem. The consequence of this is that derivative endings are to be found both in the dictionary and in the subroutines - a deviation from Hellberg's original aim not to build out theparadigms and subroutines in this way.

During an analysis the grammatical features are picked up when the paradigms and subroutines are being traversed. However, some features are attached to words in the dictionary. These are words which are simply accepted in their dictionary form (eg. *att* the infinitive marker) or others which are referred to a cross-reference dictionary (eg. *trivas* which is labelled as a verb). Features are, therefore, to be found in the paradigms, subroutines and, for certain words only, in the dictionary.

For anyone else intending to implement the system it may be mentioned that there is a misprint (?) in paradigms 825-827 in [Hellberg, 78] where the threshold level given as 0 should in fact be 1 (to allow for the genitive *s*). Also the list of abbreviations on page 130 is incomplete; the following are missing : *psp - present participle, pas - passive, kom - comparative, suv - superlative, opt - optative.*

2.3 SEMANTICS

In the first section of this paper one of Hellberg's reasons for not including the derivative endings in the paradigms and subroutines was given: restrictions governing possible derivative endings are of a semantic rather than morphological nature. The system accepts, therefore, **bilig*, **biling*, **bilarinna*, **obil*, etc.

```
SMORF Window
60-SMORF: BILIG
(BILIG (("BI" "LIG" (adj indef sing non-neut base))
        ("BIL" "IG" (adj indef sing non-neut base))))
61-SMORF:
NIL
61-SMORF: BILING
(BILING (("BI" "LING" (noun non-neut indef sing base))
         ("BIL" "ING" (noun non-neut indef sing base))))
62-SMORF:
```

Later on, though, Hellberg goes on to draw some semantic distinctions; consider the example given earlier: *Ögonblick* should be included in the dictionary as a separate item "since it is not semantically derivable" from its component parts. He continues "...the principle should be that compounds and derivatives whose meaning cannot be derived from their components must be entry words of their own in the dictionary". This means that *avlasta*, *avsluta* and *avskilja* are not listed in the dictionary, whereas *avlida*, *avse* and *avrätta* are as the latter only have a transferred meaning. Thus the following differences appear:

```
SMORF Window
96-SMORF: AVLASTA
(AVLASTA (("AV" "LASTA" (verb inf))
          ("AV" "LASTA" (verb imp))))
97-SMORF: AVSE
(AVSE (("AVSE" (verb inf))
       ("AVSE" (verb imp))
       ("AV" "SE" (verb inf))
       ("AV" "SE" (verb imp))))
98-SMORF:
```

The attention paid to semantics, then, is somewhat inconsistent and if the system is to be used for purposes other than word analysis, a more systematic approach may be required. To justify the need for including semantic considerations, try the following example, *smörgås*.

```
SMORF Window
57-SMORF: SMÖRGÅS
(SMÖRGÅS (("SMÖRGÅS" (noun non-neut indef sing base/gen))
          ("SMÖR" "GÅS" (noun non-neut indef sing base/gen))
          ("SMÖR" "GÅ" (verb inf pas))
          ("SMÖR" "GÅ" (verb pres pas))
          ("SMÖR" "GÅ" (verb imp pas))))
58-SMORF:
```

3. CONCLUSIONS (+ advantage, - disadvantage)

+ The system is a powerful analyzer of Swedish word forms.

+ It meets its claims of explicitness and exhaustiveness.

- The over-production of unlikely analyses will have to be further restricted for most purposes.

- It is not as linguistically transparent as it could be.

- It has no morphotactic rules built in; the anaylysis is purely on a character matching level.

+ It has potential as a generator where, given a word and a set of features, it would generate appropriate word forms. This would have to be limited to inflected forms (and possibly derivations), as it is difficult to predict the type of linking element (if any) otherwise required.

REFERENCES:

Brandt, Søren. *The Influence of Computers on Linguistics and Language*, Eighth Scandinavian Conference of Linguistics, ed. Tugely. 1985.

Hellberg, Staffan. *The Morphology of Present-Day Swedish,* Almqvist & Wiksell International. Stockholm. 1978.

ABBREVIATIONS USED IN THE TEXT

Sprakdata's original abreviations retained in the paradigms and subroutines:

bes	definite (Sw. *bestämd*)	obe	indefinite (Sw. *obestämd*)
gen	genitive	plu	plural
gru	base form (Sw. *grundform*)	sin	singular
nn	noun	utr	non-neuter (Sw. *utrum*)

Abbreviations used as output of the system as in the examples shown:

abbrev	abbreviation	inf	infinitive
adj	adjective	nominal-fn	nominal function
adv	adverb	non-Swed	non-Swedish
art	article	pas	passive
base	base-form	past-prt	past participle
def	definite	plur	plural
imp	imperative	pron	pronoun
indef	indefinite	sing	singular

Bengt Sigurd
Dep. of Linguistics and Phonetics
University of Lund

Computer simulation of dialogue and communication

The dialogue is the primary use of language and the study of communication in dialogue is necessary for our understanding of natural language. There has been an increased interest in communication, in spite of the negative attitudes of Chomsky (1975). (He does not even mention the word communication in the index of his classical book Aspects of the theory of syntax, 1965).

Even small-scale attempts at simulating dialogue by computer are rewarding and further our understanding of language. The insights may be used in practical applications such as expert systems and operative systems.

This paper will look at dialogue from different angles. An analysis of dialogue in cycles of two or three steps will be presented. It can be described by extending the phrase structure rules which have been used for sentence analysis. With this background a model of a turn-taker, who can participate in such a dialogue and produce the proper steps in the cycles will be presented. This model has been implemented in a small-scale computer system where two "persons" talk (Sigurd, 1985) to each other using different speech synthesis voices. This implementation shows how well the current speech synthesis devices can serve in human dialogue. The model allows the operator to study the effect of pauses, willingness to give backchannel items (supportivity), to respond and take initiatives.

Lastly the construction of a more advanced model where the transfer of information is focused is discussed. It is noted that many words in the dialogues serve to indicate how the information should be or has been accepted. An information driven model must also focus the difference between the "real" world and the verbal discourse world which is being built during the dialogue. Metacomments, such as sort of, "", precisely, typical play an important role in marking the relations between the verbal expressions and the "real" world.

Dialogue as cycles

Fig.1 presents a sample dialogue and some rules to analyze it. The rules show the main types of utterances occurring in dialogue and their sequencing. The rules also show the basic grammatical structure. Generative phrase structure rules introduced in order to characterize sentences (cf Chomsky, 1957,1965) are used in all the rules.

It may be convenient to start by looking at the sample dialogue. It consists of utterances 1-7 by partners I and II. Some utterances clearly belong together and consitute sections, rounds or cycles. Utterances 1 and 2 make up one cycle consisting of a statement and a response. The response is called a backchannel item. Utterances 3-5 make up another cycle including three constituents; a question, an answer and a backchannel item and utterances 6-7 make up a cycle with two utterances: an instruction (order) and a reaction which we also classify as a backchannel item.

We note the punctuation marks: full stop, question mark, mark of exclamation (which could perhaps have been used after the other backchannel items as well). In speech, the intonation contours would vary correspondingly.In speech, there would have been pauses at least between utterances 1-2,3-4,4-5, and 6-7, where there is a shift of speaker. But when the speaker is the same, as in 2-3,5-6, the utterances could have been delivered without a pause (cf Oreström,1983,Stenström,1984).

It is assumed that a dialogue consists of a series of cycles of this type - at least one. The types of cycles are given in rule 1, where certain terms are introduced. The terminology in the literature varies between different authors and fields. It could be oriented towards speech acts, game theory, linguistic form or function. The terms interchange, transaction and move have been used. Backchannel items have been called supports which indicates the double nature of these utterances.

The only type of question included in the rules is the truth question, which could be answered by yes or no. Traditional grammar has not analyzed dialogue in the way we do and does not make a clear distinction between yes as an answer and as a backchannel item (see below). On the other hand, it is an interesting fact, that many languages use the same word as a positive answer and as a supportive backchannel item.

The subsequent rules in fig. are the familiar rules indicating the basic structure of statements (S), questions (Q) and instructions (I). The internal structure of the constituents is not discussed in this superficial account. The addressee may be mentioned in imperatives and in questions, e.g. by pronouns, names or vocative terms such as dear, you idiot. This is not shown in these rules, but in fig.2 below.

The backchannel items (B) are an almost closed set of words, including e.g. OK (almost a language universal soon), quite, absolutely, certainly, really, yes, no and various rudimentary utterances rendered by mm, mhm in writing. They signal contact, support but at the same time reaction and attitudes e.g.:agreement, dislike, surprise, doubt (see Oreström,1983 and Sigurd,1984).

Non-verbal signals generally accompany dialogue and play a very important role. Often backchanneling is handled on the non-verbal channel, e.g. by nodding, eye movements, shaking the head while the speaker is speaking. This is very practical as it allows simultaneous communication and avoids two voices on the same channel.

In practical analysis of dialogues the model presented may seem too simplistic and many problems of identification and categorisation occur , but for our present purpose it is sufficient.

RULES

1. C(YCLE) $\longrightarrow$ { S(TATEMENT) ; I(NSTRUCTION) ; Q(UESTION) + A(NSWER) } (B(ACKCHANNEL))

A CYCLE IN A DIALOGUE MAY CONSIST OF A STATEMENT OR AN INSTRUCTION RESPONDED TO BY A BACKCHANNEL ITEM BY THE PARTNER. A CYCLE MAY ALSO CONSIST OF THREE UNITS: QUESTION, ANSWER AND BACKCHANNEL.

2. S $\longrightarrow$ NP + VP (NP) (PREP PHRASE).	A STATEMENT MAY CONSIST OF A NOUN PHRASE AND A VERB PHRASE WITH ADDITIONAL UNITS
3. Q $\longrightarrow$ VP + NP (NP) (PREP PHRASE)?	A QUESTION HAS INVERTED ORDER BETWEEN VERB AND SUBJECT
4. I $\longrightarrow$ Vimp (NP)!	AN INSTRUCTION (COMMAND ETC) HAS AN IMPERATIVE FORM
5. A $\longrightarrow$ { I DON'T NO. ; YES/NO }	A TRUTH QUESTION IS ANSWERED BY YES OR NO, IF THE ANSWER IS KNOWN
6. B $\longrightarrow$ I SEE, OK, QUITE...	BACKCHANNEL ITEMS ARE A FEW STANDARD PHRASES

SAMPLE DIALOGUE

1. I: A SHIP IS APPROACHING THE SUBMARINE.(S)	1:st CYCLE
2. II: OK.(B)	
3. II: IS IT A DESTROYER?(Q)	2:nd CYCLE
4. I: I DON'T KNOW.(A)	
5. II: I SEE.(B)	
6. II: SEND A HELICOPTER!(I)	3:rd CYCLE
7. I: OK.(B)	

I, II are the partners of the dialogue. The letters S, B, A etc are defined by the rules above. Different cycles are shown.

Fig.1. Prase structure rules indicating the grammatical structure of utterances in dialoge and how the main types of utterances occur in cycles in dialogue. The sample dialogue below illustrates the rules.

A model of a turn-taker

A person who is to participate in a dialogue must know the language, but that is not enough. He must also master the turn-taking rules; he must react properly and be willing to take initiatives. The rules of turn-taking are outlined in fig.2. (It is in fact a flow chart which has been used in computer simulation of communication, Sigurd 1985). It is, of course, related to fig.1,

A person who participates in a dialogue must listen to (and look at) his partner all the time in order to find out whether he is being addressed or not. It is not even possible to stop listening when one is talking, as the partner might want to interrupt. The partner may also give backchannel signals while one is speaking.

A participant in a dialogue must listen to certain signs which indicate that he has to do something. If his name is mentioned (which people, according to experience, notice even at very noisy parties), he has to watch out. It might be a vocative term, which means that he is being called upon to answer a question, execute an instruction etc.

The main types of utterances are characterized by special intonation contours and syntactic means as was noted above: the question by inverted word order, the instruction by a special verb form. When he finds that he is being addressed, there are various possibilities as indicated by the flow diagram. If the participant (2 in the diagram) is being asked something, it is proper to answer the question. Studies of dialogue indicate that questions are rarely left unanswered. It is considered impolite not to answer a question and it can be considered to be breaking the contract of cooperation which is signed when two persons carry on a dialogue. In order to answer, the memory - which may be thought of as a data base - must be searched. Generally, this takes time and there is naturally a delay before the answer is given. If an answer is given immediately, it sounds strange - as if the answer is completely self-evident or the question is not taken seriously. Truth questions require searching for the fact questioned, e g "Is the ship a destroyer?" WH-questions require searching for the parts of the fact given and filling in the lacking piece, if it can be found, e g "Who is in charge of the helicopter operations?"

If one is not being asked a question one might have been instructed or ordered to do something. This case is shown next in the diagram and it also requires a reaction. Normally one has to say whether one is going to carry out the request or not. (The important thing is, of course, to carry out the instruction.)

The next situation depicted in the diagram occurs when a statement has been uttered. If one is doubtful, one might, as indicated in the diagram, utter a back question, e g "Really?" This is a truth question and has to be answered by the partner by yes or no? Should the dialogue participant not be doubtful, however, he may acknowledge the statement by saying "OK, I see, Fine" etc, or some other backchannel item after having registered the fact in memory - adding it to his data base.

The right hand part of the diagram shows typical responses to the initiatives of the partner. It is, however, also important to take initiatives. The proper moment to take an initiative is when the word is free, after a cycle has been completed. One might, of course, also break in before, but that case is not illustrated by the diagram.

Two of the types of initiatives encoded by special syntactic means by natural languages have to do with information. The statement is used to inform the partner: the question is used when the speaker wants to be informed. Thinking about human knowledge in terms of data bases, we might say that the speaker makes a statement when he wants to add something to the listeners data base. Why the speaker wants to add this fact to the memory of the listener is a separate question, but a model of a communicator must have an intention to communicate.

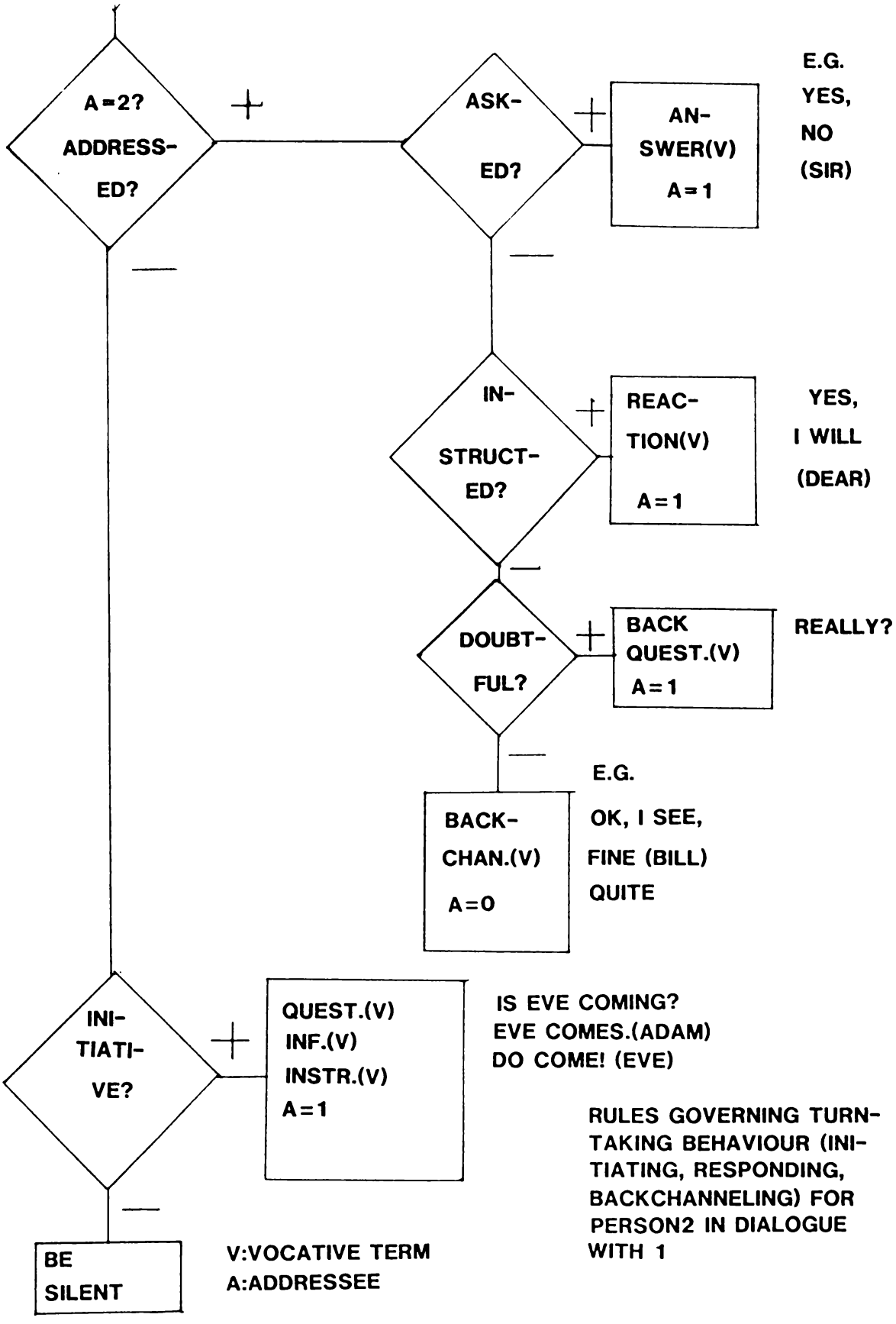

A=2?
ADDRESS-
ED?
ASK-
ED?
AN-
SWER(V)
A=1
E.G.
YES,
NO
(SIR)
IN-
STRUCT-
ED?
REAC-
TION(V)
A=1
YES,
I WILL
(DEAR)
DOUBT-
FUL?
BACK
QUEST.(V)
A=1
REALLY?
E.G.
BACK-
CHAN.(V)
A=0
OK, I SEE,
FINE (BILL)
QUITE
INI-
TIATI-
VE?
QUEST.(V)
INF.(V)
INSTR.(V)
A=1
IS EVE COMING?
EVE COMES.(ADAM)
DO COME! (EVE)
BE
SILENT
V:VOCATIVE TERM
A:ADDRESSEE
RULES GOVERNING TURN-TAKING BEHAVIOUR (INITIATING, RESPONDING, BACKCHANNELING) FOR PERSON2 IN DIALOGUE WITH 1

FIG. 2

Modeling transfer of information

As hinted at above human dialogue can partly be seen as a successive mutual updating of the databases of the partners. But this is done in a very subtle way, where the speakers ideally indicate their 'knowledges", assumptions and attitudes.

Fig.3 shows a number of different communicative situations and related utterances. Each partner is assumed to have a picture of the "real" world as he sees or imagines it. The real world which I is trying to convey information about is the marine scene mentioned above. I's and II's pictures may differ as indicated.When I says something he is adding something to a verbal discourse world which is also indicated in the figure. Similarly the partner II is assumed to have a verbal discourse world which is being changed as he understands I's utterances. It is important to note that the "real" worlds and the verbal worlds are not identical. What I says is only one out of an infinite number of comments which could be made, and his comments do by no means give all the details of the situation.

When the speaker I says:"A ship is approaching the submarine" he may have a picture of the "real" world as depicted in fig 3. His verbal world may be represented by the logical formule APPR(S,U), where APPR is short for "is approaching", S is the ship and U the submarine. It is clear that he assumes that his partner II knows about the existence of the submarine. This is indicated by the square brackets around EXT(U),and the occurence of the same item in II's world indicates that I's assumption is correct. The new information is thus only EXT(S) and APPR(S,U) and one may say that only APPR(S,U) is primary, EXT(S) may be derivable.

II may be said to have understood, when he has updated his verbal discourse world, but that does not mean that II has the same picture of the "real" world as I. There are many places where the ship may be located, satisfying the conditions imposed by approaching. The ship could approach from the North, from the East etc. It could be close, it could go fast etc. The listener fills in the details and he may guess wrongly (as in fig 3).

If the answer is OK or "I see" it shows that II has just stored the new piece of information in his memory. But human partners often give other backchannel items, e.g."Fine" which means that the information is favourable in some way. One may think of a situation here where the ship by approaching comes within reach of the torpedos of the submarine. Using such emotional backchannel items requires information of the intentions and plans of II. This is also the case if disproving items such as "Blast" are to be used. One may imagine a situation where the approaching of the ship interferes with the plans of the submarine, which is now threatened.

A common situation occurs when the partner already knows and this situation is depicted below the line. In such a case the databases of the partners I and II are identical as is shown. II could just indicate this by saying "I know". Often the partner gives comments which indicate how well he thinks his parter describes the situation,i.e. how well the verbal world agrees with the real world. Such comments may be called metacomments and they may indicate that the wording is perfect in which case backchannel items such as "Quite,Precisely" Swedish"Precis, Just det" are used. These are in fact very frequent words. The speaker II may comment by saying "Yes, to say the least", which indicates that he does not think the wording fits the situation, in this case maybe because the ship is rushing towards the submarine at great speed, in which case approaching seems to weak (almost euphemistic).In other cases the partner may agree on the situation but think that the wording is to strong. Types of metacomments are given in Sigurd(1986).

If the partner knows that the situation is different he may give the backchannel"No"or he me say"Really?" This situation is depicted by the last pictures where the verbal world includes contradictory information : The ship is going away from the submarine: DIST(S,U).

However the different worlds are depicted or represented, it is clear that dialogue utterances are not just automatic readings of facts in databases, as some designers of expert systems seem to believe. Simulating the richness of human dialogue is a great challenge.

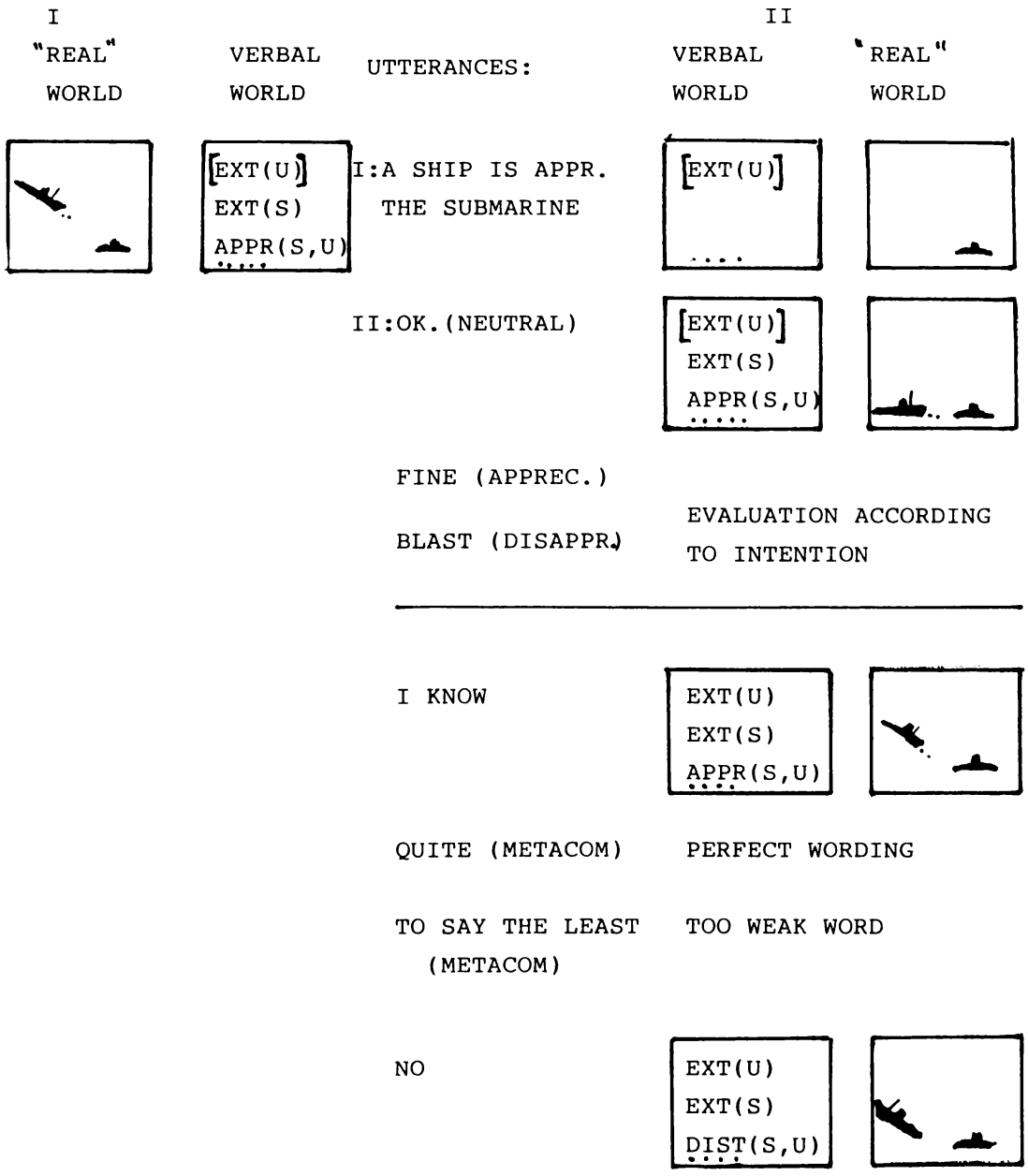


Fig.3
STATES OF THE REAL AND VERBAL WORLDS OF PARTNERS I,II AND SAMPLE UTTERANCES. NOTE THAT THE METACOMMENTS COMMENT ON THE DISCREPANCIES BETWEEN THE REAL AND THE VERBAL WORLD.

References

Brady,M.& R.C.Berwick (eds)1983.Computational models of discourse. Cambridge:MIT Press

Chomsky,N.1957. Syntactic structures. The Hague:Mouton

" 1965. Aspects of the theory of syntax.Cambridge:MIT Press

" 1975. Reflections on language. New York:Pantheon

Oreström,B.1983.Turn-taking in English conversation.Lund:Liber

Sigurd,B.1982.COMMENTATOR.A computer model of verbal production. Linguistics 1982,20

1984."Om Jaså, Bra, Precis och andra returord"Språkvård 1984:3

1985.Dialog på dator (CONVERSATOR).In:Praktisk Lingvistik 11.Lund:Dep. of linguistics

1986."Ärligt talat, s.m.s. och andra metakommentarer" Språkvård 1986 (to appear)

Stenström,A-B.1984.Questions and responses.Lund:Liber

Margareta Sjöberg
Center for Computational Linguistics
Uppsala University

ON THE IDENTIFICATION OF STEMS IN FASS

FASS is the Swedish drug catalogue. An information retrieval system FASSIS has been designed and implemented by LINFO AB and Uppsala University Computing Center (UDAC) with this as a basis. In this system every word occurrence is treated as a key word.

Work has been under way since spring 1985 at the Center for Computational Linguistics to produce a key word index of basic forms for FASSIS by semi-automatic means. One step towards this goal is to build a stem dictionary covering the material (see further Sågvall Hein 1985a). Two parallel lines have been followed side by side. On the one hand, the full text of FASS has been investigated and prepared for treatment by the Center's program package for linguistic text processing, TEXTPACK, which produces word lists, concordances, etc (see Rosén 1986 and Rosén & Sjöberg 1985 for further details). On the other hand, the present key word file in FASSIS has been used as a test file in attempting to identify stems in the material. This presentation will concentrate on these attempts.

For the sake of illustration an extremly short but in other respects representative drug description from FASS is presented in fig. 1 below.

Capsolin
Parke-Davis
Salva
Smärtstillande salva Grupp 12F 0510
Deklaration: 1 g innehåller: Oleoresin capsic. 12 mg, camphor. 52,5 mg, aetherol. tereb. 97,5 mg, eucalypti aetherol. 25 mg, cera flava, vaselin. et odor q.s.
Upplysningar. Parke-Davis, tel. 08- 82 03 50.
Egenskaper. Ökar hudgenomblödningen och ger en värmekänsla i det behandlade området.
Indikatorer. Lokal behandling vid tillfällig huvudvärk.
Försiktighet. Överkänslighet mot ingående beståndsdelar, speciellt terpentinolja, kan förekomma.
Dosering. Appliceras tunt och ingnides lätt några gånger dagligen. I barnpraxis spädes helst med 3-4 delar vaselin.
Observera. Capsolin skall ej komma i kontakt med ögon, slemhinnor eller skadad hud. Händerna tvättas väl.
Förpackningar och priser. Salva
35 g 17:40

- fig. 1. -

1. Strategy.

Our aim has been to identify automatically as many stems as possible starting from the present key word index. One problem with this file is that all words longer than fifteen letters have been chopped off at the fifteenth letter. Another problem is that the distinction between capital and common letters is not maintained in the file. When TEXTPACK produces a correct and complete file of the vocabulary of the material, with this difference kept and with deleted letters beyond the fifteenth restored, this file will replace the present test file.

A study of the material shows that there are a lot of exceptions from standard Swedish morphology. Many word forms are of foreign origin or are numerical expressions or hybrids as illustated below.

... ibland av karaktären bull's eye ...
... aktivt ulcus ventriculi et duodeni ...
... mixtur 40 mg/ml ...
... 24 st vnr 411496 ...
... cirka 1-2 timmar ...
... uppges till 80-85% ...
... 12:-/st ...

In all there are ca. 40,000 graphical words in the key word file among which ca. 10,000 are purely numerical expressions. These have been removed from the target set for the stem identification programme.

As a first step in the stem identification process I wanted to mark, and thus, for the time being, remove from further analysis, words of foreign origin, together with abbreviations and proper names. And FASS contains a lot of them! There are, for example, a large number of names of drugs and companies which produce them, of Latin names of chemical substances and abbreviations of these. There are, furthermore, a lot of Latin names of organs, illnesses, bacterial species and so forth, together with English expressions and quotations, with references. Quite a number of these "special words" can be identified by capitalising on the internal structure of the drug descriptions. Thus, names of drugs (1,372) and companies which produce them (150) occur in posts which are specially marked in the FASS source file. As for the greater part of the abbreviated Latin names of chemical substances, they occur under the subheading of 'DFK' - for "declarations" (see fig. 1). Ca. 1,200 different words ending with the full stop of abbreviation have so far been identified in these sections. These names and abbreviations are marked in the source file.

There is a concentration of Latin names of chemical substances in the 'DFK'-sections, and, as the result of the envisaged TEXTPACK treatment of the source file will allow us to treat different parts of the material as a corpus of its own, it will also for example be possible to gather a large number of the Latin words by picking out word forms in 'DFK' sections, which <u>neither</u> contain the full stop of abbreviation <u>nor</u> occur in the remainder of the text. With the distinction of words with capital and common initial letters maintained we will also be able to heuristically identify the remaining proper names in the text assuming that words which <u>only</u> occur with an initial capital letter are proper names.

While waiting for the TEXTPACK preliminaries to be completed, Latin, English and French words and expressions as well as proper names (other than the ones already marked) have been entered manually into special files as they crop up on inspection of preliminary results of rough test marking in the key word file. At present the English "dictionary" contains 110 words and expressions, the latin "dictionary" 380 and the list of names 145, the latter comprising mainly names of persons, journals and drugs. After this marking there remain ca. 27,000 unmarked word forms for further analysis.

The next step towards building a stem dictionary was to identify automatically as many stems as possible among these remaining word forms. For this purpose I have chosen to work with heuristic rules for the recognition of word endings which I have expressed in Brodda's BFTA system (for a description of the system see for example Brodda & Karlsson 1980). Because of the size of the material and the definite need for manual inspection of the result produced by the heuristic rules I found it necessary to concentrate on smaller parts of the material at a time, gradually trying to correct mistakes in the analysis more or less manually by adding "exception rules". I therefore divided the set of rules into groups that tentatively mark approximately 2,000-5,000 word forms each, the markings then being carefully checked and new rules added until the set of words is correctly analysed and marked.

A stem dictionary has been created with the help of the markings introduced into the key words, and word forms containing these stem are removed from the source file. So far, ca. 16,500 different stems have been identified in FASS, and with word forms containing these stems removed from the source file there remain ca. 7,000 unmarked forms. Of these, ca. 2,800 are words which have been chopped off at the fifteenth letter, the majority of which will be ascribed a correct analysis by our rules when restored to their

full length. The rest remain to be treated.

2. Comments on the BETA rules.

The rules I have devised analyse word endings only. They mark stems in word forms depending primarily on whether they contain a characteristic suffix or a derivational component. The rules are divided into five mutually independent groups, two of which concentrate on identifying noun suffixes, one on verbs and one on adjectives. In the fifth group I employ a list of final derivational strings which occur frequently in the text.

The first set of rules includes the most distinctive suffixes '-arna(s)', '-erna(s)', '-orna(s)', '-ar(s)','-er(s)' and '-or(s)', the central rules being*

```
'ARNA' --> '=ARNA/s'
'ERNA' --> '=ERNA/s'
'ORNA' --> '=ORNA/s'

 'AR'  -->  '=AR/s'          after 'D','G','L','M','P','S'
 'AR'  -->  'A=R/v'          otherwise

 'ER'  -->  '=ER/v'          after 'G','J'
 'ER'  -->  'ER=/s'          after 'C','K'
 'ER'  -->  '=ER/s'          otherwise

 'OR'  -->  'OR=/s'          after 'U'
 'OR'  -->  '=OR/s'          otherwise
```

There is also a set of rules by which incorrect analyses generated by these heuristic rules can be avoided. We call them "exception rules" simply. Let the following examples suffice as an illustration.

```
  'KAR'   -->  'K=AR*/s'          after 'C','J','N','S'
 'KVAR '  -->  'KVAR/adv'
'DELAR'   -->  'DELAR/hom'
'MOLAR'   -->  'MOLAR/adj'
'SPELAR'  -->  'SPELA=R/v'
```

---

* '/s', '/v' , etc. are form class tags denoting respectively nouns, verbs, etc.

In all there are 230 rules in this set, and they mark a total of 2,813 stems (455 are verbs, 50 adjectives, 22 homographs and the rest nouns. Examples of homographs are 'delar', 'klumpar', 'pumpar', 'isomer'.). The number of "exception rules" is 210.

The main rules in the next set of rules which deal with possible noun suffixes are

```
 'AN'  -->  '=AN/s'
'ANS'  -->  '=ANS/s'
'ATS'  -->  '=ATS/s'

 'EN'  -->  '=EN/s'
'ENS'  -->  '=ENS/s'
 'ET'  -->  '=ET/s'
'ETS'  -->  '=ETS/s'
```

Even more rules for exceptional cases must be added here in order that mistakes in the analysis should not multiply inordinately. There are altogether 287 rules (278 "exceptional" ones) here which together mark 3,100 stems (32 adjectives, 38 adverbs, 110 verbs, 20 homographs and the rest nouns).

The rules in the third set are mainly concerned with the identification of adjective/participle endings. The emphasis lies on suffixes like '-da', '-dd', '-ld', '-rd', '-ad', '-at', '-ade', '-ande', and '-ende'. Word forms ending with '-as' and '-es' are also marked here. There is a total of 310 rules in the group, marking 3,275 stems (ca. 2,300 verbs, including participles, 80 adjectives and the rest nouns). The number of "exception rules" is 276.

With the rules in the fourth group 2,484 stems have been identified, the majority of which (1,925) are adjective stems and the remainder nouns and verbs. The word endings recognised here are '-igt', '-iga', '-iskt', '-iska', '-fritt', '-fria', '-bart', '-bara', '-lt', '-la', '-mt', '-mma', '-nt' '-na', '-vt', '-va', '-ta', '-ärt', '-ära', and '-are', '-ast',

'-aste'. The group consists of 230 rules in all, 180 "exceptional".

A large number of words in the text contain no explicit inflexional ending and many of them therefore remain unmarked by the above rules. But many of these contain a derivational component indicating that the word as whole belongs to a given category, the stem being identical to the word itself. Examples of derivational strings of this kind are '-id' (hexicid, jodid, ureid, karbamid; gravid, fungicid, cyticid, vermicid, baktericid, tyfoid, myceloid, ...), '-fri' (valfri, alkoholfri, symptomfri, kaliumfri, ...), '-isk' (biologisk, urologisk, allergisk, kirurgisk, alkalisk, ...), '-är' (bacillär, lågosmolär, bipolär, muskulär, högmolekylär, ...), '-ig' (mjölkig, flockig, lindrig, ...), '-ing' (odling, mässling, pensling, rubbning, välling,..), '-tion' and '-sion'. A list of a number of final components in compound words frequently occurring in the text such as 'terapi' (67), 'medel' (50), 'dos' (70), 'virus', 'status' and 'enzym' is also used here. Altogether the rules in the group number 445 (ca. 210 "exceptional") and correctly mark around 7,000 words.

3. Future plans and concluding remarks.

Not all the stems in FASS have yet been identified, and work on the remaining stems continues. Certain suffixes, for example '-a', are so insignificant that it has not been practically feasible to consider them as yet, but now they might be of some help. And purely manual methods will presumably have to be resorted to in the final stages.

It also remains to deal manually with the ca. 300 words and stems marked as homographs (beroende, buffrar, format, ... and allergen=, hosta=, ...) and to check for pure adjectives in the words marked as participles.

The BETA system has proved to be practical to work with. It is a simple

matter to express rules within the system for the sort of treatment we wanted to give to our word form file. That the number of exception rules is large is not, of course, something for which the BETA system can be blamed. Certainly it is possible that we could have managed with fewer; however, the aim has not been to build up as compact a system of rules as possible, but rather to identify the stems in the text as efficiently as possible. The number of rules which are of a purely lexical character is also so large that it would seem to be very difficult to reduce them dramatically. A trial run, albeit preliminary, with a version of Brodda's SWEMORF pointed in the same direction.

In parallel with the work described in this paper preparartions are under way for the next step in our project, i.e. the integration of the stem dictonary in an automatic morphologic analyser. One morphological model for the inflectional analysis has been tested for some of the identified stems (Sägvall Hein 1985b). The specification of the particular word class to which the identified stems belong was made in order to facilitate this phase.

BIBLIOGRAPHY

Brodda Benny, and Karlsson Fred, 1980. "An experiment with Automatic morphological analysis of Finnish". Papers from the Institute of Linguistics, University of Stockholm, Publication 40. (Reprinted 1981: Department of General Linguistics, University of Helsinki, Publication, No.7.)

Rosén Valentina, 1986. "Förberedande undersökning av FASS för automatisk nyckelordsindexering." Center for Computational Linguistics, University of Uppsala. (forthcoming)

Rosén Valentina, and Sjöberg Margareta, 1985. "TEXTPACK, programpaket för språkvetenskaplig textbearbetning". Center for Computational Linguistics, University of Uppsala.

Sågvall Hein Anna, 1985a. "Automatic Key-Word indexing for FASSIS. Proposals for a project". Center for Computational Linguistics, University of Uppsala.

---- 1985b. "Parsing by means of Uppsala Chart Processor". In Bolc, L (ed.) Natural Language Parsing Systems. Springer-Verlag. (forthcoming)

Annette Östling Andersson
Dept of Romance Languages
Uppsala University

A TWO-LEVEL DESCRIPTION OF WRITTEN FRENCH

In his thesis, K. Koskenniemi presents a new model for morphological description, two-level morphology (1). The model has two components:

1) a lexicon component
2) a rule component

The lexicon component consists of lexicons for different morphological elements, such as stems and endings. Continuation classes constitute the links between the lexicons.

The rules specify equivalences between pairs of signs on two levels, a lexical level and a surface level.

lexical level	n e z + s
surface level	n e z 0 0

The two-level model has already been applied to a number of languages: Finnish (2), Swedish (3), English (4), and Polish (5) are some examples. My contribution is a description of the inflectional morphology of written French.

A very important matter to decide upon when setting up a two-level description is the following: which inflectional phenomena are to be treated in the lexicon system, and which ones are best described by rules?

In order to get a survey of the different types of inflectional phenomena that occur in French, the following classification is of help:

a) flexion réguliere

livre + s --> livres
parl + e --> parle

i.e. regular inflection

b) variation réguliere

nez + s --> nez
mang + ons --> mangeons

i.e. deviation from regular inflection occurring for all words with a certain structure

c) variation irréguliere

complet + e --> complète
achet + e --> achète

i.e. deviation from regular inflection taking place only for some words with a certain structure

The rule component is designed for those inflectional phenomena that are productive, frequent and do not affect long sequences of signs (6). The stem lexicon together with minilexicons for affixes account for regular inflection, as well as for those phenomena that deviate from regular inflection, but do not depend on the phonological or morphological context. The stem lexicon can also be used to list irregular word forms.

Category a), regular inflection, should thus be described in the lexicon component, whereas category b), context-dependent deviation from regular inflection, is best described by rules. The lexicon component is best suited to take care of those phenomena that belong to category c), deviations from regular inflection that are not context-dependent.

I have used 15 lexicons of endings to describe the nominal morphology. Two of these take care of category a), while the other 13 cover category b).

Stem lexicon

```
grandeur    /NO   "N F";
gra  s/A          "gras A";
gratuit     /A    "A";
```

Regular inflection

```
LEXICON /NO         number: singular or plural
LEXICON /A          adjectives; points to /NO
```

Variations irrégulières

```
LEXICON /NOx        plural -x: chou - choux
LEXICON ail/NM      travail - travaux
LEXICON eau/A       beau - bel - belle
LEXICON ou/A        fou - fol - folle
LEXICON et/A        complet - complète
LEXICON et/ADJ      net - nette
LEXICON s/A         gros - grosse
LEXICON c/A         public - publique
LEXICON nc/A        blanc - blanche
LEXICON ng/A        long - longue
LEXICON gu/A        ambigu - ambiguë
LEXICON eur/A       trompeur - trompeuse
LEXICON teur/A      multiplicateur - multiplicatrice
```

If the number of words with a certain "variation irréguliere" is > 1, I have always decided to set up a lexicon instead of enumerating inflected forms in the stem lexicon.

In order to handle category b), "variation réguliere", there are six rules. I show them here in a simplified form.

1)	s,x,z + s --> s,x,z	nez + s --> nez
	e + e --> e	malade + e --> malade
2)	er + e --> ère	entier + e --> entière
3)	x + e --> se	heureux + e --> heureuse
	f + e --> ve	passif + e --> passive
4)	al + s --> aux	principal + s --> principaux
5)	el + e --> elle	naturel + e --> naturelle
	en,on +e --> enne,onne	moyen + e --> moyenne
6)	eu,au + s --> eux,aux	jeu + s --> jeux

The inflectional morphology of the verbs is more complicated than that of the nominals.

The French verbs are usually divided into three regular conjugations. There are also about 150 irregular verbs. The endings of the different tenses and moods differ in most cases between the conjugations, and no element always signalling for example person or number exists. The ending -rai, for instance, has as its signifié futur, 1st person, singular, and -irent passé simple, 3rd person, plural. Thus it is obvious that the number of lexicons for endings becomes large. By keeping the endings as constant as possible and, when necessary, using more than one stem, a good descriptional economy is achieved (7). A regular verb, such as parler, gets one entry in the stem lexicon, while an irregular verb, vivre for instance, gets more than one. Exceptions exist, though. So far I have encountered one irregular verb that needs only one entry, courir:

```
cour      V19  "courir";
parl      V1   "parler";
viv       V27  "vivre";
vi        V6 ;
véc       V11 ;
```

To each stem is attached a reference to a continuation class (V1, etc., above). The continuation class then points to the lexicons containing the endings possible to add to the stem. V1, for example, points to 15 lexicons. There are 62 continuation classes, 57 of which are needed for the description of the irregular verbs. The lexicons I use are:

Finite forms

8	lexicons for	indicatif présent
1		imparfait
3		futur
3		conditionnel
4		passé simple
2		subjonctif présent
4		subjonctif imparfait
4		impératif
1		the infix -iss-

All of these are not complete lexicons for all six persons. One of the lexicons for subjonctif présent, for instance, consists of the endings for the singular and the 3rd person in the plural, whereas the other one contains the endings for the 1st and 2nd person in the plural.

Infinite forms

4	lexicons for	infinitif
5		participe passé
1		participe présent
2		the inflection of the participles

Four rules take care of the "variation réguliere" among the verbs:

1)	c + a,o --> ca,co	plac + ons --> plaçons
2)	g + a,o --> gea,geo	mang + ons --> mangeons
3)	oy,uy + 9e --> oie,uie (8)	employ + 9e --> emploie
4)	= + 9e --> =e (9)	parl + 9e --> parle

The contexts specified within the two-level framework are word-internal. The two-level model analyses the words as isolated units, and gives all the possible analyses. In this respect it works very well for French, whose inflectional phenomena are all accounted for in a satisfactory way, with one small exception. A handful of nouns, such as arc-en-ciel get their plural -s after the first noun: arcs-en-ciel. These plurals have to be listed in the stem lexicon.

Homographies are very frequent in French, a language that can be characterized neither as analytic nor as synthetic. The homographs can be divided into two groups:

1) homographies between inflectional forms of a word

nez	nom masculin singulier
	nom masculin pluriel

2) homographies between parts of speech

maintenant	adverbe
	verbe participe présent

The homographies of both types are disambiguated by the syntagmatic relations of the sentence. A good deal of the informational contents is thus to be found in the relations between the words. Since the two-level model only is concerned with the isolated sequences of signs constituting words, the number of words with more than one analysis gets very high for French. This is not the case for languages such as Finnish, whose morphological elements carry more information.

Often the analysis resulting from the two-level model is quite satisfactory, but for the purpose of vocabulary studies the definition of the word as a sequence of signs separated by blanks is not adequate for a language like French. It becomes necessary to push the analysis further and disambiguate the homographies. For the same purpose, discontinuous signs have to be accounted for. In French they occur for example among the verbs (ai/ /parlé) and the negations (ne/ /pas). Since this process necessarily involves syntax, it is no longer within the scope of the two-level model.

Which homographs can be separated automatically, which segments have to be identified to achieve an analysis, and how are these segments identified? These are interesting questions to which I will try to find some answers in my future research. I will also investigate how the two-level model is to be combined with a subsequent syntactic analysis.

In order to test the two-level description, I have a corpus consisting of 102 signed editorials from the daily French newspaper Le Figaro. Together they constitute 50012 running words.

At present there are in the stem lexicon about 2830 nouns, proper names, adjectives and abbreviations, and about 1400 verbs,

i.e. all the words belonging to these categories up to 30000 running words.

Notes

(1) Koskenniemi, K., Two-Level Morphology: a General Computational Model for Word-Form Recognition and Production, Helsinki 1983.

(2) Koskenniemi, op.cit., pp. 42-88.

(3) Blåberg, O., Svensk böjningsmorfologi. En tvånivåbeskrivning, Helsinki 1984.

(4) Karttunen, L. and K. Wittenburg, A Two-Level Morphological Analysis of English, Texas Linguistic Forum 22 (1983), pp. 217-228.

(5) Borin, L., A Two-Level Description of Polish Inflectional Morphology, Center for Computational Linguistics, Uppsala University (forthcoming).

(6) Koskenniemi. op. cit., pp. 20-21, 42.

(7) Cf. Maegaard, B. and E. Spang-Hanssen, La segmentation automatique du francais écrit, Paris 1978, pp. 23-27.

(8) "9" stands for "unstressed verb ending -e".

(9) "=" stands for "any sign that is not mentioned in this rule, but is mentioned in some other rule".