STICHTING
MATHEMATISCH CENTRUM
2e BOERHAAVESTRAAT 49
AMSTERDAM

AFDELING ZUIVERE WISKUNDE

MR 71

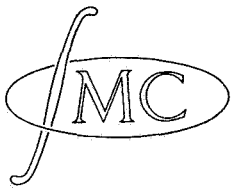
ZW 1964-014

DE COMPUTER IN DE TAALKUNDE

Voordracht in de serie "Actualiteiten"
gehouden op zaterdag 28 november 1964

door

H. Brandt Corstius



November 1964

Printed at the Mathematical Centre at Amsterdam,49,2nd Boerhaavestraat.
The Netherlands.

The Mathematical Centre, founded the 11th of February 1946, is a non - profit institution aiming at the promotion of pure mathematics and its applications, and is sponsored by the Netherlands Government through the Netherlands Organization for Pure Scientific Research (Z.W.O.) and the Central National Council for Applied Scientific Research in the Netherlands (T.N.O.), by the Municipality of Amsterdam and by several industries.

Inhoud pagina

0. Inleiding

De elektronische rekenmachine wordt voor vele taalkundige toepassingen gebruikt. Wanneer we ons beperken tot werk gedaan op het Mathematisch Centrum, dan kunnen we noemen:

het tellen van letters, bigrammen en trigrammen van letters, lettergrepen en woorden in kranten-Nederlands,

het maken van aanvaardbare merknamen voor een farmaceutische industrie,

het maken van een concordantie van een Middelnederlands boekje,

het bepalen van de betekenis van dubbelzinnige woorden uit hun context,

het ontleden van samengestelde woorden,

het splitsen van woorden in lettergrepen, en daarmee het automatisch zetten.

Hierover zullen we niet spreken. Bij het grammatikaal onderzoek (waarbij "grammatikaal" in de meest uitgebreide betekenis wordt gebruikt) kan de computer, en de computer-denkwijze, een belangrijke rol spelen. Het is op dit grammatikale aspekt van de toepassing van de computer in de taalkunde, dat we hier de nadruk willen leggen. Hierbij is het niet zo dat de computer alleen maar geeft, en de taalkunde alleen maar ontvangt. De ontwikkeling van de laatste jaren in de richting van steeds betere algemene programmeertalen maakt dat numeriek-wiskundigen, die zich vroeger bezig hielden met het oplossen van wiskundige vraagstukken, zich nu met taalkundige problemen bezig houden.
Anderzijds gebruiken de laatste jaren vele taalkundigen notaties uit de symbolische logika en de wiskunde. Er is een opvallende parallellie tussen het werk van de moderne, kort gezegd Chomskiaanse, taalkundigen en het werk van de ontwerpers en vertalers van, weer kortgezegd, talen als ALGOL 60.
Die parallellie begint al met
Backus (1959) : <term>::=< factor> or <term>×<factor> or <term>/factor>
en Chomsky (1955): $\text{Noun}_{\text{anim}} \rightarrow$ I, John, boy, etc.

en zet zich via push down vertaalmethoden tot syntax directed compilers voort. Het verschil tussen de computer-taalkundige en de computertaal-kundige is dat de eerste natuurlijke, de tweede kunstmatige talen bestudeert, maar beider arbeid is er op gericht om dat verschil te verkleinen. Het ideaal van de taalkundige is om zijn objecttaal zo beschreven te krijgen als b.v. ALGOL dat is, aan de andere kant doen de programmeertaal-ontwerpers hun best om "onnatuurlijke" elementen uit hun talen te verwijderen. Er blijft natuurlijk een belangrijk verschil bestaan: de computer-taalkundige heeft te maken met een hem gegeven structuur die hij moet zien te ordenen, terwijl de computertaal-kundige zelf de structuur kan ontwerpen waarmee hij het beste kan werken.

Er heeft zich een vak ontwikkeld dat onder vele namen bekend staat, maar misschien het beste "Algebraïsche Linguistiek" genoemd kan worden. We moeten daar, omdat het de formele basis van het grammatika-onderzoek is, een paragraaf aan wijden (§1). Daarna willen we met voorbeelden uit het geschreven Nederlands laten zien hoe de computer is te gebruiken bij de grammatika van woorden (§2), zinsdelen (§3), en zinnen (§4). Hierbij komen de aspekten van notatie, analyse, synthese en vertaling ter sprake. Vanzelf komt dan de vraag naar voren of de computer "alleen maar" symbolen kan permuteren of dat er iets van "de betekenis" kan worden gevat (§5). Tenslotte willen we nog eens benadrukken wat o.i. het belang van de computer in het taalkundig onderzoek kan zijn (§6).

1. Algebraïsche Linguistiek

Een taal L is een verzameling eindige rijen van elementen uit een eindig alfabet V. Een element van L heet een zin S.
In de interessante gevallen is L oneindig groot, maar een echt deel van de halfgroep W van alle eindige rijen van elementen van V.
Elementen van W geven we aan door Griekse letters.
Een grammatika G van L is een (rekursieve) procedure die de zinnen van L genereert. In het bijzonder beschouwen we grammatika's van het type $G = [V, \rightarrow, V_E, S]$ waarin V het alfabet is, V_E een deelverzameling uit V (de verzameling van eind-elementen), $\rightarrow$ een eindige deelverzameling van W x W (de elementen van $\rightarrow$ heten herschrijfregels. Als (ϕ,ψ) in $\rightarrow$ schrijven we $\phi \rightarrow \psi$), en S een element uit $V \setminus V_E$.
We definieren: $\phi \Rightarrow \psi$ als er een eindige rij $\phi_1 \ldots \phi_n$ is met:
$\phi = \phi_1$, $\psi = \phi_n$, als $i < n$ dan zijn er ψ_1, ψ_2, χ en ω met:
$\chi \rightarrow \omega$, $\phi_i = \psi_1 \chi \psi_2$ en $\phi_{i+1} = \psi_1 \omega \psi_2$.
ψ heet eind-rij als alle elementen uit V_E komen. De door G gegenereerde taal is dan: $L = \{ \psi \mid S \Rightarrow \psi, \psi \text{ eind-rij} \}$
Twee grammatika's heten zwak equivalent als ze dezelfde verzameling eindrijen genereren.
Meestal is men behalve in de gegenereerde zin, ook nog geinteresseerd in de wordingsgeschiedenis. Deze kan door een haakjesstructuur of door een boom gegeven worden. Een grammatika krijgt dan nog een andere betekenis: bij een gegeven zin een of meer bomen vinden. Twee grammatika's heten sterk equivalent als ze aan dezelfde zinnen dezelfde bomen geven.
Deze algemene grammatika, en de hierna te beschrijven beperkte grammatika's, vinden hun pendants in de theorie van de automaten. De automaat A is pendant van de grammatika G als hij dezelfde zinnen als G genereert, of als hij van een gegeven element van W kan uitmaken of hij tot de bij G horende taal L behoort.
De zojuist gedefinieerde grammatika's (algemene herschrijfsystemen) vinden hun pendant in de Turing machines, met alle nadelen aan die algemeenheid verbonden. We gaan nu steeds verdere beperkingen invoeren:

Een grammatika heet <u>context-gevoelig</u> als elke regel de vorm $\chi_1 A \chi_2 \rightarrow \chi_1 \omega \chi_2$ heeft, met $A \in V \setminus V_E$.
Onbeslisbaar is of twee gegeven context-gevoelige grammatika's equivalent zijn en of een gegeven rij in een herschrijfproces voorkomt. Voor het gemak van de beschrijving gaan we nog verder beperken, ook al weten we dat een natuurlijke taal in ieder geval context-gevoelige regels nodig heeft:
Een grammatika heet <u>context-vrij</u> als elke regel de vorm $A \rightarrow \psi$ heeft met $A \in V \setminus V_E$.
Bij zo'n grammatika is direct een grammatika te construeren die de haakjesstruktuur in de eindrijen aanbrengt: verander elke regel $A \rightarrow \psi$ in een regel $A \rightarrow [_A \psi]$.(Het context-vrije slaat natuurlijk op de regels, de elementen van de eind-rijen zijn wel sterk contextueel gebonden).
Een grammatika is <u>dubbelzinning</u> als eenzelfde rij op twee verschillende manieren kan worden voortgebracht. Het probleem of een gegeven context-vrije grammatika al dan niet ambigu is, is onoplosbaar. Er zijn context-vrije talen die niet door een niet-dubbelzinnige grammatika gegenereerd kunnen worden.
De context-vrije grammatika's vinden hun pendant in de push down automaten (alleen het laatst geschreven symbool kan gelezen worden).
Vele andere, al vroeger gebruikte, systemen zijn equivalent met de context-vrije grammatika's, b.v. de <u>normale</u> en de <u>categorische</u> grammatika. De normale grammatika, in de taalkunde geliefd, beperkt zich tot regels van het soort $A \rightarrow BC$ ($A, B, C \in V \setminus V_E$, dus binaire splitsing bij elke herschrijving) of $A \rightarrow a$ ($a \in V_E$, het lexicon). De categorische grammatika kent geen herschrijfregels, maar berekent volgens bepaalde reductieregels uit de bij de elementen van een rij behorende categorieën of de totaal-categorie S is.
Voorbeeld: "Klaas is zeer oud". Deze vier woorden hebben achtereenvolgens de categorieën: N, $(N \setminus S)/(N / N)$, $(N / N)/(N/N)$, (N/N) waaruit men afleidt: N, $(N \setminus S)/(N/N)$, N/N
of: N, $(N \setminus S)$
en tenslotte S.
Nog verder beperkend komen wij bij de <u>lineaire</u> grammatika, waar iedere regel de vorm $A \rightarrow x\ B\ y$ met $A,B \in V \setminus V_E$ en $x,y \in V_E$ heeft.

Pendant is de two-tape automaat.
De eenzijdige lineaire grammatika, b.v. de rechts-eenzijdige heeft regels van de vorm $A \rightarrow x\ B$ met $A, B \in V \setminus V_E$, $x \in V_E$. Het pendant hiervan wordt gevormd door de eindige automaat: komt in de grammatika de regel $A_i \rightarrow aA_j$ voor, dan heeft de pendant-automaat de regel (a, A_i, A_j), d.w.z. vindt het in staat A_i het symbool a dan komt het in staat A_j.
De interessante grammatika's bevinden zich op, of even boven het niveau van de context-vrije grammatika. ALGOL, zelf geen context-vrije taal, heeft grote delen die met een contextvrije grammatika beschreven kunnen worden. In de taalkunde werkt men de laatste tijd veel met transformatie-grammatika's waarvan de positie tussen contextsensitieve en algemene grammatika nog niet duidelijk is. Hiertoe behoort b.v. de grammatika gebruikt in §4.

Er zijn in het boven summier beschreven overzicht nog veel vragen en gaten. Het is aan de wiskundige die op te lossen, de taalkundige hoeft er zich niet te bezorgd over te maken. Wel dient hij zich bewust te zijn van de beperkingen in de vorm van de door hem gebruikte grammatikale regels.
In de hieronder te geven drie voorbeelden worden drie verschillende notaties gebruikt. In §2 wordt de grammatika van de Nederlandse getallennamen in een ALGOLprogramma uitgedrukt. De aftelling geschiedt hier in de natuurlijke volgorde, zodat men de genererende grammatika kan zien als een vertaling van cijferrepresentatie in woordrepresentatie. In §3 wordt de Nederlandse zelfstandignaamwoordsgroep gegenereerd door een context-vrije grammatika. In §4 wordt een bepaald soort zinnen beschreven met een transformatiegrammatika.

2. Getallennamen

Een context-vrije grammatika van de natuurlijke getallen in decimale schrijfwijze is:
(getal) → (cijfer), (getal)(cijfer), (getal)0
(cijfer) → 1,2,3,4,5,6,7,8,9
De grammatika van de Nederlandse namen van deze getallen is minder triviaal. Ieder Nederlander kan uit 42 basiselementen $10^{66}-1$ telwoorden maken, maar niet altijd zonder moeite.
Zo zijn er twee mogelijkheden om 1600 te zeggen,
"tweemiljoentwaalfhonderdduizendzeshonderdtien" is geen getal, en hoe spreekt U 900000000800000070000006 uit?
Het nu volgende ALGOLprogramma vertaalde op onze Electrologica X1 correct de in cijfervorm gegeven getallen tot 10^{66} ("undeciljoen") in Nederlandse woordvorm.

```
begin  comment   In cijfers aangeboden natuurlijke getallen worden
                 in Nederlandse woordvorm omgezet.

                 Of de cijfers per ponsband, ponskaart, schrijf-
                 machine of telefoonschijf worden ingevoerd, en of
                 de woorden per ponsband, schrijfmachine of spreek-
                 apparaat worden uitgevoerd is niet essentieel.
                 Daarom zijn de procedures voor invoer (Neem vol-
                 gende getal) en uitvoer (schrijf) los van het
                 taalkundig gedeelte van het programma gehouden.
                 Hun effect is in het bijbehorende commentaar be-
                 schreven, hun inhoud kan worden overgeslagen;

    integer i, aantal cijfers van getal;
    boolean overlappen; comment de getallen worden afwisselend over-
                 lappend (twaalfhonderd) en niet-overlappend (eenduizend-
                 tweehonderd) geschreven;
    integer array getal [1:66];
```

```
procedure Neem volgende getal; comment Deze invoerprocedure zorgt
                                dat het aangeboden getal in cij-
                                fervorm achteraan in het array
                                getal wordt gezet, en van voren
                                met nullen wordt aangevuld. Te-
                                vens wordt aantal cijfers van ge-
                                tal bepaald.
                                Heeft het aangeboden getal meer
                                dan 66 cijfers dan wordt dit mee-
                                gedeeld, en het te grote getal
                                overgeslagen. Is de lijst getal-
                                len afgewerkt dan stopt het pro-
                                gramma;
    begin integer cijfer;
          integer procedure volgend symbool; comment deze
                                leesprocedure werd bepaald door de
                                toevallig gebruikte codering;
                begin integer k;
                      own boolean lowercase;
                      k:= RE7BIT; if k=11 then stop; if
                      k=122 then lowercase:=true; if k=124
                      then lowercase := false; volgend
                      symbool := if k=0∨k=122∨k=124∨k=127
                      then volgend symbool else if ¬lower
                      case then 10 else if k=1∨k=2∨k=4∨k
                      = 7∨k=8 then k else if k=19∨k=21∨k
                      = 22∨k=25 then k-16 else if k=32 then
                      0 else 10
                end volgend symbool;
          i:= 1;
LEES GETAL:   cijfer:=volgend symbool; if cijfer=10 then goto
          GETAL GELEZEN;
          if i=67 then begin schrijf (‡GETAL VAN MEER DAN 66
          CIJFERS‡); UITLEZEN: if volgend symbool=10 then
```

```
                goto KLAAR else goto UITLEZEN end;
                getal [i]:=cijfer; i:= i+1; goto LEES GETAL;
GETAL GELEZEN:if i=1 then goto LEES GETAL; aantal cijfers van
                getal:= i-1;
                for i:= 66 step -1 until 1 do getal[i]:= if
                i>66-aantal cijfers van getal then getal[i-66+aan
                tal cijfers van getal] else 0;
        end Neem volgende getal;
    procedure schrijf(tekst); comment deze uitvoerprocedure zorgt er-
        voor dat de meegegeven tekst wordt geschreven;
        string tekst;
        PUTEXT1(tekst);

procedure volgende 3 cijfers (i);
    integer i;
    begin  integer i3;
        procedure kieswoord(j,k); comment een van de 37 woordjes
                wordt geschreven;
            integer j,k;
            begin  switch woordjes:= w1,w2,w3,w4,w5,w6,w7,
                                     w8,w9,w10,w11,w12,w13,w14,
                                     w15,w16,w17,w18,w19,w20,
                                     w21,w22,w23,w24,w25,w26,
                                     w27,w28,w29,w30,w31,w32,
                                     w33,w34,w35,w36,w37;
                procedure P(woord);
                    string woord;
                    begin schrijf(woord);goto
                    GESCHREVEN end;
                goto woordjes [9xj + k] ;
                w1 :P(‡een‡);       w2 :P(‡twee‡);
                w3 :P(‡drie‡);      w4 :P(‡vier‡);
                w5 :P(‡vijf‡);      w6 :P(‡zes‡);
                w7 :P(‡zeven‡);     w8 :P(‡acht‡);
```

```
                w9 :P(‡negen†);      w10:P(‡elf†);
                w11:P(‡twaalf†);     w12:P(‡dertien†);
                w13:P(‡veertien†);   w14:P(‡vijftien†);
                w15:P(‡zestien†);    w16:P(‡zeventien†);
                w17:P(‡achttien†);   w18:P(‡negentien†);
                w19:P(‡tien†);       w20:P(‡twintig†);
                w21:P(‡dertig†);     w22:P(‡veertig†);
                w23:P(‡vijftig†);    w24:P(‡zestig†);
                w25:P(‡zeventig†);   w26:P(‡tachtig†);
                w27:P(‡negentig†);   w28:P(‡milj†);
                w29:P(‡bilj†);       w30:P(‡trilj†);
                w31:P(‡kwadrilj†);   w32:P(‡kwintilj†);
                w33:P(‡sextil†);     w34:P(‡septilj†);
                w35:P(‡octilj†);     w36:P(‡nonilj†);
                w37:P(‡decilj†);
GESCHREVEN: end kieswoord;
procedure tussen 1 en 100 (j,k); comment schrijft k en
          j-tig;
        integer j,k
        if k≠0 then begin if j<2 then kieswoord (j,k)
              else begin kieswoord (0,k);schrijf(‡en†);
              kieswoord(2,j) end end else if j≠0 then
              kieswoord (2,j);
procedure zoveelhonderd (j,k); comment schrijft k en
          j-tig honderd;
        integer j,k;
        begin if j+k≠1 then tussen 1 en 100(j,k); if
        k≠0 then schrijf (‡honderd†) end zoveelhonderd;
procedure duizend tot de macht (k); comment schrijft
          duizend tot de macht k;
        integer k;
        if k=0 then goto KLAAR else if k=1
              then schrijf(‡duizend†)
              else begin kieswoord(3,k:2);if(k:2)×2=k
```

```
                                        then schrijf(⫞oen⫟)
                                        else schrijf(⫞ard⫟)
                                    end
                    i3:=3×i;
                    if overlappen ∧getal [66-i3] ×getal[67-i3]×i=0
                    ∧  getal[65-i3]+getal[64-i3]=0
                    then begin zoveelhonderd(getal[67-i3]); tussen 1 en
                            100 (getal[68-i3], getal[69-i3]); duizend tot
                            de macht (i-1); volgende 3 cijfers(i-2) end
                            overlappende geval
                    else begin zoveelhonderd(0,getal[64-i3]); tussen 1
                            en 100 (getal[65-i3], getal[66-i3]);
                            if getal[64-i3]+getal[65-i3]+getal[66-i3]≠0
                              ∨i=0 then duizend tot de macht (i);
                                volgende 3 cijfers(i-1) end
            end volgende 3 cijfers geschreven;
  schrijf(⫞resultaten van het programma dat in cijfers aangeboden getal-
            len in woordvorm omzet⫟);
            overlappen:=true;
START:      schrijf(⫞
⫟);
            Neem volgend getal;
            i:=(aantal cijfers van getal -1):3; volgende 3 cijfers(i);
KLAAR:      overlappen:= ¬overlappen; goto START
end
```

Het vertaalprogramma bestaat essentieel uit één statement:
de procedure-aanroep"volgende 3 cijfers(i)" die zichzelf met verlaagde i aanroept.
Het tegengestelde proces, van woorden naar cijfers, is eenvoudig te programmeren. Als we een dergelijk programma voor andere talen maken, kunnen we de telwoorden van de ene natuurlijke taal in de andere vertalen met de decimale representatie als tussentaal.

3. Zelfstandignaamwoordsgroep

Hoe men een Nederlandse grammatika ook gaat opbouwen, het lijkt waarschijnlijk dat daar op de een of andere manier een categorie "zelfstandignaamwoordsgroep" in zal voorkomen. De categorie b.v. van de woordenreeksen die verwisselbaar zijn met iets in de zin "ik denk aan iets". Deze categorie is even rijk als de gehele Nederlandse grammatika ("ik denk aan de man die S", waarbij S elke Nederlandse bijzin kan voorstellen). Omdat we juist een gedeelte van de Nederlandse grammatika willen ontwikkelen, beperken we ons tot die subcategorie die geen persoonsvorm bevat. Een verdere beperking: discontinue zelfstandignaamwoordsgroepen (als: "ik heb een tafel nodig met drie poten") worden niet gegenereerd.

De grammatika die hieronder ter kritiek wordt aangeboden, spreekt voor zichzelf. Men begint met Z, past de herschrijfregels toe totdat er alleen onderstreepte ("terminal") elementen zijn overgebleven. Het lege element wordt voorgesteld door 0. De 23 "woordsoorten" (d.w.z. de categorieën die direct in terminal elementen herschreven worden) zijn namen tussen haakjes. Hogere categorieën zijn aangegeven door enkele letters. Behalve de 23 woordsoorten is er nog een categorie van woorden als "ik" en "is" die, behalve tussen aanhalingstekens, niet in Z voorkomen, en een groepje woorden als "zo" en "om" die ieder een eigen woordsoort vormen.

De grammatica van Z

$Z \rightarrow$ z, Z (nevenschikker) Z, (zowel) Z (als) Z

$z \rightarrow c_{-4}\ c_{-3}\ C_{-2}\ C_{-1}\ C_0\ C_1\ C_2\ C_3\ c_4$

$C_i \rightarrow c_i$, C_i (nevenschikker) C_i, (zowel) C_i (als) C_i

$c_{-4} \rightarrow$ 0, (voorpartikel)

$c_{-3} \rightarrow$ 0, (aanwijzer), Z (z'n), (al) c_{-3}

$c_{-2} \rightarrow$ 0, (telwoord), (graadbijwoord) c_{-2}

$c_{-1} \rightarrow$ 0, a, i, d

$c_0 \rightarrow$ (substantief), b Z, " (woord) ", (infinitief)

c_1 → 0, (aanwijzersgenitief) Z, (telwoord), (aanwijzer) (telwoord)
c_2 → 0, (nabijwoord), v
c_3 → 0, i, om i
c_4 → 0, (napartikel)

a → (adjectief), m (hoeveelheidsadjectief), zo (adjectief) mogelijk, (bijwoord) a, v a, a a, (voorpartikel) a
i → te (infinitief), (bijwoord) i, Z i, v i
d → (deelwoord), (bijwoord) d, Z d, v d
v → (voorzetsel) Z, (voorzetselbijwoord) v, v v, (voorpartikel) v

b → c_{-4} c_{-3} C_{-2} C_{-1} (bevattingssubstantief) C_1 C_2 C_3 c_4
m → c_{-4} c_{-3} C_{-2} C_{-1} (maatsubstantief) C_1 C_2 C_3 c_4

(nevenschikker) → en, of,..
(zowel) → zowel, hetzij,..
(als) → als, hetzij,..
(voorpartikel) → zelfs, ook,..
(aanwijzer) → de, uw, Jans, elkaars,..
(z'n) → z'n, hun,..
(al) → al, heel,..
(telwoord) → drie, veel,..
(graadbijwoord) → vrij, ongeveer,..
(substantief) → man, paard, (bevattingssubstantief),..
(woord) → (zowel), (adjectief), (geen Z),..
(aanwijzersgenitief) → der, ener,..
(nabijwoord) → hier, daar,..
(napartikel) → alleen, zelf,..
(adjectief) → mooi, mooie, mooiere, (hoeveelheidsadjectief), ..
(hoeveelheidsadjectief) → lang, oud,..
(bijwoord) → zeer, nogal, (graadbijwoord),...
(infinitief) → geven, zien,..
(deelwoord) → gegeven, gevend,..
(voorzetsel → in, met, vol, als,..

(voorzetselbijwoord) → direct, naast,..
(bevattingssubstantief) → kist, minister,
(maatsubstantief) → meter, jaren,..

(geen Z) → is,.., geweest, ik, omdat,..
(woord op zich) → zo, mogelijk, te, om, "

Opmerkingen

1. De constructie "zowel Z_1 als Z_2" is in deze notatie niet op te schrijven. Zoals het er nu staat, komt er ook uit: "zowel Z_1 hetzij Z_2". Er moet dus een regel bij die voorschrijft uit (zowel) en (als) altijd corresponderende elementen te kiezen.
2. Niet alle mogelijke "nouns" worden gevormd. Voorbeeld: "een man gewend te bevelen".
3. Ernstiger is dat er veel niet-grammatikaals uitkomt. Er wordt nl. geen onderscheid gemaakt tussen het- en de-woorden, tussen enkel- en meervoud, tussen adjectieven zonder en met buigings-e. Toepassing van bovenstaande regels levert dus ook op: "de mooi paard". Dit is goed te maken, maar betekent veel schrijfwerk. We kunnen b.v. het "de-noun" en het "het-noun" afzonderlijk opbouwen. Ook kunnen we, net als in opmerking 1 een regel toevoegen die zegt: als je uit (substantief) een het-woord kiest, moet je uit (aanwijzer) het i.p.v. de kiezen.
4. Vervangt men de volledige zelfstandignaamwoordsgroep door de combinatie $c_{-3}\ C_0$ dan blijft de zin waarin hij voorkomt grammatikaal. Een normale krantentekst wordt hiermee tot de helft gereduceerd, terwijl de grammatikale bouw intact blijft.
5. Zoals uit opmerkingen 3 en 4 blijkt, werd deze grammatika niet in de eerste plaats opgezet als een generatieve grammatika (van de spreker) maar als een analyserende grammatika (van de hoorder). Terwijl voor dit doel fouten als in opmerking 2 genoemd ernstig zijn, worden fouten als in opmerking 3 onbelangrijk, want "de mooi paard" komt in de te analyseren tekst niet voor.

De meeste programma's voor automatische zinsontleding streven naar een volledige generatieve grammatika. Het is waar dat men dan ook een analyserende grammatika heeft. Maar men doet m.i. teveel werk: men beantwoordt nl. twee vragen:

1. Is de aangeboden zin grammatikaal?
2. Zo ja, wat is de ontleding?

Geen wonder dat, voor zover mij bekend, zo'n programma nog nergens werkt. Als we echter veronderstellen dat de ons aangeboden zin grammatikaal is, kunnen we met een grovere grammatika (die ook "de mooi paard" genereert) volstaan. De andere problemen bij de analyserende grammatika (homonieme constructies) zijn al groot genoeg. Gevallen waarin een grovere grammatika verkeerde zinsontledingen geeft, die met een "echte" generatieve grammatika voorkomen hadden kunnen worden, zijn zeldzaam.

6. Mijn doel is om de "nouns" in Nederlandse zinnen automatisch te isoleren. De overblijvende gecomprimeerde tekst biedt perspectieven voor een verdere zinsontleding. Het onderwerp van de zin is b.v. eenvoudig te vinden.

Voorbeeld

Oorspronkelijke tekst (N.R.C. 2 juli '56).

Hoewel de door de twee genoemde instanties gegeven waarderingen van hetzelfde feitencomplex een verschillend oogmerk dienden - het driemanschap adviseerde tav het bestuursbeleid, de Hoge Raad besliste omtrent het justitiële vervolgingsbeleid -, is het duidelijk dat zij een samenhang vertonen en dat zij elkaar aanvullen. Deze samenhang blijkt op zichzelf reeds hieruit, dat, indien de Hoge Raad een vervolging zou hebben gelast, dit consequenties voor de bestuurlijke positie van mr. Schokking had moeten hebben. Omgekeerd kan worden vastgesteld dat het, mede op het advies van het driemanschap gegronde, bestuursoordeel van de regering tav de Haagse burgemeester versterkt is door de, ook

voor het bestuursbeleid relevante, overwegingen van de Hoge Raad. In het bijzonder geldt deze versterking de beoordeling van de persoon van mr. Schokking.

Gecomprimeerde tekst Z → c_{-3} C_o

Hoewel de waarderingen een oogmerk dienden - het driemanschap adviseerde tav het bestuursbeleid, de Raad besliste omtrent het vervolgingsbeleid - is het duidelijk dat zij een samenhang vertonen en dat zij elkaar aanvullen. Deze samenhang blijkt op zichzelf reeds hieruit, dat, indien de Raad een vervolging zou hebben gelast, dit consequenties had moeten hebben. Omgekeerd kan worden vastgesteld dat het bestuursoordeel versterkt is door de overwegingen. In het bijzonder geldt deze versterking de beoordeling.

Hoewel Z Z dienden - Z adviseerde tav Z, Z besliste omtrent Z - is het duidelijk dat zij Z vertonen en dat zij elkaar aanvullen. Z blijkt op zichzelf reeds hieruit dat indien Z Z zou hebben gelast, Z had moeten hebben. Omgekeerd kan worden vastgesteld dat Z versterkt is door Z. In Z geldt Z Z.

Verkeerd was dus geïsoleerd: "dit consequenties".
Een onderscheiding tussen enkel- en meervoud zou dit corrigeren.

4. Hij zei dat hij dat zei

Beschouw de geschreven Nederlandse zinnen die gevormd kunnen worden uit de drie woorden "hij", "zei" en "dat" en de vijf leestekens aanhalingsteken, punt, vraagteken en uitroepteken.
De lengte van de zinnen en dus hun aantal is onbeperkt.
Voorbeeld: Zei hij dat hij "dat hij" "hij" zei hij "zei dat."" zei! "zei?

Hierbij is op te merken:

1. "dat" wordt in drie betekenissen gebruikt: als voegwoord, als oneigenlijk voegwoord in uitroepen, en als aanwijzend voornaamwoord.
2. "hij" wordt, al is dat in het geschreven Nederlands onzichtbaar in n betekenissen gebruikt. In de zin "hij zei dat hij zei dat hij "hij" "zei" komt driemaal het woord "hij" voor, waardoor de zin vijf mogelijke betekenissen krijgt. Dit speelt b.v. bij vertaling in het Latijn.
3. Door het gebruik van de "..." zei hij"..."-constructie hebben de aanhalingstekens geen gewone haakjesstructuur.
4. Elk woord kan gebruikt worden om zichzelf aan te duiden, het moet dan tussen aanhalingstekens staan.
5. De komma is niet toegelaten, en daarmee ook geen rij van woorden die zichzelf aanduiden: "hij zei "hij", "zei", "hij",..."

GRAMMATIKA

Duid de drie woorden "hij", "zei" en "dat" aan met H, Z en D. De leestekens duiden zichzelf aan. A is een van de drie echte woorden, B een van de acht woorden of leestekens. K is een eindige rij B's. W is het object dat kan voorkomen in "Hij zei...". S is een grammatikale zin uit onze taal L.
De grammatika begint met vier herschrijfregels die W genereren:

A → H,Z,D
B → A,",",.,:,!,?
K → B,BK
W → D,"A","W","S"

Daarna wordt L gegenereerd met twaalf transformatieregels:

$R_0(K)$ = "K" (operatie gebruikt bij de vorming van W)

$R_1(W)$ = HZW. $\in$ L (de naquootzin)

$R_2(W)$ = WZH. $\in$ L (de voorquootzin)

Als $S=K_1K_2$, K_1 bevat n A's en evenveel "als", K_2 begint met " of een A, R_3 werd niet eerder toegepast, dan:

$R_3(S,n)$ = "K_1"ZH"K_2". $\in$L (de inquitzin)

Als S=HZK. dan:

$R_4(S)$ = ZHK? $\in$ L (de vraagzin)

$R_5(W)$ = HZDHWZ. $\in$ L (de nabijzin)

$R_6(W)$ = WZHDHZ. $\in$ L (de voorbijzin)

Als S=HZDK dan:

$R_7(S)$ = HZDHZDK $\in$ L (de iteratieve nabijzin)

Als S=KDHZ. dan:

$R_8(S)$ = KDHZDHZ. $\in$ L (de iteratieve voorbijzin)

$R_9(W)$ = DHWZ! $\in$ L (de dat-uitroepzin)

Als S = DHKZ! dan:

$R_{10}(S)$= DHZDHKZ! $\in$ L (de iteratieve dat-uitroepzin)

Als S=K. dan:

$R_{11}(S)$= K ! $\in$ L (de uitroepzin)

ANALYSE

Een door bovenstaande grammatika gegenereerde zin kan door een computer geanalyseerd worden, met als output:
de structuur van de binnenste W, en de serie R_i-transformaties die achtereenvolgens werden toegepast.
De zin in ons voorbeeld zou geanalyseerd worden tot:

$R_4(R_5(R_0(R_9(R_0(R_3(R_1(D),$ [illegible]

Met zulke analyseprodukten kunnen we twee dingen doen:

VERTALING

Voor andere talen kan men soortgelijke R_4-regels opstellen. Zo wordt bijvoorbeeld:

$R_{4,\text{Frans}}$(K.) = est-ce que K?

$R_{4,\text{Engels}}$(he said K.) = did he say K?

$R_{3,\text{Latijn}}$(S,n) = "K_1" inquit "K_2".

Op grond van boven gegeven analyse zou dan onze voorbeeldzin als volgt in het Frans vertaald worden:

Est-ce qu'il disait qu'il disait:

" Qu'il disait: " "il" disait-il "disait cela." "! "?

Dit systeem kan uitgebreid worden door het toevoegen van woorden als "niet" en "wat". Elk toegevoegd woord maakt de grammatika gecompliceerder. We kunnen ook H,Z en D als woordklassen opvatten. De grammatika hoeft dan niet veranderd . De vertaalfunctie krijgt er een argument bij: welk woord uit de woordklasse is gekozen? Zo kan men "zij" in de klasse H plaatsen; de Russische vertaalfunctie verandert hierdoor. Voor H kan men eventueel de zelfstandignaamwoordsgroep uit §3 nemen, mits hierin geen aanhalingstekens of "dat" voorkomen.

Dit geval van automatische vertaling van zinnen met weinig verschillende woorden, maar alle grammatika, staat als uiterste tegenover het geval van de woord-voor-woord-vertaling, die alle woorden, maar geen grammatika heeft.

CONVERSATIE

Volgens Turing kan een machine pas intelligent genoemd worden als een mens er verstandig mee kan converseren. Tot nog toe blijft deze conversatie beperkt tot de vraag "hoeveel is...?" waarop de machine met een getal antwoordt. Wij kunnen nu een mechanische conversatie met zinnen van het boven beschreven type voeren.

Bij elke mededeling HZW. horen drie vragen:

1. de ja-nee vraag "ZHW?" (antwoord: ja of nee)
2. de wie-vraag "wie ZW?" (antwoord: H)
3. de wat-vraag "wat ZH?" (antwoord: W)

Wanneer de computer van de buitenwereld een mededelende zin ontvangt kan hij die na analyse bergen in een normaalvorm. Synonieme zinnen worden dus identiek. Ontvangt hij nu een vragende zin, dan kan hij die als volgt beantwoorden:

1. De ja-nee vraag "ZHK?" wordt getransformeerd tot de mededeling "HZK." en deze zin wordt gezocht onder de eerder meegedeelde. Komt hij daar voor, dan is het antwoord "ja", anders "nee".
2. De wie-vraag "wie ZW?" wordt getransformeerd tot de mededelende zin "XZW." Dan wordt er onder de eerder meegedeelde zinnen gezocht naar een exemplaar dat er op de X na mee correspondeert. Is er zo'n zin, dan is het antwoord dat wat in die zin op de plaats van X staat, anders is het antwoord: " ik weet het niet ".
3. De wat-vraag "wat ZH?" gaat over in "HZX." en wordt verder behandeld als de wie-vraag.

Uitbreiding van dit systeem is mogelijk met de passe-partout woorden "iemand", "doet" en "iets". Samengestelde vragen zijn combinaties van de drie genoemde typen. Voorbeeld:

Zei H1 hetzelfde als H2?

antwoord:=(antwoord(wat zei H1?) $\underline{?}$ antwoord(wat zei H2?))

Zei niemand W?

antwoord:= $\neg$antwoord(zei iemand W?)

Liegt H?

antwoord:= $\neg$antwoord(antwoord(wat zegt H ?))

Men kan dan de volgende conversatie houden

hij zegt iets.	o
wie zegt iets?	hij!
wat zegt hij?	iets!
zegt hij iets?	ja!
wat doet hij?	hij zegt iets!
zegt iemand iets?	ja!
zégt niemand iets?	nee!

wat zegt zij?	weet ik niet!
hij zegt dat zij iets zegt.	o
wat zegt hij?	iets, en dat zij iets zegt!
wat zegt zij?	volgens hem: iets!
zegt iemand dat iemand iets zegt?	ja!
dat zegt hij hij?	ik begrijp u niet!
wie zegt iets?	hij, en volgens hem: zij!
doet hij hetzelfde als zij?	volgens hem: ja!

5. Betekenis

Een veelgehoorde tegenwerping, wanneer men de toepassing van de computer op natuurlijke talen bepleit, is: "een computer kan tellen en alfabetiseren, maar de betekenis van de tekst blijft altijd ongrijpbaar."
Een blinde kan het woord "rood" juist leren gebruiken, d.w.z. hij kan het vertalen, hij kan alle emotionele en politieke associaties bij "rood" verwerven. Alleen kan hij niet, voor iets roods staande, zeggen "dit is rood". Een dergelijk begrip voor alle woorden is voor een computer ook mogelijk. Gaan we de drie gegeven voorbeelden na, dan blijkt dat in het eerste (de getallennamen) het begrip zelfs volledig is. Als men een in ALGOL geschreven ALGOLvertaler heeft, kan men door invoeging van het programma op pag. 6 een gehele berekening in woordvorm uitvoeren. Gezien het beperkte menselijke vermogen om grote getallen correct te benoemen, laat staan ermee te rekenen, kan men dus zeggen dat de machine hier een beter begrip heeft van de woorden dan de menselijke gebruiker.
In het tweede voorbeeld ("noun") komt de betekenis niet ter sprake. Dit is ook niet de bedoeling. We willen - op den duur - de betekenis van een zin uit de formele ontleding afleiden(met behulp van een uitgebreid woordenboek!). De traditionele zinsontleding draait dit proces om: men leest de zin, begrijpt hem, en vindt dan b.v. het onderwerp door het beantwoorden van speciale vragen, die pas geformuleerd kunnen worden als de zin werd begrepen.
In het derde voorbeeld ("hij zei dat hij dat zei") is van gedeeltelijk begrip sprake. Men kan deze zinnen in andere talen vertalen. Men kan vragen beantwoorden.
Het begrip "betekenis" kan alleen per situatie operationeel gedefinieerd worden. De betekenis van een dubbelzinnig woord is b.v. de keuze tussen twee vertaal-alternatieven (Deze keuze kan door een machine op grond van de context in een groot aantal gevallen correct gedaan worden.)
Een ingeklede vierkantsvergelijking is begrepen als men de antwoorden juist heeft gegeven.

Voor goed vertalen is begrijpend lezen noodzakelijk. Het is nog te vroeg om te zeggen of de computer, of liever de programmeur, hiertoe in staat is. Bar-Hillel meent van niet. De enige reden om te geloven dat de betekenis voor een computer ontoegankelijk is, ligt in het feit dat het nog niet vertoond is. Maar dat geldt evenzeer voor de "betekenisloze" grammatika, die nog voor geen enkele natuurlijke taal bestaat.

Er zijn grammatikale scholen die een zin als "de gracht heeft een huis getekend" niet grammatikaal vinden. De meeste voor een computer geschikte grammatika's slikken dit soort zinnen wel. Maar als men precies kan zeggen waarom deze zin niet aanvaardbaar is, dan kan de computer het ook leren. Grammatika *mét* betekenis is moeilijk te programmeren, maar niet principieel onmogelijk. Een bescheidener begin met betekenisloze grammatika lijkt aan te bevelen.

6. De computer in de taalkunde

Een computer is een apparaat dat snel en nauwkeurig volgens vooraf gegeven instructies informatie bewerkt.

snel: het taalmateriaal is groot. ALGOL kent ongeveer 2^6 basiselementen, Nederlands ongeveer 2^{17}. Het contextvrij op te schrijven gedeelte van ALGOL telt 80 herschrijfregels, Nederlands ?? Tot dusver bedreef men grammatikaal onderzoek voornamelijk op grond van materiaal dat zich toevallig in de fantasie van de onderzoeker voordeed. Nu kan men elke grammatikale hypothese onmiddellijk toetsen aan een grote verzameling materiaal. Het kwantitatieve verschil is zo groot dat het haast kwalitatief wordt.

nauwkeurig volgens vooraf gegeven instructies: dit dwingt tot precies begrip bij de onderzoeker. Wie b.v. de voorin de Nederlandse woordenlijst gegeven regels voor de lettergreepsplitsing leest, denkt al gauw dat hij het begrijpt. Maar als hij dan een computer wil leren lettergrepen te splitsen ziet hij de fouten en tekorten in deze regels. Een machineprogramma maakt ruzies over interpretaties onmogelijk. Met enige overdrijving kan gezegd worden dat het ideaal van Leibniz bereikt is:ALGOL is de taal van het wetenschappelijk geweten.

Het recursieve karakter van computertalen maakt deze bij uitstek geschikt bij het grammatikale onderzoek.

De computer neemt de taalkundige het werk niet uit handen, integendeel. Maar het nieuwe hulpmiddel kan hem bij verstandig gebruik verder brengen dan zijn traditionele werkwijze mogelijk maakte.

7. Literatuur

§0. De citaten van Backus en Chomsky zijn uit:

J.W. Backus, "The syntax and semantics of the proposed international algebraic language of the Zurich ACM-GAMM conference". ICIP Paris, June 1959.

N. Chomsky, "Logical Structure of Linguistic Theory". Microfilm M.I.T. Library, 1955.

§1. Over Algebraïsche Linguistiek:

Y. Bar-Hillel, "Four lectures on Algebraic Linguistics and Mechanical Translation". July 1962 Advanced Summer Institute "Automatic Translation of Languages" Venice, Italy.

N. Chomsky, "Formal Properties of Grammars", Handbook of Mathematical Psychology Vol. 2 ch. 12. New York-London 1963.

Over de theorie van de automaten:

R. McNaughton, "The theory of automata: a survey". in F.L. Alt, "Advances in computers" Vol. 2, New York 1961.

Over de categorische grammatika:

J. Lambek, "The mathematics of sentence structure" Amer. Math. Monthly 1958, 65, 154-170.

§2. A. van Katwijk, " A Grammar of Dutch Number Names"
H. Brandt Corstius, " Automatic Translation of Numbers into Dutch"

Beide artikelen zullen verschijnen in Foundations of Language 1, 1.

§3. H.F.A. Van der Lubbe, "Woordvolgorde in het Nederlands", Assen 1958.

Het N.R.C.-citaat is afkomstig uit de verzameling krantenteksten, die als grondslag dient voor een woordentelling waarvan de resultaten gepubliceerd worden als Mathematical Centre Tract Vol. 12.

§5. Y. Bar-Hillel, "The present state of Automatic Translation of Languages" in F.L. Alt, "Advances in Computers", Vol. 1, 1960.

§6. Een verzameling artikelen over computertoepassingen in de taalkunde vindt men in:

P.L. Garvin, "Natural Language and the Computer, New York 1963.

typ: EMN