Het geheim van de oude dame
De Nijmeegse parser Amazon

PETER-ARNO COPPEN*

Abstract

Originated in ancient times, the Amazon parser for Dutch appears to be a worthy competitor among contemporary rivalling parsers. In this article the causes of this success are discussed. From a concise history, Amazon's main characteristics are derived: it is a shallow parser, based on a structuralist descriptional theory. Moreover, Amazon's aims are modest: the Amazon parse is meant as only a first step in a total analysis. Subsequent components are needed to refine the Amazon parse tree.

Three main trends are discussed that characterize the Amazon development: modularization, separation of linguistic theory from algorithm, and the development of robustness strategies, which have led to the current Amazon parser. Next, Amazon's performance is briefly evaluated. In conclusion, it is argued that shallow parsing is a suitable first step in parsing natural language. Shallow parsing can even be motivated from linguistic theory.

1 Vooraf

Parsers hebben geen geschiedenis. Ze ontstaan in de geest van de tijd, worden –met een beetje geluk– toegepast en raken ingehaald door nieuwe ontwikkelingen. Er is voor een parser blijkbaar geen wezenlijke verdienste gelegen in een lange levensduur. Oud is ouderwets, nieuw is het magische woord.

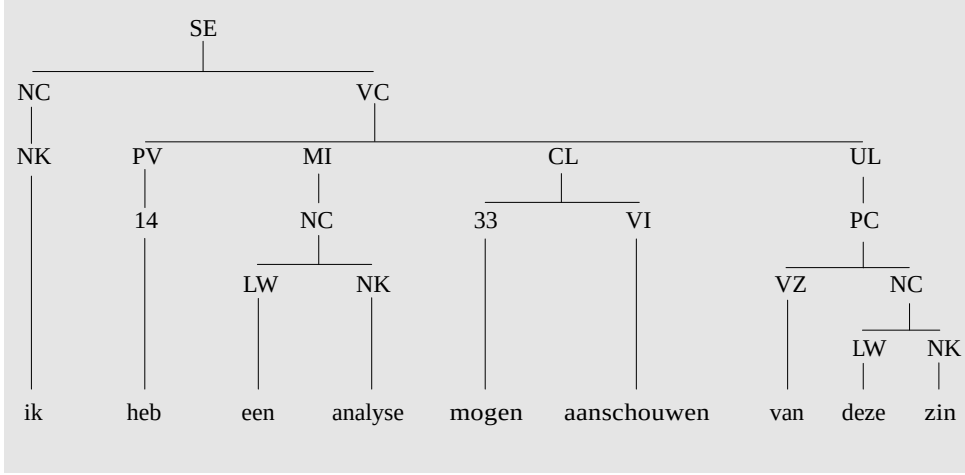
De Nijmeegse parser Amazon is in dit opzicht een buitenbeentje op het slagveld van de Nederlandse parsers. Haar geschiedenis gaat terug tot 1975, wat voor de technologie nog zo ongeveer het Stenen Tijdperk betekende. Ponskaarten, magneetbanden en computerprogramma's die 's avonds bij de operator moesten worden ingeleverd om 's nachts te worden uitgevoerd. Het verbeteren van een typfout kon zo een hele werkdag in beslag nemen.

Ondanks deze hoge ouderdom is de Amazonparser heden ten dage nog springlevend. Dat is te danken aan haar bescheiden doelstellingen, en een aantal verjongingskuren die

* Vakgroep Taal en Spraak, KU Nijmegen. Ik dank Carla Schelfhout, Jan Smeets, Bram Elffers en de reviewers van Nederlandse Taalkunde voor commentaar op een vorige versie van dit artikel.

zij in de loop der jaren heeft ondergaan. Toch is Amazon nog steeds Amazon, een oppervlakteparser[1] voor Nederlandse zinnen, op structuralistische grondslag.[2] Het lijkt zelfs meer dan ooit duidelijk dat de Amazonmethode, oppervlakteparsering, een aantrekkelijke strategie is als een eerste fase op weg naar een volledige analyse van de zin. Dat is de reden dat Amazon zich in een "Battle of the parsers" nog steeds durft te meten met haar jongere zusters.

De Amazonparser kan het best gekarakteriseerd worden aan de hand van haar historische ontwikkeling, in tandem met die van haar vervolgmodule, Casus geheten, die de verrijking van de Amazonparsering met thematische functies beoogt. In dit artikel zal ik deze geschiedenis daarom in het kort nalopen. En natuurlijk ga ik daarbij op zoek naar het Geheim van de Oude Dame: wat zijn de sterke punten van de huidige Amazonparser? Hoe zijn haar prestaties te verklaren tijdens de "Battle of the Parsers" op de winterschool 2001 van de Landelijke Onderzoeksschool Taalwetenschap (LOT)?



Figuur 1: Analyse Amazon 1975

2 De prehistorie

Eigenlijk begint de voorgeschiedenis van de Amazonparser al in 1928, toen de taalkundigen Rijpma en Schuringa hun Nederlandse Spraakkunst schreven. Deze op structuralistische leest geschoeide schoolgrammatica werd in de daaropvolgende jaren erg populair. Na 21 drukken nam de Nijmeegse taalkundige Jan van Bakel het in de tweede helft van de jaren zestig op zich om de 22e druk te moderniseren.

1 De internationale term is *shallow parser*. Het betreft natuurlijk geen oppervlakkige parser in de pejoratieve zin. De term verwijst naar de structuren die de parsering oplevert, en die als oppervlaktestructuren kunnen worden gekarakteriseerd. De hedendaagse term *shallow parser* bestond in 1975 uiteraard nog niet.

2 Amazon is gebaseerd op de structuralistische grammatica van Rijpma & Schuringa (1968). In de loop van de tijden zijn verschillende delen van de grammatica echter aangepast aan modernere inzichten. Niettemin is Amazon nog steeds een structuralistische grammatica te noemen.

Bij deze herwerking werd hij getroffen door de sterke formalisatie die het werk kenmerkte. Aangezien Van Bakel al eerder experimenten met de analyse van gedigitaliseerde teksten had uitgevoerd, vroeg hij zich af of de reikwijdte van de Rijpma & Schuringa-formalisatie ook meetbaar was met behulp van een computerprogramma. Hoe goed wás de Rijpma & Schuringa-grammatica eigenlijk? Bood zij een analysemodel voor alle Nederlandse zinnen?

Om deze onderzoeksvraag te toetsen, implementeerde Van Bakel het structuralistische formalisme van Rijpma & Schuringa in een computerprogramma dat hij Amazon noemde (een acroniem voor **A**uto**MA**tische **Z**ins**ON**tleding). Dit computerprogramma nam een Nederlandse zin (of een deel daaruit) als invoer, en gaf van deze zin een analyse volgens Rijpma & Schuringa.

De eerste Amazonparser was dus een computerprogramma. Het was *interactief*, dat wil zeggen het kon via een terminalverbinding met de mainframe computer direct worden uitgevoerd in een dialoog met de gebruiker. Daarnaast was het *syntax embedded*: de taalkundige theorie was niet in een apart formalisme gescheiden van het algoritmische deel van het programma, maar versleuteld in het programma zelf. Dat programma was weliswaar geschreven in een van de hogere programmeertalen uit die tijd (het in tekstverwerking gespecialiseerde SNOBOL4[3]) maar daarmee konden geen taalkundige regels worden weergegeven in de vorm waarin dat destijds gebruikelijk was (herschrijfregels en transformatieregels).

Hoewel Amazon dus geen expliciete formalisatie van taalkundige regels bevatte, lag zo'n formalisatie wel ten grondslag aan het parseeralgoritme. Het Amazonprogramma is achteraf te reconstrueren als een topdown links-rechts parsering,[4] met enkele handige optimalisaties waarvan de meeste uit nood geboren waren vanwege de beperkingen van de toenmalige hardware.[5] Deze parsering volgde een tamelijk rechttoe rechtaan herschrij-

3 SNOBOL4 staat voor StriNg Oriented and symBOlic Language. Dit anarchistische acroniem dreef de spot met het verschijnsel dat voor elke naam van een computerprogramma vaak achteraf een acroniem in elkaar werd geknutseld. De naam Amazon neemt deze vorm van satire over. Overigens draaide Amazon onder een dialect van SNOBOL4, het zogeheten SPITBOL (Speedy Implementation of snoBOL4), dat de acroniemensatire nog een stapje verder voerde. De programmeertaal SNOBOL4 werd door Van Bakel als meest geschikt voor parsering beoordeeld, omdat zij beschikte over uitgebreide faciliteiten voor patroonherkenning, het herkennen van patronen in teksten. Tegenwoordig geldt SNOBOL4 als een ouderwetse taal, maar de SNOBOL-patroonherkenning wordt alom geprezen en nagevolgd. De hulpprogramma's van het huidige Amazonsysteem zijn nog steeds in SPITBOL geschreven.

4 Een algoritme waarbij de analyse uitgaat van het beginsymbool van de grammatica, en de woorden in de zin van links naar rechts worden afgewerkt. De tegenhanger van het top down parseren is het bottom up parseren, waarbij de parser uitgaat van de woorden in de zin en van daaruit een analyse probeert op te bouwen. Tegenwoordig bestaan er verschillende mengvormen.

5 De analysemethode is gerelateerd aan de prestaties van de parser. Ruwweg gezegd: top down parseren kost tijd, en bottom up kost geheugenruimte. Aangezien dat laatste een meer praktische beperking was (de machines hadden toentertijd een beperkte geheugenruimte), was top down parsering vaak de voor de hand liggende keuze. De optimalisaties die Van Bakel in Amazon doorvoerde waren gericht op het beperken van het zogeheten 'backtrack-effect': doordat de parser bij verkeerde keuzes in principe op alle eerder genomen beslissingen moet kunnen terugkomen, ontstaat snel een explosie van mogelijkheden die moeten worden onderzocht. Van Bakel beperkte onder andere eenvoudigweg de diepte van de mogelijke parseerboom tot een interactief vast te stellen waarde.

ving, die de structuralistische velden en constituenten in de zin markeerde. Figuur 1 geeft een indruk van zo'n beschrijving.[6]

Deze voorbeeldanalyse laat duidelijk de mengeling zien van structuralistische velden en constituenten. De knopen MI en UL zijn structuralistische velden (resp. het middenveld en de uitloop), terwijl knopen als VC, NC en PC typische constituenten zijn. In latere versies van Amazon is deze basisstructuur regelmatig bijgesteld maar de oude indeling is nog steeds herkenbaar.

Tegenwoordig verdeelt Amazon elke Nederlandse zin in zeven structurele velden: het middenveld als kern, omgeven door twee werkwoordelijke polen: de werkwoordelijke eindgroep rechts van het middenveld, en de persoonsvorm of het voegwoord links. Vóór de persoonsvorm ligt het topicalisatieveld, nog voorafgegaan door het links-dislocatieveld (of: aanloop). Rechts van de werkwoordelijke groep ligt het extrapositieveld, nog gevolgd door het rechts-dislocatieveld (ook *uitloop* of *after thought* genoemd). In het volgende voorbeeld zijn alle velden gevuld:

(1) Zeg Jan, | tegen wie | heb | jij toch al die verhalen | verteld | over mij, | met die onsmakelijke details?

De buitenste twee velden zijn duidelijk door komma-intonatie gescheiden van de andere velden. De binnenste vijf velden vormen de vijf velden van de kernzin, die in de oudste versie van Amazon al aanwezig waren. In recente onderzoeksprojecten (Van Dreumel 1997, Gerrits 2001) worden binnen het middenveld nog nadere onderscheidingen gemaakt: zo wordt de clitische groep en de partikelgroep aan het begin apart gemarkeerd, en de afsluitende groep aan het eind met resultaatbepaling, richtingbepaling, idioom, predikaat of r-partikel.

Een opvallende eigenschap van de Amazonanalyse is het ontbreken van vrijwel elke semantische informatie. Natuurlijk is de syntactische structurering zelf in zekere zin de uitdrukking van een vorm van semantische informatie (namelijk de informatie welke woorden bij elkaar horen), maar het feit bijvoorbeeld dat *we* het onderwerp van de zin is en *een analyse van deze zin* het –discontinue– lijdend voorwerp, is in de boomstructuur niet terug te vinden.[7] Deze beperking hangt samen met de structuralistische uitgangspunten van Amazon. De thematische informatie werd geacht onder de semantiek van de zin te vallen, en niet onder de structuralistische syntaxis.

Een onderdeel van het programma dat een aparte algoritmiek had, was de routine die de werkwoordelijke groep analyseerde. Deze routine was gebouwd op de vormverwachtingen van het werkwoord. Kort gezegd komt dit hierop neer: op elk moment tijdens de parsering staat er een vormverwachting uit ten aanzien van het volgende werkwoord. Dat kan een persoonsvorm zijn, een infinitief met of zonder *te*, of een voltooid of tegenwoordig deelwoord. Bepaalde werkwoorden of constructies beïnvloeden die verwachting. Een hulpwerkwoord van tijd zet de verwachting voor een voltooid deelwoord aan, een

6 Deze en de volgende structuren zijn vereenvoudigd. Omwille van de helderheid van presentatie zijn de featurestructuren bij de knopen weggelaten.

7 Eigenlijk is dit *thematische* informatie. De vraag of thematische informatie semantisch is, laat ik hier buiten beschouwing. Dat werd in 1975 in elk geval wel zo gezien.

beknopte bijzin met *om* of *teneinde* verwacht een infinitief met *te*. Een werkwoord wordt alleen geaccepteerd als het beantwoordt aan de uitstaande verwachting. Als alle verwachtingen zijn ingelost, is het einde van de werkwoordelijke groep bereikt.

Een voorbeeld van een werkwoordelijke groep die aan dit basispatroon beantwoordt is:

(2) We *schijnen* het werkstuk zaterdag *te moeten hebben ingeleverd*.

De zin begint met een initiële verwachting *persoonsvorm*. Het werkwoord *schijnen* lost die verwachting in, maar creëert zelf de verwachting van een infinitief met *te*. Die wordt weer ingelost door *te moeten*, dat op zijn beurt de verwachting van een infinitief zonder *te* activeert. Het hulpwerkwoord van tijd *hebben* is mogelijk zo'n infinitief, maar die roept weer de verwachting van een voltooid deelwoord op. Dat voltooid deelwoord is *ingeleverd*. Dat is een zelfstandig werkwoord zonder eigen verwachting, zodat het de werkwoordelijke groep afsluit.

Uiteraard zijn veel werkwoordvormen ambigu (zo kan *hebben* ook een persoonsvorm zijn met een verwachting "infinitief met *te*", als in *ze hebben dat maar te doen*) maar Amazon zoekt naar de optimale combinatie van lexicale mogelijkheden.

Speciale werkwoordelijke constructies worden in Amazon beschreven als afwijkingen van dit basispatroon. Zo verantwoordt de oorspronkelijke Amazon al de afwijkende plaats van het voltooid deelwoord (*we schijnen het werkstuk zaterdag ingeleverd te moeten hebben*), het werkwoordpartikel (*we schijnen het werkstuk zaterdag in te moeten hebben geleverd*) en de IPP-constructie[8] (*we schijnen het werkstuk zaterdag te hebben moeten inleveren*). In latere versies is deze beschrijving nog nader verfijnd en uitgebreid (zie voor een uitgebreide bespreking Van Dreumel & Coppen (te versch.)).

In figuur 2 is het transcript weergegeven van een interactieve sessie met de allereerste versie van Amazon (nu nog beschikbaar als SNOBOL4-programma onder de naam Amazon75). Te zien is dat de gebruiker het parseerproces kan sturen door de diepte van de analyse aan te passen, en de keuze voor lexicale items te beïnvloeden. Dat lijkt op een zwaktebod van de parser, maar in het oog moet worden gehouden dat de doelstelling van het Amazon-programma slechts een "proof of principle" was: de onderzoeksvraag was of het beschrijvingssysteem Rijpma & Schuringa *in principe* geschikt was voor elke Nederlandse zin. De parseerefficiëntie diende alleen dit praktische doel, de computer was slechts een hulpinstrument bij de beantwoording van een vraag die in theorie ook handmatig kon worden beantwoord.

Ook de taggingcomponent[9] van Amazon is in dit transcript af te lezen: Amazon heeft voor de structureel dubbelzinnige woorden (*heb*, *mogen*, *deze*) de correcte keuzes gemaakt.[10]

8 In de Infinitivus Pro Participio-constructie (IPP) staat een infinitief op de plaats waar een voltooid deelwoord verwacht wordt. In het voorbeeld vraagt het hulpwerkwoord *hebben* eigenlijk om een voltooid deelwoord *gemoeten*.

9 Onder *tagging* verstaan we het toekennen van een woordklasse aan elementen in de taaluiting. Zie ook Oostdijk & Van Halteren (2002). Een computerprogramma dat tagging verricht heet een *tagger*.

10 Respectievelijk hulpwerkwoord van tijd (HVTP), hulpwerkwoord in de vorm van een infinitief met de verwachting infinitief (HVII), en attributief gebruikt demonstrativum (ILLE).

3 Uitbouw en afbraak

```
* * * * *     ZITTING   AMAZON       * * * * *
HET IS VANDAAG 02/18/02 14:20:47
ATTENTIE: HET DIEPTEBEREIK IS: 8
?
 DEBUG KLAAR OP VERZOEK.
dpt(5)
MAXIMALE DIEPTE GESTELD OP 5
ads(analyse)
–ANALYSE– TOEGEVOEGD AAN DE GRAMMATICA
ads(zin)
–ZIN– TOEGEVOEGD AAN DE GRAMMATICA
!
EINDE DEBUG–
HOE VERDER? – ANTWOORD “RETURN”, “START” OF “END”
start
ik heb een analyse mogen aanschouwen van deze zin.
1E ANALYSE; START DOOR MET ENTER OF “DEBUG”
** MET SUCCES ONTLEED IN 550 MSEC. **
TYP “JA” VOOR ANALYSE VAN HET RESULTAAT
NK  IK
NC  IK
14  HEB
LW  EEN
NK  ANALYSE
NC  EEN ANALYSE
MI  EEN ANALYSE
33  MOGEN
VI  AANSCHOUWEN
CL  MOGEN AANSCHOUWEN
VZ  VAN
LW  DEZE
NK  ZIN
NC  DEZE ZIN
PC  VAN DEZE ZIN
UL  VAN DEZE ZIN
VC  HEB EEN ANALYSE MOGEN AANSCHOUWEN VAN DEZE ZIN
SE  IK HEB EEN ANALYSE MOGEN AANSCHOUWEN VAN DEZE ZIN
?
 DEBUG KLAAR OP VERZOEK.
dmp(words)
WORDS<1> = IK
WORDS<2> = HVTP
WORDS<3> = EEN
WORDS<4> = ANALYSE
WORDS<5> = HVII
WORDS<6> = AANSCHOUWEN
WORDS<7> = VAN
WORDS<8> = ILLE
WORDS<9> = ZIN
!
EINDE DEBUG–
HOE VERDER? – ANTWOORD “RETURN”, “START” OF “END”
end
EINDE ZITTING AMAZON
HET AANTAL AANGEBODEN GROEPEN WAS: 1
```

Figuur 2: Een sessie in Amazon 75

Zo rond 1980 werd duidelijk dat de oorspronkelijke onderzoeksvraag van het Amazon-project positief kon worden beantwoord. In vijf jaar was Van Bakel geen zinnen tegengekomen die niet in principe door het Amazonprogramma konden worden geanalyseerd en die daar naar zijn oordeel wel voor in aanmerking zouden moeten komen.[11] Amazon gaf in sommige gevallen ook wel andere, en onjuiste analyses, maar steeds kon de parser het bewijs leveren dat het beschrijvingssysteem van Rijpma & Schuringa in de computerimplementatie van Amazon de aangeboden zin in elk geval ook correct afdekte. Daarmee was het Amazonproject geslaagd, maar was de parser plotseling beroofd van zijn onderzoeksvraag.

Wat te doen met een parser die geen duidelijk doel meer heeft? Weggooien was zonde, en de parser werd dus hergebruikt in een nieuwe onderzoeksvraag: is het mogelijk om Nederlandse zinnen te analyseren tot een dependentiestructuur geïnspireerd op de casustheorie van Fillmore (Fillmore 1968)? Dit is een wezenlijk andere vraag dan de Amazonvraag. Immers, Amazon streefde "slechts" een structuralistische analyse na, waarin constituenten in hun velden werden benoemd maar niet voorzien van thematische informatie. In een Amazonanalyse kon je wel zien dat een constituent een NC was aan het begin van Middenveld, maar niet of die NC het onderwerp van de zin was.

Waarom koos Van Bakel voor de casustheorie van Fillmore, en niet voor de traditionele ontleding in onderwerp of lijdend voorwerp? Dat lag hieraan, dat de onderzoeksvraag meer als een semantische vraag dan als een syntactische vraag werd gezien. Het doel was niet langer de toetsing van een syntactisch beschrijvingssysteem, maar eerder informatietechnologisch van karakter: kunnen de betekenisverhoudingen in de zin worden opgespoord in een automatische analyse? Toch bleef die onderzoeksvraag in de praktijk taaltheoretisch van aard: ook in 1980 waren werkelijke toepassingen nog ver weg.

Ter beantwoording van die nieuwe vraag werd een tweede module ontwikkeld,[12] nu simpelweg *Casus* geheten (geen acroniem dit keer). In die tijd schrok men ervoor terug om het Amazonprogramma nader te compliceren, om verschillende redenen:

- Het programma was naar toenmalige maatstaven al erg groot en complex geworden, hetgeen een substantiële uitbreiding in de weg stond. Niet alleen voor de programmeur, maar ook voor de toenmalige machines, zou het programma al snel teveel worden;
- De uitbreiding was wezenlijk anders van karakter dan de Amazonparsering: was de laatste een klassieke topdown parsering tot een constituentenstructuur van de oorspronkelijke woordvolgorde, de Casus-uitbreiding zou een transformatie moeten inhouden naar een totaal andere structuur –de dependentiestructuur– waarin de woorden in een andere volgorde zouden komen te staan dan in de oorspronkelijke zin.

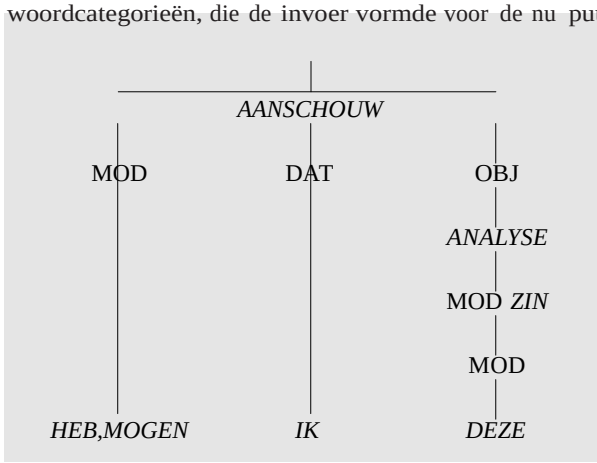
11 Er bestond in die tijd nog levendige discussie over grammaticale en ongrammaticale zinnen. De laatste werden niet geacht onder de definities van een spraakkunst te vallen. Later, in de jaren negentig, verschoof deze opvatting naar een grotere nadruk op de *robuustheid* van de parser, dat wil zeggen: ook afwijkende en zelfs ongrammaticale invoer moest geanalyseerd kunnen worden.

12 Ook weer een syntax embedded SNOBOL4-programma.

Samen met zijn doctoraalstudenten Computerlinguïstiek schreef Van Bakel in 1980 het programma Casus, dat de Amazonanalyse als invoer nam en op basis van een apart gedefinieerde set van volgorderegels voor Casusrollen[13] één of meer dependentiestructuren opleverde. In figuur 3 zien we een voorbeeld van zo'n Casusanalyse. De casusrollen hebben voor de hand liggende afkortingen.[14] De werkwoorden zijn de semantische kernen, en de casusrollen en eventuele modificeerders zijn hun dependenten. De hulpwerkwoorden en determiners worden geanalyseerd als kenmerken van het werkwoord of het zelfstandig naamwoord (in de figuur zijn deze kenmerken weggelaten).

In latere versies van Casus is het idee van de dependentiestructuur verlaten, en wordt de oppervlaktestructuur alleen verrijkt met thematische informatie. Figuur 4 geeft daarvan een indruk.

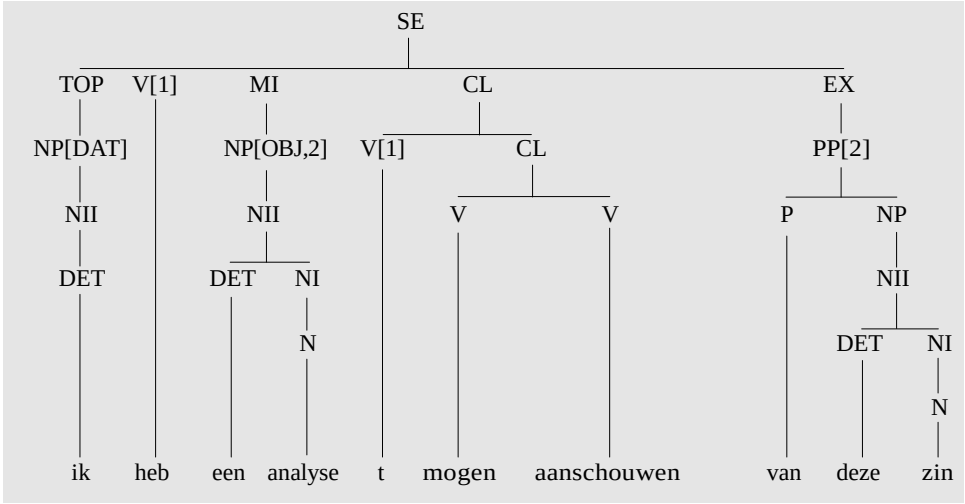
Tegelijk met de uitbouw van Amazon in de vorm van de Casusmodule vond er ook een vorm van afbraak plaats: om de complexiteit van het Amazonprogramma te reduceren werd de morfologische module uit het programma verwijderd en in een aparte module ervóór geschakeld (getiteld Amamorph). Tegenwoordig zouden we deze module als een losse *tagger* beschouwen: Amamorph analyseerde de elementen uit de zin tot een reeks van woordcategorieën, die de invoer vormde voor de nu puur syntactische Amazonparser.



Figuur 3: Een dependentie-analyse van Casus uit 1980

13 De term *thematische rol* of *thetarol* was in 1980 nog niet *en vogue* in de generatieve grammatica.

14 In het voorbeeld is DAT de datief of ondervindende persoon (experiencer) en OBJ is het object of de neutrale thematische rol. Een andere casusrol was AGE, de agens of handelende persoon.



Figuur 4: Een moderne Casus-analyse

4 Modularisering

Na 1980, toen het volledige Amazon-Casussysteem eenmaal operationeel was, vond een geleidelijk proces van modularisering plaats. De eerste betrof het Amazonprogramma. In 1983 schreef de studente Computerlinguïstiek Jenny Cals een doctoraalscriptie over de mogelijkheid om de Amazonparser te herformuleren in de vorm van een contextvrije herschrijfgrammatica die door een bestaande parser generator[15] automatisch kon worden omgezet in een parser.

De voordelen van deze operatie zijn evident: de linguïstische inhoud van de Amazonparser zit geheel in de contextvrije herschrijfgrammatica, en de parseeralgoritmiek wordt overgelaten aan de parser generator. Dat betekent dat taalkundigen en informatici min of meer onafhankelijk van elkaar kunnen werken aan de optimalisatie van hun eigen onderdelen.

De nieuwe Amazongrammatica beschreef overigens geen Nederlandse zinnen, maar structuren van woordcategorieën. Zoals gezegd was de morfologische analyse in handen van het Amamorph-programma, dat de tagging van de zin verzorgde. De lexicale ambiguïteit die Amamorph detecteerde, werd versleuteld in een zogeheten lexical lattice,[16] dat

15 Een parser generator is een soort compiler die een formele grammatica omzet in een uitvoerbaar programma. De parser generator in kwestie was getiteld GRAMMA, ontwikkeld door ir. Hans Meijer van de afdeling Informatica van de KUN. In zijn dissertatie (Meijer 1986) geeft Meijer een uitvoerige beschrijving hiervan.

16 Het Amamorph lexical lattice was als volgt gedefinieerd: de woorden uit de zin werden achter elkaar gerangschikt, elk woord geprefigeerd met al zijn mogelijke woordcategorieën (het woordje *dat* bijvoorbeeld werd geprefigeerd met de mogelijkheden *onderschikkend voegwoord* en *attributief* of *zelfstandig gebruikt demonstrativum*). Categorieën die meerdere woorden overspanden (*met behulp van*), werden genoteerd bij het eerste woord. Met een voorbeeld: in het geval van de voorzetseluitdrukking *met behulp van* werd bij het woord *met* de categorie *voorzetseluitdrukking* voorzien van een markering dat de volgende twee woorden daarbij inbegrepen waren.

de invoer vormde voor de Amazonparser. Deze construeerde op basis van de syntaxis het optimale pad door de lexicale mogelijkheden.

Het enige algoritme dat de transitie van computerprogramma naar formele grammatica niet overleefde, was het algoritme voor de werkwoordelijke eindgroep. Aangezien echter het aantal mogelijkheden in de praktijk eindig is (het Nederlands kent slechts een eindig aantal groepsvormende werkwoorden en ze mogen maar een eindig aantal keren in één werkwoordelijke groep voorkomen), werd voor dit onderdeel van de grammatica simpelweg een deelgrammatica voor een eindige taal ingelast. Pas in 1987 werd een manier gevonden om het oorspronkelijke algoritme in het toenmalige type formele grammatica terug te halen (cf. Coppen 1987 en Van Dreumel & Coppen (te versch.)).

Voordat in 1987 de deelgrammatica voor het werkwoordelijke cluster geheel werd herzien, was in 1985 de deelgrammatica voor de Noun Phrase al onder handen genomen. In Stoop (1985) wordt de NP-grammatica uit Coppen (1985) (later uitgebreider beschreven in Coppen 1991) geschikt gemaakt voor Amazon. Ook deze NP-grammatica is geconstrueerd aan de hand van een verwachtingsmodel: de gedachte is dat de NP-specifiers en premodifiers (zoals telwoorden, lidwoorden en adjectiva) de verdeling van naamval over de NP regelen. Een NP-initieel telwoord maakt een definiet lidwoord onmogelijk (**twee de aanwezigen*), tenzij dat exceptioneel gemarkeerd wordt met naamval (genitief *twee der aanwezigen*, of partitief *twee van de aanwezigen*). Een indefiniet lidwoord maakt een daaropvolgend telwoord onmogelijk (**'n ene aanwezige* of **zulke twee aanwezigen*), en verzwakt de verplichting om een verbogen adjectief te hebben (zie voor een uitgebreide uiteenzetting hiervan Coppen 1991). Deze NP-grammatica kon met succes in de bestaande grammatica worden ingebouwd.

De tweede belangrijke modularisering in het Amazon-Casussysteem vond plaats in 1989, toen ook het Casusprogramma herschreven werd tot een formele (transformationele) grammatica, die geïnterpreteerd werd door een separate algoritmiek. Op de taggingmodule na was nu het gehele systeem gesplitst in een zuiver taalkundig gedeelte en een informaticagedeelte.

Dat de taggingmodule in de loop der jaren enigszins onderbelicht is gebleven, werd nooit als een nadeel gezien. Immers, het ging hier niet zozeer om een toepassing van parsering, maar om prototypes. Daarnaast was de inbreng van de morfologische module zeer gering: de morfologische analyse bleef beperkt tot enkele productieve afleidingen (zoals werkwoordvervoeging en een aantal verbuigingen), die in feite eindig waren, en de tagger deed geen enkele poging om lexicale ambiguïteit op te lossen. Dat werd geheel aan de syntaxis overgelaten. Toen dan ook de parser generator in de jaren negentig de voorzieningen bood voor de opname van zeer grote lexica in de parser werd eenvoudigweg besloten om een groot lexicon van woordvormen op te nemen in plaats van een aparte morfologische module. Bij een efficiënte algoritmiek kost een lexicongrootte van 200000 in plaats van 100000 ingangen slechts één beslissingsstap extra.

5 Robuustheid

Na de omwerking van de deelgrammatica voor de werkwoordelijke groep in 1987 bleef het een zevental jaren stil op het Amazonfront. De parser fungeerde als een eerste stap in

de analyse van de Nederlandse zin, en het werk concentreerde zich op de Casusmodule. Zolang er sprake was van een onderzoeksinstrument was er ook weinig reden om aan Amazon te sleutelen. De parser produceerde wel eens een stuk of tien structurele analyses van een zin, maar de verwerking daarvan wierp geen technische problemen op en desnoods kon met de hand de gewenste analyse geselecteerd worden.

Intussen ontwikkelde ook de parser generator zich tot een volwassener instrument: het AGFL-systeem.[17] Er werden lexiconvoorzieningen toegevoegd en manieren om ambiguïteit te bestrijden. Zo konden regelalternatieven worden gemarkeerd als meer of minder waarschijnlijk, en sommige mogelijkheden konden als een soort "last resort" worden gemarkeerd: ze mochten alleen toegepast worden als andere mogelijkheden gefaald hadden.

In het doctoraalonderzoek van Erik Oltmans (Oltmans 1994) werden de mogelijkheden van deze nieuwe voorzieningen geëxploreerd. Oltmans onderscheidde allereerst de toevallige ambiguïteit van de structurele ambiguïteit. Toevallige ambiguïteit heeft een lexicale oorsprong. Bij substitutie van de woorden in de zin door niet-ambigue varianten verdwijnt ook de ambiguïteit. Met een voorbeeld: de zin *wij vieren feesten* is ambigu, maar bij vervanging van *vieren* door *tweeën* of vervanging van *feesten* door *feestjes* verdwijnt de ambiguïteit.

Van structurele ambiguïteit is sprake als een reeks van ondubbelzinnige lexicale categorieën meerdere syntactische analyses toelaat. Er zijn twee groepen van structurele ambiguïteiten:

- *Aanhechtingsconstructies*, waarbij het de vraag is op welk niveau van een constituent een daaropvolgende constituent moet worden aangehecht.
- *Transparante grensconstructies*, waarin onduidelijk is aan welke kant van de grens bepaalde constituenten moeten worden aangehecht.

Er zijn twee aanhechtingsconstructies: de aanhechting van mogelijke apposities (met name PP's), en de nevenschikkingconstructies. In een zin als *we hebben de verhalen van de buurman van je broer gehoord* is de veelvuldige ambiguïteit van de appositionele PP's manifest. Gaat het om de buurman van je broer, gaat het over de verhalen van de buurman, hebben we ze van de buurman gehoord of van je broer? Het is echter van belang om in te zien dat precies dezelfde syntactische ambiguïteit bestaat bij elke andere invulling van woorden met dezelfde categorie, ook als semantische of pragmatische factoren een van de mogelijkheden duidelijk bevoordelen. In een zin als *we hebben kinderen van de broer van je vrouw gezien* wordt de syntactische aanhechting van beide PP's sterk beïnvloed door de (semantische) neiging van de woorden *kinderen* en *broer* om een familierelatie te leggen met een *van*-PP, en de onwaarschijnlijkheid dat bij het werkwoord *zien* een bijwoordelijke *van*-PP is gerealiseerd. Aangezien Amazon geen toegang heeft tot informatie over subcategorisatie, laat staan tot diepere semantische of pragmatische informatie, is er binnen Amazon geen manier om uit alle mogelijkheden de juiste te kiezen.

Bij nevenschikkingsconstructies is de zaak nog problematischer, omdat er vaak ook

17 AGFL staat voor Affix Grammars over Finite Lattices. Deze term slaat op het taalkundige formalisme, dat een vorm is van de tweeniveaugrammatica: een contextvrije herschrijfgrammatica, waarin de symbolen kunnen worden voorzien van kenmerken (affixen, attributen, features) die door middel van unificatie met de kenmerken van andere symbolen in verband kunnen worden gebracht.

samentrekkingskwesties meespelen, en omdat de aanhechting niet alleen rechtsrecursief maar ook linksrecursief kan zijn. Het standaardvoorbeeld is de nevenschikking *mannen en vrouwen en kinderen*, die op drie manieren kan worden gestructureerd. Amazon heeft geen middelen om hieruit de juiste te kiezen (als er al een juiste is).

De transparante grensconstructies zijn wat hun betekenis betreft wat minder manifest, maar ze vormen zeker geen kleiner probleem. Er zijn vier problematische gevallen. De eerste betreft de grens tussen het middenveld en het werkwoordelijk cluster, geïllustreerd in de volgende zin:

(3) Zouden [MI ze die film ook ingekleurd] [CL hebben]?
(4) Zouden [MI ze die film ook] [CL ingekleurd hebben]?

Het voltooid deelwoord *ingekleurd* kan een bepaling van gesteldheid zijn (voorbeeld (3)) of het hoofdwerkwoord van de zin (voorbeeld (4)). Die ambiguïteit is in dit voorbeeld reëel, in die zin dat het correspondeert met een duidelijk betekenisverschil. Het structurele karakter maakt echter dat in elke opeenvolging van dezelfde lexicale categorieën dezelfde ambiguïteit optreedt. Dus ook in:

(5) Zouden ze die film wel bekeken hebben?

Dat betekent dat een voltooid deelwoord aan het begin van de werkwoordelijke groep in principe altijd dubbelzinnig is: het kan altijd ook als bepaling van gesteldheid gezien worden. Dat deze analyse ook de historische oorsprong van de voltooid deelwoordconstructie is, is een schrale troost voor de parser: de structurele ambiguïteit dient in de contemporaine analyse geen enkel doel.

Een tweede transparante grensconstructie is zo mogelijk nog zinlozer. Vergelijk de volgende voorbeelden:

(6) Jan heeft [MI tijdens de pauze aan de vakantie] [CL gedacht]
(7) Jan heeft [MI tijdens de pauze] [CL gedacht] [EX aan de vakantie]
(8) Jan denkt [MI tijdens de pauze aan de vakantie] [CL]
(9) Jan denkt [MI tijdens de pauze] [CL] [EX aan de vakantie]

PP's kunnen zowel aan het einde van het middenveld optreden als aan het begin van het extrapositieveld. Dat is duidelijk te zien in de zinnen (6) en (7). Maar diezelfde twee mogelijkheden bestaan er ook in de zinnen (8) en (9), waarin het werkwoordelijke cluster oningevuld is. In dat geval is er voor Amazon geen enkele manier om uit te maken welk van de twee analyses de juiste is.

Een derde transparante grensconstructie betreft de grens tussen hoofdzin en beknopte bijzin:

(10) Jan beloofde maandag [SE om het gras te maaien]
(11) Jan beloofde [SE om maandag het gras te maaien]
(12) Jan beloofde maandag [SE het gras te maaien]
(13) Jan beloofde [SE maandag het gras te maaien]

Ook hier kan een zinsdeel (*maandag*) links of rechts van een grens staan. Deze grens kan gemarkeerd zijn door het voegwoord (*om*), maar als het voegwoord achterwege blijft en het werkwoordelijke cluster in de hoofdzin is leeg, dan is er voor Amazon geen manier om de constituenten op deze grens in de goede zin te plaatsen, zelfs niet als de subcategorisatieframes van de werkwoorden één van de mogelijkheden zouden uitsluiten. De zinnen (12) en (13) vormen dan ook een structurele ambiguïteit.

Een laatste transparante grensconstructie betreft de volgende:

(14) Ze hebben [NP die twee oude sigaren] gegeven
(15) Ze hebben [NP die twee] [NP oude sigaren] gegeven
(16) Ze hebben [NP die] [NP twee oude sigaren] gegeven
(17) Ze hebben [NP die twee oude] [NP sigaren] gegeven

Hier hebben we niet zozeer te maken met constituenten die aan weerszijden van een grens kunnen staan, maar eerder met de onduidelijkheid van de grens zélf. Het Nederlands kent het kale meervoud: meervoudige substantieven kunnen zonder lidwoord of telwoord een NP vormen. Daarnaast kan bij specificatie of modificatie met demonstrativa en adjectiva de kern van de NP worden weggelaten. De combinatie van die twee eigenschappen levert een uiterst productieve structurele ambiguïteit op. De vier analyses voor dezelfde woordreeks in (14) tot en met (17) zijn technisch gezien nog niet eens de enige mogelijkheden, omdat de reeks *die twee oude sigaren* ook zou kunnen worden geanalyseerd als een opeenvolging van drie NP's:[18]

(18) [NP die] [NP twee oude] [NP sigaren]

Dit soort analyses mogen onwaarschijnlijk zijn, het is onduidelijk op welke syntactische gronden ze kunnen worden uitgesloten. Uiteraard zijn er een aantal factoren die deze ambiguïteiten in het dagelijkse taalgebruik onderdrukken (zoals de subcategorisatie van de omringende werkwoorden, de collocaties, of de menselijke parseerstrategie om zo groot mogelijke constituenten te maken: de *late closure* strategie), maar geen van deze factoren behoort tot het domein van de structurele syntaxis.

Ook al bieden robuustheidsvoorzieningen in het AGFL-systeem in principe de mogelijkheden om in geval van structurele ambiguïteit een keuze af te dwingen, aangezien de benodigde informatie voor het maken van zo'n keuze niet beschikbaar is, kan de correctheid van de keuze nooit gegarandeerd worden. Er zijn vier manieren om aan deze patstelling te ontsnappen:

- Geen keuze maken: alle mogelijkheden als analyse opleveren;
- De benodigde informatie voor het maken van de correcte keuze aan de grammatica toevoegen;
- Een ondergespecificeerde analyse afleveren, waarin meerdere keuzes in één structuur vervat zijn;
- Een willekeurige, dus soms verkeerde keuze maken.

18 Ook andere combinaties zijn mogelijk. Bovendien is onduidelijk op welke syntactische gronden een analyse in *vier* NP's zou moeten worden uitgesloten.

De eerste strategie ligt het meest voor de hand. In feite is dat voor een groot deel de manier waarop de Amazonparser tot 1994 te werk ging. Helaas geeft deze strategie in theorie aanleiding tot de zogeheten *combinatorische explosie*: het aantal analyses neemt exponentieel toe met de lengte van de zin. Snellere hardware of meer geheugen kunnen de drempel voor een werkbaar aantal analyses marginaal verleggen, principieel is deze oplossing ondeugdelijk.

De tweede strategie is ook al ondoenlijk, aangezien op den duur zeer verfijnde informatie nodig is voor desambiguering. Allerlei kennis van de wereld bepaalt in een concrete taalgebruikssituatie de voorkeursanalyse. Ook al zou het inbouwen van deze informatie in de grammatica in principe mogelijk zijn, dan nog is het onduidelijk waar we deze kennis vandaan moeten halen.

De derde strategie lijkt heel doenlijk: door het geven van een ondergespecificeerde analyse blijven alle mogelijkheden intact, en wordt de keuze uitgesteld tot latere modules die meer, of andere, informatie beschikbaar hebben.

De laatste mogelijkheid, een verkeerde keuze maken, lijkt absurd: ook al zou het mogelijk zijn om op basis van waarschijnlijkheid in veel gevallen de juiste analyse op te leveren, wat voor nut kan het hebben om in een substantieel aantal gevallen de verkeerde keuze te maken en zo verdere analyse onmogelijk te maken?

Toch is het maken van een willekeurige keuze niet zo absurd als het lijkt. Waar het om gaat is dat een mogelijk verkeerde keuze gemaakt is in een structurele context die door latere modules herkenbaar is. Met een voorbeeld: bij een opeenvolging van NP en PP kan de parser de PP aanhechten als rechterzuster dan wel meest rechtse dochter van de NP. Als een latere module, die zich bijvoorbeeld met subcategorisatie bezighoudt, beide constructies aanmerkt als mogelijk verkeerd, en een reparatiecomponent bevat die de andere mogelijkheid indien gewenst kan exploreren, is geen enkele mogelijkheid afgesneden. Welk van de mogelijkheden aanvankelijk wordt gekozen, is nu van secundair belang.

Waarom zou deze strategie beter zijn dan het ontwikkelen van een ondergespecificeerde analyse? Het antwoord is dat het eenvoudiger is. De ondergespecificeerde analyse is een extra voorziening die als enige doel heeft het markeren van de plaatsen waar later nadere specificatie moet plaatsvinden. Maar als we die plaatsen toch al kunnen aanwijzen (bijvoorbeeld als elke opeenvolging van NP en PP) is zo'n speciale voorziening dus onnodig. Waarom zou een "verkeerde" structuur dan slechter zijn dan geen structuur?

Dit is de gedachte die in het doctoraalonderzoek van Oltmans gevolgd is: het is bij structurele ambiguïteit niet nodig dat Amazon de juiste keuze maakt of een aparte representatie verzint, als de parser maar geen twee in principe gelijkwaardige analyses oplevert.

Deze robuustheidsstrategie is in latere versies van Amazon verder uitgewerkt. Voor alle structurele ambiguïteiten zijn praktische keuzes gemaakt, die wel ingegeven zijn door een logische gedachte, maar die net zo goed anders hadden kunnen uitvallen:[19]

- Bij aanhechting van PP's is gekozen voor een zo hoog mogelijke aanhechting. Alle PP's worden zo mogelijk los gegenereerd. Als de PP's appositioneel zijn, dan moet dat in latere modules maar blijken.

19 Die keuzes zijn dus wel gerelateerd aan een rudimentair begrip van waarschijnlijkheid, maar in wezen zijn ze willekeurig.

- Nevenschikkingen worden beschreven net als PP's: aparte constituenten met een voegwoord (of komma) als kern worden hoog aangehecht.
- Voltooide deelwoorden worden indien mogelijk onder de werkwoordgroep gerekend.
- Bij een lege werkwoordgroep worden zoveel mogelijk constituenten onder het middenveld gegenereerd.
- Bij beknopte bijzinnen worden lege middenvelden ontmoedigd.
- Bij opeenvolging van hoofdloze en kale meervoud-NP's wordt ernaar gestreefd zo groot mogelijke NP's te maken.

Geen van deze keuzes is principieel: voor PP's en nevenschikking geldt een *early closure*, *maximal attachment* strategie, waarbij maximale projecties zo snel mogelijk worden gesloten en zo hoog mogelijk worden aangehecht. Voor het middenveld en reeksen van NP's is gekozen voor een *late closure* strategie, waarbij de constituent of het veld zo lang mogelijk open blijft.

Met de verwijdering van structurele ambiguïteit daalde het aantal analyses dat de Amazonparser gemiddeld per zin opleverde natuurlijk dramatisch. Dat maakte het interessant om te bezien of Amazon ook inzetbaar zou zijn als technologisch instrument, voor de analyse van concreet taalgebruik in plaats van zorgvuldig geselecteerde modelzinnen.

Hiertoe werd de grammatica voorzien van een groot lexicon (ongeveer 325000 woordvormen) en een aparte *last resort* deelgrammatica, die bij ongrammaticale zinnen zo goed mogelijke deelanalyses moest opleveren. Het is deze organisatie die in de huidige versie van Amazon nog steeds in gebruik is.

6. De stand van zaken

In de 27 jaar ontwikkeling van Amazon zijn een drietal trends aan te wijzen:

- **Modularisering**: in de loop der jaren is Amazon steeds verder gemodulariseerd. Sommige onderdelen (zoals tagging en morfologische analyse) zijn geheel verwijderd uit de grammatica, andere (werkwoordgroep, NP) zijn alleen afgezonderd van de andere grammatica's.
- **Scheiding van Algoritme en Taalkunde**: de taalkundige beschrijving is neergelegd in een apart formalisme, dat niets met de algoritmiek van de parser te maken heeft.
- **Robuustheid**: door de jaren heen heeft Amazon zich steeds meer ontwikkeld van een puur taalkundig "proof of principle" prototype tot een meer technologisch instrument voor de analyse van concreet taalgebruik. Dat blijft voorlopig nog beperkt tot schriftelijk taalgebruik, maar er worden al enkele experimenten gedaan met de analyse van getranscribeerde spraak.

In de loop van haar geschiedenis is Amazon verschillende malen *from scratch* herschreven; daarbij is de terminologie enkele malen gemoderniseerd, en zijn "dichtgeslibde" deelgrammatica's opnieuw opgesteld. Belangrijke herschrijvingen hebben plaatsgevonden in 1983 (Jenny Cals), 1985 (Albert Stoop), 1987 (Peter-Arno Coppen), 1994 (Erik Oltmans) en 1997 (Simon van Dreumel). Voor 2002 staat een nieuwe versie op het programma.

1. Cathy zag hen wild zwaaien.
2. haar vader stak zijn duim omhoog alsof hij wilde zeggen: het komt wel goed, joch.
3. haar moeder kleefde bijna tegen het autoraampje aan.
4. haar neus werd platgedrukt en leek op een jonge champignon.
5. Cathy zag de BMW langzaam verdwijnen tot hij niet meer was dan een zilveren schijnsel tussen de bomen en struiken.
6. ze veegde de tranen uit haar ooghoeken, tilde haar twee koffers op en begaf zich in de richting van het landhuis.
7. de oprijlaan was niet meer dan een hobbelige zandstrook die zich voortslingerde tussen de hoge grijze boomstammen.
8. de middagzon hing klein tussen de takken en de schaduwen van de wolken drentelden over het gras.
9. het had een prachtige dag kunnen zijn in Londen.
10. ze had met haar moeder kunnen gaan winkelen, zwemmen of terrassen.
11. dat werkwoord had ze zelf uitgevonden.
12. het hoorde bij de warme zomerdag die ze ginds achter had gelaten.
13. ze hadden languit naast elkaar op de strandstoelen kunnen gaan liggen.
14 zij zou mams rug ingewreven hebben en mam de hare.
15. of ze had gewoon met haar vriendinnen rond kunnen slenteren in de buurt van Trafalgar-Square.
16. elk jaar in het hoogseizoen trokken daar massa's toeristen voorbij, hun fototoestel in de aanslag, pratend, gillend en lachend in de vreemdste talen.
17. het was een spel geworden: zij en haar vriendinnen kozen iemand uit en probeerden zijn of haar nationaliteit te raden.
18. het meisje dat vijf keer juist raadde werd getrakteerd op ijs.

Figuur 5: Het minicorpus van testzinnen

Op dit moment lopen er drie Amazonprojecten:

- Het AIO-project van Simon van Dreumel, waarin geprobeerd wordt om het einde van het middenveld meer structuur te geven. Amazon beschouwt het middenveld als een willekeurige reeks van maximale projecties en partikels, maar het is duidelijk dat de mogelijkheden beperkter zijn. Zo zal een werkwoordpartikel nooit vóór een maximale projectie staan (Van Dreumel 1997).
- Het AIO-project van Carla Schelfhout, dat een poging doet om intercalaties[20] in de Amazongrammatica op te nemen (Schelfhout 1999).
- De doctoraalprojecten van Jan Smeets en Bram Elffers, waarin een alternatieve versie van Amazon wordt geschreven die een beperkte vorm van subcategorisatie omvat (Smeets 2002, Elffers 2002).

Net afgerond is het doctoraalproject van Anouk Gerrits, waarin de clitische groep en de partikelreeksen aan het begin van het middenveld worden beschreven (Gerrits 2001).

20 Intercalaties zijn discontinuïteiten in de vorm van ingelaste tussenzinnen of andere constituenten, gekenmerkt door een breuk in de intonatie.

Het Amazonproject kent ook een website waar de laatste ontwikkelingen worden aangekondigd.[21]

7 De prestaties

Voor de LOT winterschool 2001 hebben de betrokken parseronderzoekers een minicorpus van taaluitingen gekozen uit het voorgelezen materiaal van het CGN-corpus. Het betreft dus in feite schriftelijk materiaal, waarin de interpunctie naar eigen inzicht is aangebracht. Het corpus omvat 18 zinnen met een zinslengte van 5 tot 23 woorden (zie figuur 5).

De onderzoekers mochten hun parsers op het corpus "tunen", hetgeen in elk geval zou inhouden dat het lexicon kon worden aangepast (geen van de lexica zou het werkwoord *terrassen* bevatten, dat door de hoofdpersoon van het verhaal was "uitgevonden").

Hoe moet je een parser evalueren, of meerdere parsers vergelijken? Voor de hand ligt om de uitvoer van de parser te vergelijken met een "gouden standaard": een handmatig uitgevoerde of gecontroleerde "correcte" analyse. Dat brengt echter, zo constateerden we tijdens de winterschool, een aantal problemen met zich mee. Vooreerst: wat is eigenlijk de "correcte" analyse? Een generatieve analyse in de Chomskyaanse minimalistische theorie, compleet met sporen en lege categorieën? Een klassieke zinsontleding, een predikaatlogische formule? Maar zelfs als we daaruit een keuze maken, bijvoorbeeld als minst controversiële de klassieke zinsontleding, dan levert de eerste de beste zin al problemen op: moeten we *hen* en *zwaaien* als twee lijdende voorwerpen analyseren (de gangbare opvatting), of toch maar *hen zwaaien* als beknopte lijdend voorwerpszin met *hen* als onderwerp? Of maken we van *zien* een nieuw soort hulpwerkwoord, waardoor we van *hen* het enige lijdend voorwerp kunnen maken?

Het is geen toeval dat de eerste zin al problemen oplevert. Ook over de tweede zin kan gediscussieerd worden: is *omhoog* een werkwoordpartikel (ik zou argumenteren van niet) of een bijwoordelijke bepaling van richting? Is *het* een gewoon onderwerp of loos (daar is discussie over mogelijk, maar ik zou denken het laatste)? Het zal duidelijk zijn: de klassieke zinsontleding mag dan de indruk wekken een relatief onproblematische analysemethode te zijn, in de praktijk levert bijna elke zin discussie op.

Daarbij komt dat vergelijking van de uitvoer van een parser met een klassieke zinsontleding in sommige gevallen bepaald oneerlijk zou zijn: de parser Delilah bijvoorbeeld levert een predikaatlogische formule die aanmerkelijk dieper is dan een gewone zinsontleding. Voor deze diepte wordt Delilah in een vergelijking gestraft. Er zou een extra inspanning nodig zijn om de resultaten van Delilah terug te rekenen naar een klassieke ontleding. Aan de andere kant van het spectrum geldt voor Amazon dat bepaalde onderscheidingen bewust achterwege zijn gelaten. Vergelijking met een analyse waarin die onderscheidingen wél zijn gemaakt, zou suggereren dat Amazon ze eigenlijk wel had moeten maken. Dat geeft dan wel aan hoever de Amazonanalyse verwijderd is van de gekozen standaard, maar niet van de *beoogde* analyse.

Tijdens de winterschool kwamen we er niet helemaal uit, en bleef de evaluatie van de

21 Om voor de hand liggende redenen kan het adres natuurlijk niet amazon.nl of amazon.com zijn. Daarom is het <http://lands.let.kun.nl/amazon>.

parsers beperkt tot het beoordelen van incidentele parseerresultaten. Toch is in het geval van Amazon een gerichtere evaluatie wel mogelijk, al dienen bij iedere evaluatie kanttekeningen gemaakt te worden.

Allereerst hebben we de zinnen van het testcorpus handmatig geanalyseerd volgens de Amazondescriptie. Dat wil zeggen: constituenten worden benoemd met Amazonlabels, alleen structurele velden worden onderscheiden, en de aanhechtingsproblematiek wordt handmatig opgelost door hoge aanhechting. Deze analyse wordt vergeleken met de Amazonresultaten. Wat we daarmee testen is hoe goed de parser presteert *volgens de bedoeling*. Zoals gebruikelijk meten we twee grootheden: de *precision* (het aantal correct benoemde constituenten gedeeld door het totale aantal benoemde constituenten) en de *recall* (het aantal correct benoemde constituenten gedeeld door het totale aantal constituenten dat benoemd had moeten worden). Beide grootheden worden op twee manieren gemeten: alleen op woordgroepsniveau (waarbij de benoemingslabels en losse woorden niet meetellen), en volledig (woord of woordgroep inclusief benoemingslabel). De gedachte hierachter is dat de eerste maat aangeeft hoe goed de woordgroepverdeling is, en de tweede hoe dicht de Amazonparser het doel nadert.

Soms geeft de Amazonparser meer dan één analyse (in één geval 6, in 3 gevallen 2). In die gevallen worden alle analyses meegerekend. Dit zou de recall gunstig kunnen beïnvloeden (immers de kans dat de goede benoemingen erbij zitten wordt groter), maar de precision moet kleiner worden (er zullen zeker verkeerde benoemingen bij zitten). Dubbele benoemingen tellen uiteraard maar één keer. In figuur 6 staan de resultaten van de Amazonparsering.

Zin	Precision (constituent)	Recall (constituent)	Precision (totaal)	Recall (totaal)
1	1.	1.	1.	1.
2	1.	1.	1.	1.
3	0.67	0.67	0.83	0.83
4	1.	1.	1.	1.
5	1.	1.	0.94	0.95
6	1.	1.	1.	1.
7	0.63	0.91	0.78	0.93
8	1.	1.	0.95	0.91
9	1.	1.	1.	1.
10	1.	1.	1.	1.
11	1.	1.	1.	1.
12	1.	1.	0.90	0.92
13	1.	1.	1.	1.
14	1.	1.	1.	1.
15	1.	1.	0.98	1.
16	0.56	0.82	0.69	0.88
17	1.	1.	1.	1.
18	1.	1.	1.	1.
gemiddeld	**0.87**	**0.91**	**0.89**	**0.91**

Figuur 6: Evaluatie van Amazon op minicorpus

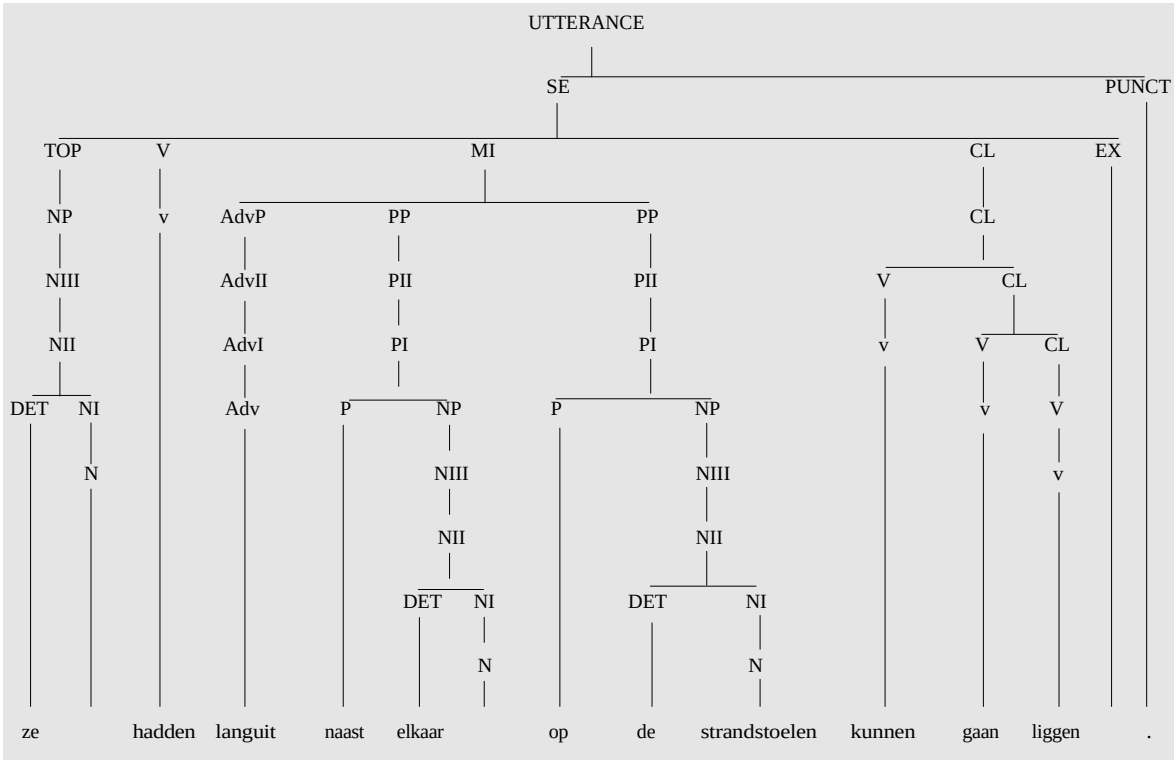
Duidelijk is dat Amazon moeite heeft met zin 16, die eindigt in een nevenschikking van bepalingen van gesteldheid (*hun fototoestel in de aanslag, pratend, gillend en lachend in de vreemdste talen*) waarvan het nog maar de vraag is hoe ze ingebed zitten. In de handmatige, "correcte" analyse staan ze als vier zinsdelen nevengeschikt in de uitloop van de zin, hoewel ook een nevenschikking van twee bepalingen van gesteldheid verdedigbaar zou zijn met een nevenschikking van drie tegenwoordig deelwoorden in het tweede conjunct. Amazon geeft hier twee mogelijkheden, waardoor met name de precision daalt. Bovendien heeft de parser moeite met het kiezen van de juiste aanhechting voor de nevenschikking. In feite wordt de totale juiste analyse niet gevonden. Dat de recall nog tamelijk hoog is ligt aan het feit dat de relatief onproblematische kleinere eenheden (losse NP's en minor constituents) allemaal goed gaan.

Men kan opmerken dat een dergelijke evaluatie geen goede indruk geeft van de prestatie van de Amazonparser in vergelijking tot een uiteindelijke gewenste parsering, waarbij alle constituenten op het juiste niveau aangehecht zijn en correct benoemd naar hun zinsdeelfunctie. Ook dat is eenvoudig handmatig te testen. Met name de handmatige correctie van de analyse naar aanhechting van nevenschikking en nabepalingen is eenvoudig. In figuur 7 staan de resultaten van deze berekening.

Zin	Precision (constituent)	Recall (constituent)	Precision (totaal)	Recall (totaal)
1	1.	1.	1.	1.
2	1.	0.9	0.95	0.87
3	0.67	0.67	0.83	0.83
4	1.	1.	1.	1.
5	0.86	1.	0.94	0.83
6	0.94	0.88	0.95	0.94
7	0.63	0.91	0.78	0.93
8	1.	1.	0.95	0.91
9	1.	1.	1.	1.
10	1.	1.	1.	1.
11	1.	1.	1.	1.
12	1.	1.	0.90	0.92
13	1.	1.	1.	1.
14	1.	1.	1.	1.
15	0.90	0.90	0.90	0.90
16	0.56	0.82	0.79	0.88
17	0.61	0.85	0.69	0.88
18	1.	1.	1.	1.
gemiddeld	**0.85**	**0.88**	**0.87**	**0.89**

Figuur 7: Evaluatie gecorrigeerd naar aanhechting

De daling van precision en recall blijkt niet zo dramatisch, maar dat lijkt een te rooskleurige voorstelling van zaken. In het minicorpus komen relatief weinig aanhechtingsproblemen voor.

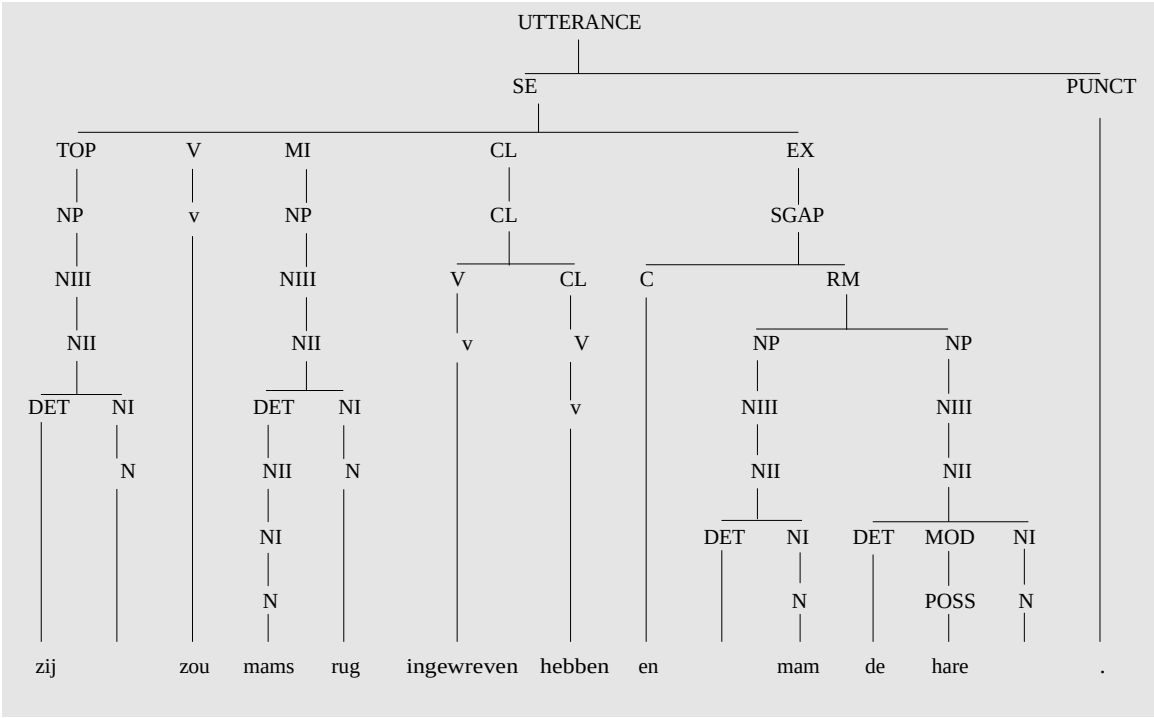


Figuur 8: Amazonanalyse van zin 13

Wanneer we ook nog de benoeming volgens de klassieke ontleding in de codering zouden opnemen, blijven de getallen voor precision en recall op constituentniveau uiteraard gelijk (ervan uitgaande dat de structuralistische Amazonanalyse alleen onderspecificeert voor functionele benoeming). Zowel totale precisie als totale recall zullen echter dalen, afhankelijk van wat we zouden toevoegen. Moeten we *haar* in zin (3) als bijvoeglijke bepaling bij *moeder* benoemen, of is dat in de structuur al uitgedrukt?

In plaats van deze verfijning toe te passen, met zijn vele onzekere parameters, geef ik in een tweetal voorbeelden een indruk van de Amazonparsering. In figuur 8 staat de analyse van zin 13, met een mooie werkwoordelijke eindgroep. De werkwoordgroep *hadden kunnen gaan liggen*, inclusief IPP-constructie, wordt door Amazon correct herkend. De PP's *naast elkaar* en *op de strandstoelen* worden naast elkaar onder het middenveld gegenereerd. Dat is niet gek, al is het nog maar de vraag wat in deze zin de beste analyse is. Moeten *languit* en *naast elkaar* niet als één bepaling van gesteldheid gezien worden?

In figuur 9 (zin 14) komt nevenschikking met samentrekking ("gapping") voor. Het tweede conjunct bestaat uitsluitend uit overblijfselen (remnants) van de samentrekking. Amazon geeft ook hier de correcte analyse (en overigens wordt het voltooid deelwoord *ingewreven* onder CL geplaatst en niet onder MI, zoals boven uitgelegd). De volledige analyses van het minicorpus zijn na te zien op de Amazon-website.



Figuur 9: Amazonanalyse van zin 14

Dit resultaat moet natuurlijk met enige scepsis bezien worden. Zoals gezegd is Amazon een parser met een bescheiden doelstelling. Door met opzet problematische keuzes buiten de analyse te houden, moet het resultaat uiteraard gunstig beïnvloed worden. En per slot van rekening is de parser "getuned" op de invoer. Niettemin werd tijdens de gezamenlijke sessie duidelijk dat Amazon wel degelijk een robuuste parser genoemd kan worden: Amazon was de enige van de deelnemende parsers die in staat was om ter plekke een onvoorbereid stuk tekst te analyseren met een acceptabel resultaat.

8 De toekomst van de Oude Dame

Wat is de toekomst van de Amazonparser? Wat heeft oppervlakteparsering voor nut in een wereld waar de computers steeds krachtiger worden? Heeft het niet meer zin om in te zetten op ambitieuzere parsers? Wat is de theoretische status van een oppervlakteanalyse?

In de loop der jaren is er binnen de Amazon-onderzoeksgroep herhaaldelijk discussie geweest over de reikwijdte van de Amazonanalyse. De verleiding om Amazon uit te breiden met allerlei diepere vormen van analyse is met de modernisering van hardware en software altijd groot geweest. Toch is er wel degelijk een theoretische basis voor de opper-

vlakteparsering zoals Amazon die levert. Een goed voorbeeld is het verschil tussen de volgende zinnen:

(19) Ik dacht dat Jan Karel Marie de hond de krant uit de bus heeft helpen leren laten halen.
(20) Ik dacht dat Jan de hond de krant uit de bus heeft helpen leren laten halen
(21) Ik dacht dat Jan Karel Marie de hond de krant uit de bus heeft helpen geleerd laten halen

Het is vrijwel onmogelijk om zonder pen en papier de grammaticaliteit van zin (19) te bevestigen. Hoewel grammaticaal, doet de constructie blijkbaar een te groot beroep op het werkgeheugen van de menselijke parser. Ook de ongrammaticaliteit in (20) is vrijwel ondetecteerbaar. In scherp contrast daarmee is zin (21) onmiddellijk herkenbaar als een foute zin. Blijkbaar is dit een ongrammaticaliteit die dichter bij de oppervlakte van de zin blijft. Een grammaticale theorie zou dit moeten verantwoorden.

De Amazonparsering detecteert zin (21) als een foute zin (dat wil zeggen: Amazon parseert deze zin met de robuustheidsgrammatica en beschouwt hem als een ellips), en produceert normale analyses van de andere twee zinnen. Daarmee is een reëel onderscheid in de parsering tot uitdrukking gebracht. De ongrammaticaliteit van zin (20) moet in de Amazonvisie door andere modules dan syntactische parsering gedetecteerd worden.

Deze bescheiden doelstelling van de Amazonparser is geen ontkenning van de noodzaak tot ambitieuzere parseerinstrumenten. Uiteraard is voor de meeste toepassingen een dieper inzicht noodzakelijk in de structuur en de betekenisverhoudingen van de taaluiting. Zeer waarschijnlijk moeten daar ook kwesties met betrekking tot de tekstuele en buitentekstuele context in meegenomen worden. Waar Amazon echter voor pleit is om oppervlakteparsering als een aparte module aan het begin van een langer analysetraject te handhaven, eventueel voorafgegaan door een module die de toekenning van woordcategorieën verzorgt. Oppervlakteparsering is een theoretisch heel goed verdedigbaar analyseniveau, en het is heel goed te doen met acceptabele resultaten. De beperkingen van oppervlakteparsering dienen niet te worden opgelost door de parser te compliceren, maar door separate verrijkings- of herstelmodules. Alleen op die manier blijft de organisatie van de analyse helder, en wordt het gevaar van de combinatorische explosie, met alle gevolgen voor de analysetijden van dien, efficiënt bestreden.

Bibliografie

Bakel, Jan van (1975). *Automatische zinsontleding met de computer*. Interne publicatie KU Nijmegen.

Cals, Jenny (1983). *Een contextvrije niet-linksrecursieve grammatica die dezelfde haakjesstructuur produceert als 'AMAZON'*. Doctoraalscriptie KU Nijmegen.

Coppen, P.A. (1985). De aard van het quantitatieve *er*. *De Nieuwe Taalgids* 78, 149-163.

Coppen, P.A. (1987). Het AMAZON-algoritme voor werkwoordelijke eindclusters. *Gramma* 11, 1-17.

Coppen, P.A. (1991). *Specifying the Noun Phrase*. Dissertatie KU Nijmegen.

Dreumel, S. van & P.A. Coppen (te versch.). Surface analysis of the Verbal Cluster in Dutch. *Linguistics*.

Dreumel, S. van (1997). A Robust parser for Dutch Sentences, abstract PhD project, <http://lands.let.kun.nl/~dreumel/PhD_project.nl.html>.

Elffers, A. (2001). Transducing Dutch utterances into Head/Modifier pairs. *Proceedings of the 2nd AGFL Conference,* <http://www.cs.kun.nl/agfl/workshop2/bramelf.pdf>.

Fillmore, C (1968). The Case for Case. In: E. Bach & R.J. Harms (eds) *Universals in Lin- guistic Theory*, New York: Holt, Rinehart & Winston.

Gerrits, A. (2001). *Het begin van het middenveld*. Doctoraalscriptie KU Nijmegen. **Meijer, H. (1986).** *Pro Grammar: A Translator Generator*. Dissertatie KU Nijmegen. **Oltmans, J.A. (1994).** *Amazon in AGFL: een contextvrije herschrijfgrammatica voor de structu-*

rele module van het AMAZON/CASUS-systeem, beschreven in het AGFL-formalisme. Doc- toraalscriptie KU Nijmegen.

Oostdijk, N. & H. van Halteren (2002). De grammaticale annotatie van tekstcorpora, *Nederlandse Taalkunde* 7, 175-181.

Rijpma, E. & F.G. Schuringa (1968). *Nederlandse spraakkunst*, bewerkt door Jan van Bakel. Groningen: Wolters-Noordhoff.

Schelfhout, C. (1999). *Intercalaties,* <http://lands.let.kun.nl/amazon/Algemeen/carla.htm>. **Smeets, J. (2002).** A subcategorisation model for Dutch and its implementation in AGFL, *Proceedings of the 2nd AGFL Conference*, <http://www.cs.kun.nl/agfl/workshop2/smeets.pdf>.

Stoop, A. (1985). *De implementatie van de NP-Coppen in Amazon en Casus.* Doctoraalscriptie KU Nijmegen.