ω-automaten

Martijn Houtepen, 0208523

Begeleider: R. Iemhoff

26 november 2008

Inhoudsopgave

1 Inleiding

In deze scriptie wil ik de theorie van eindige automaten uit het vak Logische Complexiteit bespreken en (vooral) uitbreiden. Een eindige automaat kan omgaan met eindige input, maar wat gebeurt er als de input oneindig lang is? Mijn hoofdonderzoeksvraag zal zijn:

- *Hoe gaan eindige automaten om met oneindige input?*

Deze vraag wil ik opdelen in een aantal subvragen:

- Hoe kan men de theorie over oneindige automaten in kaart brengen?
- Zijn oneindige automaten aan dezelfde regels onderhevig als eindige automaten?
- Wat zijn de verschillen tussen de varianten van oneindige automaten?
- Hoe gaan pushdown-automaten om met oneindige inputs?

Het is moeilijk een voorstelling te maken van een eindige automaat met een oneindige input. Vooral een oneindige input lijkt niet handig. Als bijvoorbeeld gedacht wordt aan de verzameling reële getallen tussen 0 en 1, dan bevat deze verzameling getallen met een oneindig aantal decimalen. Wanneer een automaat de getallen uit deze verzameling wil herkennen, moet deze automaat om kunnen gaan met een oneindige input (namelijk de decimalen). Het feit dat deze getallen herkenbaar zijn voor zo'n automaat kan gebruikt worden om iets te zeggen over de berekenbaarheid van reële getallen. Een ander voorbeeld wat ik later zal bespreken zijn de repeterende breuken. Breuken zoals $\frac{1}{3} = 0,3333\ldots$ hebben oneindig veel decimalen. Met een automaat die om kan gaan met oneindige input kunnen repeterende breuken herkent worden.

Dit artikel is op te splitsen in 2 gedeeltes. Het eerste gedeelte gaat over oneindige inputs op eindige automaten, het tweede gedeelte gaat over oneindige inputs op pushdown-automaten. Dit houdt in dat ik in het eerste gedeelte begin met het uitleggen van de overeenkomsten tussen eindige automaten en Büchi-automaten. Büchi-automaten zijn de simpelste automaten die kunnen omgaan met oneindige input. Daarna laat ik de overeenkomst tussen reguliere talen en ω-reguliere talen zien. Vervolgens zal ik aandacht besteden aan het pomplemma voor reguliere en ω-talen. Ik zal hierna uitweiden over deterministische versus non-deterministische Büchi-automaten. Deterministische eindige automaten zijn gelijk aan non-deterministische eindige automaten, ik zal de vraag beantwoorden of deze gelijkheid ook geldt voor Büchi-automaten. Naast Büchi-automaten bestaan er andere ω-automaten, ook deze zal ik bespreken en de overeenkomsten met Büchi-automaten noemen. In dit gedeelte bespreek ik ook een aantal omschrijvingen (of transformaties) van verschillende automaten naar elkaar. Deze omschrijvingen gebruik ik in het volgende hoofdstuk over het deterministisch maken van ω-automaten.

Hierna begint het tweede gedeelte van het artikel, waarin ik wil beginnen met een hoofdstuk over context-vrije talen en grammatica's. Bij deze context-vrije grammatica's horen de pushdown-automaten die ik in het hoofdstuk erna zal bespreken. Daarna zal ik de context-vrije ω-taal en de Büchi-pushdown-automaten uitleggen. Het volgende hoofdstuk gaat over het pomplemma voor

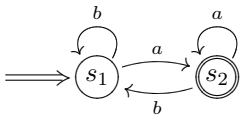
context-vrije talen en voor context vrije ω-talen. In het laatste hoofdstuk zal ik kort het emptiness probleem bespreken. Alle bovenstaande onderdelen wil ik combineren tot een overzicht waarin een duidelijk beeld wordt geschetst van de theorie over ω-automaten.

2 Eindige automaten

Een eindige automaat is een eenvoudig model van een computer. Na een input wordt er een actie ondernomen en wordt de staat van de automaat veranderd. Vanuit deze nieuwe staat wordt er een nieuwe input gelezen en deze input wordt weer verwerkt. Dit proces blijft door gaan totdat er geen input meer is. Op dat moment wordt er gekeken of de automaat een eindstaat heeft bereikt, wat betekent dat de gehele input geaccepteerd wordt. Als de automaat geen eindstaat heeft bereikt, wordt de input niet geaccepteerd.

Definitie. *Een eindige automaat over alfabet Σ is een 4-tupel $M = (Q, q_0, \Delta, F)$, waar Q een eindige set staten is, $q_0 \in Q$ de startstaat is, $\Delta \subseteq Q \times \Sigma \times Q$ de transitiefunctie en $F \subseteq Q$ een eindige set eindstaten. Een woord $u = x_0, x_1, \ldots, x_n$ wordt geaccepteerd als er een run bestaat $r_0, r_1, \ldots, r_n$ in Q zodat $r_0 = q_0$, $\Delta(r_i, x_{i+1}) = r_{i+1}$ voor $i = 0, 1, \ldots, n-1$ en $x(n) \in F$. Als A de set van alle woorden is die een machine M accepteert, dan is A de taal van M: $L(M) = A$ Een deterministische automaat heeft voor elke staat en elk symbool uit het alfabet precies één transitie.*

Een eindige automaat kan deterministisch of non-deterministisch zijn. Als een eindige automaat deterministisch is, is er op elke input vanuit elke staat precies één transitie. Als een eindige automaat non-deterministisch is, zijn er meerdere (of geen) transities mogelijk op een input.



Figuur 1: Een eindige automaat met alfabet $\{a, b\}$ die de taal herkent waarin elk woord het suffix a heeft

3 Büchi-automaten

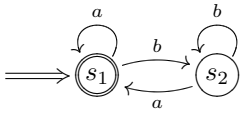
3.1 Büchi-automaten

Een eindige automaat werkt op een eindige input. Sommige problemen vereisen echter een oneindige input. Als je bijvoorbeeld met reële getallen werkt, kunnen deze oneindig veel decimalen hebben. Om deze input te kunnen verwerken is een ander soort automaat nodig. De automaat moet immers om kunnen gaan met een oneindige input (ω-woorden). Een automaat die kan omgaan met ω-woorden wordt een Büchi-automaat genoemd. Het verschil met de eindige automaat zit hem in het feit dat de Büchi-automaat een woord accepteert als een eindstaat

oneindig vaak wordt bereikt, in tegenstelling tot de eindige automaat waar de eindstaat de laatste staat moet zijn waarin de automaat verkeert na het lezen van de input.

Definitie. *Een Büchi-automaat over alfabet Σ is een 4-tupel $\mathcal{A} = (Q, q_0, \Delta, F)$, waar Q een eindige set staten is, $q_0 \in Q$ de startstaat is, $\Delta \subseteq Q \times \Sigma \times Q$ de transitiefunctie en $F \subseteq Q$ een eindige set eindstaten. Een ω-woord $u = x_0, x_1, \ldots$ uit Σ^ω wordt geaccepteerd als er een run $r_0, r_1, \ldots$ bestaat zodat $r_0 = q_0$ en $\Delta(r_i, x_{i+1}, r_{i+1})$ voor $i \geq 0$ en een staat van F oneindig vaak voorkomt in deze run.*

Van de Büchi-automaat bestaat, net als bij de eindige automaat, een deterministische en een non-deterministische variant. De deterministische Büchi-automaat accepteert een input als er precies één run is die de gegeven input herkent, terwijl de non-deterministische Büchi-automaat meerdere verschillende runs toestaat om een input te herkennen. Dit onderscheid zal ik verder bespreken in hoofdstuk 6. Het is niet mogelijk een afbeelding van een Büchi-automaat te onderscheiden van een afbeelding van een eindige automaat. Zie ook figuur 2.



Figuur 2: Een Büchi-automaat met alfabet $\{a, b\}$ die de woorden herkent waarin a oneindig vaak voorkomt

3.2 Voorbeeld: De reële getallen tussen 0 en 1

Om beter te begrijpen waarvoor Büchi-automaten gebruikt kunnen worden, kan bijvoorbeeld gekeken worden naar verzamelingen van reële getallen. Het is vrij simpel om een Büchi-automaat te maken die alle reële getallen in het interval $[0, 1]$ beschrijft. Als er vanuit wordt gegaan dat elk getal in dat interval ofwel oneindig veel decimalen heeft, of eindig veel decimalen gevolgd door oneindig veel nullen, en het alfabet van deze Büchi-automaat $\Sigma = \{0, 1, \ldots, 8, 9\}$, dan hoeft deze automaat enkel alle input te accepteren die mogelijk is met het alfabet. Büchi-automaten zullen dus vooral gebruikt worden om patronen te herkennen in oneindige reeksen.

4 Reguliere talen en ω-reguliere talen

4.1 Reguliere Talen

De automaten worden gebruikt om reguliere talen te beschrijven. Dit zijn talen die een verzameling eindige woorden beschrijven. De verzameling van reguliere talen over een alfabet Σ is recursief gedefinieerd:

Definitie.

- *De lege taal $\emptyset$ is een reguliere taal.*
- *De lege-string taal $\{\epsilon\}$ is een reguliere taal.*
- *Voor alle $a \in \Sigma$, geldt dat de singleton taal $\{a\}$ een reguliere taal is.*
- *Als A en B reguliere talen zijn , dan zijn $A \cup B$ (vereniging), $A \bullet B$ (concatenatie), A^* (Kleene ster), $\overline{A}$ (complement), $A \cap B$ (doorsnee), A^R (omgekeerde) en $\varphi(A)$ (projectie) reguliere talen (vereniging, concatenatie en Kleene ster volstaan om een taal als regulier te beschouwen).*
- *Geen andere talen over Σ zijn regulier.*

Alle eindige talen zijn regulier. Andere voorbeelden van reguliere talen zijn talen die bestaan uit alle strings over het alfabet $\{a, b\}$ met een even aantal a's of de taal van de vorm: een aantal a's gevolgd door een aantal b's. Een andere manier om een reguliere taal te omschrijven is: de taal is regulier dan en slechts dan als een eindige automaat hem herkent. Een zelfde conditie geldt voor de reguliere talen met oneindige input, de zogenaamde ω-reguliere talen.

4.2 ω-reguliere talen

Definitie. *Een taal L is ω-regulier als een Büchi-automaat L herkent.*

Stelling. *Een taal $L \subseteq \Sigma^\omega$ is ω-regulier als het van de vorm $\bigcup_{i \in \{1,2,\ldots,n\}} U_i V_i^\omega$ is, waar U_i en V_i beide een reguliere taal van eindige woorden zijn. (V_i^ω is de ω-iteratie van V, dat wil zeggen als $V \subseteq \Sigma^*$ dan $V^\omega = \{\alpha \in \Sigma^\omega | \alpha = u_0 u_1 u_2 \ldots$ waar $u_i \in V$ voor alle $i \in \mathbb{N}_0\}$.*

ω-Reguliere talen zijn gesloten onder vereniging, complement, intersectie, projectie en Kleene ster. Als A een reguliere taal is en B een ω-reguliere taal dan is hun concatenatie AB ω-regulier. Als $L \subseteq \Sigma^\omega$ Büchi-herkenbaar is, is $\Sigma^\omega - L$ dat ook (zie ook de stelling hierboven). Of de ω-reguliere talen ook gesloten zijn onder omkering is onbekend. Door de oneindigheid van de input is het moeilijk te zeggen of het omkeren van de input een ω-reguliere taal geeft.

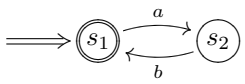
5 Pomplemma

5.1 Pomplemma voor reguliere talen

Het pomplemma stelt dat elke reguliere taal de volgende eigenschap heeft: Als A een reguliere taal is, is er een natuurlijk getal $p \geq 1$ zodat elk woord u met lengte minimaal p (de pomplengte) geschreven kan worden als $u = xyz$, waarbij x, y, z voldoen aan de volgende voorwaarden:

1. $|y| > 0$
2. $|xy| \leq p$
3. voor alle $i \geq 0$, $xy^i z \in A$

Dus y kan 'gepompt' worden (herhaald) en het resulterende woord zit ook altijd in A. Bijvoorbeeld de taal $\{ab\}^*$, de taal met woorden die bestaan uit een willekeurig aantal maal ab achter elkaar. Het woord ab zit in deze taal, net als $abab$ en $ababab$. Als dat laatste woord opgedeeld wordt in drie delen, $x = ab$, $y = ab$ en $z = ab$, dan is het zo dat het woord $xyyz$ ($abababab$) in deze taal zit (en ook $xyyyz$, $xyyyyz$, etc). Zie ook figuur 3.



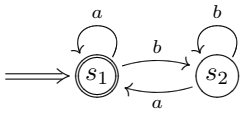
Figuur 3: Een automaat voor de taal $\{ab\}^*$

5.2 Pomplemma voor ω-reguliere talen

Het pomplemma is een eigenschap waaraan alle reguliere talen voldoen. De essentie van het pomplemma is dat om een bepaald woord te accepteren, de automaat door een specifieke opeenvolging van staten gaat. Deze opeenvolging van staten kan onbeperkt herhaald worden, de uitkomst is altijd een eindstaat. In ω-reguliere talen is het duidelijk te zien dat ook hiervoor het pomplemma geldt. Het aantal staten is eindig en de input oneindig. Om een (oneindig) woord te herkennen moet er minstens één staat zijn die oneindig vaak bezocht wordt. Het enige verschil tussen reguliere talen en ω-reguliere talen is immers de lengte van de input die deze talen kunnen verwerken. Omdat een woord pas wordt geaccepteerd als een eindstaat in een Büchi-automaat oneindig vaak bezocht wordt, is er een bepaalde 'loop' die oneindig vaak doorlopen wordt. Deze loop is dan ook de y van het pomplemma (het gedeelte wat gepompt wordt) van de ω-reguliere taal. Als we figuur 3 als een Büchi-automaat beschouwen, is de opeenvolging van staten die oneindig vaak herhaald gaat worden $s_1, s_2, s_1, \ldots$ Het is zelfs zo, dat er bij Büchi-automaten 'recursief' gepompt kan worden. Het gepompte gedeelte zelf kan ook weer gepompt worden. Wanneer een woord drie keer gepompt is, kan het gepompte gedeelte zelf ook weer gepompt worden.

6 Deterministisch vs non-deterministisch

In hoofdstuk 3 heb ik laten zien dat Büchi-automaten, net als eindige automaten, deterministisch of non-deterministisch kunnen zijn. Deterministische eindige automaten zijn gelijk aan non-deterministische eindige automaten, maar bij Büchi-automaten gaat dit niet op. Het verschil tussen deterministische Büchi-automaten en non-deterministische Büchi-automaten is dat een deterministische Büchi-automaat precies één run heeft die een gegeven input herkent. Deze eigenschap is een beperkende factor voor deterministische Büchi-automaten en zorgt er ook voor dat zij minder sterk zijn dan hun non-deterministische 'broertjes'. Het kan aangetoond worden dat deterministische Büchi-automaten niet gesloten zijn onder complement, terwijl non-deterministische Büchi-automaten dit wel zijn. Een voorbeeld hiervan is de eerder beschreven taal, waarin een woord wordt geaccepteert als het oneindig veel a's bevat. Zie figuur 4.



Figuur 4: Een Büchi-automaat met alfabet $\{a, b\}$ die de woorden herkent waarin a oneindig vaak voorkomt

Stelling. *Deterministische Büchi-automaten zijn niet gesloten onder complement, maar non-deterministische Büchi-automaten zijn dit wel.*

Bewijs: Neem de taal L die bestaat uit alle oneindige woorden α die gemaakt kunnen worden uit het alfabet $\{a, b\}$ zodat er oneindig veel voorkomens van a in α zitten. Een deterministische Büchi-automaat kan simpel gemaakt worden die L te herkent, maar om het complement $\overline{L}$ te herkennen is een non-determinische Büchi-automaat nodig. Een formeler bewijs vereist de notie 'limiettaal'.

Definitie. *Laat $U \subseteq \Sigma^*$ een taal van eindige woorden zijn. Een woord behoort tot limiet van U, $lim(U)$, desda het oneindig veel prefixen in U heeft.*

Deze notie ga ik nu gebruiken om te laten zien dat een taal L herkenbaar is door een deterministische Büchi-automaat dan en slechts dan als die taal van de vorm $lim(U)$ voor een reguliere taal $U \subseteq \Sigma^*$ is.

Stelling. *Een taal L is herkenbaar door een deterministische Büchi automaat desda hij de vorm $lim(U)$ voor een reguliere taal $U \subseteq \Sigma^*$ is.*

Bewijs: Als U een reguliere taal is, bestaat er een deterministische eindige automaat M zodat M U herkent. Als M omgeschreven wordt naar een Büchi-automaat K (alle karakteristieken blijven behouden, alleen veranderen de set eindstaten F van de eindige automaat naar de set eindstaten van K) herkent de Büchi-automaat alle woorden die gevormd worden door combinaties van door M gevormde woorden. De eindige automaat M herkent een aantal woorden en de Büchi-automaat K, doordat hij oneindig lange woorden herkent, herkent de oneindig lange opeenvolging van deze woorden achter elkaar. Met andere woorden, de woorden die de Büchi-automaat K herkent zijn een opeenvolging van woorden uit de eindige automaat M en dus zijn alle woorden die M herkent, prefixen van woorden die K herkent en geldt dat K $lim(U)$ accepteert.

Andersom geldt hetzelfde. Laat L een taal zijn die door de deterministische Büchi-automaat N herkend wordt. Er wordt een een eindige automaat P gemaakt die dezelfde eindstaten als N heeft en de taal U herkent. In plaats van de oneindig lange woorden die N herkende, zijn de woorden die P herkent korte stukjes van de oneindige woorden uit de taal L. Deze stukjes kunnen ook weer beschouwd worden als prefixen van de oneindige woorden en hier geldt dus ook dat $L = lim(U)$.

7 Rabin-, Muller- en Streett-automaten

Naast Büchi-automaten zijn er andere automaten die even sterk als of sterker dan non-deterministische Büchi-automaten zijn. Deze automaten lijken heel veel op Büchi-automaten, alleen hebben zij andere acceptatievoorwaarden. In plaats van een staat die oneindig vaak doorlopen moet worden voor acceptatie, bezitten deze automaten een tabel met staten die oneindig vaak bezocht worden. Een eerste variant van zo'n automaat is de Rabin-automaat.($inf(\rho)$ betekent hier de oneindig lange run ρ)

7.1 Rabin-automaat

Definitie. *Een Rabin-automaat is een structuur* $(\mathcal{A}, \mathcal{PT})$ *waar* $\mathcal{A} = (S, \rightarrow, S_{in})$ *een automaat is en* $\mathcal{PT} = \langle (F_1, R_1), ((F_2, R_2), \ldots, (F_k, R_k)) \rangle$ *een tabel met paren is met* $F_i, R_i \subseteq S$ *voor een* $i \in \{1, 2, \ldots, k\}$.
De automaat $(\mathcal{A}, \mathcal{PT})$ *accepteert een input* $\alpha : \mathbb{N}_0 \rightarrow \Sigma$ *als er een run* ρ *van* $\mathcal{A}$ *op* α *is zodat voor een* $i \in \{1, 2, \ldots, k\}$, $inf(\rho) \cap F_i \neq \emptyset$ *en* $inf(\rho) \cap R_i = \emptyset$

Elk paar uit de acceptatie tabel bevat een staat die oneindig vaak bezocht wordt (F) en een staat die niet oneindig vaak bezocht wordt (R). Voor de taal met oneindig veel a's, die ik al eerder heb besproken, zal de acceptatietabel er als volgt uit zien: $\langle (\{s_1\}, \emptyset) \rangle$. Büchi-automaten zijn simpel om te schrijven naar Rabin automaten, Van een Büchi-automaat dient de set G van acceptatiestaten vervangen te worden door de tabel $\langle (\{F\}, \emptyset) \rangle$.

7.2 Muller-automaat

Een Muller-automaat lijkt heel veel op een Rabin-automaat, maar in plaats van een acceptatietabel met paren van staten die oneindig vaak en niet oneindig vaak bezocht worden, heeft de Muller-automaat een acceptatietabel met staten die oneindig vaak bezocht moeten worden.

Definitie. *Een Muller-automaat is een paar* $(\mathcal{A}, \mathcal{T})$ *waar* $\mathcal{A} = (S, \rightarrow, S_{in})$ *een automaat is en* $\mathcal{T} = \langle F_1, F_2, \ldots, F_k \rangle$ *een acceptatietabel is met* $F_i \subseteq S$ *voor een* $i \in \{1, 2, \ldots, k\}$.
De automaat $(\mathcal{A}, \mathcal{T})$ *accepteert een input* $\alpha : \mathbb{N}_0 \rightarrow \Sigma$ *als er een run* ρ *van* $\mathcal{A}$ *op* α *is zodat* $inf(\rho) \in \mathcal{T}$ *oftewel* $inf(\rho) = F_i$.

Alle staten in F_i moeten dus oneindig vaak bezocht worden, en alle staten die niet in F_i zitten mogen maar eindig vaak bezocht worden. Nogmaals kijkend naar de taal met oneindig veel a's blijft de automaat die deze taal herkent hetzelfde, alleen wordt de acceptatietabel $\langle \{s_1\}, \{s_1, s_2\} \rangle$. Dit lijkt raar, omdat s_2 nu ook in de acceptatietabel staat, maar met deze tabel wordt bedoeld dat de automaat oneindig vaak door s_1 of oneindig vaak door de opeenvolgende staten $\{s_1, s_2\}$ moet. Een deterministische Muller-automaat is sterker dan een deterministische Büchi-automaat. Dit kan aangetoond worden door weer naar de taal met oneindig veel a's te kijken. Deze taal is door beide herkenbaar, maar zijn complement is niet herkenbaar door een deterministische Büchi-automaat en wel door Muller-automaat. Dit is te zien door dezelfde automaat als eerder te nemen maar de acceptatietabel voor de Muller-automaat te wijzigen in $\{s_2\}$.

Dit zorgt ervoor dat het complement wordt herkend.

Büchi-automaten zijn simpel om te schrijven naar Muller-automaten. Elke subset van staten die een die een acceptatie staat bevat in de Büchi-automaat, komt in de acceptatie tabel van de Muller-automaat.

7.3 Streett-automaat

Een Streett-automaat lijkt heel erg veel op een Rabin-automaat, alleen is de voorwaarde om een input te herkennen iets anders. De Streett-automaat beschrijft precies het complement van een Rabin-automaat met de zelfde acceptatietabel.

Definitie. *Een Streett-automaat is een structuur* $(\mathcal{A}, \mathcal{PT})$ *waar* $\mathcal{A} = (S, \rightarrow, S_{in})$ *een automaat is en* $\mathcal{PT} = \langle (F_1, R_1), ((F_2, R_2), \ldots, (F_k, R_k)) \rangle$ *een tabel met paren is met* $F_i, R_i \subseteq S$ *voor een* $i \in \{1, 2, \ldots, k\}$.
De automaat $(\mathcal{A}, \mathcal{PT})$ *accepteert een input* $\alpha : \mathbb{N}_0 \rightarrow \Sigma$ *als er een run* ρ *van* $\mathcal{A}$ *op* α *is zodat voor een* $i \in \{1, 2, \ldots, k\}$*, als* $inf(\rho) \cap F_i \neq \emptyset$ *dan* $inf(\rho) \cap R_i \neq \emptyset$.

Stelling. *Een Streett-automaat kan worden omgeschreven naar een Büchi-automaat.*

Laat $(\mathcal{A}, \mathcal{PT})$ een Streett-automaat zijn met $\mathcal{A} = (S, \rightarrow, S_{in})$. Laat k het aantal paren in $\mathcal{PT}$ zijn. Als van een paar $\langle F_i, R_i \rangle \in \mathcal{PT}$ staat F_i oneindig vaak voorkomt, dan moet staat R_i ook oneindig vaak voorkomen. Er word een Büchi automaat $\mathcal{A}', F'$ met $\mathcal{A}' = (S', \rightarrow', S'_{in})$ gemaakt zodat $L(\mathcal{A}', F') = L(\mathcal{A}, \mathcal{PT})$. De Büchi-automaat houdt twee verzamelingen bij, de eerste verzameling houdt een lijst bij van indices van paren $\langle F_i, R_i \rangle$ (uit de oorspronkelijke Streett-automaat) waarin een element van F_i oneinig vaak voorkomt. De tweede verzameling houdt een lijst bij van indices van paren $\langle F_i, R_i \rangle$ waarin een element van R_i bezocht is. Elke keer als de tweede set even groot is geworden als de eerste wordt hij leeggemaakt. Als die set oneindig vaak wordt leeggemaakt accepteert de Büchi-automaat hetzelfde als de Streett-automaat.

8 Deterministisch maken

Om het complement van een deterministische eindige automaat te maken, wordt deze eerst omgeschreven naar een non-deterministische eindige automaat. Vervolgens wordt de automaat gecomplementeerd en terug omgeschreven naar een eindige automaat. Deze manier om een automaat te complementeren is mogelijk omdat de deterministische en non-deterministische eindige automaten gelijk zijn aan elkaar in uitdrukkingskracht. In hoofdstuk zes is te zien dat dit bij Büchi-automaten niet op gaat.

Zoals ik eerder heb laten zien, is er een non-deterministische Büchi-automaat nodig om het complement van een taal die beschreven word door een deterministische Büchi-automaat weer te geven. Maar wat als het complement van een non-deterministische Büchi-automaat nodig is? Er bestaat een manier om non-determinische Büchi-automaten om te schrijven naar andere Büchi-automaten die hun complement uitdrukken, dit proces heet Safra's constructie [5]. Zie appendix A voor een schematische weergave van omschrijvingen en zie ook hoofdstuk 7 voor een meer gedetailleerde beschrijving van het omschrijven.

Safra's constructie:

Stap 1 Schrijf een non-deterministische Büchi-automaat $(\mathcal{A}, F)$ om naar een deterministische Rabin-automaat $(\mathcal{A}_F, \mathcal{PT}_F)$, zodat de taal van de Rabin-automaat $L(\mathcal{A}_F, \mathcal{PT}_F)$ gelijk is aan de taal van de Büchi-automaat $L(\mathcal{A}, F)$.

Stap 2 Beschouw de verkregen Rabin-automaat $(\mathcal{A}_F, \mathcal{PT}_F)$ als een Streett-automaat (dit geeft het complement).

Stap 3 Schrijf de Streett-automaat om naar een Büchi-automaat.

9 Context-vrije talen

Niet alle eindige talen kunnen worden herkend door een eindige automaat. Een krachtigere formele taal is nodig om talen zoals $\{a^n b^n | n \geq 0\}$ te herkennen. Een categorie van formele talen die sterker zijn dan de reguliere talen zijn de context-vrije talen.

9.1 Context-vrije grammatica's

Een context-vrije taal is herkenbaar door een context-vrije grammatica.

Definitie. *Een context-vrije grammatica is een 4-tupel $G = (V, A, R, S)$ waar V een eindig alfabet is, $A \subseteq V$ het terminal-alfabet (een terminal is een symbool zoals het in de input voorkomt en is geen variabele), $R \subset (V - A) \times V^*$ een eindige set regels en $S \in V - A$ de startvariabele.*

Als u, v en w woorden van variabelen en terminals zijn en $A \to w$ een regel is, dan wordt gesteld uAv geeft uwv, $uAv \Rightarrow uwv$. Als $u = v$ of als er een opeenvolging $u_1, u_2, \ldots, u_k$ bestaat voor $k \geq 0$ en $u \Rightarrow u_1 \Rightarrow u_2 \Rightarrow \ldots \Rightarrow u_k \Rightarrow v$, dan is de denotatie $u \Rightarrow^* v$. De taal van de context-vrije grammatica is $\{w \in A^* | S \Rightarrow^* w\}$. Context-vrije grammatica's worden vaak in de Chomsky-normaalvorm gezet, dit houdt in dat alle regels van de volgende vorm zijn

1. $A \to BC$ of
2. $A \to \alpha$ of
3. $S \to \epsilon$

waar A, B en C geen terminals zijn, α een terminal is, S het startsymbool is en ϵ het lege woord is.

9.2 Pushdown-automaat

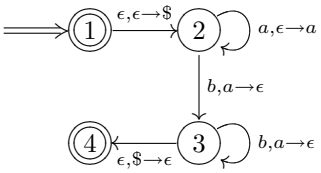
Een automaat om de context-vrije talen te herkennen is de pushdown-automaat (pda). Deze lijkt op een eindige automaat, met het verschil dat de pda over een geheugenstack beschikt waarnaar symbolen geschreven kunnen worden en waarvan gelezen kan worden. Een formele definitie van een pda is als volgt:

Definitie. *Een pushdown-automaat is een 6-tupel* $M = (Q, \Sigma, \Gamma, \Delta, q_0, F)$*, waar* Q *een eindige set van staten is,* Σ *het (eindige) input-alfabet,* Γ *het (eindige) stack-alfabet,* $\Delta : Q \times \Sigma_\epsilon \times \Gamma_\epsilon \to \mathcal{P}(Q \times \Gamma_\epsilon)$ *de transitiefunctie,* $q_0 \in Q$ *de start-staat en* $F \subseteq Q$ *de eindige set van eindstaten is.*

Een pda $M = (Q, \Sigma, \Gamma, \Delta, q_0, F)$ accepteert input w als w geschreven kan worden als $w = w_1, w_2, \ldots, w_m$ waar elke $w_i \in \Sigma_\epsilon$ en er een opeenvolging van staten $r_0, r_1, \ldots, r_m \in Q$ en strings $s_0, s_1, \ldots, s_m \in \Gamma^*$ bestaat die aan de volgende drie condities voldoet (s_i is hier de opeenvolging van stack-inhoud die M heeft op de accepterende tak van de computatie):

1. $r_0 = q_0$ en $s_0 = \epsilon$.
2. Voor $i = 0, \ldots, m-1$ geldt $(r_{i+1}, b) \in \Delta(r_i, w_i, a)$, waar $s_i = at$ en $s_{i+1} = bt$ voor een $a, b \in \Gamma_\epsilon$ en $t \in \Gamma^*$.
3. $r_m \in F$.

Als een pda een taal herkent is deze taal context-vrij. Zie ook figuur 5.



Figuur 5: Een pushdown-automaat die de taal $\{a^n b^n | n \geq 0\}$ herkent (na de komma staat wat de actie op de stack is)

10 Büchi-pushdown-automaten en context-vrije ω-talen

10.1 Büchi-pushdown-automaten

Ook pushdown-automaten kunnen net als de eindige automaten niet omgaan met oneindige input. Er zijn talen die niet door een Büchi-automaat worden herkend, maar wel door een pushdown-automaat. Een voorbeeld hiervan is de taal van de palindromen, elk woord uit deze taal is omgekeerd gelezen hetzelfde woord. Ook zijn er talen die niet door een standaard pushdown-automaat worden herkend, omdat ze oneindige woorden bevatten. Voor deze talen is er analoog aan de Büchi automaat, de Büchi-pushdown-automaat (Bpda). Dit is in feite een pushdown-automaat die kan omgaan met oneindige input.

Definitie. *Een Büchi-pushdown-automaat(Bpda) is een 7-tupel* $\mathcal{A} = (Q, \Sigma, \Gamma, \Delta, q_0, F)$*, waar* Q *een eindige set van staten is,* Σ *het (eindige) input-alfabet,* Γ *het (eindige) stack-alfabet,* $\Delta : Q \times \Sigma_\epsilon \times \Gamma_\epsilon \to \mathcal{P}(Q \times \Gamma_\epsilon)$ *de transitiefunctie,* $q_0 \in Q$ *de startstaat en* $F \subseteq Q$ *de set van eindstaten is.*

10.2 Context-vrije ω-talen

Deze Bpda accepteert een input wanneer een set van eindstaten oneindig vaak bezocht wordt tijdens de input. De Bpda wordt gebruikt om een context-vrije ω-taal te herkennen. Een formele omschrijving van een context-vrije ω-taal ('oneindige taal'):

Definitie. *Een ω-taal $L \subseteq V^\omega$ is context-vrij als er een context-vrije grammatica G is en een systeem $\mathcal{F}$ van sets van variabelen van G zodat L bestaat uit de woorden van A^ω die gegenereerd worden van startvariabele x_1 door de meest linkse afleiding van G waar de variabelen die oneindig gebruikt worden een set vormen in $\mathcal{F}$.*

10.3 Voorbeeld: Repeterende breuken

Een voorbeeld dat inzicht geeft in Büchi-pushdown-automaten zijn de repeterende breuken. Een breuk is repeterend als er een decimaal of een groep decimalen is die oneindig vaak herhaald wordt. Bijvoorbeeld $\frac{1}{3} = 0,33333\ldots$ waar de puntjes staan voor een oneindige herhaling van drieën. Het te herhalen gedeelte kan ook langer zijn dan één teken, bijvoorbeeld $\frac{1}{7} = 0,142857142857142857\ldots$. Een derde mogelijkheid is dat de repetitie pas optreedt na een aantal decimalen, zoals $\frac{1}{6} = 0,16666\ldots$ (breuken als $\frac{1}{4}$ beschouwd men als $0,2500000\ldots$ en vallen ook in deze categorie). Voor deze verzameling getallen is er geen Büchi-automaat te maken, maar wel een Büchi-pushdown-automaat. Een Bpda $\mathcal{A}$ met als alfabet $\Sigma = \{0, 1, \ldots, 8, 9\}$ die wanneer het repeterende gedeelte begint, dit op zijn stack zet. Daarna (als het repeterende gedeelte voor de tweede keer begint) haalt hij de decimalen weer van de stack af. Als hij oneindig vaak een lege stack heeft accepteert hij de input.

11 Pomplemma voor context-vrije talen en context-vrije ω-talen

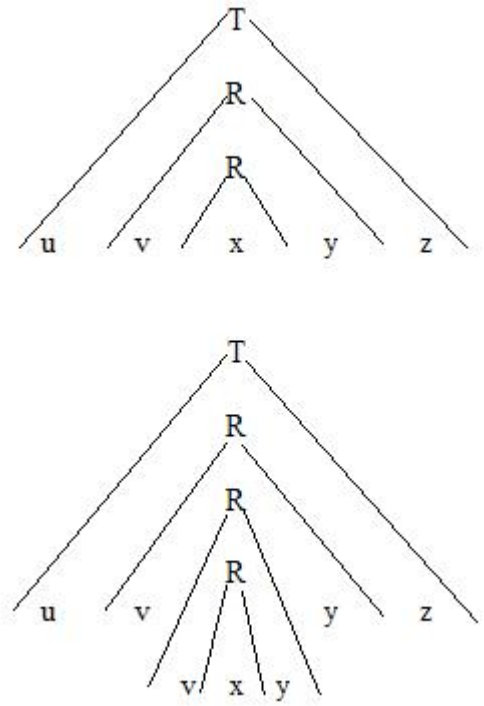
11.1 Pomplemma voor context-vrije talen

Het pomplemma voor context-vrije talen stelt dat elke context-vrije taal de volgende eigenschap heeft: Als A een context-vrije taal is, dan is er een integer $p \geq 1$ zodat elk woord u met lengte minimaal p (de pomp lengte) geschreven kan worden als $u = uvxyz$ en voldoet aan de volgende voorwaarden:

1. $|vy| > 0$
2. $|vxy| \leq p$
3. voor alle $i \geq 0$, $uv^ixy^iz \in A$

v en y kunnen 'gepompt' worden (herhaald) en het resulterende woord zit ook altijd in A. Bijvoorbeeld de taal van de palindromen, de taal met woorden die gespiegeld kunnen worden. Het woord *ababa* zit in deze taal, net als *abaaba* en *abababa*. Dat laatste woord kan opgedeeld worden in 5 delen, $u = ab$, $v = a$, $x = b$, $y = a$ en $z = ba$, nu is het zo dat het woord $uvvxyyz$ (*abaabaaba*) in deze taal zit (en ook $uvvvxyyyz$, $uvvvvxyyyyz$, etc). Om de werking van

het pomplemma voor context-vrije grammatica's weer te geven, kan er gedacht worden aan een boomstructuur. Een afleidingsboom van een woord *cabac* kan gezien worden als een afleiding voor $x = b$ met daaromheen een afleiding voor $vy = axa$ en daar weer omheen een afleiding voor $uz = cvxyc$. Een gedeelte van deze boom is los te koppelen (namelijk de afleiding $vy = axa$) en omdat de regels van de grammatica toelaten een afleidingsregel te herhalen, kan dit geheel vervangen door $vvyy$ wat gelijk is aan $aaxaa$. Zie ook figuur 6.



Figuur 6: Pomplemma bij context-vrije talen

11.2 Pomplemma voor context-vrije ω-talen

In ω-reguliere talen is het duidelijk in te zien dat ook hiervoor het pomplemma geldt. Ook bij ω-reguliere talen is de afleiding van een woord te beschouwen als een boomstructuur. Aangezien er een eindig aantal regels in de grammatica bestaat en het woord oneindig lang is, is er een regel die oneindig vaak toegepast moet worden. Het resultaat van deze regel, een knoop in de boomstructuur met zijn takken, is een set van staten die oneindig vaak bezocht wordt. Er is dus een bepaalde knoop in de boom die oneindig vaak bereikt wordt en de bladeren van deze tak zijn v en y van het pomplemma van de ω-contextvrije taal. Zie ook figuur 5.

12 Emptiness-probleem

12.1 Emptiness-probleem bij automaten

Een bekend probleem in de theorie over automaten is het probleem of een bepaalde automaat 'leeg' is, en dus geen enkele input accepteert. Dit heet het emptiness-probleem. Bij eindige automaten is dit probleem beslisbaar, door alle mogelijke transities na te gaan en te kijken of er een pad naar de eindstaat is. Omdat een Büchi-automaat, net als een eindige automaat, een eindig aantal transities en staten heeft, is ook voor Büchi-automaten dit probleem beslisbaar.

12.2 Emptiness probleem bij context-vrije grammatica's

Het emptiness probleem bestaat ook bij context-vrije grammatica's, hier is het de vraag of de grammatica 'leeg' is. Dit is beslisbaar door alle mogelijke regels te volgen en voor elke variabele te kijken of hij een terminal geeft. Als er geen enkele manier is om van het startsymbool naar een terminal te komen is de taal 'leeg'. Voor context-vrije ω-talen is dit niet bekend. De grammatica's voor deze talen bestaan wel uit eindig veel regels en variabelen, maar het feit dat er een terminal door een bepaalde regel gegenereerd kan worden garandeert niet dat deze terminal zich in de meest linkse afleiding bevindt en dus word geacccepteerd.

13 Conclusie

In mijn inleiding heb ik een aantal vragen gesteld die ik in dit artikel wilde beantwoorden, waarvan de belangrijkste was:

- *Hoe gaan eindige automaten om met oneindige input?*

Samenvattend kan gezegd worden dat automaten bijna precies hetzelfde omgaan met oneindige input als met eindige input en ook aan bijna dezelfde regels voldoen. Dit geldt zowel voor de Büchi-automaten als vervanging voor eindige automaten als voor de Büchi-pushdown-automaten als vervanging voor de pushdown-automaten.
Een tweede vraag in mijn inleiding betreft de verschillen tussen de verschillende soorten ω-automaten, de Büchi-, Rabin-, Muller- en Streett-automaten. Deze automaten lijken veel op elkaar, maar verschillen in de acceptatie-condities en in de kracht van de automaat.

A Omschrijven van Büchi-, Rabin-, Streett- en Muller-automaten

In het hoofdstuk over determinisatie en het hoofdstuk over Rabin-, Streett- en Muller-automaten heb ik laten zien hoe sommige automaten omgeschreven kunnen worden naar andere automaten. Belangrijk om hierbij in acht te nemen is dat de meeste automaten naar elkaar omgeschreven kunnen worden, al is niet elke transitie even makkelijk. Alle automaten zijn subklassen van Muller-automaten.

Stelling. *Deterministische Büchi* $\subset$ *non-deterministische Büchi* $\subseteq$ *Rabin* $\subseteq$ *Streett* $\subset$ *Muller*

Hieronder een kort schematisch overzicht van de verschillende omschrijfmanieren. Wanneer het niet in de tabel staat is het proces van omschrijven niet simpel uit te leggen of te bewijzen. Zie figuur 7:

Omschrijftabel		
Van	*Naar*	*Commentaar*
Büchi	Rabin	$\mathcal{PT} = (\langle F, \emptyset \rangle)$
Büchi	Streett	$\mathcal{PT} = (\langle Q, F \rangle)$
Büchi	Muller	$\mathcal{PT}$ = alle subsets waarin een staat uit F voorkomt
Rabin	Streett	Zijn elkaars complement
Streett	Rabin	Zijn elkaars complement
Streett	Büchi	zie de sectie over Streett automaten

Figuur 7: Omschrijftabel

Referenties

[1] J. R. Büchi. On a decision method in restricted second order arithmetic. *Z. Math. Logic Grundlag. Math*, 6:66–92, 1960.

[2] Madhavan Mukund. Finite-state automata on infinite inputs, 1996.

[3] D.E. Muller. Infinite sequences and finite machines. In *Proceedings 4th IEEE Symposium on switching circuit theory and logical design*, pages 3–16, 1936.

[4] M.O. Rabin. Decidability of second order theories and automata on infinite trees. *Transactions of the American Mathematical Society*, 141:1–37, 1969.

[5] S. Safra. On the complexitity of ω-automata. In *Proceedings 29th IEEE FOCS*, pages 319–327, 1988.

[6] Michael Sipser. *Introduction to the Theory of Computation*. PWS Publishing Company, 2 edition, 2005.

[7] Wolfgang Thomas. Automata on infinite objects. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*, volume B, pages 133–164. Elsevier, 1990.