RA

stichting
mathematisch
centrum

MC

REKENAFDELING MR 130/72 FEBRUARI

RA

G.H.A. KOK
HET AUTOMATISCH OMZETTEN VAN GESCHREVEN NEDERLANDS IN EEN FONETISCHE NOTATIE

2e boerhaavestraat 49 amsterdam

BIBLIOTHEEK MATHEMATISCH CENTRUM
AMSTERDAM

Printed at the Mathematical Centre, 49, 2e Boerhaavestraat 49, Amsterdam.

The Mathematical Centre, founded the 11-th of February 1946, is a non-profit institution aiming at the promotion of pure mathematics and its applications. It is sponsored by the Netherlands Government through the Netherlands Organization for the Advancement of Pure Research (Z.W.O.), by the Municipality of Amsterdam, by the University of Amsterdam, by the Free University at Amsterdam, and by industries.

Inhoud

1. Het probleem

De mens is in staat gesproken taal om te zetten in geschreven taal en vice versa. In hoeverre kan een machine dit ook leren? We willen hier afzien van de fysiologische kanten van dit probleem en van allerlei individuele verschillen in de uitspraak; daarom beperken wij ons tot de afbeelding van de gangbare Nederlandse schrijftaal (gespeld volgens het groene boekje [1]) naar een fonetische schrijfwijze en vice versa.

Van oorsprong is de spelling in alfabetische talen een fonetische notatie, maar in sommige talen heeft de spelling veel van zijn overeenkomst met de uitspraak verloren. Dit betekent dat in deze talen (voor)lezen geen éénduidige afbeelding van de symbolen uit de schrijftaal naar de fonemen, en schrijven geen éénduidige afbeelding in omgekeerde richting is; soms kan één bepaald symbool voor meerdere klanken gebruikt worden, en soms kan één klank op verschillende manieren door symbolen voorgesteld worden. We geven hiervan enige voorbeelden uit het Nederlands:

a. gevallen waarin een notatie uit de schrijftaal niet steeds hetzelfde foneem voorstelt: de *ch* in *chef* en *chemie*, de *a* in *baden* en *bad* (in het algemeen de *a*, *e*, *o*, *i* en *u* in open lettergrepen), de *e* in *de*, *del* en *deling*.

b. gevallen waarin een foneem niet altijd door hetzelfde symbool wordt voorgesteld: de *au* en *ou* in *rauw* en *rouw*, de *kk* en *k* in *bakken* en *baken* (in het algemeen de verdubbeling van de medeklinker achter een korte klinker), de *e*, *i* en *ij* in de tweede lettergreep van respektievelijk *lenen*, *lenig* en *lelijk*, de *ei* en *ij* in *eis* en *ijs*, en de *d*, *dt* en *t* in resp. (*ik*) *wed*, (*hij*) *wedt*, (*ik*) *wet*.

Indien we een algoritme willen construeren om schrijftaal om te zetten in een fonetische notatie dan moeten we moeilijkheden van het type a. oplossen, en voor de afbeelding in de omgekeerde richting de moeilijkheden van het type b. Voor de oplossing van de problemen uit b. moeten we in veel gevallen de betekenis of de grammaticale functie van het woord kennen, d.w.z. formeel uit de kontekst afleiden. Op grond hier-

van menen wij dat de automatische omzetting van een fonetische schrijfwijze naar de huidige schrijftaal niet praktisch haalbaar is. We hebben ons dan ook beperkt tot het construeren van een algoritme voor de afbeelding van de schrijftaal naar een fonetische notatie.

Voor een dergelijk computerprogramma zien wij de volgende toepassingen:

1: Als wij een rekenmachine die door dit programma bestuurd wordt, koppelen aan een apparaat dat in staat is bij ieder foneem de juiste klank te produceren, dan verkrijgen we een voorleesmachine. In samenwerking met het Instituut voor Perceptie Onderzoek (IPO) te Eindhoven, dat over een dergelijk apparaat voor spraaksynthese beschikt, wordt aan dit projekt gewerkt.

2: Met behulp van ons programma en een sorteerprogramma zouden we van een gewoon woordenboek een rijmwoordenboek kunnen maken.

3: We zouden het volgende onderzoek naar taalfouten van schoolkinderen kunnen doen: vertaal het fout gespelde woord en de korrekte vorm van dat zelfde woord naar de fonetische schrijfwijze en ga na in welk percentage van de foute woorden de fout in het fonetisch equivalent wegvalt. Dit percentage zou een argument in discussies over spellingwijzigingen kunnen zijn. (Dit soort onderzoeken zijn al verricht maar zonder op deze wijze de computer te gebruiken.)

4: Indien een andere (meer fonetische) spelling van kracht wordt zouden we teksten in de huidige spelling met een gewijzigde versie van ons programma in de nieuwe spelling kunnen schrijven.
Bijvoorbeeld de vervanging van de *t* in de lettergrepen *ti* en *tie(s)* door *ts* (*station*, *politie*), *s* (*potentiaal*, *pretentie*) of geen wijziging (*anti*, *politieke*) kan op deze wijze geschieden. Korrektie blijft natuurlijk noodzakelijk.

2. De fonetische notatie

Als hulpmiddel bij de definitie van de struktuur van onze fonetische notatie gebruiken we de z.g. Backus Normal Form [2]. Dit is een 'taal' die bestaat uit de volgende symbolen: een haakjespaar '< >', het symbool '::=', en het symbool '|'. Een gedefinieerde of te definiëren klasse schrijven we tussen '<' en '>', '::=' heeft de betekenis *wordt gedefinieerd als*, '|' heeft de betekenis *of*. De werking van de Backus Normal Form blijkt uit het volgende voorbeeld.

<cijfer>::= *0*|*1*|*2*|*3*|*4*|*5*|*6*|*7*|*8*|*9*
<niet negatief geheel getal>::= <cijfer>|<niet negatief geheel getal> <cijfer>

We zien dat *74* volgens deze definitie een niet negatief geheel getal is; want *7* is een cijfer en dus een nngg, en *4* is een cijfer dus *74* bestaat uit een nngg gevolgd door een cijfer en is dus een nngg. *7.4* is echter geen nngg want *7.* is geen nngg. De klassen noemen we ook wel meta-variabelen en de schuingedrukte symbolen rechts van het ::= teken heten terminale symbolen.

syntaxis

<korte klinker>::= a|o|u|e|i
<lange klinker>::= aa|oo|uu|ee|ie|oe|eu
<tweeklank> ::= ou|ei|ui
<glijklank> ::= ai|oi|aai|ooi|oei|eeu|ieu
<klinker> ::= <korte klinker>|<lange klinker>|<tweeklank>|<glijklank>

<stemhebbende medeklinker>::= v|z|g|b|d|q
<stemloze medeklinker> ::= f|s|c|p|t|k
<liquida> ::= l|m|n|r
<medeklinker> ::= <stemhebbende medeklinker>|<stemloze medeklinker>|<liquida>|j|w|h|n

<C1> ::= <onmiddelijk voor een klinker uitspreekbare combinatie van medeklinkers>|<empty>
<V> ::= <klinker>
<C2> ::= <onmiddelijk na een klinker uitspreekbare combinatie van medeklinkers>|<empty>
<lettergreep>::= <C1><V><C2>
<woord> ::= <lettergreep>|<woord><scheider><lettergreep>

In het bovenstaande ontbreken nog de definities van:

1: <onmiddelijk voor (na) een klinker uitspreekbare combinatie van medeklinkers>; deze beide klassen worden voor een belangrijk gedeelte opgesomd in [3].

2: <empty>; dit is de klasse die uit de lege symboolrij bestaat; hij kan gedefinieerd worden als <empty>::= .

3: <scheider>; als we de scheider tussen de lettergrepen willen aangeven zullen we hiervoor de spatie of het koppelteken gebruiken.

Deze syntaxis genereert natuurlijk veel meer woorden dan werkelijk in het Nederlands gebruikt worden b.v. lat-peuz-du. Onze keus van de terminale symbolen werd bepaald door het feit dat niet alle symbolen van het International Phonetic Alphabet (zie [4]) door het uitvoermechanisme van onze computer geproduceerd konden worden; daarom kozen we een notatie die voor de meeste klanken een symbool gebruikt dat als zodanig ook in de Nederlandse spelling voorkomt.

interpretatie

Een <korte klinker> wordt uitgesproken als de overeenkomstige gedekte klinker; de sjwa (stomme e) is dus u. q is de stemhebbende tegenhanger van de k (*bakbeest*) en c de stemloze tegenhanger van de g (*dacht*), n is de nasale *ng* klank (*rang*, *klank*).
De C1's en C2's worden uitgesproken als de overeenkomstige combinatie van medeklinkers; in de uitspraak van de C2 lk zal dus een zwakke sjwa hoorbaar zijn. Niet alle uitspraaknuances zijn in het programma opgenomen: zo veranderen de *ee* en *oo* onder invloed van een *r* van klank (ver-

gelijk *been*, *boon* met *beer*, *boor*), deze verfijning is niet geïmplementeerd omdat de informatie dat de klinker een beetje verandert toch al wordt gegeven door de onmiddelijk volgende *r*.

We geven ook een interpretatie van de door ons gebruikte symbolen in het internationale fonetische alfabet:

a = ɑ	aa = a	ou = ou	ai = ɑi
o = ɔ	oo = o	ei = ɛi	oi = ɔi
u = ə of œ	uu = y	ui = ʌy	aai = ai
e = ɛ	ee = e		ooi = oi
i = ɪ	ie = i		oei = ui
	oe = u		eeu = ew
	eu = ø		ieu = iw

v = v	f = f	l = l	j = j
z = z	s = s	m = m	w = w
g = ɣ	c = x	n = n	h = h
b = b	p = p	r = r	n̲ = ŋ
d = d	t = t		
q = g	k = k		

3. Beschrijving van het programma

Het programma is geschreven in de programmeertaal ALGOL 60 (zie voor de formele definitie [5] en voor een leerboek b.v. [6]). Deze taal is in de eerste plaats ontwikkeld om rekenprocessen te beschrijven, maar met een beetje handigheid kunnen er ook taalkundige processen in beschreven worden. In het programma kunnen we twee gedeelten onderscheiden: 1. de declaraties (pag. 10 t/m 24); 2. het eigenlijke programma (pag. 24 t/m 40).

Procedures zijn stukjes programma waaraan een naam verbonden is: het aanroepen van die naam in het eigenlijke programma is voldoende om de procedure uit te voeren, terwijl het voorkomen van de tekst van een procedure in de declaraties geen aanleiding geeft tot het uitvoeren van die procedure maar alleen dient om aan te geven welke procedure er bij een naam (identifier) hoort.

Ook variabelen hebben een identifier. Een belangrijk aantal variabelen van ons programma zijn feitelijk constanten (ALGOL 60 kent als constanten alleen maar dingen als 734), dit zijn de letters en de lettercombinaties. We hebben deze op de volgende wijze gecodeerd:

a	1	*j*	7	*v*	9	*f*	15	*l*	21	*x*	25
o	2	*w*	8	*z*	10	*s*	16	*m*	22	*h*	26
u	3			*g*	11	*c*	17	*n*	23	Apostrof	27
e	4			*b*	12	*p*	18	*r*	24	*n1* (= n)	28
i	5			*d*	13	*t*	19				
y	6			*q*	14	*k*	20				

Een combinatie van een aantal letters geven we een code door die combinatie op te vatten als een getal in een 32-tallig stelsel, waarbij de code van de meest linkse letter de eenheden voorstelt, de code van de tweede letter de 32-vouden, enz.

b.v. *au* $1 + 32 \times 3 = 97$,

sch $16 + 32 \times 17 + 32^2 \times 26 = 27184$.

In het programma zijn de identifiers van deze 'constanten' steeds met kleine letters geschreven en de andere identifiers beginnen met een hoofdletter of zijn geheel met hoofdletters geschreven.
In sectie 4. zult u op de linkerpagina steeds programmatekst aantreffen en op de rechterpagina een uitleg hiervan. Op de linkerpagina zou b.v. kunnen staan

if V[J] = a ∧ C2[J] = 0 ∧ (J = Laatste ∨ C1[J+1] ≠ ch) then V[J]:= aa,

hetgeen betekent:

"als van de J-de lettergreep de <V> een *a* is en de <C2> leeg is, èn de J-de is de laatste lettergreep of de <C1> van de (J+1)-de is geen *ch*, maak dan de <V> van de J-de lettergreep *aa*".

Op de rechterpagina zou u dan als uitleg kunnen treffen:

pa → *paa*
laten → *laaten*
lachen → *laachen.*

Of u treft op de linkerpagina aan

if V[J] = au then V[J]:= ou

en op de rechterpagina

au → *ou.*

Tot nu toe hebben we steeds een duidelijk verschil gemaakt tussen woorden in de Nederlandse spelling en woorden in onze fonetische schrijfwijze: het is echter zo dat de voorbeelden op de rechterpagina's in een soort overgangssituatie zitten; het zijn woorden uit de gewone schrijftaal waaraan in het algemeen al een aantal veranderingen zijn aangebracht. Bijvoorbeeld: voordat *economische* veranderd wordt in *economiesu* is de *ch* al vervangen door c. We hebben echter in de voorbeelden alleen de relevante veranderingen aangebracht: dus *economische* → *economiesu*. Er komen twee typen transformaties op de rechterpagina's voor:

één zoals bij de *au* die *ou* wordt, met weinig of geen kontekst en één zoals bij de *a* die *aa* wordt, met kontekst op het niveau van een woord; deze uitgebreide kontekst duidt er op dat we te maken hebben met een transformatie die onder speciale voorwaarden (te vinden in het programma op de linkerpagina) plaats vindt, het woord op de rechterpagina is slechts een voorbeeld.

De door het programma bestuurde computer leest een woord van de pons-band af, brengt de nodige veranderingen daarin aan, waarbij allerlei stadia tussen de gewone schrijftaal en onze fonetische notatie doorlopen worden, en drukt uiteindelijk het "vertaalde" woord af. We zullen het verloop van die veranderingen schetsen. De cursief gedrukte woorden verwijzen naar labels in het eigenlijke programma.

Splits in lettergrepen: wij gebruikten hiervoor het programma Sylsplit [3] van Brandt Corstius.
Medeklinkers en *Klinkers*: een aantal transformaties waarbij geen of weinig informatie over andere lettergrepen nodig is, worden per lettergreep aangebracht.

b.v.	*ce, ci, cy*	→	*se, si, sy*
	eau	→	*oo*
	jour	→	*zjoer*

Eenlettergrepig woord: een aantal maatregelen voor deze woorden

b.v.	*m'n*	→	*mun*
	de	→	*du.*

Een groot gedeelte van de rest van de transformaties is op deze woorden nooit van toepassing en daarom springen we als het te behandelen woord eenlettergrepig is nu naar *SJWA*.
Voor en achtervoegsels worden afgebroken. Als de eerste overgebleven lettergreep *be, ge, ver, te* of *ont* is ga dan naar *Klemtoonverschuiving*, anders naar *Woorduitgangen*.
Klemtoonverschuiving: de speciale voorvoegsels *be, ge, ver, te* en *ont* hebben de eigenschap dat als zij de eerste overgebleven lettergreep

zijn de klemtoon niet op hen komt maar op de volgende lettergreep. Het is vaak moeilijk om uit te maken of de lettergrepen *be*, *ge*, *ver* en *te* in een bepaalde situatie zo'n speciaal voorvoegsel zijn of gewoon onderdeel van de stam van het woord: vergelijk b.v. *bevel* met *bever*.
Sla *Woorduitgangen* over.
Woorduitgangen: meestal van bastaardachtige afkomst krijgen een speciale behandeling.

NG: *zin-gen* → *zin-en*
zin-ken → *zin-ken*.

Situaties als *zing* en *zink* zijn al behandeld in *Medeklinkers*.
Verkleinwoord: er wordt o.a. voor gezorgd dat de twee laatste e's uit *pennetje* een sjwa worden.
SJWA: de *e*, *i* of *ij* die als sjwa moet worden uitgesproken wordt vervangen door *u*. Hierna kunnen we *ij* door *ei* vervangen en *e* of *i* in een open lettergreep door resp. *ee* of *ie*. (De *a*, *o* en *u* in open lettergrepen zijn al in *Klinkers* verdubbeld.)
Assimileren: hierin worden veranderingen aangebracht zoals:

vaatdoek → *vaaddoek*
vaaddoek → *vaadoek*.

Uitvoer van het nu geheel in onze fonetische notatie geschreven woord.

4. Het programma

```
begin integer Nvoorvgsl1, Nvoorvgsl2;
Nvoorvgsl1:= read; Nvoorvgsl2:= read;
begin integer J, K, Hulp, Hulp2,
  Spatie, Onderstreping,
  Eind, Laatste;

  integer array Code[1:127], Decode[1:27], Let[0:10],

  C1, V, C2[1:15];


  real array Voorvgsl1[1:Nvoorvgsl1], Voorvgsl2[1:Nvoorvgsl2 * 2],
  Letgr[1:15];
  boolean array Klem[1:15];

  procedure Lees in en splits in lettergrepen;
  comment De body van deze procedure is vanwege
  zijn lengte niet afgedrukt;;




  procedure Druk het woord;
  comment De body van deze procedure is niet afgedrukt ;;

  integer procedure Aantal(LC); value LC; integer LC;
  begin integer K, Hulp;
    Hulp:= 0; if LC ≤ 0 then goto STOP;
    for K:= 1, K * 32   while K ≤ LC do Hulp:= Hulp + 1;
  STOP: Aantal:= Hulp
  end Aantal;
```

Aantallen af te splitsen een- en twee lettergrepige voorvoegsels.

Hulpvariabelen.

De betekenis van *Eind* volgt bij de procedure *Verdeel*, de betekenis van *Laatste* volgt bij de procedure *Voor plus*.
In *Code* en *Decode* staan de codering en de decodering. De betekenis van het array *Let* volgt bij de procedure *Verdeel*.
In deze arrays staat per lettergreep resp. de medeklinkercombinatie in aanvang van de lettergreep, de klinkercombinatie en de medeklinkercombinatie aan het eind.
In deze arrays staan de voorvoegsels.
Hierin staan de lettergrepen van het woord.
Klem[I] is true betekent dat de *I*-de lettergreep een klemtoon heeft.

Deze procedure is een gewijzigde versie van het programma 'Silsplit' [3]. De belangrijkste wijzigingen zijn: *'s* wordt tegen het opvolgende woord geplakt. b.v. *'s zaterdags* wordt *'szaterdags*, woorden als *examen* worden niet zoals dat volgens [1] behoort gesplitst in *exa-men* maar in *ex-a-men*. De procedure zorgt er ook voor dat de arrays *Letgr*, *C1*, *V* en *C2* gevuld worden. 'lettergrepen' zonder klinker als b.v. *m'n* of *s'vp* worden in hun geheel in V opgeslagen en niet in C1 en C2.
Deze procedure drukt het in onze fonetische schrijfwijze 'vertaalde' woord aan het eind van het programma af.

De volgende procedures zijn operaties op lettercombinaties (*LC*'s).
Aantal krijgt als waarde het aantal letters van de *LC*.
b.v. *Aantal(sch)=3*

```
real procedure Combineer(LC1, LC2); value LC1, LC2;
integer LC1, LC2;
Combineer:= LC1 + 32 ↑ Aantal(LC1) * LC2;

integer procedure Voorste letters(LC, N); value LC, N; integer LC, N;
Voorste letters:= if N ≤ 0 then 0 else if N ≥ 5 then LC else LC -
(LC : 32 ↑ N) * 32 ↑ N;

integer procedure Achterste letters(LC, N); value LC, N;
integer LC, N;
begin Voorste letters weg(LC, Aantal(LC) - N);
  Achterste letters:= LC
end;

procedure Voorste letters weg(LC, N); value N; integer LC, N;
LC:= if N ≤ 0 then LC else if N ≥ 5 then 0 else LC : 32 ↑ N;

procedure Achterste letters weg(LC, N); value N; integer LC, N;
LC:= Voorste letters(LC, Aantal(LC) - N);

procedure Verdeel(LC); value LC; integer LC;
begin integer K;
  Eind:= Aantal(LC); Let [0]:= 0;
  for K:= 1 step 1 until Eind do
  begin Let[K]:= Voorste letters(LC, 1); Voorste letters weg(LC, 1)
  end
end Verdeel;

procedure Verenig(Begin, Eind, LC); value Begin, Eind;
integer Begin, Eind, LC;
begin integer K;
  LC:= 0; if Eind > 10 then Eind:= 10;
  if Eind > Begin + 4 then Eind:= Begin + 4;
  for K:= Begin step 1 until Eind do LC:= Combineer(LC, Let[K])
end Verenig;
```

Combineer krijgt de waarde van de lettercombinatie die ontstaat door de concatenatie van *LC1* en *LC2*.
b.v. *Combineer(s,ch)=sch*

Voorste letters krijgt de waarde van de eerste *N* letters van de *LC*.
b.v. *Voorste letters(sch,1)=s*

b.v. *Achterste letters(sch,2)=ch*

Voorste letters weg heeft tot effekt dat *LC* de lettercombinatie wordt die ontstaat door van *LC* de eerste *N* letters af te breken.
b.v. *LC=sch. Voorste letters weg(LC,1)* maakt van *LC ch*.
Van de *LC* worden de achterste *N* letters afgebroken.

Verdeel heeft tot effekt dat *Eind* het aantal letters van *LC* wordt en dat de samenstellende letters in het array *Let* komen te staan. (*LC* behoudt daarbij zijn waarde.)

Verenig geeft *LC* de waarde van de lettercombinatie die bestaat uit de letters: *Let*[*Begin*],...,*Let*[*Eind*].

```
  Spatie:= 93; Onderstreping:= 126;
  for K:= 1 step 1 until 15 do
  begin Letgr[K]:= 0; C1[K]:= V[K]:= C2[K]:= 0 end;
  for K:= 0 step 1 until 10 do Let[K]:= 0;
  for K:= 1 step 1 until Nvoorvgsl1 do Voorvgsl1[K]:= read;
  for K:= 1 step 1 until 2 * Nvoorvgsl2 do Voorvgsl2[K]:= read;
  for K:= 1 step 1 until 127 do Code[K]:= 31;
  for K:= 1 step 1 until 27 do
  begin Decode[K]:= READ; Code[Decode[K]]:= K end;
Uitspraak:
  begin integer Voor, Achter, Nletgr, Volgende,
    Apostrof, Sjwa,
    a, o, u, e, i, y, j, w, v, z, g, b, d, q, f, s, c, p, t, k, l,
    m, n, r, x, h, n1, aa, uu, ae, ai, au, oe, on, ont, oo, ou,
    ee, ei, en, er, ie, ij, je, eeu, ieu, eau, jes, ve, ven, ver,
    ge, ges, be, sc, ks, sche, schen, sj, sk, st, te, ti, tie,
    ties, ts, ke, ken, ld, le, ls, lt, mbt, ms, nt, nd, nds, ndst,
    ne, ns, nst, rd, rds, rdst, re, rs, rst, rt, rwt, he, den,
    heid, jour, lij, lijk, aau, oui, oei, eui, oeu, mt, md, zj, ch,
    aal;

    procedure Codeer de lettercombinaties;
    comment De body van deze procedure is vanwege zijn lengte
    niet afgedrukt;;
    switch L:= L1, L2, L3;

    procedure Voor plus(J); integer J;
    begin Voor:= Voor + J; Nletgr:= Nletgr - J end;


    procedure Achter min(J); integer J;
    begin Achter:= Achter - J; Nletgr:= Nletgr - J end;

    Boolean procedure Klinker(Letter); value Letter; integer Letter;
    Klinker:= Letter < 7;
```

Spatie en *Onderstreping* krijgen hun waarde.
Enkele arrays worden schoongemaakt.

De voorvoegsels en de codering worden ingelezen.

Zie voor de betekenis van *Voor*, *Achter* en *Nletgr* de procedure *Voor plus*.
Volgende is een hulpvariabele.

De taalkundige identifiers worden gedeclareerd.

Een aanroep van *Codeer de lettercombinaties* geeft bovenstaande constanten hun waarde in onze code.

Voor plus en *Achter min* worden gebruikt om lettergrepen af te breken; een aantal van de uitspraakregels worden op de afgebroken lettergrepen niet meer toegepast. Aan het begin van de bewerkingen is *Voor* 1 en *Achter=Laatste=Nletgr*=het aantal lettergrepen van het woord. *Voor plus(J)* verhoogt *Voor* met *J* en *Achter min(J)* vermindert *Achter* met *J*. *Nletgr* wordt in beide gevallen met *J* verminderd. Sommige uitspraakregels worden alleen toegepast op de lettergrepen met nummer *Voor* tot en met *Achter* en andere op de lettergrepen met nummer 1 tot en met *Laatste* (dus op allemaal).
true voor: *a*, *o*, *u*, *e*, *i*, *y*.

```
Boolean procedure Medeklinker met assimilatie(Letter);
value Letter; integer Letter;
Medeklinker met assimilatie:= Letter > 8 ∧ Letter < 21;

Boolean procedure Liquida(Letter); value Letter; integer Letter;
Liquida:= Letter > 20 ∧ Letter < 25;

Boolean procedure Klinkercombinatie(LC); value LC; integer LC;
Klinkercombinatie:= Klinker(Voorste letters(LC, 1));


procedure Maak stemloos(Letter); integer Letter;
if letter  > 8 ∧ Letter < 15 then Letter:= Letter + 6;

procedure Maak stemhebbend(Letter); integer Letter;
if Letter > 14 ∧ Letter < 21 then Letter:= Letter - 6;

procedure Alleenstaande(Letter, J); value Letter, J;
integer Letter, J;
if ¬Klinker(Letter) ∧ Letter < 27 ∧ J < 15 then
begin C1[J]:= V[J]:= C2[J]:= 0; Laatste:= J; if Letter = z then
  begin C1[J]:= z; V[J]:= e; C2[J]:= t end
  else if Letter = q then
  begin C1[J]:= k; V[J]:= uu end
  else if Letter = x then
  begin V[J]:= i; C2[J]:= ks end
  else if Letter = k ∨ Letter = h then
  begin C1[J]:= Letter; V[J]:= aa end
  else if Letter = c then
  begin C1[J]:= s; V[J]:= ee end
  else if Letter = f ∨ Letter = s ∨ Liquida(Letter) then
  begin V[J]:= e; C2[J]:= Letter end
  else
  begin C1[J]:= Letter; V[J]:= ee end
end Alleenstaande letter;
```

true voor: *v,z,g,b,d,q,f,s,c,p,t,k.*

true voor: *l,m,n,r.*

true als *LC* uitsluitend uit klinkers bestaat. In de situaties waarin *Klinkercombinatie* in het programma wordt aangeroepen is het voldoende te kijken of de eerste letter van *LC* een klinker is.

v,z,g,b,d,q	→	*f,s,c,p,t,k*
f,s,c,p,t,k	→	*v,z,g,b,d,q*

Deze procedure wordt gebruikt om alleenstaande medeklinkers en afkortingen waarin geen klinker voorkomt uit te spreken.

z	→	*zet*
q	→	*kuu*
x	→	*iks*
k,h	→	*kaa,haa*
c	→	*see*
f,s,l,m,n,r	→	*ef,es,el,em,en,er*
b,d,g,j,p,t,v,w	→	*bee,dee,gee,jee,pee,tee,vee,wee*

```
Boolean procedure Ing klank(J); value J; integer J;
Ing klank:= if V[J] ≠ i ∨ J > Achter then false else Voorste
letters(C2[J], 1) = n1 ∨ (J < Achter ∧ C2[J] = n ∧ C1[J + 1] = g
∧ E is sjwa(J + 1));

Boolean procedure I is sjwa(J); value J; integer J;
I is sjwa:= if Klem[J] ∨ V[J] ≠ i ∨ J > Achter ∨ Aantal(C1[J])
> 1 ∧ C1[J] ≠ st then false else Voorste letters(C2[J], 1) = g
∨ C2[J] = k ∨ (J < Achter ∧ C2[J] = 0 ∧ (Letgr[J + 1] = ken ∨
C1[J + 1] = g ∧ (E is sjwa(J + 1) ∨ Ing klank(J + 1))));

Boolean procedure I is bijzonder(J); value J; integer J;
I is bijzonder:= I is sjwa(J) ∨ Ing klank(J);

Boolean procedure E is sjwa(J); value J; integer J;
if V[J] = Sjwa then E is sjwa:= true else if Klem[J] ∨ V[J] ≠ e
∨ J > Achter then E is sjwa:= false else
begin
  Boolean procedure Achterste e is sjwa;
  Achterste e is sjwa:= if J ≠ Achter then false else C2[J] = lt
  ∨ C2[J] = ld ∨ C2[J] = rt ∨ C2[J] = rd ∨ C2[J] = mt ∨ C2[J] =
  md ∨ (C2[J] = nt ∧ C1[J] = k) ∨ C2[J] = nd ∨ C2[J] = nds ∨
  C2[J] = ndst ∨ C2[J] = rst ∨ C2[J] = rdst;

  Boolean procedure Afgesloten e;
  Afgesloten e:= if J ≥ Achter ∨ Aantal(C2[J]) ≠ 1 then false
  else if Liquida(C2[J]) ∧ (C1[J + 1] = d ∨ C1[J + 1] = s)
  ∧ V[J + 1] ≠ i then false else Aantal(C1[J + 1]) = 1 ∧
  Bijzondere e of i(J + 1);


  Boolean procedure Open e;
  Open e:= if J ≥ Achter ∨ C2[J] ≠ 0 then false else
  ¬Liquida(C1[J + 1]) ∧ Aantal(C1[J + 1]) = 1 ∧ Bijzondere e
  of i(J + 1);
```

true in: *lezingen, vergeving*. (Als deze procedure wordt aangeroepen dan is de *ng* wanneer deze in een lettergreep staat al vervangen door *n* en wanneer deze over twee lettergrepen verdeeld is nog niet.)

true in: *zalig, lenigst, havik, haviken, verenigen, vereniging, ernstig*.

true als de *i* een sjwa of onderdeel van een ing-klank is.

Als de *J*-de lettergreep de klemtoon heeft of $V[J] \neq e$ dan in ieder geval false.

Als *J* niet het nummer van de laatste lettergreep is dan false.
true in: *stapelt, gestapeld, verbetert, gebeterd, asemt, geasemd, rekent*(maar false in: *dirigent, consument*), *gerekend, vervelends, vervelendst, uiterst, verwonderdst*.

false in: *bovenste, stelpende, westerse* (maar true in: *koerswending*).
true in: *veiligstelling, bloedstelpende*.

true in: *salpeter, aardbeving, paardedeken*.
false in: *stamelende, stameling, wasemen*.

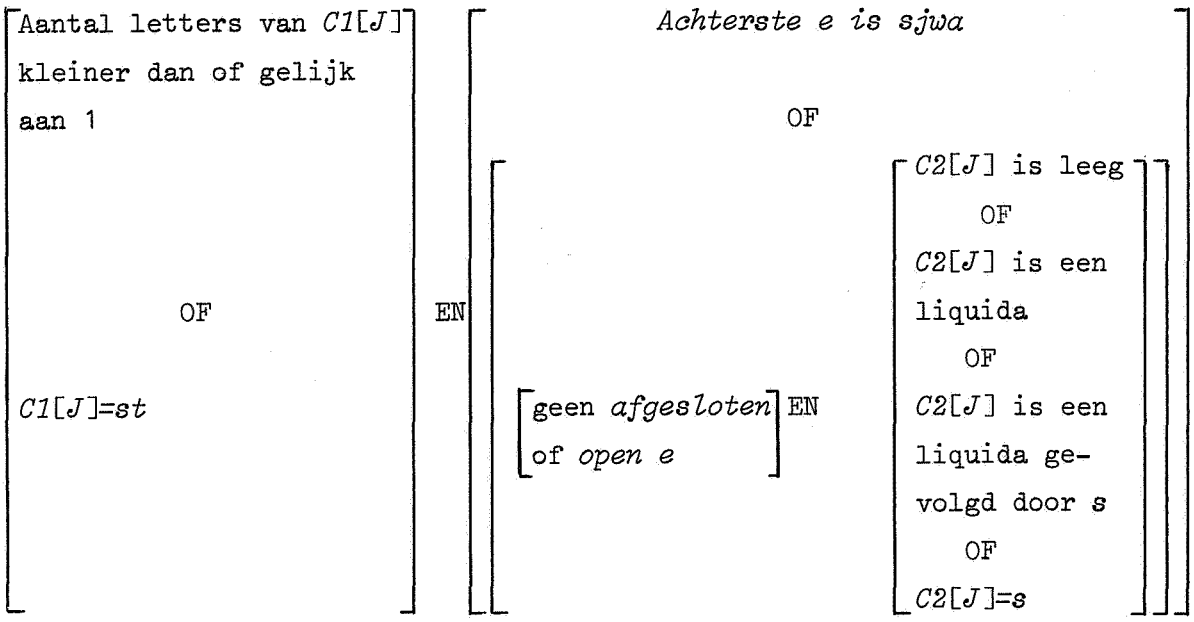
```
E is sjwa:= (Aantal(C1[J]) ≤ 1 ∨ C1[J] = st) ∧ (Achterste e
is sjwa ∨ (¬(Afgesloten e ∨ Open e) ∧ (C2[J] = 0 ∨ C2[J] = r ∨
C2[J] = rs ∨ C2[J] = m ∨ C2[J] = ms ∨ (C2[J] = n ∧ (J = Achter
∨ C1[J + 1] ≠ t)) ∨ C2[J] = ns ∨ (C2[J] = l ∧ C1[J] ≠ w) ∨
C2[J] = ls ∨ (C2[J] = s ∧ (J = Achter ∨ C1[J + 1] ≠ s) ∧ C1[J]
≠ r ∧ C1[J] ≠ s))))
```

```
end E is sjwa;

Boolean procedure Bijzondere e of i(J); value J; integer J;
Bijzondere e of i:= I is bijzonder(J) ∨ E is sjwa(J);

Boolean procedure IJ is sjwa(J); value J; integer J;
IJ is sjwa:= if Klem[J] ∨ C1[J] ≠ l ∨ V[J] ≠ ij then false else
Voorste letters(C2[J], 1) = k ∨ (J < Laatste ∧ C2[J] = 0 ∧ C1[J
+ 1] = k ∧ Bijzondere e of i(J + 1));
```

De e wordt een sjwa als aan de volgende voorwaarden is voldaan.



Hierop zijn nog uitzonderingen waarbij de e geen sjwa wordt in het programma opgenomen.

-wel	*(hoewel)*
en-t	*(commentaar)*
es-s	*(agressor)*
-res	*(onderwijzeres)*
-ses	*(prinses)*

true als de klinker een e is die sjwa wordt of een *i* die of sjwa wordt of onderdeel van een ing-klank is.

true in: *makkelijk, achterlijke.*

```
procedure Breek eventueel een voorvoegsel af;
begin integer Hulp;
  real VC2;
  Boolean Vr1, Vr2;

  Boolean procedure Voorvoegsel 1;
  begin integer K;
    Boolean B;
    B:= false;
    for K:= 1, K + 1 while ¬B ∧ K ≤ Nvoorvgsl 1 do if Letgr[Voor]
    = Voorvgsl1[K] then B:= true; Voorvoegsel 1:= B
  end Voorvoegsel 1;


  Boolean procedure Voorvoegsel 2;
  begin integer K;
    Boolean B;
    B:= false;
    for K:= 1, K + 2 while ¬B ∧ K < 2 * Nvoorvgsl 2 do if
    Letgr[Voor] = Voorvgsl 2[K] ∧ Letgr[Voor + 1] = Voorvgsl 2[K
    + 1] then B:= true; Voorvoegsel 2:= B
  end Voorvoegsel 2;

  Vr2:= Nletgr ≥ 2 ∧ Voorvoegsel 2;
  Vr1:= if Vr2 then false else Nletgr ≥ 1 ∧ Voorvoegsel1;
  if Vr1 ∨ Vr2 then
  begin Klem[Voor]:= true; if Vr2 then
    begin V[Voor + 1]:= u; Hulp:= 2 end
    else Hulp:= 1; if Voor + Hulp > Achter then goto SJWA;
    Voor plus(Hulp); Breek eventueel een voorvoegsel af;
    if Voor = Achter then
    begin if Letgr[Achter - 1] = ge then
      begin Voor plus(- 1); V[Voor]:= e; goto Twee lettergrepen
      end;
      VC2:= Combineer(V[Achter], C2[Achter]);
```

Afbreken wil hier zeggen: *Voor* krijgt het nummer van de eerste lettergreep achter het voorvoegsel en *Nletgr* wordt aangepast. In een aantal van de verdere veranderingen aan het woord doen we alsof de lettergreep met het nummer *Voor* de eerste lettergreep van het woord is.
Bij alle voorvoegsels die we tegenkomen krijgt de eerste lettergreep een klemtoon. Van de tweelettergrepigen wordt de *e* van de tweede lettergreep een *u*.
Niet alle voorvoegsels worden afgesplitst; er zijn vier redenen waarom wel afgebroken wordt.

1 Achter het voorvoegsel staat een van de klemtoon-verschuivende lettergrepen (*be, ge, ver, te, ont*).
b.v. *voorgesteld*; de laatste lettergreep moet straks een klemtoon krijgen en de *e* uit die lettergreep zal daarom nooit *u* worden.

2 De *V* van de volgende lettergreep is *e*.
b.v. *mededelen*; de derde lettergreep krijgt een klemtoon en de *e* uit deze lettergreep kan daardoor geen sjwa meer worden.

3 De *C1* van de volgende lettergreep is een *k* of een *g*.
b.v. *aankomen*, *aangeven*; een eventuele *n* mag niet de nasale *ŋ* worden.

4 Op het voorvoegsel volgt een voorvoegsel dat om een van deze vier redenen afgebroken moet worden(de procedure is dus recursief).
b.v. *onopgevoede*.

Wij gebruiken twintig eenlettergrepige voorvoegsels: *in, on, aan, uit, waar, op, voor, af, her, on, bij, wel, een, bouw, hand, deel, bonds, jaar, toon, vak* en dertien tweelettergrepigen: *binnen, mede, over, onder, tele, buiten, boven, samen, achter, inter, tegen, alge, nage*.
Zoals u ziet hebben we het begrip voorvoegsel erg ruim opgevat en

```
        if (Aantal(C1[Achter]) = 1 ∨ C1[Achter] = st) ∧ (VC2 = en
        ∨ VC2 = er ∨ VC2 = e) then
        begin voor plus(- Hulp); V[Achter]:= u; goto SJWA end
      end;
      if Letgr[Voor] = be ∨ Letgr[Voor] = ge ∨ Letgr[Voor] = ver ∨
      Letgr[Voor] = te ∨ Letgr[Voor] = ont then goto Klemtoon
      verschuiving else if V[Voor] = e ∨ C1[Voor] = g ∨ C1[Voor]
      = k then goto Woorduitgangen; Voor plus(- Hulp);
      goto Woorduitgangen
    end
  end Breek eventueel een voorvoegsel af;
Eigenlijke programma: Codeer de lettercombinaties;
Splits: Lees in en splits in lettergrepen; Voor:= 1;
  Nletgr:= Achter:= Laatste;
  for J:= 1 step 1 until Laatste do
  begin Klem[J]:= false; if C1[J] = ch then
    begin if V[J] ≠ e ∧ V[J] ≠ ie then C1[J]:= sj;
      if V[J] = au then V[J]:= oo
    end;
    if J > 1 ∧ C1[J] = s then
    begin if C2[J - 1] = 0 then C1[J]:= z end;
    if C1[J] = c ∨ C1[J] = sc then
    begin volgende:= Voorste letters(V[J], 1);
      C1[J]:= if Volgende = e ∨ Volgende = i ∨ volgende = y then s
      else if C1[J] = c then k else sk
    end;
    if Letgr[J] = jour then
    begin C1[J]:= zj, V[J]:= oe; goto Volgende lettergreep end;
    if C2[J] = w then
    begin if V[J] = o ∨ V[J] = e then
      begin C2[J]:= f; goto Volgende lettergreep end;
      if V[J] = u then
      begin V[J]:= uu; goto Volgende lettergreep end
    end;
    Hulp:= 1;
```

behalve echte voorvoegsels ook woorddelen opgenomen die vaak onderdeel van samenstellingen zijn. De 'voorvoegsels' *alge* en *nage* worden gebruikt om de voorvoegsels *al* en *na* kontekst-sensitief af te breken. (Het feit dat een lettergreep afgebroken is zullen we aangeven door die lettergreep tussen haakjes te zetten.)
b.v. *algemene* wordt *(al)gemene* maar *allemaal* wordt niet *(al)lemaal*, omdat anders *le* de klemtoon krijgt.
Speciale maatregelen zijn nodig om te zorgen dat *voorste* niet *(voor)ste* maar *voorstel* wel *(voor)stel* wordt. Indien het mogelijk is om of een eenlettergrepig of een tweelettergrepig voorvoegsel af te splitsen wordt de voorkeur aan het laatste gegeven.
Hier begint het hoofdprogramma. De lettercombinaties worden nu daadwerkelijk gecodeerd en een woord wordt van de band af ingelezen, in lettergrepen gesplitst en opgeslagen in de arrays: *Letgr*, *C1*, *V* en *C2*. Ook *Nletgr*, *Achter* en *Laatste* krijgen hun startwaarde.

chagrijn	→	*sjagrijn*
chauffeur	→	*sjoffeur*
basis	→	*bazis*
citroen	→	*sitroen*
cactus	→	*kactus*
scala	→	*skala*
journalist	→	*zjoernalist*
sowjet	→	*soffjet*
duw	→	*duuw*

```
Medeklinkers: Verdeel(if Hulp = 1 then C1[J] else C2[J]);
  if Eind = 0 then goto medeklinkers klaar;
  if J = 1 ∧ Eind ≥ 3 then
  begin if Let[2] = s ∧ (Let[3] = s ∨ Let[3] = z) then Let[3]:= 0
  end;
  if Let[1] = x ∧ Hulp = 1 then
  begin Let[0]:= k; Let[1]:= s end;
  for K:= 1 step 1 until Eind - 1 do
  begin if Let[K] = c then
    begin if Let[K + 1] = h then Let[K + 1]:= 0 else Let[K]:= k
    end
    if Let[K] = Apostrof then
    begin Let[K]:= 0; if K = 1 ∧ Hulp = 2 ∧ Klinker(V[J]) then
      begin if V[J] = i ∨ V[J] = y then V[J]:= ie else V[J]:=
        Combineer(V[J], V[J])
      end
    end;
    if Hulp = 2 ∧ Let[K] = n then
    begin if Let[K + 1] = g then
      begin Let[K + 1]:= 0; Let[K]:= n1 end
      else if Let[K + 1] = k then Let[K]:= n1
    end
  end;
  if Let[Eind] = c then Let[Eind]:= k;
  if Let[Eind] = Apostrof then Let[Eind]:= 0;
  if Let[Eind] = q then
  begin Let[Eind]:= k; Eind:= Eind + 1; Let[Eind]:= w;
    Voorste letters weg(V[J], 1)
  end;
  if Let[Eind] = x then
  begin Let[Eind]:= k; if J = Laatste then goto KS;
    if C1[J + 1] = 0 then
    begin C1[J + 1]:= s; goto Medeklinkers klaar end;
  KS: Eind:= Eind + 1; Let[Eind]:= s
  end;
```

'szaterdags → *'saterdags*

(in *Lees in en splits in lettergrepen* is de *'s* tegen het opvolgende woord aangeplakt)

xeon → *kseon*

ch → *c*

crisis → *krisis*

opa's → *opaas*

ski's → *skies*

zo'n → *zoon*

ng → *n*

nk → *nk*

(Alleen als de *ng* of de *nk* in één lettergreep staan)

kactus → *kaktus*

Frits' → *Frits*

qu → *kw*

Boxer → *Bokser*

Max → *Maks*

```
Medeklinkers klaar: if Hulp = 1 then
  begin Verenig(0, Eind, C1[J]); Hulp:= 2;
    goto Medeklinkers
  end
  else Verenig(1, Eind, C2[J]);
Klinkers: if C2[J] = 0 ∧ (J = Laatste ∨ C1[J + 1] ≠ ch) then
  begin if V[J] = a ∨ V[J] = o ∨ V[J] = u then
    begin V[J]:= Combineer(V[J], V[J]); goto Volgende lettergreep
    end
  end;
  if Achterste letters(V[J], 1) = y then
  begin Achterste letters weg(V[J], 1); V[J]:= Combineer(V[J], i)
  end;
  if Aantal(V[J]) < 4 ∧ Aantal(V[J]) > 0 then goto
  L[Aantal(V[J])];
L2: if V[J] = au then V[J]:= ou else if V[J] = ae then V[J]:=
  e; goto Volgende lettergreep;
L3: if V[J] = eeu ∨ V[J] = ieu then
  begin if C2[J] = 0 then Achterste letters weg(V[J], 1);
    goto Volgende lettergreep
  end leeuwen, kieuwen;
  if V[J] = eui ∨ V[J] = aau ∨ V[J] = oeu then
  begin Voorste letters weg(V[J], 1); goto L2 end;
  if V[J] = oui then
  begin V[J]:= oei; goto Volgende lettergreep end;
  if V[J] = eau then
  begin V[J]:= oo; goto Volgende lettergreep end;
  if J < Laatste ∧ C2[J] = 0 ∧ C1[J + 1] = 0 then
  begin if Klinkercombinatie(V[J]) ∧ Bijzondere e of i(J + 1)
    then
    begin C1[J + 1]:= j; Achterste letters weg(V[J], 1) end
  end;
L1:
Volgende lettergreep:
end;
```

laten	→	*laaten*
lopen	→	*loopen*
lachen	≠	*laachen*

(Met ≠ geven we aan dat de transformatie niet optreedt.)

y	→	*i*

au	→	*ou*
ae	→	*e*
leeuwen	→	*leewen*
kieuwen	→	*kiewen*

eui	→	*ui*
aau	→	*ou*
oeu	→	*eu*
oui	→	*oei*
eau	→	*oo*

aaien	→	*aajen*

```
Eenlettergrepig woord: if Nletgr = 1 then
  begin Klem[1]:= true;
    if C1[1] > 0 ∧ V[1] = e ∧ C2[1] = 0 then V[1]:= u;
    if C1[1] = 0 ∧ ¬Klinkercombinatie(V[1]) ∧ C2[1] = 0 then
    begin Verdeel(V[1]);
      for K:= 1 step 1 until Eind do if Let[K] = Apostrof then
      begin Verenig(1, K - 1, C1[1]); V[1]:= u;
        Verenig(K + 1, Eind, C2[1]); goto Uitvoer
      end;
      for K:= 1 step 1 until Eind do Alleenstaande(Let[K], K)
    end;
    goto SJWA
  end Eenlettergrepig woord;
Voor en achtervoegsels:
  begin if Letgr[Achter] = heid then
    begin Achter min(1); if Nletgr = 1 then goto SJWA end;
    if Letgr[Achter - 1] = he ∧ Letgr[Achter] = den then
    begin V[Achter - 1]:= ee; V[Achter]:= u; Achter min(2);
      if Nletgr = 0 then goto Assimileren else if Nletgr = 1 then
      goto SJWA
    end;
    Breek eventueel een voorvoegsel af;
    if ¬(Letgr[Voor] = be ∨ Letgr[Voor] = ge ∨ Letgr[Voor] = te ∨
    Letgr[Voor] = ver ∨ Letgr[Voor] = ont) then goto
    Woorduitgangen
  end Voor en Achtervoegsels;
Klemtoonverschuiving:
  begin if (Letgr[Voor] = ver ∨ Letgr[Voor] = be ∨ Letgr[Voor] =
    te) ∧ Letgr[Voor + 1] = ge then
    begin if Nletgr ≥ 4 ∧ Bijzondere e of i(Voor + 2) then
      begin V[Voor]:= u; Voor plus(1); Klem[Voor]:= true; goto NG
      end;
      if Nletgr = 3 then
      begin V[Voor]:= u; Voor plus(1); goto Twee lettergrepen end;
      V[Voor]:= u; V[Voor + 1]:= u; Voor plus(2); Klem[Voor]:= true;
```

de	→	*du*
m'n	→	*mun*
svp	→	*es-vee-pee*
bekendheid	→	*bekend(heid)*
zekerheden	→	*zeker(heedun)*
voorkomen	→	*(voor) komen*
onopgemerkt	→	*(on) (op) gemerkt*

Als de voorste overgebleven lettergreep niet *ge*, *be*, *ver*, *te* of *ont* is ga dan naar *woorduitgangen* (pag. 34).

In het algemeen krijgt de eerste lettergreep een klemtoon; in *Klemtoonverschuiving* worden uitzonderingen gemaakt.

Klemtoonverschuiving:

begevende	→	*(bu) gevende*
vergevingen	→	*(vur) gevingen*
vergeten	→	*(vur) geten*
tegemoet	→	*(tu) gemoet*
vergezellen	→	*(vur) (gu) zellen*
begeleiden	→	*(bu) (gu) leiden*

```
    goto NG
  end;
  if Letgr[Voor] = te ∧ ¬Bijzondere e of i(Voor + 1) then
  begin V[Voor]:= u; Voor plus(1) end;
  if Letgr[Voor] = be ∨ Letgr[Voor] = ge ∨ Letgr[Voor] = ver
  then
  begin if Nletgr = 2 then goto Twee lettergrepen;
    if ¬I is sjwa(Voor + 1) then
    begin V[Voor]:= u; Voor plus(1) end
   end;
  if Letgr[Voor] = ont then Voor plus(1);
  Breek eventueel een voorvoegsel af; Klem[Voor]:= true; goto NG;

Twee lettergrepen: if I is sjwa(Achter) ∨ (Aantal(C1[Achter]) =
  1 ∧ V[Achter] = e ∧ (Liquida(C2[Achter]) ∨ C2[Achter] = 0 ∨
  C2[Achter] = ls ∨ C2[Achter] = ms ∨ C2[Achter] = rs ∨
  C2[Achter] = ns)) then V[Achter]:= u else
  begin V[Voor]:= u; Voor plus(1) end;
  Klem[Voor]:= true; goto SJWA
end Klemtoonverschuiving;
```

teleurstellen	→	*(tu)leurstellen*

vervelen	→	*(vur)velen*
bezige	↛	*(bu)zige*
ontkende	→	*(ont)kende*

Het is nodig om na het afbreken van klemtoonverschuivende lettergrepen *Breek eventueel een voorvoegsel af* aan te roepen (b.v. *bevoordelen*). Deze procedure zelf zorgt weer voor eventueel terugspringen naar *Klemtoonverschuiving* (b.v. *verongelijkt*).
Over blijven nu nog tweelettergrepige woorden (ook b.v. *(vur)geten*) waarvan de eerste lettergreep *be, ge* of *ver* is.
Als de tweede lettergreep aan de volgende voorwaarde voldoet:

[[De V is een *i* die sjwa wordt]
OF
[[Aantal letters van de C1 is 1] EN [V is *e*] EN [C2 is leeg OF C2 is een liquida OF C2 is een liquida gevolgd door *s*]]]

dan wordt de V van de achterste lettergreep *u* anders wordt de V van de eerste *u* en wordt deze lettergreep afgesplitst.

b.v.	*benig*	→	*benug*
	verse	→	*versu*
	gevers	→	*gevurs*
	geval	→	*(gu)val*
	verteld	→	*(vur)teld*

```
Woorduitgangen:
  begin Klem[Voor]:= true; if Nletgr = 1 then goto NG;
  TIE: for J:= Voor + 1 step 1 until Achter do
    begin if Letgr[J] = tie then goto if J ≠ Achter ∧ (Letgr[J + 1]
      = ve ∨ Letgr[J + 1] = ven ∨ Letgr[J + 1] = ke ∨ Letgr[J + 1]
      = ken) then J en 1 else TS of S:
      if Letgr[J] = ti then goto if J = Achter then MIE else if
      Letgr[J + 1] = o ∨ Letgr[J + 1] = on ∨ Letgr[J + 1] = a ∨
      Letgr[J + 1] = aal then TS of S else J en 1;
      if Letgr[J] = ties then goto TS of S; goto J en 1;
    TS of S: if C2[J - 1] = s then goto J en 1;
      Klem[J - 1]:= true; C1[J]:= if C2[J - 1] = 0 then ts else s;
      if J ≥ Achter - 1 then goto NG;
    J en 1:
    end TIE;
    if Nletgr = 2 then goto ISCH;
  MIE: for J:= Achter - 1, Achter do if V[J] = ie ∧ C2[J] = 0
    then
    begin if V[J - 1] = e ∧ C2[J - 1] = 0 then Klem[J - 1]:= true
      else Klem[J]:= true; goto NG
    end MIE;
  ISCH: if V[Achter] = i ∧ C2[Achter] = sc then
    begin Klem[Achter - 1]:= true; V[Achter]:= ie; C2[Achter]:= s;
      goto NG
    end ISCH;
    if Nletgr ≤ 2 then goto NG;
  SCHE: if Letgr[Achter] = sche ∨ Letgr[Achter] = schen then
    begin if Nletgr ≥ 3 ∧ V[Achter - 1] = i ∧ C2[Achter - 1] = 0
      then
      begin V[Achter - 1]:= ie; Klem[Achter - 2]:= true end;
      C1[Achter]:= s; V[Achter]:= u
    end SCHE;
```

Hierbij ontstaan vele fouten:

bevel	→	* *bevul*
vertel	→	* *vertul*

Woorduitgangen:

Woorduitgangen en *Klemtoonverschuiving* staan parallel; nooit doorloopt een woord beiden. Dit hebben we gedaan omdat woorden die met een klemtoonverschuivende lettergreep beginnen i.h.a. inheemse woorden zijn en wij willen deze groep in *woorduitgangen* vermijden. Als een lettergreep een klemtoon krijgt zullen wij dat aangeven door de *V* van de lettergreep te onderstrepen.

politie	→	*polit̲sie*
politieke	→	*polit̲sieke*
station	→	*stat̲sion*
potentiaal	→	*poten̲siaal*
petities	→	*petit̲sies*
amnestie	→	*amnest̲sie*

economie	→	*economi̲e*
amnestie	→	*amnesti̲e*

economisch	→	*econ̲omies*

economische	→	*econ̲omiesu*

```
  LE: if Letgr[Achter - 1] = ne ∧ Letgr[Achter] = le then
    begin Klem[Achter - 1]:= true; V[Achter]:= u; goto NG
    end LE;
  GE: if Letgr[Achter] = ge ∨ Letgr[Achter] = ges then
    begin if V[Achter - 1] ≠ i ∧ C2[Achter - 1] = 0 then
      begin Klem[Achter - 1]:= true; C1[Achter]:= zj; V[Achter]:= u;
        goto NG
      end
    end GE;
    goto if Nletgr ≥ 4 ∨ Letgr[Voor] = re then ERING of EREN else
    NG;
  ERING of EREN: for J:= Achter - 3, Achter - 2 do if J ≥ Voor
    then
    begin if Aantal(V[J]) ≤ 2 ∧ C2[J] = 0 then
      begin if V[J + 1] = e ∧ C2[J + 1] = 0 ∧ C1[J + 2] = r ∧ (V[J
        + 2] = e ∧ (C2[J + 2] = n ∨ C2[J + 2] = nd) ∨ Ing klank(J
        + 2)) then
        begin Klem[J]:= false; Klem[J + 1]:= true; V[J + 1]:= ee;
          goto NG
        end
      end
    end ERING of EREN
  end Woorduitgangen;
NG: for J:= Voor step 1 until Achter - 1 do if C2[J] = n ∧
  Aantal(V[J]) = 1 then
  begin if C1[J + 1] = g then
    begin C1[J + 1]:= 0; C2[J]:= n1 end;
    if C1[J + 1] = k then C2[J]:= n1
  end NG;
Verkleinwoord: if Nletgr ≥ 2 then
  begin if Letgr[Achter] = je ∨ Letgr[Achter] = jes then
    begin V[Achter]:= u; Achter min(1);
      if Nletgr = 1 then goto SJWA;
      if V[Achter] = e ∧ C2[Achter] = t then
      begin if C2[Achter - 1] = C1[Achter] ∧ (Liquida(C1[Achter])
```

criminele → *criminelu*

garage → *garazju*
menige → *menizju*

Alleen woorden die uit meer dan drie lettergrepen bestaan of die met 're' beginnen.

consumeren → *consumeeren*
regering → *regeering*
corrigerende → *corrigeerende*

zingen → *zinen*
zinken → *zinken*

kopje → *kop(ju)*
pennetje → *pennet(ju)*
pennet(ju) → *pennut(ju)*

```
        ∨ C1[Achter] = b ∨ C1[Achter] = g) ∨ (C2[Achter - 1] = 0
        ∧ C1[Achter] = n1) then V[Achter]:= Sjwa
      end pennetje
    end
  end Verkleinwoord;
SJWA: for J:= Achter step - 1 until Voor do if E is sjwa(J) then
  V[J]:= Sjwa;
  for J:= 1 step 1 until Laatste do
  begin if IJ is sjwa(J) ∨ I is sjwa(J) ∨ V[J] = Sjwa then V[J]:= u;
    if (V[J] = e ∨ V[J] = i) ∧ C2[J] = 0 then V[J]:=
    Combineer(V[J], e); if V[J] = ij then V[J]:= ei;
    if V[J] = ai then V[J]:= e
  end SJWA;
Assimileren: for J:= 1 step 1 until Laatste do
  begin if Aantal(C1[J]) > 1 then
    begin if Achterste letters(C1[J], 1) = h then Achterste letters
      weg(C1[J], 1);
    end theater;
    if C2[J] = h then C2[J]:= 0; if C2[J] > 0 then
    begin Verdeel(C2[J]);
      Volgende:= if J < Laatste then Voorste letters(C1[J + 1], 1)
      else 0;
      if Let[Eind] = g ∧ Volgende = g then Eind:= Eind - 1;
      if Eind ≤ 4 then
      begin Hulp:= Voorste letters(Combineer(C2[J], Volgende), 3);
        if Hulp = rwt ∨ Hulp = mbt then Let[2]:= 0;
      end ambtstermijn, erwten;
      Maak stemloos(Let[Eind]);
      for K:= Eind - 1 step - 1 until 1 do
      begin Maak stemloos(Let[K]);
        if Let[K] = Let[K + 1] then Let[K]:= 0
      end baadt;
      if J = Laatste ∨ Eind = 0 then goto Assimileren klaar;
    Assimilatie: if ¬Medeklinker met assimilatie(Let[Eind]) then
      goto DD;
```

e	(die sjwa is)	→	*u*
ij	(die sjwa is)	→	*u*
i	(die sjwa is)	→	*u*
i, e	(open lettergreep)	→	*ie, ee*
ij		→	*ei*
ai		→	*e*

th	→	*t*
bah	→	*ba*

zeggen	→	*zegen*

rwt	→	*rt*
mbt	→	*mt*
had	→	*hat*
baadden	→	*baatden*
hebzucht	→	*hepzucht*
baadt	→	*baatt*
baatt	→	*baat*

```
      Progressief: if Volgende = v ∨ Volgende = z ∨ Volgende = g
        then Maak stemloos(Volgende);
      Regressief: if Volgende = b ∨ Volgende = d then Maak
        stemhebbend(Let[Eind]);
      DD: if Let[Eind] = Volgende then Eind:= Eind - 1;
        if Eind < 2 then goto SJ;
        if Let[Eind - 1] = s ∧ Let[Eind] = t then
        begin if Voorste letters(C1[J + 1], 2) = st then Eind:= Eind
          - 2; if Letgr[J + 1] = je ∨ Letgr[J + 1] = jes then
          begin Eind:= Eind - 2; Volgende:= sj end
        end Vaststeken, kastje;
      SJ: if (Let[Eind] = s ∨ Let[Eind] = t ∨ Let[Eind] = p) ∧
        (Letgr[J + 1] = je ∨ Letgr[J + 1] = jes) then
        begin Volgende:= Combineer(Let[Eind], J); Eind:= Eind - 1
        end boompje;
        Voorste letters weg(C1[J + 1], 1);
        C1[J + 1]:= Combineer(Volgende, C1[J + 1]);
      Assimileren klaar: Verenig(1, Eind, C2[J])
      end
    end Assimileren;
  Uitvoer: Druk het woord; goto Splits
  end
end
end
```

hepzucht	→	*hepsucht*
lacgas (lachgas)	→	*laccas*
zakdoek	→	*zaqdoek*
baatden	→	*baadden*
baadden	→	*baaden*
laccas	→	*lacas*
vaststeken	→	*vasteken*
kastje	→	*kasje*
boomp-je	→	*boom-pje*
kat-je	→	*ka-tje*

Het 'vertaalde' woord wordt afgedrukt en we springen terug naar *Splits* op pagina 24.

5. Resultaten

Kranten

In "Formal Properties of Newspaper Dutch" [7] staat een frekwentietelling van kranteteksten (totaal 44299 woorden waaronder 9380 verschillende). Wij hebben ons programma getest op de 3573 verschillende woorden die in dit materiaal meer dan eenmaal voorkomen (in totaal 38492 woorden).
Bij het evalueren van de test werden we met de volgende moeilijkheden geconfronteerd.

1: Er bestaat niet zoiets als een fonetisch woordenboek voor de Nederlandse taal; bij ieder woord moesten we zelf beslissen of de fonetische representatie al dan niet juist was.

2: In feite kan de fonetische representatie van een woord niet goed geevalueerd worden met de qualificaties "goed" en "fout", er is behoefte aan tussenliggende qualificaties als "acceptabel", "nog net acceptabel" etc. Bijvoorbeeld voor het woord *notaris* lijkt de representatie noo-taa-rus ons goed, noo-taa-ris acceptabel en noo-taa-res fout. Wij hebben gehanteerd als qualificaties "acceptabel" en "niet acceptabel": de scheidingslijn tussen beiden is natuurlijk subjectief.

3: In het materiaal komen "woorden" voor waarvan wij de uitspraak niet weten. b.v. *oonk, ochab, enduh.* Bij deze (kleine) groep woorden hebben we interpretaties die voldoende met onze intuïtie overeenstemden acceptabel geacht. (Bij bovengenoemde woorden: oonk, o-sjan, en-du.)

4: In het materiaal komen "woorden" voor, waarvan de uitspraak niet op formele gronden alleen gevonden kan worden. Het betreft hier een aantal afkortingen zoals: *herv, gereƒ, gvav,* en woorden als *belgie* en *deƒile*; trema's en accenten ontbreken in dit materiaal. De fonetische interpretatie van deze woorden is door ons in het algemeen niet acceptabel geacht.

In totaal werd de uitspraak van 768 woorden (waaronder 219 verschillende) niet acceptabel gevonden. Gerekend over het totale aantal woorden is dit ± 2% en over het aantal verschillende woorden ± 6%. Hierin kunnen we de volgende categorieën onderscheiden:

1: Van het totale aantal niet acceptabele woorden wordt 9% veroorzaakt door een onjuiste indeling in lettergrepen. b.v. *verklaard* gesplitst als *verkl-aard*.

2: 16% wordt veroorzaakt door vreemde woorden: b.v. de plaatsnaam *cannes* wordt ka-nus.

3: 24% wordt veroorzaakt door bovengenoemde afkortingen en het ontbreken van trema's en accenten.

4: In de rest (51%) is geen duidelijke indeling te maken; een gedeelte kan misschien nog opgevangen worden door het programma te verbeteren, aan de andere kant lijkt het onmogelijk om zonder gebruik te maken van uitzonderingslijsten een programma te maken dat een goede representatie aflevert voor zowel *tafel* en *kabel* als voor *tabel*.

We geven een aantal voorbeelden uit dit materiaal:

economische	ee-koo-noo-mie-su
internationale	in-tur-naa-tsie-oo-naa-lu
hoogleraar	* hoog-lu-raar
tegenover	* tu-gu-noo-vur
consert	* kon-surt
gepubliceerd	gu-puu-blie-seert
mogelijkheden	moo-gu-luk-hee-dun
mevrouw	* mee-vrouw
militaire	mie-lie-te-ru
verslaggevers	vur-sla-gee-vurs
joernalisten	zjoer-naa-lis-tun
persconferentie	pers-kon-fu-ren-sie
slachtoffers	slac-to-furs
technische	tec-nie-su
prestige	* pres-tu-gu
automatisering	ou-too-maa-tie-zee-ring
concentratiekampen	kon-sun-traa-tsie-kam-pun
details	* dee-tels
dividend	* die-vie-dunt
hebt	hept
liefde	liev-du

nvm	en-vee-em
opdracht	ob-dract
parlementaire	par-lu-men-te-ru
prachtige	prac-tu-gu

Poëzie

De projektgroep "Kwantitatieve benadering van moderne Poëzie" heeft ons programma gebruikt om een uit 13280 woorden bestand materiaal van eksperimentele en niet-eksperimentele poëzie in fonemen te herschrijven (zie [8]). Volgens de door deze projektgroep aangelegde maatstaven waren hiervan 410 woorden (± 3%) verkeerd herschreven. Voor de niet-eksperimentele poëzie geldt een percentage van ± 2 en voor de eksperimentele van ± 4.

Als de oorzaken voor de fouten geven zij aan:

1: Een voor het programma verkeerde indeling in lettergrepen (± 30% van de fouten).

2: Het optreden, vooral in de eksperimentele poëzie, van buitenlandse en dialektische woorden (± 20% van de fouten).

3: Het voorkomen van bepaalde dichterlijke elisies, vooral in de niet-eksperimentele poëzie, waar het programma niet op berekend is (± 6% van de fouten), b.v. *andre* wordt an-dree i.p.v. an-dru.

Voor de overige fouten (44%) is geen duidelijke oorzaak te noemen. Uit de door hen opgestelde lijst van deze restcategorie blijkt dat zij hogere maatstaven hebben aangelegd dan wij bij onze evaluatie. Uit deze lijst geven we een aantal woorden met hun fonetische representatie die wij representatief achten voor het soort fouten dat het programma maakt

bedelaar	* bu-dee-laar
gezel	* gee-zul
herken	* her-kun
hertogin	* her-too-gin
lichaam	* lie-sjaam
onmiskenbaar	* on-mis-kun-baar
verzenschrijver	* vur-zen-schrei-vur
generaal	* gu-nee-raal
losliggend	* los-lu-gunt

E-Legende

De *ee*-klanken zijn in onze taal het sterkst vertegenwoordigd: Jb. van Lennep heeft een "E-legende" geschreven waarin geen andere klinkers voorkomen. We hebben de spelling van dit verhaal enigzins gemoderniseerd: de *sch* is vervangen door *s* in woorden als *heerscher* en *mensch*, de *ee* door *e* in woorden als *breede* en *leeren* en de eigennamen *BERTHE* en *ETHELBERT* hebben we geschreven als *BERTE* en *ETELBERT*. Deze veranderingen maken de tekst voor ons programma eerder moeilijker dan makkelijker. We geven het laatste gedeelte van deze legende en de omgezette versie op de volgende pagina's.

ETELBERT heeft sterke benden vereend:
de veldtekenen geven deze regel te lezen:
"BERTE leve en regere. De wrede WERNER
sterve."
WERNER heeft mede het veld met legers
bedekt. Men velt de speren: men trekt
het welgewet geweer, de degens kletteren.
BERTES held heeft het gevreesde lemmer
geheven en rent den wrevelen WERNER tegen.
Des degens scherpe snede heeft WERNERS
sterken helm gespleten, en deze, neergezegen,
heeft den vegen geest gegeven. Geen der
medgezellen des vreemden heersers wede-
streeft meer den edelen Zweed: enkele sne-
ven, velen vrezen en smeken het leven, de
meesten leggen degens en speren neder.
ETELBERT heeft den zege. BERTES wreker
geeft der weze het erfdeel der Wenden
weder. De tedere BERTE zweert het te
delen met den edelen beschermer: ene
stem, een kreet: "ETELBERT strekke den
Wenden ten Heer en meester. ETELBERT
leve en heerse met BERTE." vereent
edelen en gemeen ter ere des Zweedsen
helds. Hem heffen ze met BERTE ten zetel.
De Deken der Stevenskerk heeft den echt
gezegend. Vete en wrevel hebben gezwegen.
BERTE vergeet het geleden leed: de Hemel
verleent zegen en vrede, en hetgeen de
legende wegens hen vermeldt neemt een

END.

eetelburt heeft sterku bendun vureent.
du velteekunun geevun deezu reegul tu leezun:
bertu leevu en reeguru. du wreedu wernur
stervu.
wernur heeft meedu het velt met leegurs
budekt. men velt du speerun: men trekt
het welguwet guweer, du deeguns kleturun.
bertus helt heeft het guvreezdu lemur
guheevun en rent den wreevulun wernur teegun.
des deeguns scerpu sneedu heeft wernurs
sterkun helm guspleetun, en deezu, neerguzeegun,
heeft den veegun geest gugeevun. geen der
metcuzelun des vreemdun heersurs weedu-
streeft meer den eedulun zweet: enkulu snee-
vun, veelun vreezun en smeekun het leevun, du
meestun legun deeguns en speerun needur.
eetelburt heeft den zeegu. bertus wreekur
geeft der weezu het erfdeel der wendun
weedur. du teeduru bertu zweert het tu
deelun met den eedulun buscermur: eenu
stem, een kreet: eetelburt streku den
wendun ten heer en meestur. eetulburt
leevu en heersu met bertu. vureent
eedulun en gumeen ter eeru des zweetsun
helts. hem hefun zu met bertu ten zeetul.
du deekun der steevunskerk heeft den ect
guzeegunt. veetu en wreevul hebun guzweegun.
bertu vurgeet het guleedun leet. du heemul
vurleent zeegun en vreedu, en hetceen du
leegundu weeguns hen vurmelt neemt een

ent.

Spraaksynthese

Zoals we in 1. al vermeld hebben werken we op het ogenblik met het IPO samen om ons programma aan te passen aan hun apparaat voor spraaksynthese. Deze aanpassingen bestaan uit:

1: Een andere notatie voor de fonemen, waarvan de opvallendste is dat de klinkers die in onze notatie bestaan uit één symbool, nu bestaan uit één symbool gevolgd door een punt.
We geven enkele voorbeelden:

Nederlands	onze notatie	IPO-notatie
goed	goet	(/xoet)
dag	dac	(/da.x)
zang	zan	(/za.q)
sjaal	sjaal	(/caal)
de	du	(/dy.)
put	put	(/pu.t)
piet	piet	(/piit)
zakdoek	zaq-doek	(/za.g/doek)

2: Het tweede verschil is dat de uitspraak minder nadrukkelijk is dan bij onze notatie; d.w.z. *een* wordt altijd uitgesproken als *'n*, en de *n* achter een sjwa valt in de tweede en volgende lettergrepen weg.

een	een	(/y.n)
hebben	he-bun	(/he./by.)

3: We proberen een zinsintonatie aan te geven. Dit doen we als volgt: We hebben in het programma een lijst van hoogfrekwente woorden, die meer tot de structuur dan de betekenis van de zin bijdragen, opgenomen. Alle woorden die niet in deze lijst voorkomen zijn zogenaamde "belangrijke" woorden. Dat een woord belangrijk is wordt weergegeven door een plusje onmiddelijk achter het openingshaakje van dat woord. Een tekst wordt verdeeld in "uitingen", in de gewone schrijftaal van elkaar gescheiden door leestekens, wij laten de laatste lettergreep van iedere uiting voorafgaan door een minteken. Binnen een uiting vindt assimilatie plaats tussen opeenvolgende woorden. Binnen een uiting wordt in het eerste woord van een aaneengesloten

groepje belangrijke woorden de klemtoon aangegeven door het eerste symbool van de beklemtoonde lettergreep te doorbalken.

Deze eerste poging tot koppeling van ons programma aan de spraakmachine was geen groot succes: aan onze output valt nog wel iets te verbeteren bovendien is het stukje op pagina 50 voor ons doel erg moeilijk.
Toch geloven we wel dat het zal lukken een koppeling tot stand te brengen waarbij een redelijk herkenbare uitspraak gegenereerd wordt. Deze koppeling lijkt ons het beste hulpmiddel om ons programma te vervolmaken.

GRONDRECHT

Van alle kanten wordt de laatste tijd geëist dat de betrokkenen een wettelijk grondrecht moeten hebben eigen gegevens in te zien, te corrigeren bij onjuistheden, te weten wie toegang tot de gegevens hebben, schadevergoeding te claimen bij misbruik en verwijdering te vragen van informatie die eigen idealen of intieme gewoonten betreft. Staatssecretaris Van Veen van binnenlandse zaken heeft bij de opening van het rijkscomputercentrum op tien maart in Apeldoorn toegezegd dat in de op stapel staande privacy-wet (die overigens nog wel twee jaar op zich kan laten wachten) bijzondere aandacht aan dit probleem gegeven zal worden. De regering is daar rijkelijk laat mee. De databanken waar persoonsgegevens verwerkt worden schieten als paddestoelen uit de grond. De overheid heeft zelf al een hele serie geautomatiseerde bestanden met persoonsgegevens. De centrale persoonsadministratie (opgehangen aan het persoonsnummer) komt begin volgend jaar tot stand.

(+/ɬro.n/dre.xt)
(/fa.n) (+/ꞩ./ly.) (+/ka.n/ty.) (/wo.rd) (/dy.) (+/ɫaat/sty.) (+/teit)
(+/xy.) (+/eisd) (/da.d) (/dy.) (+/by./ŧro./ky./ny.) (/y.n)
(+/ʋe./ty./ly.k) (+/xro.n/dre.xt) (+/moe/ty.) (/he./by.) (/ei/xy.)
(+/xy./ɬee/vy.ns) (/i.n) (/ty.) (+/ŧziin) (/ty.) (+/ɬo+/rii/ɬee/ry.)
(/bei) (+/ɸ.n/juis/tee/-dy.) (/ty.) (+/ʋee/ty.) (/wii) (+/ŧoe/xa.q)
(/to.d) (/dy.) (+/xy./ɬee/vy.ns) (/he./-by.) (+/ʂxaa/dee/vy.r/xoe/di.q)
(/ty.) (+/ɬle./my.) (/bei) (+/ɱi.z/bruik) (/e.n) (+/vy.r/ʋei/dy./ri.q)
(/ty.) (+/ʋraa/xy.) (/va.n) (+/ɫ.n/fo.r/ɱaa/tsii) (+/dii) (/ei/xy.)
(+/ɫi/dy./aa/ly.) (/o.f) (+/ɫ.n/sii/my.) (+/xy./woon/ty.) (+/by./-tre.ft)
(+/staat/sy./kree/taa/ri.s) (/fa.n) (+/ʋeen) (/va.n)
(+/ƀi./ny./la.nt/sy.) (+/zaa/ky.) (/heefd) (/bei) (/dy.) (+/ɸo/py./ni.q)
(/va.n) (/he.t) (+/ŧeiks/ko.m/puu/ty.r/sy./tru.m) (/o.p) (+/ŧiin)
(+/maart) (/i.n) (+/ꞩa/py.l/doorn) (+/toe/xy./ze.xd) (/da.t) (/i.n)
(/dy.) (/o.p) (+/ʂtaa/py.l) (+/staan/dy.) (+/prii/vaa/sii) (+/we.d)
(+/dii) (+/oo/vy.r/y./xy.ns) (/no.x) (+/ʋe.l) (+/twee) (+/jaar) (/o.p)
(/si.x) (/ka.n) (+/ɫaa/ty.) (+/wa.x/ty.) (+/bei/zo.n/dy./ry.)
(+/aan/da.xt) (/aan) (+/ɗi.t) (+/proo/bleem) (+/xy./xee/vy.) (/za.l)
(/wo.r/-dy.) (/dy.) (+/ry./xee/ri.q) (/i.z) (/daar) (+/ŧei/ky./ly.k)
(+/laat) (/-mee) (/dy.) (+/ɗaa/taa/ba.q/ky.) (/waar)
(+/ƥe.r/soons/xy./xee/vy.ns) (+/fy.r/we.rkt) (/wo.r/dy.) (+/ʂxii/ty.)
(/a.ls) (+/ƥa./dy.s/toe/ly.) (+/uid) (/dy.) (+/ŧxro.nt) (/dy.)
(+/ɸo/vy.r/heit) (/heeft) (/se.lf) (+/ꞩ.l) (/y.n) (+/ɦee/ly.)
(+/see/rii) (+/xoo/too/maa/tii/zeer/dy.) (+/by./sta.n/dy.) (/me.t)
(+/ƥe.r/soons/xy./xee/-vy.ns) (/dy.) (+/ʂe.n/traa/ly.)
(+/pe.r/soon/sa.t/mii/nii/straa/tsii) (+/o.p/xy./ha.q/y.) (/aan) (/he.t)
(+/ƥe.r/soons/nu./my.r) (+/ko.md) (+/by./xi.n) (+/vo.l/xy.nt) (+/jaar)
(/to.t) (+/ŧsta.nt)

6. Referenties

[1] Woordenlijst van de Nederlandse Taal, staatsdrukkerij- en uitgeverijbedrijf Martinus Nijhoff, 's-Gravenhage, 1954.

[2] J.W. Backus, The syntax and semantics of the proposed international algebraic language of the Zürich ACM-GAMM conference. ICIP, June 1959.

[3] H. Brandt Corstius, Exercises in Computational Linguistics, MC Tract 30, Mathematisch Centrum, Amsterdam, 1970.

[4] Principles of the International Phonetic Association etc. London, 1949.

[5] J.W. Backus e.a., Revised Report on the Algorithmic Language ALGOL 60, Regnecentralen Copenhagen, 1964.

[6] Cursus ALGOL 60, Electronisch Rekencentrum van de Rijksuniversiteit Utrecht, 1967.

[7] J.A.Th.M. van Berckel e.a., Formal Properties of Newspaper Dutch, MC Tract 12, Mathematisch Centrum, Amsterdam, 1965.

[8] Eindverslag van de Projektgroep kwantitatieve Benadering van moderne Poëzie, Instituut Nederlands, Nijmegen, 1971.