S Y N T A X T E S T I.

(A Deterministic Contextfree Grammar Parser)

&

S Y N T A X T E S T II.

(A Nondeterministic Transition Network Parser)

Luc Steels

UNIVERSITEIT ANTWERPEN (U.I.A.)

Dept. Germ. Fil., Afd. Nederlandse Taalkunde.

Technisch Rapport. LS/1.

december 1974

0. Inleiding

Deze paper beschrijft resultaten van onderzoek op gebied van automatische syntaktische analyse. De bedoeling van het onderzoek was het programmeren van een neutrale processor die een grammatika G accepteert, daarna een zin uit L(G) als input aanneemt en tenslotte een strukturele deskriptie, in de vorm van een string met labelled bracketings, voor de ingegeven zin produceert als output.
De processor zou hulp kunnen bieden bij de ontwikkeling van syntaktische modellen, doordat experimenten kunnen gedaan worden die aan alle eisen dienaangaande, zoals o.m. herhaalbaarheid, expliciëtheid, kontrole van alle variabelen, enz...,voldoen. Daarnaast kan het systeem ook beschouwd worden als een onderdeel van een komplexer model dat taal verwerkt.

De automatische syntaktische analyse werd als probleem gesteld rond '60. Sindsdien zijn er verscheidene systemen operationeel waarvan de belangrijkste voor natuurlijke taal de predictive analyzer, gebaseerd op een phrase structure grammatika (zie Oettinger (1961) en Kuno (1966)), de parsingsystemen met een transformationele grammatika als basis (zie bijv. Petrick (1965) en (1973))en parsers met een augmented transition network grammatika (zie Woods (1970)). Neutrale processors werden geïntroduceerd door Kay en Kaplan, met als eerste realisatie in deze richting de G.S.P. (general syntactic processor).(cfr. Kay (1974), Kaplan (1973)).

We beschrijven in dit rapport twee generaties van een parsingsysteem
(a) een deterministische parser op basis van een kontekstvrije grammatika
(b) een nondeterministische parser op basis van een rekursief transitienetwerk.

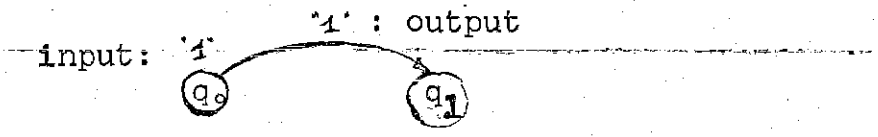
Bij wijze van inleiding gaan we in op de finite state machines die zoals bekend aan de basis liggen van de ATN-grammatika's.

Dit rapport handelt hoofdzakelijk over de technische aspekten van het systeem dat operationeel beschikbaar is in de programmeertaal BASIC. (We hopen in de nabije toekomst een FORTRAN-versie te realiseren). Voor metodologische kwesties omtrent computersimulatie van modellen, zie Harbordt (1974). We veronderstellen dat de lezer bekend is met de teorie van de formele talen (zie hiervoor bv. Levelt(1973),Salomaa (1973)) en dat programma's in de programmeertaal BASIC kunnen gelezen worden (zie hiervoor Manual (1973),Peluso (1972),Lindahl (1971)).

1. Het uitgangspunt: de Finite State Machines.

Een proces (óf een machine) stelt men wel eens voor als een doos (black box) waar dingen ingaan, de input, en waar even later dingen uitkomen, de output. Verder analyseert men de relatie tussen input en output in termen van onderscheiden toestanden (states). Het innemen van een toestand noemt men een transitie: de machine gaat van de ene toestand over in de andere. Een transitie gebeurt in funktie van een input. De toestand van waaruit men vertrekt wordt bepaald door de vroegere aktiviteiten.

Vb. 1. 1. Stellen we een toestand q_0; bij een input "1" wordt (i) een transitie uitgevoerd van toestand q_0 naar q_1 en (ii) een output gegeven, nl. "1".



Welke transities er gebeuren en wanneer wordt aangegeven in een tabel ofwel in een zogenaamde state-diagram (ook state transition diagram).

We bespreken nu twee voorbeelden van finite state machines. Het eerste is een binaire optelmachine (cfr. Minsky, 1967,23) het tweede een eenvoudige taalkorrektor.

Vb. 1. 2. Een binaire optelmachine.

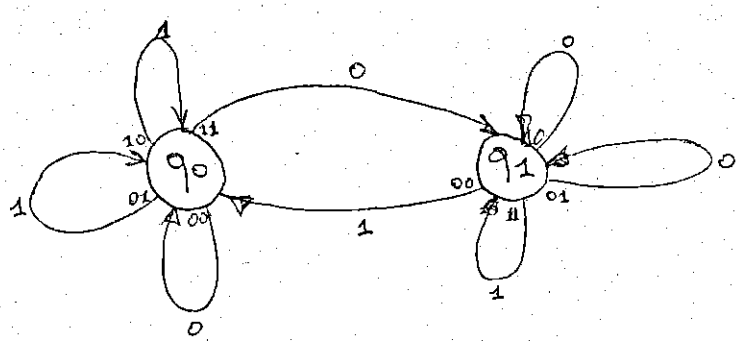
Deze machine kan twee getallen bij elkaar optellen wanneer zij in binaire kode worden ingegeven. De getallen worden van rechts naar links ingegeven en iedere keer vertikaal opgeteld, vandaar dat er telkens twee elementen worden beschouwd als inputsignaal. De funktie G beschrijft welke nieuwe toestand wordt ingenomen gegeven een bepaald inputsignaal en een toestand, en de funktie F beschrijft de output die hoort bij een transitie.

De transitietabel:

G	q_0	ql
00	q_0	q_0
01	q_0	ql
10	q_0	ql
11	ql	ql

F	q_0	ql
00	0	1
01	1	0
10	1	0
11	0	1

De state transition diagram:



Men kan het systeem gemakkelijk begrijpen door state q_o te beschouwen als een toestand waarin niets wordt 'onthouden' en q1 als een toestand waarin een 1 wordt overgebracht naar de volgende stand.
Even een voorbeeld: 8 + 10 = 18, binair:

```
  1000
+ 1010
------
 10010
```

tijd	0	1	2	3	4	5	6
8	0	0	0	1	0	0	0
10	0	1	0	1	0	0	0
signaal	00	01	00	11	00	00	00
toestand	q_o	q_o	q_o	q_o	q1	q_o	q_o
output	-	0	1	0	0	1	0

(Merk op dat de getallen in omgekeerde volgorde worden opgeschreven.)

De lezer wordt aangeraden dit voorbeeld goed door te werken en eventueel zelf een andere optelling te proberen. De tabel kan men als volgt lezen: neem een element van het eerste getal (vb. 0) neem een element van het tweede (0) voeg ze tesamen tot een signaal (00) vergelijk met de toestand op het moment (vb q_o), bereken via G de volgende toestand (q_o) en bereken via F de output (0).

Vb. 1. 2. Een eenvoudige taalkorrektor.

Nu een meer linguïstische toepassing: een eenvoudige taalkorrektor die de zinnen "de { hond / poes } loopt zacht" accepteert en alle andere kombinaties met dit alfabet verwerpt. Wanneer een woord geaccepteerd wordt, geeft het systeem dit woord zelf weer, wanneer niet, komt er als output "-" en kan men opnieuw een (nu verbeterde)input geven.

De transitietabellen:

F	Q0	Q1	Q2	Q3
DE	Q1	Q1	Q2	Q3
HOND	Q0	Q2	Q2	Q3
LOOPT	Q0	Q1	Q3	Q3
ZACHT	Q0	Q1	Q2	Q0
POES	Q0	Q2	Q2	Q3

G	Q0	Q1	Q2	Q3
DE	DE	-	-	-
HOND	-	HOND	-	-
POES	-	POES	-	-
LOOPT	-	-	LOOPT	-
ZACHT	-	-	-	ZACHT

Even een tabel van de gang van zaken:

tijd	0	1	2	3	4	5	6
INPUT	DE	LOOPT	POES	LOOPT	HOND	ZACHT	
TOESTAND	Q0	Q1	Q1	Q2	Q3	Q3	Q0
OUTPUT	-	DE	-	POES	LOOPT	-	ZACHT

De toepassing van Finite State Machines op natuurlijke taal is eigenlijk nooit van de grond gekomen. Dit is te danken aan Chomsky die het model verwierp op basis van teoretische en praktische gronden (cfr. Chomsky, 1957,18-25). Hij stelde hier tegenover de transformationeel generatieve teorie die voldoende bekend is om hier voorgesteld te moeten worden. Nochtans is het mogelijk om een F.S.machine verder uit te bouwen zodanig zelfs dat de nieuwe machine ekwivalent is met een kontekstvrije grammatika, die toch de basis vormt van het alternatief dat door Chomsky werd ontwikkeld. Deze uitbouw werd voorgesteld door Woods (1970) en het resulterende systeem heet een rekursief transitienetwerk. We komen hier later op terug.

2. SYNTAXTEST 1. Een deterministische parser op basis van een kontekstvrije grammatika.

0. We zullen in een eerste faze een processor ontwerpen en programmeren die
   (i) als input een kontekstvrije grammatika accepteert, dwz een axiomatisch systeem met een axioma, primitieve elementen, afleidingsregels en interpretatieregels.
   (ii) deze grammatika omzet in gekodeerde informatie,
   (iii) als input proefzinnen, bedoeld om de grammatika te testen, accepteert,
   (iv) voor deze proefzinnen een strukturele deskriptie in de vorm van labelled bracketings konstrueert volgens de richtlijnen van de ingegeven grammatika.

Simulatie van een model is pas mogelijk als het expliciet gedefiniëerd is. Het eerste waarover we absolute klaarheid moeten hebben is de representatie ('format') van de input. Daarna gaan we in op de kontrolestruktuur en de algoritmen van het programma. De technische details zijn te vinden in appendix I en II.

Laten we echter vooraf even de werking van het systeem bekijken aan de hand van de output. (cfr fig 1)

```
L.STEELS 19:10.74  TAPE 21 FILE 5
SYNTAXTEST.I (DET.CFG-PARSER)
```

1.SYNTAXIS

```
NONTERMINALS:SENPVPPP

TERMINALS:DTNOVEPR

V(VERZ. SYMBOLEN)=AXSENPVPPPDTNOVEPR

AANTAL NONTERMINALS: 5

PRODUKTIES:
AX -> SE
SE -> NP VP PP
NP -> DT NO
PP -> PR NP
VP -> VE NP
↑

SYNTAXMATRIX:
 22    0     0     0     0
 23    24    25    0     0
 6     7     0     0     0
 9     23    0     0     0
 8     23    0     0     0
```

2.LEXIKON

```
DT -> DEHETEENGEENZIJN
NO -> JONGENPOESMUISKAMERLIEDJEHUIS
VE -> VERJOEGZINGTSPEELT
PR -> METUITOP
↑
LEXIKON:
DEHETEENGEENZIJNJONGENPOESMUISKAMERLIEDJEHUISVERJOEGZINGTSPEELTMETUITOP

INFORMATIEMATRIX:
 1     2     3     5     4     1     17    46    64
 1     2     3     5     4     17    46    64    72
```

3.TEST

```
INPUT: DE JONGEN ZINGT EEN LIEDJE OP ZIJN KAMER

SYNT.ANAL. :
(SE(NP(DT DE)(NO JONGEN))(VP(VE ZINGT)(NP(DT EEN)(NO LIEDJE)))(PP(PR OP)
(NP(DT ZIJN)(NO KAMER))))


INPUT: DE POES VERJOEG EEN MUIS UIT HET HUIS

SYNT.ANAL. :
(SE(NP(DT DE)(NO POES))(VP(VE VERJOEG)(NP(DT EEN)(NO MUIS)))(PP(PR UIT)(
NP(DT HET)(NO HUIS))))
```


Zoals men kan zien zijn er drie onderdelen:
(i) Syntaxis, waarin nonterminale en terminale symbolen, het axioma en de produkties worden ingegeven. De verzameling symbolen wordt aan elkaar geschreven, vb. SENPVP is SE, NP, VP. Het axioma wordt expliciet als een produktie ingegeven. De syntaktische informatie wordt verwerkt en opgeslagen in een syntaxmatrix.

(ii) Lexikon: In deze faze kan men de interpretaties ingeven voor de nonterminale symbolen. Deze informatie wordt ook verwerkt en opgeslagen in een informatiematrix (die ook voor andere dingen dienst doet).
Belangrijke opmerking: Wanneer we spreken over terminale symbolen, bedoelen we steeds de preterminale symbolen. De interpretatie van deze preterminale is een systeem dat verschilt van de syntaxis. Steeds zullen we andere algoritmen nodig hebben om de lexikale interpretatie te organiseren.
(iii) Test: Tenslotte kunnen proefzinnen ingevoerd worden waarna een syntaktische analyse volgt.
Om duidelijk te laten zien dat de gebruiker zelf een grammatika kan ingeven, geven we nog een voorbeeld, nu van een strikt formele taal.

```
1.SYNTAXIS

NONTERMINALS:A1A2A3A4A5

TERMINALS:B1B2B3B4B5B6

V(VERZ. SYMBOLEN)=AXA1A2A3A4A5B1B2B3B4B5B6

AANTAL NONTERMINALS: 6

PRODUKTIES:
AX -> A1
A1 -> B1 A2 B1
A2 -> B2 A3 B2
A3 -> B3 A4 B3
A4 -> B4 A5 B4
A5 -> B5 B6 B5
↑

SYNTAXMATRIX:
 22    0    0    0    0
  7   23    7    0    0
  8   24    8    0    0
  9   25    9    0    0
 10   26   10    0    0
 11   12   11    0    0

2.LEXIKON

B1 -> ****
B2 -> $$
B3 -> ##%%
B4 -> &&
B5 -> (<)>
B6 -> 000
↑
LEXIKON:****$$##%%&&(<)>000

INFORMATIEMATRIX:
 1    2    3    4    5    6    1    5    7   11   13   17
 1    2    3    4    5    6    5    7   11   13   17   20

3.TEST

INPUT: ** $$ %% && (< 000 )> && %% $$ **

SYNT.ANAL. :
(A1(B1 **)(A2(B2 $$)(A3(B3 %%)(A4(B4 &&)(A5(B5 (<)(B6 000)(B5 )>))(B4 &&
))(B3 %%))(B2 $$))(B1 **))
```

2.1. Representatie

(a) Input

Het is voldoende bekend dat een formele grammatika een quadrupel is G = ⟨ Vn,Vt,P,A ⟩, waarbij Vn de verzameling nonterminale symbolen, Vt de verzameling terminale symbolen, P de produkties en A het axioma. Een kontekstvrije grammatika in het bijzonder heeft produkties van de vorm A → α waar A ∈ Vn, α ∈ (Vn ∪ Vt)*
We zullen dus vier stappen hebben bij de input van een grammatika:
(i) Geef de nonterminale symbolen in
(ii) Geef de terminale symbolen in
(iii) Geef de produkties in
(iv) Geef het axioma in.
Volgende problemen stellen zich nu:
(a) De gebruiker van het systeem heeft een alfanumerische kode, (vb. VP, NP, e.d.) het is echter handiger voor de machine om numerische kode te verwerken. We noemen de alfanumerische kode de gebruikerskode en de numerische de systeemkode. De verzameling symbolen Vn ∪ Vt zal door een programma moeten gekodeerd worden. De systeemkode moet aan de volgende voorwaarden voldoen:
1. De kode van de gebruiker staat in funktie van de systeemkode zodat herkodering langs de twee kanten moet mogelijk zijn
2. Het kodesysteem moet relatief zijn, dwz elke gebruiker geeft een andere kode in; het systeem moet in staat zijn al deze kodes te verwerken.
(b) De werking van het eigenlijke ontledingssysteem zou erg tijdrovend worden indien de produkties enkel beschikbaar zouden zijn zoals ze werden ingegeven. Daarom zal de syntaktische informatie op een bijzondere wijze tijdens de ontleding aanwezig zijn in het geheugen.
We bespreken nu de oplossingen die voor beide problemen werden gevolgd.

Door de machine zelf wordt een kode gegenereerd die elk symbool uit het alfabet een numerische kode geeft in rangorde van binnenkomst in het systeem. Daarna worden twee matrices ontwikkeld S en I. Op S staan de produkties en op I van waar tot waar de produkties gaan. Nonterminale symbolen krijgen een bijzondere kode namelijk "2". S wordt min of meer genormaliseerd door 0 toe te voegen op lege plaatsen.

Even een voorbeeld:(2.1.)
Zij het axioma: SE
andere nonterminale symbolen: NP,VP
terminale symbolen: dt,no,ve.

Kode (in rangorde van binnenkomst) Ax = 1 (wordt voorzien 'by default'), SE = 2,NP = 3,VP = 4;DT = 5, NO = 6, VE = 7.

De produkties:	in kode:
AX → SE	1 → 2
SE → NP VP	2 → 3 4
NP → DT NO	3 → 5 6
VP → VE NP	4 → 7 3

De syntaxmatrix:

S	1	2
1	2	
2	3	4
3	5	6
4	7	3

De informatiematrix:

I	1	2	3	4
1	1	2	3	4
2	1	2	3	4

De syntaxmatrix met kode "2" voor nonterminale symbolen:

S	1	2
1	22	
2	23	24
3	5	6
4	7	23

De genormaliseerde syntaxmatrix:

S	1	2	3	4	5
1	22	0	0	0	0
2	23	24	0	0	0
3	5	6	0	0	0
4	7	23	0	0	0

De lexikonregels verschillen op geen enkele wijze van de PS-regels. Ze worden wel afzonderlijk opgeslagen.

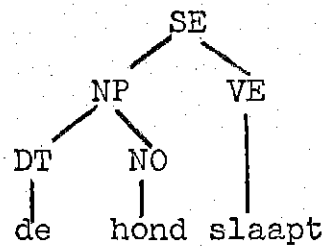
Details van deze kodering zijn te vinden in Appendix I, waar we ook het algoritme en een koderingsprogramma beschrijven.

(b) Output

De strukturele deskriptie (het resultaat van de ontleding met een kontekstvrije grammatika) wordt uitgedrukt in een boomdiagram of in een string met labelled bracketings. Omdat de definitie of beter het algoritme van boomdiagrammen veel ingewikkelder is dan dat van labelled bracketings zullen we labelled bracketings als output produceren. Deze worden als volgt gedefiniëerd (cfr Brainerd, 1971,212):

> Wanneer een regel A -> α wordt toegepast op een string $\gamma = \chi_1 A \chi_2$ druk het resultaat uit als $\chi_1(A\ \alpha)\chi_2$ waarbij (A α) een 'bracketing' van α 'labelled' door A betekent.

Merk op dat we de linkerhaakjes benoemen en dat we het teken dat benoemt niet onder de lijn maar gewoon op de lijn schrijven.
De boomdiagram



komt overeen met de string:

(SE (NP (DT de)(NO hond)) (VE slaapt))

De elementen tussen haakjes kunnen beschouwd worden als lijsten (lists) en zijn aldus een direkte representatie van boomdiagrammen. (Nota: in feite is het zelfs zo dat deze string geen string met labelled bracketings is maar een representatie van een boomdiagram. Zie voor de teoretische achtergronden van representaties en de datastruktuur bij implementatie Steels (1974 c).)

2.2. De kontrolestruktuur en het algoritme.

Een kontrolestruktuur is de manier waarop een programma georganiseerd wordt. In dit geval is dit een FS-machine die als volgt wordt gedefiniëerd.

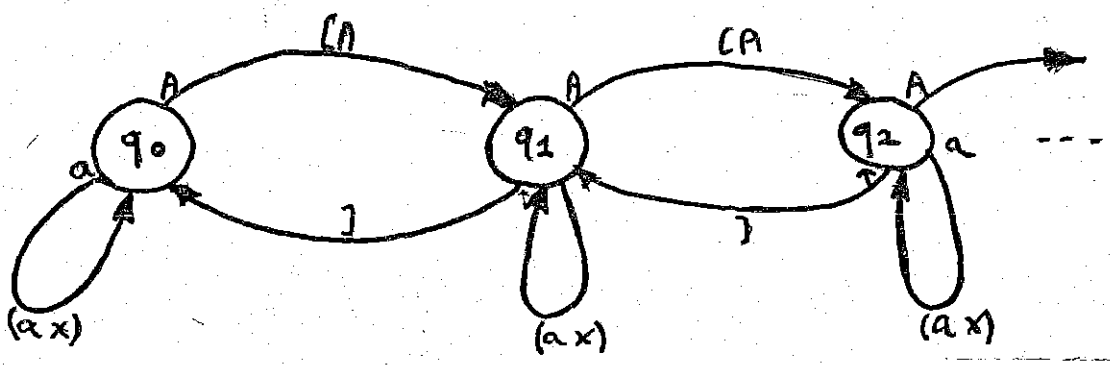
F	Q0	Q1	Q2	Q3	Q4	....
Vn	Q1	Q2	Q3	Q4	Q5	....
↑	Q0	Q0	Q1	Q2	Q3	...
Vt	Q0	Q1	Q2	Q3	Q4	...

Merk op dat er eigenlijk teoretisch oneindig veel standen zijn, doch dat het in realiteit altijd een eindig aantal is.
We kunnen ook direkt de output geven zodat op het einde de string in labelled bracketings verschijnt:

G	Q0	Q1	Q2	Q3	Q4	....
A ε Vn	(A	(A	(A	(A	(A	...
" ↑ "	)	)	)	)	)	...
a ε Vt	(a x)	(a x)	(á x)	(a x)	(a x)	...

(x is het element uit de inputstring, "↑" is een kode die het einde van een nonterminaal symbool aanduidt).

De state-transition diagram



Telkens als er dus een nonterminaal verschijnt, schuiven we naar rechts op en als er een nonterminaal volledig verwerkt is, komen we terug. Als er een terminaal als input gegeven is, blijven we op dezelfde toestand staan.

Wat nu moet geregeld worden is welk symbool wanneer mag komen. Dit wil zeggen, de input is niet 'een' terminaal of nonterminaal symbool maar is beperkt tot dat bepaalde terminaal of nonterminaal symbool dat is voorgeschreven door de grammatika.
Dit symbool kunnen we vinden in de ingegeven grammatika. We lezen m.n. die regel van de grammatika die wordt aangeduid door het nonterminaal symbool waar we mee bezig zijn (dit is bij de begintoestand het axioma) in de syntaxmatrix. Het algoritme gaat dan zo:

Overloop de regels (= rijen van de syntaxmatrix) van het nonterminaal symbool waar je mee bezig bent

(i) zijn er geen elementen meer rechts die nog kunnen verwerkt worden (m.a.w. is het gelezen element 0), ga dan terug waar je bezig was voor je aan dit nonterminaal symbool begon.
(ii) is het gelezen element een nonterminaal symbool, onthoudt de regel en waar je bezig was op de regel (= rij en kolom van S) en begin het algoritme opnieuw. We spreken af van het einde van een nonterminaal symbool met een pijl aan te geven.
(iii) is het gelezen element een terminaal symbool, vergelijk dan of het woord dat uit de zin die op dat moment onderzocht wordt, behoort tot de syntaktische klasse van dit terminaal symbool aan de hand van interpretatieregels. Is dit het geval, zet een stap vooruit in uw inputstring en in uw matrix. Is dit niet het geval, neem de volgende rij.

De lezer wordt aangeraden een voorbeeld op te lossen (bijv dat in fig 1.) om de werking van het algoritme goed te begrijpen. Men ziet ook in dat het algoritme rekursief is opgebouwd en dat de 'parsingmode' topdown is. Dit heeft als interessant gevolg dat lexikale ambiguïteit (dezelfde woordvorm behoort tot twee verschillende syntaktische klassen) kan opgelost worden.

Zie voor de technische details en de programmering van het algoritme Appendix II.

2.3. Bespreking.

(i) In dit eerste deel werd een deterministische processor gekonstrueerd die als input een kontekstvrije grammatika aanneemt, en daarna de mogelijkheid geeft deze grammatika te testen aan de hand van proefzinnen.
(ii) Hoe elegant het vorige algoritme ook is, het heeft ernstige restrikties en ze zullen moeten ondervangen worden om tot resultaten te kunnen komen. De zwaarste restriktie is dat het een DETERMINISTISCHE processor is, terwijl de analyse van taal essentieel een non-deterministisch proces is: niet alleen kunnen zinnen verschillende strukturele deskripties hebben, bepaalde symbolen hebben ook verschillende herschrijfmogelijkheden.
De volgende 'generatie' van onze parsingsystemen komt juist tegemoet aan het nondeterministische aspekt van taalanalyse.
(iii) Bovendien zit er een kontradiktie in de systeembouw. We maken gebruik van een finite state machine als kontrolestruktuur en aan de andere kant hebben we als input een kontekstvrije grammatika. Wie op de hoogte is van de verhouding tussen abstrakte automaten meer bepaald de ekwivalentie van F.S.-machines met een type 3 p.s.-grammatika, en de argumentatie van Chomsky tegen de toepassing van F.S.-machines in de produktie van taal (zie inleiding), zal allicht om enige opheldering hieromtrent verlangen. Deze opheldering komt in het volgende hoofdstuk.

3. SYNTAXTEST II. Een nondeterministische parser op basis van een kontekstvrije grammatika.

3.0. Inleiding.

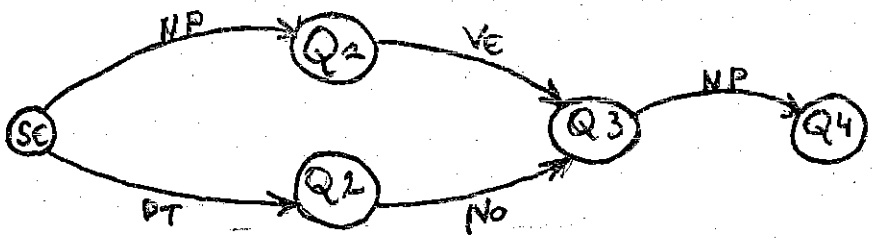
3.0.1. Van een kontekstvrije grammatika naar een rekursieve transitie-netwerk grammatika.

Finite state machines verschillen van kontekstvrije grammatika's vooral op het vlak van de rekursiviteit. We kunnen finite state machines echter op een eenvoudige wijze rekursief maken door als input niet alleen terminale elementen maar ook nonterminale toe te laten.

VB. 3.1. Het eenvoudige P.S.-regelsysteem:

S → NP VP NP
S → VE NP VE
NP → DT NO

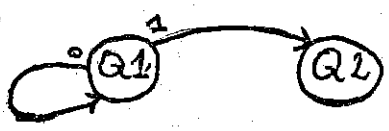
ziet er in een state diagram als volgt uit:



Deze machine verwerkt zinnen zoals 'de man speelde de piano', 'bespeelde de man de piano', enz...
We zien dat er reeds bij de eerste transitie een nonterminaal is verondersteld (nl. NP). Op dit punt onthouden we waar we bezig zijn in SE en gaan naar een andere diagram (een soort subroutine) voor NP. Daar werken we verder tot het netwerk is doorlopen en gaan dan terug.

Een transitienetwerk zal voorgesteld worden in een matrix T met 3 kolommen. In de eerste kolom staat de input van een transitie, in de tweede kolom staat de eerste toestand en in de kolom de toestand die bereikt wordt na de transitie. Volgende kode wordt bovendien aangenomen: de verschillende toestanden duiden we aan met het symbool A/n, waarbij A ∈ V en n het nummer van de toestand. De eerste toestand van een nonterminaal symbool wordt aangeduid met "1", de laatste met "↑".(POP in ATN-jargon), Als A ∈ Vt dan is n = 0.

VB. 3.2.



wordt

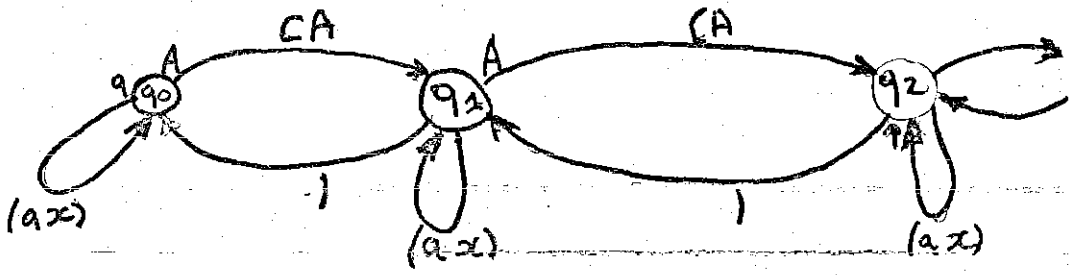
input	1e toest.	2e toest
1/0	Q/1	Q/2
0/0	Q/1	Q/1

Meer voorbeelden volgen later.

Nota: De representatie van het lexikon blijft hetzelfde als voor het vorige systeem. We zullen hier niet verder op ingaan.

3.0.2. Van een deterministische processor naar een nondeterministische.

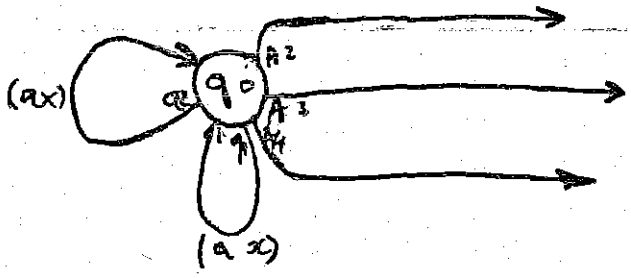
De kontrolestruktuur voor het ontleden van de zinnen met een deterministische processor zag er als volgt uit:



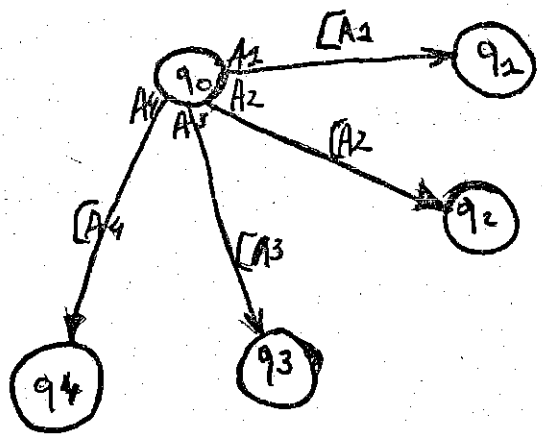
Herinner U dat A ε Vn, a ε Vt en x een woord uit de inputstring is.

We willen nu twee extensies:

(i) Er vertrekt vanuit één toestand meer dan 1 pijl:



(ii) de pijlen komen in verschillende plaatsen toe:



Bekijken we nu even aandachtig Voorbeeld 3. 3.:

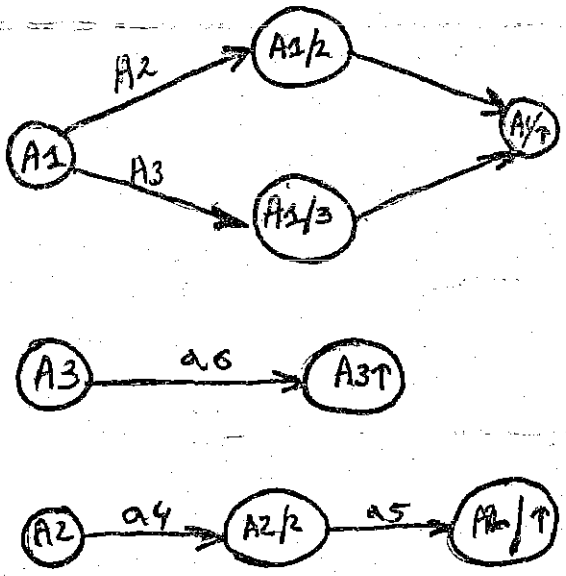
De herschrijfregels:

A1 → A2 A3	(i)
A1 → A3 A2	(ii)
A2 → a4 a5	(iii)
A3 → a6	(iv)

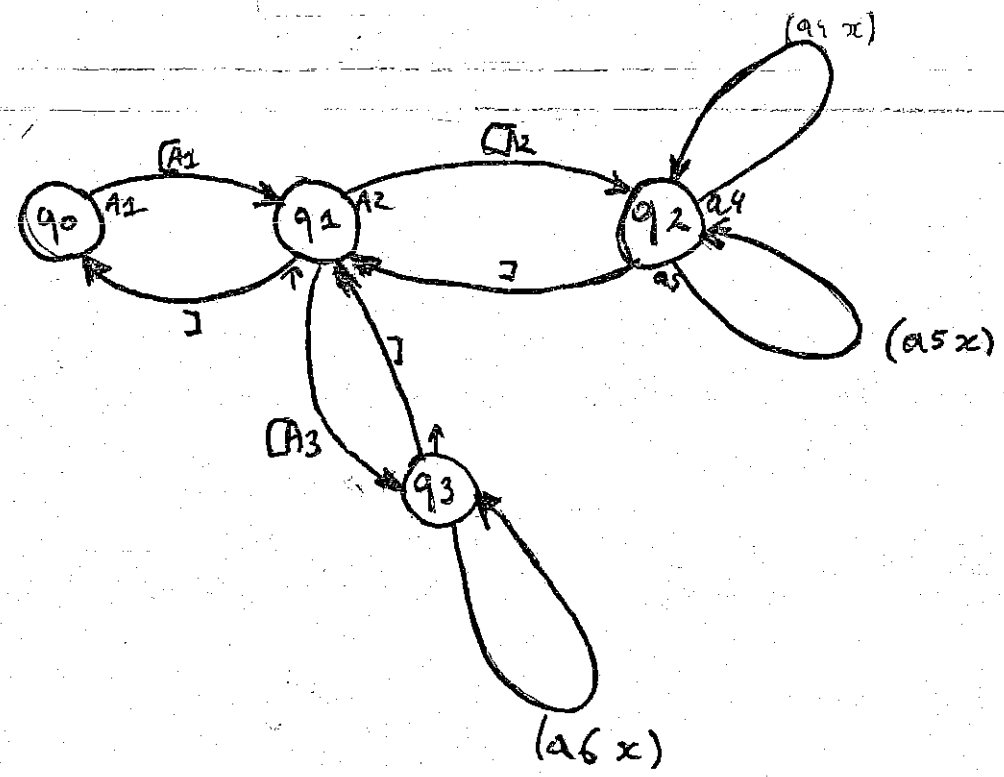
De interpretaties

a6 → 1,2	
a4 → 1,3	
a5 → 2,3	

In een transitienetwerk:



Gegeven de ambigue zin "132". De kontrolestruktuur van het algoritme bij de verwerking neemt de volgende vorm aan:



De strukturele deskriptie in labelled bracketings:

(a) (A1 (A2 (a4 1) (a5 3)) (A3 (a6 2)))
(b) (A1 (A3 (a6 1))(A2 (a4 3) (a5 2)))

De kontrolestruktuur in tabelvorm:

	input	toest 1.	toest 2.	output
1.	A1	q0	q1	(A1
2.	A2	q1	q2	(A2
3.	A3	q1	q3	(A3
4.	↑	q1	q0	)
5.	a4	q2	q2	(a4 x)
6.	a5	q2	q2	(a5 x)
7.	↑	q2	q1	)
8.	a6	q3	q3	(a6 x)
9.	↑	q3	q1	)
10.	↑	q1	q0	)

De grammatika in tabelvorm

	input	toest 1	toest 2
1.	A2/1	A1/1	A1/2
2.	A3/1	A1/2	A1/↑
3.	A3/1	A1/1	A1/3
4.	A2/1	A1/3	A1/↑
5.	a6/0	A3/1	A3/↑
6.	a4/0	A2/1	A2/↑
7.	a5/0	A2/2	A2/↑

De tabellen van de analyses:

input	A1	A2	a4	a5	↑	A3	a6	↑	↑	-
toest	q0	q1	q2	q2	q2	q1	q3	q3	q1	q0
output	-	(A1	(A2	(a4)	(a5)	)	(A3	(a6)	)	)

input	A1	A3	a6	↑	A2	a4	a5	↑	↑	-
toest	q0	q1	q3	q3	q1	q2	q2	q2	q1	q0
output	-	(A1	(A3	(a6)	)	(A2	(a4)	(a5)	)	)

Het probleem is nu hoe we met de kontrolestruktuur en de grammatika als input een systeem kunnen realiseren dat als output de bovenstaande tabel produceert.

De oplossing die zal voorgesteld worden is een vereenvoudigde en gewijzigde versie van de implementatie die door Woods, et.al. werd ontwikkeld (cfr Woods (1972),Woods (1973)) en is ook verwant aan de G.S.P.-systemen door Kay en Kaplan ontworpen (cfr Kay (1973) Kay(1974) en Kaplan (1973)).

3.1. Syntaxtest II, een informele beschrijving.

In plaats van onmiddellijk te werken op de syntaxmatrix zoals in het vorige algoritme, introduceren we een 'tussenmatrix' (kladblok) Hierop staan 'taken' of 'konfiguraties' die in toevallige orde kunnen uitgevoerd worden. De uitvoering van een taak impliceert dat er een nieuwe verschijnt, enzovoort. We leggen eerst even uit hoe de taken eruit zien en dan hoe de algoritmen die de taken uitvoeren, werken.

(i) De taken.

De taken zijn vektoren met zes elementen : $\langle \alpha_1, \alpha_2, \alpha_3, \alpha_4, \alpha_5, \alpha_6 \rangle$

(a) α_1 is het nummer van het woord dat onderzocht moet worden bij het uitvoeren van de taak
(b) α_2 is de input van de transitie die op het moment van de uitvoering van de taak wordt beschouwd.
(c) α_3 geeft het kodenummer van het element in α_2.
(d) α_4 is het taaknummer van de taak die heeft aanleiding gegeven tot de aktuele taak.
(e) α_5 is de regel (uit de grammatika) die aanleiding gaf tot deze konfiguratie.
(f) α_6 is de konfiguratie waar een inbedding begon (dwz waar de eerste toestand van een nonterminaal symbool begon).

Nota: De taken worden opgeslagen in een matrix (de kladblok) met 6 kolommen en zoveel rijen als men noodzakelijk acht.

(ii) De algoritmen

Er is vooreerst een kontroleorgaan nodig dat de akties koördineert en vooruitstuwt. Dit is PARSER. PARSER begint met het opvullen van de kladblok met de begintaak die in α_2 het axioma bevat, in α_3 de kode voor α_2 , en in α_1 het eerste woord. Voor $i > 3$ is α_i onbepaald.
Dan wordt er een algoritme namelijk STEP opgeroepen die de taken uitvoert. Als er geen taken meer zijn, onderzoekt PARSER of er goede parsings gevonden zijn en geeft hiervan een output.

STEP heeft als input een konfiguratie en als resultaat de mogelijke taken die uit deze konfiguratie kunnen afgeleid worden. Het ideale parsingsysteem doorzoekt de lijst van de mogelijke konfiguraties op een intelligente manier. Doch we zullen al beginnen met de eenvoudigste oplossing: alle konfiguraties bekijken die er mogelijk zijn. Hierbij zijn er 3 mogelijkheden: (cfr. het deterministische systeem in vorig hoofdstuk):

(i) het element in α_2 is een nonterminaal symbool, dan wordt een ander algoritme opgeroepen (NEWTASK I) dat nieuwe taken creeërt volgens de regels van de grammatika.
(ii) het element in α_2 is een terminaal symbool, dan wordt het algoritme NEWTASK II opgeroepen dat eerst nagaat of het woord uit de inputstring in α_1 behoort tot de syntaktische klasse aangegeven door het terminaal symbool, als dit algoritme 'ja' geeft, komt er een nieuwe taak door een transitie te maken voorgeschreven in de desbetreffende regel van de grammatika.
Als het neen is, laten we de taak uitsterven door geen nieuwe taken meer te maken.
(iii) het element in α_2 is de laatste toestand (↑) van een nonterminaal symbool, dan voeren we NEWTASK III uit, die opnieuw een transitie doorvoert, nu in de regel van de grammatika die als input dit nonterminaal symbool aannam. Hierdoor ontstaat dan de passende taak.

Merk op dat het systeem rekursief is opgebouwd en als een vorm van multiprocessing is georganiseerd.

Voorbeeld 3.2.

We bespreken nu even de parsing aangegeven in het begin van dit hoofdstuk. De lezer wordt aangeraden dit voorbeeld met grote aandacht te bestuderen. De grammatika is dezelfde als in vb. 3.1. en we werken op de transitietabel van de grammatika gegeven op pag.
De inputstring is opnieuw "132" dat zoals bekend ambigu is volgens de gegeven grammatika. De kladblok van de ontleding van "132" ziet er als volgt uit, waarbij "1" het eerste woord "3" het tweede en "2" het derde is.

	α_1	α_2	α_3	α_4	α_5	α_6
1.	1	A1	1	0	0	0
2.	1	A2	1	1	1	
3.	1	A3	1	1	3	
4.	1	a4	0	2	6	2
5	1	a6	0	3	5	3
6.	2	A2	2	4	6	2
7.	2	A3	↑	5	5	3
8.	2	a5	0	6	7	2
9.	2	A1	3	7	3	1
10.	3	A2	↑	8	7	2
11.	2	A2	1	9	4	1
12.	3	A1	2	10	1	1
13.	2	a4	0	11	6	11
14.	3	A3	1	12	2	1
15.	3	A2	2	13	6	11
16.	3	a6	0	14	5	1
17.	3	a5	0	15	7	11
18.	4	A3	↑	16	5	1
19.	4	A2	↑	17	7	11
20.	4	stop				
21	4	A1	↑	19	4	1
22	4	stop				

We beschrijven enkele stappen.

De eerste taak zet de verwijzing naar het woord op 1, en geeft het axioma op, dit is A1/1, dwz. een nonterminaal symbool. Dit nonterminaal symbool komt voor in de regels 1 en 3 van de grammatika. We maken dus twee nieuwe taken (op 2 en 3) met in $\alpha 2$ en $\alpha 3$ het symbool resp. de kode van de transitieregels 1 en 3 namelijk A2/1 en A3/1. In $\alpha 1$ staat nog altijd hetzelfde woord en de regels in $\alpha 5$ zijn resp. 1 en 3. Omdat in beide gevallen een nieuw nonterminaal symbool wordt begonnen staat op de plaats van de begintoestand ($\alpha 6$) het regelnummer zelf van de nieuwe taak dus 2 en 3. In $\alpha 4$ dat de vroegere konfiguratie aangeeft, komt in beide gevallen '1'. Hiermee is de eerste taak uitgevoerd en beginnen we aan de nieuwe taken. Deze bevatten als input opnieuw een nonterminaal symbool, namelijk A2 en A3. We herschrijven dus zoals in het vorige geval en komen zo tot taak 4 en 5. Deze bevatten twee terminale symbolen. We gaan na of ze kloppen met de input. Dit is voor beide het geval. Taak 6 komt van 4 en bevat in $\alpha 2$ de toestand bekomen na de transitie voorgeschreven door de regel opgeslagen in $\alpha 4$ van taak 4. Wanneer er geen nieuwe nonterminale symbolen werden begonnen, kan men (op de zesde plaats) dezelfde konfiguraties overnemen (hier 3.). Deze zesde plaats komt goed van pas wanneer het einde van een nonterminaal symbool werd bereikt zoals in taak 7. De verwerking van deze taak met het resultaat in 9 is zodanig dat het woordnummer en de vroegere konfiguratie wordt gehaald uit taak 7. Voor de rest gaan we naar taak 3 en voeren die uit, dwz herschrijven volgens de richtlijnen van de regel 3, enz...

(iii) de output

Nu hebben we wel een mooie matrix maar nog geen strukturele deskriptie. Deze wordt verkregen door een pad te rekonstrueren vanaf de taken waar een stop voorkomt, dus de taken die het axioma hebben verwerkt en die voldoen aan de volgende voorwaarde:

'een 'stoptaak' kan enkel als einde van een grammatikaal pad beschouwd worden als in $\alpha 1$ het aantal woorden van de zin plus 1 staat.'

Eens zo'n taak gevonden klimmen we naar boven geleid door het nummer op de vierde plaats dat de vroegere konfiguratie aangeeft. Bij 1 aangekomen, kan de output starten die de kontrolestrukturen volgt besproken bij het begin van deze paragraaf (pag.).

(i) als in $\alpha 2$ een nonterminaal symbool wordt begonnen, schrijf dat '(x' en x is het nonterminaal symbool.

(ii) als in $\alpha 2$ een terminaal symbool staat, schrijf dat '(x y)' waarbij x het terminaal symbool is en y het element uit de inputstring in $\alpha 1$.

(iii) als het einde van een nonterminaal symbool is bereikt ($\alpha 3 = \uparrow$) geef dan een haakje: ')'.

In de andere gevallen, doe niets.

Toegepast op ons voorbeeld geeft dat de reeksen:

1. 22 21 19 17 15 13 11 9 7 5 3 1
2. 20 18 16 14 12 10 8 6 5 2 1

of:

1. (A1 (A3 (a6 1))(A2(a4 3) (a5 2)))
2. (A1 (A2 (a4 1)(a5 3)) (A3 (a6 3)))

en

De lezer kan nagaan met vroegere resultaten dat dit wel degelijk de beoogde strukturele deskriptie is.

Voorbeeld 3.3.

We geven nu een voorbeeld van een ontleding met een natuurlijke taal grammatika.

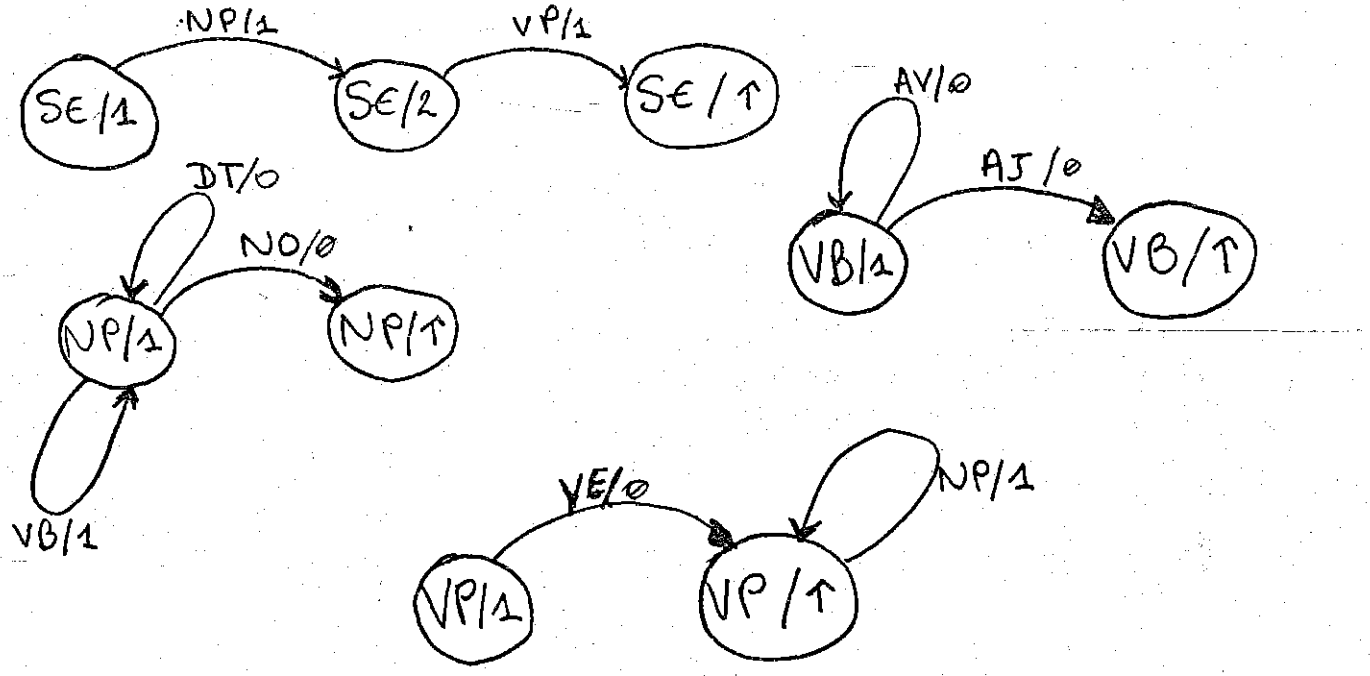
De grammatika als een CF-regelsysteem:

NP → (DT)* (VB)*NO
VB → (AV)* AJ
VP → VE (NP)*

DT → de
AJ → grote
NO → spelen, spelletjes, grote
VE → spelen
AV → dikwijls

Het sterretje is de Kleene-operator en wil zeggen 'gelijk welk aantal van ', de haakjes duiden op optionaliteit.

De grammatika als een state transition diagram



In tabelvorm:

1.	NP1	SE1	SE2
2.	DTØ	NP1	NP1
3.	VB1	NP1	NP1
4.	AJØ	VB1	VB50
5.	AVØ	VB1	VB1
6.	NOØ	NP1	NP50
7.	VP1	SE2	SE50
8.	VEØ	VP1	VP2
9.	VEØ	VP1	VP2
10.	VEØ	VP1	VP50
11.	NP1	VP2	VP2
12.	NP1	VP2	VP50

We gaan de volgende zin ingeven:

'DE GROTE SPELEN SPELLETJES'.

Het bijzondere hiervan is dat er ambiguïteit aanwezig is tot en met 'spelen', namelijk of 'de grote spelen' een NP is of een NP-VE konstruktie. Deze ambiguïteit wordt opgelost door 'spelletjes' .

KLADBLOK:

1	1	1	0	1	1
1	2	1	1	1	2
1	1	0	2	0	2
2	2	1	3	0	2
1	3	1	2	3	5
2	3	1	4	3	6
2	3	0	4	0	2
2	2	50	7	0	2
2	2	0	6	0	2
2	1	50	9	0	2
2	1	2	0	1	1
2	2	1	10	3	2
3	4	1	11	7	13
3	3	1	12	3	14
3	3	0	12	0	2
3	2	50	15	0	2
4	4	0	13	1	13
4	4	2	17	7	13
4	1	2	16	1	1
4	2	1	18	9	20
4	4	1	19	7	21
4	3	1	20	3	20
4	3	0	20	0	20
5	4	50	23	3	20
5	4	50	24	9	13
5	1	50	25	7	1

```
AANTAL PARSINGS: 1
STRUKTURELE BESKRIPTIE 1
PAD DOOR KLADBLOK:
 26    25    24    23    20    18    17    13    11    8    7    4
 3     2     1     PARSING:
(SE(NP(DT DE)(NO GROTE))(VP(VE SPELEN)(NP(NO SPELLETJES))))
```

De aandachtige lezer zal hebben opgemerkt dat de kladblok niet alle taken bevat die redelijkerwijze kunnen verwacht worden. Inderdaad werden enkele middelen toegepast om het proces te vereenvoudigen:

Versnelling van de algoritmen

a) We kunnen NEWTASK II uitvoeren tijdens de vorming van de taken en niet bij de uitwerking. Wanneer het element uit de inputstring behoort tot de syntaktische kategorie gegeven in een transitie, dan pas wordt een nieuwe taak gekonstrueerd met daarop de informatie omtrent deze stap. Ook kan rechtstreeks de transitie uitgevoerd worden die nodig is volgens de gegeven grammatika.
Hierdoor wordt niet alleen geheugenruimte uitgespaard (± 30 % voor dit voorbeeld), ook gaat het proces sneller omdat minder taken moeten gekonstrueerd worden.

b) Door de regels te ordenen bij de input, kunnen we de zoek-procedure in onze grammatika te versnellen. Oorspronkelijk moest immers de ganse transitiematrix doorlopen worden.

c) Soms kunnen we ook $\alpha 5$ weglaten. $\alpha 5$ is er enkel nodig voor taken die een inbedding beginnen.

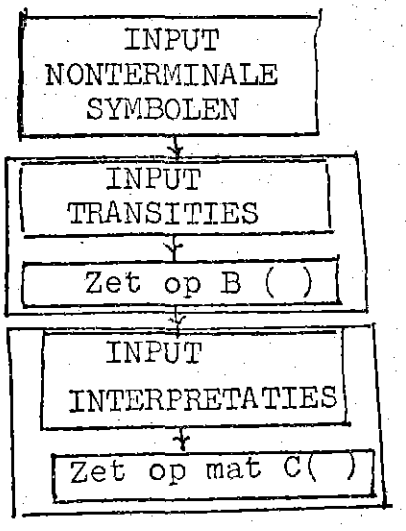
Nu kunnen we overgaan tot een meer expliciete beschrijving van het systeem.

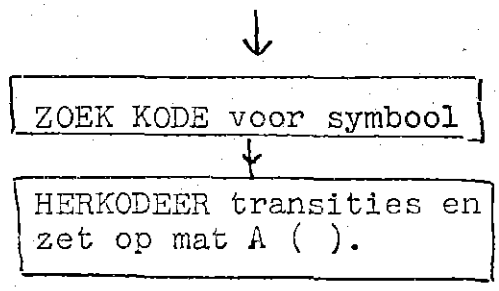
3.2. SYNTAXTEST II, Formalisering en implementatie.

I. Kodering.

Zoals in het vorige programma dat een deterministische parser simuleerde, zal er een koderingsfaze aanwezig zijn die de symbolen van de grammatika in numerische kode omzet. Het algoritme gaat als volgt:

input:





In appendix III vindt men het programma voor dit algoritme.

Nota:

1. Het axioma wordt expliciet ingevoerd bij elke ontleding omdat er zo experimenten kunnen gedaan worden voor kleinere syntaktische groepen. Voorbeeld, als NP het axioma is, zal er enkel een NP als input worden geaccepteerd.

2. De terminale symbolen worden niet onmiddellijk ingegeven (in tegenstelling met de koderingsfaze van het vorige hoofdstuk). Zij worden afgeleid uit de interpretatieregels.

II. Algoritmen

De transities staan op matrix A, de interpretaties staan op matrix C, als kladblok gebruiken we matrix D. Deze kent 6 kolommen korresponderend met de zes elementen van een taak. We vragen eerst naar het axioma, we zoeken de kode hiervoor en konstrueren een eerste taak.
Dan geven we de inputstring in die zal onderzocht worden. Deze heet C$. We kijken na waar er woorden zijn en nummeren ze van links naar rechts.

De matrix wordt van boven naar beneden met twee tellers overlopen:
I1: nummer van de taak waar we mee bezig zijn
I2: aantal taken dat er aanwezig is.
Wanneer een nieuwe taak verwerkt is, wordt taak I1 + 1 verwerkt en wanneer er een nieuwe taak wordt gevormd staat die op I2 + 1.

We bespreken achtereenvolgens NEWTASK I, NEWTASK II, NEWTASK III.

1. NEWTASK I.

Het nonterminaal symbool moet herschreven worden, dit wil zeggen we zoeken de verschillende regels waar het nonterminaal symbool met de juiste Kode voorkomt op A als begin van een transitie. Wanneer de input van deze transitie een terminaal symbool is gaan we onmiddellijk naar NEWTASK II , wanneer de input een nonterminaal symbool is maken we een nieuwe taak en dit gaat als volgt:

I2 = I2 + 1 (we beginnen een nieuwe taak)
D(I2,2) = A(I3,1) dit is de input van de transitie. I3 is de regel.
D(I2,3) = kode van de input A(I3,2)
D(I2,4) = I1 : nummer van de taak die aanleiding gaf tot deze taak.
D(I2,1) = D(I1,1) : woordnummer blijft hetzelfde.
D(I2,6) = inbeddingstaak. Deze is gelijk aan het taaknummer zelf als (a) de kode op plaats 3 gelijk is aan 1 en (b) als het nonterminaal symbool zelf niet reeds voorkwam. (Bijvoorbeeld in taak 2 van

het laatste voorbeeld. We zijn bezig in regel 1 van de grammatika. Deze bevat als input NP/1, in kode 2 1. Dus we schrijven in het laatste vakje het nummer van de taak , d.i. 2.
In taak 4 echter heeft het vinden van een determinator ertoe geleid van de transitie te maken in de tweede regel van de grammatika. Dit leidt opnieuw tot NP/1. In taak 4 staat dus opnieuw op de tweede en derde plaats 2 1, doch op de laatste plaats staat 2 en niet 5.)

2. NEWTASK II.

Dit algoritme gaat na of een woord uit de inputstring voorkomt in de syntaktische kategorie aangegeven door een terminaal symbool. Wanneer dit niet het geval is gaan we gewoon verder zonder enige akties te ondernemen. Wanneer wel, konstrueren we een nieuwe taak op I2 + 1, en dit gaat dan zo:

I2 = I2 + 1
D(I2,1) = volgende woord in de inputstring: D(I1,1) + 1
D(I2,2) = symbool dat verschijnt na transitie, namelijk het laatste symbool van de regel aangeduid in de regel van herkomst. Dit symbool vinden we in de vijfde kolom van de regel op matrix A, dus A(I3,5)
D(I2,3) = kode van dit symbool: A(I3,6)
D(I2,4) = I1, nummer van de taak dat aanleiding gaf tot deze taak.
D(I2,6) = Nummer van de inbedding, uiteraard dezelfde als in de regel van afkomst: D(I1,6).

Nu is er echter het volgende probleem: de taak waar een terminaal symbool in voorkwam staat niet op de matrix, we zijn namelijk direkt van NEWTASK I naar NEWTASK II gegaan zonder tussentaak. Bij de output zou er bijgevolg vruchteloos worden gezocht naar een taak met een terminaal symbool. Dit is echter vlug verholpen door vooraleer bovenstaande reeks procedures wordt doorlopen, NEWTASK I uit te voeren en dan terug te komen. De lezer kan voor zichzelf nagaan dat dit inderdaad het gewenste resultaat heeft.

3. NEWTASK III

Het element in α_2 is de laatste toestand van een nonterminaal symbool. We doen dan een transitie in die regel van de grammatika die aangegeven staat op de plaats van de inbedding (α 6).
Dit leidt tot de volgende reeks aktiviteiten:
I2 = I2 + 1 s(volgende taak)
D(I2,1) = D(I1,1) namelijk hetzelfde woord uit de input.
Voor de rest maken we de transitie en dit verloopt analoog met de konstruktie van een taak in NEWTASK II. (cfr supra)

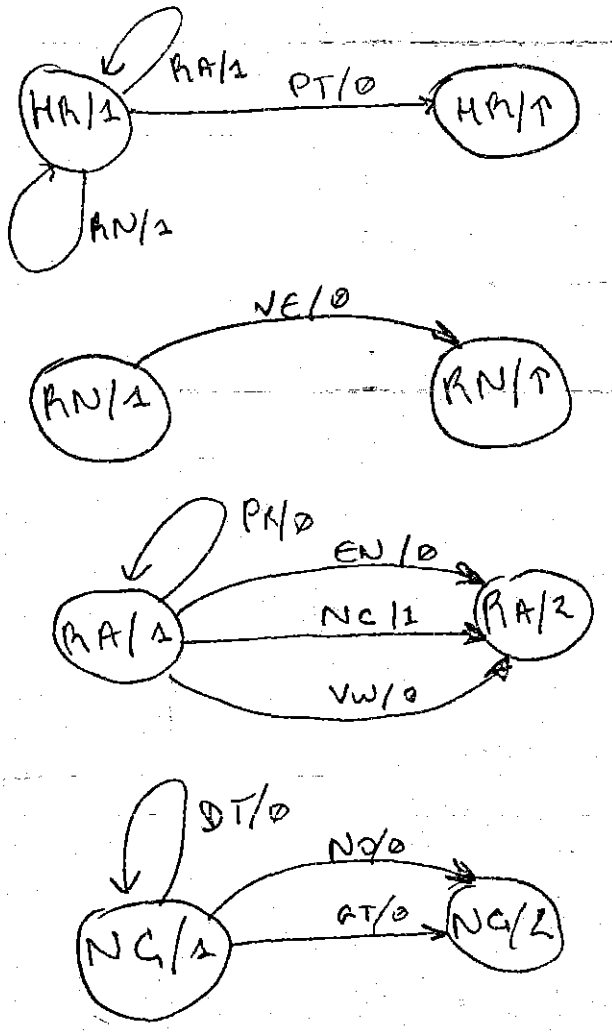
Wanneer er nu een stop-taak wordt onmoet die voldoet aan alle voorwaarden vroeger gegeven op een vektor gezet namelijk ( ). Vanuit stoptaken worden uiteraard geen nieuwe taken meer gekonstrueerd.

Voor de output volgen we de gedachtengang die hieromtrent werd gegeven. We overlopen S en zetten alle taken van het pad dat eindigt in het nummer aangegeven op S op een vektor R Dan overlopen we R van achter naar voor en geven de output.

Zie Appendix IV voor de programma's van deze algoritmen.

Ter illustratie geven we nog een uitgebreid experiment met de processor. De grammatika die hier gebruikt wordt, is niet alleen exemplarisch van belang. De beschrijving is een onderdeel van een dieper uitgebouwd systeem en speelt een rol in komende onderzoeksprojekten. (Zie Steels, 1974 b)

De grammatika als een transitiediagram:



```
L.STEELS 1.12.74 TAPE 21 FILE 7
S/MTAN.TEST II (NONDET. ATN-PARSER)

NONTERMINALS: HRRARNNC
INPUT TRANSITIES :
RA1 HR1 HR1
RN1 HR1 HR1
PT0 HR1 HR50
VE0 RN1 RN50
PR0 RA1 RA1
NC1 RA1 RA50
DT0 NC1 NC1
NO0 NC1 NC50
EN0 NC1 NC50
GT0 NC1 NC50
VW0 NC1 NC50
INPUT INTERPRETATIES:
PT =  . , ?
VE =  IS GEBEURT WERKT
NO =  VADER MOEDER ZOON DOCHTER ARBEIDER FABRIEK ZIJSTRAAT SOM VERSCHIL APPEL
EN =  JAN PIET KLAAS ANNIE LIEVE RIA FRANKRIJKLEI PAARDEMARKT
DT =  DE HET EEN
PR =  VAN ALS IN OP NA VOOR TUSSEN EN OF
GT =  1 2 3 4 5 6 7 8 9 0
 W = VW
VW =  WIE WAT WAAR HOEVEEL
^  =
TRANSITIEMATRIX
 2     1     1     1     1     1
 3     1     1     1     1     1
 1     0     1     1     1    50
 2     0     3     1     3    50
 6     0     2     1     2     1
 4     1     2     1     2    50
 5     0     4     1     4     1
 3     0     4     1     4    50
 4     0     4     1     4    50
 7     0     4     1     4    50
 9     0     4     1     4    50


 AXIOMA :RA

 INPUT: DE VADER


 KLADBLOK:
  1     2     1     0     1     1
  1     4     1     1     6     2
  1     5     0     2     2     2
  2     4     1     3     6     2
  2     3     0     4     0     2
  3     4    50     5     6     2
  3     2    50     6     6     1

 AANTAL PARSINGS: 1
 STRUKTURELE DESKRIPTIE 1
 PAD DOOR KLADBLOK:
  7     6     5     4     3     2     1    PARSING:
 (RA(NC(DT DE)(NO VADER)))
```

```
AXIOMA :HR

INPUT: DE FRANKRIJKLEI IS EEN ZIJSTRAAT VAN DE PAARDEMARKT .

KLADBLOK:
 1    1    1    0    1    1
 1    2    1    1    1    2
 1    3    1    1    2    3
 1    4    1    2    6    4
 1    5    0    4    0    4
 2    4    1    5    0    4
 3    4    0    6    0    4
 3    4    1    7    0    4
 3    2    1    8    6    2
 3    1    1    9    1    1
 3    2    1   10    1   11
 3    3    1   10    2   12
 3    4    1   11    6   13
 4    2    0   12    0   12
 4    3    50  14    0   12
 4    1    1   15    2    1
 4    2    1   16    1   17
 4    3    1   16    2   18
 4    4    1   17    6   19
 4    5    0   19    0   19
 5    4    1   20    0   19
 5    3    0   21    0   19
 6    4    50  22    0   19
 6    2    50  23    6   17
 6    1    1   24    1    1
 6    2    1   25    1   26
 6    3    1   25    2   27
 6    5    0   26    0   26
 7    5    1   28    0   26
 7    4    1   26    5   30
 7    5    1   29    6   31
 8    5    0   31    0   31
 8    4    1   32    0   31
 9    4    0   33    0   31
 9    4    50  34    0   31
 9    2    50  35    5   26
 9    1    1   36    1    1
 9    2    1   37    1   38
 9    3    1   37    2   39
 9    1    0   37    0    1
 10   1    50  40    0    1
 9    4    1   38    6   42

AANTAL PARSINGS: 1
STRUKTURELE BESKRIPTIE 1
PAD DOOR KLADBLOK:
 41   40   37   36   35   34   33   32   31   29   28   26
 25   24   23   22   21   20   19   17   16   15   14   12
 10   9    8    7    6    5    4    2    1    PARSING:
(HR(RA(NC(DT DE)(EN FRANKRIJKLEI))(RN(VE IS))(RA(NC(DT EEN)(NO
ZIJSTRAAT))(RA(PR VAN)(NC(DT DE)(EN PAARDEMARKT)))(PT .))

---

AXIOMA :HR

INPUT: HOEVEEL IS DE SOM VAN 1 EN 2 ?
```

KLADBLOK:

1	1	1	0	1	1	
1	1	1	1	1	2	
1	1	1	1	3	3	
1	1	1	2	6	4	
1	1	0	4	0	4	
1	1	50	5	0	4	
1	1	50	6	0	4	
1	1	1	7	1	4	
1	1	1	8	2	10	
1	1	1	9	6	11	
1	1	0	10	0	10	
1	1	50	12	0	10	
1	1	1	13	1	1	
1	1	1	14	1	15	
1	1	1	15	2	16	
1	1	1	16	6	17	
1	1	0	17	2	17	
1	1	1	18	6	17	
1	1	0	19	0	17	
1	1	50	20	0	17	
1	1	50	21	6	15	
1	1	1	22	1	1	
1	1	1	23	1	24	
1	1	0	24	1	24	
1	1	1	26	2	24	
1	1	1	24	6	24	
1	1	1	27	6	24	
1	1	0	29	6	24	
1	1	50	30	6	24	
1	1	50	31	6	24	
1	1	1	32	1	1	
1	1	1	33	1	34	
1	1	1	33	2	35	
1	1	0	34	6	34	
1	1	1	36	1	34	
1	1	1	34	6	30	
1	1	1	37	6	30	
1	1	0	39	0	30	
1	1	50	40	0	30	
1	1	50	41	6	34	
1	1	1	42	1	1	
1	1	1	43	1	44	
1	1	1	43	2	45	
1	1	0	43	0	1	
1	1	50	46	0	1	
1	1	1	44	6	48	

AANTAL PARSINGS: 1
STRUKTURELE DESKRIPTIE 1
PAD DOOR KLADBLOK:

47	46	43	42	41	40	39	37	36	34	33	32
31	30	29	27	26	24	23	22	21	20	19	18
17	15	14	13	12	10	0	7	6	5	4	2
1											

PARSING:
(HR(RA(NC(VW HOEVEEL))(RN(VE IS))(RA(NC(DT DE)(NO SOM)))(RA(PR VAN)(NC(GT 1)))(RA(PR EN)(NC(GT 2)))(PT ?))

AXIOMA :HR

INPUT: KLAAS WERKT ALS ARBEIDER IN EEN FABRIEK .

KLADBLOK:

1	1	1	0	1	1
1	2	1	1	1	2
1	3	1	1	2	3
1	4	1	2	6	4
1	4	0	4	0	4
2	4	50	5	0	4
2	2	50	6	6	2
2	1	1	7	1	1
2	2	1	8	1	9
2	3	1	8	2	10
2	4	1	9	6	11
3	2	0	10	2	10
3	3	50	12	6	10
3	1	1	13	2	1
3	2	1	14	1	15
3	3	1	14	2	16
3	6	0	15	0	15
4	2	1	17	2	15
4	4	1	15	6	10
4	4	1	19	0	20
4	3	0	20	0	20
5	4	50	21	6	20
5	2	50	22	6	15
5	1	1	23	1	1
5	2	1	24	1	25
5	3	1	24	2	26
5	6	0	25	2	25
6	2	1	27	6	25
6	4	1	28	6	29
6	4	1	28	6	30
6	5	0	30	6	30
7	4	1	31	6	30
7	3	0	32	1	30
8	4	50	33	1	30
8	2	50	34	6	25
8	1	1	35	1	1
8	2	1	36	1	37
8	3	1	36	2	38
8	1	0	38	6	1
9	1	50	39	0	1
8	4	1	37	6	41

AANTAL PARSINGS: 1
STRUKTURELE DESKRIPTIE 1
PAD DOOR KLADBLOK:

40	39	36	35	34	33	32	31	30	28	27	25
24	23	22	21	20	18	17	15	14	13	12	10
8	7	6	5	4	2	1	PARSING:				

(HR(RA(NC(EN KLAAS))(RN(VE WERKT))(RA(PR ALS)(NC(NO ARBEIDER)))(RA(PR IN)(NC(DT EEN)(NO FABRIEK)))(PT .))

AXIOMA :HR

INPUT: ANNIE IS DE DOCHTER VAN DE MOEDER VAN KLAAS .

```
KLADBLOK:
  1       1       1       0       1       1
  1       2       1       1       1       2
  1       3       1       1       2       3
  1       4       1       2       6       4
  2       4       0       4       0       4
  2       4      50       5       0       4
  2       2      50       6       6       2
  2       1       1       7       1       1
  2       1       1       8       1       9
  2       3       1       9       6      10
  2       4       1      10       6      11
  3       2       0      11       6      10
  3       3      50      12       6      10
  3       1       1      13       2       1
  3       2       1      14       1      15
  3       3       1      14       2      16
  3       4       1      15       6      17
  3       5       0      17       2      17
  4       4       1      18       6      17
  4       3       0      19       6      17
  5       4      50      20       0      17
  5       2      50      21       6      15
  5       1       1      22       1       1
  5       2       1      23       1      24
  5       3       1      23       2      25
  5       6       0      24       2      24
  5       1       1      25       2      24
  5       4       1      24       6      20
  6       4       1      27       6      29
  6       5       0      29       6      29
  7       4       1      30       6      29
  7       3       0      31       6      29
  8       4      50      32       1      29
  8       2      50      33       6      24
  8       1       1      34       1       1
  8       2       1      35       1      36
  8       3       1      35       2      37
  8       6       0      36       2      36
  9       2       1      38       6      36
  8       4       1      36       6      40
  9       4       1      39       6      41
  9       4       0      41       6      41
 10       4      50      42       1      41
 10       2      50      43       6      36
 10       1       1      44       1       1
 10       2       1      45       1      46
 10       3       1      45       2      47
 10       1       0      45       6       1
 11       1      50      48       0       1
 10       4       1      46       6      50

AANTAL PARSINGS: 1
STRUKTURELE DESKRIPTIE 1
PAD DOOR KLADBLOK:
 49      48      45      44      43      42      41      39      38      36      35      34
 33      32      31      30      29      27      26      24      23      22      21      20
 19      18      17      15      14      13      12      10       0       7       6       5
  4       2       1    PARSING:
(HR(RA(NC(EN ANNIE)))(RN(VE IS))(RA(NC(DT DE)(NO DOCHTER)))(RA(PR VAN)(
NC(DT DE)(NO MOEDER)))(RA(PR VAN)(NC(EN KLAAS)))(PT .))
```

```
AXIOMA :HR

INPUT: OP DE BOMEN VAN HET PARK ZITTEN .


KLADBLOK:
 1      1      1      0      1      1
 1      2      1      1      1      2
 1      3      1      1      2      3
 1      5      0      2      6      2
 2      3      1      4      0      3
 1      4      1      2      5      6
 2      4      1      5      6      7
 2      5      0      7      1      7
 3      4      1      8      6      7

AANTAL PARSINGS: 0
INGEVOERDE ZIN IS ONGRAMMATIKAAL

-------------------------------------------------------------------------------

AXIOMA :HR

INPUT: BOMEN OP DE VAN .


KLADBLOK:
 1      1      1      0      1      1
 1      2      1      1      1      2
 1      3      1      1      2      3
 1      4      1      2      6      4

AANTAL PARSINGS: 0
INGEVOERDE ZIN IS ONGRAMMATIKAAL

-------------------------------------------------------------------------------

AXIOMA :HR

INPUT:SGFRT RIUTO CMVN .


KLADBLOK:
 1      1      1      0      1      1
 1      2      1      1      1      2
 1      3      1      1      2      3
 1      4      1      2      6      4

AANTAL PARSINGS: 0
INGEVOERDE ZIN IS ONGRAMMATIKAAL

-------------------------------------------------------------------------------
```

4. BESLUIT

In deze paper werden twee parsingsystemen beschreven. Het eerste (SYNTAXTEST I) accepteerde als input een kontekst-vrije grammatika, dit wil zeggen een quadrupel G = (Vn,Vt,P,S), genereerde een kode voor deze grammatika, en bood daarna de mogelijkheid om de beschrijving die wordt voortgebracht door deze grammatika te toetsen aan zinnen uit het bedoelde bereik. Syntaxtest I was deterministisch omdat geen ambigue zinnen konden ontleed worden en omdat nonterminale symbolen een unieke herschrijving moesten hebben.

Het tweede systeem, SYNTAXTEST II, accepteerde ook een syntaktische beschrijving, nu in de vorm van een rekursief transitie netwerk, genereerde een kode en bood experimenteer-mogelijkheden binnen deze grammatika.

SYNTAXTEST II is vooral geschikt voor linguïsten die een rekognitieve grammatika willen schrijven voor een bepaald onderdeel van de taal. Men kan allerlei experimenten doen zoals de strukturele deskriptie kontroleren en adekwater maken (zwakke generatieve kapaciteit), het bereik vergroten (sterke generatieve qkapaciteit de grammatika empirisch toetsen aan verzamelingen zinnen, de grammatika versnellen, enz...
Ook kan men het systeem koppelen aan andere programma's die de strukturele deskriptie omzetten in een semantische representatie die als input kan dienst doens voor een semantische komponent. Momenteel is onderzoek in deze richting bezig (SYNTAXTEST III). Hierdoor krijgt men een processor op het niveau van een 'augmented' transitie netwerk grammatika (cfr Woods (1972), Simmons (1973)) of een transformationeel generatieve grammatika.

Nota:

Ik dank hierbij Martin Kay voor de introduktie in de problematiek van de automatische syntaktische analyse en William Woods voor een inleiding in de Augmented Transition Network Grammars. Verder dank ik ook prof. Dr. R. Van de Velde voor waardevolle kritiek op delen van het manuskript en prof. Dr. G. De Schutter die me in de gelegenheid heeft gesteld dit onderzoek uit te voeren.

L. STEELS
Antwerpen 1974

5. Bibliografie

BRAINERD, B. (1971) Introduction to the mathematics of Language study. New York.

CHOMSKY, N. (1957) Syntactic Structures. 1971 (9). Den Haag.

HARBORDT, R. (1974) Computersimulation in de Sozialwissenschaften. Hamburg.

KAY, M. (1973) , K. Sparck Jones. Linguistics and Information Science. New York.

KAY, M. (1973) The MIND system. in Rustin (1973).

KAPLAN, R. (1973) A General Syntactic Processor. in Rustin (1973).

KUNO, S. (1966) The predictive Analyzer. in Hays, D.G. (ed) Readings in Automatic Language Processing. 1966. New York.

LINDAHL, T. (1971) An introduction to Basic. A time sharing language. California.

LEVELT, W.J.M. (1973) Formele grammatica's in linguïstiek en taalpsychologie. Deventer.

Manual (1973) Hewlett Packard Model 1830 A.

MINSKY, M. (1967) Computation finite and infinite machines. London.

OETTINGER, (1961) Automatic syntactic analysis and the Pushdown store. Proceedings of the symposia in applied mathematics. Vol XII. Structure of Language in its mathematical Aspects. American Mathematical Society; 1961.

PELUSO, A. (1972) Basic BASIC programming, Self-instruction manual and text. Reading Mass..

PETRICK, S.R. (1965) A recognition procedure for Transformational Grammars, Ph.D. Thesis. M.I.T..

PETRICK, S.R. (1973) Transformational Analysis. Rustin (1973).

SALOMAA, (1973) Formal Languages.

STEELS, L. (1974 a) Representatie van linguïstische informatie. (op komst)

STEELS, L. (1974 b) Seminaries in procedurële semantiek. U.I.A. Afd. Ned. Taalkunde. Interne nota.

WOODS, W. (1970) Transition Network Grammars for Natural Language Analysis. Communications of the ACM, nr 10. oct 1970.

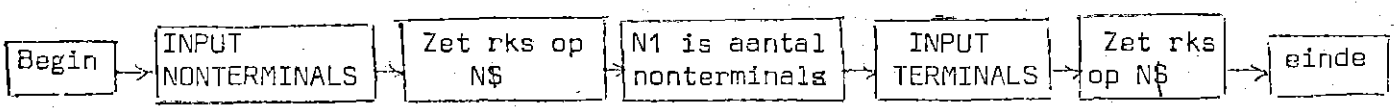
WOODS, W. (1972) et.al. Lunar Sciences Natural Language Information system. BBN report 2378. 1972 june.

APPENDIX I.

① KODE

De kode wordt toegekend naargelang elementen binnenkomen. Dus het symbool dat als eerste wordt ingegeven heeft systeemkode 1, het symbool dat als vierde wordt ingegeven heeft interne kode 4, enz...
De kode van de gebruiker wordt in alfanumerische vorm opgeslagen op de string N$. We stellen nu als eis van een symbool dat het twee letters bevat en niet meer. Indien 1 dan moet iedere keer een spatie worden bijgegeven, na of voor het symbool in gebruikerskode.
Door deze normalisering wordt de vorming van N$, de kodering en herkodering triviaal. Immers zij I de interne kode dan geeft N$((I * 2) -1, I*2) de gebruikers kode. Vorming van N$ gebeurt bij het ingeven van de symbolen en de reeks wordt in zijn geheel op N$ gezet: N$(I, LEN(C$) + I) = C$
Omdat het handig zal blijken later om te weten hoeveel nonterminale symbolen er zijn onthouden we dit door de variabele N1.
Het axioma is steeds AX in de gebruikerskode en 1 in numerische kode, zodat dit niet meer moet ingegeven worden. Wel moet in een produktie dit axioma herschreven worden. Deze oplossing voor wat betreft de input van het axioma werd gekozen omwille van de technische eigenschappen (namelijk 'recursive control') van het ontledingsysteem.

Het algoritme voor de kodering gaat als volgt:



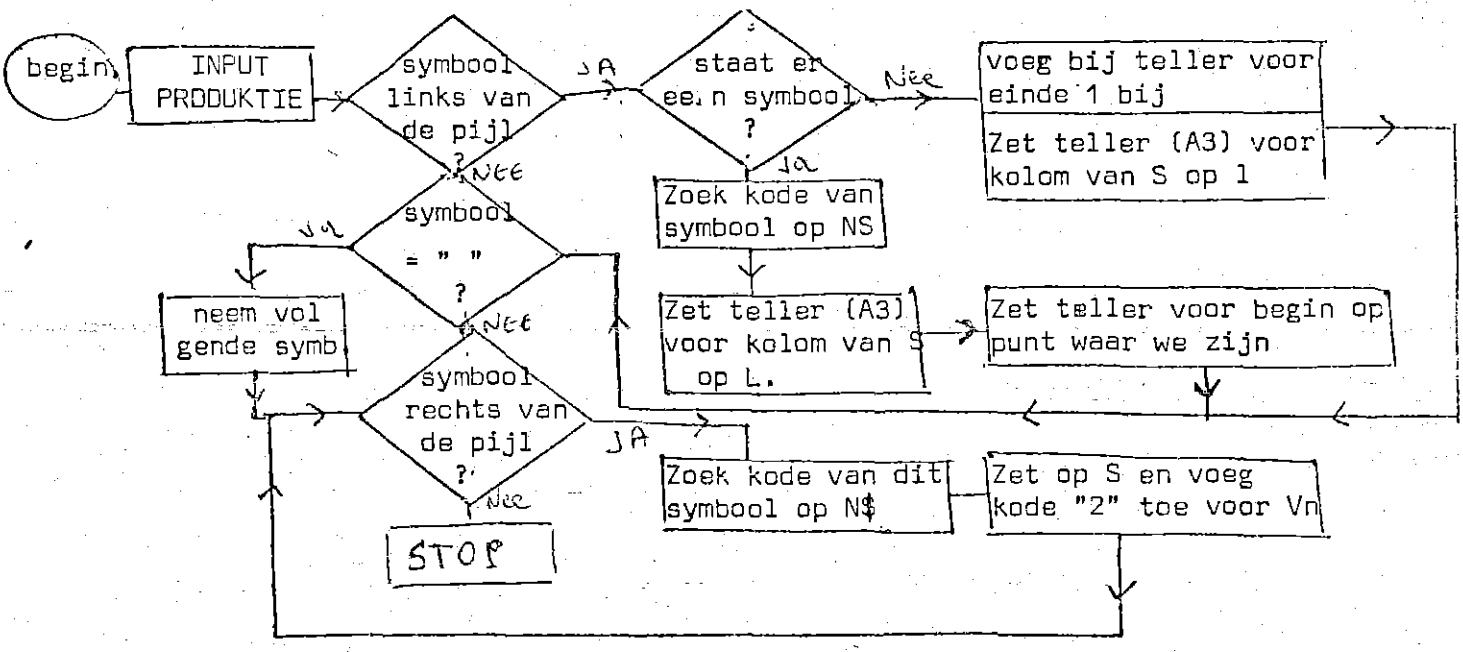
Het programma dat met dit algoritme korrespondeert :

```
80 I=3
90 N$="AX"
100 DISP "INPUT NONTERMINALS :";
110 INPUT C$
120 PRINT "NONTERMINALS:";C$
130 PRINT
140 N$[I,LEN(C$)+I]=C$
150 I=I+LEN(C$)
160 N1=((I-1)/2)
170 DISP "INPUT TERMINALS";
180 INPUT C$
190 PRINT "TERMINALS:";C$
200 PRINT
210 N$[I,LEN(C$)+I]=C$
220 PRINT "V(VERZ. SYMBOLEN)=";N$
230 PRINT
240 PRINT "AANTAL NONTERMINALS:";N1
```

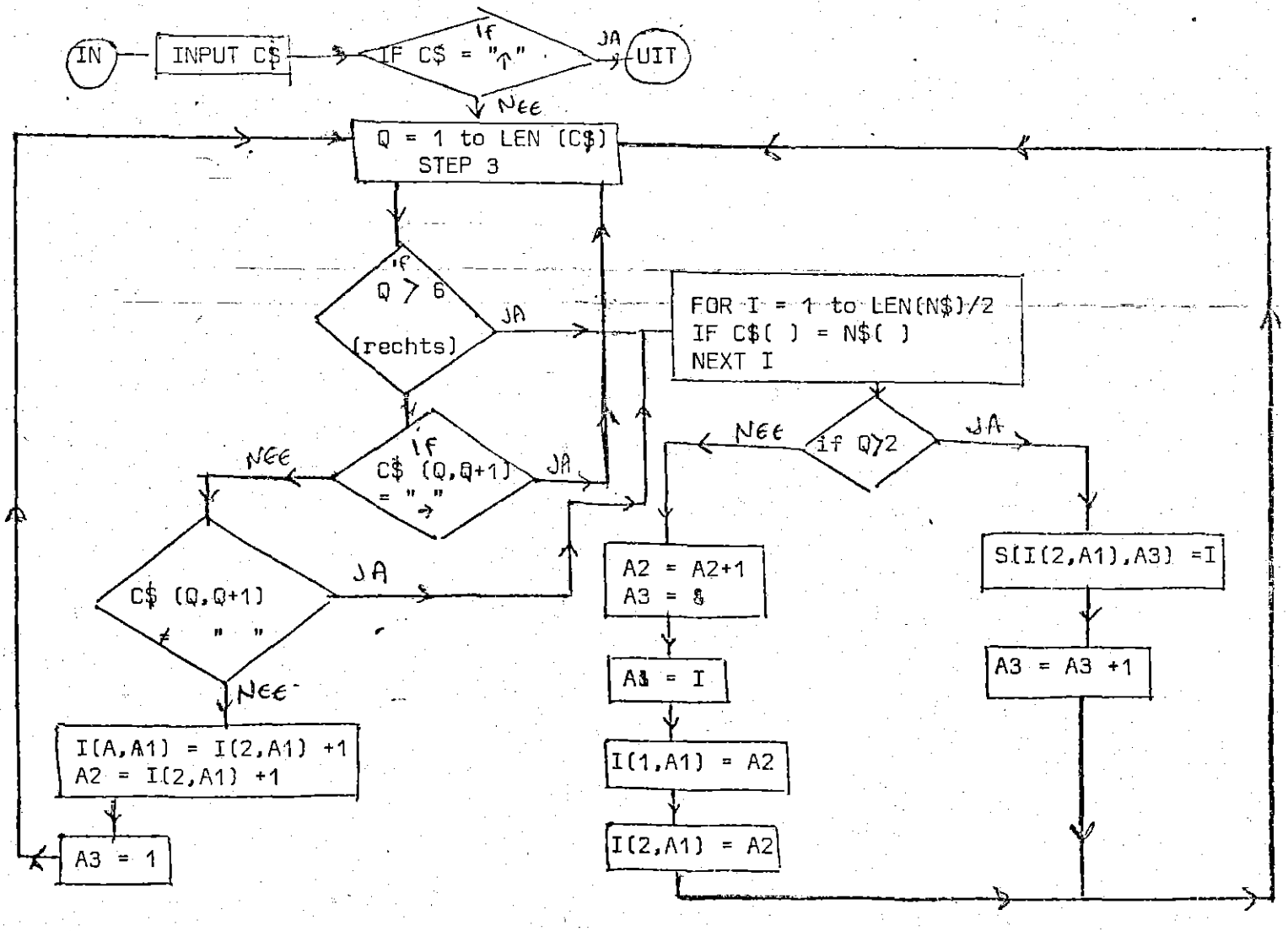
Opdat de gebruiker een duidelijk overzicht zou hebben over wat hij invoert, wordt eerst uitgeprint welke terminale en nonterminale symbolen hij heeft ingegeven, daarna de gehse reeks (met het axioma) en ook het aantal nonterminale symbolen (N1).

We gaan nu even na hoe de syntaxmatrix kan gekonstrueerd worden met als input de produkties van de grammatika.

Het algoritme:



Een meer expliciet algoritme:



Het programma:

```
260 A2=0
270 PRINT "PRODUKTIES:"
280 DISP "INPUT PRODUKTIE";
290 INPUT C$
300 PRINT C$
310 IF C$="†" THEN 610
320 C=LEN(C$)
330 FOR Q=1 TO C STEP 3
340 IF Q>6 THEN 410
350 IF C$[Q,Q+1]="->" THEN 560
360 IF C$[Q,Q+1]#"  " THEN 410
370 I[2,A1]=I[2,A1]+1
380 A2=I[2,A1]
390 A3=1
400 GOTO 560
410 FOR J=1 TO LEN(N$)/2
420 IF C$[Q,Q+1]=N$[(J*2)-1,J*2] THEN 450
430 NEXT J
440 GOTO 280
450 IF Q>2 THEN 520
460 A2=A2+1
470 A3=1
480 A1=J
490 I[2,A1]=A2
500 I[1,A1]=A2
510 GOTO 560
520 IF J>N1 THEN 540
530 J=J+20
540 S[I[2,A1],A3]=J
550 A3=A3+1
560 NEXT Q
570 FOR J=A3 TO 5
580 S[I[2,A1],J]=0
590 NEXT J
600 GOTO 280
610 PRINT
620 PRINT "SYNTAXMATRIX:"
630 FOR I=1 TO A2
640 FOR J=1 TO 5
650 PRINT S[I,J];
660 NEXT J
670 PRINT
680 NEXT I
```

Na de syntaxis is er een klein algoritme toegevoegd dat de mogelijkheid geeft om te herbeginnen wanneer het ergens fout ging:

```
690 DISP "TOTHIERTOE ALLES O.K.?";
700 INPUT C$
710 IF POS(C$,"JA")>0 THEN 740
720 PRINT "INPUT FOUTIEF BEGIN OPNIEUW"
730 GOTO 40
```

.2. Lexikon

De lezer zal reeds opgemerkt hebben dat de interpretatie van de terminale symbolen niet werd ingevoerd in de produkties. Dit lijkt gek voor formele talen, doch het is algemeen gangbaar voor natuurlijke taalgrammatika's. Ook bij logische taalstelsels, gebeurt de interpretatie pas na de deduktie van uitdrukkingen.

De input van het lexikon gaat als volgt: Wanneer een regel binnenkomt, zoeken we de kode van het linkersymbool op. Eens dit gevonden, zetten we de rechterkant op een lexikonstring L$. We onthouden op de informatiematrix het begin en het einde op L$ per terminaal symbool.

Nota. Restrikties ontstaan ook hier, namelijk de ingevoerde lexikale elementen mogen in totaal niet meer dan 250 eenheden bevatten. Dit is een beperking opgelegd door de programmeertaal waar we mee bezig zijn.

We geven direkt het programma omwille van de eenvoud van het probleem en de analogie met het vorige algoritme.

```
780 DISP "INPUT INTERPRETATIEREGEL";
790 S1=1
800 C$[1,1]=" "
810 INPUT C$
820 PRINT C$
830 IF C$="↑" THEN 940
840 FOR Q=1 TO LEN(N$)/2
850 IF N$[(Q*2)-1,Q*2]=C$[1,2] THEN 890
860 NEXT Q
870 PRINT "LINKERSYMBOOL NIET BEKEND VOER OPNIEUW IN"
880 GOTO 780
890 I[1,Q]=S1
900 I[2,Q]=S1+LEN(C$)-6
910 L$[I[1,Q],I[2,Q]]=C$[7,LEN(C$)]
920 S1=LEN(L$)
930 GOTO 800
940 PRINT "LEXIKON:";L$
950 PRINT
960 PRINT "INFORMATIEMATRIX: "
970 FOR A=1 TO 2
980 FOR I=1 TO LEN(N$)/2
990 PRINT I[A,I];
1000 NEXT I
1010 PRINT
1020 NEXT A
```

Zoals men kan zien, wordt na de input van het lexikon de volledige informatiematrix als output gegeven.

Hiermee hebben we alles ingevoerd wat redelijkerwijze voor een syntaktische ontleding van een proefzin nodig is en kunnen we de tweede faze aanvatten, het eigenlijke ontledingsysteem.

APPENDIX II

We onderkennen:

(a) Een deelsysteem dat één voor één een woord van de zin die werd ingevoerd 'leest'. Dit is een eenvoudige zaak en gaat als volgt, met input Q1 (begin van een woord) en C$ (zin).

```
1160 FOR I1=Q1 TO LEN(C$)
1170 IF C$[I1,I1]=" " THEN 1230
1180 NEXT I1
```

Het woord is nu gevonden met begin Q1 en einde I1-1

(b) Het eigenlijke analysesysteem.

Er zullen alvast 3 kondities zijn die in de rekursieve definitie voorkomen.
(i) is het element Ø :
IF S(I2,A4) = Ø
(ii) is het een nonterminal
IF S(I2,A4) > 20 (herinner u dat er een supplementaire kode was voor nonterminale symbolen)
(iii) is het een terminal
geen konditie want het moet een terminal zijn.

S is de syntaxmatrix, I2 is de rij en A4 de kolom van een element.

We gaan nu na wat er gebeurt in de 3 gevallen.

(i) Het element is Ø

Dit betekent op de syntaxmatrix dat het nonterminale symbool volledig werd herschreven en dat we dus terug kunnen gaan waar we vandaan komen. Waar we juist vandaan komen staat op de kladblok. Dit is een PDS (last in first out) die de rij en de kolom aangeeft waar we mee bezig waren.
De PDS is gelokaliseerd op I(3) en I(4) van de informatiematrix.
Als we nu al even denken aan de OUTPUT, dan geven we hier de instruktie van een haakje uit te printen, immers we zijn aan het einde van een nonterminal.

(ii) het element is groter dan 20

Dit betekent een nonterminaal symbool. We zetten de rij en de kolom op de informatiematrix. (kladblok) We geven als output een linkerhaakje gevolgd door het nonterminale symbool (na herkodering in gebruikerskode).

(iii) het element is een terminal

Hier vergelijken we de mogelijke interpretaties van dit terminaal element met het woord uit de zin door de nogal uitvoerige funktie:

IF POS(L$ (I(1,S(I2,A4)),I(2,S(I2,A4)));C$(Q1,I1-1))> Ø

Het is bekend dat in BASIC de POS-funktie substrings opspoort. Links staat de rij uit het lexikon (L$) die voldoet aan het terminaal symbool. (Herinner u dat op de informatiematrix het begin en einde voor elk terminaal symbool, hier bekend via de syntaxmatrix S( ), werd opgeslagen). Rechts staat het woord uit de zin.

Als deze funktie groter is dan nul, geven we als output een linkerhaakje, gevolgd door het terminaal symbool in gebruikerskode (via herkodering vanuit N$) dan het woord uit de zin, gevolgd door een rechterhaakje.

Hiermee zijn we klaar. Omwille van de uitvoerige be schrijving is de 'flow chart' min of meer redundant geworden en geven we onmiddellijk het programma.
Het is jammer dat we niet beschikken over een LISP-compiler, omwille van de elegante wijze waarop recursiviteit kan uitgedrukt worden in deze programmeertaal.
In BASIC ging het echter ook, zoals men kan nagaan in onderstaand programma.

```
1230 Q3=I[2,K-20]
1240 Q2=I[1,K-20]
1250 FOR I2=Q2 TO Q3
1260 IF S[I2,A4]=0 THEN 1460
1270 IF S[I2,A4]>20 THEN 1330
1280 IF POS(L$[I[1,S[I2,A4]],I[2,S[I2,A4]]],C$[Q1,I1-1])>0 THEN 1410
1290 NEXT I2
1300 PRINT "ONGRAMMATIKAAL"
1310 STOP
1320 GOTO 1090
1330 A5=A5+1
1340 I[3,A5]=I2
1350 I[5,A5]=Q3
1360 I[4,A5]=A4
1370 K=S[I2,A4]
1380 PRINT "(";N$[((K-20)*2)-1,(K-20)*2];
1390 A4=1
1400 GOTO 1230
1410 PRINT "(";N$[((S[I2,A4]-20)*2)-1,S[I2,A4]*2];" ";C$[Q1,I1-1];")";
1420 A4=A4+1
1430 Q2=I2
1440 Q1=I1+1
1450 GOTO 1160
1460 IF A5=0 THEN 1530
1470 PRINT ")";
1480 Q2=I[3,A5]
1490 Q3=I[5,A5]
1500 A4=I[4,A5]+1
1510 A5=A5-1
1520 GOTO 1250
```

P.S. ENKELE OPMERKINGEN VOOR DE GEBRUIKER

Het programma bevindt zich op tape 21, file 5. Wanneer men het programma runt zal men verschillende mogelijkheden onderkennen tot debugging van de grammatika.

1. Voor alles is een nauwkeurige input van de gegevens vereist. Het is niet nodig spaties in te geven bij de input van de grammatika. het is WEL nodig een spatie na (niet voor) het laatste woord van de proefzin in te typen. (als het niet gebeurt, wordt het laatste woord niet ontleed)

2. Wanneer per ongeluk symbolen in de produkties voorkomen die vroeger niet werden ingevoerd, verschijnt op het scherm "SYMBOOL NIET BEKEND VOER OPNIEUW IN" Men kan dan zonder enige moeilijkheid een nieuwe produktie invoeren. De verkeerde produktie werd vergeten.

3. Na het invoeren van de syntaxis verschijnt op het scherm "TOT HIER TOE ALLES O.K. ?" . Kijk dan de syntaxmatrix na en antwoord ja of nee. Als het niet goed was, krijgt men de kans om helemaal opnieuw te beginnen. Is men verkeerd bij een produktie en wil men enkel nieuwe produkties ingeven met behoud van de symbolen, typ dan CONT (van continue) 250,EXECUTE en geef uw produkties opnieuw in.

4. Wanneer in de interpretatieregels een onbekend symbool (zowel voor de verzameling op N$ als voor de syntaxis) voorkomt,verschijnt
"LINKERSYMBOOL NIET BEKEND, VOER OPNIEUW IN"
Men kan opnieuw een interpretatieregel invoeren, de oude is weggewist.

5. Kontroleer ook de informatiematrix. Wanneer deze niet klopt, kan men nog altijd op de knop voor END duwen en teruggaan naar het begin. (RUN-EXECUTE) of naar de produkties (CONT 250-EXECUTE) of naar het lexikon (CONT 740-EXECUTE). Wanneer er bij zinnen moeilijkheden zijn en men wil opnieuw invoeren geef dan (CONT 1040- EXECUTE)
Wanneer de zin ongrammatikaal is, wordt "ONGRAMMATIKAAL" uitgeprint. Er wordt halt gehouden. Men kan dan terug een zin invoeren of naar een andere komponent gaan, zoals daarjuist werd aangegeven.

Nota: Niet geoefende gebruikers maken praktisch zeker fouten bij het ingeven van de data. Denk dus niet te snel dat het programma slecht is. Naast input fouten kunnen er nog andere fouten voorkomen zoals slechte syntaxis (wat onbehoorlijk is voor een ernstig linguïst), ongeïnterpreteerde terminale symbolen, onbekende woorden in een zin, enz...
Nogmaals nauwkeurigheid is de enige garantie tot succes.

Ter illustratie van de laatste opmerkingen geven we tweemaal een output die verkeerd loopt. Het eerste is een INPUT-fout, het tweede is een linguïstische fout. We laten het over aan de lezer van te ontdekken waar het nu precies verkeerd gaat.

```
RUN
L.STEELS 19.10.74  TAPE 21 FILE 5
SYNTAXTEST  (GENERAL CFG-PARSER)

----------
I.SYNTAXIS
----------

NONTERMINALS:SENPVPPP

TERMINALS:DTNOVEPR

V(VERZ. SYMBOLEN)=AXSENPVPPPDTNOVEPR

AANTAL NONTERMINALS: 5

PRODUKTIES:
AX => SE
SE => NP VP PP
NP => DT NO
PP => PR NP
VP => VE NP
↑

SYNTAXMATRIX:
 22    0     0     0     0
 23    24    25    0     0
 6     7     0     0     0
 9     23    0     0     0
 8     23    0     0     0
----------
2.LEXIKON
----------

DT => DEHETEENGEEN
NO =>JONGENPOESLIEDJEKAMERMUISHUIS
PR => METOPINUITAAN
VE => VERJOEGZINGT
↑
LEXIKON:
DEHETEENGEENONGENPOESLIEDJEKAMERMUISHUISMETOPINUITAANVERJOEGZINGT

INFORMATIEMATRIX:
 1     2     3     5     4     1     13    54    41
 1     2     3     5     4     13    41    66    54

------
3.TEST
------
DE JONGEN ZINGT EEN LIEDJE OP DE KAMER

SYNT.ANAL. :
(SE(NP(DT DE)ONGRAMMATIKAAL


L.STEELS 19.10.74  TAPE 21 FILE 5
SYNTAXTEST  (GENERAL CFG-PARSER)

----------
I.SYNTAXIS
----------

NONTERMINALS:SENPVPPP

TERMINALS:DTNOVEPR

V(VERZ. SYMBOLEN)=AXSENPVPPPDTNOVEPR

AANTAL NONTERMINALS: 5
```

```
PRODUKTIES:
AX => SE
SE => NP VP NP PP
NP => DT NO
VP => VE NP
PP => PR NP
↑

SYNTAXMATRIX:
 22     0      0      0      0
 23     24     23     25     0
 6      7      0      0      0
 8      23     0      0      0
 9      23     0      0      0
```

2.LEXIKON

```
DT => DEHETEENGEEN
PR => METOPINUITAAN
NO => JONGENPOESAPENOOTJEKAMERLIEDJEMUISHUIS
VE => VERJOEGZINGT
↑
LEXIKON:
DEHETEENGEENMETOPINUITAANJONGENPOESAPENOOTJEKAMERLIEDJEMUISHUISVERJOEGZI

INFORMATIEMATRIX:
 1      2      3      4      5      1      26     64     13
 1      2      3      4      5      13     64     76     26
```

3.TEST

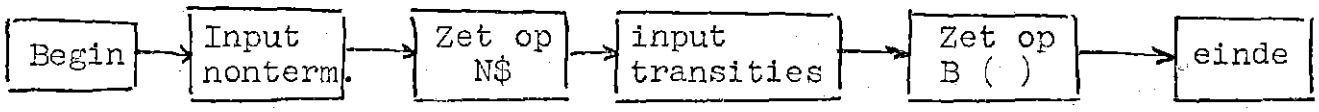
```
DE JONGEN ZINGT EEN LIEDJE OP DE KAMER

SYNT.ANAL. :
(SE(NP(DT DE)(NO JONGEN))(VP(VE ZINGT)(NP(DT EEN)(NO LIEDJE))(NP
ONGRAMMATIKAAL
```

APPENDIX III.

Kodering

1. Nonterminale symbolen en transities

De flow chart van het programma:



Enkel de nonterminale symbolen worden dus expliciet ingevoerd bij het begin, zij komen op N$, de terminale worden afgeleid uit de interpretatieregels.

```
80 DISP "VOER NONTERMINALE SYMBOLEN IN";
90 INPUT N$
100 PRINT "NONTERMINALS: ";N$
110 MAT D=ZER
120 DISP "HOEVEEL TRANSITIES ";
130 INPUT T
140 PRINT "INPUT TRANSITIES :"
150 FOR I=1 TO T
160 DISP "VOER TRANSITIE IN";
170 A$=" "
180 INPUT A$
190 FOR J=1 TO 6
200 TRANSFER A$[(J*2)-1,J*2] TO B[I,(J*2)-1]
210 PRINT A$[(J*2)-1,J*2];
220 NEXT J
230 PRINT
240 NEXT I
```

2. Interpretaties:

```
250 DISP "VOER INTERPRETATIEREGELS IN"
260 WAIT 1000
270 PRINT "INPUT INTERPRETATIES: "
280 K=0
290 DISP "INPUT TERMINAAL SYMBOOL";
300 K=K+1
310 INPUT D$[(K*2)-1,K*2]
320 PRINT D$[(K*2)-1,K*2];" = ";
330 IF D$[(K*2)-1,K*2]="↑ " THEN 410
340 DISP "INTERPRETATIE";
350 A$=" "
360 INPUT A$
370 PRINT A$
380 TRANSFER A$ TO C[K,1]
390 TRANSFER C[K,1] TO A$
400 GOTO 290
```

3. De konstruktie van de transitiematrix

```
410 FOR F1=1 TO T
420 FOR F3=1 TO 3
430 IF B[F1,(F3*4)-1]#12320 THEN 520
440 TRANSFER B[F1,(F3*4)-3] TO A$[1,2]
450 FOR F2=1 TO K-1
460 IF D$[(F2*2)-1,F2*2]=A$[1,2] THEN 490
470 NEXT F2
480 STOP
490 A[F1,(F3*2)-1]=F2
500 A[F1,F3*2]=0
510 GOTO 600
520 TRANSFER B[F1,(F3*4)-3] TO A$[1,2]
530 FOR F2=1 TO LEN(N$)/2
540 IF A$[1,2]=N$[(F2*2)-1,(F2*2)] THEN 570
550 NEXT F2
560 STOP
570 A[F1,(F3*2)-1]=F2
580 TRANSFER B[F1,(F3*4)-1] TO A$[1,2]
590 A[F1,F3*2]=VAL(A$[1,2])
600 NEXT F3
610 NEXT F1
```

APPENDIX IV

De parsingalgoritmen

```
710 INPUT A$
720 PRINT "AXIOMA :";A$
730 PRINT
740 FOR F3=1 TO LEN(N$)/2
750 IF N$[(F3*2)-1,F3*2]=A$[1,2] THEN 780
760 NEXT F3
770 STOP
780 D[1,1]=1
790 D[1,2]=F3
800 D[1,3]=1
810 D[1,5]=1
820 D[1,6]=1
830 DISP "INPUT PROEFZIN";
840 INPUT C$
850 IF C$[1,6]="GEDAAN" THEN 2060
860 S=0
870 I3=2
880 W=0
890 FOR I2=2 TO LEN(C$)
900 IF C$[I2,I2]#" " THEN 950
910 W=W+1
920 P[W]=I3
930 E[W]=I2-1
940 I3=I2+1
950 NEXT I2
960 PRINT "INPUT:";C$
970 PRINT
980 I1=0
990 I2=1
1000 PRINT
1010 PRINT "KLADBLOK:"
1020 GOTO 1440
1030 DISP "TERMINAAL"
1040 TRANSFER C[A[I3,1],1] TO A$
1050 IF POS(A$,C$[P[D[I1,1]]-1,E[D[I1,1]]+1])=0 THEN 1140
1060 GOSUB 1160
1070 I2=I2+1
1080 DISP "VOLGEND WOORD"
1090 D[I2,1]=D[I1,1]+1
1100 D[I2,2]=A[I3,5]
1110 D[I2,3]=A[I3,6]
1120 D[I2,4]=I2-1
1130 D[I2,6]=D[I1,6]
1140 RETURN
1150 DISP "NONTERM"
1160 I2=I2+1
1170 D[I2,1]=D[I1,1]
1180 D[I2,2]=A[I3,1]
1190 D[I2,3]=A[I3,2]
1200 D[I2,4]=I1
1210 IF I1=1 THEN 1240
1220 IF D[I2,3]#1 THEN 1270
1230 IF D[D[I1,6],2]=D[I2,2] THEN 1270
1240 D[I2,6]=I2
1250 D[I2,5]=I3
1260 GOTO 1280
1270 D[I2,6]=D[I1,6]
1280 RETURN
```

```
1290 DISP "EINDE NONTERMINAAL"
1300 IF D[I1,2]#D[1,2] THEN 1360
1310 IF D[I1,3]#50 THEN 1360
1320 IF D[I1,1]#W+1 THEN 1430
1330 S=S+1
1340 S[S]=I1
1350 GOTO 1440
1360 I2=I2+1
1370 D[I2,1]=D[I1,1]
1380 D[I2,2]=A[D[D[I1,6],5],5]
1390 D[I2,3]=A[D[D[I1,6],5],6]
1400 D[I2,4]=I1
1410 D[I2,5]=D[D[I1,6],5]
1420 D[I2,6]=D[D[D[I1,6],4],6]
1430 RETURN
1440 IF I1=I2 THEN 1680
1450 I1=I1+1
1460 FOR Z=1 TO 6
1470 PRINT D[I1,Z];
1480 NEXT Z
1490 PRINT
1500 DISP "VOLGENDE TAAK"
1510 IF D[I1,3]=0 THEN 1440
1520 IF D[I1,3]#50 THEN 1550
1530 GOSUB 1290
1540 GOTO 1440
1550 FOR I3=1 TO T
1560 IF A[I3,3]#D[I1,2] THEN 1650
1570 IF A[I3,4]#D[I1,3] THEN 1650
1580 IF A[I3,2]#0 THEN 1620
1590 IF D[I1,1]>W THEN 1650
1600 GOSUB 1030
1610 GOTO 1650
1620 GOSUB 1150
1630 IF I3+1 >= T THEN 1650
1640 IF A[I3+1,3]#D[I1,2] THEN 1440
1650 NEXT I3
1660 GOTO 1440
1670 STOP
```

De output procedure

```
1680 PRINT
1690 PRINT "AANTAL PARSINGS:";S
1700 IF S#0 THEN 1730
1710 PRINT "INGEVOERDE ZIN IS ONGRAMMATIKAAL"
1720 PRINT
1730 FOR Q=1 TO S
1740 PRINT "STRUKTURELE DESKRIPTIE";Q
1750 L=S[Q]
1760 PRINT "PAD DOOR KLADBLOK:"
1770 FOR R=1 TO I2
1780 R[R]=L
1790 PRINT R[R];
1800 IF L=1 THEN 1850
1810 L=D[L,4]
1820 NEXT R
1830 PRINT
1840 PRINT
1850 PRINT "PARSING:"
1860 FOR U=1 TO R
1870 Y=R-U+1
1880 IF D[R[Y],3]=50 THEN 1970
1890 IF D[R[Y],3]=0 THEN 1940
1900 IF D[R[Y],3]#1 THEN 1980
1910 IF D[R[Y],6]#R[Y] THEN 1980
1920 PRINT "(";N$[(D[R[Y],2]*2)-1,D[R[Y],2]*2];
1930 GOTO 1980
1940 PRINT "(";D$[(D[R[Y],2]*2)-1,D[R[Y],2]*2];" ";
1950 PRINT C$[P[D[R[Y],1]],E[D[R[Y],1]]];")";
1960 GOTO 1980
1970 PRINT ")";
1980 NEXT U
1990 PRINT
2000 PRINT
2010 NEXT Q
2020 PRINT
```