Logische Complexiteit
Context-Vrije Grammatica's

Jeroen Goudsmit

22 februari 2012

In dit document worden een paar varianten op PDA's gegeven. We laten zien dat ze allemaal equivalent zijn. Uiteindelijk tonen we ook aan dat iedere taal die herkent wordt door een CFG ook herkent wordt door een PDA.

Definitie 1 (Context-Vrije Grammatica)**.** Een context-vrije grammatica (CFG) is een tupel $\langle V, \Sigma, R, \mathsf{S} \rangle$ met:

V	eindige verzameling	*variabelen*
Σ	eindige verzameling	*alfabet*
R	eindige verzameling	*regels*
$\mathsf{S} \in V$		*startvariabele*

De *regels* zijn hier paren van variabelen en rijtjes over $\Sigma \cup V$ geschreven als $\mathsf{A} \to w$, waar $\mathsf{A} \in V$ en $w \in (\Sigma \cup V)^*$. Als afkorting schrijven we wel eens $\mathsf{A} \to u \mid v$, dit staat voor de twee regels $\mathsf{A} \to u$ en $\mathsf{A} \to v$.

Om gemakkelijk een afleiding van een woord te maken introduceren we de relatie $\Rightarrow \subseteq (V \cup \Sigma)^* \times (V \cup \Sigma)^*$. Deze is gedefiniëerd als volgt.

Definitie 2. Zij $\langle V, \Sigma, R, \mathsf{S} \rangle$ een CFG. Gegeven u en v uit $(\Sigma \cup V)^*$ zeggen we nu dat $u\ \mathsf{A}\ v$ *geeft* (yields) $u\ w\ v$ wanneer $\mathsf{A} \to w \in R$. We zeggen dat u *genereert* v indien

$$u = w_0 \Rightarrow w_1 \Rightarrow \ldots \Rightarrow w_n = v \quad \text{voor zekere } w_0, \ldots, w_n \in (\Sigma \cup V)^*. \tag{1}$$

In dit geval schrijven we $u \overset{*}{\Rightarrow} v$.

Merk op dat in (1) je krijgt $u = v$ wanneer $n = 0$. Oftewel, het is altijd zo dat $u \overset{*}{\Rightarrow} u$. Het is ook niet lastig om te zien dat als $u \overset{*}{\Rightarrow} v$ en $v \overset{*}{\Rightarrow} w$ dat dan $u \overset{*}{\Rightarrow} w$ geldt. Als alternatieve definitie hadden we ook simpelweg kunnen zeggen "$\overset{*}{\Rightarrow}$ is de reflexief-transitieve afsluiting van $\Rightarrow$". In symbolen komt dat neer op

$$\overset{*}{\Rightarrow} = \bigcap \left\{ R \;\middle|\; R \text{ is een reflexieve en transitieve relatie op } (\Sigma \cup V)^* \text{ en } \Rightarrow \subseteq R \right\}.$$

De *taal* van een CFG $G = \langle V, \Sigma, R, \mathsf{S} \rangle$ is nu gegeven door

$$\mathcal{L}(G) := \left\{ w \in \Sigma^* \;\middle|\; \mathsf{S} \overset{*}{\Rightarrow} w \right\}.$$

Een taal $\mathcal{L}$ waarvoor een CFG G bestaat met $\mathcal{L}(G) = \mathcal{L}$ heet een *context-vrije taal.*

Voorbeeld 1 (Palindromen)**.** De taal $\mathcal{L} = \{w \in \Sigma^* \mid w \text{ is een palindroom}\}$ over het alfabet $\Sigma = \{a, b\}$ is een context-vrije taal. Ze wordt herkent door de volgende CFG, die S als startsymbool heeft.

$$\mathsf{S} \quad \to \quad a\ \mathsf{S}\ a \mid b\ \mathsf{S}\ b \mid a \mid b \mid \varepsilon \tag{2}$$

Laten we dit formeel bewijzen. We willen aantonen dat voor elke $w \in \Sigma^*$ we hebben $\mathsf{S} \overset{*}{\Rightarrow} w$ dan en slechts dan als w een palindroom is. We doen dit in twee stappen.

Eerst laten we zien dat als $\mathsf{S} \overset{*}{\Rightarrow} w$ dat dan w een palindroom is. Dit gaat met inductie naar het aantal stappen van $\Rightarrow$ in deze afleiding. Voor 1 stap zijn er maar drie mogelijkheden, namelijk $w = a$, $w = b$ of $w = \varepsilon$. In al deze gevallen is het woord een palindroom. Stel nu dat S het woord w afleidt in $n + 1$ stappen. Dit betekent dat $\mathsf{S} \Rightarrow x\ \mathsf{S}\ x \overset{*}{\Rightarrow} x\ v\ = w$ voor $x = a$ of $x = b$. In ieder geval weten we nu dat $\mathsf{S} \overset{*}{\Rightarrow} v$ in n stappen. Met inductie is v nu een palindroom, en daarom is $x\ v\ x = w$ er ook een. Dit bewijst de ene kant.

Om nu de andere kant aan te tonen, nemen we aan dat w een palindroom is. We bewijzen hier dat $\mathsf{S} \overset{*}{\Rightarrow} w$ met inductie over de lengte van w. In het basisgeval geldt $|w| = 0$. Er volgt $w = \varepsilon$, dit is zeker een palindroom en $\mathsf{S} \overset{*}{\Rightarrow} \varepsilon$ is ook waar. Stel nu dat $|w| = 1$, dan geldt $w = a$ of $w = b$, in beiden gevallen is de equivalentie ook waar. Dan nu het inductiegeval, waarin we aannemen dat $|w| = n + 2$. Dan is w te schrijven als $x\ v\ y$ voor zekere $x, y \in \Sigma$ en $v \in \Sigma^*$. Omdat w een palindroom is moet het gelden dat $x = y$ en dat v een palindroom is. Merk op dat $|v| = n$, dus met inductie weten we dat $\mathsf{S} \overset{*}{\Rightarrow} v$. Nu zien we ook $\mathsf{S} \Rightarrow x\ \mathsf{S}\ x \overset{*}{\Rightarrow} x\ v\ x = w$, wat het gewenste bewijst.

Door het bovenstaande argument weten we dus helemaal zeker dat de grammatica van (2) de taal der palindromen beschrijft.

Voorbeeld 2 (Palindromen zijn niet Regulier)**.** In het voorgaande Voorbeeld 1 zagen we dat de taal der palindromen een context-vrije taal is. We gebruiken nu de Myhill–Nerode stelling om aan te tonen dat deze taal *niet* regulier is.

Laten we kijken naar $\Sigma^*/\equiv$. Zodra we oneindig veel woorden $w_0, w_1, \ldots \in \Sigma^*$ hebben kunnen vinden zodanig dat $w_i \not\equiv w_j$ voor $i \neq j$ weten we dat $\Sigma^*/\equiv$ oneindig is. De Myhill–Nerode stelling vertelt ons dan dus dat de taal *niet* regulier is.

Bekijk de woorden a en aa. Zijn deze twee woorden equivalent onder $\equiv$? Zeker niet! Immers, $aba = aba$ is een palindroom, maar $aaba = aaba$ is dat niet. Op dezelfde manier zie je dat aa en aaa niet equivalent zijn, want $aabaa$ is een palindroom en $aaabaa$ niet. Zijn aaa en a dan equivalent misschien? Ook al niet, want $aaaba$ is geen palindroom en aba wel. In het algemeen, a^iba^j is enkel en alleen een palindroom als $i = j$, dus $a^i \equiv a^j$ kan enkel gelden wanneer $i = j$. Dit geeft ons een rijtje $w_i = a^i$ van niet-equivalente woorden, zoals omschreven in de vorige paragraaf. Dus zijn er oneindig veel equivalentieklassen, dus is de taal niet regulier.

Soms is het fijn om je grammatica in een zekere vorm te hebben staan. Sipser (2005) behandeld de Chomsky normaalvorm. Een andere normaalvorm is die van Greibach (1965), de Greibach normaalvorm. Deze is gegeven als volgt.

Definitie 3. Een CFG $\langle V, \Sigma, R, \mathsf{S} \rangle$ is in *Greibach normaalvorm* indien elke regel van de vorm $\mathsf{A} \to x\ \mathsf{B}_1 \ldots \mathsf{B}_n$ is voor $\mathsf{A} \in V$ en $\mathsf{B}_1, \ldots \mathsf{B}_n \in V - \{\mathsf{S}\}$ en $x \in \Sigma$, óf gelijk is aan $\mathsf{S} \to \varepsilon$.

Voorbeeld 3 (Oneven Palindromen)**.** De taal der oneven palindromen (palindromen van oneven lengte) kun je geven door de CFG in Greibach normaalvorm.

$$\begin{array}{lcl} \mathsf{S} & \to & a\ \mathsf{S}\ \mathsf{A} \mid b\ \mathsf{S}\ \mathsf{B} \mid a \mid b \\ \mathsf{A} & \to & a \\ \mathsf{B} & \to & b \end{array}$$

Voorbeeld 4 (Alle Palindromen)**.** De taal van alle palindromen kun je geven door de onderstaande CFG in Greibach Normaalvorm.

$$\begin{array}{lcl} \mathsf{S} & \to & \mathsf{C} \mid \varepsilon \\ \mathsf{C} & \to & a\ \mathsf{C}\ \mathsf{A} \mid b\ \mathsf{C}\ \mathsf{B} \mid a \mid b \mid a\ \mathsf{A} \mid \mathsf{B} \\ \mathsf{A} & \to & a \\ \mathsf{B} & \to & b \end{array}$$

Voorbeeld 5. De onderstaande CFG herkent de taal van woorden $w \in \{a, b\}^*$ waar precies één a meer in voorkomt dan er b's in voorkomen.

$$\begin{array}{lcl} \mathsf{S} & \to & \mathsf{E}\ a\ \mathsf{E} \\ \mathsf{E} & \to & a\ \mathsf{E}\ b \mid b\ \mathsf{E}\ a \mid \mathsf{E}\ \mathsf{E} \mid \varepsilon \end{array} \tag{3}$$

Het is vrij gemakkelijk te zien dat deze grammatica *niet* in Greibach normaalvorm staat. Maar dat is te verhelpen. Eerst zorgen we dat alles behalve het eerste teken rechts van de pijl een variabele is, dat geeft de volgende grammatica.

$$\begin{array}{lcl} \mathsf{S} & \to & \mathsf{E}\ \mathsf{A}\ \mathsf{E} \\ \mathsf{E} & \to & a\ \mathsf{E}\ \mathsf{B} \mid b\ \mathsf{E}\ \mathsf{A} \mid \mathsf{E}\ \mathsf{E} \mid \varepsilon \\ \mathsf{A} & \to & a \\ \mathsf{B} & \to & b \end{array}$$

Nu verwijderen we de lege regel $\mathsf{E} \to \varepsilon$, middels de op het vierde hoorcollege behandelde grammaticatransformatie. Dat geeft de volgende grammatica.

$$\begin{array}{lcl} \mathsf{S} & \to & \mathsf{E}\ \mathsf{A}\ \mathsf{E} \mid \mathsf{A}\ \mathsf{E} \mid \mathsf{E}\ \mathsf{A} \\ \mathsf{E} & \to & a\ \mathsf{E}\ \mathsf{B} \mid a\ \mathsf{B} \mid b\ \mathsf{E}\ \mathsf{A} \mid b\ \mathsf{A} \mid \mathsf{E}\ \mathsf{E} \mid \mathsf{E} \\ \mathsf{A} & \to & a \\ \mathsf{B} & \to & b \end{array}$$

Merk op dat de regel $\mathsf{E} \to \mathsf{E}$ niks doet, die kunnen we zonder meer verwijderen. De regel $\mathsf{E} \to \mathsf{E}\ \mathsf{E}$ is links-recursief, dat willen we niet. Die links-recursie halen we weg met een andere grammaticatransformatie, wat resulteert in het onderstaande.

$$\begin{array}{lcl} \mathsf{S} & \to & \mathsf{E}\ \mathsf{A}\ \mathsf{E} \mid \mathsf{A}\ \mathsf{E} \mid \mathsf{E}\ \mathsf{A} \\ \mathsf{E} & \to & a\ \mathsf{E}\ \mathsf{B} \mid a\ \mathsf{B} \mid b\ \mathsf{E}\ \mathsf{A} \mid b\ \mathsf{A} \\ \mathsf{E} & \to & a\ \mathsf{E}\ \mathsf{B}\ \underline{\mathsf{E}} \mid a\ \mathsf{B}\ \underline{\mathsf{E}} \mid b\ \mathsf{E}\ \mathsf{A}\ \underline{\mathsf{E}} \mid b\ \mathsf{A}\ \underline{\mathsf{E}} \\ \underline{\mathsf{E}} & \to & \mathsf{E} \mid \mathsf{E}\ \underline{\mathsf{E}} \\ \mathsf{A} & \to & a \\ \mathsf{B} & \to & b \end{array}$$

De regels voor Een S zijn nog niet van de propere vorm. Hier hoeven we enkel substitutie toe te passen. Dit is veel werk, maar mechanisch werk. Het resultaat is hieronder gegeven. Deze grammatica is in Greibach normaalvorm.

$$\begin{array}{lcl} \mathsf{S} & \to & a\ \mathsf{E}\ \mathsf{B}\ \mathsf{A}\ \mathsf{E} \mid a\ \mathsf{B}\ \mathsf{A}\ \mathsf{E} \mid b\ \mathsf{E}\ \mathsf{A}\ \mathsf{A}\ \mathsf{E} \mid b\ \mathsf{A}\ \mathsf{A}\ \mathsf{E} \mid a\ \mathsf{E}\ \mathsf{B}\ \underline{\mathsf{E}}\ \mathsf{A}\ \mathsf{E} \mid a\ \mathsf{B}\ \underline{\mathsf{E}}\ \mathsf{A}\ \mathsf{E} \mid b\ \mathsf{E}\ \mathsf{A}\ \underline{\mathsf{E}}\ \mathsf{A}\ \mathsf{E} \mid b\ \mathsf{A}\ \underline{\mathsf{E}}\ \mathsf{A}\ \mathsf{E} \\ \mathsf{S} & \to & a\ \mathsf{E} \\ \mathsf{S} & \to & a\ \mathsf{E}\ \mathsf{B}\ \mathsf{A} \mid a\ \mathsf{B}\ \mathsf{A} \mid b\ \mathsf{E}\ \mathsf{A}\ \mathsf{A} \mid b\ \mathsf{A}\ \mathsf{A} \mid a\ \mathsf{E}\ \mathsf{B}\ \underline{\mathsf{E}}\ \mathsf{A} \mid a\ \mathsf{B}\ \underline{\mathsf{E}}\ \mathsf{A} \mid b\ \mathsf{E}\ \mathsf{A}\ \underline{\mathsf{E}}\ \mathsf{A} \mid b\ \mathsf{A}\ \underline{\mathsf{E}}\ \mathsf{A} \\ \mathsf{E} & \to & a\ \mathsf{E}\ \mathsf{B} \mid a\ \mathsf{B} \mid b\ \mathsf{E}\ \mathsf{A} \mid b\ \mathsf{A} \mid a\ \mathsf{E}\ \mathsf{B}\ \underline{\mathsf{E}} \mid a\ \mathsf{B}\ \underline{\mathsf{E}} \mid b\ \mathsf{E}\ \mathsf{A}\ \underline{\mathsf{E}} \mid b\ \mathsf{A}\ \underline{\mathsf{E}} \\ \underline{\mathsf{E}} & \to & \mathsf{E} \\ \underline{\mathsf{E}} & \to & a\ \mathsf{E}\ \mathsf{B}\ \underline{\mathsf{E}} \mid a\ \mathsf{B}\ \underline{\mathsf{E}} \mid b\ \mathsf{E}\ \mathsf{A}\ \underline{\mathsf{E}} \mid b\ \mathsf{A}\ \underline{\mathsf{E}} \mid a\ \mathsf{E}\ \mathsf{B}\ \underline{\mathsf{E}}\ \underline{\mathsf{E}} \mid a\ \mathsf{B}\ \underline{\mathsf{E}}\ \underline{\mathsf{E}} \mid b\ \mathsf{E}\ \mathsf{A}\ \underline{\mathsf{E}}\ \underline{\mathsf{E}} \mid b\ \mathsf{A}\ \underline{\mathsf{E}}\ \underline{\mathsf{E}} \\ \mathsf{A} & \to & a \\ \mathsf{B} & \to & b \end{array}$$

Het vorige voorbeeld is bedoeld om aannemelijk te maken dat elke grammatica die te schrijven is in Greibach normaalvorm. De algemene truc lijkt op die om een grammatica in Chomsky normaalvorm te zetten. De stappen zijn ongeveer als volgt:

- Maak een nieuwe start-regel;
- Verwijder lege regels;
- Zorg ervoor dat iedere regel van de vorm $\mathsf{A} \to xw$ is, met $x \in \Sigma \cup V$ en $w \in V^*$ door variabelen te maken voor de tekens in het alfabet;
- Verwijder niet-nuttige regels;
- In een van-te-voren gekozen volgorde ga je de regels af. Per regel substitueer je steeds het eerste symbool, net zo lang tot alle regels links-recursief of in orde zijn. Verwijder dan links-recursie voor deze regel.

- Uiteindelijk moet er misschien nog wat gesubstitueerd worden in de nieuwe regels die je kreeg uit het verwijderen van links-recursie, en daarna werkt alles.

Als een grammatica al in Chomsky-normaalvorm staat, zijn de eerste drie stappen al gedaan. Daarom dat ook veel teksten de eerste drie stappen vervangen door "zet de grammatica in Chomsky normaalvorm".

In de volgende definitie geven we een soort van veralgemeniseerde DFA, die context-vrije talen kan herkennen. Om dit formeel aan te tonen zullen we kleine variaties maken op de PDA die 'm ogenschijnlijk sterker maken, maar die eigenlijk alleen onze bewijzen fijner maken.

Definitie 4 (Pushdown Automaat)**.** Een pushdown automaat (PDA) is een tupel $\langle\, Q, \Sigma, \Gamma, \delta, q_0, F \,\rangle$ met:

Q	eindige verzameling	*toestanden*
Σ	eindige verzameling	*alfabet*
Γ	eindige verzameling	*stack alfabet*
$\delta : Q \times \Sigma_\varepsilon \times \Gamma_\varepsilon \to \mathbf{P}\,(Q \times \Gamma_\varepsilon)$		*transitiefunctie*
$q_0 \in Q$		*begintoestand*
$F \subseteq Q$		*eindtoestanden*

Berekening is in het boek gedefinieerd als volgt.

Definitie 5 (Berekening)**.** Een *berekening* van $w \in \Sigma^*$ op $M = \langle\, Q, \Sigma, \Gamma, \delta, q_0, F \,\rangle$ bestaat uit

$$\underbrace{v_1 \dots v_n}_{\|}\ \in \Sigma_\varepsilon^* \qquad r_0, \dots, r_n \in Q \qquad u_0, \dots, u_n \in \Gamma^*$$
$$\|\qquad\qquad \|\qquad\qquad \|$$
$$w \qquad\qquad q_0 \qquad\qquad \varepsilon$$

zodat $\langle\, r_{i+1}, y \,\rangle \in \delta\big(r_i, v_{i+1}, x\big)$ voor $u_i = xu$ en $u_{i+1} = yu$ met $x, y \in \Gamma_\varepsilon$ en $u \in \Gamma^*$. Dit *accepteert* als $r_n \in F$. De *taal* van M is nu gegeven door

$$\mathcal{L}\,(\,M\,) := \big\{ w \in \Sigma^* \;\big|\; M \text{ heeft een accepterende berekening voor } w \big\}$$

In het vijfde hoorcollege bekeken we een relatie $\vdash$ op $Q \times \Sigma^* \times \Gamma^*$ om handig te kunnen spreken over berekeningen. De regel is dat áls $\langle\, r, y \,\rangle \in \delta(q, z, x)$, dán geldt $q, zw, xv \vdash r, w, yv$, natuurlijk voor $q, r \in Q$, $x, y \in \Gamma_\varepsilon$, $z \in \Sigma_\varepsilon$, $w \in \Sigma^*$ en $v \in \Gamma^*$.

We kunnen nu kijken naar $\overset{*}{\vdash}$, de reflexief-transitieve afsluiting van $\vdash$. Het onderstaande lemma bindt deze relatie strak vast aan de notie van berekening.

Lemma 1. *Een woord w wordt geaccepteerd door $M = \langle\, Q, \Sigma, \Gamma, \delta, q_0, F \,\rangle$ precies als $q_0, w, \varepsilon \overset{*}{\vdash} r, \varepsilon, u$ met $r \in F$ voor een $u \in \Gamma^*$.*

Bewijsidee. Stel dat er een berekening is voor w, oftewel rijtjes $v_1, \dots, v_n \in \Sigma_\varepsilon, r_0, \dots, r_n \in Q$ en $u_0, \dots, u_n \in \Gamma^*$ die voldoen aan alle eisen van Lemma 5. Nu is het niet lastig om na te gaan dat

$$r_i, v_i v_{i+1} \dots v_n, u_i \vdash r_{i+1}, v_{i+1} v_{i+2} \dots v_n, u_{i+1}$$

geldt voor elke $0 \leq i < n$. Dit toont dus aan dat $q_0, w, \varepsilon \overset{*}{\vdash} r_n, \varepsilon, u_n$. Als de oorspronkelijke berekening accepterend was, dan geldt $r_n \in F$. Dit is precies wat we wilde laten zien.

De andere kant op kun je ook vinden door de definities een beetje uit te pluizen. □

In Lemma 1 zie je heel duidelijk dat er nog allemaal rommel mag staan op de stack van een PDA als 'ie klaar is met rekenen. Het is fijner voor onze bewijzen als PDA's wat netter zouden zijn. Hiertoe de volgende definitie.

Definitie 6 (Nette PDA)**.** Een *nette PDA* is hetzelfde als een gewone PDA $M = \langle\, Q, \Sigma, \Gamma, \delta, q_0, F \,\rangle$, alleen zeggen we nu dat een woord $w \in \Sigma^*$ alleen geaccepteerd wordt indien $q_0, w, \varepsilon \overset{*}{\vdash} r, \varepsilon, \varepsilon$ voor een $r \in F$.

Natuurlijk wordt iedere taal die herkent wordt door een gewone (slordige) PDA ook herkent door een nette PDA. Immers, gegeven een gewone PDA kun je vanuit iedere eind-toestand ε pijlen toevoegen naar een nieuwe toestand $q_{\text{schoonmaak}}$. Deze toestand wordt de nieuwe accepterende toestand, en stel in dat $\delta(q_{\text{schoonmaak}}, \varepsilon, x) = \{\langle\, q_{\text{schoonmaak}}, \varepsilon \,\rangle\}$ voor elke $x \in \Gamma$. Wanneer je deze nieuwe automaat ziet als een nette PDA, dan is het duidelijk dat ze dezelfde woorden accepteert als de originele machine. De andere kant op is wat lastiger. Sipser (2005, pagina 113) noemt dit ook, en een oplossing wordt gesuggereerd.

Als je een nette PDA M hebt, dan kun je daar een gewone PDA bij maken met dezelfde taal op een vrij eenvoudige manier. De truc zit 'm er in dat je helemaal in het begin een markering op de stack kan plaatsen (een nieuw, dus verder ongebruikt, symbool). Verder maak je een nieuwe accepterende toestanden, waar je kan komen vanuit een oude accepterende toestand door de eindmarkering van de stack te eten. Er kan *nooit* iets na die markering op de stack staan, dus de stack moet altijd leeg zijn wanneer de nieuwe machine accepteert. Dit maakt 'r een nette PDA. Het wordt van harte aanbevolen dit voor jezelf formeel uit te werken.

Het is soms ook handig om in een transitie meerdere dingen op de stack te pushen. Hiertoe de volgende variatie op PDA's.

Definitie 7 (Lange Pushdown Automaat)**.** Een lange pushdown automaat (lange PDA) is een tupel $\langle\, Q, \Sigma, \Gamma, \delta, q_0, F \,\rangle$ met:

Q	eindige verzameling	*toestanden*
Σ	eindige verzameling	*alfabet*
Γ	eindige verzameling	*stack alfabet*
$\delta : Q \times \Sigma_\varepsilon \times \Gamma_\varepsilon \to \mathbf{P}\,(Q \times \Gamma^*)$		*transitiefunctie*
$q_0 \in Q$		*begintoestand*
$F \subseteq Q$		*eindtoestanden*

Om te definiëren welke woorden geaccepteerd worden gaan we direct naar een veralgemenisering van de relatie $\vdash$ op $Q \times \Sigma^* \times \Gamma^*$ die aanwezig was bij PDA's. Voor de handzaamheid gebruiken we precies dezelfde naam, maar nu is de relatie gedefiniëerd als:

$$\text{als } \langle\, r, u \,\rangle \in \delta(q, z, x), \text{ dan geldt } q, zw, xv \vdash r, w, uv$$

voor $q, r \in Q$, $x \in \Gamma_\varepsilon$, $z \in \Sigma_\varepsilon$, $w \in \Sigma^*$ en $u, v \in \Gamma^*$. Het verschil is dus dat de $y \in \Gamma_\varepsilon$ veralgemeniseerd is naar een $u \in \Gamma^*$. Hier is het niet lastig om te zien dat een gewone PDA ook een lange PDA is. Immers, $y \in \Gamma_\varepsilon$ kun je net zo goed zien als een element van Γ^*. De andere kant is iets lastiger, maar wordt toegelicht door Sipser (ibid., pagina 117). De truc is om de "lange" transitie op te breken in de "kleine" transities die we al hadden.

Het leuke is dat we het idee van lange- en nette PDA's kunnen combineren, en zodoende werken met een lange en nette PDA. Dit soort automaten kan wegens het bovenstaande exact hetzelfde als een gewone PDA, maar het is veel fijner om met ze te werken. Lees met dit in je achterhoofd Sipser (ibid., Lemma 2.21) eens.

Referenties

Greibach, Sheila A. (jan 1965). „A New Normal-Form Theorem for Context-Free Phrase Structure Grammars". In: *Journal of the ACM* 12 (1), p. 42–52. ISSN: 0004-5411. DOI: 10.1145/321250.321254.
Sipser, Michael (2005). *Introduction to the theory of computation.* Course Technology. ISBN: 0-619-21764-2.