Mitschrift der Vorlesung vom 27.10.2000

Jens R. Calamé (Matr.-Nr. 702 133)
calame@cs.uni-potsdam.de

6. November 2000

Zusammenfassung

Die Vorlesung vom 27.10.2000 beschäftigte sich mit dem Beweis, dass zu jeder kontextfreien Grammatik ein Kellerautomat konstruiert werden kann.

Inhaltsverzeichnis

Rückschau

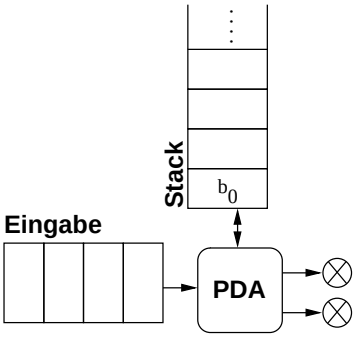


Abbildung 1: Die Komponenten um den PDA.

Die vorangegangene Vorlesung behandelte unter anderem den *Push-Down-Akzeptor* (Kellerautomat, (D/N)PDA). Er wurde definiert als ein „endlichen Akzeptor, ergänzt um einen endlichen Speicher in Stackform". Das Stackalphabet eines solchen Akzeptors wird Γ genannt, auf dem Stack selbst wird immer zuunterst das „Bottom-Symbol" b_0 gespeichert. Ein Push-Down-Akzeptor muss *nicht bei jedem Arbeitsschritt* Symbole aus der Eingabe lesen, vielmehr kann er auch das Symbol ε als Eingabe annehmen.

Der Akzeptor selbst ist definiert über das Sieben-Tupel $\varrho = (\mathcal{X}, \Gamma, \mathcal{S}, \delta, s_0, b_0, \mathcal{F})$ mit dem Eingabealphabet $\mathcal{X}$, dem Stackalphabet Γ mit Bottom-Symbol b_0, der Zustandsmenge $\mathcal{S}$ mit Startzustand s_0 sowie der Endzustandsmenge $\mathcal{F}$.

Der nun folgende Teil beschäftigt sich speziell mit nichtdeterministischen Push-Down-Akzeptoren (NPDAs) und der Klasse der von ihnen erkannten Sprachen, L_{NPDA}.

5 PDAs ↔ kontextfreie Sprachen

Satz. *Die Klasse der von nichtdeterministischen Kellerautomaten akzeptierten Sprachen ist gleich der Klasse der kontextfreien Sprachen.[4]*

Um diesen Satz zu beweisen, sind zwei Richtungen zu betrachten:

1. Es kann zu jeder kontextfreien Sprache ein NPDA konstruiert werden.
2. Es kann zu jedem NPDA eine kontextfreie Grammatik konstruiert werden.

Diese Vorlesung wird den ersten Punkt beweisen. Dazu wird erst einmal ein NPDA für kontextfreie Sprachen entwickelt und dann seine Funktionsweise anhand eines Beispiels verdeutlicht. Zum Schluss wird allgemein bewiesen, dass der Akzeptor jedes Wort einer kontextfreien Sprache in endlich vielen Schritten (Endziel sind „unendlich" viele) aus dem Axiom ableiten kann. Zuletzt muss noch bewiesen werden, dass auch die Sprache des NPDA tatsächlich gleich der Sprache der kontextfreien Grammatik ist.

5.5 Konstruktion von PDAs zu kontextfreien Sprachen

Lemma N. *Zu jeder kontextfreien Grammatik $G = (\mathcal{N}, \mathcal{T}, \mathcal{P}, \sigma)$ existiert ein $\varrho = (\mathcal{T}, \Gamma, \mathcal{S}, \delta, s_0, b_0, \mathcal{F})$ mit $L(G) = L(\varrho)$, das heißt CF $\subseteq L_{NPDA}$.*

5.5.1 Konstruktion eines allgemeinen PDA für kontextfreie Sprachen

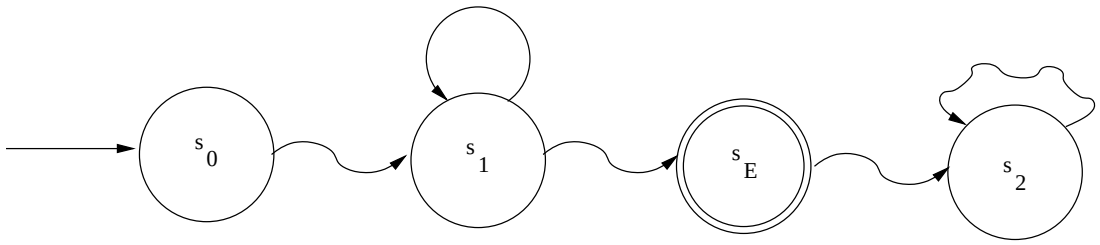


Abbildung 2: Allgemeines Modell eines nichtdeterministischen Akzeptors (Push-Down-Automaten) für kontextfreie Sprachen.

Zur Veranschaulichung soll zunächst der folgende, in Abbildung 2 dargestellte, nichtdeterministische Kellerautomat konstruiert werden. Im Groben besteht er aus vier Zuständen: Dem Anfangszustand, dem Zustand s_1, in dem er die Eingabe verarbeitet, dem Endzustand s_E, in den er zu gegebenem Zeitpunkt wechselt, sowie einem absorbierenden Zustand s_2, den er aufsucht, wenn das eingelesene Wort nicht der gegebenen Sprache entspricht. Die exakte Definition dieses Push-Down-Akzeptors sieht so aus: Akzeptor $\varrho = (\mathcal{T}, \Gamma, \mathcal{S}, \delta, s_0, b_0, \mathcal{F})$ mit

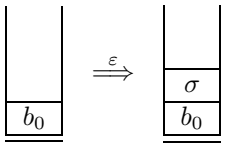
- $\mathcal{T}$ als Menge der *Terminalsymbole* (der zu akzeptierenden Sprache),
- $\mathbf{\Gamma} := b_0 \cup \mathcal{N} \cup \mathcal{T}$ als *Kelleralphabet* ($\mathcal{N}$ ist die Menge der Nonterminals der zu akzeptierenden Sprache),
- $\mathcal{S} := \{s_0, s_1, s_2, s_E\}$ als *Zustandsmenge*,
- $\delta : \mathcal{S} \times (\mathcal{N} \cup \mathcal{T}) \times \Gamma \rightarrow 2^{\mathcal{S} \times \Gamma}$ als noch zu definierende *Zustandsübergangsfunktion*,
- $\mathbf{s_0}$ als *Startzustand*,
- $\mathbf{b_0}$ als *Bottomsymbol* sowie
- $\mathcal{F} := \{s_E\}$ als Menge der *finalen Zustände*.

Die Zustandsübergangsfunktion werde wie folgt definiert:

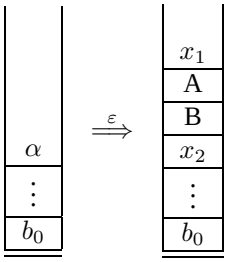
(1) $\delta(s_0, \varepsilon, b_0) = \{(s_1, b_0, \sigma)\}$
(2) $\delta(s_1, \varepsilon, \alpha) = \{(s_1, v^R) | \text{für alle } v \in \mathcal{P} \text{ mit } \alpha \to v, \text{ für alle } \alpha \in \mathcal{N}\}$
(3) $\delta(s_1, \varepsilon, b_0) = \{(s_E, b_0)\}$
(4)[1] $\delta(s_1, t, t) = \{(s_1, \varepsilon)\}$
(5) $\delta(s_E, t, b_0) = \{(s_2, b_0)\}$

Zur Erklärung der Zustandsübergänge:

1. Als erstes, noch bevor irgendein Symbol von der Eingabe gelesen wird, legt der Push-Down-Akzeptor das Axiom der Grammatik der zu akzeptierenden Sprache auf dem Stack oberhalb des Bottom-Symbols ab. Dieses Symbol ist die Keimzelle des Ableitungs- und damit des Akzeptanzprozesses. Bildlich dargestellt, sieht dieser Schritt so aus:

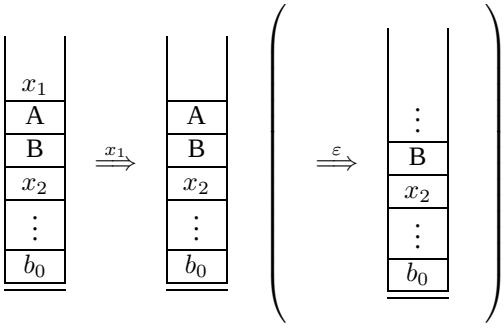


2. Als zweites wird die Substitution eines Nonterminals auf dem Stack durch eine Zeichenkette definiert. Dieser Vorgang wird durch die Regeln der zugrundeliegenden Grammatik bestimmt, wobei hier der Nichtdeterminismus dieses Push-Down-Akzeptors greift: Stehen mehrere Regeln bei einem Ableitungsschritt zur Auswahl, *kann* der Akzeptor theoretisch genau die richtige auswählen, um das Wort auf der Eingabe zu erkennen.
   Es ist definiert, dass das Nonterminal α auf dem Stack durch eine Zeichenkette v^R ersetzt wird, wenn die entsprechende Regel in der Grammatik lautet: $\alpha \to v$. v^R ist die gespiegelte Variante der Zeichenkette v. Gespiegelt deshalb, weil ein Stack einen *LIFO*-Speicher darstellt, von dem gespeicherte Werte nur in umgekehrter Reihenfolge wieder gelesen werden können.
   Nun wieder die bildliche Veranschaulichung des Vorgangs am Beispiel $\alpha \to x_1 A B x_2$:



3. Als drittes wird nur definiert, dass der Akzeptor, sobald er nichts von der Eingabe liest, sich nicht im Anfangszustand befindet *und* als oberstes Symbol auf dem Stack das Bottom-Symbol liegt, in den Endzustand wechselt.

4. Als vorletztes wird definiert, was zu geschehen hat, wenn ein Terminalsymbol von Eingabe gelesen wird, das exakt dem obersten Symbol auf dem Stack entspricht. In diesem Fall werden Eingabe- und Kellersymbol *gematcht*, d.h. sowohl auf der Eingabe (durch das Einlesen), als auch auf dem Stack gelöscht:



Auf diese Weise wird der Stack abgebaut, bis einer von folgenden drei Schlusszuständen erreicht ist:

[1] Das „t" steht für „Terminalsymbol".

(a) Die Eingabe ist komplett eingelesen und der Stack ist leer: Das Wort gehörte zur Sprache; der Akzeptor wechselt in den Endzustand s_E (Regel (3)).

(b) Die Eingabe ist komplett eingelesen, aber der Stack nicht leer: Das Wort gehört nicht zur Sprache; der Akzeptor endet im Nichtendzustand s_1.

(c) Die Eingabe ist noch nicht komplett eingelesen, der Stack hingegen leer: Das Wort gehört nicht zur Sprache; der Akzeptor wechselt über den Endzustand in den absorbierenden Zustand s_2 (Regeln (3) und (5)).

Der geklammerte Schritt ist übrigens die weitere Ableitung nach Regel (2), die immer dann stattfindet, wenn nach dem Matchen ein Nonterminal zuoberst auf dem Keller liegt. Andernfalls wird weitergematcht, bis ein Nonterminal auftritt oder der Stack leer ist.

5. Die letzte Regel besagt, dass der Akzeptor, wenn er mit leerem Stack im Endzustand steht, er aber noch Zeichen aus der Eingabe liest, in den absorbierenden Zustand s_2 wechselt, die Eingabe zu Ende einliest und danach endet.

5.5.2 Ein konkreter Anwendungsfall

Es wird nun ein konkretes Beispiel gegeben. Gegeben sei die Sprache $L(G)$ mit der Grammatik $G = (\mathcal{N}, \mathcal{T}, \mathcal{P}, \sigma)$, die wie folgt definiert ist:

- $\mathcal{N} := \{\sigma\}$ als Menge der Nonterminals,
- $\mathcal{T} := \{a, b, c\}$ als Menge der Terminals,
- $\mathcal{P} := \{\sigma \to a\sigma c, \sigma \to b\sigma c, \sigma \to \varepsilon\}$ als Menge der Produktionregeln und
- σ als Axiom der Grammatik.

Diese Grammatik erzeugt Wörter, deren erste Hälfte aus einer beliebigen Folge aus a's und b's und deren zweite Hälfte nur aus c's besteht; kurz: $L(G) = \{wc^{|w|} | w \in \{a, b\}^*\}$. Der PDA ist definiert wie in Abschnitt 5.5.1, wobei die δ-Tabelle an dieser Stelle genau angegeben ist[2]:

	von				**nach**	
Regel	$\mathcal{S}$	$X \cup \{\varepsilon\}$	Γ	$\Rightarrow$	$\mathcal{S}$	Γ^*
(1)	s_0	ε	b_0	$\Rightarrow$	s_1	$b_0\sigma$
(2)	s_1	ε	σ	$\Rightarrow$	s_1	$c\sigma a$
	s_1	ε	σ	$\Rightarrow$	s_1	$c\sigma b$
	s_1	ε	σ	$\Rightarrow$	s_1	ε
(3)	s_1	a	a	$\Rightarrow$	s_1	ε
	s_1	b	b	$\Rightarrow$	s_1	ε
	s_1	c	c	$\Rightarrow$	s_1	ε
(4)	s_E	a	b_0	$\Rightarrow$	s_2	b_0
	s_E	b	b_0	$\Rightarrow$	s_2	b_0
	s_E	c	b_0	$\Rightarrow$	s_2	b_0
(5)	s_1	ε	b_0	$\Rightarrow$	s_E	b_0

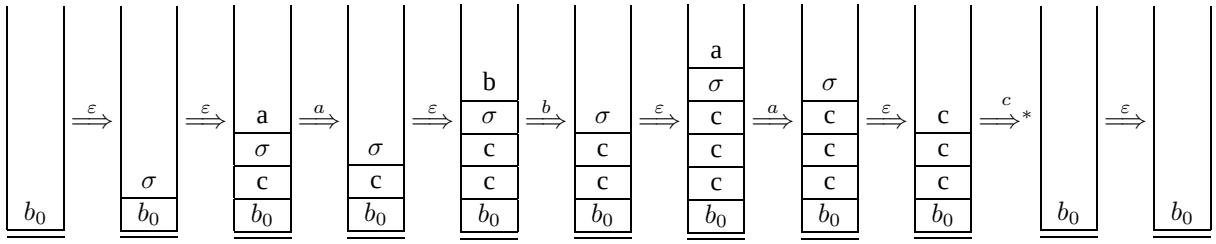
Nun wird das Wort „**abaccc**", das den Bedingungen der Sprache genüge tut, dem Akzeptor eingegeben. Im Folgenden werden die Konfigurationen während der Abarbeitung sowie die bei den Konfigurationsübergängen verwendeten Regeln

[2] In Spalte 5 ist in dieser Tabelle das Kellerwort, d.h. die Änderung auf dem Stack bei der gegebenen Konfiguration, angegeben. Das *Stack-Top* ist das am weitesten rechts stehende Symbol. Dieses wird geprüft und durch das neue Kellerwort ersetzt.

(siehe Tabelle oben und Seite 2) angegeben:

$$\Longrightarrow(s_0,\ abaccc,\ b_0) \tag{1}$$
$$\xrightarrow{(1)}(s_1,\ abaccc,\ b_0\sigma) \tag{2}$$
$$\xrightarrow{(2)}(s_1,\ abaccc,\ b_0c\sigma a) \tag{3}$$
$$\xrightarrow{(3)}(s_1,\ baccc,\ b_0c\sigma) \tag{4}$$
$$\xrightarrow{(2)}(s_1,\ abaccc,\ b_0cc\sigma b) \tag{5}$$
$$\xrightarrow{(3)}(s_1,\ accc,\ b_0cc\sigma) \tag{6}$$
$$\xrightarrow{(2)}(s_1,\ accc,\ b_0ccc\sigma a) \tag{7}$$
$$\xrightarrow{(3)}(s_1,\ ccc,\ b_0ccc\sigma) \tag{8}$$
$$\xrightarrow{(2)}(s_1,\ ccc,\ b_0ccc) \tag{9}$$
$$\xrightarrow{(3)}{}^{*}(s_1,\ \varepsilon,\ b_0) \tag{10}$$
$$\xrightarrow{(5)}(s_E,\ \varepsilon,\ b_0) \Rightarrow \boxed{\text{akzeptiert}} \tag{11}$$

Zur Veranschaulichung noch einmal die Entwicklung auf dem Stack während der Worterkennung:



5.5.3 Der allgemeine Beweis

Um exakt zu beweisen, dass $L(G) = L(\varrho)$, zeigen wir folgende Eigenschaft von ϱ:

Behauptung 1. *Es gibt ein* $k \in \mathbb{N}$ *mit* $(s_1, w', b_0z^R) \in \hat{\delta}^k(s_0, uw', b_0)$ *für* $u,\ w' \in \mathcal{T}^*,\ z \in (\mathcal{N} \cup \mathcal{T})^*$.

Diese Behauptung ist äquivalent zu folgender Aussage:
„Es gibt eine Linksableitung bezüglich $\sigma \rightarrow_G^* uz$ mit $u \in \mathcal{T}^*,\ z \in (\mathcal{N} \cup \mathcal{T})^*$."[3]

Beweis. **„⇒"** Der Beweis in dieser Richtung behandelt die Berechnungsgänge des Akzeptors und läuft über vollständige Induktion über k.

Induktionsanfang: $k = 1$: $u = \varepsilon \wedge z^R = \sigma = z$ wegen Regel (1) der δ-Funktion $\Rightarrow\ \sigma \rightarrow^* \sigma$ ✓

Induktionsvoraussetzung: Sei die Behauptung bewiesen für k. Man betrachte nun:

$$(s_1, w', b_0z^R) \in \hat{\delta}^{k+1}(s_0, uw', b_0)$$

Dann gibt es ein $(s_1, xw', b_0z_1^R) \in \hat{\delta}^k(s_o, uw', b_0)$ mit $x \in \mathcal{T} \cup \{\varepsilon\}$.

Fallunterscheidung:

Fall 1. $x \in \mathcal{T}$. Dann muss im letzten $\hat{\delta}$-Schritt $\left[(s_1, xw', b_0z_1^R) \xrightarrow{\hat{\delta}} (s_1, w', b_0z^R)\right]$ Regel (4) angewandt worden sein und daher gilt:

$$(xz)^R = z^R \cdot x = z_1^R.$$

Nach Induktionsannahme gibt es eine Ableitung $\sigma \rightarrow_L^* u'z_1$ (mit $u = u'x$). Aus $z_1^R = (xz)^R$ folgt $z_1 = xz$, also $u'z_1 = u'xz = uz$. Also existiert eine Linksableitung $\sigma \rightarrow_L^* uz$ und die Induktionsbehauptung gilt auch für $k + 1$. ✓

[3] Das anvisierte Ziel ist hierbei $(s_1, \varepsilon, b_0) \in \hat{\delta}^{\text{„}\infty\text{"}}(s_0, uw', b_0)$ mit $\sigma \rightarrow_G^* uw'$.

Fall 2. $x = \varepsilon$. Dann muss im letzten $\hat{\delta}$-Schritt die Überführung (2) angewandt worden sein. Daher gibt es ein $\alpha \to v \in \mathcal{P}$ mit $z_1^R = z'^R_1\alpha$ und $z^R = z'^R_1 v^R = (vz'_1)^R$. Nach Induktionsannahme existiert eine Linksableitung $\sigma \to_L^* uz_1 = u\alpha z'_1$. Folglich gibt es einen Linksableitungsschritt $\sigma \to_L^* u\alpha z'_1 \to uvz'_1 = uz$.

Damit gilt die Induktionsbehauptung für alle k.

„⇐" Der Beweis in dieser Richtung läuft über vollständige Induktion über die Länge der Linksableitungen.

Induktionsanfang: Länge 0. $uz = \sigma,\ \sigma \to^* \sigma$ als Linksableitung:

$$(s_1, uw, b_0\sigma) \in \hat{\delta}^*(s_0, uw, b_0)\ \checkmark$$

Induktionsvoraussetzung: Sei die Richtung „⇐" der Behauptung 1 für alle Linksableitungen bis zur Länge j bewiesen.

Man betrachte die Ableitung der Länge $j + 1$:

$\sigma \to_L^j u'z' \underbrace{=}_{1)} u'u_1\alpha z_1 \underbrace{\to}_{2)} u'u_1vz_1 = uz,\ u', u_1 \in \mathcal{T}^*,\ u = u'u_1,\ \alpha \in \mathcal{N},\ \alpha \to w \in \mathcal{P},\ z_1 \in \Gamma^*,\ z = vz_1.$

Anmerkung zu 1. Dies kann man mit dem Akzeptor ϱ nachmodellieren.

Anmerkung zu 2. Hier stellt sich die Frage, ob der Automat diesen Ableitungsschritt mittels seiner Regeln (1) bis (5) nachvollziehen kann.

Nach Induktionsannahme gilt: Es existiert ein k mit $(s_1,\ u_1w',\ b_0z'^R) = (s_1,\ u_1w',\ b_0z_1^R\alpha u_1^R) \in \hat{\delta}^k(s_0,\ uw',\ b_0)$. Wendet man $|u_1|$-mal die Regel (4) an, so folgt $(s_1,\ w',\ b_0z_1^R\alpha) \in \hat{\delta}^{k+|u_1|}(s_0,\ uw',\ b_0)$. Man wende nun Regel (2) an:

$$(s_1,\ w',\ b_0z_1\alpha) \in \hat{\delta}^{k+|u_1|+1}(s_0,\ uw',\ b_0).$$

Damit gilt die Induktionsbehauptung für $j + 1$ und damit für alle j.

Behauptung 2. $L(\varrho) = L(G)$.

„⊆" Sei $w \in L(\varrho)$, das heißt $(s_E,\ \varepsilon,\ b_0) \in \hat{\delta}^i(s_0,\ w,\ b_0)$ mit $i \in \mathbb{N}$. Wegen der Konstruktion von ϱ muss aber $(s_1,\ \varepsilon,\ b_0) \in \hat{\delta}^{i-1}(s_0,\ w,\ b_0)$ sein. Wegen Behauptung 1 gibt es dann eine Linksableitung

$$\sigma \to_L^* w \Rightarrow w \in L(G).$$

„⊇" Sei $w \in L(G)$. Dann existiert $\sigma \to_L^* w$. Wegen Behauptung 1 folgt: $(s_1,\ \varepsilon,\ b_0) \in \hat{\delta}^k(s_0,\ w,\ b_0)$ für ein $k \in \mathbb{N}$. Wegen Regel (3) gilt:

$$(s_E,\ \varepsilon,\ b_0) \in \hat{\delta}^{k+1}(s_0,\ w,\ b_0),$$

woraus $w \in L(\varrho)$ folgt. □

5.5.4 Zusammenfassung/Literatur

Es wurde also bewiesen, dass zu jeder kontextfreien Grammatik ein NPDA konstruiert werden kann. Diesen Beweis findet man in der Literatur auch in alternativer Form. Als ein meiner Ansicht nach gelungenes, weil einleuchtendes Beispiel, sei an dieser Stelle der Beweis von Schöning gezeigt.

Schöning Der Beweis in [3] läuft nicht über die vollständige Induktion, sondern lediglich über die Angabe eines geeigneten Akzeptors:

Beweis. Der Akzeptor wird ähnlich definiert wie im Beispiel ((Abschnitt 5.5.1), weicht an einigen Stellen allerdings ab. Ich habe einige Anpassungen des Originals an dieses Skript vorgenommen, jedoch den Grundgedanken nicht verändert.

Es sei der Akzeptor $\varrho = (\mathcal{T}, \mathcal{N} \cup \mathcal{T} \cup \{\varepsilon\}, \{z\}, \delta, z, \varepsilon, z)$ definiert. Beim Start wird das Axiom σ der Grammatik als unterstes Kellersymbol gespeichert. Es wird nun die Zustandsübergangsfunktion definiert:

- $(z, \alpha) \in \delta(z, \varepsilon, \alpha) \forall A \to a \in \mathcal{P}$ mit $\alpha \in (\mathcal{N} \cup \mathcal{T})^*$, sowie

- $(z, \varepsilon) \in \delta(z, a, a)$.

Immer, wenn das Stack-Top eine Grammatikregel darstellt, wird diese, ohne vom Eingabeband zu lesen, aufgelöst; ist das Stack-Top ein Terminal, das mit dem aktuellen der Eingabe übereinstimmt, wird es vom Stack entfernt (sollte aus dem o.g. Beispiel bekannt vorkommen). Das Wort gilt als akzeptiert, wenn der Stack nach dem Einlesen der Eingabe leer ist.

Es gilt nun für alle Terminals $x \in T^*$:

$$\begin{aligned} x \in L(G) \quad & g.d.w. \quad \text{eine Ableitung in } G\text{existiert mit } \sigma \Rightarrow \ldots \Rightarrow x \\ & g.d.w. \quad \text{eine Folge von Konfigurationen } (z, x, \sigma) \rightarrow^k (z, \varepsilon, \varepsilon), k \in \mathbb{N} \text{ existieren} \\ & g.d.w. \quad x \in L(\varrho) \end{aligned}$$

□

Dies scheint Schöning bereits als Beweis zu genügen. Die übrigen Beweise in [1], [2] und [4] verzichten allerdings nicht auf die vollständige Induktion.

Vorausschau auf die folgenden Veranstaltungen

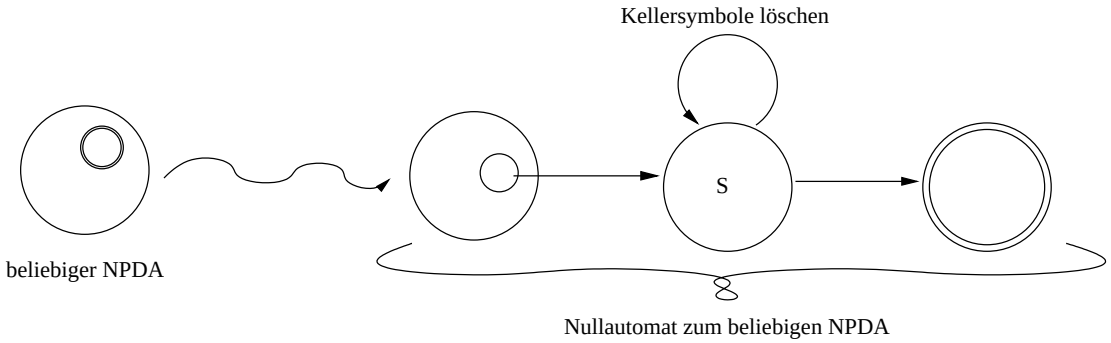


Abbildung 3: Konstruktion eines Null-Automaten aus einem beliebigen NPDA.

Nachdem hier bewiesen wurde, dass zu jeder beliebigen kontextfreien Grammatik ein Push-Down-Akzeptor konstruiert werden kann, werden sich die nächsten Veranstaltungen mit dem Beweis in umgekehrter Richtung zu beschäftigen haben. Erst wenn auch nachgewiesen ist, dass zu jedem PDA eine kontextfreie Grammatik konstruiert werden kann, kann man die Aussage, dass die Klasse der kontextfreien Sprachen gleich der Klasse der von PDAs akzeptierten Sprachen ist, als wahr ansehen.

Die Beweisidee ist folgende: Es ist ein „sauberer Kellerautomat" *(Null-Automat)* zu konstruieren. Diese besondere Variante eines PDA terminiert mit grundsätzlich komplett leerem Stack. Dies erreicht man, indem man einen beliebigen (N)PDA statt sofort in den End- zunächst in einen Zwischenzustand wechseln lässt. Hier lässt man ihn nun den Keller definitiv leerräumen, um ihn dann im (neuen) Endzustand terminieren zu lassen (siehe Abbildung 3).

Ein Null-Automat lässt sich auf die in der Abbildung dargestellte Weise aus jedem NPDA konstruieren. Für die Konstruktion einer Grammatik G zu den NPDAs ϱ mit $L(G) = L(\varrho)$ beschränkt man sich nun nur noch auf diese Akzeptoren.

Literatur

[1] ERK, KATRIN und LUTZ PRIESE: *Theoretische Informatik/Eine umfassende Einführung*, Kapitel 6.6 (Push-Down-Automaten (PDA)), Seiten 133–144. Springer, Berlin, Heidelberg, New York, erste Auflage, 2000.

[2] HOPCROFT, J. E. und J. D. ULLMAN: *Einführung in die Automatentheorie, Formale Sprachen und Komplexitätstheorie*, Kapitel 5.3 (Kellerautomaten und kontextfreie Sprachen), Seiten 121–132. Oldenbourg, München, Wien, vierte Auflage, 2000.

[3] SCHÖNING, UWE: *Theoretische Informatik - kurzgefaßt*, Kapitel 1.3.5 (Kellerautomaten), Seiten 64–72. Spektrum Akademischer Verlag, Heidelberg, Berlin, Oxford, zweite Auflage, 1995.

[4] WEGENER, INGO: *Theoretische Informatik - eine algorithmische Einführung*, Kapitel 7.3 (Kellerautomaten und kontextfreie Sprachen), Seiten 188–192. B. G. Teubner, Stuttgart, Leipzig, zweite Auflage, 1999.