GÖTEBORGS UNIVERSITET
INST FÖR SVENSKA SPRÅKET

GU-ISS-2016-04

Huvudansatser för parsningsmetoder

Om programutvecklingens förutsättningar i en svensk kontext

Kenneth Wilhelmsson

Forskningsrapporter från institutionen för svenska språket, Göteborgs universitet
Research Reports from the Department of Swedish

ISSN 1401-5919

www.svenska.gu.se/publikationer/GU-ISS

Förord

Syftet med denna text var att ge en inblick i området (syntaktisk) parsning. Tanken var att ge en bild av utvecklingen som var 1) fri från alltför tekniska detaljer, då området är programmeringstekniskt, och 2) beskriven ur ett svenskt perspektiv.

Bakgrunden till valet av ämne till texten, som var tänkt att finnas med i antologin *Text och kontext*, var att parsning är relativt okänt för många personer verksamma inom närliggande områden, samtidigt som det är ett absolut nyckelbegrepp för den som ägnar sig åt datorlingvistik eller språkteknologi.

Målet var alltså att ge en ganska allmän utifrånblick på några centrala sidor av utvecklingen, samtidigt som det tydligt är så att den som själv arbetat med utveckling kan ha starka åsikter och preferenser rörande metodval, något som i ärlighetens namn kanske inte heller denna text är lösgjord från.

Hur ska det göras? Konsten att utveckla automatisk syntaxanalys av naturlig text kan läras ut från ett flertal perspektiv. Det kan t.ex. ske med fokus på användandet av en viss grammatikformalism, med fokus på beräkningssnabbhet, med fokus på entydiggörande av möjliga ambiguiteter. Tolkningsval kan göras med hjälp av antingen handskrivna regler eller inhämtad statistik.

En sorts huvudtema i denna text är hur metoder för parsning på senare år uppvisar förändringar som kanske kan förklaras med att programmen har fått andra användningsområden och att metoderna har anpassats därefter (en annan tolkning är att flera senare system inte längre gör parsning i strikt mening).

När detta tänkta ”kapitel” var färdigt fick det kommentaren att det inte var anpassat för antologins målgrupp. Det fick skrivas en annan kapiteltext, men det kom samtidigt ett förslag att publicera texten om parsning här som denna rapport.

Sammandrag

Parsning är kanske för språkteknologin och datorlingvistiken vad grammatik är för språkvetenskapen. Denna text redogör för några huvuddrag inom forskningsfältet parsning, som för svenska pågått sedan 1970-talet i form av olika datorimplementationer. Uppgiften beskrivs här inledningsvis med en klassisk definition, men det beskrivs hur det kan finnas anledning att modifiera antingen den eller namnet på de senaste implementationer som går under denna beteckning. Några grundläggande frågor och problem för uppgiften parsning finns här i fokus. Dessa frågor återfinns både för svenska språket och i den internationella utvecklingen.

1. Satslösning och parsning

Denna text tar upp det inom språkteknisk forskning mycket centrala begreppet (automatisk) syntaktisk parsning (eng. *parse/parsing*). Utgångsläget är svenska språket och närmare bestämt skriftspråk i digital form. Texten är riktad till läsare utan direkta förkunskapskrav inom språkteknologi/datorlingvistik. En förtrogenhet med grammatikskrivning med hjälp av kontextfri grammatik och liknande är dock en fördel i sammanhanget. För läsare med datavetenskaplig bakgrund kan istället kännedom om kompilatorteknik vara av nytta.

Etymologiskt har termen *parsning* kommit från latin och ursprungets betydelse hänger samman med *dela*. En närliggande term i ett allmänt perspektiv som förmodligen klargör mycket för den som inte är bevandrad i ämnet är att området helt enkelt behandlar automatisk *satslösning*. Till skillnad från i engelskan kommer de ”svenska” termerna *parsning* och *parser* här genomgående att användas för den datorimplementerade uppgiften, respektive programtypen. Det är också bara *syntaktisk* analys som avses.

Det som behandlas i texten är några grundläggande förutsättningar och frågeställningar för uppgiften parsning. Det rent tekniska tillvägagångssättet utelämnas dock. Texten innehåller inte heller en uttömmande sammanställning av alla existerande svenska implementationer (se dock sista avsnittet för en viss listning).

Denna text pekar på hur syftena för de moderna implementationerna har förändrat förutsättningarna till den grad att det enligt somliga inte längre är fråga om program för parsning – eller att det är läge för en ny definition.

2. Varför parsning?

I en modern omgivning med myriader av digital skriven text, framför allt på Internet, finns flera lockande möjligheter där text kan analyseras maskinellt istället för enligt dess primära funktion, dvs. att bli läst av människor. Ett vanligt

exempel är *sökning i eller efter dokument* med hjälp av en sökmotor. Sökning är kanske den idag mest använda tillämpningen som kan kallas språkteknisk. Men det ska klargöras att det som är ett skrivet dokument i ett sådant sammanhang enbart representeras på ett tämligen ”kvantitativt” sätt, som *en påse av ord – bag-of-words*. Det är ofta fråga om en strukturlös representation av ett dokument genom dess förekommande ord, och med information som hur många gånger de finns med. Metoder med kvantitativa *bag-of-words*-modeller för språktekniska sammanhang, når ofta resultat som är förvånansvärt bra för uppgifter som sökning och automatisk textkategorisering.

En annan analysmodell behövs för områden som *grammatikkontroll* – eller framför allt i den prestigetyngda och från ett ekonomiskt perspektiv tilldragande uppgiften *automatisk översättning (maskinöversättning).* Just den mycket krävande uppgiften maskinöversättning har beskrivits som den första icke-numeriska datamaskinella tillämpningen. Trots att denna tillämpningstyp kontinuerligt har utforskats sedan datorbegynnelsen och mycket stora ekonomiska belopp har satsats på forskning är mänskliga översättare ännu inte riktigt hotade.

Uppenbart är i sådana tillämpningar hur löpords positioner i en text, till skillnad från i de tidigare nämnda modellerna, har en okänd, men potentiellt helt avgörande, betydelse för tolkningen och resultatet.[1] Med andra ord krävs mycket mer än en *bag-of-words*-representation av texten. Parsning i någon form, dvs. en syntaktisk analys, verkar alltså helt ofrånkomlig. För ambitiösare AI-betonade tillämpningar är syntaxnivån (vilket oftast är resultatet av det som kallas parsning) dessutom bara ett delsteg mot en mer semantisk analys och ambitionen ”förståelse av texten” (ett uttryck som antyder t.ex. medvetande hos maskineriet, vilket det egentligen inte är fråga om).

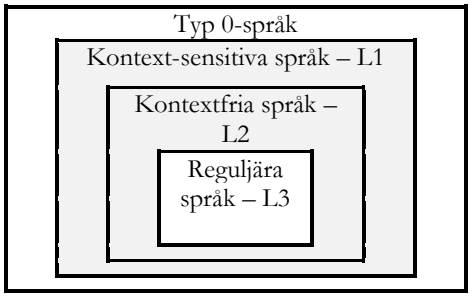
Enbart för svenska har försök till parsning av naturlig text förekommit sedan 1970-talet. Vad olika parsningsprogram som utvecklats för svensk text har åstadkommit i sina syntaktiska analyser är dock något som är märkbart skiftande. Det finns grundläggande skillnader i själva de olika metoder som använts och de utdataformat för resultaten hos tongivande programmerade lösningar internationellt och för svensk text. Beskrivningen här görs dock utan närmare fördjupning i vissa frågeställningar om *matematisk komplexitet* och *tidsåtgång* hos olika metoder, då detta lätt kan leda diskussionen i en annan riktning och riskerar att få ämnet att verka olyckligt esoteriskt, eller matematiskt komplicerat. (Dessa aspekter hör i och för sig också till detta forskningsområde!)

Det finns många andra tillämpningsområden för parsning än de nämnda, t.ex. frågebesvarande expertsystem, informationssökning och informationsextraktion, dialogsystem och talteknologiska system. Men det verkar också rimligt att anta att helt nya applikationstyper som idag inte existerar kommer att utvecklas i framtiden.

[1] Det går ju att rekonstruera olika möjliga texter av samma påse med ord, *Kalle lyssnar inte på kvinnan han känner / Kvinnan Kalle lyssnar på känner han inte,* osv. Dessa (korta) ”dokument” skulle kunna representeras lika i ett sökperspektiv.

3. Den klassiska modellen: grammatiken definierar språket

I forskningslitteraturen återfinns stränga matematiska definitioner av uppgiften datoriserad syntaktisk parsning: uppgiften är *att tilldela en grammatisk sträng som tillhör språket en syntaktisk representation* (ofta i trädform och oftast ett enda träd).[2] De tidigaste programmerade parsningsmodellerna både för svenska och internationellt var mycket präglade av den generativa grammatiken och av dess matematiskhet. För den som kommer från ett mindre matematiskt eller formellt håll är det här viktigt att tydliggöra att ett språk enligt detta synsätt är lika med de satser/strängar som tillhör språket och som kan genereras av språkets grammatikregler med lexikon. Grammatiken (*lexikonet*, vilket består av *terminaler*, räknas ofta in som en del av grammatiken) bestämmer alltså exakt vad språket innefattar – enligt detta rent formbetonade synsätt. En parser som inbegriper en sådan här grammatik kan för varje indata-sträng (ofta sats eller textmening) säga om den tillhör språket, såsom grammatiken definierar det. Parsern, enligt det som kommer att kallas *den klassiska modellen* här, består enkelt uttryckt av två delar: 1) av grammatiken och 2) av en parsningsalgoritm som utför själva arbetet, dvs. jämför strängen mot grammatiken och oftast kan visa eventuella syntaxträd för strängen, enligt grammatiken – förutsatt att strängen tillhör språket som grammatiken beskriver.



Figur: Chomsky-hierarkin (Chomsky, 1959) är inom datorlingvistiken en populär beskrivning av olika grammatikformalismers (sätt att skriva regler i en grammatik) uttryckskraft. Den visar hur starkare språkklasser inbegriper de svagare: Ett språk som uttrycks i en språk-/grammatikklass av typ L3 kan alltså beskrivas av alla de andra, L2-språk kan beskrivas exakt av alla utan L3 osv.

När en grammatik med omskrivningsregler (som t.ex. S → NP VP, enligt exemplet nedan) skapas kan själva reglerna konstrueras enligt olika mönster: olika *språkklasser/grammatikklasser*, kan urskiljas. Skillnaderna kommer här inte att beskrivas i detalj men olika grammatikklasser medger alltså att reglerna får se ut på olika sätt, enligt ovanstående figur.[3] De olika klasserna skiljer sig åt i

[2] Ljunglöf och Wirén (2009) beskriver grammatikdriven parsning just som att avgöra en textsträngs strukturella beskrivning enligt en formell grammatik. Detta är en mycket välfungerande och accepterad definition. Kanske klargör denna text varför nya implementationer inte så tydligt platsar i kategorin.

[3] Om det nu är så att Typ 0-språk (och Typ 0-gremmatiker) är starkast, kan man fråga sig varför inte enbart denna klass används? Svaret hänger ihop med hur mycket arbete som krävs för att skriva grammatiken, och vilka egenskaper ett program får. Program

uttryckskraft, vilket betyder att de är olika bra för att skriva en grammatik som kan skilja ut satser som tillhör ett språk från de som inte gör det. Ett språk som inbegriper komplicerade syntaktiska konstruktioner kräver en starkare grammatikklass. Här kommer framför allt den mycket vanligt förekommande grammatikklassen *kontextfri grammatik (CFG, L2)* att tas upp.

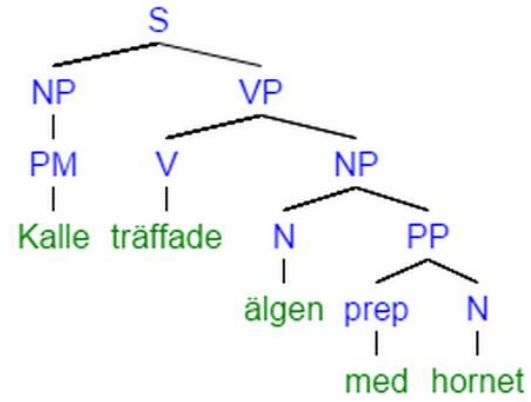
Naturliga språk som svenska brukar karakteriseras som L1 eller L2 – dvs. det krävs (i en sträng teoretisk mening) en parser med grammatik skriven i kontext-sensitivt eller -fritt språk för att känna igen dem om uppgiften tolkas strängt.

Grammatik och lexikon	Syntaxträd för satsen

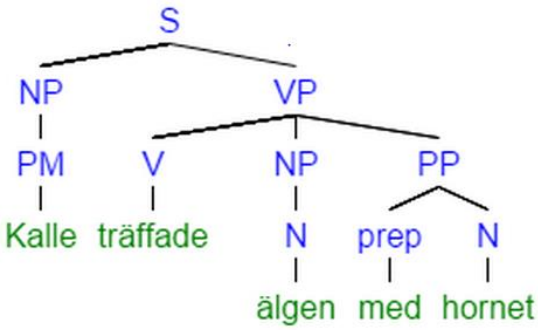
Kontextfri grammatik
(omskrivningsregler)

```
S  → NP VP
VP → V NP
VP → V NP PP
NP → PM
NP → N
NP → N PP
PP → Prep N

Lexikon:
PM → Kalle
V  → träffade
N  → älgen, hornet
prep  → med
```



Analysträd 1



Analysträd 2

Det som här kommer att kallas den klassiska modellen innebar alltså att en parser skulle bestå av två huvudkomponenter: En grammatik, som väldigt ofta var en kontextfri grammatik *(CFG)* [4] samt en parsningsalgoritm. Den här modellen för

som använder L3 (reguljära uttryck/finite state-modeller) är t.ex. ofta snabba och effektiva, och det kan vara en mycket betydelsefull faktor.

[4] Definitionen av ett kontextfritt språk (CFG) är formellt:
`Gcf = <N, T, S, P>`
`N`: Icke-terminaler
`T`: Terminaler
`S`: Startsymbol (S, dvs. Sats, alltså toppnoden i trädet)
`P`: Produktionsregler (Själva grammatiken, vilken består av omskrivningsregler med de ovanstående)

parsning har haft ett enormt stort inflytande på bilden av vad parsning är, både internationellt och i Sverige. För 20 år sedan innebar en kurs i parsning vid svenska universitet i vissa fall enbart studier kring denna modell och olika algoritmer som kunde användas, speciellt för kontextfri grammatik. En språkbeskrivning som görs med en kontextfri grammatik kommer också att få likhet med programmeringsspråk, mer om detta nedan.

I det ovanstående språket rörande Kalle och älgen kan ett litet behändigt antal satser genereras med hjälp av den kontextfria grammatiken och lexikonet. Det viktiga här är att grammatik plus lexikon är en definition av språket. Det betyder att en programmerad version enkelt kan generera alla språkets satser (om det som här är ett ändligt antal). Uppgiften parsning, enligt det som här kallas den klassiska modellen, är att med dessa medel göra motsatsen: att ta emot en sats som indata och med hjälp av grammatiken visa det eller de träd som den motsvarar (dessa är *bevis* för att den tillhör språket), alternativt förkasta den eftersom de inte tillhör språket som grammatiken beskriver. Om uppgiften är fritextparsning för naturliga språk kommer det dock vara fråga om språk med obegränsat antal satser, mer om detta nedan.

I denna rapport kommer termen *träd* att användas generellt för analys, även om parsningsresultat inte alltid ser ut som just träd med en sådan hierarkisk struktur osv., utdataformatet är något som varierar med typ av parser.

Utgångsläget här är att grammatiken med lexikon har ställts upp och bl.a. täcker satsen *Kalle träffade älgen med hornet*. Denna sats är ambiguös (*med hornet* är attribut eller adverbial): det finns två syntaxträd som kan täcka in den. Detta är helt i sin ordning i den aktuella satsen, eftersom det verkligen finns två rimliga tolkningar, just precis i denna sats.

Tyvärr finns det nog ofta ett missförstånd när det gäller varför denna flertydiga exempeltyp används så ofta i undervisningen gällande parserbygge (eller grammatikbygge rent teoretiskt). Användning av en trädgrammatik är alltså utmärkt för uppgiften ibland just eftersom det så tydligt klargör hur samma sats kan ha två synbart olika syntaktiska tolkningar som förklarar flertydigheten.

– Men det verkliga problemet finns hos satser som bara borde ha en enda rimlig analys. Även de får ofta många olika träd i den klassiska ansatsen, och många av dessa träd är orimliga.

Om samma grammatik (med utökat lexikon) används för att korrekt tolka *Jag läste brevet till familjen* är det tydligt så att *till familjen* bara kan vara *attribut* – bara ett träd är giltigt. Men det är alltså så att en grammatik som byggs för att täcka många olika tolkningar ofta ger många orimliga tolkningar i (för mänskliga läsare) entydiga fall. För att utveckla vidare kan en annan sats: *Jag läste brevet till maten* exemplifiera hur lexikal information (uttryckt som valens eller liknande) måste beaktas – i det fallet ska *till maten* istället vara adverbial. Denna typ av ambiguitetslösning: att bestämma just vad efterställda prepositionsfraser

---

Generellt regelformat för produktionsregler:
$A \rightarrow \gamma$, där A är icke-terminal och γ är icke-terminaler eller terminal

(PP) ska vara i ett visst sammanhang *(PP attachment*-ambiguitet) är en så central fråga inom språkteknologin att det nästan är ett eget forskningsfält.

Detta exemplifierar syntaktisk flertydighet som ett absolut kärnproblem inom parsning, inte bara för svenska utan för många världsspråk.

Och PP-attachment är tyvärr långt från den enda typen av strukturell flertydighet (konstruktionshomonymi) som gör parsning av fri text svårt. (Annars skulle språk utan prepositioner som finska kanske sakna syntaktisk flertydighet.)

4. Kort om praktiskt förarbete i parsningssystem

Innan problemen med parsning enligt den klassiska modellen behandlas vidare ska här sägas något om de allmänna processer som måste inleda parsning av fri okänd text i så gott som alla olika ansatser, gamla som nya. När en parser byggs för fri text så *förprocessas* texten i minst två steg före det att den jämförs mot grammatiken (själva parsningen). För det första sker en *tokenisering* där varje löpord identifieras. Det är en uppgift som i stora stycken är trivial: teckensekvenser åtskilda av mellanslag och skiljetecken är ofta lika med dessa löpord. Därefter vidtar processen *ordklasstaggning (part-of-speech tagging, POS-tagging).* Detta är en relativt komplicerad uppgift där varje löpord ska ges rätt ordklassinformation och morfologisk information enligt en speciell taggуppsättning. För svensk språkteknologi används ofta taggarna i *Stockholm Umeå Corpus*. Med särdrag finns det lite över 150 möjliga taggar (ordklass och annan morfo-syntaktisk information, enligt nedan) och varje löpord ska tilldelas en riktig sådan tagg som ordet svarar mot just i den aktuella kontexten.

Löpord:	*Det*	*finns*	*inga*	*mysterier*	*där*	*.*
Ordklass-taggning med särdrag:	**PN**	**VB**	**DT**	**NN**	**AB**	**MAD**
	NEU	PRS	UTR/NEU	NEU		
	SIN	SFO	PLU	PLU		
	DEF		IND	IND		
	SUB/OBJ			NOM		

Exempel: Det ovanstående exemplet visar en ordklasstaggning, från *Stockholm Umeå Corpus 2.0* (gb13-042). Första taggen anger t.ex. *”pronomen, neutrum, singularis, bestämd form och oklart rörande subjekt/objekt”.*

Det är lätt att underskatta ordklasstaggningens problem: ett språk som svenska innehåller väldigt många ord som har många möjliga taggar där helt rätt tagg ska väljas för varje ord i sin aktuella kontext. Den andra komplikationen är att ordklasstaggaren sällan själv har ett lexikon med språkets alla ord utan måste ”gissa” rätt tagg för en stor del av löporden om fri text analyseras. Uppgiften ordklasstaggning har i många moderna system företagits med en statistisk modell, ofta en s.k. *dold Markov-modell (HMM)*. Exemplet ovan visar ordklasstaggad text, vilket är indata till själva parsningsprocessen (alternativt ses det som en del

av den). En grammatik enligt ovan har alltså dessa ordklasstaggar ”nederst i träden”. En tänkbar regel i en lite mer detaljerad CFG skulle alltså kunna vara `NP → PN-NEU-SIN-DEF-SUB/OBJ` för att matcha det första ordet i exemplet ovan.

5. Riktiga parserbyggen för riktiga naturliga språk

Den minimala exempelgrammatiken rörande Kalle och älgen ovan visar hur undervisningen om grammatikskrivning för parsning kan inledas. Om en riktig parser ska byggas, som ska kunna hantera fri okänd svensk text, på samma sätt – vilka skillnader finns då? En första skillnad är att mängden satser som ska kunna godkännas inte är en handfull. Ett naturligt språk som svenska består av en *obegränsad* mängd grammatiska satser. För att kunna beskriva det behövs *rekursiva* regler. Det är rekursion som möjliggör att skriva en grammatik, som ju måste bestå av ett ändligt antal regler, men som ska kunna beskriva obegränsat många olika grammatiska satser – ett naturligt språk. Det går att med olika exempel visa att visa att denna egenskap (obegränsad längd/rekursion i reglerna krävs) behöver finnas i språkets grammatikbeskrivning. Exempelvis kan man konstatera att det är svårt att dra en gräns för hur många ingående ord varje NP kan ha: *En bil, en fin bil, en ny fin bil, en ny fin blå bil, en ny fin stor blå bil, en ny fin stor bil som vi sett…*[5] Det är likaså svårt att slutgiltigt säga hur många gånger i rad NP och andra konstituenter får samordnas eller nästlas utan att resultatet blir ogrammatiskt i svenska. En kanske enklare förklaring: Ett naturligt språk som svenska kan också sägas innehålla obegränsat många olika satser eftersom det inte går att sätta någon gräns i antal ord för hur lång en hel textmening kan vara.

Om ett språk som svenska istället bara hade bestått av t.ex. en miljon möjliga textmeningar som hela tiden återkom så skulle uppgiften parsning vara mycket enklare.

Det som en konstruktör av en parsergrammatik kan göra för att försöka ”fånga in” denna obegränsade mängd är att helt enkelt skriva väldigt många regler för att täcka grammatiska fall som förekommer i språket och att använda rekursion i reglerna. Vad blir konsekvensen?

[5] Rekursion innebär att en och samma omskrivningsregel kan användas upprepade gånger. I en kontextfri grammatik kan rekursion i reglerna beskrivas som att en någon icke-terminal (t.ex. NP) enligt grammatiken kan finnas i en trädgren så att samma kategorislag har sig självt någonstans längre ned i samma gren (dvs. en nod kan enligt reglerna ge upphov till samma typ av nod) – det betyder ju att denna delgren kan upprepas obegränsat antal gånger och därmed ge obegränsade längder av strängar.

6. Den klassiska modellen: utmaningar och problem

Det som ovan kallades den klassiska modellen för parsning består alltså av: en språkdefinierande grammatik (oftast kontextfri eller kontextsensitiv) enligt Chomsky-hierarkin och en parsningsalgoritm[6] som när den får en sträng som indata *godkänner* (genom att visa trädanalys enligt grammatiken) eller *förkastar* strängen (den är då *ogrammatisk* eller *tillhör ej språket*, vilket konkret från parserns perspektiv innebär att den inte kan hitta eller bygga ett träd för den).

Det går nu att abstrahera över de svårigheter som en ”renlärig” parser av denna klassiska typ stöter på. Det går att skönja två kategorier av analysfel som kan sägas vara egentliga kärnproblem för parsning av fri naturlig text och som, när de bemötts med olika strategier, mer har styrt utvecklingen de senaste decennierna. Den första svårigheten är helt enkelt att varje korrekt textmening som påträffas ska ges analys av en parser som är menad för fri text. Detta kräver en enormt väl utarbetad grammatik med lexikon. Ett naturligt språk innefattar som nämnts en oändlig mängd möjliga korrekta satser, samtidigt ska de strängar som ej tillhör språket förkastas.

När den första svårigheten, att ge analys åt varje textmening, bemöts genom rekursion i regelstrukturen och genom att skriva ett mycket stort antal grammatikregler – leder det generellt till en explosion av möjliga analyser för textmeningar som parsas.

Enkelt uttryckt: Ju fler regler som skrivs och ju mer rekursion, desto fler olika syntaxträd ges i genomsnitt till en indatasträng.

Detta är alltså olyckligt eftersom det i de flesta fall antas finnas precis ett korrekt träd per textmening. Det är inte alls ovanligt att med denna parsningsstrategi ge hundratals möjliga träd för en enda textmening (detta gäller exempelvis för några aktuella parser-implementationer för svenska). För praktisk språkteknisk användning (t.ex. för översättning av en svensk textmening till engelska) duger inte en myriad av möjliga analyser utan målsättningen är istället den enda som korrekt pekar ut precis vad som är subjekt osv.[7]

Den andra mycket betydelsefulla svårigheten i sammanhanget kan låta väldigt lik den första, och den är alltså att endast ge en enda analys, den korrekta, per textmening. De två problemen är således att sträcka ut parserns kapacitet till att täcka alla möjliga (t.ex. svenska) textmeningar, men att göra det på ett sådant sätt att den rätta analysen för det aktuella analysobjektet också kan utpekas.

---

[6] Här kommer ingen fördjupning att göras inom parsningsalgoritmer för kontextfria grammatiker – även om ämnet är stort nog för att ge universitetskurser om. Det är dock ganska fascinerande att det finns så många olika parsningsalgoritmer som nästan alla ger precis samma resultat, nämligen de syntaxträd som en textmening kan ges enligt grammatiken. Parsningsalgoritmer som levererar matchande träd skiljer sig i många fall bara åt genom olika tidsåtgång. Om algoritmen även har som uppgift att försöka välja *rätt* träd i den aktuella situationen så är den något verkligt annorlunda.

[7] Det är värt att påpeka att det finns en tillämpningstyp som är ett lysande undantag. I *automatisk grammatikkontroll* behöver programmet enbart känna igen just textmeningar som ges syntaxträd över huvud taget enligt språket/grammatiken (och det spelar ingen roll hur många träd, eller vilket träd som är det rätta) och signalera för *dem som inte kan ges något syntaxträd alls* – detta är ju definitionen på att språket/grammatiken ej är korrekt enligt grammatiken. Grammatiken definierar ju det korrekta språket. Ironiskt nog fungerar grammatikkontroller praktiskt ändå på andra sätt (nämligen genom att leta efter vanliga feltyper), vilket bl.a. kan förklaras av att ingen parser har en så fulländad grammatik.

Problemet att ett stort gediget grammatikbygge ofta ger många alternativa syntaxanalyser och inte bara den rätta innebar att en mening godkändes som grammatisk och tillhörande språket, men det var oklart vilket träd (vilken analys) som var det egentligen rätta för textmeningen – det betyder t.ex. att det var oklart precis var subjektet var eller hur långt ett objekt sträckte sig, eftersom det fanns olika lösningar.

7. Programmeringsspråk kontra naturliga språk

Kanske verkar beskrivningen av användning av en ganska odetaljerad kontextfri grammatik ovan lite raljerande – det verkar kanske vara ett oklokt sätt att så ofta generera överflödiga träd. Men detta är ofta den praktiska konsekvensen av att arbeta storskaligt med en modell av detta slag. Det finns många grammatikformalismer (som inte tas upp här) där större vikt läggs vid ett senare delsteg då felaktiga träd undanröjs (eller vid att de ej ska kunna genereras alls). Detta gäller t.ex. implementationer med formalismen HPSG (i skrivande stund saknas detta veterligen för svenska). Mer om sätt att undanröja felaktiga träd nedan.

Om mänskligt språk i stället hade varit lite mer som programmeringsspråk skulle en ansats med kontextfri grammatik förmodligen vara utmärkt. Program kan vara godtyckligt långa och varje sats (*statement* i programsammanhang) kan i och för sig även de ha godtycklig längd, och syntaxreglerna som tolkar programspråket innehåller även de rekursion. Skillnaden är att sådana konstruerade språk från början skapats för att kunna parsas entydigt (ofta med en variant av kontextfri grammatik), och kompileras. Även om de alltså kan innehålla satser av godtycklig längd så tillåts aldrig mer än en tolkning (dvs. strukturell flertydighet förekommer aldrig som ”möjlighet/svårighet” på samma sätt som i naturliga språk).

En naturlig imperativ satssekvens på svenska kan ju vara syntaktisk flertydig:

```
Om villkor A är uppfyllt: om villkor B är uppfyllt, skriv ut. Annars blinka.
```

Flertydigheten i exemplet just här gäller den avslutande ”*annars*-delen”. – Ska det blinkas när villkor A inte uppfylls eller när A och inte B uppfylls? Inom programmering kallas detta – i grunden syntaktiska – problem *dangling else*. Programmeringsspråk tillåter helt enkelt inte möjlig flertydighet, varje programspråk har syntaxregler som programmeraren måste följa så att varje statement (vilket oftast är lika med en imperativ sats) som ovan bara får ha en tolkning i programmeringsspråket. Kanske har den idealiserade bild av språk, som en kompilatortekniker kan få, haft ett visst inflytande på utvecklingen av parsning även av naturligt språk.

8. Ansatser för att finna rätt träd i en klassisk modell

Det har naturligtvis inte saknats ansatser för att komma till rätta med problemet att en stor samling träd genereras för en textmening – fastän högst ett sådant i allmänhet eftertraktas. Det finns många parsningsmodeller som fungerar genom att först generera för många träd och därefter försöker nå fram till det egentliga trädet för en sats som parsas.

Dels har statistiska modeller använts för att välja det mest sannolika trädet. Det går exempelvis att förse själva produktionsreglerna i en kontextfri grammatik med olika probabilitetsvärden och på så sätt skapa en *probabilistisk kontextfri grammatik (PCFG)*. Detta blir alltså ett sätt att ranka producerade träd genom uppfattad eller antagen sannolikhet hos de olika träden för en viss textsträng och välja det träd som ger högst sannolikhetspoäng, t.ex. genom att delträdens respektive sannolikhetspoäng beräknas.

En annan metod går ut på att försöka utesluta så många omöjliga trädtolkningar som möjligt. En mycket känd – och framgångsrik – modell som med hjälp av grammatiska regler utesluter oönskade träd, tills det rätta idealiskt återstår, är *restriktionsgrammatiken (constraint grammar)* utvecklad vid Helsingfors universitet på 1990-talet för bl.a. svenska, av Karlsson m.fl. Denna serie av implementationer (där även vissa andra strategier som dependensparsning senare användes) tillhör de absolut mest framgångsrika parserimplementationerna för svensk text idag i fråga om korrekthet. Exempelvis grammatikkontrollen för svenska i Microsoft Word har sitt ursprung i detta arbete.

9. Nya metoder för samma(?) uppgift

Det har under hela resans gång funnits många olika parsningsmodeller i omlopp parallellt med den som ovan har kallats den klassiska modellen. Denna har haft ett visst akademiskt understöd, inte minst genom dess förankring inom den välkända generativa grammatikskolan. En samling liknande program som uppkom hade istället grammatikregler skrivna i reguljära språk (inom datalogin och sökuttryck är *regexp* ett välkänt begrepp, denna språkklass svarar även mot *finita*, dvs. *ändliga automater*). Den reguljära grammatikklassen är formellt sett otillräcklig för att användas till en korrekt fullständig språkbeskrivning av t.ex. svenska. Detta har dock inte hindrat att *finite-state parsers* länge använts för parsning, ofta benämnt *shallow parsning* och liknande. Exempelvis har regeluttryck i reguljär grammatik kombineras till s.k. kaskader. Abney (för engelska) och Kokkinakis & Johansson-Kokkinakis (för svenska) har byggt mycket stora grammatiker genom att inhämta programdata från svensk text och samla in sekvenser för NP och andra frasstrukturer. Det nedanstående är ett exempel på en enda regel som matchar NP av längden 11 ord (egentligen är det

flera regler eftersom parenteser anger optionalitet). Över 700 regler av detta slag används för att matcha svenska NP-strukturer utan efterställda attribut.[8]

```
np11 → ADJ ADJ* CC (ADJ | ADV-X| NUM) NOB CC (ADJ | ADV-X| NUM)* hd=NOB
```

Denna frastyp ”NP utan efterställt attribut” är exempel på något som kallas en syntaktisk *chunk*, en ofta använd segmenttyp i dessa sammanhang. En chunk motsvarar bara delvis fraskategorin (t.ex. NP) enligt generativ grammatik. Denna utveckling på 1990-talet och tidigare var del av en utveckling delvis bort från den klassiska modellen. Dessa implementationer var styrda av nytillkomna praktiska behov på ett annat sätt: att kunna samla in förekommande nominalfraser är t.ex. användbart inom informationsextraktion. Det är fråga om programmerade snabba implementationer som identifierar dessa *chunkar* i text snabbt. För sina syften har den kunnat vara användbar.

Implementationerna med reguljära uttryck som ovan och några andra viktiga modeller som utvecklats mycket under de senare åren har det gemensamt att de mycket mer fokuserar just på det som var den klassiska modellens svaghet – att peka ut precis det rätta trädet, eller åtminstone delar av uppgiften exakt, som precis var subjektet i en sats löper. Det är olika praktiska syften i bakgrunden som gjort att dessa olika modeller utvecklats.

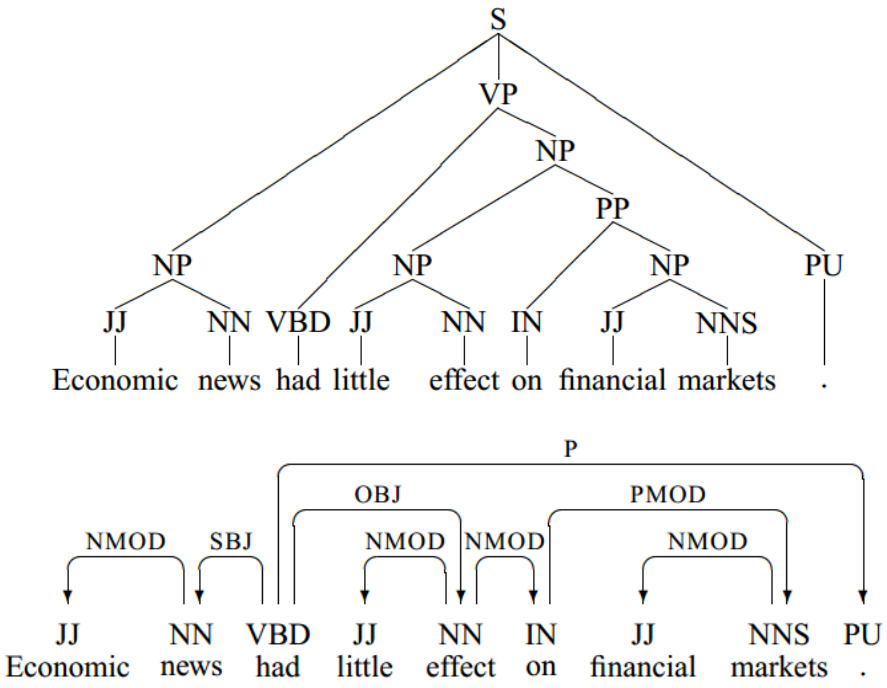
Det kan tyckas lite märkligt att uppgiften att leverera precis ett träd, det korrekta, skulle vara något nytt. Var det inte detta som var syftet (egentligen) också med parsning enligt den klassiska modellen? Ett annat sätt att påtala skillnaden hos den nya generationers metoder är dock att uppgiftens fokus flyttats till precis denna uppgift så starkt, och med sådana konsekvenser, att ett grundläggande kännetecken för uppgiften parsning inte längre uppfylls: De nya metoder som används för att göra rätt analys har nämligen på vägen förlorat förmågan att avgöra huruvida en textmening alls tillhör språket, dvs. att den är grammatisk svenska, eller annat språk. Det är alltså fråga om nya program som inte längre har syftet att bära med sig en fullständig definition av språket med vilken programmet kan känna igen korrekthet – implementationer för andra syften har tagit över stora delar av området.

Med dessa nya spelregler för detta område: identifiera grammatiska konstituenter (t.ex. funktionella kategorier) i indata (eventuellt på trädform) – men ignorera huruvida indata egentligen är språkligt korrekt eller felaktig – har några metoder framkommit som har kunnat leverera god korrekthet med avseende på uppgiften, som den nu har omformulerats. Att inte arbeta genom att som ett delsteg författa en grammatik över det aktuella språket och sedan nysta upp träd utefter den med en klassisk parsningsalgoritm – ger parsningsmetoder som ser helt annorlunda ut. Som påpekas ibland har den förlorade förmågan att skilja korrekta språkliga strängar från andra gjort att ett program som byggs på dessa grunder egentligen inte ens kan kallas för en parser, enligt äldre definition.

[8] Kategorierna svarar mot ordklasstaggar från *Parole*-korpusen, eller är variabler som står för flera taggar.

10. Andra implementationer av ”den nya skolan”

En viktig modern implementation för godtyckligt språk är den som används i den fritt tillgängliga *MaltParser*. Här används en statistisk metod för parsningen (och inte bara – vilket är vanligt – för ordklasstaggningen). Det betyder att metoden inte har inbegripit att skriva en svensk grammatik för hand. Parsningen består av en funktionell analys i dependensformat (se nedan). Det är en analysform som kan ses som fullständig och som kan översättas till kontextfri grammatik eller liknande. Dependensgrammatisk analys skiljer sig från generativ grammatik genom att inte förse en textmening med ett okänt antal kategorier ”i luften” ovanför löporden nederst i träden. En intressant matematisk skillnad mellan en parsning med en CFG och dependensformatet är därför att antalet bågar är ungefärligt känt från början. I litteraturen av bl.a. Nivre påpekas att intresset är finna den korrekta analysen – dvs. det är fråga om ett *optimeringsproblem*. Det går också att argumentera för att detta analysprogram och andra liknande ändå kan ses som parsning.



Figur: CFG-analys (överst) och motsvarande dependensgrammatisk analys (nederst).[9]

Den *induktiva dependensparsern* MaltParser använder sig av de trädbanker som gjorts manuellt (eller är manuellt kontrollerade) och som oftast är stora som SUC (ca 74.000 textemeningar). Metoden är en maskininlärningsansats och programmet är i princip tillämpbart för godtyckligt trädbanksförsett språk.

[9] Exemplet är hämtat från Joakim Nivre. 2005. Dependency grammar and dependency parsing. Technical Report. Växjö University, Sweden.

Att detta är en maskininlärningsalgoritm innebär att algoritmen till att börja med tränas på en trädbank. När den tränas samlar den statistik rörande hur analyser av allting statistiskt sett fördelat sig enligt de manuella analyserna i trädbanken. Vad algoritmen bygger upp är en sannolikhetsmodell för de olika bågar mellan orden i textmeningen (dependenser) och etiketter på bågarna – dessa är oftast, men inte uteslutande, funktionella kategorier som subjekt (vilket är en av de vanliga bågarna från satsens huvudverb, roten). Detta beror helt enkelt på hur analysen i trädbanken som den försökt lära av ser ut.

Avslutningsvis ska här nämnas en implementation under utveckling som också tillhör den nya kategorin implementationer och som enbart är specialiserad på svenska. Som redan nämnts tar denna rapport verkligen inte upp alla viktiga och ofta relativt lyckade implementationer som gjorts för svenska. Men många kan faktiskt i det aktuella fågelperspektivet beskrivas som tillhörande den beskrivna klassiska modellen, när det gäller just parsningen.

```
<subjekt>Proportionell konsolidering</subjekt>
<pfv>innebär</pfv>
<objekt>att endast de egna andelarna i bolaget
redovisas</objekt>
<tom>.</tom>
```

Exempel: Den huvudsatsanalys schemaparsning ger genereras bl.a. i ett XML-format. *Pfv: primärt finit verb* (hb09a-051 från SUC 2.0).

Schemaparsning är en metod som på ett ganska radikalt annorlunda sätt identifierar funktionella satsled (för närvarande enbart på huvudsatsnivå) i enlighet med Paul Diderichsens kärnfulla satsschema. Detta sker genom separata processer som först identifierar verb och andra begränsade led på huvudsatsnivå. Därefter vidtar en identifiering av de övriga leden (subjekt, objekt/predikativ och adverbial) som därmed redan delvis har avgränsats. Positioner och uteslutningsmetoder, tillsammans med andra heuristiska regler och en speciell segmenteringsmetod har möjliggjort hög korrekthet utan att programmet inbegriper någon fullständig, explicit beskrivning av flerordsled, t.ex. NP, alls. ”Om en student kan identifiera subjekt i en godtycklig sats utan att behöva känna igen varje möjlig NP-struktur i svenska språket bör väl subjektidentifikation också kunna implementeras utan explicit beskrivning av t.ex. NP?” Denna åsikt kan sägas vara en sorts bakgrund till tankegångarna i denna metod. Korrektheten för identifikation av de syntaktiska leden antas kunna bli väldigt hög med denna metod. Detta kan emellertid sägas om många modeller som i grunden bygger på manuell förbättring, och där det är tidsåtgång för arbetet snarare än metodologiska problem som sätter stopp.

När program för syntaxanalys fungerar enligt de nya metoderna så saknar de alltså möjligheten att känna igen en textmening som ogrammatisk eller inte

tillhörande språket. Men från någon sorts funktionsperspektiv kan detta också välkomnas eftersom det kan verka absurt i ett empiriskt perspektiv att på det sättet gå ut och ”godkänna” autentisk text. Om en sats i fri text inte kan matchas med en handskriven grammatik är det nog i många fall så att grammatiken är ofullständig – snarare än att den autentiska texten är felaktig.

11. Slutord

Denna text har visat hur de klassiska och kanske bäst lämpade modellerna för beskrivning av obegränsat stora naturliga språk inbegriper regeltyper av sådant slag att användning av grammatiken åt andra hållet (dvs. för *parsning* istället för *generering*) innebär ett komplicerat problem. Ju fler regler och ju mer rekursion som används för att täcka strängar som ingår i ett språk på detta sätt, desto större är risken för att indata förses med felaktiga analysträd utöver det korrekta. En lösning är att arbeta minst lika flitigt med uteslutningsregler som ser till att endast ett träd, det korrekta, lämnas som analys.

Den klassiska modellen har på senare år fått alltmer sällskap som helt och fullt är fokuserade på att finna den korrekta analysen, eller delar av den – t.ex. analys på huvudsatsnivå. Detta har inneburit helt nya metoder inom detta område – men det har samtidigt inneburit att dessa nya metoder och program har förlorat en egenskap som tidigare setts som karakteriserande för en syntaktisk parser – de har inte möjlighet att avgöra om en sträng är grammatisk (och tillhör det aktuella språket) eller är ogrammatisk (alternativt är skriven på ett annat språk).

12. Litteratur om ansatser för svensk text

Det följande ska ses som exemplifierande litteratur och publikationer gällande parsning av svenska från olika perspektiv. Det är inte ambitionen att här redogöra för alla implementationer och teoretiska arbeten inom området. En lite mer fullödig uppställning återfinns i kapitel 1 i Wilhelmsson (2010). De flesta ansatserna har vidare använts för språkteknisk forskning.

Parsningsprojekt som utgår från kontextfri grammatik (CFG) eller liknande tillsammans med en parsingsalgoritm

Sågvall Hein, Anna. ”Parsing by means of Uppsala Chart Processor (UCP).” i *Natural Language Parsing Systems*, av L Bolc. Berlin & Heidelberg: Springer Verlag, 1987.

Karlsson, Fred, Atro Voutilainen, Juha Heikkilä, och Arto Anttila. *Constraint Grammar - A Language-Independent System for Parsing Unrestricted Text.* Berlin och New York: Mouton de Gruyter, 1995.

Gambäck, Björn. *Processing Swedish Sentences: A Unification-Based Grammar and some Applications (doktorsavhandling).* Stockholm: KTH och Stockholms universitet, 1997.

Parsningsprojekt som utgår från CFG eller reguljär grammatik för att storskaligt göra delanalyser eller shallow parsing, till skillnad från ’fulla träd’

Källgren, Gunnel. *Making maximal use of surface criteria in large-scale parsing: the MorP parser.* Papers from the Institute of Linguistics, University of Stockholm, (PILUS) 60, Stockholm: Institutionen för lingvistik, Stockholms universitet, 1992.

Kokkinakis, Dimitrios, och Sofie Johansson Kokkinakis. *A Cascaded Finite-State Parser for Syntactic Analysis of Swedish.* Research Reports from the Department of Swedish, Göteborg: Institutionen för svenska språket, Göteborgs universitet, 1998.

Sofkova Hashemi, Sylvana. *Automatic Detection of Grammar Errors in Primary School Children's Texts (Doktorsavhandling).* Göteborgs universitet: Institutionen för lingvistik, 2003.

Knutsson, Ola, Johhny Bigert, och Viggo Kann. ”A Robust Shallow Parser for Swedish.” *Nodalida 2003.* Reykjavik, Island, 2003.

Megyesi, Beáta. *Data-driven Syntactic Analysis: Methods and Applications for Swedish.* Doktorsavhandling, Stockholm: Institutionen för Tal, Musik och Hörsel, KTH, 2002.

Parsningsprojekt inriktade på många språk inkl. svenska där dependensgrammatik är en viktig komponent

Voutilainen, Atro. ”Parsing Swedish.” *Proc. 13th Nordic Conference on Computational Linguistics (Nodalida-01).* Uppsala, 2001.

Nivre, Joakim. *Inductive Dependency Parsing.* Dordrecht: Springer, Text, speech, and language technology series, Volume 34, 2006.

Typteoretisk ansats som syftar främst till maskinöversättning av hög kvalitet och inbegriper chart parsing

Ranta, Aarne. *Type-Theoretical Grammar.* Oxford University Press, 1994.

Ansatser där Paul Diderichsens positionsgrammatik har tilldelats en viktigare roll än bara ett språkspecifikt komplement

Ahrenberg, Lars. ”A grammar combining phrase structure and field structure.” *Proceedings of the 13th conference on Computational linguistics - Volume 2.* Helsingfors: Association for Computational Linguistics, 1990. 1 - 6.

Wilhelmsson, Kenneth. Heuristisk analys med Diderichsens satsschema – Tillämpningar för svensk text (doktorsavhandling). Göteborgs universitet: Institutionen för filosofi, lingvistik och vetenskapsteori, 2010

Övriga referenser i denna text

Chomsky, Noam. "On certain formal properties of grammars*", Information and Control. 2 (2): 137–*167. 1959

Ljunglöf Peter, Wirén Mats. Syntactic parsing. I: *Handbook of Natural Language Processing.* 2 Boca Raton, Florida: Chapman & Hall/CRC. p. 59-91. Machine Learning & Pattern Recognition Series. 2010

Nivre Joakim. Dependency grammar and dependency parsing. Technical Report. Växjö University. 2005

GU-ISS, Forskningsrapporter från Institutionen för svenska språket, är en oregelbundet utkommande serie, som i enkel form möjliggör spridning av institutionens skriftliga produktion. Det främsta syftet med serien är att fungera som en kanal för preliminära texter som kan bearbetas vidare för en slutgiltig publicering. Varje enskild författare ansvarar för sitt bidrag.

GU-ISS, Research reports from the Department of Swedish, is an irregular report series intended as a rapid preliminary publication forum for research results which may later be published in fuller form elsewhere. The sole responsibility for the content and form of each text rests with its author.

Forskningsrapporter från institutionen för svenska språket, Göteborgs universitet
Research Reports from the Department of Swedish

ISSN 1401-5919

www.svenska.gu.se/publikationer/GU-ISS