TECHNISCHE HOGESCHOOL EINDHOVEN
Afdeling Algemene Wetenschappen
Onderafdeling der Wiskunde

AUTOMATENTHEORIE

en

FORMELE TALEN

Auteur Onbekend

Lentetrimester 1984

Inhoudsbeschrijving

AUTOMATENTHEORIE en FORMELE TALEN

Auteur onbekend

Lentetrimester 1984

JdG, 31 Juli 2005

TECHNISCHE HOGESCHOOL EINDHOVEN

Onderafdeling der Wiskunde en Informatica

AUTOMATENTHEORIE EN FORMELE TALEN (2K170)

Hoofdstuk 1. Formele talen

lentetrimester 1984

HOOFDSTUK 1. Formele talen

§1. Symbolen en symboolrijen

In talen worden objecten achter elkaar gezet om daarmee eindige rijen te vormen. Zo kan men rijen letters vormen, die woorden genoemd worden, maar ook rijen woorden, die zinnen opleveren. Er zijn echter andere gevallen denkbaar: Een wiskundige formule is (soms) op te vatten als een rij wiskundige tekens en evenzo een programma in een programmeertaal.

Om al deze gevallen gezamenlijk te behandelen verdient het aanbeveling om voor de objecten en de eruit gevormde rijen neutrale namen te kiezen. In de theorie van de formele talen kiest men vaak de benamingen "letters" en "woorden", die voor sommige toepassingen niet zo gelukkig zijn. Wij zullen "symbolen" en "symboolrijen" gebruiken.

Bij het vormen van rijen uit symbolen stellen we de eis, dat uit de rij de symbolen die bij de vorming gebruikt zijn, weer terug te verkrijgen zijn. Bij het vormen van woorden uit letters kan het wat dat betreft geen kwaad om de letters gewoon achter elkaar te zetten; bij het vormen van zinnen uit woorden last men tussen de woorden spaties in. In andere toepassingen zijn vaak scheidingssymbolen nodig; de zojuist genoemde spatie is ook als zodanig op te vatten. Een voorbeeld krijgt men, als men als symbolen de op de gebruikelijke wijze in het tientallig stelsel geschreven natuurlijke getallen kiest. Wil men dan rijen natuurlijke getallen opschrijven, dan is er behoefte aan een scheidingssymbool, waarvoor de keuze van de komma gebruikelijk is: 31,7,302 en niet 317302. In sommige programmeertalen is een "identifier" op te vatten als een symbool; het is een rij letters en cijfers.

Bij de opzet van een algemene theorie zullen we de symbolen gewoon achter elkaar schrijven en geen scheidingssymbolen gebruiken.

Laat $\sum$ een verzameling zijn, die altijd niet-leeg en gewoonlijk eindig verondersteld wordt. De elementen van $\sum$ noemen we symbolen. Men noemt $\sum$ wel eens een alfabet; maar deze naam past weer bij de betiteling van symbolen als letters.

We vormen nu eindige rijen van symbolen en verkrijgen dan een verzameling $\sum^+$ van symboolrijen. Om dit wat formeler te beschrijven, hebben we behoefte aan aanduidingen van elementen van $\sum$ en van $\sum^+$. Een gangbare gewoonte is om elementen van $\sum$ aan te duiden met letters uit het begin van het alfabet (a,b,c,...) en elementen van $\sum^+$ met letters verderop in het alfabet, zoals p, u, w. Wij leggen ons niet op een bepaalde keuze vast.

Om een wat formelere beschrijving van de elementen van $\sum^+$ te krijgen, denken we ons een symboolrij van links naar rechts opgeschreven, waarbij telkens een nieuwe symboolrij wordt verkregen door aan een reeds verkregen symboolrij rechts een symbool te hechten. Dit proces kan onbepaald vaak herhaald worden. Dit ondervangen we met een recurrente beschrijving (ook wel recursieve of inductieve beschrijving genoemd).

Definitie $\sum^+$.
Als $s \in \sum$, dan $s \in \sum^+$; als $x \in \sum^+$ en $s \in \sum$, dan $xs \in \sum^+$. Alleen hetgeen op grond van bovenstaande voorschriften is gevormd, is een element van $\sum^+$.

Let wel, dat $\sum \subset \sum^+$: er wordt geen verschil gemaakt tussen een symbool en de symboolrij die uit dat ene symbool bestaat.

Voorbeeld.

$\sum = \{a,b,c\}$; dan $a \in \sum^+$, $accabb \in \sum^+$, $bbbb \in \sum^+$.

Vaak wordt ook de zogenaamde "lege symboolrij" meegeteld, die ontstaat door helemaal geen symbolen op te schrijven; het resultaat van dit opschrijven is dan echter onzichtbaar, hetgeen onpraktisch is om mee te werken. Om dit te ondervangen, voeren we een apart teken in voor de lege symboolrij; wij kiezen daarvoor: Λ. In de literatuur zijn de keuzen λ en ε ook gangbaar. Uiteraard houdt deze keuze in, dat we Λ nergens anders voor gebruiken. Ook aan Λ moeten we aan de rechterzijde een symbool kunnen hechten. De verzameling symboolrijen inclusief de lege rij noemen we $\sum^*$.

Definitie.

$\sum^* := \sum^+ \cup \{\Lambda\}$; als $s \in \sum$, dan $\Lambda s = s$.

Uit twee symboolrijen v en w kan een nieuwe gevormd worden door w achter v te plaatsen. Deze operatie noemen we concatenatie en het resultaat zullen we voorlopig $(v \cdot w)$ noemen. Om dit formeel te beschrijven, denken we de symboolrijen weer van links naar rechts opgebouwd. Uitgaande van de symboolrij v wordt w er letter voor letter achtergevoegd. Het resultaat is een definitie door recurrentie naar de opbouw van w.

Definitie $(v \cdot w)$.

Stel $v \in \sum^*$, $x \in \sum^*$, $s \in \sum$; $(v \cdot \Lambda) := v$; $(v \cdot xs) := (v \cdot x)s$.

Enkele eenvoudige eigenschappen van de concatenatie zijn de volgende.

Stel $v \in \sum^*$, $w \in \sum^*$, $u \in \sum^*$, $s \in \sum$. Dan geldt:

$$((v \cdot w) \cdot u) = (v \cdot (w \cdot u)) ,$$

$$(v \cdot s) = vs , \quad (\Lambda \cdot v) = v .$$

Om te laten zien hoe zulke eigenschappen uit de definitie kunnen worden afgeleid behandelen we bij wijze van voorbeeld de associatieve eigenschap $((v \cdot w) \cdot u) = (v \cdot (w \cdot u))$. We geven een bewijs door recurrentie langs de opbouw van u. Daartoe kiezen we eerst $u = \Lambda$:

$$((v \cdot w) \cdot \Lambda) = (v \cdot w) = (v \cdot (w \cdot \Lambda)) .$$

Daarna stellen we $u = xs$ ($x \in \sum^*$, $s \in \sum$) en veronderstellen dat de eigenschap geldt voor $u = x$:

$$((v \cdot w) \cdot xs) = ((v \cdot w) \cdot x)s = (v \cdot (w \cdot x))s = (v \cdot (w \cdot x)s) =$$

$$= (v \cdot (w \cdot xs)) .$$

Opgave.
Leid de andere twee eigenschappen uit de definitie af.

Op grond van de associatieve eigenschap kunnen we de haakjes weglaten en $v \cdot w$ schrijven; er is geen gevaar voor verwarring bij $v \cdot w \cdot u$. Weliswaar kunnen wij dit opvatten als $(v \cdot w) \cdot u$ en als $v \cdot (w \cdot u)$, maar dat hindert niet, want het resultaat is hetzelfde. We laten vervolgens ook nog de punt weg en schrijven vw; verwarring met het achterschrijven van een symbool is er ook niet, omdat $v \cdot s = vs$ als $s \in \sum$.

De concatenatie is een binaire operatie op $\sum^*$, die associatief is. Een verzameling met een associatieve operatie wordt een semigroep genoemd: $(\sum^*,)$ is een semigroep. Verder fungeert Λ als een eenheidselement (of neutraal element): $\Lambda v = v\Lambda = v$. Een semigroep met eenheidselement noemt men een monoïde.

Zoals in semigroepen gebruikelijk, schrijven we herhaalde produkten als machten: $v^2 = vv$, $v^3 = vvv$ enz. Algemeen geformuleerd (inductie naar n) voor $v \in \sum^*$:

$$v^0 := \Lambda , \quad v^{n+1} := v^n v .$$

Enkele eigenschappen: Stel $v \in \sum^*$, $n \in \mathbb{N} \cup \{0\}$, $m \in \mathbb{N} \cup \{0\}$:

$$v^m v^n = v^{m+n} , \quad (v^m)^n = v^{mn} .$$

Opmerking.

Wij rekenen het getal 0 niet tot de natuurlijke getallen: $0 \notin \mathbb{N}$.

Pas op!

Neem niet klakkeloos produkt- en machtseigenschappen van getallen over! Concatenatie is niet commutatief: vw en wv kunnen verschillend zijn (voorbeeld v = a, w = ab). Ook $(vw)^n$ en $v^n w^n$ kunnen verschillend zijn.

We introduceren nu de lengte $|v|$ van een symboolrij v: Het aantal malen dat we bij het maken van de rij een symbool hebben opgeschreven. Formeel.

Definitie $|v|$.

Stel $x \in \sum^*$, $s \in \sum$; $|\Lambda| := 0$; $|xs| := |x| + 1$.

Er geldt dan voor $v \in \sum^*$, $w \in \sum^*$, $n \in \mathbb{N} \cup \{0\}$:

$$|vw| = |v| + |w| ;$$
$$|v^n| = n|v| .$$

Vaak nummeren we de symbolen in een symboolrij en schrijven een rij van lengte ℓ in de gedaante $t_1 \ldots t_\ell$ of ook $t_0 \ldots t_{\ell-1}$; we noemen t_i dan het i^e symbool in de rij. In overeenstemming met het gangbare spraakgebruik kiezen we de eerste gedaante en definiëren $s_i(v)$ voor $v \in \sum^*$, $i \in \mathbb{N}$, $i \leq |v|$ als volgt:

Stel $x \in \sum^*$, $s \in \sum$, $i \in \mathbb{N}$; $s_i(xs) := s_i(x)$ voor $i \leq |x|$; $s_{|x|+1}(xs) := s$.

Dan is voor $a \in \sum$, $b \in \sum$, $c \in \sum$:

$$s_1(a) = a , \; s_1(ab) = s_1(a) = a , \; s_2(ab) = b ,$$
$$s_1(abc) = s_1(ab) = a , \; s_2(abc) = s_2(ab) = b , \; s_3(abc) = c , \text{ enz.}$$

Een belangrijke en intuïtief duidelijke eigenschap is nu, dat een symboolrij van lengte $k + \ell$ op één en slechts één wijze geschreven kan worden als concatenatie van een symboolrij van lengte k en een symboolrij van lengte ℓ.

Eigenschap.

Stel $v \in \sum^*$, k geheel, $0 \leq k \leq |v|$. Dan bestaat er één en slechts één $w \in \sum^*$ en één en slechts één $u \in \sum^*$ zo dat $v = wu$ en $|w| = k$.

We geven geen bewijs van deze eigenschap. Uit deze eigenschap kunnen makkelijk enkele andere worden afgeleid:

Stel $v \in \sum^*$, $w \in \sum^*$, $u \in \sum^*$, $x \in \sum^*$. Dan geldt

als $vw = vu$, dan $w = u$;

als $wv = uv$, dan $w = u$;

als $vw = ux$, dan bestaat er een $p \in \sum^*$ zo dat

$(v = up \text{ en } pw = x)$

of

$(vp = u \text{ en } w = px)$.

We definiëren nog de <u>gespiegelde</u> van een symboolrij v (engels: reverse). Notatie v^R.

<u>Definitie v^R.</u>

Stel $x \in \sum^*$, $s \in \sum$; $\Lambda^R := \Lambda$; $(xs)^R := sx^R$.

Enkele eigenschappen. Stel $v \in \sum^*$, $w \in \sum^*$, $s \in \sum$, $n \in \mathbb{N} \cup \{0\}$:

$s^R = s$,

$(vw)^R = w^R v^R$,

$(v^n)^R = (v^R)^n$,

$(v^R)^R = v$.

Ter afsluiting nog een opgave.

<u>Opgave.</u>

Stel $v \in \sum^*$, $w \in \sum^*$, $vw = wv$. Dan bestaan er $u \in \sum^*$, $n \in \mathbb{N} \cup \{0\}$, $m \in \mathbb{N} \cup \{0\}$ zo dat $v = u^n$, $w = u^m$.

§2. Talen en grammatica's

Een deelverzameling van $\sum^*$ wordt een taal genoemd. Deze benaming is in sommige toepassingen misschien minder gelukkig, maar zo gebruikelijk, dat wij ons eraan zullen houden. We geven enkele voorbeelden van talen.

Voorbeeld.

$\sum = \{a,b\}$.

$\{a^m b^n \mid m \in \mathbb{N},\ n \in \mathbb{N}\}$, $\{a^n b^n \mid n \in \mathbb{N}\}$,

$\{w w^R \mid w \in \sum^*\}$ (even palindromen),

$\{w s w^R \mid w \in \sum^*, s \in \sum\}$ (oneven palindromen),

$\{w \in \sum^* \mid |w| = 5\}$, $\{w \in \sum^* \mid$ in w komen meer a's dan b's voor$\}$.

(Een palindroom is een symboolrij x, waarvoor geldt $x^R = x$.)

Op talen kunnen operaties worden uitgevoerd. Omdat talen verzamelingen zijn komen de gebruikelijke verzamelingstheoretische operaties in aanmerking. Laat L_1 en L_2 talen zijn.

De vereniging van L_1 en L_2 : $L_1 \cup L_2$,

De doorsnede van L_1 en L_2 : $L_1 \cap L_2$.

Het complement van L_1 (t.o.v. $\sum^*$): $\overline{L}_1 = \sum^* \backslash L_1$.

Daarnaast zijn er operaties, die samenhangen met het bijzondere karakter van de semigroep $(\sum^*,)$.

De concatenatie van L_1 en L_2 bestaat uit de concatenaties van symboolrijen in L_1 en symboolrijen in L_2:

$$L_1L_2 := \{vw \mid v \in L_1 , \ w \in L_2\} .$$

Concatenatie van talen is een binaire operatie op de verzameling der talen, dat is $P(\sum^*)$, de verzameling van alle deelverzamelingen van $\sum^*$. Het is eenvoudig na te gaan, dat concatenatie van talen associatief is: $(L_1L_2)L_3 = L_1(L_2L_3)$. De aldus ontstane semigroep heeft echter ook een eenheidselement, nl. $\{\Lambda\}$, de taal die uitsluitend het lege woord bevat. Voor $L \in P(\sum)$ geldt namelijk:

$$\{\Lambda\}L = L\{\Lambda\} = L .$$

$(P(\sum^*),)$ is dus een monoïde.

Pas op!

Verwar $\{\Lambda\}$ niet met $\emptyset$, de lege taal, die helemaal geen woorden bevat, en die zich t.o.v. de concatenatie heel anders gedraagt dan $\{\Lambda\}$:

$$\emptyset L = L\emptyset = \emptyset .$$

We kunnen nu ook machten van talen invoeren:

$$L^0 := \{\Lambda\} ; \quad L^{n+1} := L^n L .$$

Een zeer belangrijke operatie is de vereniging van alle machten van een taal L. Deze operatie wordt de steroperatie van Kleene genoemd en L^* genoteerd:

$$L^* := \bigcup_{n=0}^{\infty} L^n .$$

Opmerking.

We hebben al eerder een ster gebruikt en vragen ons af of dat geen verwarring kan geven. Eerder gaf $\sum^*$ de verzameling van alle symboolrijen gevormd uit de symbolen van $\sum$ aan. Omdat $\sum \subset \sum^*$, is $\sum$ ook een taal, zodat $\sum^*$ volgens de nieuwe definitie ook zin heeft. De uitkomst is dan echter hetzelfde, immers

$$\sum^n = \{x \in \sum^* \mid |x| = n\},$$

waaruit de overeenstemming direct volgt.

Wat minder gangbaar is L^+ voor een willekeurige taal met definitie

$$L^+ := \bigcup_{n=1}^{\infty} L^n .$$

Pas op!

Als $\Lambda \in L$, dan $L^+ = L^*$.

Opgave.

Bewijs dat $\{a^m b^n \mid m \in \mathbb{N} \cup \{0\},\ n \in \mathbb{N} \cup \{0\}\}^* = \{a,b\}^*$.

De gespiegelde L^R van een taal wordt als volgt gedefinieerd:

$$L^R := \{x^R \mid x \in L\} .$$

Het algemene begrip taal is zeer algemeen en vertoont weinig structuur. Wij leggen ons beperkingen op door te eisen dat de taal gevormd is met behulp van een grammatica. De definitie is geïnspireerd door de grammatica's voor

natuurlijke talen maar ook zoals ze gebruikt worden bij de definitie van programmeertalen. De bij een grammatica gebruikte begrippen zoals "onderwerp" en "gezegde" bij natuurlijke talen, of "identifier", "procedure", "statement", "if clause" bij programmertalen, geven we formeel weer met hulpsymbolen. In een symboolrij, waarin hulpsymbolen voorkomen, kunnen deze door symboolrijen worden vervangen op grond van vervangingsregels, die er in een programmeertaaldefinitie bijvoorbeeld als volgt uit kunnen zien:

<identifier> ::= <letter> ,

<identifier> ::= <identifier> <letter> ,

<identifier> ::= <identifier> <digit> .

Er wordt verder een hulpsymbool als startsymbool gekozen. Een symboolrij, die geen hulpsymbolen bevat en die verkregen kan worden door, uitgaande van het startsymbool, een rij toegelaten vervangingen uit te voeren, wordt tot de taal gerekend.

Bij de formele definitie beginnen we met een iets algemener geval, waarbij in plaats van één hulpsymbool een symboolrij wordt vervangen.

Definitie grammatica G:

$G = (V, \sum, R, S)$, waarin

V is eindige verzameling symbolen ,

$\sum \subset V$,

$R \subset V^* \times V^*$, R eindig ,

$S \in V \setminus \sum$.

De elementen van $\sum$ heten eindsymbolen (engels: terminal symbols). De elementen van $V\backslash\sum$ heten hulpsymbolen (engels: nonterminal symbols). In de theorie van de formele talen is het algemeen gebruikelijk om hulpsymbolen met hoofdletters aan te duiden. De elementen van R heten regels en zijn per definitie paren (u,v) met $u \in V^*$, $v \in V^*$; R is dus op te vatten als een relatie op V^*. Zoals gebruikelijk bij relaties schrijven we de geldigheid ervan met een tussengeplaatst symbool. In ons geval:

$$\text{in plaats van } (u,v) \in R \text{ schrijven we } u \underset{G}{\rightarrow} v .$$

We noemen u het linkerlid en v het rechterlid van de regel. Vaak wordt van R nog geëist dat $u \neq \Lambda$, of zelfs dat u tenminste één hulpsymbool bevat. Het hierboven beschreven geval van vervanging van één hulpsymbool levert een contekstvrije grammatica.

Definitie.
Een grammatica $G = (V,\sum,R,S)$ heet contekstvrij als voor alle $(u,v) \in R$ geldt $u \in V\backslash\sum$.

Tenslotte heet S het startsymbool.

De vervanging van u door v op grond van een regel wordt formeel beschreven met een relatie $\underset{G}{\Rightarrow}$ op V^*:

Stel $x \in V^*$, $y \in V^*$;

$$x \underset{G}{\Rightarrow} y \;:\Leftrightarrow\; \exists_{w\in V^*}\, \exists_{z\in V^*}\, \exists_{(u,v)\in R}\, [x = wuz\,,\; y = wvz] .$$

Herhaaldelijke vervanging op grond van een regel wordt beschreven met een relatie $\overset{*}{\underset{G}{\Rightarrow}}$ op V^*; als $x = x_1 \underset{G}{\Rightarrow} x_2 \underset{G}{\Rightarrow} \ldots \underset{G}{\Rightarrow} x_n = y$, dan schrijven we $x \overset{*}{\underset{G}{\Rightarrow}} y$.
Wat formeler opgeschreven:

Stel $x \in V^*$, $y \in V^*$;

$$x \overset{*}{\underset{G}{\Rightarrow}} y :\Leftrightarrow \exists_{n \in \mathbb{N}} \exists_{(x_1,\ldots,x_n) \in (V^*)^n} [x_1 = x,\ x_n = y,$$

$$\forall_{i \in \{1,\ldots,n-1\}} [x_i \underset{G}{\Rightarrow} x_{i+1}]] .$$

Let wel, dat in deze definitie $n = 1$ toegestaan is; in dat geval is $x = y$:

$$x \overset{*}{\underset{G}{\Rightarrow}} x \quad \text{geldt voor alle } x \in V^* .$$

De rij $x_1,\ldots,x_n$ in bovenstaande definitie heet een <u>(n - 1)-stapsafleiding</u>. De relatie $\overset{*}{\underset{G}{\Rightarrow}}$ heet de <u>reflexief-transitieve afsluiting</u> van $\underset{G}{\Rightarrow}$; het is namelijk de als verzameling paren kleinste relatie op V^*, die reflexief en transitief is en die $\underset{G}{\Rightarrow}$ omvat.

Als het uit de contekst duidelijk is over welke grammatica G we het hebben, schrijven we wel $\Rightarrow$ in plaats van $\underset{G}{\Rightarrow}$ en $\overset{*}{\Rightarrow}$ in plaats van $\overset{*}{\underset{G}{\Rightarrow}}$. Verder schrijven we $x \overset{*}{\underset{G}{\Rightarrow}} y$ ook wel eens voor een afleiding van x naar y.

Tenslotte de taal voortgebracht door een grammatica:

<u>Definitie</u>.

Stel $G = (V, \sum, R, S)$ een grammatica;

$$L(G) := \{x \in \sum^* \mid S \overset{*}{\underset{G}{\Rightarrow}} x\} .$$

Voorbeelden.

1. $V = \{S,a,b\}$, $\sum = \{a,b\}$,

   $R = \{(S,aSa) , (S,bSb) , (S,\Lambda)\}$.

   Een voorbeeld van een afleiding is

   S , aSa , abSba , abbSbba , abbbba .

   L(G) = verzameling der even palindromen over $\{a,b\}$.

2. Zelfde V en $\sum$ als in 1; $R = \{(S,aSb) , (S,ab)\}$.

   $L(G) = \{a^n b^n \mid n \in \mathbb{N}\}$.

3. $V = \{S,T,a,b\}$, $\sum = \{a,b\}$,

   $R = \{(S,aS) , (S,abT) , (T,bT) , (T,\Lambda)\}$.

   $L(G) = \{a^n b^m \mid n \in \mathbb{N} , m \in \mathbb{N}\}$.

4. $V = \{S,A,B,a,b\}$, $\sum = \{a,b\}$,

   $R = \{(S,ABS) , (AB,BA) , (BA,AB) , (S,\Lambda) , (A,a) , (B,b)\}$.

   L(G) is de verzameling van de elementen van $\{a,b\}^*$ die evenveel a's als b's bevatten.

De grammatica's in Voorbeelden 1, 2 en 3 zijn contekstvrij, de grammatica in Voorbeeld 4 niet.

Definitie.

Een taal $L \subset \sum^*$ heet contekstvrij als er een contekstvrije grammatica G bestaat met $\sum$ als verzameling eindsymbolen, waarvoor L(G) = L.

Let wel dat bij gegeven taal L bij de keuze van G een verzameling hulpsymbolen kan worden gekozen. Die verzameling moet eindig zijn, maar overigens is er aan de omvang ervan geen grens gesteld.

Opmerking.

Het is mogelijk dat een grammatica, die niet contekstvrij is een taal voortbrengt, die wel contekstvrij is, omdat er een andere grammatica bestaat, die dezelfde taal voortbrengt en deze laatste grammatica contekstvrij is. Zo is de grammatica in voorgaand Voorbeeld 4 niet contekstvrij, maar de erdoor voortgebrachte taal wel.

In de hiervoor gegeven voorbeelden van contekstvrije grammatica's komt in de rechterleden van de regels ten hoogste één hulpsymbool voor. Ten gevolge daarvan geldt hetzelfde voor elke in een afleiding voorkomende symboolrij. De plaats waar in die symboolrij een regel kan worden toegepast, ligt dus ook vast. Omdat er echter wel eens meer dan één regel is met hetzelfde linkerlid, kan er toch keuzevrijheid zijn in de toe te passen regel. In het nu volgende voorbeeld kan er ook vrijheid van keuze zijn in de plaats waar een regel wordt toegepast.

Voorbeeld 5.

$V = \{S,A,B,a,b\}$, $\sum = \{a,b\}$,

$R = \{(S,AB)\ ,\ (A,AA)\ ,\ (B,BB)\ ,\ (A,a)\ ,\ (B,b)\}$.

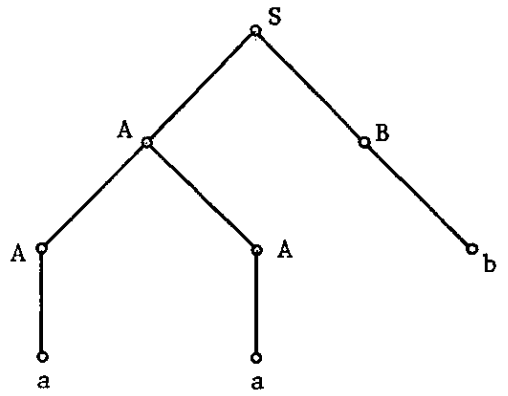
De taal is dezelfde als in Voorbeeld 3. Voorbeeld van een afleiding:

S , AB , AAB , aAB , aaB , aab .

Na één stap staat er AB en is er keuze of op A of op B een regel zal worden toegepast. Hier is A gekozen; B blijft dan staan en zal later ook nog eens moeten worden vervangen, omdat de laatste symboolrij uitsluitend eindsymbolen bevat. De symboolrijen, die successievelijk uit A voortkomen en die uit B, ontwikkelen zich geheel onafhankelijk van elkaar: A , AA , aA , aa en B , b. Men kan de volgorde van toepassingen van de regels dus veranderen zonder dat dit invloed heeft op het eindresultaat. Zo zou men de volgende afleiding kunnen verkrijgen:

S , AB , Ab , AAb , Aab , aab .

Hoewel deze twee afleidingen er geheel verschillend uitzien, willen we ze toch als niet essentiëel verschillend beschouwen. Men kan dit verschil op meer aanschouwelijke wijze wegwerken door bovenstaande afleidingen te vervangen door een afleidingsboom, hiernaast afgebeeld. Om in de boom de afleiding te volgen, moet men de boom van boven naar beneden bekijken. De eerste stap gaat van S naar AB. Wat er daarna met die A en B gebeurt, wordt gescheiden behandeld en de boom geeft geen uitsluitsel over de volgorde waarin dit gebeurt. De A gaat over in AA en elk van deze A's in a, de B gaat over in b.



Er zou nu aanleiding zijn tot een formele definitie van een afleidingsboom behorende bij een contekstvrije grammatica en van de symboolrij die door die afleidingsboom wordt afgeleid. We zien daat echter vanaf, maar zullen desondanks in de toekomst wel met afleidingsbomen werken.

Een andere manier om de in Voorbeeld 5 gesignaleerde veelvuldigheid van afleidingen met hetzelfde resultaat weg te werken, is de afspraak om steeds een regel toe te passen op het meest links geplaatste hulpsymbool in de symboolrij. Een dergelijke afleiding wordt linkspreferent (engels: leftmost) genoemd. Door verwisseling van volgorde in de toepassing van regels kan men bij een gegeven afleiding een linkspreferente afleiding vinden, die dezelfde symboolrij oplevert. De eerste afleiding in Voorbeeld 5 is linkspreferent, de tweede niet.

We maken de afspraak, dat als we, zonder specificatie van de verzameling $\sum$ der eindsymbolen, in een bepaalde contekst over meer dan één taal spreken, al deze talen bij dezelfde $\sum$ behoren.

Stelling 2.1.

De vereniging en de concatenatie van twee contekstvrije talen zijn contekstvrij; de Kleene-ster van een contekstvrije taal is contekstvrij.

Bewijs. Stel dat L_1 en L_2 contekstvrije talen zijn. Er bestaan dan contekstvrije grammatica's $G_1 = (V_1,\sum,R_1,S_1)$ resp. $G_2 = (V_2,\sum,R_2,S_2)$, die L_1 resp. L_2 genereren. We mogen nu veronderstellen, dat $(V_1\backslash\sum) \cap (V_2\backslash\sum) = \emptyset$. Als dat namelijk niet het geval is, vervangen we de hulpsymbolen van G_2 door andere met dienovereenkomstige aanpassing van R_2 en S_2. De gewijzigde grammatica

zal dan ook L_2 genereren. Om een grammatica voor $L_1 \cup L_2$ (en later voor L_1L_2 en L_1^*) te maken, kiezen we een nieuw symbool dat niet in $V_1 \cup V_2$ ligt. We definiëren nu $G_3 = (V_3, \Sigma, R_3, S_3)$ als volgt. Noem het nieuwe symbool S_3 en kies

$$V_3 := V_1 \cup V_2 \cup \{S_3\},$$

$$R_3 := R_1 \cup R_2 \cup \{(S_3, S_1), (S_3, S_2)\}.$$

Deze grammatica is contekstvrij.

Bewering.

$L(G_3) = L_1 \cup L_2$. Stel $x \in L_1 \cup L_2$. Twee gevallen:

1° $x \in L_1$. Dan is er een afleiding $S_1 \overset{*}{\underset{G_1}{\Rightarrow}} x$, maar dan is er ook een afleiding $S_3 \underset{G_3}{\Rightarrow} S_1 \overset{*}{\underset{G_3}{\Rightarrow}} x$; immers de afleiding $S_1 \overset{*}{\underset{G_1}{\Rightarrow}} x$ kan als bestanddeel van een afleiding bij G_3 worden gebruikt, omdat $R_1 \subset R_3$. Dus $x \in L(G_3)$.

2° $x \notin L_1$. Omdat $x \in L_1 \cup L_2$, geldt dan $x \in L_2$. Verder analoog met geval 1°.

Daarmee is $L_1 \cup L_2 \subset L(G_3)$ bewezen. Stel nu $x \in L(G_3)$. Er is dan een afleiding $S_3 \overset{*}{\underset{G_3}{\Rightarrow}} x$. Deze begint met S_3, S_1 of met S_3, S_2; er zijn namelijk geen andere regels met S_3 in het linkerlid. Als het begin S_3, S_1 is, dan volgt daarop $S_1 \overset{*}{\underset{G_3}{\Rightarrow}} x$. Dit levert echter ook een afleiding $S_1 \overset{*}{\underset{G_1}{\Rightarrow}} x$. Met inductie langs de afleiding bewijst men namelijk, dat de in de afleiding voorkomende symboolrijen uitsluitend hulpsymbolen uit $V_1 \backslash \Sigma$ bevatten en dat er daarom uitsluitend regels uit R_1 worden toegepast (hierbij wordt $(V_1 \backslash \Sigma) \cap (V_2 \backslash \Sigma) = \emptyset$ gebruikt!). Daaruit volgt $x \in L_1$, dus $x \in L_1 \cup L_2$. Als het begin S_3, S_2 is, vinden we op analoge wijze $x \in L_2$, dus $x \in L_1 \cup L_2$. Daarmee is het bewijs voltooid, dat $L_1 \cup L_2 = L(G_3)$. Daaruit volgt dat $L_1 \cup L_2$ contextvrij is.

We definiëren nu $G_4 = (V_4, \sum, R_4, S_4)$ als volgt. Noem het nieuwe symbool nu S_4 en kies

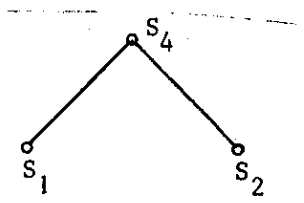
$$V_4 := V_1 \cup V_2 \cup \{S_4\} ,$$

$$R_4 := R_1 \cup R_2 \cup \{(S_4, S_1 S_2)\} .$$

Deze grammatica is contekstvrij.

Bewering.

$L(G_4) = L_1 L_2$. Stel $x \in L_1 L_2$, dan bestaan er $y \in L_1$ en $z \in L_2$ zó, dat $x = yz$. Er zijn dan afleidingen $S_1 \overset{*}{\underset{G_1}{\Rightarrow}} y$ en $S_2 \overset{*}{\underset{G_2}{\Rightarrow}} z$, maar dan is er ook een afleiding $S_4 \underset{G_4}{\Rightarrow} S_1 S_2 \overset{*}{\underset{G_4}{\Rightarrow}} y S_2 \overset{*}{\underset{G_4}{\Rightarrow}} yz$; immers door in de afleiding $S_1 \overset{*}{\underset{G_1}{\Rightarrow}} y$ achter alle symboolrijen S_2 te schrijven, krijgen we $S_1 S_2 \overset{*}{\underset{G_4}{\Rightarrow}} y S_2$ en door in de afleiding $S_2 \overset{*}{\underset{G_2}{\Rightarrow}} z$ voor alle symboolrijen y te schrijven, krijgen we $y S_2 \overset{*}{\underset{G_4}{\Rightarrow}} yz$. Dus $x \in L(G_4)$; $L_1 L_2 \subset L(G_4)$ is daarmee aangetoond. Stel nu $x \in L(G_4)$. Een afleidingsboom bij G_4 die x oplevert, ziet er bovenaan als volgt uit:



omdat $(S_4, S_1 S_2)$ de enige regel in R_4 is met linkerlid S_4. Wat in deze boom onder S_1 zit, bevat uitsluitend hulpsymbolen uit $V_1 \backslash \sum$, waarbij uitsluitend regels uit R_1 worden toegepast. Dit gedeelte van de boom levert dan $S_1 \overset{*}{\underset{G_1}{\Rightarrow}} y$ voor zekere $y \in \sum^*$. Op analoge wijze levert het gedeelte van de boom onder S_2 een afleiding $S_2 \overset{*}{\underset{G_2}{\Rightarrow}} z$ voor zekere $z \in \sum^*$. Maar dan is $y \in L_1$, $z \in L_2$ en $x = yz \in L_1 L_2$. Daarmee is het bewijs voltooid, dat $L_1 L_2 = L(G_4)$. Daaruit volgt dat $L_1 L_2$ contekstvrij is.

We definiëren nu $G_5 = (V_5, \sum, R_5, S_5)$ als volgt. Noem het nieuwe symbool S_5 en kies:

$$V_5 := V_1 \cup \{S_5\} ;$$

$$R_5 := R_1 \cup \{(S_5, S_5S_1), (S_5, \Lambda)\} .$$

Deze grammatica is contekstvrij.

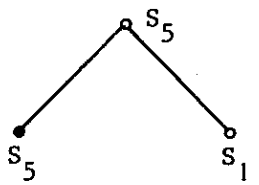
Bewering.

$L(G_5) = L_1^*$. Stel $x \in L_1^*$, dan is $x = \Lambda$ of er is een $n \in N$ en een rij $(x_1, \ldots, x_n) \in \sum^n$ zo dat $x = x_1 \ldots x_n$. Als $x = \Lambda$, dan $S_5 \underset{G_5}{\Rightarrow} \Lambda = x$. In het andere geval zijn er afleidingen $S_1 \overset{*}{\underset{G_1}{\Rightarrow}} x_i$ voor $i = 1, \ldots, n$. We maken nu de volgende afleiding:

$$S_5 \underset{G_5}{\Rightarrow} S_5S_1 \underset{G_5}{\Rightarrow} S_5S_1S_1 \underset{G_5}{\Rightarrow} \cdots \underset{G_5}{\Rightarrow} S_5S_1^n \underset{G_5}{\Rightarrow} S_1^n \overset{*}{\underset{G_5}{\Rightarrow}} x_1S_1^{n-1} \overset{*}{\underset{G_5}{\Rightarrow}}$$

$$\overset{*}{\underset{G_5}{\Rightarrow}} x_1x_2S_1^{n-2} \overset{*}{\underset{G_5}{\Rightarrow}} \cdots \overset{*}{\underset{G_5}{\Rightarrow}} x_1x_2 \cdots x_n = x .$$

Dus $x \in L(G_5)$; $L_1^* \subset L(G_5)$ is daarmee aangetoond. Stel nu $x \in L(G_5)$. Er is dan een afleidingsboom bij G_5, die x oplevert. We bewijzen $x \in L_1^*$ met inductie naar het aantal knopen in de afleidingsboom. De eerste stap in een afleiding bij G_5 is (S_5, Λ) of (S_5, S_5S_1). In het eerste geval geldt $x = \Lambda \in L_1^*$. In het tweede geval ziet de afleidingsboom er bovenaan als volgt uit. Wat in deze boom onder de knoop • met S_5 zit is een afleidingsboom bij G_5 met minder knopen dan de hele boom. Uit de inductieveronderstelling mogen we concluderen, dat deze een afleiding $S_5 \overset{*}{\underset{G_5}{\Rightarrow}} y$ met $y \in L_1^*$ oplevert.



Wat in de boom onder S_1 zit, levert op analoge wijze als in de vorige gevallen een afleiding $S_1 \overset{*}{\underset{G_1}{\Rightarrow}} z$ op en dus $z \in L_1$. Maar dan is $y = yz \in$ $\in L_1^* L_1 \subset L_1^*$. Daarmee is het bewijs voltooid, dat $L_1^* = L(G_5)$. Daaruit volgt, dat L_1^* contekstvrij is. □

Met doorsnede en complement van talen ligt de zaak anders. Er bestaan contekstvrije talen L_1 en L_2 zó, dat $L_1 \cap L_2$ niet contekstvrij is en er bestaat een taal L zó, dat $\overline{L}$ niet contekstvrij is. Om voorbeelden daarvan te kunnen behandelen, moeten we beschikken over een methode om vast te stellen, dat een taal niet contekstvrij is. Een gebruikelijke gang van zaken om dit te bewerkstelligen is de volgende.

Kies een eigenschap van talen en bewijs dat alle contekstvrije talen die eigenschap hebben. Een taal, die die eigenschap niet heeft, is dan niet contekstvrij.

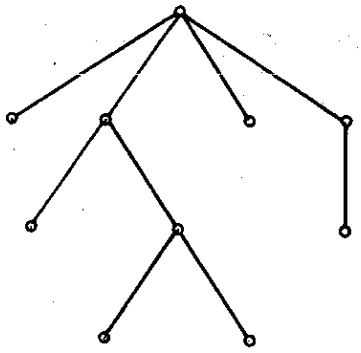
Er zijn verscheidene zulke eigenschappen bekend. Wij behandelen er hier één van. Daartoe is echter nog een inleiding nodig.

Laat G een contekstvrije grammatica zijn en definiëer

n := aantal hulpsymbolen ,

m := maximum van de lengtes van de rechterleden van regels .

Aan een afleidingsboom kunnen we een <u>hoogte</u> toekennen, dat is het maximale aantal stappen, dat men langs takken van knoop naar knoop van boven naar beneden kan doen.

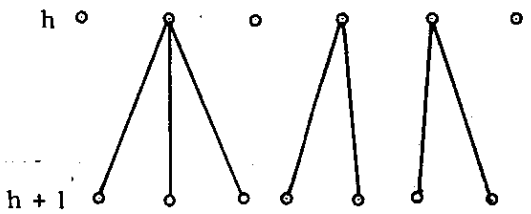


De hiernaast getekende boom heeft hoogte 3. We zullen nu aantonen, dat als een afleidingsboom bij G een hoogte h heeft en de bij deze boom behorende symboolrij w is, dan $|w| \leq m^h$. Knopen in een boom, van waaruit naar beneden geen takken gaan, noemen we <u>eindknopen</u> (of ook wel <u>bladeren</u>) van de boom. Een knoop van een afleidingsboom, waar een eindsymbool bij staat, is een eindknoop. Bij een eindknoop kan echter ook een hulpsymbool A staan, namelijk als ter plaatse de regel (A,Λ) wordt toegepast. Elk voorkomen van een symbool in w correspondeert met een eindknoop (het zijn namelijk eindsymbolen), dus $|w| \leq$ aantal eindknopen in de boom.

Verder geldt voor een afleidingsboom bij G, dat bij iedere knoop het aantal naar beneden gaande takken (de vertakking van de knoop) $\leq m$ is. Het is dus voldoende het volgende te bewijzen:

Een boom van hoogte h, waarin alle knopen vertakking $\leq m$ hebben, heeft ten hoogste m^h eindknopen.

Om dit te bewijzen, behandelen we eerst het geval $m = 0$. In dat geval is $h = 0$ en het aantal eindknopen $\leq 1 = 0^0$. We veronderstellen nu $m \geq 1$ en passen volledige inductie naar h toe. Dat de bewering klopt voor $h = 0$ (of $h = 1$) is eenvoudig na te gaan. Stel nu de bewering waar voor bomen met hoogte h en neem een boom met hoogte $h + 1$. We maken daarvan een boom met hoogte h door hetgeen onder niveau h zit eraf te knippen. Stel dat de zo verkregen

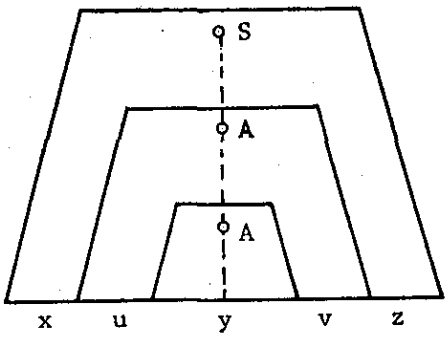


boom i eindknopen heeft en er op niveau h j knopen zijn, waar in de oorspronkelijke boom takken naar niveau h + 1 van uitgaan. Omdat de vertakking van iedere knoop ≤ m is, is het aantal eindknopen in de oorspronkelijke boom

$$\leq i - j + mj = i + j(m-1) .$$

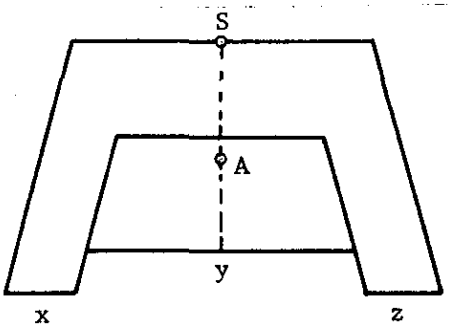
Het is duidelijk dat $j \leq i$, dus $i + j(m-1) \leq mi$. De inductieveronderstelling levert $i \leq m^h$; het aantal eindknopen in de gegeven boom is dus $\leq m^{h+1}$.

Stel nu $w \in L(G)$, waarvoor geldt $|w| > m^n$. Omdat $w \in L(G)$, is er een afleidingsboom bij G, waar w bijhoort. Er zouden er meer dan één kunnen zijn; we kiezen er een met een minimaal aantal knopen. Voor de hoogte h van deze boom geldt dan $|w| \leq m^h$ en dus $h > n$. Er is dan van boven naar beneden een pad ter lengte h in deze boom; in dit pad zitten h + 1 knopen. Behalve eventueel de onderste knoop, hoort bij alle knopen echter een hulpsymbool en omdat $h > n$ zijn er zeker twee verschillende knopen waar hetzelfde hulpsymbool bijhoort. We kunnen dit nog wat verscherpen door van dit pad slechts het onderste stuk ter lengte n + 1 (met n + 2 knopen) te nemen. In dit pad zitten twee verschillende knopen, waar hetzelfde hulpsymbool A bijhoort. We kunnen de boom nu schematisch in stukken verdelen zoals weergegeven in de volgende tekening.



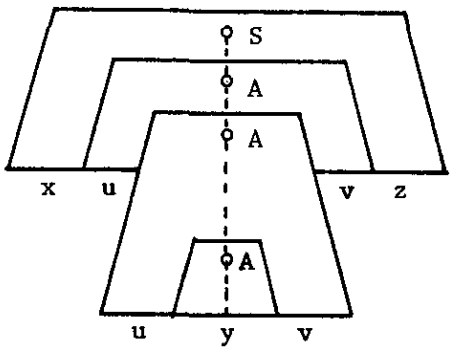
Er geldt dan $w = xuyvz$. Verder geldt, omdat de deelboom, die bij de bovenste knoop met A behoort, een hoogte $\leq n + 1$ heeft, dat $|uyv| \leq m^{n+1}$.

We gaan nu "boomchirurgie" bedrijven. We kunnen de grote deelboom bij A eruit snijden en vervangen door de kleine deelboom bij A. Het resultaat is dan weer een afleidingsboom, die er schematisch als volgt uitziet:



De bijbehorende symboolrij is $xyz \in L(G)$. Omdat deze boom minder knopen heeft dan de vorige geldt $xyz \neq xuyvz$, waaruit volgt, dat u en v niet beide de lege symboolrij zijn (dit is korter weer te geven als $uv \neq \Lambda$ of $|uv| > 0$).

We kunnen echter ook uit de oorspronkelijke boom de kleine deelboom bij A eruit snijden en deze vervangen door de grote deelboom bij A. Dit levert een afleidingsboom die er schematisch als volgt uitziet:



De bijbehorende symboolrij is $xu^2yv^2z \in L(G)$. Dit procédé kan onbepaald vaak herhaald worden, hetgeen oplevert dat $xu^kyv^kz \in L(G)$ voor alle $k \in \mathbb{N}$.

Hiermee is de volgende stelling bewezen.

Stelling 2.2.

Stel $G = (V,\sum,R,S)$ een contekstvrije grammatica, $n = \#V\backslash\sum$, m = maximum van de lengtes van de rechterleden van de elementen van R, $w \in L(G)$, $|w| > m^n$. Dan bestaan er x, y, z, u, v, alle in $\sum^*$ zó, dat

$$w = xuyvz\ ,\quad |uyv| \leq m^{n+1}\ ,\quad |uv| > 0$$

en

$$\{xu^kyv^kz \mid k \in \mathbb{N} \cup \{0\}\} \subset L(G)\ .$$

Opmerking.

Omdat $|uv| > 0$, is $|xu^kyv^kz|$ een monotoon stijgende functie van k en daaruit volgt dat $\{xu^kyv^kz \mid k \in \mathbb{N} \cup \{0\}\}$ een oneindige verzameling is. Daaruit volgt weer, dat als $L(G)$ eindig is, voor alle $w \in L(G)$ geldt $|w| \leq m^n$.

Bovenstaande stelling gaat over een contekstvrije grammatica. Zij is echter makkelijk om te zetten in een stelling over contekstvrije talen, waarin niet meer expliciet over grammatica's gesproken wordt. Dit levert een stelling, die in de engelstalige literatuur meestal als "pumping theorem" (of ook "pumping lemma") wordt betiteld.

Stelling 2.3 (pompstelling).

Als L een contekstvrije taal is met $\sum$ als verzameling symbolen, dan bestaat er een $p \in \mathbb{N}$ zó, dat voor iedere $w \in L$ met $|w| > p$ geldt, dat er x, y, z, u, v, alle in $\sum^*$, bestaan zó, dat

$$w = xuyvz\ ,\quad |uyv| \leq p\ ,\quad |uv| > 0$$

en

$$\{xu^kyv^kz \mid k \in \mathbb{N} \cup \{0\}\} \subset L\ .$$

Bewijs. Kies een contekstvrije grammatica G zó, dat L = L(G). Bij G behoren getallen m en n als in de vorige stelling. Kies nu $p := m^{n+1}$ en pas de vorige stelling toe. □

We geven een voorbeeld. Kies $\sum := \{a,b,c\}$ en

$$L_1 := \{a^n b^n c^m \mid n \in \mathbb{N} \cup \{0\},\ m \in \mathbb{N} \cup \{0\}\},$$

$$L_2 := \{a^m b^n c^n \mid n \in \mathbb{N} \cup \{0\},\ m \in \mathbb{N} \cup \{0\}\}.$$

Opgave.

Bewijs dat L_1 en L_2 contekstvrij zijn (denk aan Voorbeeld 2).

Uiteraard geldt $L_1 \cap L_2 = \{a^n b^n c^n \mid n \in \mathbb{N} \cup \{0\}\}$. We tonen aan, dat deze taal niet contekstvrij is. Stel dat $L_1 \cap L_2$ wèl contekstvrij is en laat p een getal zijn, dat op grond van de pompstelling bij $L_1 \cap L_2$ bestaat. Kies een $n > \frac{1}{3}p$, dan kan $a^n b^n c^n$ geschreven worden in de vorm xuyvz als in de pompstelling. In u kunnen geen twee verschillende symbolen voorkomen. Zouden in u bij voorbeeld a en b beide voorkomen, dan zou in xu^2yv^2z een b links van een a voorkomen in strijd met $xu^2yv^2z \in L_1 \cap L_2$. Op analoge wijze kunnen in v geen twee verschillende symbolen voorkomen. Tenminste één van de drie symbolen a, b, c komt noch in u, noch in v voor. Noem zo'n symbool t ($t \in \{a,b,c\}$). Dan is echter het aantal malen dat t voorkomt in xuyvz = = het aantal malen dat t voorkomt in xu^2yv^2z; verder $|xuyvz| < |xu^2yv^2z|$ en $xuyvz \in L_1 \cap L_2$, $xu^2yv^2z \in L_1 \cap L_2$. Dit is een contradictie, want het ene aantal is $\frac{1}{3}|xuyvz|$ en het andere aantal is $\frac{1}{3}|xu^2vy^2z|$.

Hiermee zijn twee contekstvrije talen gevonden, waarvan de doorsnede niet contekstvrij is. Daaruit volgt nu ook, dat het niet waar is, dat van iedere contekstvrije taal het complement contekstvrij is. Zou dat namelijk wèl waar zijn, dan zouden we op grond van de verzamelingstheoretische betrekking

$$L_1 \cap L_2 = \overline{(\overline{L_1} \cup \overline{L_2})} ,$$

toegepast op bovengenoemde talen L_1 en L_2, vinden, dat het rechterlid contekstvrij is, terwijl het linkerlid het niet is.

Opmerking.

We hebben hiermee geen expliciet voorbeeld gevonden van een contekstvrije taal, waarvan het complement niet contekstvrij is. We weten namelijk nog niet bij welk van de drie complementeringen in het rechterlid het misgaat met het contekstvrij zijn. Het zou kunnen dat $\overline{L_1}$ of $\overline{L_2}$ niet contekstvrij is en dan is L_1 of L_2 een voorbeeld. Het zou echter ook kunnen, dat $\overline{L_1}$ en $\overline{L_2}$ en dan ook $\overline{L_1} \cup \overline{L_2}$ contekstvrij zijn; dan is $\overline{L_1} \cup \overline{L_2}$ een voorbeeld. Zonder bewijs wordt hier vermeld, dat de laatstgenoemde mogelijkheid in werkelijkheid vervuld is.

Opgave.

Als L contekstvrij is, dan is L^R contekstvrij. Bewijs dat.

We besluiten met enkele opmerkingen over algemene grammatica's. De ontwerper van de theorie van deze grammatica's, N. Chomsky, heeft een hiërarchie van grammatica's en bijbehorende talen opgesteld, die hij met nummers heeft beschreven. Deze nummers worden typen genoemd en wel gebruikt hij 0, 1, 2, 3.

Daarbij correspondeert type 0 met het hierboven gedefinieerde algemene begrip grammatica en type 2 met de contekstvrije grammatica. Voor de beschrijving van type 1 leggen we in eerste instantie de beperking op, dat alle regels de volgende gedaante hebben:

$$(qAr,qpr) \quad \text{met } A \in V\setminus\textstyle\sum, \quad q \in V^*, \quad r \in V^*, \quad p \in V^+.$$

Let wel: $p \neq \Lambda$. Men ziet makkelijk in, dat in dat geval bij een afleiding de lengtes $|x|$ van de in die afleiding voorkomende symboolrijen, langs die afleidingen een niet-dalende functie vormen. Daaruit volgt $\Lambda \notin L(G)$. Om nu echter ook talen te verkrijgen, die de lege symboolrij bevatten, laten we ook nog de regel (S,Λ) toe; als deze regel voorkomt, stellen we de eis, dat in de andere regels S niet in het rechterlid voorkomt. Door dit te doen, kan de regel (S,Λ) uitsluitend voor de afleiding van Λ worden gebruikt. (Opgave. Ga dit na!)

Een grammatica van type 1 wordt ook contekstgevoelig genoemd (engels: context sensitive).

Een contekstvrije grammatica met de eigenschap, dat in elke regel in het rechterlid ten hoogste één voorkomen van een hulpsymbool is, wordt lineair genoemd. Als bovendien zo'n hulpsymbool altijd geheel rechts staat, noemen we de grammatica rechtslineair of van type 3. Meer expliciet geformuleerd: Alle regels hebben de gedaante

$$(A,qB) \text{ of } (A,q) \text{ met } A \in V\setminus\textstyle\sum, \quad B \in V\setminus\textstyle\sum, \quad q \in \textstyle\sum^*.$$

Een taal, waarvoor een grammatica van type i bestaat, die die taal genereert, wordt van type i genoemd. De verzameling van talen van type i duiden we aan met L_i ($i \in \{0,1,2,3\}$). Let wel dat L_i ook afhangt van de keuze van $\sum$. Op de eigenaardigheden van sommige der bovenstaande definities komen we nog terug. Zonder bewijs vermelden we nu alvast:

$$L_3 \subsetneq L_2 \subsetneq L_1 \subsetneq L_0 .$$

TECHNISCHE HOGESCHOOL EINDHOVEN

Onderafdeling der Wiskunde en Informatica

AUTOMATENTHEORIE EN FORMELE TALEN (2K170)

Hoofdstuk 2. Automaten

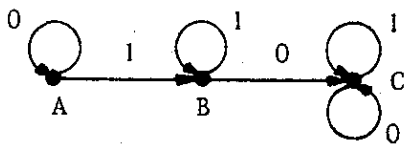
lentetrimester 1984

HOOFDSTUK 2. Automaten

§3. Eindige automaten

In Hoofdstuk 1 hebben we talen verkregen door ze met grammatica's te genereren. We willen nu automaten bij de definitie van talen gaan gebruiken. Een eerste ruwe beschrijving van een automaat levert een doos, waar informatie kan worden ingestopt en waar vervolgens ook weer informatie uitkomt. Informatie brengen we in een automaat door invoer (engels: "input") van een symbool; herhaling van dit proces leidt tot invoer van een symboolrij. De automaat zal op grond van ingevoerde symbolen en op grond van wat in zijn geheugen over eerdere invoer is opgeslagen bepaalde handelingen verrichten. Bij het eenvoudigste soort automaat bekommeren wij ons niet om hetgeen zich in het inwendige van de automaat afspeelt; men drukt dit wel eens uit door te zeggen, dat de automaat een zwarte doos is. We constateren alleen dat de automaat zich in verschillende toestanden (engels: "states") kan bevinden, waarvan het aantal bij een eindige automaat bovendien eindig is. Bij invoer van een symbool neemt de automaat een nieuwe toestand aan, die alleen afhangt van dat symbool en van de toestand voor de invoer van dat symbool. Dit leidt tot een overgangsfunctie (engels: "transition function" of "next state function"), die aan een paar, bestaande uit een toestand en een symbool, een toestand toevoegt. Deze stand van zaken kan zichtbaar worden gemaakt met een toestandsdiagram (engels: "state diagram"), dat is een gemerkte gerichte graaf (engels: "labeled directed graph"), waarvan de knopen de toestanden voorstellen en de gerichte takken (pijlen) gemerkt zijn met een symbool.

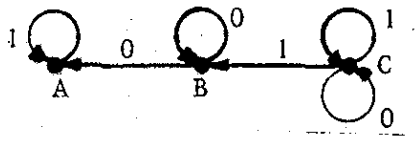
Als voorbeeld nemen we drie toestanden {A,B,C} en twee symbolen {0,1} en het volgende diagram:



De met een symbool gemerkte pijlen geven aan hoe bij lezing van dit symbool de toestand verandert: als in toestand B het symbool 0 gelezen wordt, gaat de automaat over in toestand C. Het feit echter, dat in elke situatie de actie van de automaat vastgelegd moet zijn, legt beperkingen op aan het toestandsdiagram. Bij elke knoop K en elk invoersymbool t moet er precies één pijl zijn, die in K begint en met t gemerkt is. Bovenstaand diagram voldoet aan deze eis.

We verruimen nu echter het begrip eindige automaat door de zojuist genoemde beperking te laten vervallen. Dit betekent dat we toelaten, dat in een knoop van het toestandsdiagram meer dan één pijl begint die met hetzelfde symbool gemerkt is. We interpreteren dit voor de werking van de automaat als volgt: in een bepaalde toestand kan de automaat bij lezing van een symbool nog keuze hebben voor de toestanden waarin hij overgaat. Verder is het mogelijk, dat er bij een knoop en een symbool geen pijl is, die in die knoop begint en met dat symbool gemerkt is: de werking van de automaat is in die toestand bij lezing van dat symbool geblokkeerd. Het feit dat er soms verschillende mogelijkheden voor de automaat zijn om zijn werking voort te zetten wordt uitgedrukt door te zeggen dat de automaat niet deterministisch is; een automaat, die wel aan de hierboven vermelde beperkingen voldoet, heet deterministisch.

Een voorbeeld van een diagram van een niet-deterministische automaat, wederom met knopen {A,B,C} en symbolen {0,1}:



In toestand B kan de automaat bij lezing van 0 zowel in toestand A als in toestand B overgaan; bij lezing van 1 blokkeert de automaat echter.

Behalve het opnemen van informatie door middel van invoer, willen we nu ook informatie door de automaat laten afgeven. Dit zou kunnen door de automaat uitvoer (engels: "output") te laten produceren, bijvoorbeeld in de vorm van uitvoersymbolen. Bij de toepassingen, die we nu op het oog hebben, is daaraan nog geen behoefte. We vinden het voorlopig voldoende, dat we kunnen waarnemen in welke toestand de automaat zich bevindt.

We hebben al opgemerkt, dat een symboolrij symbool voor symbool ingevoerd kan worden, waarbij telkens de toestand wordt aangepast. Als we nu een taal L hebben bij een symboolverzameling $\sum$ (dus $L \subset \sum^*$) en een automaat, waarvoor de elementen van $\sum$ invoersymbolen zijn, dan kunnen we de elementen w van $\sum^*$ invoeren. De automaat kan nu als acceptor voor de taal L dienen, als hij in staat is om ja of neen tegen een symboolrij te zeggen (ja als $w \in L$, neen als $w \notin L$). Na invoer van w is de toestand van de automaat waarneembaar. We verdelen daarom de verzameling der toestanden in twee klassen: de toegelaten en de verworpen toestanden. Verder merken we nog op, dat het effect van de invoer ook nog afhangt van de toestand, waarin begonnen wordt. Deze leggen we vast door de keuze van een begintoestand. Tenslotte kan er bij een niet-deterministische automaat tijdens de werking nog keuze zijn en deze keuzemogelijkheid kan zich nog herhalen bij het successievelijk invoeren van symbolen. De afspraak is nu, dat de symboolrij wordt toegelaten als tenminste

één rij mogelijke keuzen na lezing van de hele symboolrij tot een toegelaten toestand leidt. Dit betekent echter ook, dat als de automaat blokkeert, voordat de hele symboolrij gelezen is, dit tot verwerping leidt.

Definitie (niet-deterministische) eindige automaat.

Een 5-tupel $A = (S,\sum,\delta,s,F)$ heet een niet-deterministische eindige automaat (of kortweg eindige automaat), als geldt:

S is een eindige verzameling (elementen van S heten toestanden),

$\sum$ is een eindige niet-lege verzameling (elementen van $\sum$ heten invoersymbolen),

$\delta \subset S \times \sum \times S$,

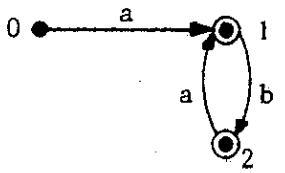
$s \in S$ (begintoestand),

$F \subset S$ (elementen van F heten toegelaten toestanden).

De interpretatie van δ is, dat, als $(q,t,r) \in \delta$, de automaat in toestand q bij lezing van t in toestand r kan overgaan.

Voorbeeld.

$A = (\{0,1,2\}, \{a,b\}, \{(0,a,1), (1,b,2), (2,a,1)\}, 0, \{1,2\})$ met toestandsdiagram



Op grond van het niet-deterministische karakter van de automaat zijn we gedwongen voor de beschrijving van de werking van de automaat meer dan één

toestand tegelijk te beschouwen en na te gaan wat er bij lezing van een symbool gebeurt. Dit geldt in versterkte mate als we de symbolen van een symboolrij stuk voor stuk invoeren. Dit formaliseren we als volgt.

Bij een eindige automaat $A = (S,\sum,\delta,s,F)$ definiëren we een afbeelding $\Delta : P(S) \times \sum^* \to P(S)$ door recurrentie naar de opbouw van de symboolrij in $\sum^*$:

Stel $V \in P(S)$, $u \in \sum^*$, $t \in \sum$:

$$\Delta(V,\Lambda) := V ,$$

$$\Delta(V,ut) := \{r \in S \mid \exists_{q\in\Delta(V,u)} [(q,t,r) \in \delta]\} .$$

De interpretatie van $\Delta(V,w)$ is de verzameling van de toestanden, die verkregen kunnen worden bij start in een toestand van V en lezing van de symboolrij w. Voor Δ geldt dan een <u>concatenatieformule</u>:

Als $V \in P(S)$, $v \in \sum^*$, $w \in \sum^*$, dan

$$\Delta(V,vw) = \Delta(\Delta(V,v),w) .$$

Dit bewijzen we door recurrentie naar de opbouw van w:

$$\Delta(V,v\Lambda) = \Delta(V,v) = \Delta(\Delta(V,v),\Lambda) .$$

$$\Delta(V,vut) = \{r \in S \mid \exists_{q\in\Delta(V,vu)} [(q,t,r) \in \delta]\} =$$

$$= \{r \in S \mid \exists_{q\in\Delta(\Delta(V,v),u)} [(q,t,r) \in \delta]\} =$$

$$= \Delta(\Delta(V,v),ut) .$$

Definitie taal geaccepteerd door eindige automaat.

Als $A = (S,\sum,\delta,s,F)$ een eindige automaat met bijbehorende afbeelding Δ is, dan heet

$$L(A) := \{x \in \sum^* \mid \Delta(\{s\},x) \cap F \neq \emptyset\}$$

de door A geaccepteerde taal.

Definitie deterministische eindige automaat.

Een eindige automaat $(S,\sum,\delta,s,F)$ heet deterministisch, als geldt

$$\forall_{q\in S} \forall_{t\in\sum} \exists!_{r\in S} [(q,t,r) \in \delta] .$$

Opmerking.

Dat de automaat deterministisch is, betekent, dat δ opgevat kan worden als een afbeelding $S \times \sum \to S$.

Dit is de gebruikelijke methode om determinisme bij eindige automaten te bepalen. Bij andere typen automaten is echter een andere begripsbepaling in zwang, die voor eindige automaten de volgende voorwaarde zou opleveren:

$$\forall_{q\in S} \forall_{t\in\sum} \forall_{r_1\in S} \forall_{r_2\in S} [(q,t,r_1) \in \delta \wedge (q,t,r_2) \in \delta \Rightarrow r_1 = r_2] .$$

Dit betekent, dat meer dan één keuze in een bepaalde situatie wordt uitgesloten, maar dat blokkering mogelijk blijft. Anders gezegd: δ kan opgevat worden als een partiële afbeelding $S \times \sum \to S$: misschien voor sommige paren niet gedefinieerd, maar ingeval gedefinieerd wel ondubbelzinnig vastgelegd. We zouden in dit geval van een zwak deterministische automaat kunnen spreken en bij een totale afbeelding van een sterk deterministische automaat.

Bij een deterministische eindige automaat $(S,\sum,\delta,s,F)$ bestaat er nu één en slechts één afbeelding $\delta' : S \times \sum^* \to S$ zó, dat geldt:

$$\forall_{q\in S}\forall_{x\in\sum^*}\ [\Delta(\{q\},x) = \{\delta'(q,x)\}] .$$

Dat er ten hoogste één zo'n δ' bestaat is duidelijk. Nu definiëren we $\delta'(q,x)$ door recurrentie naar de opbouw van x:
Stel $q \in S$, $u \in \sum^*$, $t \in \sum$:

$$\delta'(q,\Lambda) = q ,$$

$$\delta'(q,ut) = \text{het element r van S, waarvoor geldt } (\delta'(q,u),t,r) \in \delta .$$

We bewijzen nu het verband tussen δ' en Δ door recurrentie naar de opbouw van x:

$$\delta'(q,\Lambda) = q , \quad \Delta(\{q\},\Lambda) = \{q\} .$$

$$\Delta(\{q\},ut) = \{r \in S \mid \exists_{p\in\Delta(\{q\},u)}\ [(p,t,r) \in \delta]\} .$$

De recurrentieveronderstelling levert $\Delta(\{q\},u) = \{\delta'(q,u)\}$, dus

$$\Delta(\{q\},ut) = \{r \in S \mid (\delta'(q,u),t,r) \in \delta\} = \{\delta'(q,ut)\} .$$

Op grond van het bovenstaande geldt voor een deterministische eindige automaat:

$$L(A) = \{x \in \sum^* \mid \delta'(s,x) \in F\} .$$

Opmerking.
In de literatuur wordt vaak slordigerwijze bij deterministische automaten geen verschil in notatie tussen δ en δ' gemaakt.

Stelling 3.1.

Bij iedere eindige automaat $A = (S,\sum,\delta,s,F)$ bestaat een deterministische eindige automaat $A_1 = (S_1,\sum,\delta_1,s_1,F_1)$ zó, dat $L(A_1) = L(A)$.

Bewijs. We definiëren eerst A_1:

$$S_1 = \mathcal{P}(S) .$$

Stel $V \in S_1$, $t \in \sum$, $W \in S_1$:

$$(V,t,W) \in \delta_1 :\Leftrightarrow \Delta(V,t) = W .$$

$$s_1 := \{s\} , \quad F_1 := \{V \in S_1 \mid V \cap F \neq \emptyset\} .$$

Het is duidelijk, dat A_1 deterministisch is. Er is dus, zoals eerder aangegeven een afbeelding $\delta_1' : S_1 \times \sum^* \to S_1$.

We tonen nu aan, dat $\delta_1' = \Delta$. Omdat $S_1 = \mathcal{P}(S)$, zijn het beide afbeeldingen $\mathcal{P}(S) \times \sum^* \to \mathcal{P}(S)$. We passen weer recurrentie toe naar de opbouw van $x \in \sum^*$. Stel

$$V \in \mathcal{P}(S) .$$

$$\delta_1'(V,\Lambda) = V = \Delta(V,\Lambda) .$$

$$\delta_1'(V,ut) = \text{het element r van } S_1, \text{ waarvoor geldt } (\delta_1'(V,u),t,r) \in \delta_1 .$$

Nu levert de definitie van δ_1 en de recurrentieveronderstelling dat $\Delta(\Delta(V,u),t) = r$. Gebruikmakend van de concatenatieformule voor Δ vinden we

$$\delta_1'(V,ut) = \Delta(V,ut) .$$

Uit $\delta_1' = \Delta$ volgt nu

$$L(A_1) = \{x \in \sum^* \mid \Delta(s_1,x) \in F_1\} =$$

$$= \{x \in \sum^* \mid \Delta(\{s\},x) \cap F \neq \emptyset\} = L(A) . \qquad \square$$

Op grond van Stelling 3.1 hadden we ons voor het accepteren van talen tot deterministische eindige automaten kunnen beperken. We noemen twee argumenten om ook niet-deterministische automaten in de beschouwing te betrekken.

1. Bij andere typen automaten maakt het wel verschil en zijn we gedwongen om ook niet-deterministische automaten te behandelen. We wennen nu vast aan het verschijnsel van het niet-determinisme.
2. Om voor een op een andere wijze gegeven taal aan te tonen dat hij door een eindige automaat geaccepteerd kan worden, is het soms makkelijker om een niet-deterministische automaat te construeren, die dit doet. Op grond van Stelling 3.1 kan die dan desgewenst altijd nog door een deterministische worden vervangen.

Stelling 3.2.

Als de talen L_1 en L_2 (behorend bij een symboolverzameling $\sum$) door een eindige automaat kunnen worden geaccepteerd, dan geldt hetzelfde voor $L_1 \cup L_2$, L_1L_2, L_1^*, $\overline{L_1}$ $(= \sum^* \backslash L_1)$ en $L_1 \cap L_2$.

Bewijs. Kies een eindige automaat $A_1 = (S_1, \sum, \delta_1, s_1, F_1)$ met $L(A_1) = L_1$ en een eindige automaat $A_2 = (S_2, \sum, \delta_2, s_2, F_2)$ met $L(A_2) = L_2$. We mogen veronderstellen dat $S_1 \cap S_2 = \emptyset$. Als dat namelijk niet het geval is, vervangen we de elementen van S_2 door andere met dienovereenkomstige aanpassing van δ_2, s_2 en F_2. De gewijzigde automaat zal dan ook L_2 accepteren. Verder kiezen we een nieuwe toestand \$, d.w.z. $\$ \notin S_1 \cup S_2$. We definiëren nu $A_3 = (S_3, \sum, \delta_3, s_3, F_3)$ als

volgt.

$$S_3 := S_1 \cup S_2 \cup \{\$\} ,$$

$$\delta_3 := \delta_1 \cup \delta_2 \cup \{(\$,t,q) \mid (s_1,t,q) \in \delta_1\} \cup \{(\$,t,q) \mid (s_2,t,q) \in \delta_2\} ,$$

$$s_3 := \$,$$

$$F_3 := \begin{cases} F_1 \cup F_2 & \text{als } s_1 \notin F_1 \text{ en } s_2 \notin F_2 , \\ F_1 \cup F_2 \cup \{\$\} & \text{als } s_1 \in F_1 \text{ of } s_2 \in F_2 . \end{cases}$$

Stel nu $w \in L(A_3)$. Als $w = \Lambda$, dan $\$ \in F_3$, dus $s_1 \in F_1$ of $s_2 \in F_2$, dus $\Lambda \in L_1$ of $\Lambda \in L_2$, dus $w \in L_1 \cup L_2$. Als $w \neq \Lambda$, dan is er een rij overgangen, die tot acceptatie in A_3 leidt. Daarin is bij lezing van het eerste symbool van w een overgang uit $\{(\$,t,q) \mid (s_1,t,q) \in \delta_1\}$ of uit $\{(\$,t,q) \mid (s_2,t,q) \in \delta_2\}$ toegepast. In het eerste geval is een toestand in S_1 verkregen en kunnen daarna nog slechts overgangen uit δ_1 worden toegepast. De eindtoestand is dan in $F_3 \cap S_1 = F_1$, dus $w \in L_1$, omdat een overeenkomstige rij overgangen bij A_1, beginnend met s_1 kan worden toegepast. In het tweede geval vinden we op analoge wijze $w \in L_2$ en dus in ieder geval $w \in L_1 \cup L_2$. Dus $L(A_3) \subset L_1 \cup L_2$. Dat $L_1 \cup L_2 \subset L(A_3)$ is eenvoudig, dus A_3 accepteert $L_1 \cup L_2$. Nu definiëren we $A_4 = (S_4, \sum, \delta_4, s_4, F_4)$ als volgt:

$$S_4 := S_1 \cup S_2 ,$$

$$\delta_4 := \delta_1 \cup \delta_2 \cup \{(q,t,r) \mid q \in F_1 \wedge (s_2,t,r) \in \delta_2\} ,$$

$$s_4 := s_1 ,$$

$$F_4 := \begin{cases} F_2 & \text{als } s_2 \notin F_2 , \\ F_1 \cup F_2 & \text{als } s_2 \in F_2 . \end{cases}$$

Stel $w \in L(A_4)$. Dan is er een rij overgangen bij A_4, die tot acceptatie in A_4 leidt. Daarbij kan ten hoogste éénmaal een overgang uit $\{(q,t,r) \mid q \in F_1 \wedge (s_2,t,r) \in \delta_2\}$ worden toegepast, omdat $q \in S_1$ en $r \in S_2$. Stel eerst dat een dergelijke overgang niet wordt toegepast. Dan kunnen er alleen overgangen uit δ_1 worden toegepast met start in s_1 en eindtoestand in $F_4 \cap S_1$. Dus $F_4 \cap S_1 \neq \emptyset$, $F_4 \cap S_1 = F_1$, $w \in L_1$. Bovendien is dan $s_2 \in F_2$, $\Lambda \in L_2$, dus $w = w\Lambda \in L_1L_2$. Stel nu dat een overgang als boven éénmaal wordt toegepast. Op de plaats waar dat gebeurt verknippen we w tot een concatenatie xy. Bij lezing van x worden alleen overgangen uit δ_1 toegepast met start in s_1 en eindtoestand in F_1, dus $x \in L_1$. Van de overgangen bij y kan de eerste vervangen worden door één die in s_2 start en zo verkrijgen we een rij overgangen uit δ_2 met start in s_2 en eindtoestand in $F_4 \cap S_2 = F_2$, dus $y \in L_2$ en $w = xy \in L_1L_2$. Dus $L(A_4) \subset L_1L_2$. Dat $L_1L_2 \subset L(A_4)$ is eenvoudig, dus A_4 accepteert L_1L_2. Nu definiëren we $A_5 = (S_5, \Sigma, \delta_5, s_5, F_5)$ als volgt:

$$S_5 := S_1 \cup \{\$\} \quad \text{(volgens afspraak geldt } \$ \notin S_1) \text{ ,}$$

$$\delta_5 := \delta_1 \cup \{(q,t,r) \mid q \in F_1 \cup \{\$\} \wedge (s_1,t,r) \in \delta_1\} \text{ ,}$$

$$s_5 := \$ \text{ ,}$$

$$F_5 := F_1 \cup \{\$\} \text{ .}$$

Nu is $\Lambda \in L(A_5)$ en $\Lambda \in L_1^*$. Stel nu $w \in L(A_5)$, $w \neq \Lambda$. Er is dan een rij overgangen bij A_5, die tot acceptatie in A_5 leidt. Bij de eerste stap wordt in ieder geval een overgang uit $\{(q,t,r) \mid q \in F_1 \cup \{\$\} \wedge (s_1,t,r) \in \delta_1\}$ toegepast. We verknippen nu w op grond van elke toepassing van een overgang uit deze verzameling tot een concatenatie $x_1 \ldots x_n$ $(n \geq 1)$. Voor elk der x_i

verkrijgen we nu, eventueel na wijziging van de eerste overgang, een rij overgangen uit δ_1 met start in s_1 en eindtoestand in F_1. Dus $w \in L_1^*$, dus $L(A_5) \subset L_1^*$. Dat $L_1^* \subset L(A_5)$ is eenvoudig, dus A_5 accepteert L_1^*. Voor het complement is het eenvoudiger om van een deterministische automaat uit te gaan. Op grond van Stelling 3.1 bestaat er een deterministische eindige automaat $A_6 = (S_6, \sum, \delta_6, s_6, F_6)$ zó, dat $L(A_6) = L(A_1) = L_1$. We definiëren nu $A_7 = (S_7, \sum, \delta_7, s_7, F_7)$ als volgt:

$$S_7 := S_6 \ , \quad \delta_7 := \delta_6 \ , \quad s_7 := s_6 \ , \quad F_7 := S_6 \backslash F_6 \ .$$

Dan is A_7 ook deterministisch. Omdat A_6 en A_7 deterministisch zijn, behoren er afbeeldingen δ_6', resp. δ_7' bij en wel $\delta_7' = \delta_6'$. Nu geldt:

$$L(A_7) = \{x \in \sum^* \mid \delta_7'(s_7,x) \in F_7\} = \{x \in \sum^* \mid \delta_6'(s_6,x) \in S_6 \backslash F_6\} =$$

$$= \sum^* \backslash L(A_6) = \sum^* \backslash L_1 \ .$$

Dat er een eindige automaat is die $L_1 \cap L_2$ accepteert, volgt uit het voorgaande en $L_1 \cap L_2 = \overline{(\overline{L_1} \cup \overline{L_2})}$. □

Opgave.

Maak rechtstreeks uit A_1 en A_2 een eindige automaat die $L_1 \cap L_2$ accepteert (aanwijzing: gebruik $S_1 \times S_2$ voor de verzameling toestanden).

Stelling 3.3.

Alle eindige talen kunnen door een eindige automaat worden geaccepteerd.

Bewijs. We beginnen met enkele speciale gevallen.

De automaat $(\{0\},\sum,\emptyset,0,\emptyset)$ accepteert $\emptyset$. Als $a \in \sum$, dan accepteert de automaat $(\{0,1\},\sum,\{(0,a,1)\},0,\{1\})$ de taal $\{a\}$. Alle overige eindige talen kunnen hieruit door herhaalde toepassing van Stelling 3.2 worden verkregen.

Stel $L \subset \sum^*$, L eindig. We maken een gevalonderscheiding.

1. $\#L = 0$. Dan $L = \emptyset$.
2. $\#L = 1$. Dan is er een $w \in \sum^*$ zó, dat $L = \{w\}$. Als $w = \Lambda$, dan $L = \{\Lambda\} = \emptyset^*$.
   Als $w \neq \Lambda$, dan bestaan er $t_1,\ldots,t_\ell$, alle in $\sum$ zó, dat $w = t_1\ldots t_\ell$, $L = \{w\} = \{t_1\}\ldots\{t_\ell\}$.
3. $\#L = n > 1$. Dan bestaan er $L_1,\ldots,L_n$, alle in $\sum^*$ zó, dat

$$L = \bigcup_{j=1}^{n} L_j \quad \text{en} \quad \#L_j = 1 \quad \text{voor } j = 1,\ldots,n \ . \qquad \square$$

§4. Reguliere talen

Talen, die door een eindige automaat worden geaccepteerd, kunnen ook op een geheel andere manier worden verkregen. De grondgedachte, waarop dit berust is, dat men de talen beschouwt die verkregen kunnen worden uit de eindige talen door herhaaldelijke toepassing van vereniging, concatenatie en steroperatie van Kleene. Men kan daarbij nog bezuinigen op de keuze van de eindige talen als uitgangspunt en zich beperken tot de lege taal en de talen $\{t\}$ voor $t \in \sum$, omdat daaruit alle eindige talen met behulp van de zojuist genoemde operaties kunnen worden gevormd (zie het bewijs van Stelling 3.3). Het proces van vorming van talen kan worden geformaliseerd door keuze van een symbool ϕ voor de lege taal en t voor de taal $\{t\}$ en door vervolgens met

behulp van symbolen voor vereniging, concatenatie en steroperatie een expressie op te bouwen, die aangeeft welke operaties achtereenvolgens moeten worden uitgevoerd. Zulke expressies heten reguliere expressies.

Laat $\sum$ een symboolverzameling zijn. We voegen er nog wat losse symbolen aan toe, namelijk (,), $\emptyset$, $\cup$, $*$ en veronderstellen, dat deze niet in $\sum$ zitten: $\sum \cap \{(\ ,\ ), \emptyset, \cup, *\} = \emptyset$. Een reguliere expressie is een symboolrij, gevormd uit de symbolen van $\sum$ en de zojuist toegevoegde symbolen op grond van de volgende definitie.

Definitie reguliere expressie bij $\sum$.

$\emptyset$ is een reguliere expressie bij $\sum$;

als $t \in \sum$, dan is t een reguliere expressie bij $\sum$;

als α en β reguliere expressies bij $\sum$ zijn, dan zijn $(\alpha \cup \beta)$ en $(\alpha\beta)$ reguliere expressies bij $\sum$;

als α een reguliere expressie bij $\sum$ is, dan is $\alpha*$ een reguliere expressie bij $\sum$;

alleen hetgeen op grond van bovenstaande voorschriften is gevormd, is een reguliere expressie bij $\sum$.

Voorbeeld.

Stel $\sum = \{a,b,c\}$. Dan zijn $\emptyset*$, $((a \cup b)(b \cup c))*$, $(a * b)*$, $(\emptyset * \cup ((a \cup b) * b))$ reguliere expressies bij $\sum$.

Bij iedere reguliere expressie α hoort een taal $L(\alpha)$ op grond van de volgende definitie.

Definitie taal van een reguliere expressie.

$L(\phi) := \emptyset$;

als $t \in \sum$, dan $L(t) := \{t\}$;

als α en β reguliere expressies bij $\sum$ zijn, dan $L((\alpha \cup \beta)) := L(\alpha) \cup L(\beta)$ en $L((\alpha\beta)) := L(\alpha)L(\beta)$;

als α een reguliere expressie bij $\sum$ is, dan $L(\alpha*) = (L(\alpha))^*$.

Definitie reguliere taal bij $\sum$

Als $L \subset \sum^*$, dan heet L een reguliere taal bij $\sum$, als er een reguliere expressie α bij $\sum$ is zó, dat $L = L(\alpha)$.

Opmerking.

Bij verschillende reguliere expressies kan dezelfde taal horen. Zo is $L((a*b)*) = L((\phi*\cup((a \cup b)*b)))$.

Uit de Stellingen 3.2 en 3.3 volgt direct, dat iedere reguliere taal door een eindige automaat kan worden geaccepteerd (pas recurrentie toe naar de opbouw van de reguliere expressie). Het omgekeerde blijkt echter ook waar te zijn. Allereerst volgt uit de definities van reguliere expressie en reguliere taal direct, dat vereniging, concatenatie en Kleene-ster, toegepast op reguliere talen, weer reguliere talen opleveren. Verder zijn alle eindige talen regulier (bewijs analoog het bewijs van Stelling 3.3).

Stelling 4.1.

Iedere taal, die door een eindige automaat kan worden geaccepteerd, is een reguliere taal.

Bewijs. Laat L de taal zijn. We kunnen dan een deterministische eindige automaat $A = (S,\sum,\delta,s,F)$ kiezen zó, dat $L = L(A)$. Bij A hoort dan een afbeelding δ'. We nummeren de elementen van S. Stel $\#S = n$, $S = \{q_1,\ldots,q_n\}$; we kunnen aanemen, dat $q_1 = s$. We definiëren nu $n^2(n+1)$ hulptalen $L(i,j,k)$ voor $i \in \{1,\ldots,n\}$, $j \in \{1,\ldots,n\}$, $k \in \{1,\ldots,n+1\}$ als volgt:

$$L(i,j,k) := \{x \in \textstyle\sum^* \mid \delta'(q_i,x) = q_j,\ \forall_{y\in\sum^+}\forall_{z\in\sum^+}\ [x = yz \Rightarrow$$

$$\Rightarrow \exists_{h\in\{1,\ldots,n\}}\ [\delta'(q_i,y) = q_h \wedge h < k]]\} .$$

We bewijzen, dat al deze hulptalen regulier zijn en wel met volledige inductie naar k. Uit de definitie volgt direct, dat als $x \in L(i,j,1)$, dan $|x| \leq 1$, dus $L(i,j,1)$ eindig en dus regulier. Stel dat de regulariteit geldt voor zekere $k \leq n$. Uit de definitie volgt

$$L(i,j,k+1) = L(i,j,k) \cup L(i,k,k)(L(k,k,k))^*L(k,j,k) .$$

Uit deze formule en de inductieveronderstelling volgt, dat $L(i,j,k+1)$ regulier is. Verder is

$$L(i,j,n+1) = \{x \in \textstyle\sum^* \mid \delta'(q_i,x) = q_j\} .$$

Als $F = \emptyset$, dan $L = \emptyset$ en dus L regulier; als F niet-leeg is, dan

$$L = L(A) = \bigcup_{\substack{j=1 \\ q_j\in F}}^{n} L(1,j,n+1) ,$$

een vereniging van reguliere talen en dus regulier. □

We behoeven nu geen verschil meer te maken tussen reguliere talen en talen, die door een eindige automaat kunnen worden geaccepteerd; de laatste noemen we ook regulier. Voor reguliere talen is er ook een pompstelling.

Stelling 4.2 (pompstelling).

Als L een reguliere taal is met $\sum$ als verzameling symbolen, dan bestaat er een $p \in \mathbb{N}$ zó, dat voor alle $x \in \sum^*$, $y \in \sum^*$, $z \in \sum^*$ waarvoor geldt $|y| = p$ en $xyz \in L$, er $u \in \sum^*$, $v \in \sum^+$, $w \in \sum^*$ bestaan zó, dat $y = uvw$ en $\{xuv^k wz \mid k \in \mathbb{N} \cup \{0\}\} \subset L$.

Bewijs. Kies een deterministische eindige automaat $A = (S, \sum, \delta, s, F)$ zó, dat $L = L(A)$. Bij deze automaat bestaat een afbeelding δ'. Kies $p = \#S$. Stel nu $x \in \sum^*$, $y \in \sum^*$, $z \in \sum^*$, waarvoor geldt $|y| = p$ en $xyz \in L$. We kunnen y op $p+1$ manieren schrijven als een concatenatie $y = y_1 y_2$ ($y_1 \in \sum^*$, $y_2 \in \sum^*$). De toestanden $\delta'(s, xy_1)$ kunnen daarom voor deze y_1 niet allen verschillend zijn. D.w.z. er bestaan $u \in \sum^*$, $v \in \sum^+$, $w \in \sum^*$ zó, dat $y = uvw$ en $\delta'(s,xu) = \delta'(s,xuv)$. Nu geldt

$$\delta'(s,xyz) = \delta'(s,xuvwz) = \delta'(\delta'(s,xuv),wz) = \delta'(\delta'(s,xu),wz) =$$

$$= \delta'(s,xuwz) ,$$

dus $xuwz \in L$, maar ook

$$\delta'(s,xyz) = \delta'(s,xuvwz) = \delta'(\delta'(s,xu),vwz) = \delta'(\delta'(s,xuv),vwz) =$$

$$= \delta'(s,xuvvwz) ,$$

dus $xuv^2wz \in L$. Zo voortgaande vinden we ook $xuv^k wz \in L$ voor $k \in \mathbb{N}$. □

De pompstelling kan gebruikt worden om aan te tonen, dat talen niet regulier zijn. Neem $\sum = \{a,b\}$ en $L = \{a^n b^n \mid n \in \mathbb{N}\}$. Stel dat L regulier is en kies een p als in de pompstelling, $x = a^p$, $y = b^p$, $z = \Lambda$. Volgens de pompstelling is $y = uvw$, $|v| > 0$ en $\{xuv^k wz \mid k \in \mathbb{N} \cup \{0\}\} \subset L$, maar u, v en w bevatten uitsluitend het symbool b, dus $xuv^2wz = a^p b^{p+|v|} \notin L$.

We leggen nu nog een verband tussen reguliere talen en talen, die door een grammatica worden gegenereerd. Het blijkt dat de verzameling der reguliere talen precies de verzameling L_3 in de hiërarchie van Chomsky is.

Bij een grammatica van type 3 zijn alle regels van de gedaante (A,qB) of (A,q), waarin $A \in V\backslash\sum$, $B \in V\backslash\sum$, $q \in \sum^*$. Met inductie langs een afleiding is nu makkelijk aan te tonen, dat bij een grammatica van type 3 in een afleiding $S \overset{*}{\underset{G}{\Rightarrow}} w$ met $w \in \sum^*$ alle erin voorkomende symboolrijen behalve de laatste van de gedaante xA met $x \in \sum^*$ en $A \in V\backslash\sum$ zijn, waarbij $|x|$ niet-dalend is langs de afleiding, terwijl de toegepaste regel bij elke stap behalve de laatste van de gedaante (A,qB) is en in de laatste stap van de gedaante (A,q) met $A \in V\backslash\sum$, $B \in V\backslash\sum$, $q \in \sum^*$.

Stelling 4.3.

Iedere reguliere taal is van type 3.

Bewijs. Stel L een reguliere taal en $A = (S,\sum,\delta,s,F)$ een eindige automaat zó, dat $L = L(A)$. We mogen veronderstellen, dat $S \cap \sum = \emptyset$. We definiëren een grammatica $G = (V,\sum,R,S_1)$ als volgt:

$V := S \cup \sum$,

$R := \{(q,tr) \mid (q,t,r) \in \delta\} \cup \{(q,\Lambda) \mid q \in F\}$,

$S_1 := s$ (het startsymbool van G mag niet S heten, omdat de letter S al gebruikt is voor de verzameling toestanden van A) .

Het is duidelijk, dat dit een grammatica van type 3 is. Nu geldt voor $W \in P(S)$, $x \in \sum^*$:

$$\Delta(W,x) = \{r \in S \mid \exists_{q\in W} [q \overset{*}{\underset{G}{\Rightarrow}} xr]\} .$$

Voor $x = \Lambda$ klopt dit, omdat uit $q \overset{*}{\Rightarrow} r$, $q \in S$, $r \in S$ volgt, dat $q = r$. Met recurrentie verder ($u \in \sum^*$, $t \in \sum$):

$$\Delta(W,ut) = \{r \in S \mid \exists_{q\in\Delta(W,u)} [(q,t,r) \in \delta]\} =$$

$$= \{r \in S \mid \exists_{q\in S} \exists_{p\in W} [p \overset{*}{\underset{G}{\Rightarrow}} uq \wedge (q,tr) \in R]\} =$$

$$= \{r \in S \mid \exists_{p\in W} [p \overset{*}{\Rightarrow} utr]\} .$$

Nu geldt

$$L(A) = \{x \in \sum^* \mid \Delta(\{s\},x) \cap F \neq \emptyset\} =$$

$$= \{x \in \sum^* \mid \{r \in S \mid s \overset{*}{\underset{G}{\Rightarrow}} xr\} \cap F \neq \emptyset\} =$$

$$= \{x \in \sum^* \mid \exists_{r\in F} [s \overset{*}{\underset{G}{\Rightarrow}} xr]\} = \{x \in \sum^* \mid S_1 \overset{*}{\underset{G}{\Rightarrow}} x\} = L(G) . \qquad \square$$

Opmerking.

Bij de in het bewijs van Stelling 4.3 gedefinieerde grammatica voldoen de regels aan nog verdergaande beperkingen dan voor type 3 vereist is. Ze zijn van de gedaante (A,tB) of (A,Λ) met $A \in V\backslash\sum$, $B \in V\backslash\sum$, $t \in \sum$.

Dat omgekeerd iedere taal van type 3 regulier is, zullen we niet uitvoerig bewijzen. Eerst een stelling over grammatica's van type 3.

Stelling 4.4.

Bij iedere grammatica $G = (V,\sum,R,S)$ van type 3 bestaat een grammatica $G_1 = (V_1,\sum,R_1,S_1)$, waarvan alle regels de gedaante (A,tB) of (A,Λ) met $A \in V_1\backslash\sum$, $B \in V_1\backslash\sum$, $t \in \sum$ hebben, en waarvoor $L(G_1) = L(G)$.

Het bewijs van deze stelling slaan we over.

Stelling 4.5.

Iedere taal van type 3 is regulier.

We geven slechts een schets van een bewijs. Voor de taal kiezen we een grammatica $G = (V,\sum,R,S)$, waarvan we mogen aannemen, dat de regels aan de in Stelling 4.4 voor G_1 genoemde beperkingen zijn onderworpen. Vervolgens definiëren we een eindige automaat $A = (S_1,\sum,\delta,s,F)$ als volgt:

$S_1 := V\backslash\sum$ (deze verzameling mag niet S heten, omdat de letter S al gebruikt is voor het startsymbool van G) ,

$\delta := \{(B,t,C) \mid (B,tC) \in R\}$,

$F := \{B \mid (B,\Lambda) \in R\}$,

$s := S$.

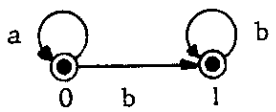
Met recurrentie bewijst men dan

$$\Delta(W,x) = \{B \in V\setminus\textstyle\sum \mid \exists_{C\in W}\ [C \overset{*}{\underset{G}{\Rightarrow}} xB]\}\ ,$$

en daaruit $L(A) = L(G)$.

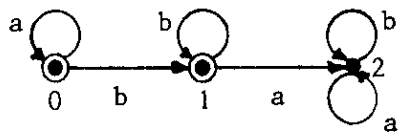
We merken nog op, dat de verschillende karakteriseringen van reguliere talen, die we gegeven hebben, het mogelijk maken in elk voorkomend geval die karakterisering te kiezen, die het makkelijkst uitkomt. Zo is het feit dat voor een reguliere taal L ook $\overline{L}$ regulier is, makkelijk aan te tonen met een eindige automaat (zie bewijs van Stelling 3.2). Dat voor een reguliere taal L ook L^R regulier is, ziet men makkelijk met behulp van een reguliere expressie. Dat een reguliere taal L contekstvrij is, is een triviaal gevolg van het feit dat L door een 3-type grammatica wordt gegenereerd.

De taal $\{a^m b^n \mid m \in \mathbb{N} \cup \{0\},\ n \in \mathbb{N} \cup \{0\}\}$ is regulier, omdat hij bij de



de reguliere expressie (a*b*) hoort. Een eindige automaat, die de taal accepteert is

$$(\{0,1\}\ ,\ \{a,b\}\ ,\ \{(0,a,0),(0,b,1),(1,b,1)\}\ ,\ 0\ ,\ \{0,1\})\ .$$



Een deterministische eindige automaat, die de taal accepteert is

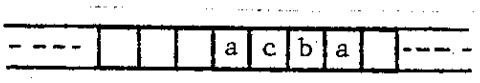
$$(\{0,1,2\}\ ,\ \{a,b\}\ ,\ \{(0,a,0),(0,b,1),(1,a,2),(1,b,1),(2,a,2),(2,b,2)\}\ ,\ 0\ ,\ \{0,1\})\ .$$

Een 3-type grammatica, die de taal genereert, is

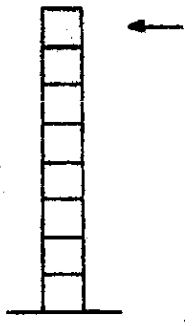
$$(\{S,B,a,b\}\ ,\ \{a,b\}\ ,\ \{(S,aS),(S,B),(B,bB),(B,\Lambda)\}\ ,\ S)\ .$$

§5. Stapelautomaten

De eindige automaat is geschikt om reguliere talen te accepteren. We zoeken nu naar een automaat, die contekstvrije talen kan accepteren. Deze automaat zal meer moeten kunnen dan een eindige automaat. Een eindige automaat beschikt over een vaste, eindige verzameling toestanden. Met "vast" is hier bedoeld, dat deze verzameling tijdens de werking van de automaat ongewijzigd blijft. Een verruiming zou zijn, dat het aantal toestanden weliswaar eindig is, maar tijdens de werking van de automaat kan toenemen. Dit helpt alleen, als aan die toeneming geen grens gesteld is: als er een bovengrens zou zijn voor het mogelijke aantal toestanden, dan had men van het begin af aan het aantal toestanden gelijk aan die bovengrens kunnen nemen en zodoende toch weer een eindige automaat verkregen. Deze onbegrensdheid kan men realiseren door aan te nemen, dat in het inwendige van de automaat een symboolrij kan worden opgeslagen. Op ieder moment vertegenwoordigt zulk een symboolrij een eindige hoeveelheid informatie. Door de symboolrij tijdens de werking van de automaat steeds langer te maken, kan de hoeveelheid opgeslagen informatie worden vergroot en hieraan is geen grens gesteld. Deze stand van zaken wordt vaak met de volgende fysische terminologie beschreven. De automaat bevat een geheugen (engels: store) bestaande uit een band (engels: tape), die verdeeld is in hokjes (engels: squares). In elk hokje is plaats voor een symbool, maar het kan ook leeg zijn. In een hokje kan een symbool geschreven, uitgewist of gewijzigd worden.



Als men een automaat wenst, die contekstvrije talen accepteert, moet men



zich wat de toegankelijkheid van de band betreft, beperkingen opleggen, die het meest aansprekend kunnen worden weergegeven door de band verticaal op te stellen en hem een stapel (engels: stack) te noemen. Het essentiële van een stapel is, dat hij alleen aan de bovenzijde toegankelijk is (in de figuur weergegeven met een pijl). Aan de bovenzijde kan iets uit de stapel worden weggenomen of iets aan de stapel worden toegevoegd; daarbij daalt resp. stijgt de hoogte van de stapel. Ook het raadplegen van de inhoud van de stapel kan alleen aan de bovenzijde geschieden. Dat betekent, dat hetgeen lager in de stapel zit, wel in de toekomst geraadpleegd kan worden, maar pas als alles wat er boven zit, is verwijderd. Als men aanneemt, dat een rekenproces met een lege stapel is begonnen, dan is hetgeen op een bepaald moment op de stapel staat er eerder ingebracht. Als men iets van de stapel afhaalt, dan is dat hetgeen men er het laatst had ingebracht. Daarom wordt het principe van de stapelvorming in het engels wel met "last in first out" aangeduid.

Bij de formele beschrijving maken we gebruik van een verzameling Γ van stapelsymbolen naast uiteraard een verzameling Σ van invoersymbolen. We maken over de doorsnede van deze verzamelingen geen enkele veronderstelling. Ze zouden disjunct kunnen zijn, maar het hoeft niet. Verder geven we de inhoud van een stapel weer met een symboolrij, die we zoals gebruikelijk horizontaal schrijven. De vraag rijst dan, of de kop van de stapel links of rechts komt; in overeenstemming met de gangbare gewoonte zullen we hem links leggen. Wijziging van de stapel wordt beschreven door het bovenste (of meest linkse)

symbool T van de stapel te vervangen door een symboolrij γ ($\gamma \in \Gamma^*$). De keuze $\gamma = \Lambda$ levert weglaten op, de keuze $\gamma = \delta T$ ($\delta \in \Gamma^+$) levert toevoegen op, de keuze $\gamma = T$ laat de stapel ongewijzigd. Let wel dat in één stap slechts één symbool kan worden weggelaten, maar meer dan één symbool kan worden toegevoegd. Evenals bij de eindige automaat kiezen we een verzameling S van toestanden met een deelverzameling F van toegelaten toestanden en een begintoestand s. Evenals bij de eindige automaat nemen we aan dat de automaat niet deterministisch is en hebben we daarom te maken met een overgangsrelatie δ, die hier echter wat ingewikkelder is. Bij de eindige automaat hing de overgang af van de geldende toestand en het gelezen invoersymbool; daar komt nu het symbool op de top van de stapel bij. Bij de eindige automaat werd een nieuwe toestand gekozen; daar komt nu de wijziging van de stapel bij. Dit proces van wijziging maakt gebruik van het bovenste symbool van de stapel en is dus alleen uitvoerbaar als de stapel niet leeg is. Dat betekent, dat we ook niet met een lege stapel kunnen beginnen. Daarom wordt ook nog een beginsymbool Z voor de stapel gekozen. Dit alles samen levert een stapelautomaat.

Definitie stapelautomaat

Een 7-tupel $A = (S, \sum, \Gamma, \delta, s, Z, F)$ heet een stapelautomaat, als geldt:

S is een eindige verzameling (elementen van S heten toestanden),

$\sum$ is een eindige niet-lege verzameling (elementen van $\sum$ heten invoersymbolen),

Γ is een eindige verzameling (elementen van Γ heten stapelsymbolen),

$\delta \subset S \times \sum \times \Gamma \times S \times \Gamma^*$, δ eindig,

$s \in S$ (begintoestand),

$Z \in \Gamma$ (beginsymbool),

$F \subset S$ (elementen van F heten toegelaten toestanden).

Opmerking.

Een stapelautomaat heet in het engels een pushdown automaton. Ter verklaring van deze naam het volgende. De beeldspraak van een stapel suggereert dat de stapel op een vaste ondergrond ligt. Dan is echter bij variabele hoogte van de stapel ook de hoogte van de top van de stapel variabel. Men kan echter ook een ander beeld kiezen en de hoogte van de top vasthouden, bijvoorbeeld om deze met een vaste leeskop te kunnen aflezen. In dat geval moet men echter, als men iets aan de stapel wil toevoegen, de hele stapel naar beneden drukken; dit verklaart de term "pushdown". Uiteraard moet ook bij verwijdering van een symbool van de stapel deze in zijn geheel omhoog gaan.

De interpretatie van δ is, dat als $(q,t,T,r,\gamma) \in \delta$, de automaat in toestand q en met T boven op de stapel bij lezing van t in toestand r overgaat en op de stapel T door γ vervangt. Bij een eindige automaat werd het inwendige van de automaat uitsluitend met een toestand beschreven; daar komt nu de stapelinhoud bij. Bij de beschrijving van de werking van de automaat vervangen we de vroeger gebruikte S nu door $S \times \Gamma^*$. Verder moeten we ook hier rekening houden met het niet-determinisme, dat ons dwingt in plaats van elementen van $S \times \Gamma^*$ deelverzamelingen van $S \times \Gamma^*$ (dat zijn elementen van $P(S \times \Gamma^*)$) te beschouwen. We zullen een element van $S \times \Gamma^*$ een inwendige configuratie noemen.

Bij een stapelautomaat $A = (S,\sum,\Gamma,\delta,s,Z,F)$ definiëren we een afbeelding $\Delta : P(S \times \Gamma^*) \times \sum^* \to P(S \times \Gamma^*)$ door recurrentie naar de opbouw van de symboolrij in $\sum^*$:

Stel $V \in P(S \times \Gamma^*)$, $u \in \sum^*$, $t \in \sum$:

$$\Delta(V,\Lambda) := V ,$$

$$\Delta(V,ut) := \{(r,\gamma\beta) \mid \beta \in \Gamma^* \wedge \exists_{q\in S} \exists_{T\in\Gamma} [(q,T\beta) \in \Delta(V,u) \wedge (q,t,T,r,\gamma) \in \delta]\} .$$

De interpretatie van $\Delta(V,w)$ is, dat het de verzameling van interne configuraties is, die uitgaande van een interne configuratie in V na lezing van w kan worden bereikt. De hierboven gegeven formule voor de overgang van $\Delta(V,u)$ naar $\Delta(V,ut)$ kan aldus worden geïnterpreteerd, dat als $(q,T\beta) \in \Delta(V,u)$ (en deze interne configuratie dus na lezing van u uitgaande van een interne configuratie in V kan worden bereikt), en als de automaat in toestand q en met T op de top van de stapel bij lezing van t in toestand r kan overgaan waarbij op de stapel T door γ wordt vervangen, dan $(r,\gamma\beta) \in \Delta(V,ut)$.

Evenals bij eindige automaten is er hier ook een concatenatieformule voor Δ: Als $V \in P(S \times \Gamma^*)$, $v \in \sum^*$, $w \in \sum^*$, dan

$$\Delta(V,vw) = \Delta(\Delta(V,v),w) .$$

We slaan het bewijs over, omdat het geheel analoog is met het bewijs bij eindige automaten.

Definitie taal geaccepteerd door stapelautomaat.

Als $A = (S,\sum,\Gamma,\delta,s,Z,F)$ een stapelautomaat met bijbehorende afbeelding Δ is, dan heet

$$L(A) := \{x \in \sum^* \mid \exists_{q\in F} \exists_{\gamma\in\Gamma^*} [(q,\gamma) \in \Delta(\{(s,Z)\},x)]\}$$

de door A geaccepteerde taal.

We merken op, dat acceptatie geschiedt als de toestand toegelaten is en dat daarbij de stapelinhoud niet ter zake doet. Verder is het net als bij de eindige automaat zo, dat de werking op een bepaald ogenblik geblokkeerd kan zijn omdat geen passend element van δ voorhanden is. Een nieuwe omstandigheid is dat dit nu ook kan optreden, omdat de stapel leeg is (d.w.z. de interne configuratie de gedaante (q,Λ) heeft).

Voorbeeld.

We kiezen $\sum = \{a,b\}$ en de taal $L = \{a^n b^n \mid n \in \mathbb{N}\}$ en proberen een stapelautomaat te maken, die L accepteert. We kunnen dit doen door als een a gelezen wordt op de stapel een a toe te voegen, zodat na lezing van a^n ook a^n op de stapel staat. Bij lezing van een b veranderen we de procedure door in een andere toestand over te gaan. Bij lezing van b schrappen we een a van de stapel. Na lezing van $a^n b^n$ zou de stapel dan leeg zijn. We moeten zorgen, dat dan een toegelaten toestand ontstaat. Dat kunnen we bewerkstelligen door in het begin te zorgen, dat onderaan de stapel een ander symbool staat, dat als we daar weer terugkeren signaleert, dat we bij het onderste symbool van de stapel zijn aangekomen. Er is ook nog een voorziening nodig om te zorgen, dat bij de start het beginsymbool goed wordt verwerkt. We geven nu de volgende definitie.

$$S := \{0,1,2\}\ ,\ \sum := \{a,b\}\ ,\ \Gamma := \{Z,a,A\}\ ,\ s := 0\ ,\ Z := Z\ ,$$

$$F := \{2\}\ ,$$

$$\delta = \{(0,a,Z,0,A),(0,a,a,0,aa),(0,b,a,1,\Lambda),(0,a,A,0,aA),$$

$$(0,b,A,2,\Lambda),(1,b,a,1,\Lambda),(1,b,A,2,\Lambda)\}\ .$$

Men bewijst nu makkelijk voor $n \in \mathbb{N}$, $k \in \mathbb{N}$ (zo nodig met volledige inductie):

$$\Delta(\{(0,Z)\},a^n) = \{(0,a^{n-1}A)\} ,$$

$$\Delta(\{(0,Z)\},a^k b^n) = \{(1,a^{k-n-1}A)\} \quad \text{als } n \leq k-1 ,$$

$$\Delta(\{(0,Z)\},a^k b^k) = \{(2,\Lambda)\} ,$$

$$\Delta(\{(0,Z)\},w) = \emptyset \quad \text{voor alle andere } w \in \textstyle\sum^+ .$$

Hieruit volgt direct dat L door deze automaat wordt geaccepteerd.

De hier gegeven definitie van stapelautomaat wijkt in één opzicht af van de gangbare. Aan de mogelijkheden, die de automaat heeft, pleegt men toe te voegen het vermogen om een stap te doen zonder een invoersymbool te lezen; dergelijke stappen worden Λ-stappen of ε-stappen genoemd. Daarbij gaat de automaat op grond van de geldende toestand en het topsymbool van de stapel over in een nieuwe toestand met aanpassing van de stapel; daarbij wordt geen invoersymbool verwerkt. Dit heeft verstrekkende gevolgen voor de werking van de automaat. Zo kan hij tussen het lezen van twee invoersymbolen één of meer van zulke Λ-stappen inschakelen. Terwijl bij een automaat zonder Λ-stappen na verwerking van een invoersymboolrij het aantal gedane stappen gelijk is aan de lengte van die symboolrij, kan dit aantal bij een automaat met Λ-stappen groter zijn en zelfs bij gegeven invoersymboolrij onbegrensd. Het blijkt nu zo te zijn dat de verzameling van talen, die door stapelautomaten met Λ-stappen kunnen worden geaccepteerd, dezelfde is als de verzameling van talen, die door stapelautomaten zonder Λ-stappen kunnen worden ge-

accepteerd. Voor het accepteren van talen is de invoering van Λ-stappen dus overbodig. Een van de redenen om toch Λ-stappen te introduceren is, dat het soms makkelijker is om voor een op een andere wijze gegeven taal een automaat met Λ-stappen te maken, die de taal accepteert, dan een automaat zonder Λ-stappen.

De definitie van een stapelautomaat met Λ-stappen wijkt slechts weinig af van die van een stapelautomaat zonder Λ-stappen. We geven daarom alleen de wijziging aan. De voorwaarde voor δ wordt vervangen door:

$$\delta \subset S \times (\sum \cup \{\Lambda\}) \times \Gamma \times S \times \Gamma^* \ , \quad \delta \text{ eindig} \ .$$

Hierbij correspondeert een element van δ van de gedaante (q,Λ,T,r,γ) met een Λ-stap. De definitie van Δ verandert nu echter wel aanmerkelijk, omdat hierin het effect van Λ-stappen moet worden verwerkt. Hiertoe introduceren we twee hulpafbeeldingen λ en μ, beide $P(S \times \Gamma^*) \to P(S \times \Gamma^*)$. Als $V \in P(S \times \Gamma^*)$, dan

$$\lambda(V) := V \cup \{(r,\gamma\beta) \mid \beta \in \Gamma^* \wedge \exists_{q \in S} \exists_{T \in \Gamma} \ [(q,T\beta) \in V \wedge (q,\Lambda,T,r,\gamma) \in \delta]\} \ .$$

De n-voudige iteratie van de afbeelding λ wordt met λ^n aangeduid.

$$\mu(V) := \bigcup_{n=1}^{\infty} \lambda^n(V) \ .$$

Het is duidelijk, dat λ het effect van ten hoogste één en μ het effect van willekeurig veel Λ-stappen representeert.

De definitie van de afbeelding Δ vertoont een grote gelijkenis met die bij een automaat zonder Λ-stappen:
Stel $V \in P(S \times \Gamma^*)$, $u \in \sum^*$, $t \in \sum$:

$$\Delta(V,\Lambda) := V ,$$

$$\Delta(V,ut) := \{(r,\gamma\beta) \mid \beta \in \Gamma^* \wedge \exists_{q\in S} \exists_{T\in\Gamma} [(q,T\beta) \in \mu(\Delta(V,u)) \wedge (q,t,T,r,\gamma) \in \delta]\} .$$

Het verschil met vroeger is, dat elke leesstap vooraf kan worden gegaan door een aantal Λ-stappen. Het feit, dat ook aan het eind nog Λ-stappen kunnen worden gedaan verwerken we in de definitie van de geaccepteerde taal:

$$L(A) := \{x \in \sum^* \mid \exists_{q\in F} \exists_{\gamma\in\Gamma^*} [(q,\gamma) \in \mu(\Delta(\{(s,Z)\},x))]\} .$$

Opmerking.
In de literatuur wordt de werking van een automaat met Λ-stappen in plaats van met een afbeelding Δ veelal beschreven met een relatie die leesstappen en Λ-stappen over één kam scheert. Dit lijkt eenvoudiger, maar de hier gegeven beschrijving weerspiegelt duidelijker hetgeen zich in de automaat afspeelt. Verder valt nog op te merken, dat een stapelautomaat zonder Λ-stappen op voor de hand liggende wijze op te vatten is als een speciaal geval van een stapelautomaat met Λ-stappen.

Een variant op het bovenstaande wordt verkregen door de acceptatie op een andere wijze te regelen. Acceptatie geschiedde tot nu toe op grond van de toestand na lezing van de invoer; de inhoud van de stapel was daarbij niet van belang. Men kan echter ook de acceptatie baseren op de inhoud van de

stapel; een veel gebruikt criterium is of de stapel leeg is. Dit heet acceptatie op grond van lege stapel (engels: "empty store" of " empty stack"). De formele definitie ziet er als volgt uit:

$$L_{es}(A) := \{x \in \sum^* \mid \exists_{q \in S} [(q,\Lambda) \in \Delta(\{(s,Z)\},x)]\} .$$

Dit geldt uiteraard voor een automaat zonder Λ-stappen. Bij een automaat met Λ-stappen moet na de Δ-functie nog μ worden toegepast, evenals bij de definitie van L(A). Let wel, dat bij $L_{es}(A)$ de verzameling F geen rol meer speelt.

We bespreken nu het verband tussen stapelautomaten en contekstvrije talen.

Stelling 5.1.

Als A een stapelautomaat zonder Λ-stappen is, dan is L(A) contekstvrij.

We zullen het vrij gecompliceerde bewijs van deze stelling niet behandelen. Uiteraard wordt uitgaande van de automaat een contekstvrije grammatica G geconstrueerd. We zullen die constructie aanduiden, maar niet bewijzen dat G de taal L(A) genereert. We moeten daarbij proberen de informatie, die in de toestand en de stapelinhoud van de automaat is opgeslagen, weer te geven met een rij hulpsymbolen van de grammatica. Bij de stapelautomaat $A = (S,\sum,\Gamma,\delta,s,Z,F)$ definiëren we een grammatica $G = (V,\sum,R,S_1)$ als volgt:

$$V := \sum \cup (S \times S \times \Gamma) \cup (S \times \Gamma) .$$

De hulpsymbolen zijn dus hetzij tripels (q,r,T) hetzij paren (q,T) met $q \in S$, $r \in S$, $T \in \Gamma$. R verkrijgen we door bij iedere $(q,t,T,r,\gamma) \in \delta$ een eindige verzameling regels te definiëren. Als $\gamma \neq \Lambda$, schrijven we $\gamma = U_1 \ldots U_m$ uit

in symbolen $U_1,\ldots,U_m$ en kiezen alle regels van de gedaante:

$$((q,p,T),t(q_1,q_2,U_1)(q_2,q_3,U_2)\ldots(q_m,q_{m+1},U_m)) \ ,$$

waarin $q_1 = r, q_{m+1} = p$ en $p,q_2,\ldots,q_m$ willekeurig in S (het aantal van deze regels is dus $(\#S)^m$); verder ook nog alle regels van de gedaante

$$((q,T),t(q_1,q_2,U_1)(q_2,q_3,U_2)\ldots(q_{k-1},q_k,U_{k-1})(q_k,U_k)) \ ,$$

waarin $q_1 = r$, $k \in \{1,\ldots,m\}$ en $q_2,\ldots,q_k$ willekeurig in S (het aantal van deze regels is $\sum_{j=0}^{m-1} (\#S)^j$); verder als $r \in F$ ook nog de regel $((q,T),t)$.
Als $\gamma = \Lambda$ kiezen we de regel $((q,r,T),t)$ en als $r \in F$ ook nog de regel $((q,T),t)$.

Als $s \in F$, wordt in R ook nog de regel $((s,Z),\Lambda)$ opgenomen.

Het startsymbool $S_1 := (s,Z)$.

We tonen niet aan, dat de aldus gedefinieerde grammatica L(A) genereert.

We merken nog op, dat de in bovenstaande constructie gegeven regels een bijzondere gedaante hebben. In het bijzonder als $\Lambda \notin L(A)$, en dus $s \notin F$, zijn alle regels van de vorm (A,tv) met $A \in V\backslash\sum$, $t \in \sum$, $v \in (V\backslash\sum)^*$. Grammatica's, waarvoor dit geldt, hebben een naam.

Definitie normaalvorm van Greibach

Een contekstvrije grammatica $G = (V,\sum,R,S)$ heeft de normaalvorm van Greibach als alle elementen van R de gedaante (A,tv) met $A \in V\backslash\sum$, $t \in \sum$, $v \in (V\backslash\sum)^*$ hebben.

Zonder bewijs vermelden we de volgende stelling.

Stelling 5.2.

Als L een contekstvrije taal is, dan bestaat er een contekstvrije grammatica G in de normaalvorm van Greibach zó, dat $L(G) = L\backslash\{\Lambda\}$.

Een tegenhanger van Stelling 5.1 is de volgende stelling.

Stelling 5.3.

Bij iedere contekstvrije taal L bestaat een stapelautomaat A zonder Λ-stappen zó, dat $L = L(A)$.

We geven een korte schets van een bewijs. Op grond van Stelling 5.2 is er een contekstvrije grammatica in de normaalvorm van Greibach, die $L\backslash\{\Lambda\}$ genereert. Het is niet moeilijk om daarbij een stapelautomaat zonder Λ-stappen te maken die $L\backslash\{\Lambda\}$ accepteert en ook niet om deze te veranderen in een stapelautomaat zonder Λ-stappen, die L accepteert.

Op grond van de Stellingen 5.1 en 5.3 is de verzameling van de contekstvrije talen dezelfde als de verzameling van de talen, die door een stapelautomaat zonder Λ-stappen worden geaccepteerd. Om aan te tonen, dat dit ook voor stapelautomaten met Λ-stappen geldt, zou men nog moeten bewijzen, dat ook de door deze automaten geaccepteerde talen contekstvrij zijn, hetgeen we niet zullen doen. Verder levert acceptatie op grond van lege stapel ook hetzelfde op: voor een stapelautomaat A met Λ-stappen is $L_{es}(A)$ contekstvrij (dit geldt dus ook voor een stapelautomaat zonder Λ-stappen) en bij iedere contekstvrije taal L bestaat een stapelautomaat A met Λ-stappen (en als $\Lambda \notin L$ zelfs één zonder Λ-stappen) zó, dat $L = L_{es}(A)$.

Tenslotte nog een opmerking over deterministische automaten. We formuleren dit voor automaten met Λ-stappen en stellen de eis, dat bij iedere interne configuratie en invoersymbool ten hoogste één berekeningsstap mogelijk is. Daarbij mag dus ook niet de keuze bestaan tussen het verrichten van een Λ-stap en een leesstap.

Definitie deterministische stapelautomaat.

Een stapelautomaat $A = (S,\sum,\Gamma,\delta,s,Z,F)$ met Λ-stappen heet deterministisch, als de volgende twee voorwaarden vervuld zijn:

$$\forall_{q\in S}\forall_{t\in\sum\cup\{\Lambda\}}\forall_{T\in\Gamma}\forall_{r_1\in S}\forall_{r_2\in S}\forall_{\gamma_1\in\Gamma^*}\forall_{\gamma_2\in\Gamma^*}$$

$$[(q,t,T,r_1,\gamma_1)\in\delta\wedge(q,t,T,r_2,\gamma_2)\in\delta\Rightarrow r_1=r_2\wedge\gamma_1=\gamma_2]\ ,$$

$$\forall_{q\in S}\forall_{T\in\Gamma}\forall_{r_1\in S}\forall_{\gamma_1\in\Gamma^*}$$

$$[(q,\Lambda,T,r_1,\gamma_1)\in\delta\Rightarrow\forall_{t\in\sum}\forall_{r_2\in S}\forall_{\gamma_2\in\Gamma^*}[(q,t,T,r_2,\gamma_2)\notin\delta]]\ .$$

De eerste voorwaarde houdt in, dat in iedere situatie ten hoogste één leesstap en ook ten hoogste één Λ-stap mogelijk is. De tweede voorwaarde houdt in, dat als in een situatie een Λ-stap mogelijk is, dan geen leesstap mogelijk is.

Definitie deterministische taal.

Een taal heet deterministisch, als er een deterministische stapelautomaat met Λ-stappen bestaat, die de taal accepteert.

Zonder bewijs vermelden we, dat er contekstvrije talen bestaan, die niet-deterministisch zijn en dat er deterministische talen bestaan, die niet geaccepteerd kunnen worden door een deterministische stapelautomaat zonder Λ-stappen. Deze laatste omstandigheid levert een nieuw argument om ook stapelautomaten met Λ-stappen in de beschouwing te betrekken.

Een voorbeeld van een bewering over contekstvrije talen, die makkelijk met stapelautomaten kan worden bewezen, is de bewering, dat de doorsnede van een contekstvrije taal en een reguliere taal contekstvrij is. Om dit aan te tonen, is het voldoende om bij een stapelautomaat A_1 en een eindige automaat A_2 een stapelautomaat A_3 te maken zó, dat $L(A_3) = L(A_1) \cap L(A_2)$. Dit blijkt niet moeilijk te zijn.

TECHNISCHE HOGESCHOOL EINDHOVEN

Onderafdeling der Wiskunde en Informatica

AUTOMATENTHEORIE EN FORMELE TALEN (2K170)

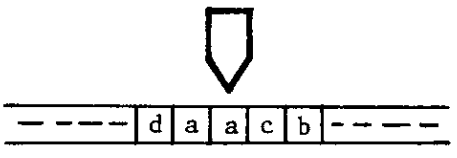
Hoofdstuk 3. Beslisbaarheid

lentetrimester 1984

HOOFDSTUK 3. Beslisbaarheid

§6. Turing-machines

Bij de invoering van de stapelautomaat hebben we een automaat voorzien van een geheugen in de vorm van een band, die echter slechts beperkt toegankelijk was. Deze beperking willen we nu opheffen. Het geheugen moet kunnen worden geraadpleegd en de inhoud moet kunnen worden gewijzigd. Beide handelingen worden telkens slechts in één aangewezen hokje (engels: "scanned square") uitgevoerd. De fysische beeldspraak geeft een leeskop die op een bepaald hokje staat gericht. Het symbool onder de leeskop kan worden gelezen, maar ook kan daar een symbool worden vervangen, uitgewist of, als het hokje leeg was, worden geschreven. Om later ook andere informatie aan de band te kunnen ontlenen moet de leeskop kunnen worden verschoven en wel zowel naar links als naar rechts. Om te zorgen dat dergelijke rechts- en linksverschuivingen altijd mogelijk zijn, denken we ons de band naar beide zijden tot in het oneindige uitgestrekt. Invoer en uitvoer worden nu echter ook anders geregeld dan in de tot nu toe behandelde automaten. In plaats van een invoer symbool voor symbool van buitenaf denken we ons nu de invoer bij het begin van de werking aanwezig op de geheugenband. Dit kan alleen als de verzameling der invoersymbolen deelverzameling is van de verzameling der bandsymbolen. Verder opent dit de mogelijkheid dat de automaat tijdens zijn werking een symbool uit de invoer meermalen raadpleegt. Evenzo zal als uitvoer worden opgevat datgene wat op de band staat als de automaat zijn werking heeft beëindigd. Verder heeft de automaat een eindige verzameling toestanden, waarvan er één als begintoestand wordt aangemerkt.



Het vervangen, schrijven en uitwissen van symbolen kan onder één hoedje worden gevangen door de introductie van een loos symbool (engels: "blank"), dat we ons in alle lege hokjes geplaatst denken. Dan zijn alle genoemde handelingen vervangingen: schrijven is vervangen van een loos symbool door een ander symbool en uitwissen is vervangen van een ander symbool door een loos symbool. Overigens zullen we "vervangen" voortaan "schrijven" noemen.

Een handeling van de automaat hangt af van de geldende toestand en het gelezen symbool en bestaat uit het overgaan in een nieuwe toestand, het schrijven in het gelezen hokje en mogelijk het één hokje naar links of naar rechts schuiven van de leeskop. Het blijkt nu geen essentiële beperking van het vermogen van de automaat te zijn en de formele beschrijving te vereenvoudigen als we aannemen dat van de handelingen schrijven en schuiven bij iedere berekeningsstap slechts één wordt uitgevoerd. Er moet nu ook nog een voorziening worden getroffen om te zorgen dat de automaat wel eens ophoudt met werken. Dit zou kunnen door in de overgangsrelatie blokkades in te bouwen. Wij doen het hier echter anders. Wij beperken ons tot deterministische automaten en kiezen een speciale toestand als stoptoestand (engels: "halting state"). Als de automaat in die toestand komt, houdt hij op; in alle andere toestanden zet hij zijn werking altijd voort.

Voor de formele beschrijving kiezen we vier vaste symbolen, die we met B, L, R en h aanduiden. Ze duiden respectievelijk het loze symbool, links, rechts en de stoptoestand aan. Deze symbolen mogen niet voor andere doeleinden worden gebruikt en zijn voor alle Turing-machines hetzelfde.

Definitie Turing-machine.

Een 4-tupel $A = (S,\Gamma,\delta,s)$ heet een Turing-machine, als geldt

S is een eindige verzameling (elementen van S heten toestanden) ,

$h \in S$,

Γ is een eindige verzameling (elementen van Γ heten bandsymbolen) ,

$B \in \Gamma$,

δ is een afbeelding $(S\backslash\{h\}) \times \Gamma \to S \times (\Gamma \cup \{L,R\})$,

$s \in S$ (begintoestand) .

Het deterministische karakter van de automaat vloeit voort uit het feit, dat voor δ een afbeelding is gekozen. Het is ook mogelijk om niet-deterministische Turing-machines te definiëren; wij zien daarvan af. We schrijven ter afkorting $S' := S\backslash\{h\}$.

In de nu volgende beschouwing nemen we aan dat een Turing-machine (S,Γ,δ,s) gegeven is. Definities hangen af van deze keuze.

De werking van de overgangsfunctie hangt af van geldende toestand en gelezen symbool, een element van $S' \times \Gamma$. Om de totale werking van de automaat te beschrijven moet ook de inhoud van de band worden beschreven, die uiteenvalt in een linkerdeel en een rechterdeel, namelijk hetgeen links, resp. rechts van het gelezen hokje staat. Op grond van de invoering van B zijn nu alle oneindig veel hokjes beschreven, maar de afspraak is nu, dat er maar eindig veel zijn, die met een ander symbool dan B zijn beschreven. Gaat men langs het linkerdeel naar links, dan komt men op den duur niets anders dan B meer tegen. Men kan dat dus weglaten en de inhoud met een eindige symbool-

rij beschrijven. In deze symboolrij kunnen B's voorkomen, omdat loze symbolen ook tussen andere symbolen kunnen staan. Men zou aan die symboolrij nog een zuinigheidseis kunnen opleggen, die voortvloeit uit het streven om aan de linkerkant zoveel mogelijk loze symbolen af te knippen. Dit leidt tot de eis dat als de symboolrij niet leeg is het meest linkse symbool $\neq B$ is. Omdat de zorg om deze eis bij het uitvoeren van berekeningsstappen vervuld te houden nogal wat formele complicaties oplevert, zullen wij hem niet stellen. Analoge beschouwingen kunnen worden gehouden voor het rechterdeel, uiteraard onder passende vervangingen van links door rechts. Wij zullen elementen van $\Gamma^* \times (S \times \Gamma) \times \Gamma^*$ interne configuraties noemen.

Bij de afbeelding δ voeren we in de componenten δ_1 en δ_2 van het resultaat van de uitvoering van δ, als volgt: $\delta_1 : S' \times \Gamma \to S$ en $\delta_2 : S' \times \Gamma \to \Gamma \cup \{L,R\}$ worden gedefinieerd door:

$$\forall_{q \in S'} \forall_{t \in \Gamma} [\delta(q,t) = (\delta_1(q,t) , \delta_2(q,t))] .$$

Als $\delta_2(q,t) \in \Gamma$ spreken we van een schrijfopdracht, als $\delta_2(q,t) \in \{L,R\}$ van een schuifopdracht.

Om schuifopdrachten formeel goed te kunnen beschrijven, maken we linkssplitsingen en rechtssplitsingen van symboolrijen. Bij een linkssplitsing splitsen we een symboolrij in het meest linkse symbool en de rest; we spreken af dat dit bij de lege symboolrij B en Λ oplevert. Analoog gaan wij bij de rechtssplitsing te werk. We definiëren afbeeldingen $r_1 : \Gamma^* \to \Gamma^*$, $r_2 : \Gamma^* \to \Gamma$, $\ell_1 : \Gamma^* \to \Gamma$ en $\ell_2 : \Gamma^* \to \Gamma^*$ als volgt:

Stel $u \in \Gamma^*$, $t \in \Gamma$:

$$r_1(\Lambda) := \Lambda \quad , \quad r_2(\Lambda) := B \ ,$$

$$r_1(ut) := u \quad , \quad r_2(ut) := t \ ,$$

$$\ell_1(\Lambda) := B \quad , \quad \ell_2(\Lambda) := \Lambda \ ,$$

$$\ell_1(ut) := \begin{cases} t & \text{als } u = \Lambda \ , \\ \ell_1(u) & \text{als } u \neq \Lambda \ , \end{cases} \qquad \ell_2(ut) := \begin{cases} \Lambda & \text{als } u = \Lambda \ , \\ \ell_2(u)t & \text{als } u \neq \Lambda \ . \end{cases}$$

Er geldt dan:

$$\forall_{w \in \Gamma^*} \left[r_1(w)\, r_2(w) = \begin{cases} B & \text{als } w = \Lambda \\ w & \text{als } w \neq \Lambda \end{cases} \right] ,$$

$$\forall_{w \in \Gamma^*} \left[\ell_1(w)\, \ell_2(w) = \begin{cases} B & \text{als } w = \Lambda \\ w & \text{als } w \neq \Lambda \end{cases} \right] .$$

We kunnen nu het effect van een berekeningsstap op de interne configuratie formeel definiëren. We doen dat met een afbeelding $\Delta : \Gamma^* \times (S' \times \Gamma) \times \Gamma^* \rightarrow$ $\rightarrow \Gamma^* \times (S \times \Gamma) \times \Gamma^*$ als volgt:

Stel $\gamma_1 \in \Gamma^*$, $q \in S'$, $t \in \Gamma$, $\gamma_2 \in \Gamma^*$,

$$\Delta(\gamma_1,(q,t),\gamma_2) := \begin{cases} (\gamma_1,(\delta_1(q,t),\delta_2(q,t)),\gamma_2) & \text{als } \delta_2(q,t) \in \Gamma \ , \\ (r_1(\gamma_1),(\delta_1(q,t),r_2(\gamma_1)),t\gamma_2) & \text{als } \delta_2(q,t) = L \ , \\ (\gamma_1 t,(\delta_1(q,t),\ell_1(\gamma_2)),\ell_2(\gamma_2)) & \text{als } \delta_2(q,t) = R \ . \end{cases}$$

De drie gevallen corresponderen met resp. schrijven, linksverschuiven en rechtsverschuiven.

Het is nu de bedoeling, dat de Turing-machine doorgaat met het doen van berekeningsstappen zolang dat mogelijk is, d.w.z. zolang de in de interne configuratie voorkomende toestand $\neq h$ is. We beschrijven dit met relaties $\vdash\!\!\!\overset{n}{}$ op de verzameling I der interne configuraties voor iedere $n \in \mathbb{N} \cup \{0\}$. Als $M \in I$, $N \in I$, dan definiëren we met inductie naar n:

$$M \vdash\!\!\!\overset{0}{} N :\Leftrightarrow M = N ,$$

$$M \vdash\!\!\!\overset{n+1}{} N :\Leftrightarrow \exists_{N' \in I} [M \vdash\!\!\!\overset{n}{} N' \wedge N = \Delta(N')] .$$

In woorden: $M \vdash\!\!\!\overset{n}{} N$ betekent, dat N uit M ontstaat in n berekeningsstappen. De volgende twee stellingen zijn dan eenvoudig met volledige inductie naar n te bewijzen.

Stelling 6.1.
Als $M \vdash\!\!\!\overset{n}{} N_1$ en $M \vdash\!\!\!\overset{n}{} N_2$, dan $N_1 = N_2$.

Stelling 6.2.
Als $M \vdash\!\!\!\overset{n}{} N$, $k \in \mathbb{N} \cup \{0\}$ en $0 \leq k \leq n$, dan bestaat er een $N' \in I$ zó, dat $M \vdash\!\!\!\overset{k}{} N'$ en $N' \vdash\!\!\!\overset{n-k}{} N$.

We definiëren nu voor $M \in I$:

$$B(M) := \{n \in \mathbb{N} \cup \{0\} \mid \exists_{N \in I} [M \vdash\!\!\!\overset{n}{} N]\} .$$

We onderscheiden nu twee gevallen voor B(M).

1° B(M) is niet naar boven begrensd. Uit Stelling 6.2 volgt dan dat $B(M) = \mathbb{N} \cup \{0\}$. We zeggen in dit geval, dat de Turing-machine niet stopt op invoerconfiguratie M.

2° B(M) is wel naar boven begrensd. Omdat $\emptyset \neq B(M) \subset \mathbb{N} \cup \{0\}$ heeft B(M) dan een grootste element n. Uit Stelling 6.2 volgt nu dat $B(M) = \{k \in \mathbb{N} \cup \{0\} \mid k \leq n\}$. Verder bestaat er één en slechts één $N \in I$ zó, dat $M \vdash^{n} N$ en de in N voorkomende toestand is h. We zeggen in dit geval, dat de Turing-machine stopt op invoerconfiguratie M en de uitvoerconfiguratie N in n stappen levert.

We leggen nu een verband tussen Turing-machines en talen. Als $\sum$ een symboolverzameling is, willen we de elementen van $\sum^*$ in een Turing-machine invoeren. Daartoe veronderstellen we dat $\{B,L,R\} \cap \sum = \emptyset$ en $\sum \subset \Gamma$. Voor iedere $u \in \sum^*$ kiezen we de invoerconfiguratie $(\Lambda,(s,B),u)$ en noemen deze de invoerconfiguratie behorende bij invoer u. Omdat de Turing-machine nu ook een uitvoer kan produceren, bekijken we eerst een afbeelding, die aan symboolrijen symboolrijen toevoegt, eventueel uit verschillende symboolverzamelingen.

Laat $\sum_0$ en $\sum_1$ verzamelingen van symbolen zijn, waarvoor geldt $\sum_0 \cap \{B,L,R\} = \emptyset$ en $\sum_1 \cap \{B,L,R\} = \emptyset$. Een afbeelding $f : \sum_0^* \to \sum_1^*$ heet <u>Turing-berekenbaar</u> (engels: "Turing-computable"), als er een Turing-machine (S,Γ,δ,s) met $\sum_0 \cup \sum_1 \subset \Gamma$ bestaat, die voor iedere $w \in \sum_0^*$ op de invoerconfiguratie $(\Lambda,(s,B),w)$ stopt met uitvoerconfiguratie $(\Lambda,(h,B),f(w))$.

Men kan dit toepassen op allerlei gevallen, bijvoorbeeld functies $\mathbb{N} \to \mathbb{N}$. Daartoe moeten natuurlijke getallen gecodeerd worden als symboolrijen, hetgeen op verschillende manieren kan geschieden. Een eenvoudige methode is om een symbool | te kiezen, dat we "streepje" noemen. Een element w van $\{|\}^*$ staat dan voor het getal $|w| \in \mathbb{N} \cup \{0\}$. Hiermee is eigenlijk niet $\mathbb{N}$, maar $\mathbb{N} \cup \{0\}$ gecodeerd; dit is te ondervangen door de definitie van Turing-berekenbaarheid uit te breiden tot afbeeldingen die op een deelverzameling van $\sum_0^*$ zijn gedefinieerd.

Ook functies van meer variabelen, afbeeldingen $\mathbb{N}^k \to \mathbb{N}$, kunnen we behandelen. Voor een rij $n_1,\ldots,n_k$ van natuurlijke getallen kiezen we ter vervanging van van de komma een <u>scheidingssymbool</u>, bijv. ⌈, dat we in de invoer tussen de coderingen van de natuurlijke getallen zetten. Zo krijgen we voor de rij 3,1,1,2 de invoerconfiguratie $(A,(s,B),|||⌈|⌈|⌈||)$.

Bij een taal $L \subset \sum^*$ hoort een karakteristieke functie $\chi_L : \sum^* \to \{0,1\}$, gedefinieerd door

$$\chi_L(w) = \begin{cases} 1 & \text{als } w \in L \\ 0 & \text{als } w \in \sum^* \backslash L \end{cases} .$$

Een taal L heet <u>Turing-beslisbaar</u> (engels: "Turing-decidable"), als χ_L Turing-berekenbaar is.

In dit geval fungeert de Turing-machine als een acceptor: hij maakt uit of een $w \in \sum^*$ wel of niet tot L behoort. Toch wordt aanvaardbaarheid op een andere wijze gedefinieerd.

Een taal $L \subset \sum^*$ heet Turing-aanvaardbaar (engels: "Turing-acceptable"), als er een Turing-machine (S,Γ,δ,s) met $\sum \subset \Gamma$ bestaat, die op de invoerconfiguratie $(\Lambda,(s,B),w)$ wel stopt als $w \in L$ en niet stopt als $w \in \sum^* \backslash L$.

Het hier gebezigde spraakgebruik is misschien niet in alle opzichten gelukkig, omdat in dit geval de Turing-machine, intuïtief gesproken, niet als acceptor dienst kan doen. Stopt men er een $w \in \sum^* \backslash L$ in, dan blijft men eeuwig op een antwoord wachten en krijgt dus eigenlijk geen uitsluitsel.

Het kan bewezen worden, dat de verzameling van Turing-aanvaardbare talen dezelfde is als de verzameling L_0 van talen van type 0 in de hiërarchie van Chomsky. Dit geeft ons aanleiding op deze hiërarchie terug te komen. De talen in L_2 kunnen met een stapelautomaat geaccepteerd worden en de talen in L_0 met een Turing-machine. De vraag is of de talen in L_1 (de contekstgevoelige talen) kunnen worden geaccepteerd met een Turing-machine waaraan zekere beperkingen zijn opgelegd. Dit blijkt inderdaad het geval te zijn. De beperking blijkt hierop neer te komen, dat tijdens de berekening niet meer van de geheugenband mag worden gebruikt dan het gedeelte waarop de invoer is geschreven. De beschikbare geheugenomvang is dus gelijk aan de lengte van de ingevoerde symboolrij. Dit kan worden verwezenlijkt door een speciaal markeersymbool \$ te kiezen, dat ter weerszijden van de invoer wordt geplaatst en te eisen dat de machine bij lezing van \$ terstond in de stoptoestand overgaat. We moeten ook nog eisen, dat de Turing-machine niet-deterministisch is.

Definitie lineair begrensde automaat.

Een niet-deterministische Turing-machine (S,Γ,δ,s) heet een lineair begrensde automaat als geldt:

$$\{0,1,\$\} \subset \Gamma\backslash\{B\} \quad , \quad \forall_{q\in S'}\,[\delta(q,\$) = (h,\$)] \ .$$

Een taal $L \subset \sum^*$ wordt geaccepteerd door een lineair begrensde automaat (S,Γ,δ,s), als $\sum \subset \Gamma\backslash\{B,\$\}$ en als voor iedere $w \in \sum^*$ de invoerconfiguratie $(\$,(s,B),w\$)$ stopt met uitvoerconfiguratie $(\$,(h,e),B^{|w|}\$)$, waarin

$$e = \begin{cases} 1 & \text{als } w \in L \ , \\ 0 & \text{als } w \notin L \ . \end{cases}$$

Men kan bewijzen, dat de verzameling L_1 dezelfde is als de verzameling der talen, die door een lineair begrensde automaat wordt geaccepteerd. Hieruit kan nu ook de inclusie $L_2 \subset L_1$ worden afgeleid. In een contekstvrije grammatica mogen regels van de gedaante (A,Λ) met $A \neq S$ voorkomen. In een grammatica van type 1, die de regel (S,Λ) bevat, is dat echter niet geoorloofd. Een contekstvrije grammatica hoeft dus geen type 1 grammatica te zijn. Toch is een contekstvrije taal wel een taal van type 1. Het is namelijk niet moeilijk om bij een stapelautomaat een lineair begrensde automaat te maken die dezelfde taal accepteert. Daaruit volgt $L_2 \subset L_1$.

§7. De universele Turing-machine. Stopprobleem

We hebben gezien, dat we Turing-machines kunnen gebruiken om afbeeldingen te realiseren of talen te accepteren. Voor elke afbeelding of taal moet echter een aparte Turing-machine worden gekozen. De vraag rijst of dit alles niet met één machine kan; een Turing-machine die het gedrag van alle Turing-machines kan nabootsen. Deze universele machine U heeft als Turing-machine vaste verzamelingen toestanden en bandsymbolen, terwijl aan de aantallen toestanden en bandsymbolen van de na te bootsen machines geen bovengrens gesteld is. Welke machine A in een bepaald geval moet worden nagebootst leggen we vast door de gegevens van A op de band van U te zetten en hier lijkt de begrenzing van het aantal bandsymbolen van U een probleem op te leveren. Dit probleem lossen we op door de symbolen van A niet als zodanig op de band te zetten maar gecodeerd, bijvoorbeeld met natuurlijke getallen. Voor de vaste, niet van de keuze van de Turing-machine afhankelijke, symbolen kiezen we vaste nummers evenals voor een vast markeersymbool en een vast scheidingssymbool.

Kies vaste en verschillende natuurlijke getallen als nummers voor B, L, R, h, \$, $\vdash$. Als $A = (S,\Gamma,\delta,s)$ een Turing-machine is, kiezen we andere natuurlijke getallen als nummers voor de elementen van $S\backslash\{h\}$ en $\Gamma\backslash\{B\}$; uiteraard krijgen verschillende elementen verschillende nummers en elementen van S andere nummers dan elementen van Γ. Dit levert een genummerde Turing-machine A'. De werking van de Turing-machine wordt vastgelegd door de afbeelding δ, die een afbeelding $S\backslash\{h\} \times \Gamma \to S \times (\Gamma \cup \{L,R\})$ is en dus door de eindige verzameling

$$\{(q,t,\delta_1(q,t),\delta_2(q,t)) \mid q \in S\backslash\{h\}, t \in \Gamma\}$$

welker elementen we quadrupels zullen noemen. De nummers van de elementen van zo'n quadrupel vormen een rij van vier natuurlijke getallen. We willen de quadrupels in een rij zetten en moeten daartoe een volgorde afspreken waarin we dat doen. Men kan dit bijvoorbeeld doen door ze lexicografisch te ordenen op grond van de voorste symbolen q,t , waarbij $S\{h\}$ en Γ geordend worden op grond van de eraan toegekende nummers. De aldus verkregen quadrupels noemen we de machinerij A" van de genummerde Turing-machine A'. Van de bijbehorende rij van natuurlijke getallen vormen we nu de codering $\rho(A")$, die geschikt is voor invoer in een Turing-machine; $\rho(A")$ is een symboolrij $\in \{|,\Gamma\}^*$, die op de band van U kan worden gezet.

Behalve over een beschrijving van A moet U ook beschikken over de invoer van een berekening op A of algemener over interne configuraties van A. Een interne configuratie M is een element van $\Gamma^* \times (S \times \Gamma) \times \Gamma^*$ en is dus op te vatten als een rij elementen van $\Gamma \cup S$. De codering van de bijbehorende rij nummers noemen we $\rho(M)$.

Aan de universele Turing-machine U stellen we nu de eis, dat als M een interne configuratie bij A is, waarvoor $\Delta(M)$ gedefinieerd is, U zijn invoerconfiguratie, die hoort bij de invoer $\$\rho(A")\$\rho(M)$ in een aantal stappen laat overgaan in de invoerconfiguratie, die hoort bij de invoer $\$\rho(A")\$\rho(\Delta(M))$. Als echter $\Delta(M)$ niet gedefinieerd is, stopt U op de invoerconfiguratie behorende bij de invoer $\$\rho(A")\$\rho(M)$ met een uitvoerconfiguratie die gelijk is aan deze invoerconfiguratie.

We bewijzen niet, dat een dergelijke universele Turing-machine gemaakt kan worden. Het is duidelijk, dat hij in staat is om de werking van een willekeurige Turing-machine na te bootsen.

De universele Turing-machine kan worden opgevat als een idealisering van een rekenautomaat zonder de invoer- en uitvoervoorzieningen. Een Turing-machine A correspondeert dan met een programma en een invoerconfiguratie M bij A met invoergegevens behorende bij dat programma. Het programma en de invoergegevens worden gecodeerd in het geheugen van de rekenautomaat gezet (bij U in de vorm $\rho(A")\rho(M)$), waarna de rekenautomaat het programma uitvoert en na voltooiing de uitvoergegevens in gecodeerde vorm in zijn geheugen heeft staan. U is een idealisering van een rekenautomaat, omdat zijn geheugenomvang onbegrensd is.

Het feit, dat een Turing-machine soms op een invoer niet stopt, is een hinderlijke omstandigheid bij het gebruik van de machine als beslisser. Men zou zich kunnen afvragen of het niet mogelijk is om uit te maken of een gegeven Turing-machine op een gegeven invoer stopt; men noemt dat het stopprobleem (engels: "halting problem"). Voor bepaalde concreet gegeven Turing-machines kan dit probleem zeer wel oplosbaar zijn. Het gaat ons echter om een algemene methode, die voor alle Turing-machines met bijbehorende invoeren werkt. Het ligt dan voor de hand om deze op dezelfde wijze te coderen als ten behoeve van de universele Turing-machine. Dit brengt ons tot het beschouwen van de volgende taal.

L_0 := {$\$\rho(A'')\$\rho(M)$ | A'' is machinerij van een genummerde Turing-machine A' en M is een interne configuratie bij A, waarop de Turing-machine A stopt}.

Het blijkt nu zo te zijn, dat L_0 niet Turing-beslisbaar is. Van deze bewering geven we een schets van een bewijs. Daartoe eerst een tweetal hulpstellingen, die niet moeilijk te bewijzen zijn.

Stelling 7.1.
Als de taal $L \subset \sum^*$ Turing-beslisbaar is, dan is $\bar{L} = \sum^* \backslash L$ ook Turing-beslisbaar.

Stelling 7.2.
Als de taal L Turing-beslisbaar is, dan is L ook Turing-aanvaardbaar.

We beschouwen naast L_0 ook nog de volgende taal.

L_1 := {$\rho(A'')$ | A'' is machinerij van een genummerde Turing-machine A' en de Turing-machine A stopt op de invoer $\rho(A'')$}.

Laat nu A een willekeurige Turing-machine zijn, A' een bijbehorende genummerde Turing-machine en A'' de bijbehorende machinerij. Uit de definitie van L_1 volgt dan direct

$\rho(A'') \in L_1 \leftrightarrow$ A stopt op invoer $\rho(A'')$.

Stel nu dat $\bar{L}_1 = \{|,\urcorner\}^* \backslash L_1$ Turing-aanvaardbaar is, dan is er een Turing-machine A_1, die $\bar{L}_1$ aanvaardt, waaruit volgt

$$\rho(A_1'') \in \bar{L}_1 \leftrightarrow A_1 \text{ stopt op } \rho(A_1'') .$$

Uit hetgeen hierboven voor willekeurige Turing-machines is afgeleid volgt echter ook

$$\rho(A_1'') \in L_1 \leftrightarrow A_1 \text{ stopt op } \rho(A_1'') .$$

Dit is een contradictie. Dus $\bar{L}_1$ is niet Turing-aanvaardbaar. Uit Stelling 7.2 volgt nu dat $\bar{L}_1$ niet Turing-beslisbaar is en dan uit Stelling 7.1 dat L_1 niet Turing-beslisbaar is.

Stel nu dat L_0 Turing-beslisbaar is, dan is er een Turing-machine A_2 die L_0 beslist. Men kan dan een Turing-machine A_3 maken, die een $w \in \{|,\urcorner\}^*$ eerst omzet in $\$w\$\rho(w)$ en deze vervolgens als invoer in A_2 stopt en daarop A_2 laat werken. Als $w \in \{|,\urcorner\}^*$, dan volgt uit de definities van L_0 en L_1 dat

$$w \in L_1 \leftrightarrow \$w\$\rho(w) \in L_0 .$$

Daaruit volgt dat A_3 de taal L_1 beslist, maar we hebben aangetoond dat L_1 niet Turing-beslisbaar is. Daaruit volgt dat L_0 niet Turing-beslisbaar is.

Men kan nu ook aantonen, dat het omgekeerde van Stelling 7.2 niet geldt. De taal L_0, die niet Turing-beslisbaar is, is namelijk wel Turing-aanvaardbaar. Men zou kunnen denken dat men daarvoor de universele Turing-machine U kan gebruiken, die inderdaad op alle elementen van L_0 stopt. We weten echter

niet wat U doet met symboolrijen, die niet de gedaante hebben van een gecodeerde machinerij en een gecodeerde invoerconfiguratie. Men kan echter een Turing-machine maken, die een invoer eerst controleert of deze de vereiste gedaante heeft. Als dat niet zo is, wordt een niet-stoppende berekening ingeschakeld; als het wel zo is, wordt U ingeschakeld. Deze Turing-machine zorgt voor de aanvaarding van L_0.

§8. These van Church. Beslisbaarheid en opsombaarheid

Met Turing-machines kunnen zeer ingewikkelde functies worden uitgerekend en talen worden beslist. Dit heeft geleid tot de opvatting, dat alle processen, die effectief kunnen worden uitgevoerd door een Turing-machine kunnen worden uitgevoerd. In het tijdvak 1930 - '40 heeft men zich intensief beziggehouden met mathematische precisering van begrippen als "effectief uitvoerbaar" of "beslisbaar". Er zijn daarvoor een aantal verschillende oplossingen bedacht; één daarvan is het gebruik van de door A.M. Turing in 1936 bedachte machine. In hetzelfde tijdvak is ook bewezen, dat al deze oplossingen gelijkwaardig zijn: wat met de ene methode kan, kan ook met de andere en omgekeerd. A. Church heeft de bewering opgesteld, dat hetgeen met een dergelijke formalisering kan worden verwezenlijkt hetzelfde is als wat intuïtief gesproken als effectief uitvoerbaar kan worden aangemerkt. Dit wordt de these van Church genoemd of als het meer specifiek op de Turing-machine betrekking heeft these van Church-Turing. Deze these kan uiteraard niet wiskundig worden bewezen, omdat een wiskundig gedefinieerd begrip wordt vergeleken met een intuïtief gegeven begrip. Hetgeen voor de juistheid pleit is het feit,

dat langs verschillende wegen ondernomen pogingen tot formalisering, die tot zeer verschillend ogende uitkomsten hebben geleid, gelijkwaardige begrippen hebben opgeleverd. Het succesvol werken met deze begrippen in de daarop volgende decennia heeft ertoe geleid, dat de overgrote meerderheid der wiskundigen de these van Church als juist aanvaardt. Talen, die Turing-beslisbaar zijn, worden daarom ook wel kortweg beslisbaar of oplosbaar genoemd, of ook wel met een term die aan een andere benaderingswijze is ontleend: recursief. Het in §7 gesignaleerde feit, dat L_0 niet Turing-beslisbaar is, wordt ook wel uitgedrukt door te zeggen dat het stopprobleem voor Turing-machines onoplosbaar is. Van een groot aantal vragen betreffende grammatica's en talen is aangetoond, dat ze niet oplosbaar zijn; wij gaan daarop hier niet in.

Tenslotte nog een opmerking over Turing-aanvaardbare talen. Deze worden ook recursief opsombaar (engels: "recursively enumerable") genoemd. De vraag is, wat het woord "opsombaar" in dit verband betekent. Intuïtief bedoelen we ermee, dat het mogelijk is op effectieve wijze een opsomming te geven van de elementen van de taal. Als de taal oneindig is, vereist dit dan wel een tot in het oneindige doorlopende lijst. Het is op het eerste gezicht niet duidelijk wat dit met Turing-aanvaardbaarheid te maken heeft. We geven een globale schets van dit verband.

Men kan een meer formeel begrip opsombaarheid verkrijgen door een Turing-machine te nemen en een keuze van een toestand q bij die Turing-machine. We laten de machine nu starten met de lege symboolrij als invoer. Elke keer dat de machine tijdens zijn berekening in toestand q komt, ontlenen we aan

de interne configuratie een symboolrij, die we in de opsomming opnemen. Als de machine stopt levert dit een eindige lijst; als de machine niet stopt een eindige of oneindige lijst. De aldus verkregen talen noemen we Turing-opsombaar. Dit kan als een redelijke formalisering van het intuïtieve begrip opsombaar worden beschouwd.

Stelling 8.1.

Een taal is Turing-aanvaardbaar, dan en slechts dan als hij Turing-opsombaar is.

We geven een ruwe schets van een bewijs. Laat A een Turing-machine zijn, die op de beschreven wijze een opsomming van de taal L verzorgt. We kunnen nu een Turing-machine maken, die een ingevoerde symboolrij w bewaart en vervolgens de werking van A gaat nabootsen en elke keer dat A in toestand q komt de verkregen symboolrij met w vergelijkt. Zijn die symboolrijen gelijk, dan stopt de machine, anders gaat zij door met de werking van A. Mocht A zelf stoppen, dan wordt een niet-stoppende machine ingeschakeld. Het is duidelijk dat deze machine L accepteert. Stel nu omgekeerd, dat een Turing-machine A een taal L accepteert. Het procédé om een Turing-machine te maken die L opsomt is vrij ingewikkeld. Allereerst moet de machine alle symboolrijen (elementen van $\sum^*$) successievelijk kunnen genereren. Tussen het genereren van de n^e en de $(n+1)^e$ symboolrij laat hij de machine A de n^e stap uitvoeren van de berekeningen met de reeds gegenereerde symboolrijen als invoer, uiteraard voor zover dit mogelijk is. Als bij dit proces één dezer berekeningen stopt gaat de machine in een extra toestand q over en wordt de

symboolrij waarop deze berekening was gestart als uitvoer voor de opsomming afgeleverd. Die berekening doet daarna niet meer mee, waarna de berekening verdergaat. De aldus in de opsomming opgenomen symboolrij is uiteraard element van L. Omgekeerd als $w \in L$, dan stopt A op invoer w; laat het aantal daarbij gebruikte berekeningsstappen k zijn. Verder heeft w een nummer m in de opsomming van alle symboolrijen van $\sum^*$. Het is duidelijk dat na verwerking van max(m,k) + 1 symboolrijen van $\sum^*$ door de geconstrueerde machine, w in de opsomming is opgenomen.