7280 ML
ZW

STICHTING
MATHEMATISCH CENTRUM
2e BOERHAAVESTRAAT 49
AMSTERDAM
AFDELING ZUIVERE WISKUNDE

ZW 1969-005

Voordracht in de serie

"Elementaire onderwerpen vanuit hoger standpunt belicht"

door

Prof.Dr.Ir. A.J.W. Duyvesteyn

Zinsontleding

Inleiding

Ieder van U herinnert zich nog wel het probleem om een zin redekundig te moeten ontleden. Het ging er daarbij om van een gegeven zin aan te geven welk gedeelte het onderwerp, het gezegde, het lijdend voorwerp en het medewerkend voorwerp was. We hadden daar meestal niet veel moeite mee. Toch was het moeilijk aan te geven hoe we dat nu precies deden. Neem eens het voorbeeld: "De man geeft een das aan zijn vrouw".

Hierin is	de man	- onderwerp
	geeft	- gezegde
	een das	- lijdend voorwerp
	aan zijn vrouw	- medewerkend voorwerp

ZW
BIBLIOTHEEK MATHEMATISCH CENTRUM
AMSTERDAM

Een zin die uit dezelfde woorden bestaat: "Aan zijn vrouw geeft de man een das" kan ook in dezelfde vier stukken ontleed worden. Overigens geldt in beide voorbeelden dat de ontleding slechts op één manier mogelijk is. De zin is syntactisch ondubbelzinnig. Voorbeelden van syntactisch dubbelzinnige zinnen zijn: "De jongens houden van schatten minnen en plussen".
"Time flies like arrows".
De leer van de syntax is de leer van de zinsbouw. Deze zegt ons welke symbolen achter elkaar mogen voorkomen. Aan de symbolenrij kan een betekenis of semantiek toegekend zijn. Er bestaan syntactisch ondubbelzinnige en semantisch dubbelzinnige zinnen. Een voorbeeld is: "De man zit in de bak".

Voorbeelden van zinnen uit een geheel andere taal zijn:

a + b*(a + b)
a*b + c*d
a + b*(a + d)

Het zijn rekenkundige uitdrukkingen waarin de operatoren +,* voorkomen en waar eveneens haakjesparen worden gebruikt.

Uit de rekenkunde weten we dat: een vorm a*b uit twee factoren bestaat, terwijl a + b uit twee termen bestaat. De ontleding van een dergelijke zin bestaat erin dat we aangeven waar de termen, factoren etc. zich bevinden.

We zullen nu een poging ondernemen het geheel te formaliseren. We zullen ons daarbij concentreren op de ontleding in rekenkundige uitdrukkingen.

Syntactische definitie.

We voeren daartoe een aantal productieregels in (soms worden deze syntactische regels genoemd).

1.	Z→expressie	Z→E
2.	expressie→expressie + term	E→E+T

3.	expressie→term	E→T
4.	term→term*factor	T→T*F
5.	term→factor	T→F
6.	factor→ (expressie)	F→(E)
7.	factor→a	F→a
8.	factor→b	F→b
9.	factor→c	F→c
10.	factor→d	F→d

We gaan uit van het zgn. initiële symbool Z. Beschouwen we de productieregels dan kan Z worden vervangen door E. We hebben dan een symbolenrij bestaande uit één symbool nl. E. Daarna kan E worden vervangen. Er zijn twee alternatieven nl.

E→E+T of E→T

Laten we regel 2 eens kiezen. We hebben dan een symbolenrij E+T. Deze rij symbolen (nu bestaande uit 3 symbolen) noemen we een string. In deze string kunnen we hetzij E of T vervangen door gebruik te maken van een van de productieregels. Het symbool + kan niet worden vervangen. Het is een eindsymbool of terminal symbool. De symbolen die wel vervangen kunnen worden zijn non-terminal symbolen. Op een gegeven moment kan het zijn dat de string die van Z wordt afgeleid (een derivaat) geen non-terminal symbolen meer bevat. Er is dan een zin ontstaan of gegenereerd. De tussenstrings (met non-terminals) heten zinsvormen.

Voorbeeld 1: Generatierij Z

E	toegepaste regel =	1
E+T		2
T+T		3
T*F+T		4
T*F+T*F		4
F*F+T*F		5
a*F+T*F		7
a*b+T*F		8
a*b+F*F		5
a*b+c*F		9
a*b+c*d		10

Voorbeeld 2: Generatierij Z

E

T

T*F

F*F

(E)*F

(E+T)*F

(T+T)*F

(F+T)*F

(a+T)*F

(a+F)*F

(a+b)*F

(a+b)*c

De verzameling van terminal symbolen duiden we aan met $\underline{T}$.
De verzameling van non-terminal symbolen duiden we aan met $\underline{V}-\underline{T}$, terwijl de totale verzameling $\underline{V}$ (de woordenschat of alphbet) is. In het geval van rekenkundige uitdrukkingen geldt:

$$\underline{T} = \{+,*,(,),a,b,c,d\}$$
$$\underline{V} - \underline{T} = \{E,T,F,Z\}$$

De verzameling van productieregels noemen we $\underline{R}$.
Een formele definitie van een grammatica G is de volgende:

G is een viertal bestaande uit: $G=(\underline{V},\underline{T},\underline{R},Z)$.

Eerst behandelen we nog een paar notaties en definities.

Notaties: $\alpha,\beta,\gamma, \ldots \in \underline{V}$ $\underline{V}$ is woordenschat of alphabet.
$a,b,c, \ldots \in \underline{T}$ $\underline{T}$ is verzameling terminal symbolen

Aan de verzameling $\underline{V}$ voegen we een verzameling $\underline{V}^*$ toe met elementen ϕ,ψ,ξ als volgt:
1. Als $\alpha \in \underline{V}$ dan ook $\alpha \in \underline{V}^*$.

2. We definiëren een tweezijdige operator $\oplus$ op $\underline{V}^*$ en we veronderstellen dat

a. $\underset{\phi\in\underline{V}^*,\psi\in\underline{V}^*}{\forall}$ $\phi\oplus\psi\in\underline{V}^*$

b. $\underset{\phi\in\underline{V}^*,\psi\in\underline{V}^*,\xi\in V^*}{\forall}$ $(\phi\oplus\psi)\oplus\xi=\phi\oplus(\psi\oplus\xi)$

c. $\underset{e\in\underline{V}}{\exists}\ \underset{\phi\in\underline{V}^*}{\forall}$ $e\oplus\phi=\phi\oplus e=\phi$

Verzameling $\underline{V}^*$ met operator $\oplus$ waarvoor de eigenschappen a en b gelden worden semigroepen genoemd, terwijl verzamelingen $\underline{V}^*$ met operator $\oplus$ waarvoor a, b en c gelden monoïden heten. Hierdoor ontstaan strings van symbolen $\alpha,\beta,\gamma, \ldots \in \underline{V}$ die dan elementen van $\underline{V}^*$ zijn. Deze elementen hebben we met ϕ,ψ,ξ aangeduid. Uiteraard zijn strings van ϕ, ψ en ξ weer elementen van $\underline{V}^*$. Het element e is nu de lege string die we voortaan met ε zullen aanduiden.

$\underline{V}^*$ wordt de vrije monoïde over $\underline{V}$ (gegenereerd door $\underline{V}$) genoemd. Op dezelfde wijze voeren we strings van symbolen $a,b,c,V,\ldots\in\underline{T}$ in die dan elementen zijn van $\underline{T}^*$. Elementen van $\underline{T}^*$ duiden we aan met x,y,z, ...
Evenzo met $\underline{V} - \underline{T}$. De elementen hiervan zijn Z,A,B,C, ...
De elementen van $(\underline{V} - \underline{T})^*$ zijn X,Y,W,V, ...
Z is weer het initiële symbool.

Tabel

$\alpha,\beta,\gamma, \ldots$	$\in \underline{V}$
$\phi,\psi,\xi,\eta, \ldots$	$\in \underline{V}^*$
a,b,c, ...	$\in \underline{T}$
x,y,z, ...	$\in \underline{T}^*$
Z,A,B,C, ...	$\in \underline{V} - \underline{T}$
V,W,X,Y, ...	$\in (\underline{V} - \underline{T})^*$

Door het toepassen van productieregels worden strings in elkaar overgevoerd.

We schrijven

$$\phi \rightarrow \psi;\ \phi \in \underline{V}^{*} - \{\varepsilon\}$$

ϕ is het linkerlid, ψ het rechterlid in de productieregel.
Als $\xi_1, \xi_2 \in \underline{V}^{*}$ dan verstaan we onder

$$\xi_1 \overset{}{\underset{G}{\Rightarrow}} \xi_2$$

$$\exists_{\sigma,\tau,\phi,\psi \in V^{*}} \quad \xi_1 = \sigma\phi\tau,\ \xi_2 = \sigma\phi\tau, \phi \rightarrow \psi \in \underline{R}$$

($\underline{R}$ is de verzameling productieregels).

onder $\xi_1 \overset{*}{\underset{G}{\Rightarrow}} \xi_2$ verstaan we

een rij $\omega_1, \omega_2, \ldots, \omega_n$ $(\omega \geq 1)$ $\omega_1 = \xi_1$, $\omega_n = \xi_2$ en

$$\omega_i \overset{}{\underset{G}{\Rightarrow}} \omega_{i+1} \quad (1 \leq i \leq n-1).$$

We zeggen dat ξ_2 uit ξ_1 gegenereerd wordt.

Een $\phi \in \underline{V}^{*}$ zodanig dat $Z \overset{*}{\underset{G}{\Rightarrow}} \phi$ heet zinsvorm.
Als $\phi \in \underline{T}^{*}$ met $Z \overset{*}{\underset{G}{\Rightarrow}} \phi$ dan is ϕ een zin.
We kunnen nu de formele definitie van een taal L_G door G voortgebracht definiëren.

$$L_G = \{x \mid Z \overset{*}{\underset{G}{\Rightarrow}} x \mid x \in \underline{T}^{*}\}$$

Voorbeeld:

$\underline{V} = \{Z, D, a, b, c\}$
$T = \{a, b, c\}$
$Z = Z$

$\underline{R}$ bevat de volgende regels:

$Z \rightarrow aZD$
$Z \rightarrow abc$
$bD \rightarrow bbc$
$cD \rightarrow DC$

Voorbeeld: generatie van een zin

Z
aZD
aabcD
aabDC
aabbcc

Stelling: bovengenoemde grammatica genereert de taal

$$L_{GI} = \{a^n b^n c^n \mid n \geq 1\}.$$

Structuur beschrijving van een zin.

Een zin $s \in \underline{T}^*$ die wordt verkregen uit $Z \overset{*}{\Rightarrow} s$ kan mogelijk op meer manieren verkregen worden.
We willen nu de weg die gevolgd is om s te genereren vastleggen.
We zoeken een kanonieke beschrijving. De weg terug noemen we de kanonieke reductie.
In principe streven we ernaar steeds wanneer we een string van terminals en non-terminals verkregen hebben zo veel mogelijk naar links in de string een productieregel resp. een reductieregel toe te passen.

Een generatie-volgorde $\omega_1, .., \omega_n$; $n < 1$ is een kanonieke volgorde als:

$$\forall_i, 1 \leq i \leq n-1 \; \exists x_i \in \underline{T}^*, A_i \in \underline{V}-\underline{T}, \tau_i, \phi_i \in \underline{V}^* \left[\omega_i = x_i A_i \tau_i, \; \omega_{i+1} = x_i \phi_i \tau_i \text{ en } A_i \to \phi_i \in \underline{R}\right].$$

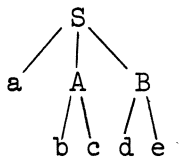
Op een soortgelijke wijze definiëert men een kanonieke reductie volgorde $\omega_n, \omega_{n-1}, \ldots, \omega_1$ door:

$$\forall_i, 1 \leq i \leq n-1 \; \exists y_i \in \underline{T}^*, A_i \in \underline{V}-\underline{T}, \tau_i, \phi_i \in \underline{V}^* \left[\omega_{i+1} = \tau_i \phi_i y_i, \; \omega_i = \tau_i A_i y_i\right],$$

waarbij $A_i \to \phi_i \in \underline{R}$.

Voorbeeld: S → a A B
A → bc
B → de

in figuur:



De reductie-volgorde:

ω_4 = a bc de met $y_3 = \varepsilon$, ϕ_3 = de, τ_3 = abc
ω_3 = a bc B y_2 = B, ϕ_2 = bc, τ_2 = a, waarbij $y_2 \notin \underline{T}^*$
ω_2 = a A B $y_1 = \varepsilon$, ϕ_1 = a A B, $\tau_1 = \varepsilon$
ω_1 = S

is wegens $y_2 \notin \underline{T}^*$ niet kanoniek.

De volgorde:

ω_4 = a bc de met y_3 = de, ϕ_3 = bc, τ_3 = a
ω_3 = a A de $y_2 = \varepsilon$, ϕ_2 = de, τ_2 = a A
ω_2 = a A B $y_2 = \varepsilon$, ϕ_2 = a A B, $\tau_2 = \varepsilon$

is wèl kanoniek.

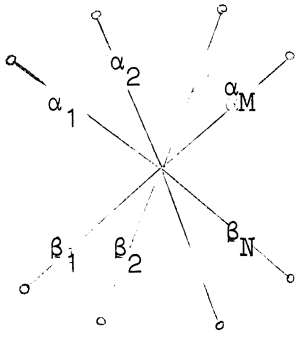
Structuur beschrijving van de syntactische graph.

De syntactische graph is een representatie van de generatie resp. reductie. Aan iedere toegepaste productie voegen we een knooppunt toe. Als bij het toepassen van deze productie de string $\phi \Rightarrow \psi$, en we veronderstellen

$\phi = \alpha_1\ \alpha_2 \ldots, \alpha_M \Rightarrow \beta_1\ \beta_2 \ldots\ \beta_N = \psi$

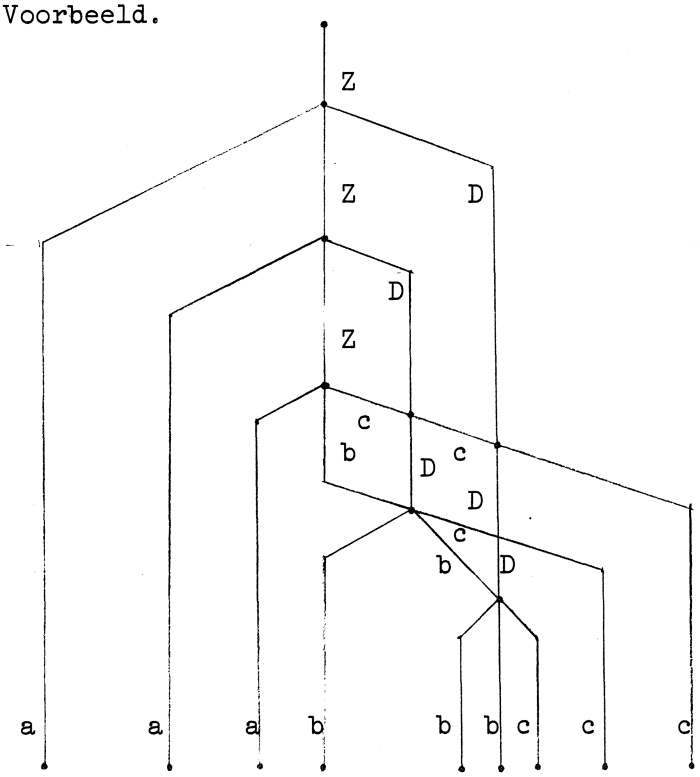
met $\alpha_i, \beta_i \in \underline{V}$; $M > 0$, $N \geq 0$.

dan voegen we aan iedere α_i en β_i een tak toe waarvan een knooppunt het knooppunt is behorende bij de toegepaste productie.



Het is duidelijk dat bij iedere kanonieke generatie en reductie één syntactische graph behoort.

Voorbeeld.



syntactische graph behorende by grammatica

$Z \rightarrow aZD$

$Z \rightarrow abc$

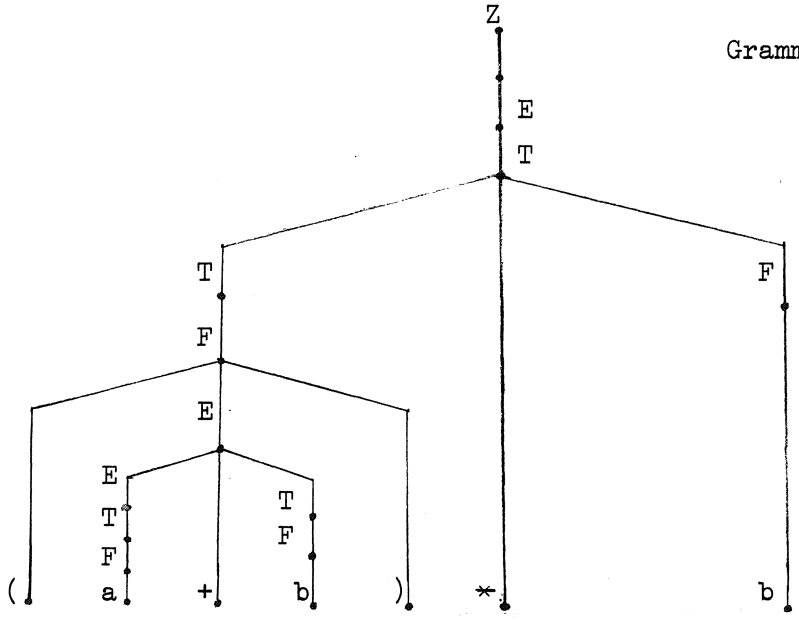
$cD \rightarrow Dc$

$bD \rightarrow bbc$

en de generatie $Z \overset{*}{\Rightarrow} a^3b^3c^3$

De syntactische graph reduceert tot een syntactische boom in het volgende voorbeeld.

Voorbeeld van syntactische boom:



Grammatica:

$Z \rightarrow E$

$E \rightarrow T$

$E \rightarrow E+T$

$T \rightarrow F$

$T \rightarrow TxF$

$F \rightarrow (E)$

$F \rightarrow a$

$F \rightarrow b$

zin: $(a+b) * b$

Voorbeelden van kanonieke generatie:

Grammatica: Z → aZD
Z → abc
cD → Dc
bD → bbc

string $\underline{V}^*$	toegepaste regel
$\underline{Z}$	
a$\underline{Z}$D	Z → aZD
aa$\underline{ZD}$D	Z → aZD
aaabc$\underline{D}$D	Z → abc
aaab$\underline{D}$cD	cD → Dc
aaabbcc$\underline{D}$	bD → bbc
aaabbc$\underline{D}$C	cD → Dc
aaabb$\underline{D}$cc	cD → Dc
aaabbbccc	bD → bbc

grammatica: Z → E
E → T
E → E+T
T → F
T → T*F
F → (E)
F → a
F → b

string	toegepaste regel
$\underline{Z}$	
$\underline{E}$	Z → E
$\underline{T}$	E → T
$\underline{T}$*F	T → T*F
$\underline{E}$*F	T → F
($\underline{E}$)*F	F → (E)
($\underline{E}$+T)*F	E → E+T
($\underline{T}$+T)*F	E → T
($\underline{F}$+T)*F	T → F
(a+$\underline{T}$)*F	F → a
(a+$\underline{F}$)*F	T → F
(a+b)*F	F → b
(a+b)*b	F → b

Voorbeelden kanonieke reductie.

Van de twee bovenstaande grammatica's zullen we nu 2 voorbeelden van kanonieke reducties bekijken.

string:	toegepaste regels: (de productieregels zijn voor het gemak omgekeerd tot reductieregels)
aaabbbccc	
aaabbDcc	bbc → bD
aaabbcDc	Dc → cD
aaabDDc	bbc → bD
aaabDcD	Dc → cD
aaabcDD	Dc → cD
aa Z DD	abc → Z
a Z D	aZD → Z
Z	aZD → Z

Het kan ook niet-kanoniek:

aaabbbccc	
aaabbDcc	bbc → bD
aaabbcDc	Dc → cD
aaabbccD	Dc → cD
aaabDcD	bbc → bD

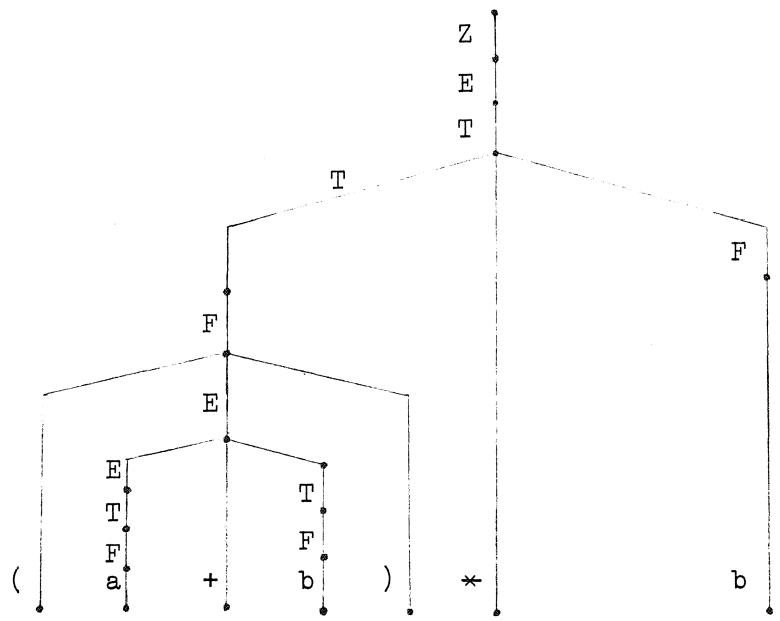
enz.

	string:	
1.	a*b + a*(a+b)	
2.	F*b + a*(a+b)	a → F
3.	T*b + a*(a+b)	F → T
4.	T*F + a*(a+b)	b → F
5.	T + a*(a+b)	T*F → T
6.	E + a*(a+b)	T → E
7.	E + F*(a+b)	a → F
8.	E + T*(a+b)	F → T
9.	E + T*(F+b)	a → F
10.	E + T*(T+b)	F → T
11.	E + T*(E+b)	T → E
12.	E + T*(E+F)	b → F
13.	E + T*(E+T)	F → T
14.	E + T*(E)	E+T → E
15.	E + T*F	(E) → F
16.	E + T	T*F → T
17.	E	E+T → E
18.	Z	E → Z

Ook hier zien we, dat de benodigde reductieregels zó zijn toegepast dat er rechts van een te reduceren stuk string alleen terminals (inclusief ε) staan. We kunnen dezelfde zin ook wel met dezelfde reductieregels in een andere volgorde reduceren. Bijv. zou de reductie van (a+b) tot F wel eerst plaats hebben kunnen vinden. De reductie is dan echter niet kanoniek (wèl eenduidige structuur: zelfde boom).
Als we ervan spreken dat steeds de meest linkse reductieregel moet worden toegepast, wordt natuurlijk niet bedoeld de meest linkse die op een gegeven ogenblik überhaupt mogelijk is (bijv. in regel 6 t/m 16 E door Z vervangen, of in regel 16 T door E, die dan nog "kanoniek" is ook), maar de meest linkse die nog wel tot een volledige reductie tot Z leidt en niet doodloopt.
M.a.w. van parallelle "goede" mogelijkheden moet de meest linkse genomen worden.

Representatie van de syntactische graph.

We beginnen met de representatie van een syntactische boom die bij een context-vrije taal optreedt (voor def. van context-vrije taal zie Chomsky).



We zullen een tweetal representaties bepreken.

1: top-to-bottom representatie.

We starten bij de tak die toegevoegd is aan het initiële symbool en construeren een weg door de boom als volgt:
Als we aankomen bij een knooppunt gaan we met dié tak verder die bereikt wordt door links cyclisch rond te gaan tot dat we een tak tegenkomen.
Bij een knooppunt van de graad 1 gaan we terug doordat we dezelfde tak tegenkomen. De weg wordt zover voortgezet tot dat we het initiële symbool weer tegenkomen.
Bij het passeren van takken van de weg nemen we bepaalde aan deze takken toegevoegde elementen van V in een string op.
Bij de top-to-bottom representatie wordt het toegevoegde element opgenomen als de bijbehorende tak voor de eerste maal wordt tegengekomen.
In het voorbeeld krijgen we:

Z,E,T,T,F,(,E,E,T,F,a,+,T,F,b,),*,F,b.

Uit deze representatie kunnen we de oorspronkelijke boom echter niet afleiden.
Dit kan wel als iedere non-terminal vervangen wordt door de productie die wordt toegepast bij iedere tak. We krijgen dan

Z → E, E → T, T → T*F, T → F, F → (E), (, E → E+T, E → T, T → F, F → a, a, +, T → F, F → b, b,), *, F → b, b.

We kunnen deze string hercoderen door de producties te nummeren en de producties door hun nummer te vervangen.
Als de nummering als volgt is:

1 Z → E
2 E → T
3 E → E+T
4 T → F
5 T → T*F
6 F → (E)
7 F → a
8 F → b

dan wordt de top-to-bottom representatie:

1, 2, 5, 4, 6, (, 3, 2, 4, 7, a, +, 4, 8, b,), *, 8, b.

De bottom-to-top representatie.

Deze verschilt van de top-to-bottom representatie doordat de tak pas in de verzameling wordt opgenomen als we deze voor de laatste keer tegenkomen. In plaats van de tak wordt de productie weer genomen als de tak geen terminal representeert.
We krijgen dan in het bovenstaande voorbeeld:

(, a, F → a, T → F, E → T, +, b, F → b, T → F, E → E+T,), F → (E), ×, b, F → b, T → T×F, E → T, Z → E.

of gecodeerd:

(, a, 7, 4, 2, +, b, 8, 4, 3,), 6, ×, b, 8, 5, 2, 1.

Deze representaties stellen de boom eenduidig voor. Ze kunnen uit elkaar afgeleid worden.
Opgave: Schrijf een ALGOL-60 programma dat de top-to-bottom representatie over voert naar de bottom-to-top representatie en een programma dat het omgekeerde doet door de representatie als array op te vatten.

De Chomsky-phrase-marker.

Deze lijkt sterk op de bottom-to-top representatie.
Er wordt ingevoerd het symbool:

$[\text{string}]_A$ dit betekent string komt voort uit A.

We krijgen voor ons voorbeeld dan:

$$[\,[\,[\,[\,[\,[\,(\,[\,[\,[\,a]_F\,]_T\,]_E + [\,[\,b]_F\,]_T\,]_E\,)\,]_F\,]_T * [\,b]_F\,]_T\,]_E\,]_Z$$

Door de vierkante haken te vervangen door ronde haken en de subscripts op de regel te plaatsen krijgen we een lijst zoals we deze uit LISP kennen.

Dubbelzinnigheid.

Een zin is structureel dubbelzinnig als er meer dan één structuurbeschrijving van deze zin bestaat.
Een taal wordt structureel dubbelzinnig genoemd als deze één of meer zinnen bevat die structureel dubbelzinnig zijn.

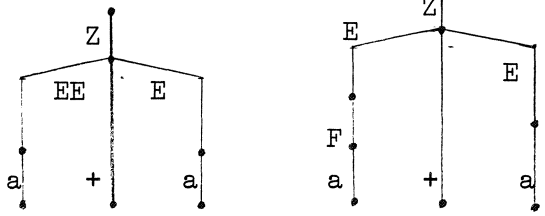
Voorbeeld:

Grammatica : Z → E+E
Z → a
E → F
F → a

Generatie zin: dezelfde zin wordt kanoniek gegenereerd door:

Z	Z
E+E	E+E
a+E	F+E
a+a	a+E
	a+a

bijbehorende boom:



In feite hebben we hier met een syntactische dubbelzinnigheid te maken. De taal kan ook nog semantisch dubbelzinnig zijn. Dit is nog moeilijk te formaliseren.

Voorbeelden uit de natuurlijke taal.

Syntactisch en semantisch dubbelzinnig.

De jongens willen schatten minnen en plussen.
They are flying planes.
Dat is de vrouw die de man een boek gaf.

Syntactisch ondubbelzinnig en semantisch dubbelzinnig.

Hij zit in de bak.

kan betekenen a: Hij zit in de nor.
b: Hij zit in de bak (die b.v. water kan bevatten).

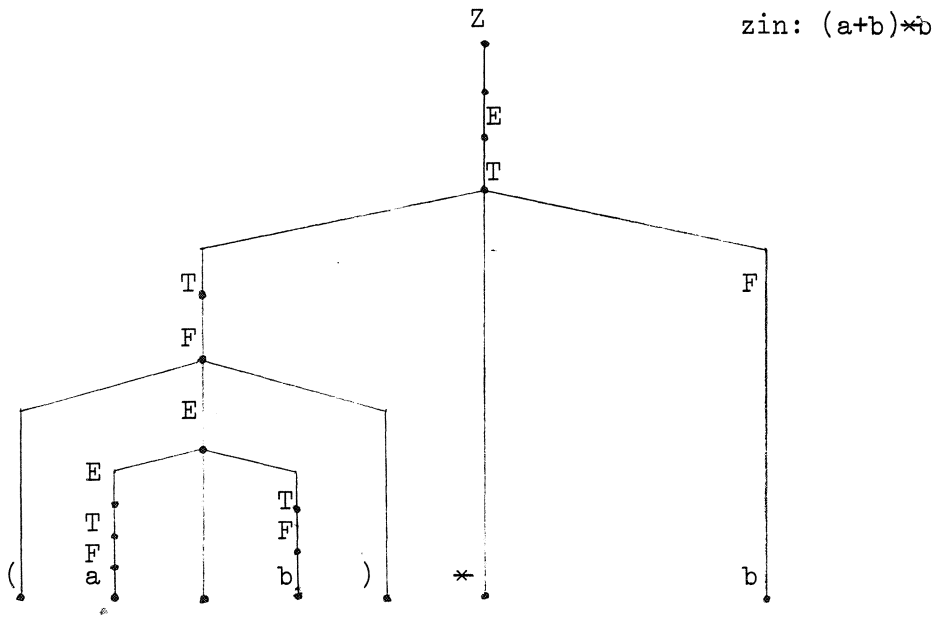
Syntactische analyse:

We zullen ons nu bezighouden met de grammatica $G_{II} = (\underline{V},\underline{T},\underline{R},Z)$
met $\underline{V}$ = {Z,E,T,F,+,*,(,),a,b,c,d}
$\underline{T}$ = {+,*,(,),a,b,c,d}
Z = Z

en $\underline{R}$ bevat de regels

Z → E
E → E+T
E → T
T → T*F
T → F
F → (E)
F → a
F → b
F → c
F → d

Speciaal houden we ons bezig met de herkenning van zinnen door G_{II} gegenereerd. Het gaat er om om te ontdekken welke structuurbeschrijving de zin heeft.



zin: (a+b)*b

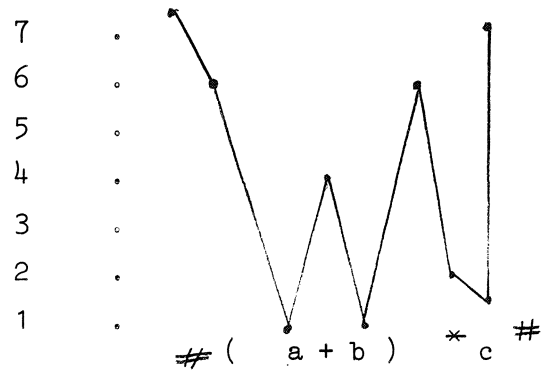
We zullen dit eerst proberen op een ad hoc manier door aan de symbolen een getalwaarde toe te kennen.

Na enig proberen komen we op

(	6
)	6
*	2
+	4
a	1
b	1
c	1
d	1
F	1
T	3
E	5
Z	1
#	7

N.B. # geeft het begin en einde van een zin aan.

Voor de zin (a+b)*c tekenen we het volgende plaatje:

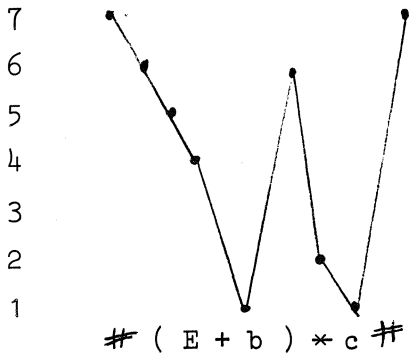


We gaan nu proberen de weg terug te vinden naar het symbool Z door de productieregels in omgekeerde richting toe te passen.

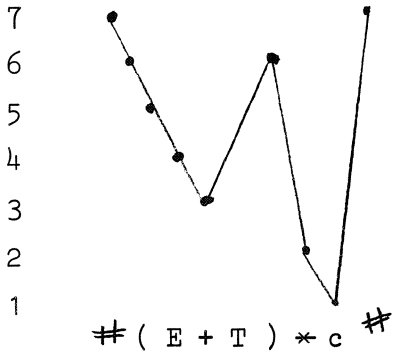
Bekijken we de figuur dan lijkt het erop dat de volgende procedure uitkomst kan bieden. Zoek van links naar rechts naar het eerste dal in de figuur. Dat treedt op bij a. Reduceer a tot F (toepassing F→a in omgekeerde richting).

De nieuwe string is #(F+b)*c#. Hiervan kunnen we weer een plaatje tekenen. We krijgen exact hetzelfde tekeningetje. We besluiten daaruit dat F tot T wordt gereduceerd. Daarna T tot E.

We krijgen dan het volgende beeld.

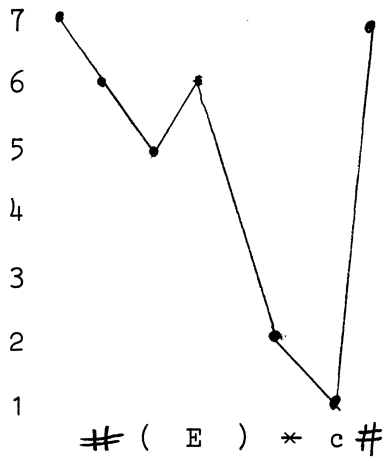


Hierna passen we achter elkaar de reductie b→F; F→T toe waardoor we het volgende beeld krijgen.



Nu ligt het dichtstbijzijnde dal bij T. In aanmerking komen T→E of E+T*E. We weten dat we de laatste regel moeten toepassen.
In het eerste geval hebben we te maken met een reductie waarbij geen terminal symbolen betrokken zijn. E heeft niveau5. We voeren daarom de regel in dat slechts een zodanige reductieregel mag worden toegepast dat de ontstane non-terminal een niet hoger niveau heeft dan welk symbool dan ook dat zich links van hem bevindt.

We passen dus de regel E + T → E toe en krijgen dan:



We lopen nu toch vast daar we (E) → F moeten toepassen.
Waarom zouden we dat doen?
We breiden het systeem uit met twee functies f en g als volgt:

	f	g
a,b,c,d	1	1
+	4	5
*	2	3
(	6	1
)	2	5
F	1	1
T	3	1
E	5	1
Z	1	1
#	6	6

We proberen het volgende:
In het rijtje $\alpha_1 \ldots \alpha_n$ kan een reductie toegepast worden als in
$\alpha_1 \ldots \alpha_n \beta_1$

geldt $f(\alpha_n) < g(\beta_1)$ en $f(\alpha_i) \geq g(\alpha_{i+1})$, $i = 1(1)n-1$

Als er een keuze is dan moeten we zoveel mogelijk symbolen reduceren.
We doorlopen dan de volgende toestanden:

#(a + b) * c#
#(F + b) * c#
#(T + b) * c#
#(E + b) * c#
#(E + F) * c#
#(E + T) * c#

We zijn nu even ver als de vorige keer.

f(#) = 7	g(() = 1
f(() = 6	g(E) = 1
f(E) = 5	g(+) = 5
f(+) = 4	g(T) = 1
f(T) = 3	g()) = 5

Dus in de rij #(E + T zit een reduceerbaar stuk.

We reduceren zoveel mogelijk symbolen.

E + T → E

We krijgen

#(E) * c #

Er geldt

f(E) = 5	g()) = 5
f()) = 2	g(*) = 3 zodat in de string:

#(E) een reduceerbaar stuk voorkomt.

We krijgen dan achtereenvolgens:

#F * c #
#T * c #

#T * F #
T
E
Z # einde analyse.

We hebben nu experimenteel een methode bepaald, we kunnen niet bewijzen of het goed gaat. We zullen nu de zaak theoretisch funderen.
We zullen nu wat specialeren vanwege de beperkte tijd en beschouwen een operator grammatica en -taal.

Operator grammatica en -taal

Een grammatica met louter producties van de vorm $A \to \omega$ met $\omega \in \underline{V}^* - \{\varepsilon\}$ en $A \in \underline{V} - \underline{T}$, heet operator grammatica als er geen productie bestaat van de vorm

$$A \to \phi\ A_1\ A_2\ \psi\ ,\ A_1,\ A_2,\ A \in \underline{V} - \underline{T} \text{ en } \phi\ ,\ \psi \in \underline{V}^* \ .$$

Operator grammatica en -taal.

Een type-2 grammatica wordt een operator grammatica genoemd als er geen productie bestaat van de vorm:
$A \to \phi\ A_1\ A_2\ \psi\ ,\ A_1, A_2,\ A \in \underline{V} - \underline{T}$ en $\phi,\ \psi \in \underline{V}^*$

Stelling.

In een operator grammatica bevat geen enkel derivaat van een non-terminal twee opvolgende non-terminal symbolen. Dus geen enkele zinsvorm bevat twee opvolgende non-terminal symbolen.

Hulpstelling.

Als $\phi \overset{*}{\Longrightarrow} \psi$ en ϕ bevat geen twee opvolgende non-terminal symbolen dan bevat ψ deze ook niet.

Bewijs

Als $\phi = \psi$ dan is de stelling juist. Stel de stelling is juist voor iedere $\overset{*}{\Longrightarrow}$bestaande uit p stappen (p maal een productieregel toegepast): de inductie-veronderstelling.

$\phi \overset{*}{\Longrightarrow} \zeta \Rightarrow \psi$ waarbij $\phi \overset{*}{\Longrightarrow} \zeta$ in p stappen wordt bereikt.

ζ zal tenminste een non-terminal moeten bevatten:

$$\zeta = \zeta_1 A_1 \zeta_2 \quad \text{en } A_1 \to \zeta_3 \quad \text{zodat } \psi = \zeta_1 \zeta_3 \zeta_2.$$

Noch ζ_1 noch ζ_2 bevat volgens de inductieveronderstelling twee naast elkaar liggende non-terminals, maar zelfs is het laatste symbool van ζ_1 een terminal terwijl het eerste symbool van ζ_2 eveneens een terminal is, of ze zijn leeg. Verder bevat ζ_3 geen twee naast elkaar liggende non-terminal symbolen wegens het feit dat we met een operator grammatica te maken hebben.
Derhalve bevat ψ eveneens geen twee opvolgende non-terminal symbolen.
Hiermee is de hulpstelling bewezen.
Gevolg: daar iedere zinsvorm een derivaat is van Z (een string zonder twee naast elkaar liggende non-terminal symbolen) bevat een zinsvorm nooit twee naast elkaar gelegen non-terminal symbolen.
We voeren een drietal relaties in tussen terminal symbolen:

1) GH: gelijke hoogte
2) LL: links van en lager
3) LH: links van en hoger

Er geldt een relatie GH tussen twee symbolen a_1 en a_2,
genoteerd: a_1 GH a_2 met $a_1, a_2 \in \underline{T}$,
als er een productieregel $a \to \xi_1 a_1 a_2 \xi_2$ of $A \to \xi_1 a_1 A_1 a_2 \xi_2$ bestaat met

$$A, A_1 \in \underline{V} - \underline{T}; \quad \xi_1, \xi_2 \in \underline{V}^*.$$

Er geldt een relatie LL tussen a_1 en a_2 (a_1 LL a_2)
als er een $A \to \xi_1 A_1 a_2 \xi_2$ bestaat met $A_1 \overset{*}{\Longrightarrow} \xi_3 = \xi_4 a_1 B$ met a_1 het meest rechtse terminal symbool van ξ_3; $A_1, A \in \underline{V} - \underline{T}$; $\xi_1, \xi_2, \xi_3, \xi_4 \in \underline{V}^*$;

$$B \in (\underline{V} - \underline{T}) \quad \text{of } B = \varepsilon.$$

Dus: $A \to \xi_1 A_1 a_2 \xi_2$

$\Downarrow *$

$\xi_4 a_1 B$

Er geldt de relatie LH: (a_1 LH a_2) als er
weer een $A \rightarrow \xi_1\ a_1\ A_1\ \xi_2$ bestaat met $A_1 \overset{*}{\Longrightarrow} \xi_3 = B\ a_2\ \xi_4$

Dus $A \rightarrow \xi_1 a_1 A_1 \xi_2$

$\Downarrow *$

$B\ a_2\ \xi_4$

Definitie.

Een precedentie grammatica is een operator grammatica waarvoor er tussen het geordende paar (a_1,a_2) slechts één van de drie bovengenoemde relaties geldt. We noemen deze de precedentierelaties.

Een voorbeeld van een precedentie grammatica is G_{IV}.

Z→ E
E→ E+T
E→ T
T→ T×F
T→ F
F→ (E)
F→ a
F→ b

Tussen (en) bestaat de relatie GH.
Dus (GH) wegens F→ (E).
Verder is uit F→ (E) te halen

$\Downarrow *$
E+T, dus + LL)

en (LH +

Uit F→ (E)
$\Downarrow *$
T
$\Downarrow *$
T×F volgt: (LH× en
×LL).

Uit F→ (E)
$\Downarrow *$
(E) volgt: (LH (en
) LL).

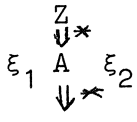
Op deze wijze gaan we gemakkelijk na dat de volgende precedentie matrix kan worden opgesteld:

	a	b	+	*	(	)	#
a			LL	LL		LL	LL
b			LL	LL		LL	LL
+	LH	LH	LL	LH	LH	LL	LL
*	LH	LH	LL	LL	LH	LL	LL
(	LH	LH	LH	LH	LH	GH	
)			LL	LL		LL	LL
#	LH	LH	LH	LH	LH		

Phrase.

Als $\xi = \xi_3 \eta \quad \xi_4$ een zinsvorm is met ξ_3, ξ_4, ξ en $\eta \in \underline{V}^*$.
dan is η een phrase als er A, ξ_1, ξ_2 bestaan zodanig dat
$A \overset{*}{\Longrightarrow} \eta$ en $\xi \overset{*}{\Longrightarrow} \xi_3$, $\xi_2 \overset{*}{\Longrightarrow} \xi_4$ met $\xi_1, \xi_2 \in \underline{V}^*$ en $A \in \underline{V} - \underline{T}$.

Gevolg:
Als $\xi = \xi_1 \eta \quad \xi_2$ een zinsvorm is met ξ_1, ξ_2, ξ, $\eta \in \underline{V}^*$ en er is een zinsvorm $\xi_1 A \; \xi_2$ terwijl $A \overset{*}{\Longrightarrow} \eta$, dan is η een phrase.



Definitie een priemphrase is een phrase welke tenminste een terminal symbool bevat, echter geen andere priemphrase meer bevat.

Voorbeelden: we nemen G_{IV} weer als voorbeeld.

In de zin (dus zinsvorm) (a * b) * b zijn priemphrasen a, b, en b immers: $Z \overset{*}{\Longrightarrow} F \overset{*}{\Longrightarrow}$ a
en $Z \overset{*}{\Longrightarrow} F \overset{*}{\Longrightarrow}$ b

In de zinsvorm:

(F ⋆ F) ⋆F zit de priemphrase F ⋆ F

Immers: (E) ⋆F is een zinsvorm

en E→ T →T ⋆ F→ F ⋆ F,

Dus: (E)⋆F ⋆ (F ⋆ F) ⋆ F.

Dus F ⋆ F is een phrase en wel een priemphrase.

Eigenschappen operator grammatica.

Stelling 1: Als in een zinsvorm ζ de combinatie a_1A_1 voorkomt, dan zal iedere phrase van ζ die a_1 bevat tevens A_1 bevatten.

Bewijs: Stel $\zeta=\xi_1\eta\xi_2$ waarin η een phrase, dan geldt volgens de definitie van phrase $Z\overset{*}{\Longrightarrow}\xi_3A\xi_4\overset{*}{}\ \xi_1\eta\xi_2$ met $\xi_3\overset{*}{\Longrightarrow}\xi_1$, $\xi_4\overset{*}{\Longrightarrow}\xi_2$, $A\overset{*}{\Longrightarrow}\eta$
Als η wel a_1 en niet A_1 bevat kan dit alleen als het eerste symbool van ξ_2 gelijk is aan A_1.
Maar dan geldt dat ξ_4 ook met een non-terminal A_1' begint.

Opgave : Bewijs dat uit $\xi_4\overset{*}{\Longrightarrow}\xi_2$ en ξ_2 begint met A_1 volgt dat ξ_4 begint met een non-terminal.
Dat betekent dat $Z\overset{*}{}\ \xi_3A\xi = \xi_3AA_1'\xi_5$ met $A_1'\xi_5=\xi_4$.
Het is echter onmogelijk in een operator grammatica dat er twee non-terminals aangrenzend zijn.

Stelling 2: Als in een zinsvorm ζ de combinatie A_1a_1 voorkomt, dan zal iedere phrase van ζ die a_1 bevat tevens A_1 bevatten.

Opgave: Bewijs dit.

Stelling 3: Geen enkele phrase in een zinsvorm wordt ingesloten door non-terminal symbolen.

Bewijs: Stel $\zeta=\xi_1\eta\xi_2$ met η een phrase. ξ_1 en ξ_2 mogen leeg zijn.

Veronderstel echter het eerste symbool van ξ_2 een non-terminal A_1 is dan is is het laatste symbool van η een terminal a_1. Dit zou in strijd zijn met stelling 1. Immers als van a_1A_1 a_1 in η ligt dan zal ook A_1 in η liggen. Op dezelfde wijze geldt dit met stelling 2 voor ξ_1.

Stelling 4: Als a_1A_1 in een zinsvorm ζ voorkomt en $A_1 \overset{*}{\Rightarrow} Ba_2\xi$ met $B \in (\underline{V}-\underline{T})$, $\xi \in \underline{V}^*$, a_1 en $a_2 \in \underline{T}$, en $A_1 \in \underline{V}-\underline{T}$ dan geldt de relatie a_1 LH a_2.
De zinsvorm ζ is een derivaat van Z. Stel:
$Z \overset{*}{\Rightarrow} \eta \Longrightarrow \zeta$
waarbij η door p maal een productieregel toe te passen uit Z afgeleid kan worden (p stappen).
We maken de inductieveronderstelling dat de stelling juist is voor p stappen en minder.
Daar $\eta \Rightarrow \zeta$ zal $\eta=\eta_1 A \eta_3$, $A \rightarrow \eta_2$.
Derhalve: $\eta = \eta_1 A \eta_3$ $\zeta = \eta_1\eta_2\eta_3$.

We kunnen nu een aantal gevallen onderscheiden:

1): a_1A_1 komt in η_1 voor
2): a_1A_1 komt in η_3 voor
3): a_1A_1 komt in η_2 voor
4): a_1 is het laatste symbool van η_1 en A_1 het eerste symbool van η_2
5): Het geval dat a_1 het laatste symbool is van η_2 en A_1 het eerste symbool van η_3 kan niet vanwege stelling 2, daar η_2 een phrase is.

Ad 1 en 2): In deze gevallen komt a_1A_1 reeds in η voor, zodat we de inductieveronderstelling kunnen toepassen.

Ad 3) : Als a_1A_1 in η_2 voorkomt hebben we:
de productieregel $A \rightarrow \eta_2 = \psi_1 a_1 A_1 \psi_2$.
Daar $A_1 \overset{*}{\Rightarrow} Ba_2\xi$ is volgens de definitie a_1 LH a_2.

ad 4) : a_1 is het laatste symbool van η_1 en A_1 is het eerste symbool van η_2.
Dus: $\eta_2 = A_1\psi_1$.
Er geldt dus: $A \rightarrow \eta_2 = A_1\psi_1 \overset{*}{\Rightarrow} Ba_2\xi\psi_1 = Ba_2\psi_2$
of $A \overset{*}{\Rightarrow} Ba_2\psi_2$.
Derhalve $Z \overset{*}{\Rightarrow} \eta = \eta_1 A\eta_3 = \eta_4 a_1 A\eta_3$ en $A \overset{*}{\Rightarrow} Ba_2\psi_2$.
p stappen
volgens de inductie-veronderstelling geldt weer
a_1 LH a_2
zodat de stelling bewezen is.

Stelling 5: Als A_1a_2 in een zinsvorm ζ voorkomt en $A_1 \overset{*}{\Rightarrow} \xi a_1 B$ dan geldt de reductie a_1 LL a_2

Bewijs dit.

Stelling 6: Als de string $a_1A_1a_2$ in een zinsvorm ζ voorkomt dan geldt tenminste één van de relaties
GH, LH, LL tussen a_1 en a_2.

Bewijs: $Z \overset{*}{\Rightarrow} \eta \Rightarrow \zeta$
Er geldt: $\eta = \eta_1 A_3$ en $A \rightarrow \eta_2$ zodat:
$\zeta = \eta_1\eta_2\eta_3$ Derhalve is η_2 een phrase.

Er zijn weer verschillende gevallen te onderscheiden.
1): de string $a_1A_1a_2$ ligt in η_1
2): de string $a_1A_1a_2$ ligt in η_3
3): de string $a_1A_1a_2$ ligt in η_2
4): a_1 is het laatste symbool van η_1
5): a_2 is het eerste symbool van η_3

Het geval a_1A_1 ligt in η_1 en a_2 in η_2 en het geval a_1 in η_2 en A_1a_2 in η_3 zijn niet mogelijk omdat een phrase niet door non-terminal symbolen omgeven kan zijn.

ad 1 en 2): Daar in deze gevallen a_1Aa_2 reeds in η voorkomt kunnen we weer m.b.v. een inductie-veronderstelling dat de stelling voor p stappen juist is besluiten dat tenminste één van de relaties GH, LH of LL tussen a_1 en a_2 geldt.

ad 3): Nu geldt: $A \rightarrow \eta_2$ en η_2 bevat $a_1A_1a_2$ dus:
$\eta_2 = \eta_4 a_1 A_1 a_2 \eta_5$ of $A \rightarrow \eta_4 a_1 A_1 a_2 \eta_5$.
Dus geldt de relatie a_1 GH a_2.

ad 4): a_1 is het laatste symbool van η_1. De phrase η_2 begint met de string A_1a_2. Dus $\eta_2 = A_1a_2\eta_6$ of

$A \rightarrow A_1a_2\eta_6$

verder is dus $\eta_1 = \eta_7 a_1$ en

$$Z \overset{*}{\Rightarrow} \eta = \eta_7 a_1 A \eta_3 \Downarrow * \; A_1 a_2 \eta_6$$

Volgens stelling 4 geldt dan a_1 LH a_2.

ad 5): a_2 is het eerste symbool van η_3. Dus $\eta_3 = a_2\eta_7$.
Verder is a_1A_1 het eind van η_2. Derhalve is

$A \rightarrow \eta_2 = \eta_8 a_1 A_1$

Er geldt dus $$Z \overset{*}{\Rightarrow} \eta = \eta_1 A \eta_3 = \eta_1 A a_2 \eta_7 \Downarrow * \; \eta_8 a_1 A_1$$

Volgens stelling 5 geldt dus: a_1 LL a_2.

<u>Gevolg:</u> Als in een zinsvorm ζ een substring a_1a_2 voorkomt dan geldt één van de drie relaties:
a_1 GH a_2, a_1 LL a_2 of a_1 LH a_2.

<u>Opgave:</u> Bewijs dit.

Stelling 7: Als ζ een zinsvorm is die tenminste 1 karakter bevat, dan bevat de string $\#\zeta\#$ een substring $a_1\ a_2$ zodanig dat a_1 LH b_1 GH b_2 GH ..... b_n LL a_2, waarin $b_1 b_2 \ldots, b_n$ de terminal symbolen zijn die in ξ voorkomen.

Volgens de vorige stelling bestaat er een relatie GH, LL of LH tussen twee opvolgende of door een non-terminal gescheiden terminal symbolen.

Er geldt tevens $\#$LH c_1 en c_m LL$\#$ als c_1 het meest linkse terminal symbool in ζ is en c_m het meest rechtse symbool in ζ is.

Stel $c_0 = \#$, $c_1, c_2, \ldots, c_{m+1} = \#$ zodanig dat $c_1, \ldots, c_m$ de terminal symbolen van ζ zijn.

Laat j het kleinste gehele getal zijn waarvoor

c_j LL c_{j+1}.

Er bestaat zo'n j daar c_m LL c_{m+i} Blijkbaar is $1 \leq j \leq m$

Laat verder i het grootste gehele getal $<j$ zijn waarvoor

c_i LH c_{i+1}

Een dergelijke i bestaat daar c_0 LH c_1.

Nu geldt voor elke k met $i<k<j$ dat het niet mogelijk is dat

c_k LL c_{k+1}, immers j was de kleinste. Het is ook niet mogelijk dat

c_k LH c_{k+1}, immers i was de grootste gehele $<j$.

Daar er tenminste één relatie moet gelden is dus:

c_k GH c_{k+1}, $i<k<j$.

waarmee de stelling bewezen is.

We zullen nu een aantal eigenschappen van een precedentie grammatica afleiden.

Precedentiegrammatica

Stelling 1

Als in een zinsvorm hetzij $a_1\ A_1\ a_2$ hetzij $a_1\ a_2$ voorkomt is er precies één relatie GH, LL of LH tussen a_1 en a_2.

Bewijs:
Dit is duidelijk met stelling 7 van de vorige paragraaf en met de definitie van precedentie grammatica waaruit volgt dat er niet meer dan één relatie geldt.

Stelling 2

Als in een zinsvorm ζ hetzij $a_1\ A_1\ a_2$ of $a_1\ a_2$ voorkomt, terwijl a_1 LH a_2 is dan is er een phrase in ζ waartoe wel a_2 doch niet a_1 behoort.

Bewijs:
Daar ζ een zinsvorm is geldt:

$$Z = \xi_0 \Longrightarrow \xi_1 \Longrightarrow \xi_2 \Longrightarrow \ldots \Longrightarrow \xi_n = \zeta,\ n \geq 1.$$

Laat i het grootste gehele getal zijn waarvoor geldt:

a_2 komt niet voor in ξ_i, $0 \leq i \leq n-1$. a_2 komt dan wel voor in ξ_{i+1}.

Derhalve moet in de stap $\xi_i \Longrightarrow \xi_{i+1}$ het terminal symbool a_2 gemaakt zijn.

Dus: $\xi_i = \eta_1\ A\ \eta_2$, $A \rightarrow \eta_3\ a_2\ \eta_4 = \eta_5$

$$\xi_{i+1} = \eta_1\ \eta_3\ a_2\ \eta_4\ \eta_2 = \eta_6\ a_2\ \eta_7 = \eta_1\ \eta_5\ \eta_2$$

Er zijn 3 gevallen te onderscheiden:

1. a_1 komt voor de eerste maal voor in ξ_{i+1}. In dit geval komt in ξ_{i+1} dus òf $a_1\ A_1\ a_2$ òf $a_1\ a_2$ voor.
   We hebben dus te maken met resp.
   a) $\xi_{i+1} = \eta_8\ a_1\ A_1\ a_2\ \eta_7$ of
   b) $\xi_{i+1} = \eta_8\ a_1\ a_2\ \eta_7$.

   ad a: η_5 is een phrase. In de zinsvorm ξ_{i+1} komt dus de phrase η_5 voor. Deze bevat a_2, terwijl ξ_{i+1} de combinatie $A_1\ a_2$ bevat
   Volgens stelling 2 v.d. vorige par. bevat η_5 dan ook A_1.
   Als a_1 voor de eerste maal in ξ_{i+1} voorkomt betekent dit, dat hij niet in η_1 voorkomt. Hij moet ons voorkomen in η_5.
   Dus $A \rightarrow \eta_5$ bevat $a_1\ A\ a_2$. Dus a_1 GH a_2.
   Dit is in tegenspraak met het uitgangspunt.

ad b: Daar a_1 niet in η_1 voorkomt en wel in ξ_{i+1} , dus in η_5, geldt $A \rightarrow \eta_5$ bevat $a_1\ a_2$ dus a_1 GH a_2.
Dit is eveneens in tegenspraak met het uitgangspunt

2. a_1 komt voor de eerste maal voor in ξ_j met $j > i+1$.
We hebben:

$$\xi_i = \eta_1\ A\ \eta_2,\ A \rightarrow \eta_3\ a_2\ \eta_4 = \eta_5,\ \xi_{i+1} = \eta_6\ a_2\ \eta_7.$$

Verder is:

$$\xi_i \overset{*}{\Longrightarrow} \xi_j \quad . \qquad \xi_j \text{ bevat } a_1\ A_1\ a_2$$

Dus zal uit $\eta_6 \overset{*}{\Longrightarrow} \eta_8\ a_1\ A_1$ afgeleid moeten zijn.

Dus zal η_6 moeten zijn $\eta_6 = \eta_9\ B \overset{*}{\Longrightarrow} \eta_{10} a_1\ A_1$.

In ξ_{i+1} komt dus voor B a_2 met $B \overset{*}{\Longrightarrow}_{10} a_1\ A_1$.

Derhalve: volgens stelling 5 van de vorige par. geldt: a_1 LL a_2.

Dit is echter in strijd met de veronderstelling dat

a_1 LH a_2 was.

3. Er blijft dus niets anders over dan dat:

a_1 al in ξ_i voorkomt daar hij in ζ moet voorkomen.

We hebben dan: $\xi_i = \eta_1\ A\ \eta_2$ bevat al a_1,

$$A \rightarrow \eta_3\ a_2\ \eta_4 = \eta_5,$$

$$\xi_{i+1} = \eta_1\ \eta_5\ \eta_2 \quad \text{bevat } a_1 \text{ vóór } a_2.$$

a_1 moet dus in η_1 voorkomen . Bekijken we dan ξ_{i+1}m dan zien we dat in de phrase $\eta_5\ a_1$ niet voorkomt, waarmee de stelling bewezen is.

Stelling 3

Op dezelfde wijze geldt dat:
wanneer $a_1\ A_1\ a_2$ of $a_1\ a_2$ in een zinsvorm ζ voorkomt en a_1 LL a_2, dan bestaat er een phrase in ζ waartoe a_1 wel en a_2 niet behoort.

Opgave: Bewijs dit.

Stelling 4

Als $a_1\ A_1\ a_2$ of $a_1\ a_2$ in een zinsvorm ζ voorkomt en a_1 GH a_2, dan bevat iedere phrase die a_1 bevat ook a_2 en iedere phrase die a_2 bevat, bevat ook a_1.

Bewijs:

Veronderstel er is een phrase die wel a_1 en niet a_2 bevat.
Deze phrase zal dan volgens stelling 1 van de vorige par. ook A_1 moeten bevatten (we bekijken eerst het geval $a_1\ A_1\ a_2$).
Dus:

$$Z \overset{*}{\Longrightarrow} \xi_1\ A\ \xi_2 \text{ en } \xi_1 \overset{*}{\Longrightarrow} \xi_3\ ,\ \xi_2 \overset{*}{\Longrightarrow} \xi_4,\ A \rightarrow \eta, \text{ zodat}$$

$$Z \overset{*}{\Longrightarrow} \xi_1\ A\ \xi_2 \overset{*}{\Longrightarrow} \xi_3\ A\ \xi_4 \overset{*}{\Longrightarrow} \xi_3\ \eta\ \xi_4 = \zeta.$$

η is dus een phrase die a_1 en dus $a_1\ A_1$ bevat.
Blijkbaar begint ξ_4 met a_2 en eindigt η met $a_1\ A_1$.

Er moet dus $\xi_4 = a_2\ \xi_5$ gelden.

Derhalve $Z \overset{*}{\Longrightarrow} \xi_3\ A\ a_2\ \xi_5$ en $A \rightarrow \eta = \xi_6\ a_1\ A_1$.

Dus: a_1 LL a_2.

Maar de veronderstelling was dat a_1 GH a_2 was.
Dus moet η ook a_2 bevatten.

Op dezelfde manier kan worden aangetoond dat wanneer ζ $a_1\ a_2$ bevat en η a_1 bevat, dat η dan ook a_2 bevat.

Opgave: bewijs dit.

Verder kan worden aangetoond dat wanneer ζ hetzij $a_1\ A_1\ a_2$ of $a_1\ a_2$ bevat en η a_2 bevat dat η dan ook a_1 bevat.

Opgave: bewijs dit.

Stelling 5

Als η een phrase is die de terminal symbolen $a_1, a_2, \ldots, a_n$ bevat, $n \geq 1$ en

$$a_i \text{ GH } a_{i+1},\ 1 \leq i \leq n-1$$

dan is η een priemphrase.

Bewijs:

Een priemphrase bevat tenminste een terminal symbool.
Elke priemphrase ξ in η moet één van de a_i's bevatten.
Volgens de vorige stelling moet ξ dan alle a_i's bevatten.
Volgens stelling 1 en 2 van de vorige par. moet ξ dan ook alle non-terminal symbolen van η bevatten ($A_i\ a_i$, $a_i\ A_{i+1}$).

Dus ξ bevat alle symbolen van η en η is dus zelf een priemphrase.

Stelling 6

Als in ζ de string $a_0\ \xi\ a_{n+1}$ voorkomt, terwijl $a_1, a_2, \ldots, a_n$ in ξ voorkomen en bovendien geldt:

$$a_0 \text{ LH } a_1;\ a_i \text{ GH } a_{i+1},\ 1 \leq i \leq n-1;\ a_n \text{ LL } a_{n+1}$$

dan is ξ een priemphrase.

Bewijs:

Veronderstel: $Z = \xi_0 \Longrightarrow \xi_1 \Longrightarrow \xi_2 \Longrightarrow \ldots \Longrightarrow \xi_n = \zeta$.

Laat i het grootste gehele getal zijn waarvoor a_1 niet in ξ_i voorkomt.
Wegens stelling 4 komen dan ook $a_2, \ldots, a_n$ niet voor in ξ_i.

Immers ξ_{i+1} bevat wel a_1. Dus moet er een productie geweest zijn $B \rightarrow \eta$

zodat: ξ_i B bevat en $\xi_i \Longrightarrow \xi_{i+1}$, terwijl ξ_{i+1} η bevat.

Als één van de symbolen a_i, $1 \leq i \leq n$, in η voorkomt dan komen ze allemaal voor.
Wegens stelling 2 geldt dat er een phrase bestaat die a_0 niet en a_1 wel bevat. Deze phrase moet dan weer alle a_i, $j \leq i < n$, bevatten.
Volgens stelling 1 van de vorige par. moeten dan ook alle non-terminal symbolen die grenzen aan a_1 t/m a_n tot die phrase behoren.
Derhalve is deze phrase gelijk aan ξ.
Volgens de vorige stelling is het een priemphrase.

Stelling 7

Als $a_0\ \xi\ a_{n+1}$ in $\#\ \zeta\ \#$ voorkomt, $Z \overset{*}{\Longrightarrow} \zeta$ en de terminal symbolen van ξ zijn $a_1, \ldots, a_n$, $n \geq 1$ en a_0 LH a_1 GH a_2 GH ... GH a_n LL a_{n+1} dan is ξ een priemphrase van ζ.

Bewijs:

We moeten 4 gevallen bekijken n.l.:

1. $a_0 \neq \#$ en $a_{n+1} \neq \#$
   We kunnen dan de vorige stelling toepassen.

2. $a_0 = \#$ en $a_{n+1} \neq \#$
   In dat geval is er volgens stelling 3 een phrase die a_n bevat en niet a_{n+1}.

Deze phrase bevat daarom ook a_1 t/m a_{n+1} en alle aangrenzende non-terminals. De phrase is dus identiek met ξ.

Volgens stelling 5 is het een priemphrase.

3. $a_0 \neq \#$ en $a_{n+1} = \#$. Zie 2.

4. $a_0 = \#$ en $a_{n+1} = \#$. Dan is $\xi = \zeta$ en volgens stelling 5 is ξ een priemphrase.

Stelling 8

Iedere zinsvorm ζ is of een non-terminal symbool of bevat een priemphrase.
Als ζ een terminal is, is het zelf een priemphrase.
Als ζ meer dan één symbool bevat dan moet er een terminal bij zijn.
Volgens stelling 7 van de vorige par. en stelling 7 van deze paragraaf bevat ζ een priemphrase.

Priemphrase herkenning

We passen in principe stelling 7 toe.
De syntactische analyse is dan eenvoudig geworden.
We zoeken een priemphrase op en vervangen deze door de bijbehorende non-terminal. Bij deze reductie verdwijnt in ieder geval een terminal symbool.
Als we zo doorgaan zullen we in een eindig aantal stappen de zin tot het initiële symbool Z gereduceerd hebben.

Voorbeeld:

We nemen een zin die gegenereerd wordt door G_{II}

$\#(a+b) \times b\#$. De priemphrasen zijn a, b en b.

We reduceren tot:

$\#(F+b) \times b\#$. $\#$ LH (LH + LH b LL) LL $\times$ LH b LL $\#$

Dus b en ook b zijn priemphrasen.
Deze liggen tussen LH en LL in.

We reduceren weer

$\#(F+F) \times b\#$. $\#$ LH (LH + LL) LL + LH b LL $\#$

Alles tussen (en) dus priemphrase

F+F
en b } priemphrase

$\#$(E) $\times$ F$\#$. $\#$ LH (GH) LL $\times$ LL $\#$

(E) priemphrase

$\#F \times F\#$. $\#$LH $\times$ LL $\#$

$\#T\#$ Dus F $\times$ F priemphrase.

$\#Z\#$

De syntactische analysator ziet er in principe als volgt uit, als we aannemen dat in de pushdownstapel S het gedeelte van de zin staat dat zover mogelijk is gereduceerd.
De stapel S is uitgerust met een wijzer w die naar de eerste vrije plaats wijst.
De inputstapel I heeft een index t zodat I [t] het eerste te lezen symbool voorstelt.
In de begintoestand bevindt zich de zin # ..... # in de inputstapel.

```
t:= lengte inputstring;
S [1] = ⊦##⊦; w:= 2; t:= t-1;
if S [w-1] = terminal V⊦ ## ⊦
then begin
          i:= w-1; x: = S[ i ]end
else begin
          i:= w-2; x:= S [ i ] end i is de index in S waar zich de
                                    eerste terminal bevindt gerekend
                                    vanaf de top;
if  x  LL I [t ]
then begin
          zoek priemphrase en ga reduceren; goto B end
else begin
          S [ w ] : = I [ t ]; w: = w + 1; t:= t-1; goto B end;
```

Hierbij wordt gebruikt de:

```
procedure zoek priemphrase en ga reduceren;

begin for j:=i-1 step-1 until j do
      if S [j] = terminal
      then begin if S [ j ] LH  x
                 then vervang top plaatsen stapel S (j+1,w-1)
                 else x:= S [ j ]  end
end;
```

Ten behoeve van de grammatica G_{IV} zullen we nu de procedure vervang top plaatsen stapel S beschrijven.

```
procedure vervang topplaatsen stapel S(p,q);
begin
     if p=q
     then begin
          if S[p] ≠ {a}∧S[p] ≠ {b}
          then fout
          else S[p]:={F}
          end
     else begin
          if S[p]= {( }
          then begin
               if S[p+1] ≠ { F }∨S[p+2] ≠ { ) }
               then fout;
               S[p]:={F}; w:= p+1
               end;
          if S[p+1] = { * }
          then begin
               if S[p] ≠ {F}∨S |p+2| ≠ { F }
               then fout;
               S|p|:={F}; w:= p+1
               end;
          if S[p+1] = {+}
          then begin
               if S[p]≠{F}∨S[p+2]≠{ F }
               then  fout;
               S[p]:={F}; w:=p+1
               end
          end
end vervang top stapel;
```

Tot nu toe hebben we steeds naar de relatie tussen terminals gekeken. We willen nu de relaties iets uitbreiden met de definitie van een relatie tussen non-terminals en terminals, zó dat we bij ons onderzoek alleen nog naar de top van S hoeven te kijken.

Als er een productie bestaat $A_1 \rightarrow \eta_1 A a_2 \eta_2$ dan definieren we A GH a_2.
Als er een productieregel bestaat $A_1 \rightarrow \eta_1 A a_2 \eta_2$ en $A \overset{*}{\Rightarrow} \eta_3 B$, η_3 eventueel leeg, terwijl tenminste één productieregel wordt toegepast dan definieren we B LL a_2. We beschouwen nu precedentiegrammatica's waarin tussen iedere non-terminal en een terminal ten hoogste één relatie bestaat. Ga na dat de grammatica Z → E; E → T+E; E → T; T → F * T; T → F; F →(E); F → a; F → b; wel een precedentiegrammatica is doch dat er non-terminals zijn die twee relaties met een terminal symbool hebben.

Voorbeeld grammatica G_{IV}.

Dus uit F →(E) volgt	:	E GH)
uit F →(E) en E→ T volgt	:	T LL)
en $E \overset{*}{\Rightarrow} F$ volgt	:	F LL)
uit T →T*F volgt	:	T GH *
uit T →T*F en T→ F volgt	:	F LL *
uit E →E+T volgt	:	E GH +
en uit E →E+T en E→ T volgt	:	T LL +
en uit $E \overset{*}{\Rightarrow} F$ volgt dan	:	F LL +

Stelling.

Als de relatie A GH a_2 bestaat en er is een zinsvorm
$\zeta = \xi_1 a_1 A a_2 \xi_2$ dan kan er geen relatie LL bestaan tussen a_1 en a_2.

Bewijs.

Als a_1 LL a_2 geldt dan moet de string $\xi_1 a_1 A$ een phrase bevatten, zeg η, waarbij $a_1 A$ in η ligt.
Dus: $\xi_i a_1 A = \xi_3 \eta$, met $a_1 A$ in η.
$Z \overset{*}{\Rightarrow} \psi \Rightarrow \zeta$ zodanig dat
$Z \overset{*}{\Rightarrow} \psi \Rightarrow \xi_3 B a_2 \xi_2$
$\Downarrow$
Dus $\eta = a_1 A$ Derhalve A LL a_2

De syntactische analyse kan dan als volgt verlopen.
In de inputstapel bevindt zich een (terminal) string x.
In de pushdown stapel bevindt zich een string ξ_1 met de volgende eigenschappen:
In ξ_1 komen de terminal symbolen $b_1, b_2, \ldots, b_n$ voor zodanig dat:

$$b_i \text{ LH } b_{i+1} \text{ of } b_i \text{ GH } b_{i+1}, \quad 1 \leq i \leq n-1.$$

Verder bevat ξ_1 de non-terminal symbolen

$B_1, \ldots, B_m$ zodanig dat wanneer

$b_s B_i b_{s+1}$ als substring in ξ_1 voorkomt geldt:

B_i GH b_{s+1}.

We noemen deze toestand van S de toestand Σ.
Let wel: als het laatste symbool van ξ_1 B_m is, is hiervan geen relatie met een terminal uit ξ_1 gedefinieerd.
We onderscheiden nu verschillende gevallen.

1) Het laatste symbool van ξ_1 is een terminal (dus b_n).
   Als nu b_n LH c, waarbij c het meest linkse symbool van x is, dan brengen we c over naar de pushdown stapel S
   Hierdoor komt S in de toestand Σ.
   Als b_n GH c dan brengen we c ook naar S over.
   Ook dan komt S in de toestand Σ.

Is echter b_n LL c dan moet er een priemphrase in S zijn die kan worden gereduceerd tot een non-terminal. Dan ontstaat de volgende situatie.

2) Het laatste symbool van ξ_1 is, of geworden, een non-terminalsymbool zeg D.
   Als nu D LL c dan wordt een reductie toegepast van de vorm
   $A \overset{*}{\Longrightarrow} \eta D$ (η eventueel leeg).
   Als er keuze is tussen meer reductieregels wordt de priemphrase gekozen.
   Dit proces stopt zodra de string ξ_1 op een non-terminal eindigt die de relatie GH met c heeft, tenzij de toestand wordt bereikt #Z in S en # in I.
   In dat geval moet c weer in S opgenomen worden etc.

De precedentie matrix die dan ontstaat is

	a	b	+	*	(	)	#
a			LL	LL		LL	LL
b			LL	LL		LL	LL
+	LH	LH	LL	LH	LH	LL	LL
*	LH	LH	LL	LL	LH	LL	LL
(	LH	LH	LH	LH	LH	GH	
)			LL	LL		LL	LL
F			LL	LL		LL	LL
T			LL	GH		LL	LL
E			GH			GH	LL
Z							
#	LH	LH	LH	LH	LH		

De vraag die we nu nog willen beantwoorden is of de matrix niet vervangen kan worden door een of andere tabel waarvoor we minder ruimte nodig hebben.
Daarvoor proberen we functies f en g te vinden, zodanig dat

$f(\alpha) <_g (a_2)$ geldt als α LL a_2
$f(a_1) >_g (a_2)$ geldt als a_1 LH a_2
$f(\alpha) >_g (a_2)$ geldt als α GH a_2

$a_1, a_2 \in \underline{T}, \alpha \in \underline{V}$

We kunnen deze functies als volgt construeren.
We stellen $f(a) = g(a) = 1$ voor iedere $a \in T$
en $f(A) = 1$ $A \in \underline{V}-\underline{T}$

1. voor elke a_1 LL a_2 als $f(a_1) \geq g(a_2)$
   $g(a_2) := f(a_1) + 1$
2. voor elke a_1 LH a_2 als $f(a_1) \leq g(a_2)$ dan
   $f(a_1) := g(a_2) + 1$
3. voor elke a_1 GH a_2 als $f(a_1) < g(a_2)$ dan
   $f(a_1) := g(a_2)$
4. voor elke A LL a_2 als $f(A) \geq g(a_2)$ dan
   $g(a_2) := f(A) + 1$
5. voor elke A GH a_2 als $f(A) < g(a_2)$ dan
   $f(A) := g(a_2)$

g(A) is niet gedefinieerd.

Herhaal de stappen 1 t/m 4 tot dat het proces geen verandering meer teweegbrengt.
We krijgen dan de volgende tabel voor G_{IV}.

	f	g
a,b	1	1
+	4	5
*	2	3
(	6	1
)	2	5
F	1	-
T	3	-
E	5	-
Z	1	-
#	6	6

Hiermee zijn we terecht gekomen op onze probeerselen blz. 19 en 20.