Samenvatting

Talen en Automaten

FMF

Huygens

Prijs: f 2,00

Inhoudsopgave

1 Wiskundige ouverture

1.1 Logica & verzamelingen

commutativiteit	$p \wedge q$	$\equiv$	$q \wedge p$
	$p \vee q$	$\equiv$	$q \vee p$
associativiteit	$(p \wedge q) \wedge r$	$\equiv$	$p \wedge (q \wedge r)$
	$(p \vee q) \vee r$	$\equiv$	$p \vee (q \vee r)$
neutraal element	$p \wedge \mathbf{true}$	$\equiv$	p
	$p \vee \mathbf{false}$	$\equiv$	p
nul element	$p \wedge \mathbf{false}$	$\equiv$	$\mathbf{false}$
	$p \vee \mathbf{true}$	$\equiv$	$\mathbf{true}$
distribitiviteit	$(p \wedge q) \vee r$	$\equiv$	$(p \vee r) \wedge (q \vee r)$
	$(p \vee q) \wedge r$	$\equiv$	$(p \wedge r) \vee (q \wedge r)$
De Morgan	$\neg(p \wedge q)$	$\equiv$	$\neg p \vee \neg q$
	$\neg(p \vee q)$	$\equiv$	$\neg p \wedge \neg q$
absorptie	$(p \wedge q) \vee p$	$\equiv$	p
	$(p \vee q) \wedge p$	$\equiv$	p
complementsregel	$(p \vee \neg q) \wedge q$	$\equiv$	$p \wedge q$
	$(p \wedge \neg q) \vee q$	$\equiv$	$p \vee q$

complement	$\bar{A}$	$=$	$\{x \mid x \notin A\}$
vereniging	$A \cup B$	$=$	$\{x \mid x \in A \vee x \in B\}$
doorsnede	$A \cap B$	$=$	$\{x \mid x \in A \wedge x \in B\}$
verschil	$A \setminus B$	$=$	$\{x \mid x \in A \wedge x \notin B\}$
product	$A \times B$	$=$	$\{(a,b) \mid a \in A \wedge b \in B\}$
machtsverzamelingen	$\mathcal{P}(A)$	$=$	$\{V \mid V \subseteq A\}$
	$\mathcal{P}_0(A)$	$=$	$\{V \mid V \subseteq A \wedge V \text{ eindig}\}$

1.2 Relaties & functies

$\neg\forall(x : D(x) : P(x)) \equiv \exists(x : D(x) : \neg P(x))$

$(a,b) \in R \equiv aRb$
$xR^{-1}y \equiv yRx$
$xR^0y \equiv x = y$
$xR^{k+1}y \equiv \exists(z \in A :: xR^kz \wedge zRy)$ voor $k \geq 0$
$xR^*y \equiv \exists(k \in \mathbf{N} : k \geq 0 : xR^ky)$
$xR^+y \equiv \exists(k \in \mathbf{N} : k > 0 : xR^ky)$
$[a]_R = \{x \in A \mid xRa\}$

$F_m : \Gamma^* \to \Sigma^*$

$F_m(x) = y \equiv \exists((f,\alpha,\beta) \in CONF(m) : \int CF :$

$1\ (k_0, \varepsilon, \tilde{x}) \vdash_m^* (f,\alpha,\beta) \wedge$
$output((f,\alpha,\beta)) = y)$

reflexief	$\forall(x : x \in A : xRx)$
symmetrisch	$\forall(x, y : x, y \in A : xRy \Rightarrow yRx)$
anti-symmetrisch	$\forall(x, y : x, y \in A : (xRy \wedge yRx) \Rightarrow x = y)$
transitief	$\forall(x, y, z : x, y, z \in A : (xRy \wedge yRz) \Rightarrow xRz)$

totaal	$\forall(x \in A :: \exists(y \in B : f(x) = y))$
surjectief	$\forall(y \in B :: \exists(x \in A :: f(x) = y))$
injectief	$\forall(x, y \in A :: f(x) = f(y) \Rightarrow x = y)$
bijectief	f is injectief en surjectief

1.3 aftelbaarheid

1 De verzamelingen $\mathbf{Z}$, $\mathbf{N} \times \mathbf{N}$ en $\mathbb{Q}$ zijn aftelbaar.
2 Als V een oneindige verzameling is en $F : V \to \mathbf{N}$ een totale injectie, dan is V aftelbaar.
3 Als V een oneindige verzameling is en $F : \mathbf{N} \to V$ een surjectie, dan is V aftelbaar.

Diagonalisatie argument

- Laat V een oneindige verzameling zijn. Stel nu dat V aftelbaar is. Dan is er een totale bijectie $F : \mathbf{N} \to V$.
- Stel je nu voor dat we de beelden van $f(0), f(1), f(2), ...$ onder elkaar schrijven en bekijk de diagonaal in dit schema, en definieer de rij $c = c(0)c(1)c(2)...$ door:
  $c(i) \neq d(i)$, met $d(i) = i$-de symbool van $f(i)$
- Dan is $c \in V$. Nu is f een bijectie van $\mathbf{N}$ op V, dus is er een $k \in \mathbf{N}$ met $f(k) = c$. Echter het k-de symbool van $f(k)$ is $d(k)$ en $d(k) \neq c(k)$, dus $f(k) \neq c$
- Tegenspraak. Conclusie: de aanname 'V is aftelbaar' is fout. V is wel oneindig, dus V is overaftelbaar.

1.4 Strings, alfabetten en talen

v is een prefix van $x \equiv \exists(w :: vw = x)$
w is een suffix van $x \equiv \exists(v :: vw = x)$

$$L^k = \begin{cases} k = 0 & \{\varepsilon\} \\ k + 1 & LL^k \text{ voor } k \in \mathbf{N} \end{cases}$$
$L^+ = \cup(i \in \mathbf{N} : i > 0 : L^i)$
$L^* = \cup(i \in \mathbf{N} : i \geq 0 : L^i)$

1.5 Inductie & recursie

$\forall(n : n \in \mathbf{N} : P(n))$ geldt als:
(basis) P(0)
(inductie) $\forall(n : n \in \mathbf{N} : P(n) \Rightarrow P(n + 1))$

2 Grammatica's

2.1 Type-0 grammatica / vrije grammatica

Een grammatica is een 4-tupel (N, T, P, S) waarvoor geldt:

- N en T zijn alfabetten met $N \cup T = \emptyset$
- $P \subseteq (N \cup T)^+ \times (N \cup T)^*$
- $S \in N$

Hierin zijn:

- N de nonterminale symbolen, aangeduid met hoofdletters
- T de terminale symbolen, aangeduid met kleine letters
- P de productieregels
- S het startsymbool

Laat $G = (N, T, P, S)$ een vrije grammatica zijn. We definieren de relatie $\Rightarrow_G$ op $(N \cup T)^*$ als:
$x \Rightarrow_G y \equiv \exists(\alpha, \beta, \gamma, \delta :: x = \gamma\alpha\delta \wedge y = \gamma\beta\delta \wedge (\alpha, \beta) \in P)$

Laat $G = (N, T, P, S)$ een grammatica zijn en $\lambda, \mu \in (N \cup T)^*$. Dan geldt:
$\forall(k \in \mathbf{N}; \alpha, \beta \in (N \cup T)^* : \alpha \Rightarrow_G^k \beta : \lambda\alpha\mu \Rightarrow_G^k \lambda\beta\mu)$

Laat $G = (N, T, P, S)$ een grammatica zijn. Een string $\alpha \in (N \cup T)^*$ heet een zinsvorm d.e.s.d. als $S \Rightarrow_G^* \alpha$

Laat $G = (N, T, P, S)$ een grammatica zijn. De door G gegenereerde taal $\mathcal{L}(G)$ wordt gedefinieerd door: $\mathcal{L}(G) = \{x \in T^* | S \Rightarrow_G^* x\}$

2.2 Type-1 grammatica / CSG (Context Sensitive Grammar)

Een grammatica $G = (N, T, P, S)$ wordt een type-1 grammatica genoemd d.e.s.d. als P voldoet aan $P \subseteq P0 \vee P \subseteq P1$ waarbij:

$P0 = \{(a, b) | a \in (N \cup T)^+ \cap b \in (N \cup T)^+ \cap |a| \leq |b|\}$

en

$P1 = \{S, \varepsilon\} \cup \{(a, b) | a \in (N \cup T)^+ \cap b \in (N \setminus \{S\} \cup T)^+ \cap |a| \leq |b|\}$

Het is mogelijk om het start symbool S te vervangen door een start symbool S', waarbij S' niet in het rechterlid van productieregel voorkomt. Hierdoor kan een ε gegenereerd worden in een type-1 taal.

2.3 Type-2 grammatica / CFG (Context Free Grammar)

Een type-1 grammatica $G = (N, T, P, S)$ wordt een type-2 grammatica genoemd d.e.s.d. als $(a, b) \in P$ voldoet aan $a \in N$.

Een nonterminaal symbool A in een grammatica G heet ε-genererend d.e.s.d. als $A \Rightarrow_G^* \varepsilon$.

Laat $G = (N, T, P, S)$ een vrije grammatica zijn waarin elke $(a, b) \in P$ voldoet aan $a \in N$. Dan is $\mathcal{L}(G)$ een context vrije taal.

2.4 Type-3 grammatica / reguliere grammatica

Een type-2 grammatica $G = (N, T, P, S)$ wordt een type-3 grammatica of reguliere grammatica genoemd d.e.s.d. als elke $(A, a) \in P$ voldoet aan $a \in \{\varepsilon\} \cup T \cup TN$.
Oftewel: als $P \subseteq P0 \vee P \subseteq P1$ waarbij:

$P0 = \{(A, a) | A \in N \cap a \in T \cup TN\}$

en

$P1 = \{(A, a) | (A \in S \cap a = \varepsilon) \vee (A \in N \cap a \in T \vee T(N \setminus \{S\}))\}$

3 Reguliere taal vs. DFSA

Reguliere taal	DFSA
$G = (N, T, P, S)$	$M = (K, T, t, k_0, F)$
N en T zijn alfabetten met $N \cap T = \emptyset$ $P \subseteq P0 \vee P \subseteq P1$ $P0 = \{(A, \alpha) \| A \in N \wedge \alpha \in T \cup TN\}$ $P1 = \{(A, \alpha) \| (A = S \wedge \alpha = \varepsilon) \vee (A \in N \wedge \alpha \in T \cup T(N \setminus \{S\}))\}$ $S \in N$	K en T zijn alfabetten met $K \cap T = \emptyset$ $t : K \times T \to K$ $k_0 \in K$ $F \subseteq K$
$x \Rightarrow_G y$ $L = \mathcal{L}(G)$ $\mathcal{L}(G) = \{x \in T^* \| S \Rightarrow_G x\}$	$t(k, ax) = t(t(k, a), x)$ $L = \mathcal{T}(M)$ $\mathcal{T}(M) = \{x \in T^* \| t(k_0, x) \in F\}$

Terminologie bij een DFSA:

- K zijn de toestanden. K is een eindige verzameling $\Rightarrow$ *eindige* automaat.
- T is het invoeralfabet. De elementen van T worden weergegeven met kleine letters.
- t is de overgangsfunctie. Bij elke $k \in K$ en $a \in T$ is er ten hoogste één element $k' \in K$ met $t(k, a) = k'$. Daarom wordt zo'n automaat ook wel *deterministisch* genoemd.
- k_0 is de begintoestand.
- F is de verzameling eindtoestanden / accepterende toestanden.

Een nondeterministische eindige automaat (NFSA) is een vijftupel (K, T, t, k_0, F) waarin:

- K en T zijn alfabetten met $K \cap T = \emptyset$;
- T is een totale functie, $t : K \times T \to \mathcal{P}(K)$;
- $k_0 \in K$;
- $F \subseteq K$.

Voor de taal $\mathcal{T}(M)$ van strings die door de NFSA M worden geaccepteerd, hebben we in feite twee mogelijke keuzes:

- $\mathcal{T}(M)$ bestaat uit die strings x waarvoor geldt dat elk pad, beginnend in de begintoestand en gelabeld volgens x, eindigt in een accepterende toestand. In dit geval spreken we van demonisch nondeterminisme.
- $\mathcal{T}(M)$ bestaat uit die strings x waarvoor er een pad, beginnend in de begintoestand en gelabeld volgens x, bestaat dat eindigt in een accepterend toestand. Dit noemen we angeliek nondeterminisme.

3.1 Equivalentie van reguliere grammatica's, DFSA's en NFSA's

Laat $G = (N, T, P, S)$ een reguliere grammatica zijn. Dan is er een NFSA M met $\mathcal{T}(M) = \mathcal{L}(G)$.

De functie $\hat{t} : \mathcal{P}(K) \times T \to \mathcal{P}(K)$ wordt gedefinieerd door:
$\hat{t}(V, a) = \bigcup(k \in V :: t(k, a))$

De functie $t' : \mathcal{P}(K) \times T^* \to \mathcal{P}(K)$ wordt gegeven door:
$t'(V, \varepsilon) = V$ voor elke $V \in \mathcal{P}(K)$
$t'(V, ax) = t'(\hat{t}(V, a), x)$ voor elke $V \in \mathcal{P}(K), a \in T, x \in T^*$

Laat L een taal zijn. De volgende uitspraken zijn equivalent:

- L is een reguliere taal
- L wordt geaccepteerd door een NFSA
- L wordt geaccepteerd door den DFSA.

Een NFSA $M = (K, T, t, k, F)$ kan worden omgebouwd tot een DFSA $M_d = (K_d, T, t_d, k_d, F_d)$ d.m.v.

- $K_d = \mathcal{P}(K)$
- $t_d(V, a) = \hat{t}(V, a)$ voor $V \in \mathcal{P}(K), a \in T$
- $k_d = \{k\}$
- $F_d = \{V \in \mathcal{P}(K) | V \cap F \neq \emptyset\}$

Een NFSA met ε-overgangen is een vijftupel (K, T, t, k_0, F) waarin:

- K en T zijn alfabetten met $K \cap T = \emptyset$;
- T is een functie, $t : K \times (T \cup \{\varepsilon\}) \to \mathcal{P}(K)$;
- $k_0 \in K$;
- $F \subseteq K$.

Laat $M = (K, T, t, k_0, F)$ een NFSA met ε-overgangen zijn. De relatie $\to_\varepsilon$ op K wordt gedefinieerd door: $k \to_\varepsilon k' \equiv k' \in t(k, \varepsilon)$

3.2 Minimalisatie

$xR_M y \equiv t(k_0, x) = t(k_0, y)$
$xR_L y \equiv \forall(z \in T^* :: xz \in L \equiv yz \in L)$
$xR_M y \Rightarrow xR_L y$
Er bestaat een DFSA met een aantal toestanden dat gelijk is aan de index van

Is L een reguliere taal, dan bestaat er een (op naamgeving na) unieke DFSA met totale overgangsfunctie waarvan het aantal toestanden gelijk is aan de index van R_L.

Minimalisatie in de praktijk:

1 maak een tabel met daarin de toestanden die onder vorming van een $a \in T$ bereikt kunnen worden en of die toestanden accepteren zijn:

	0	1	2	3	4	5
a	1	3	1	2	3	4
b	2	4	0	5	1	2
acc.		*			*	

2 maak een tabel waarop beide assen de nonterminalen zijn uitgezet. Vul de bovenste helft met getallen. De eerste keer vul je een nul in als de ene toestand wel en de andere niet accepterend is. Daarna ga je de open plaatsen langs om te kijken of er onder een creatie van een nonterminaal en positie kan worden berijkt, die al een getal heeft. Als je een keer de hele matrix door kunt lopen, zonder nog maar ergens iets in te vullen, ben je klaar.
Bijvoorbeeld: (stap 1) $(0,1)$: 0 is niet accepterend, 1 wel $\Rightarrow$ een nul invullen.
(stap 2) (0,3): onder vorming van een a kom je terecht in $(1,2)$, hierin is al een nul ingevuld $\Rightarrow$ 1 invullen

	0	1	2	3	4	5
0	-	0		1	0	
1		-	0	0		0
2			-	1	0	
3				-	0	1
4					-	0
5						-

3 Loop nu horizontaal door de tabel, en schrijf de niet ingevulde velden op, dit zijn de minimale klassen. $\{0,2,5\}$, $\{1,4\}$ en $\{3\}$

4 Met behulp van de tabel van punt 1 kunnen de paden gevonden worden:

	0,2,5	1,4	3
a	1,4	3	0,2,5
b	0,2,5	1,4	0,2,5
acc.		*	

4 Reguliere talen

4.1 Geslotenheidseigenschappen

De klasse van reguliere talen is gesloten onder:

1 $L_0 \cup L_1$
2 $L_0 \cap L_1$
3 $L_0 L_1$
4 $\overline{L_0}$

5 L_0^+ en L_0^* (Kleene-afsluiting)
6 L_0^r
7 de prefix.

D.w.z. als L_0 en L_1 reguliere talen zijn, dan zijn de bovengenoemde talen ook regulier.

Een bewijs hiervoor kan op twee manieren:

1 Via een reguliere taal: Laat G_0 en G_1 reguliere grammatica's zijn voor L_0 resp. L_1 met $G_i = (N_i, T, P_i, S_i)$, met $T = T_0 \cup T_1$
Constueer nu $G = (N, T, P, S)$ met bijvoorbeeld $\mathcal{L}(G) = L_0 \cup L_1$ de te genereren reguliere taal.
Begin te rekenen:
$x \in \mathcal{L}(G)$
$\equiv$ {definitie $\mathcal{L}(G)$}
$x \in T^* \wedge S \Rightarrow_G^+ x$
$\equiv$ {rekenen}
...
$\equiv$ {rekenen}
$x \in \mathcal{L}(G_0) \cup \mathcal{L}(G_1)$

2 Via een DFSA: Laat $M = (K, T, t, k_0, F)$ een DFSA zijn met $\mathcal{T}(M) = L_0$ en neem aan dat t een totale functie is.
Definieer bijvoorbeeld $\overline{M} = (K, T, t, k_0, K \setminus F)$. Dan is $\overline{M}$ een DFSA en er geldt:
$x \in \mathcal{T}(\overline{M})$
$\equiv$ {definitie taal van een DFSA}
$x \in T^* \wedge t(k_0, x) \in K \setminus F$
$\equiv$ {rekenen}
...
$\equiv$ {rekenen}
$x \in \overline{L_0}$
waarmee aangetoond is dat $\overline{L_0}$ wordt geaccepteerd door een DFSA, oftewel $\overline{L_0}$ is een reguliere taal.

4.2 Pompstelling voor reguliere talen

Laat L een reguliere taal zijn. Dan bestaat er een constante n waarvoor geldt:

$$\begin{array}{l}\exists(n : n \geq 1 : \\ \quad \forall(z : z \in L \wedge |z| \geq n : \\ \quad\quad \exists(u, v, w :: |uv| \leq n \wedge |v| \geq 1 \wedge uvw = z \wedge \\ \quad\quad\quad\quad \forall(i : i \geq 0 : uv^i w \in L))))\end{array}$$

Met de pompstelling kan worden aangetoont dat talen niet regulier zijn. Laat $L = \{a^j b^j | j \geq 0\}$ zijn. Stel dat L regulier is. De pompstelling garandeert dan het bestaan van een constante $n \geq 1$ met:

$$\begin{array}{l}\forall(z : z \in L \wedge |z| \geq n : \\ \quad \exists(u, v, w :: |uv| \leq n \wedge |v| \geq 1 \wedge uvw = z \wedge \forall(i : i \geq 0 : uv^i w \in L)))\end{array}$$

Kies nu $z = a^n b^n$. Dan is $z \in L$ en $|z| \geq n$. We mogen dus aannemen dat er u,v en w bestaan met:

$|uv| \leq n \wedge |v| \geq 1 \wedge uvw = a^n b^n \wedge \forall(i : i \geq 0 : uv^i w \in L)$.

Uit $uvw = a^n b^n \wedge |uv| \leq n$ concluderen we dat uv bestaat uit alleen maar symbolen a; uit $|v| \geq 1$ halen we vervolgens dat v bestaat uit een positief aantal symbolen a. Maar dan bevat $uw = uv^0 w$ minder symbolen a dan symbolen b, dus $uv^0 w \notin L$.

5 Reguliere expressies

5.1 Verband met Reguliere talen & DFSA's

Laat T een alfabet zijn. De verzameling reguliere expressies over T wordt als volgt gedefinieerd.

1 **0** en **1** zijn reguliere expressies;
2 als $a \in T$ dan is a een reguliere expressie;
3 als r_0 en r_1 reguliere expressies zijn, dan zijn ook
   (a) $(r_0 + r_1)$
   (b) $(r_0.r_1)$
   (c) r_0^+ en r_0^*
4 alleen uitdrukkingen die met behulp van (1), (2) en (3) geconstrueerd kunnen worden zijn reguliere expressies.

Laat T een alfabet zijn en RE de verzameling van reguliere expressies over T. De interpretatiefunctie $\phi : RE \to \mathcal{P}(T^*)$ wordt gegeven door:

- $\phi(0) = \emptyset$; $\phi(1) = \{\varepsilon\}$;
- als $a \in T$ dan is $\phi(a) = \{a\}$;
- als r_0, $r_1 \in RE$ dan
  - $\phi(r_0 + r_1) = \phi(r_0) \cup \phi(r_1)$;
  - $\phi(r_0.r_1) = \phi(r_0) \cap \phi(r_1)$;
  - $\phi(r_0^+) = \phi(r_0)^+$ en $\phi(r_0^*) = \phi(r_0)^*$

Laat L een reguliere taal zijn. Dan zijn er een reguliere grammatica, een DFSA, een een reguliere expressie te maken die L accepteren.

6 CFG's en Stapelautomaten

6.1 Afleidingsbomen en Ambiguïteit

Als in een afleiding van een *afleidingsboon* steeds de meest linker nonterminaal vervangen wordt, noemen we dat een linker afleiding.

Een contextvrije grammatica G heet ambigu of dubbelzinnig desd als er een $x \in \mathcal{L}(G)$ is waarovor twee verschillende afleidingsbomen bestaan.

Elke afleidingsboom legt eenduidig eenlinker afleiding vast. We kunnen dus ook zeggen: een grammatica G is ambigu desd als er voor enige string $X \in \mathcal{L}(G)$ twee verschillende linker afleidingen bestaan.

Een (nondeterministische) stapelautomaat (NPDA: nondeterministic push-down automaton) is een 7-tupel $M = (K, T, V, p, k_0, A_0, F)$ met:

- K, T en V alfabetten;
- $p : K \times V \times (T \cup \{\varepsilon\}) \to \mathcal{P}_0(K \times V^*)$ een totale functie;
- $k_0 \in K$ en $A_0 \in V$;
- $F \subseteq K$.

Terminologie:

- K heet de toestandenverzameling; T het invoeralfabet;
- V wordt het stapelalfabet genoemd. Dit zijn de symbolen die op de stapel gezet kunnen worden;
- p heet de stapelfunctie;
- k_0 noemen we de begintoestand, A_0 het startsymbool. Initieel bevindt de stapelautomaat zich in toestand k_0 en staat A_0 als enige op de stapel;
- F is de verzameling van accepterende toestanden;
- Een paar $(K, \phi) \in K \times V^*$ heet een configuratie van M.

Laat $M = (K, T, V, p, k_0, A_0, F)$ een stapelautomaat zijn.
De relatie $\to_M$ op $(K \times V^* \times T^*)$ wordt gedefinieerd door:
$(k, A\phi, ax) \to_M (l, \psi\phi, x) \equiv (L, \psi) \in p(k, A, a)$
waarbij $k, l \in K$; $A \in V$; $\phi, \psi \in V^*$; $a \in T \cup \{\varepsilon\}$ en $x \in T^*$.

Voor een stapelautomaat $M = (K, T, V, p, k_0, A_0, F)$ worden de talen $\mathcal{T}(M)$ en $\mathcal{N}(M)$ gedefinieerd door:
$\mathcal{T}(M) = \{x \in T^* | \exists (K \in F, \phi \in V^* :: (k_0, A_0, x) \to_M^* (k, \phi, \varepsilon))\}$
$\mathcal{N}(M) = \{x \in T^* : \exists (k \in K :: (k_0, A_0, x) \to_M^* (k, \varepsilon, \varepsilon))\}$

Laat $M = (K, T, V, p, k_0, A_0, F)$ een stapelautomaat zijn en $k, l \in K$; $x, y, z \in T^*$ en $\phi, \psi, \theta \in V^*$
Dan geldt:

- $(k, \phi, x) \to_M^* (l, \psi, y) \equiv (k, \phi, xz) \to_M^* (l, \psi, yz)$
- $(k, \phi, x) \to_M^* (l, \psi, y) \Rightarrow (k, \phi\theta, x) \to_M^* (l, \psi\theta, y)$

Laat T een alfabet zijn en $L \subseteq T^*$. Dan zijn equivalent:

1 er is een CFG G met $L = \mathcal{L}(G)$.
2 er is een NPDA M met $L = \mathcal{N}(M)$.
3 er is een NPDA M met $L = \mathcal{T}(M)$.

Een deterministische stapelautomaat (DPDA: deterministic push-down automaton) is een stapelautomaat $M = (K, T, V, p, k_0, A_0, F)$ waarvan de stapelfunctie p voldoet aan: voor elke $k \in K$; $A \in V$:

1 als $p(k, A, \varepsilon) \neq \emptyset$ dan $p(k, A, a) \neq \emptyset$ voor elke $a \in T$;
2 voor elke $a \in T \cup \{\varepsilon\}$ bestaat $p(k, A, a)$ hoogstens uit 1 element.

Elke reguliere taal is deterministisch.
Elke deterministische taal is contextvrij.

6.2 Chomsky Normaal-vorm

Een productie regel heet:

- *primair* desd als het rechter lid bestaat uit 1 nonterminaal.

- *secundair* desd als het rechter lid meer dan 1 symbool bevat waarvan minstens 1 terminaal.
- *tertiar* desd als het rechter lid meer dan 1 symbool bevat en geen terminalen.

Een grammatica $G = (N, T, P, S)$ heet in Chomsky Normaal-vorm (CNV) desd als $P \subseteq N \times (T \cup NN)$.

Laat $G = (N, T, P, S)$ een ε-vrije contextvrije grammatica zijn. Dan bestaat er een met G equivalente grammatica in Chomsky Normaal-vorm. Bewijs in 3 stappen, de derde grammatica is in CNV:

- verwijder de primaire prod. regels.:
  Vervang P door:
  $P_1 = \bigcup(A : A \in N : \{A \to \beta | A \Rightarrow^* B \wedge B \to \beta \in F(B)\})$
  met $F(A) = \{A \to \alpha \in P | A \notin N\}$
- verwijder de secundaire prod. regels.:
  Introduceer voor elke $a \in T$ een nieuw nonterminaal symbool A_a. De enige prod. regel die dit symbool heeft, is degene die de vervangende nonterminaal produceert.
  Vervang in de oorspronkelijke prod. regels alle terminale symbolen (a) door de bijbehorende nonterminale (A_a).
- verwijder de tertiare prod. regels.:
  Vervang alle productieregels waarvan het rechterlid een lengte heeft die groter is dan 2 door het eerste element gevolgd door een nieuwe nonterminaal. Deze nonterminaal heeft als enige prod. regel één die hetzelfde rechterlid heeft min de eerste nonterminaal.

7 Contextvrije talen

7.1 Geslotenheidseigenschappen

De klasse van contextvrije talen is gesloten onder vereniging, concatenatie, afsluiting en doorsnede met reguliere talen, d.w.z. laat L_0 en L_1 contextvrije talen zijn en R een reguliere taal, dan ook de volgende:

1 $L_0 \cup L_1$;
2 $L_0 L_1$;
3 L_0^+ en L_0^*;
4 $L_0 \cap R$.

Een bewijs hiervoor kan analoog aan de reguliere talen gemaakt worden met behulp van twee contextvrije grammatica's of twee NPDA's.

7.2 Pompstelling

Laat $G = (N, T, P, S)$ een grammatica in Chomsky Normaalvorm zijn. Als $z \in \mathcal{L}(G)$ en de diepte van een afleidingsboom van z is k, dan geldt $|z| \leq 2^{k-1}$.

Laat $G = (N, T, P, S)$ een grammatica in Chomsky Normaalvorm zijn voor de taal L en laat $n = \#N$. Dan geldt:

$$\begin{aligned}\forall(z \in L : |z| \geq \mathbb{2}^n : \\ \exists(u, v, w, x, y \in T^* :: z = uvwxy \wedge |vwx| \leq \mathbb{2}^n \wedge vx \neq \varepsilon \\ \wedge\forall(i : i \geq 0 : uv^iwx^iy \in L)))\end{aligned}$$

Voor de constante 2^n kan ook $m = 2^n$ gekozen worden.

8 Turingmachines en berekenbaarheid

Een Turingmachine (TM) is een 6-tupel $(K, \Sigma, T, t, k_0, F)$ waarin:

- K, Σ alfabetten met $\Lambda \in \Sigma$.
- $T \subseteq \Sigma \setminus \{\Lambda\}$.
- $t : (K \setminus F) \times \Sigma \to K \times \Sigma \times \{L, R, 0\}$ een partiële functie.
- $k_0 \in K$.
- $F \subseteq K$.

Terminologie en interpretatie:

- De elementen van K heten toestanden.
- Σ het het bandalfabet. De elementen van Σ zijn de symbolen die in de cellen van de band geschreven kunnen worden. Een bijzondere rol speelt Λ, de lege cel.
- T wordt het invoeralfabet genoemd. Hieruit bestaat de invoerstring.
- T noemen we de overgangsfunctie of ook wel het programma. Omdat T een functie is, wordt de hier gedefinieerde turingmachine deterministische genoemd.
- k_0 heet de begintoestand. Dit is de toestand waarin de machine initieel verkeert.
- De elementen van F noemen we accepterende toestanden. Als de TM in een van deze toestanden eindigt, dan stopt hij met succes.

In initiele toestand staat de kop van de TM aan het begin van de invoerstring. Op de invoerstring na is de band leeg ($\Lambda^\infty\gamma\Lambda^\infty$, met γ de invoerstring).

Laat $M = (K, \Sigma, T, t, k_0, F)$ een Turingmachine zijn. Een tripel (K, α, β) heet een *configuratie* van M indien:

- $k \in K$
- $\alpha \in \Sigma^*$
- $\beta \in \Sigma^+$

De verzameling van alle configuraties van M noteren we met $CONF(M)$, dus $CONF(M) = K \times \Sigma^* \times \Sigma^+$.

Laat $M = (K, \Sigma, T, t, k_0, F)$ een Turingmachine zijn. De relatie $\vdash_M$ op $CONF(M)$ wordt gedefinieerd als de kleinste relatie die voldoet aan:

als $t(k,X) = (l,Y,\mathbf{L})$ dan $(k,\varepsilon,X\beta) \vdash_M (l,\varepsilon,\Lambda Y\beta)$
en $(k,\alpha Z,X\beta) \vdash_M (l,\alpha,ZY\beta)$
als $t(k,X) = (l,Y,\mathbf{R})$ dan $(k,\alpha,X) \vdash_M (l,\alpha Y,\Lambda)$
en $(k,\alpha,XZ\beta) \vdash_M (l,\alpha Y,Z\beta)$
als $t(k,X) = (l,Y,\mathbf{0})$ dan $(k,\alpha,X\beta) \vdash_M (l,\alpha,Y\beta)$

Laat $M = (K,\Sigma,T,t,k_0,F)$ een Turingmachine zijn en $x \in T^*$.
Laat $\tilde{\varepsilon} = \Lambda$ en $\tilde{x} = x$ voor $x \in T^+$. We definiëren nu

1 M stopt met succes op $X \in T^*$ d.e.s.d. als:
$\exists((f,\alpha,\beta) \in CONF(M) : f \in F : (k_0,\varepsilon,\tilde{x}) \vdash_m^* (f,\alpha,\beta))$.

2 M stopt zonder succes op $x \in T^*$ d.e.s.d. als
$\exists((k,\alpha,\beta) \in CONF(M) : t(k,hd(\beta))$ ongedefinieerd
$\wedge k \notin F : (k_0,\varepsilon,\tilde{x}) \vdash_M^* (k,\alpha,\beta))$

Laat $M = (K,\Sigma,T,t,k_0,F)$ een Turingmachine zijn. De door M berekende functie $\mathcal{F}_M : T^* \to \Sigma^*$ wordt gedefinieerd door:

$$(\mathcal{F}_M(x) = y) \equiv \exists((f,\alpha,\beta) \in CONF(M) : f \in F : (k_0,\varepsilon,\tilde{x}) \vdash_M^* (f,\alpha,\beta) \wedge output(f,\alpha,\beta) = y)$$

Een (partiële functie f heet TM-berekenbaar d.e.s.d. als er een Turingmachine M bestaat met $f = \mathcal{F}_M$.
Elke effectieve berekenbare functie is TM-berekenbaar.
De effectief berekenbare functies zijn precies de λ-definieerbare functies.

Laat T een alfabet zijn. Dan zijn er slechts aftelbaar veel Turingmachines met invoeralfabet T. Want er is een injectieve functie te geven, die een Turingmachine kan coderen in een decimaal of binair getal (en dat zijn er aftelbaar veel).

Er zijn overaftelbaar beel functies op $\mathbf{N}$. Bewijs met diagonalisatie argument. (zie boven)

Aanpassingen zoals hieronder beschreven staan veranderen niets aan de berenbaarheid van een functie:

- er meer dan 1 band is,
- de Turingmachine meerdere koppen heeft,
- het opslagmedium een 2-dimensionaal veld is,
- de Turingmachine nondeterministisch is.
- de turingmachine aan een kant begrensd is

Een lineair begrensde automaat (LBA) met parameters c_0 en c_1 is een nondeterministische Turingmachine $M = (K,\Sigma,T,t,k_0,F)$, waarbij t voldoet aan:
voor elke $k,l \in K; X \in \sigma$ en $r \in 0,L,R$ geldt:

- als $(l,X,r) \in t(k,<)$ dan $X =< \wedge r = R$
- als $(l,X,r) \in t(k,>)$ dan $X => \wedge r = L$

De taal $\mathcal{T}(M)$ die hoort bij deze LBA wordt gedefinieerd door:

$$\mathcal{T}(M) = \{x \in T^* | \exists(\gamma \in \Sigma^*; (f,\alpha,\beta) \in CONF(M) : \gamma = x\Lambda^{(c_0-1)|x|+c_1} > \wedge f \in F : (k_0,<,\gamma) \vdash_M^* (f,\alpha,\beta))\}$$

Laat $M = (K,\Sigma,T,t,k_0,F)$ een LBA zijn. Dan is er een LBA M' met parameters $c_0 = 1$ en $c_1 = 0$ die voldoet aan $\mathcal{T}(M) = \mathcal{T}(M')$. M' is dan een strikt begrensde automaat.

9 Turingmachines en Talen

Laat T een alfabet zijn. Een taal $L \subseteq T^*$ heet recursief opsombaar (r.e.: recursively enumerable) d.e.s.d. als er een Turingmachine bestaat met de eigenschap dat voor elke $x \in T^*$ geldt:
$x \in L \equiv M$ stopt met succes op x

Een taal $L \subseteq T^*$ heet recursief d.e.s.d. als er een Turingmachine M bestaat met de eigenschap dat voor elke $x \in T^*$ geldt:
$x \in L \equiv M$ stopt met succes op x
$x \notin L \equiv M$ stopt zonder succes op x

Laat T een alfabet zijn en $L \subseteq T^*$. De volgende uitspraken zijn equivalent:
- L is recursief opsombaar.
- Er is een Turingmachine M met $\mathcal{T}(M) = L$.
- Er is een vrije grammatica G met $\mathcal{L}(G) = L$.

De klasse van recursieve talen is gesloten onder vereniging, doorsnede, concatenatie, afsluiting en complement, d.w.z. als L_0 en L_1 recursieve talen zijn, dan ook:
1 $L_0 \cup L_1$;
2 $L_0 \cap L_1$;
3 $L_0 L_1$;
4 L_0^*;
5 $\bar{L_0}$

recursieve talen.

De klasse van recursief opsombare talen is gesloten onder vereniging, doorsnede, concatenatie en afsluiting d.w.z. als L_0 en L_1 recursief opsombare talen zijn, dan ook:
1 $L_0 \cup L_1$;
2 $L_0 \cap L_1$;
3 $L_0 L_1$;
4 L_0^*;

recursief opsombare talen.
Laat T een alfabet zijn en $L \subseteq T^*$. De volgende uitspraken zijn equivalent:
1 Er is een LBA M met $\mathcal{T}(M) = L$.
2 Er is een CSG G met $\mathcal{L}(G) = L$.

Elke contextgevoelige taal is recursief, maar niet elke recursieve taal is contextgevoelig.

10 Onbeslisbaarheid

Voor de presentatie van een beslissingsprobleem wordt een vast opmaak gebruikt:

Probleemnaam

Parameter: ...
Gevraagd: ...

Wordt de vraag zo gesteld dat de enig mogelijke antwoorden 'ja' en 'nee' zijn, dan spreken we van een *beslissingsprobleem.* We noemen een beslissingsprobleem *beslisbaar* indien er een effectieve procedure bestaat om bij willekeurige instantie van het probleem uit te maken of dit een ja- dan wel nee-instantie is. Zo'n procedure noemen we een *(beslissings-)algoritme.* Een beslissingsalgoritme kan ook worden gezien als een Turingmachine die als invoer een codering van een instantie van P krijgt aangeboden en als uitvoer een uitspraak genereert die aangeeft of de invoer overeenkomt met een ja-instantie van wel een nee-instantie van P.

Het Haltingprobleem (HP)
Parameter: een Turingmachine $M = (K, \Sigma, T, t, k_0, F)$ en een string $w \in T^*$.
Gevraagd: stopt M op w?

Dit gaat met behulp van het feit dat er regulier opsombare talen bestaan die niet regulier zijn.

Het Lege Woord Haltingprobleem (εHP)
Parameter: een Turingmachine $M = (K, \Sigma, T, t, k_0, F)$.
Gevraagd: stopt M op ε?

Maak een TM M' die TM M simuleert met string w als een willekeurige geldige invoer. Nu stopt M' op ε als M stopt op w, maar dat is niet in eindige tijd uit te maken (**HP**).

Laten $P0$ en $P1$ twee beslissingsproblemen zijn. $P0$ heet *reduceerbaar* tot $P1$ indien er een *algoritme* R bestaat dat iedere instantie I van $P0$ afbeeldt op een instantie $R(I)$ van $P1$, waarbij wordt voldaan aan de eis:
$I \in \mathcal{Y}_{P0} \equiv R(I) \in \mathcal{Y}_{P1}$

Als $P0$ reduceerbaar is tot $P1$, dan schrijven we $P0 \rightsquigarrow P1$.
Reductiestelling:

- Als $P0 \rightsquigarrow P1$ en $P1$ is beslisbaar, dan is ook $P0$ beslisbaar.
- Als $P0 \rightsquigarrow P1$ en $P0$ is onbeslisbaar, dan is ook $P1$ onbeslisbaar.

Het equivalentieprobleem voor Turingmachines (EqTM)
Parameter: Twee Turingmachines M_0 en M_1
Gevraagd: $\mathcal{T}(M_0) = \mathcal{T}(M_1)$

Het equivalentieprobleem voor Turingmachines is onbeslisbaar.

Reduceer **(HP)** tot **(EqTM)**. M_0 accepteert T^* en M_1 accepteert T^* als M de invoerstring w accepteert, en accepteert er anders geen.

Het elementprobleem voor vrije grammatica's (MFG)
Parameter: een grammatica $G = (N, T, P, S)$ en een $x \in T^*$
Gevraagd: $x \in \mathcal{L}(G)$?

Het elementprobleem voor vrije grammatica's is onbeslisbaar.

Het niet-leegheidsprobleem voor vrije grammatica's (NEFG)
Parameter: een grammatica G
Gevraagd: $\mathcal{L}(G) \neq \emptyset$?

Het Niet-leegheidsprobleem voor vrije grammatica's is onbeslisbaar.

Post's Correspondentieprobleem (PCP)
Parameter: een alfabet T, een natuurlijk getal n en twee n-tupels $\tilde{x}$, $\tilde{y}$ over T
Gevraagd: bestaat er een niet-lege rij natuurlijke getallen $\langle i_1, ..., i_m \rangle$ met
$x_{i_1} x_{i_2} ... x_{i_m} = y_{i_1} y_{i_2} ... y_{i_m}$

Ambuguïteitsprobleem voor contextvrije grammatica's (AmCFG)
Parameter: een contextvrije grammatica G.
Gevraagd: is G ambigu?

Het ambiguïteitsprobleem voor contextvrije grammatica's is onbeslisbaar.
Bewijs via **(PCP)**. Nog een keer doorlezen.

		type 0	type 1	type 2	type 3
element	$w \in \mathcal{L}(G)$?	–	+	+	+
leegheid	$\mathcal{L}(G) = \emptyset$?	–	–	+	+
eindigheid	$\mathcal{L}(G)$ eindig?	–	–	+	+
totaliteit	$\mathcal{L}(G) = T^*$?	–	–	–	+
equivalentie	$\mathcal{L}(G) = \mathcal{L}(G')$?	–	–	–	+
lege doorsnede	$\mathcal{L}(G) \cap \mathcal{L}(G') = \emptyset$?	–	–	–	+
eindige doorsnede	$\mathcal{L}(G) \cap \mathcal{L}(G')$ eindig?	–	–	–	+

11 Complexiteitstheorie

11.1 Ordeberekeningen

Laat $M = (K, \Sigma, T, t, k_0, F)$ een Turingmachine zijn en $x \in T^*$.
Laat $\tilde{\varepsilon} = \Lambda$ en $\tilde{x} = x$ voor $x \in T^+$. Als M stopt (al dan niet met succes) op x dan wordt $\mathcal{B}(M, x)$, de lengte van de berekening van M op x, gegeven door:

$$\mathcal{B}(M,x) = \text{MIN}(n \in N : \exists((k,\alpha,\beta) \in CONF(M) : t(k, hd(\beta)) \text{ ongedef. } : (k_0, \varepsilon, \tilde{x}) \vdash_M^n (k, \alpha, \beta)) : n)$$

Laat $M = (K, \Sigma, T, t, k_0, F)$ een Turingmachine zijn.
De functie $\mathcal{C}(M) : \mathbf{N} \to \mathbf{N} \cup \{\infty\}$, de tijdcomplexiteit van M, wordt gedefinieerd door:
$\mathcal{C}(M)(n) = \text{MAX}(x \in T^n :: \mathcal{B}(M, x))$

Laat $f, g : \mathbf{N} \to \mathbf{N}$. We zeggen dat f van de orge g is d.e.s.d. als er positieve constanten c en N bestaan zo dat:
$\forall(n : n \geq N : f(n) \leq c \cdot g(n))$
We schrijven in dat geval: f is $\mathcal{O}(g)$.

11.2 N, NP en NPC

Laat M een TM zijn. M heet van polynomiale tijdcomplexiteit d.e.s.d. als er een polynoomfunctie f bestaat met de eigenschap dat de tijdcomplexiteit van M van de orde f is, ofwel: $\mathcal{C}(M)$ is $\mathcal{O}(f)$.

De klasse **P** van polynomiale talen wordt als volgt gedefinieerd. Laat L een taal zijn. L behoort tot **P** d.e.s.d. als er een DTM M bestaat met de eigenschappen:

- $L = \mathcal{T}(M)$
- M is van polynomiale tijdscomplexiteit

De klasse **NP** van nondet. polynomiale talen wordt als volgt gedefinieerd. Laat L een taal zijn. L behoort tot **NP** d.e.s.d. als er een TM M bestaat met de eigenschappen:

- $L = \mathcal{T}(M)$
- M is van polynomiale tijdscomplexiteit

De klasse **NPC** van NP-volledige talen wordt gedefinieerd door: $P \in$ **NPC** precies dan als wordt voldaan aan

- $P \in$ **NP**, en
- voor elke $P' \in$ **NP** geldt $P' \propto P$.

P $\subseteq$ **P**
Als **NPC** $\cap$ **P** $\neq \emptyset$ dan geldt **NP**$=$ **P**.

Laten $P0$ en $P1$ twee beslissingsproblemen zijn. $P0$ heet polynomiaal reduceerbaar tot $P1$ indien er een algoritme R bestaat dat iedere instantie i van $P0$ afbeeldt op een isntantie $R(I)$ van $P1$, waarbij wordt voldaan aan de eisen:

1 $I \in \mathcal{Y}_{P0} \equiv R(I) \in \mathcal{Y}_{P1}$
2 R is van polynomiale tijdcomplexiteit.

Als $P0$ polynomiaal reduceerbaar is tot $P1$, dan schrijven we $P0 \propto P1$

Als $P0 \propto P1$ en $P1 \in$ **P**, dan is ook $P0 \in$ **P**
Als $P_0 \in$ **NPC** $\wedge$ $P_1 \in$ **NP** $\wedge$ $P_0 \propto P_1$ dan $P_1 \in$ **NPC**.

Traveling Salesman Problem (TSP)
Parameter: een verzameling $C = \{c_1, ..., c_n\}$,
een afbeelding $\mathbf{d} : C \times C \to \mathbf{N}_+$,
een grenswaarde $B \in \mathbf{N}$
Gevraagd: is er een permutatie π van $\{1, ..., n\}$ zo, dat
$\Sigma(i : 1 \leq i < n : d(c_{\pi(i)}, c_{\pi(i+1)})) + d(c_{\pi(n)}, c_{\pi(1)}) \leq B$?

Satisfiability (SAT)
Parameter: Een eindige verzameling proportionele variabelen U
en een verzameling C van clausulen over U
Gevraagd: Is er een valuatie op U die C vervult?

Drie dimensionale matching (3DM)
Parameter: drie disjuncte verzamelingen W, X en Y met $|W| = |X| = |Y|$
en een verzameling $M \subseteq W \times X \times Y$
Gevraagd: bevat M een matching?