DEMOEN

1 Talen en Automaten

1.1 Wat is een Taal?

Definitie 1.1

String over een alfabet Σ
*Een **string** over een alfabet Σ is een opeenvolging van nul, één of meer elementen van Σ*

Definitie 1.2

Taal L over een alfabet Σ
*Een **taal** L over een alfabet Σ is een verzameling van eindige strings over Σ*

Zelf Doen 1.1 (*Kan fout zijn*)

1. *Kan je een beschrijving van de even getallen gebruiken om alle even getallen te genereren?*
   ***Antwoord:** Ja, bijvoorbeeld als de beschrijving $2n$ is kan, voor elke $n \in \mathbb{N}$ een uniek even getal gegeneerd worden.*

2. *Kan je de string in de taal testen?*
   ***Antwoord:** Ja, bijvoorbeeld elk getal modulo 2 doen en kijken of het 1 of 0 is.*

1.2 Een algebra van talen

De verzameling van alle talen over een alfabet Σ wordt een algebra als we er de volgende operaties op definiëren: Als L_1 en L_2 twee talen zijn dan:

- De unie van twee talen is een taal: $L_1 \cup L_2$
- De doorsnede van twee talen is een taal: $L_1 \cap L_2$
- het complement van een taal is een taal: $\overline{L_1}$

2 Talen en berekenbaarheid

Definitie 2.1

*Een **Turingmachine** is een 7-tal $(Q, \Sigma, \Gamma, \delta, q_s, q_a, q_r)$, waarbij Q, Σ, Γ eindige verzamelingen zijn en:*

- *Q is een verzameling toestanden;*
- *Σ is een input alfabet dat $\#$ niet bevat;*
- *Γ is het tape alfabet: $\# \in \Gamma$ en $\Sigma \subset \Gamma$*
- *q_s is de starttoestand;*
- *q_a is de accepterende eindtoestand;*
- *q_r is de verwerpende eindtoestand verschillend van q_a*
- *δ is de transitiefunctie: $Q \times \Gamma \to Q \times \Gamma \times \{L, R, S\}$*

Definitie 2.2

Een Turingmachine TM herkent L_{TM}

Definitie 2.3

Een taal L is Turing-herkenbaar indien er een Turingmachine TM bestaat zodanig dat $L = L_{TM}$

Voorbeeld 2.1

Stel $\Sigma = \{a, b\}$ en $L = \{a\}$

Q	Γ	Q	Γ	LRS
q_s	a	q_s	$\#$	R
q_s	b	x	b	S
q_s	$\#$	q_a	-	-
x	a	x	a	S
x	b	x	b	S
x	$\#$	x	$\#$	S

Is een voorbeeld van δ van een TM die $\{a\}^*$ herkent.
Hierin is duidelijk dat ∞_{TM} niet leeg is (∞_{TM} is de verzameling strings waarvoor TM niet stopt).

Voorbeeld 2.2

Herneem vorig alfabet en taal.

Q	Γ	Q	Γ	LRS
q_s	a	q_s	#	R
q_s	b	q_r	-	-
q_s	#	q_a	-	-

Is een voorbeeld van een TM die L beslist.

Hieruit kunnen we volgende definitie concluderen:

Definitie 2.4

Een Turingmachine TM beslist een taal L als TM L herkent en dat daarbovenop er geen mogelijkheid tot oneindige lussen bestaat. M.a.w.: $\infty_{TM} = \emptyset$

Hieruit volgt meteen volgende definitie:

Definitie 2.5

Een taal L heet Turing-beslisbaar als er een Turingmachine is die L beslist.

En ook:

Definitie 2.6

Een taal L is ***co-herkenbaar/co-beslisbaar*** *als* $\overline{L}$ *herkenbaar/beslisbaar is*

Stelling 2.1

Als L beslisbaar is, dan is L ook co-beslisbaar.

Bewijs 2.1

Als je in de Turingmachine, die L beslist, q_a *en* q_r *van rol verwisselt, krijg je een nieuwe Turingmachine die* $\overline{L}$ *beslist en is dus voor elke TM die L beslist, een TM te construeren die* $\overline{L}$ *beslist.*

□

Stelling 2.2

Als L herkenbaar is en co-herkenbaar, dan is L beslisbaar.

Bewijs 2.2

Laat M_1 de machine zijn die L herkent en M_2 die $\overline{L}$ herkent.
We laten M_1 en M_2 parallel lopen in een nieuwe machine M.

- *M_1 accepteert $\Rightarrow$ M accepteert;*
- *M_2 accepteert $\Rightarrow$ M verwerpt;*

. Voor elke string zal ofwel M_1 ofwel M_2 stoppen en ze kunnen niet samen accepteren. Hieruit besluiten we dat M de taal L beslist (want $\infty_{TM} = \emptyset$).

□

Opmerking: *Formeler kan M gedefinieerd worden als een 2-tape machine.*
$M_1 = (Q_1, \Sigma, \Gamma_1, \delta_1, q_s, q_{a1}, q_{r1})$
$M_2 = (Q_2, \Sigma, \Gamma_2, \delta_2, q_s, q_{a2}, q_{r2})$

We construeren M zodanig dat:
$M = (Q, \Sigma, \Gamma, \delta, q_s, q_a, q_r)$

Waarbij:
$Q = Q_1 \times Q_2$
$\Gamma = \Gamma_1 \cup \Gamma_2$
$\delta : Q \times \Gamma^2 \to Q \times \Gamma^2 \times \{L, R, S\}^2$
$q_s = (q_{s1}, q_{s2})$
$q_a = (q_{a1}, q_{r2})$
$q_r = (q_{r1}, q_{a2})$

Stel:
$w_i \in \{L, R, S\}$
$\delta_1(q_{i1}, a_1) = (q_{j1}, b_1, w_1)$
$\delta_2(q_{i2}, a_2) = (q_{j2}, b_2, w_2)$

dan:
$\delta((q_{i1}, q_{i2}), a_1, a_2) = ((q_{j1}, q_{j2}), b_1, b_2, w_1, w_2)$

□

SIPSER

3 Bewijzen

3.1 Reguliere Talen

3.1.1 De unie van twee reguliere talen is opnieuw een reguliere taal

Bewijs idee 3.1

We hebben reguliere talen: A_1 en A_2 en willen aantonen dat $A_1 \cup A_2$ ook regulier is. Omdat A_1 en A_2 regulier zijn weten we dat er een eindige automaat M_1 bestaat die A_1 aanvaardt en een tweede eindige automaat M_2 die A_2 aanvaardt. Om te bewijzen dat $A_1 \cup A_2$ regulier is, geven we een eindige automaat M die $A_1 \cup A_2$ aanvaardt.
*Het bewijs is per constructie. We construeren M vanuit M_1 en M_2. M moet een string accepteren wanneer M_1 **of** M_2 dit zou doen. Dit doen we door M_1 en M_2 te simuleren en de string te aanvaarden wanneer één van de simulaties de string aanvaardt.*
Hoe kunnen we automaat M de automaten M_1 en M_2 laten simuleren? Hier moet er opgelet worden. Vanaf de symbolen van de string gelezen zijn kunnen we de "tape"(met de symbolen op) niet terugdraaien. We moeten dus M_1 en M_2 tegelijkertijd simuleren. Er moet dus een paar van toestanden onthouden worden. Als M_1 k_1 toestanden heeft en M_2 k_2 toestanden, dan is het aantal toestanden in M: $k_1 \cdot k_2$. De transities in M gaan dus van paar tot paar.

Bewijs 3.1

Stel
M_1 aanvaardt A_1 en $M_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$, en
M_2 aanvaardt A_2 en $M_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$.

Construeer M zodat $A_1 \cup A_2$ aanvaard worden met $M = (Q, \Sigma, \delta, q_0, F)$.

1. $Q = \{(r_1, r_2) | r_1 \in Q_1 \wedge r_2 \in Q_2\}$.
   *Deze verzameling is het **Cartesisch Product** van de verzamelingen Q_1 en Q_2, geschreven als $Q_1 \times Q_2$. Het is de verzameling van alle paren van toestanden.*
2. Σ: *het alfabet is het zelfde als in M_1 en M_2.*
3. δ: *de transitiefunctie is op volgende manier gedefinieerd: $\forall (r_1, r_2) \in Q$ en $\forall a \in \Sigma$:*

$$\delta((r_1, r_2), a) = (\delta_1(r_1, a), \delta_2(r_2, a))$$

δ heeft als invoer een toestand uit M en geeft de volgende toestand uit M.

4. *q_0 is het paar (q_1, q_2)*
5. *F is de verzameling van paren van eindtoestanden uit M_1 **of** M_2.*

$$F\{(r_1, r_2)|r_1 \in F_1 \vee r_2 \in F_2\}$$

3.1.2 De concatenatie van twee reguliere talen is opnieuw een reguliere taal

Bewijs 3.2

Stel
$N_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$ de taal A_1 accepteren, en $N_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$ de taal A_2.

Construeer $N = (Q, \Sigma, \delta, q_1, F_2)$ die de taal $A_1 \circ A_2$ accepteert.

1. *$Q = Q_1 \cup Q_2$. De toestanden van N zijn alle toestanden van N_1 en N_2.*
2. *De toestand q_1 is dezelfde als de start toestand van N_1.*
3. *De eindtoestanden van F_2 zijn dezelfde als die van N_2.*
4. *Definieer δ zodat voor elke $q \in Q$ en een $a \in \Sigma$,*

$$\delta(q, a) = \begin{cases} \delta_1(q, a) & q \in Q_1 \text{ en } q \notin F_1 \\ \delta_1(q, a) & q \in F_1 \text{ en } a \neq \epsilon \\ \delta_1(q, a) \cup \{q_2\} & q \in F_1 \text{ en } a = \epsilon \\ \delta_1(q, a) & q \in Q_2 \end{cases}$$

3.1.3 De ster-operatie toegepast op een reguliere taal is opnieuw een reguliere taal

Bewijs 3.3

Stel $N_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$ de taal A_1 accepteren.

Construeer $N = (Q, \Sigma, \delta, q_0, F)$ die de taal A_1^* accepteert.

1. *$Q = \{q_0\} \cup Q_1$. De toestanden van N zijn de toestanden van N_1 plus een nieuwe starttoestand.*
2. *De toestand q_0 is de nieuwe start toestand.*
3. *$F = \{q_0\} \cup F_1$. De eindtoestanden zijn de oude plus de nieuwe starttoestand.*

3.1.4 Elke niet-deterministische eindige automaat heeft een equivalente deterministische eindige automaat.

Bewijs idee 3.2

Als de taal aanvaard wordt door een NFA, moeten we het bestaan van een DFA aantonen, die deze taal ook aanvaardt. Het idee is dus om een NFA om te zetten naar een equivalente DFA die de NFA simuleert.
Als k het aantal toestanden is in een NFA, dan heeft het 2^k deelverzamelingen van toestanden. Elke deelverzameling komt overeen met een mogelijke toestand die de DFA moet onthouden. De DFA zal dus 2^k toestanden hebben. Het enige dat rest, is het vinden van een start -en eindtoestand van de DFA en zijn transitie-functie.

Bewijs 3.4

Stel $N = (Q, \Sigma, \delta, q_0, F)$ de NFA die een taal A aanvaardt. We construeren een DFA $M = (Q', \Sigma, \delta', q_0', F')$ die ook de taal A aanvaardt. Vooraleer we de volledige DFA construeren bekijken we eerst het makkelijkere geval waarin automaat N geen ϵ pijlen heeft. Daarna nemen we deze pijlen er wel bij.

1. *$Q' = \mathcal{P}(Q)$.*
   Elke toestand van M is een verzameling toestanden van N.
2. *Voor $R \in Q'$ en $a \in \Sigma$, stel $\delta'(R, a) = \{q \in Q | q \in \delta(r, a), r \in R\}$ Als R een toestand is van M, dan is het ook een verzameling toestanden uit N. Formeel geschreven wordt dit:*

$$\delta'(R, a) = \bigcup_{r \in R} \delta(r, a)$$

   .

3. *$q_0' = \{q_0\}$.*

M start in de toestand die overeenkomt met de starttoestand van N

4. $F' = \{R \in Q' | s \in R \Rightarrow s \in F\}$.
   De automaat M aanvaardt de taal als één van de toestanden waarin M zich bevindt, een eindtoestand is van N.

3.2 Context Vrije Talen

3.2.1 Elke context vrije taal is gegenereerd door een context vrije grammatica in Chomsky-normaal vorm

Bewijs idee 3.3

We kunnen elke grammatica G omzetten in de Chomsky-normaal vorm. Deze omzetting gebeurt door regels, die de condities schenden, om te zetten in equivalente regels die aan de condities voldoen. We beginnen met een start-variabele toe te voegen. Daarna elimineren we alle ϵ ***regels*** *van de vorm $A \to \epsilon$. We elimineren ook alle eenheidsregels van de vorm $A \to B$. Tenslotte zetten we alle overblijvende regels om in zijn juiste vorm.*

Bewijs 3.5

Eerst voegen we een nieuwe start-variabele S_0 en de regel $S_0 \to S$ toe, waar S de originele start-variabele was. Deze verandering garandeert dat de start-variabele niet ergens aan de rechterkant van een regel voorkomt. Daarna behandelen we alle ϵ regels. We verwijderen een ϵ-regel $A \to \epsilon$, waar A niet de start-variabele is. Dan, voor elke A aan de rechterkant van een regel, voegen we een regel toe waarin deze A niet meer voorkomt. Met andere woorden als $R \to uAv$ een regel is met u en v strings van variabelen en eind-symbolen, dan voegen we de regel $R \to uv$ toe. Dit doen we voor elk voorkomen van A. Bijvoorbeeld: $R \to uAvAw$ zorgt voor het toevoegen van volgende regels: $R \to uvAw$, $R \to uAvw$ en $R \to uvw$. Als we de regel $R \to A$ tegenkomen, voegen we de regel $R \to \epsilon$ toe, tenzij we deze voorafgaand verwijderd hadden. We herhalen deze stappen tot we alle ϵ regels zonder start-variabele hebben geëlimineerd.
Hierna behandelen we de eenheidsregels. We verwijderen de eenheidsregel $A \to B$. Vanaf er een regel $B \to u$ optreedt, voegen we de regel $A \to u$ toe, tenzij dit een eenheidsregel was die eerder verwijderd is. Deze stap herhalen we tot er geen eenheidsregels meer zijn.
Tenslotte zetten we alle overblijvende regels om in hun juiste vorm. Dit

wil zeggen dat we elke regel $A \to u_1 u_2 \cdots u_k$, met $k \geq 3$ en elke u_i een variabele of een eind-symbool, omzetten naar $A \to u_1 A_1$, $A_1 \to u_2 A_2, \cdots, A_{k-2} \to u_{k-1} u_k$. De A_i's zijn nieuwe variabelen. Als $k = 2$ vervangen we elk eind-symbool u_i met een nieuwe variabele U_i en voegen de regel $U_i \to u_i$ toe.

3.2.2 Een taal is context vrij als en slechts als ze geaccepteerd wordt door een PDA.

Lemma 3.1 ($\Rightarrow$)

Als een taal context vrij is dan bestaat er een PDA die deze accepteert.

Bewijs 3.6

Stel $P = (Q, \Sigma, \Gamma, \delta, q_1, F)$. Stel q en r twee toestanden van de PDA P en laat a een symbool zijn uit Σ_ϵ en s een symbool zijn uit Γ_ϵ. We voeren volgende transitie in: P gaat van toestand q naar r als ze a leest (van de string) en s popt (van de stack). Met andere woorden: $(a, s \to u)$. u is in dit geval een hele string: $u_1 \cdots u_l$ en wordt volledig op de stack gepushed.
Deze actie kunnen we implementeren door nieuwe toestanden: $q_1, q_2, \cdots q_{l-1}$ te introduceren en de transitiefunctie als volgt te definiëren:

$$\begin{aligned}
\delta(q, a, s) &= \{(q_1, u_l)\} \\
\delta(q_1, \epsilon, \epsilon) &= \{(q_2, u_{l-1})\} \\
\delta(q_2, \epsilon, \epsilon) &= \{(q_3, u_{l-2})\} \\
&\vdots \\
\delta(q_{l-1}, \epsilon, \epsilon) &= \{(r, u_1)\}
\end{aligned}$$

Met andere woorden een reeks tussen toestanden die één voor één een symbool van u op de stapel zetten. In transitienotatie zou dit zijn:

$$\begin{aligned}
a, s &\to u_l \\
\epsilon, \epsilon &\to u_{l-1} \\
\epsilon, \epsilon &\to u_{l-2} \\
&\vdots \\
\epsilon, \epsilon &\to u_1
\end{aligned}$$

Deze signaturen zouden staan tussen de toestanden $q_1, q_2, \cdots, q_{l-1}$

De toestanden van P zijn $Q = \{q_{start}, q_{loop}, q_{accept}\}$ ***TODO***

4 Theorie

4.1 Reguliere Talen

4.1.1 Eindige Automaten

Definitie 4.1

*Een **eindige automaat** is een 5-tal $(Q, \Sigma, \delta, q_0, F)$, waar*

1. *Q is een eindige verzameling met de **toestanden** van de automaat.*
2. *Σ is een eindige verzameling, het **alfabet** genoemd.*
3. *$\delta : Q \times \Sigma \to Q$ is de **transitiefunctie**.*
4. *$q_0 \in Q$ is de **starttoestand**.*
5. *$F \subseteq Q$ is de verzameling **eindtoestanden**.*

Nu er een formele definitie gegeven is voor een eindige automaat, kunnen we formele begrippen definiëren voor berekenbaarheid.

Definitie 4.2

*Een taal noemt men een **reguliere taal** als er een eindige automaat bestaat die deze accepteert.*

Stel $M = (Q, \Sigma, \delta, q_0, F)$ een eindige automaat en $w = w_1 w_2 \cdots w_n$ een string waar elke w_i een element is van het alfabet Σ. Dan accepteert M de string w als er een reeks toestanden $r_0, r_1, \cdots, r_n$ bestaat in Q onder de volgende voorwaarden:

1. $r_0 = q_0$.
2. $\delta(r_i, w_{i+1}) = r_{i+1}$, voor $i = 0, \cdots, n-1$.
3. $r_n \in F$.

Reguliere operaties Er zijn drie hoofdoperaties binnen de reguliere talen en eindige automaten:

Definitie 4.3

*Stel A en B twee talen. We definiëren de **unie**, **concatenatie**, en de **ster** operatie als volgt:*

- ***Unie**: $A \cup B = \{x | x \in A \vee x \in B\}$.*
- ***Concatenatie**: $A \circ B = \{xy | x \in A \wedge y \in B\}$.*
- ***Ster**: $A^* = \{x_1 x_2 \cdots x_k | k \geq \wedge \forall x_i \in A\}$.*

4.1.2 Niet-Determinisme

Verschil NFA DFA Elke toestand van een DFA heeft precies één uitgaande pijl voor elk symbool van het alfabet. Bij een NFA is hoeft dit niet het geval te zijn. Een DFA heeft ook alleen transitielabels voor symbolen uit het alfabet, terwijl een NFA labels mag hebben met het lege string symbool (ϵ) ook al zit dit niet in het alfabet.

Definitie 4.4

*Een **niet-deterministische eindige automaat** is een 5-tal $(Q, \Sigma, \delta, q_0, F)$, waar*

1. *Q is een eindige verzameling toestanden.*
2. *Σ is een eindig alfabet.*
3. *$\delta : Q \times \Sigma_\epsilon \to \mathcal{P}(Q)$.*
4. *$q_0 \in Q$ is de start toestand.*
5. *$F \subseteq Q$ is de verzameling eindtoestanden.*

4.2 Context Vrije Talen

4.2.1 Context Vrije Grammatica

Wat volgt is een voorbeeld van context vrije grammatica. We noemen dit G_1.

$$A \to 0A1$$
$$A \to B$$
$$B \to \#$$

Een grammatica bestaat uit een verzameling **substitutie regels**. Elke regel is een lijn in de grammatica. Deze regel houdt een symbool en een string in gescheiden door een pijl. Het symbool noemen we een **variabele**. De string bestaat uit variabelen en eind-symbolen.
Eén variabele is de start-variabele (meestal aan de linkerkant van de bovenste regel). In het voorbeeld wordt dit:
$\{A, B\}$: Variabelen
$\{0, 1, \#\}$: Eind-symbolen
A : Start-symbool.

Een taal wordt op de volgende manier beschreven door een grammatica:

1. Schrijf de start variabele;
2. Neem de eerst volgende variabele en vind een regel die start met die variabele. Vervang deze variabele met de rechterkant van de regel.
3. Herhaal stap 2 tot er geen meer variabelen zijn om te vervangen

Voorbeeld 4.1

Voorbeeld van de afleiding van grammatica G_1:

$$A \Rightarrow 0A1 \Rightarrow 00A11 \Rightarrow 000A111 \Rightarrow 000B111 \Rightarrow 000\#111$$

Elke taal die gegenereerd kan worden door context vrije grammatica noemt men een **context vrije taal**.

Definitie 4.5

*Een **context vrije grammatica** is een vier-tal (V, Σ, R, S) waar:*

1. *V is een eindige verzameling van variabelen.*
2. *Σ is een eindige verzameling, disjunct van V met elementen die we de eind-symbolen noemen.*
3. *R is de eindige verzameling regels. Elke regel bestaat uit een variabele en een string van variabelen en eind-symbolen.*
4. *$S \in V$ is de start-variabele.*

Context Vrije Grammatica's construeren Context vrije grammatica's construeren vergt veel creativiteit. Vooral voor niet-reguliere expressies.

Voorbeeld 4.2

CVG voor de volgende taal: $\{0^n1^n | n \geq 0\} \cup \{1^n0^n | n \geq 0\}$:

$$S \rightarrow S_1 | S_2$$
$$S_1 \rightarrow 0S_1 1 | \epsilon$$
$$S_2 \rightarrow 1S_2 0 | \epsilon$$

Voor reguliere talen is het eenvoudiger. Maak eerst een DFA voor de taal. Een DFA is a.d.h.v. een algoritme om te zetten in een CVG.

1. Maak een variabele R_i voor elke toestand q_i van de DFA.
2. Voeg de regel $R_i \to aR_j$ aan de CVG toe als $\delta(q_i, a) = q_j$ een overgangsregel is in de DFA.
3. Voeg de regel $R_i \to \epsilon$ toe als q_i een eindtoestand is in de DFA.
4. Maak van R_0 de start-variabele, waar q_0 de starttoestand is van de DFA.

Chomsky Normaal Vorm **Definitie 4.6**

Een context-vrije grammatica staat in ***Chomsky normaal vorm*** *als elke regel van de vorm:*

$$A \to BC$$
$$A \to a$$
$$S \to \epsilon$$

waar a een eindsymbool is en A, B en C variabelen zijn. B en C mogen geen start variabelen zijn. Als extra wordt de laatste regel toegelaten waar S het startsymbool is.

Voorbeeld 4.3

Laat G een CFG zijn. We zetten G om in Chomsky normaal vorm aan de hand van het constructief bewijs in de sectie: Bewijzen.

1. *De originele CFG G staat links en de volgende iteratie van het algoritme rechts. Het resultaat van de eerste iteratie is de nieuwe startvariabele S_0.*

$$S \to ASA|aB \qquad S_0 \to S$$
$$A \to B|S \qquad S \to ASA|aB$$
$$B \to b|\epsilon \qquad A \to B|S$$
$$B \to b|\epsilon$$

2. *De tweede stap is om de ϵ regels te verwijderen. Deze worden verwijderd door op de plekken waar er in dit geval een A voorkomt aan*

de rechter kanten, deze te vervangen door alle mogelijke configuraties. in dit geval: ASA wordt $SA|AS|S$

$$\begin{aligned} S_0 &\to S \\ S &\to ASA|aB|a \\ A &\to B|S|\epsilon \\ B &\to b|\epsilon \end{aligned} \qquad \begin{aligned} S_0 &\to S \\ S &\to ASA|aB|a|SA|AS|S \\ A &\to B|S \\ B &\to b \end{aligned}$$

3. *Daarna verwijderen we de eenheidsregels (regels van de vorm: $A \to B$):*

$$\begin{aligned} S_0 &\to S \\ S &\to ASA|aB|a|SA|AS|S \\ A &\to B|S \\ B &\to b \end{aligned} \qquad \begin{aligned} S_0 &\to ASA|aB|a|SA|AS \\ S &\to ASA|aB|a|SA|AS \\ A &\to B|S \\ B &\to b \end{aligned}$$

$$\begin{aligned} S_0 &\to ASA|aB|a|SA|AS \\ S &\to ASA|aB|a|SA|AS \\ A &\to B|S \\ B &\to b \end{aligned} \qquad \begin{aligned} S_0 &\to ASA|aB|a|SA|AS \\ S &\to ASA|aB|a|SA|AS \\ A &\to ASA|aB|a|SA|AS|b \\ B &\to b \end{aligned}$$

4. *In de laatste stap moeten we de overblijvende regels nog omvormen naar de juiste vormen. Dit houdt bijvoorbeeld in dat regels zoals: $S \to ASA$ en $S \to aB$ (dus met drie letters in de rechterkant of met een eindsymbool en een variabele in de rechterkant) omgevormd moeten worden naar regels met twee letters. Dit doen we door hulpvariabelen te introduceren.*

$$\begin{aligned} S_0 &\to ASA|aB|a|SA|AS \\ S &\to ASA|aB|a|SA|AS \\ A &\to ASA|aB|a|SA|AS|b \\ B &\to b \end{aligned} \qquad \begin{aligned} S_0 &\to AA_1|UB|a|SA|AS \\ S &\to AA_1|UB|a|SA|AS \\ A &\to AA_1|UB|a|SA|AS|b \\ A_1 &\to SA \\ U &\to a \\ B &\to b \end{aligned}$$

4.2.2 PDA: Push-Down Automaten

Push-Down automaten zijn zoals niet-deterministische automaten maar hebben een extra component: De *stack*. We spreken in het heden altijd over een PDA. PDA's zijn equivalent aan CFG. Dit geeft ons een nieuwe manier om te bewijzen of een bepaalde taal al dan niet context vrij is. De *stack* is zoals een stapel papieren. We kunnen enkel de inhoud van het bovenste papier bekijken. We kunnen er ook een papier bij opleggen of een papier afhalen (*push* en *pop*).

Er bestaan zowel deterministische als niet-deterministische PDA's. Opmerkelijk is dat deze niet equivalent zijn.

Definitie 4.7

Een ***push-down automaat*** *is een 6-tal:* $(Q, \Sigma, \Gamma, \delta, q_0, F)$ *waar:*

1. Q *is de eindige verzameling toestanden;*
2. Σ *is het invoer alfabet (eindig);*
3. Γ *is het stack alfabet (eindig);*
4. $\delta : Q \times \Sigma_\epsilon \times \Gamma_\epsilon \to \mathcal{P}(Q \times \Gamma_\epsilon)$ *is de overgangsfunctie;*
5. $q_0 \in Q$ *is de starttoestand;*
6. $F \subseteq Q$ *is de eindige verzameling eindtoestanden.*

Overgangen (bij pijlen) worden als volgt genoteerd: $\alpha, \beta \to \gamma$ waar

- α het invoersymbool;
- β het symbool op de top van de *stack*;
- γ het symbool dat de top van de *stack* vervangt.

Elk van de symbolen kunnen ϵ zijn. Er wordt volgend gevallenonderscheid gemaakt:

- $\alpha = \epsilon$: De automaat mag de transitie doen zonder een symbool te lezen van de invoer;

- $\beta = \epsilon$: De automaat mag de transitie doen zonder een symbool van de *stack* te lezen of te *poppen*;
- $\gamma = \epsilon$: De automaat schrijft niets op de *stack* bij het maken van de transitie.

4.2.3 Niet-context vrije talen

4.3 De Church-Turing Thesis

4.3.1 Turing machines

Het verschil tussen een eindige automaten en Turing machines kan samengevat worden in volgende lijst:

1. Een Turing machine kan schrijven en lezen op en van een tape;
2. De leeskop kan zowel links als rechts bewegen;
3. De tape is oneindig;
4. De eindtoestanden (zowel aanvaarding als verwerping) hebben onmiddellijk effect.