Von Pushdown-Automaten zu kontextfreien Grammatiken

Niklas Rieken

28. Juni 2018

Die folgende Definition aus der Vorlesung beschreibt, wie sich aus einem Pushdown-Automaten (PDA), der mit leerem Stack akzeptiert, eine kontextfreie Grammatik (CFG) konstruieren lässt, die die selbe Sprache erzeugt.

Definition. Sei $\mathcal{A} = (Q, \Sigma, \Gamma, \Delta, q_0, Z_0)$ ein PDA, der mit leerem Stack akzeptiert. Die kontextfreie Grammatik $\mathcal{G}_\mathcal{A} = (N, \Sigma, P, S)$ sei wie folgt definiert:

- S ist ein neues Startsymbol.
- Nichtterminale $N \coloneqq \{S\} \cup \{[pZq] \mid p, q \in Q, Z \in \Gamma\}$.
- Produktionsregeln

$$\begin{aligned} P \coloneqq \{&[pZq] \to \sigma[p_0 Z_1 p_1][p_1 Z_2 p_2] \dots [p_{m-1} Z_m q] \mid \\ &(p, \sigma, Z, Z_1 \dots Z_m, p_0) \in \Delta, p_1, \dots, p_{m-1}, q \in Q\} \\ \cup \{&S \to [q_0 Z_0 q] \mid q \in Q\}. \end{aligned}$$

Das Verfahren ist eigentlich sehr einfach, allerdings ist die Funktionsweise nicht sehr intuitiv, da sich die Produktionsregeln der erzeugten Grammatik nicht so leicht interpretieren lassen (oder: sie lassen sich nicht so einfach von links nach rechts lesen). Wir betrachten also einen PDA, der mit leerem Stack akzeptiert $\mathcal{A} = (Q, \Sigma, \Gamma, \Delta, q_0, Z_0)$ und konstruieren daraus die Grammatik $\mathcal{G}_\mathcal{A} = (N, \Sigma, P, S)$ wie in der Definition beschrieben. Die entscheidene Kernidee des Algorithmus ist folgende: Das Nichtterminalsymbol $[pZq]$ repräsentiert einen Lauf von p nach q bei dem das Stacksymbol Z aus dem Stack entfernt wird (d.h. wir kommen dem leeren Stack "einen Schritt näher").
Ein Ableitungsschritt in der Grammatik entspricht dabei nicht einfach einer Transition im PDA (außer im Fall von $[pZq] \to \sigma$, was der Transition $(p, \sigma, Z, \varepsilon, q)$ entspricht), sondern auch der *Hoffnung* den Stack, der sich in der Transition $(p, \sigma, Z, Z_1 \dots Z_m, p_0)$ aufbaut (für $m > 0$) wieder abzubauen. Die Nichtterminale $[pZq]$ *hoffen* also, dass der PDA von p mit Stackinhalt $Z\gamma$ nach q kommt und dabei nur γ auf dem Stack stehen lässt, wobei $\gamma \in \Gamma^*$.
Wir gehen nun einmal auf die drei Arten der Produktionsregeln ein, hierfür sei $\sigma \in \Sigma \cup \{\varepsilon\}$:

- Mit dem Startsymbol S haben wir folgende Produktionsregeln: $S \to [q_0 Z_0 q]$ **für alle** $q \in Q$.
  Die Akzeptanzbedingung des Automatenmodells ist allein vom Stack abhängig, in welchem Zustand ein Lauf terminiert ist also egal, deswegen bilden wir diese Regel für jedes q. Hier rufen wir uns nochmal die Intuition der Nichtterminale (außer S) in Erinnerung: Gesucht wird also ein Lauf auf $\mathcal{A}$ vom Startzustand q_0 zu einem beliebigen q, sodass das Bottom-of-Stack-Symbol Z_0 weggenommen wird (d.h. nur noch ε auf dem Stack steht). Diese Regeln sind drücken also genau unseren gewünschten Lauf aus: Von der Startkonfiguration (q_0, Z_0) zu einer Konfiguration (q, ε).

- Für die Transitionen der Form $(p, \sigma, Z, \varepsilon, q)$ erhalten wir die Produktionsregel $[pZq] \to \sigma$. Dies sind die Regeln in denen der Stack tatsächlich abgebaut wird ohne ihn vorher zu erweitern (s. nächster Punkt).

- Regeln der Form $[pZp_m] \to \sigma[p_0 Z_1 p_1][p1 Z_2 p_2] \dots [p_{m-1} Z_m p_m]$ für eine Transition $(p, \sigma, Z, Z_1 \dots Z_m, p_0) \in \Delta$ und alle möglichen(!) Zustände $p_1, \dots, p_m \in Q$:
  Dies ist die schwerste Art der konstruierten Produktionsregeln. Wir versuchen dies trotzdem einmal in einem Satz von links nach rechts zu beschreiben, auch wenn das etwas konstruiert wirkt: Wir befinden uns in Zustand p mit Z oben auf dem Stack, welches wir entfernen wollen, und dabei den Zustand p_m erreichen mit einer Transition über σ, wo wir den Stack mit $Z_1, \dots, Z_m$ auffüllen, welche wir dann in der Zukunft, d.h. in einem Lauf über p_0 bis p_m, abbauen müssen über weitere Regeln dieser Form und – schlussendlich – über Regeln der Art aus dem vorigen Punkt.
  Eine Ableitung eines Wortes repräsentiert also genau einen akzeptierenden (d.h. mit ε auf dem Stack am Ende) Lauf auf dem PDA. In Abbildung 1 ist die Idee nochmal an einer Transition veranschaulicht.

Um die Philosophie der Konstruktion einmal zusammenzufassen: Wenn in einem akzeptierenden Lauf der aktuelle Stackinhalt zu einem Zeitpunkt $Z\gamma$ ist, dann wird zu einem späteren Punkt des Laufes der Stackinhalt einmal nur noch γ sein.
Wir betrachten als Beispiel den PDA, der mit leerem Stack akzeptiert in Abbildung 2. Sei dafür

$$\mathcal{A} = (\{p, q\}, \{a, b\}, \{Z\}, \{(p, a, Z, ZZ, p), (p, a, Z, Z, q), (q, b, Z, \varepsilon, q)\}, p, Z).$$

Wir erhalten zunächst für die Grammatik $\mathcal{G}_\mathcal{A} = (N, \{a, b\}, P, S)$ mit $N = \{S, [pZp], [pZq], [qZp], [qZq]\}$. Die Produktionsregeln P konstruieren wir nacheinander:

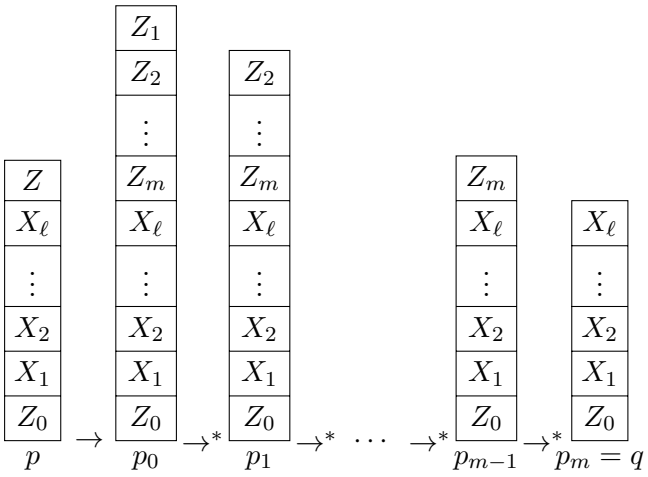

Abbildung 1: Ein Teillauf im PDA, welche durch das Nichtterminal $[pZq]$ simuliert wird. Zunächst benutzen wir die Transition $(p, \sigma, Z, Z_1 \dots Z_m, p_0)$ um Z zu entfernen, dadurch entstehen jedoch die Symbole $Z_1, \dots, Z_m$ neu auf dem Stack, die im weiteren Verlauf abgebaut werden müssen. Dabei werden ist es für $[pZq]$ *egal* mit welchen Zuständen $p_1, \dots, p_{m-1}$ dies erreicht wird. Daher gibt es für jede m-Auswahl aus Q eine Produktion. Beachte außerdem, dass die Schritte zwischen p_i zu p_{i+1} (für $0 \leq i < m$) *Makro-Schritte* ($\to^*$ – mehrere Transitionen) sind, d.h. nicht wie von p nach p_0 eine einzelne Transition (nur durch Transition $(p_i, \sigma, Z_{i+1}, \varepsilon, p_{i+1})$), sondern durch weitere Zwischenschritte mit zwischenzeitlichem Aufbau des Stacks.

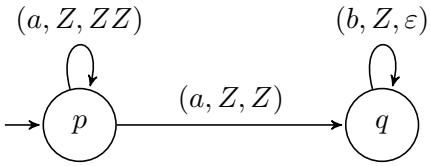

Abbildung 2: PDA, der mit leerem Stack akzeptiert für die Sprache $L = \{a^n b^n : n \in \mathbb{N}_+\}$.

- Zunächst ist es uns egal in welchem Zustand wir terminieren (auch wenn ein akzeptierender Lauf offensichtlich nur in q enden kann), die einzige Bedingung, die wir an den Lauf stellen ist, dass er im Startzustand p beginnt mit dem Bottom-of-Stack-Symbol Z oben auf dem Stack. Insgesamt erhalten wir also die Regeln

$$S \to [pZp] \mid [pZq] \in P.$$

- Nun betrachten wir die Transitionen, die den Stack abbauen. Davon gibt es nur eine im Loop über q, d.h. $(q, b, Z, \varepsilon, q)$. Wir erhalten also:

$$[qZq] \to b \in P.$$

- Zuletzt noch die Transitionen, die den Stack vorübergehend aufbauen. Für die Transition (p, a, Z, ZZ, p) erhalten wir die Regeln

$$\begin{aligned}[pZp] &\to a[pZp][pZp] \mid a[pZq][qZp],\\ [pZq] &\to a[pZp][pZq] \mid a[pZq][qZq] \in P\end{aligned}$$

  und für die Transition (p, a, Z, Z, q) nur die Regel

$$[pZq] \to a[qZq] \in P.$$

- Wir stellen fest, nachdem wir alle Regeln aus den Transitionen gebaut haben, dass die Regel $[pZp] \to a[pZq][qZp]$ weggelassen werden kann, da wir keine Regel mit $[qZp]$ als linker Seite haben. Ein Anwenden dieser Regel könnte also nicht terminieren. Dies ist auch intuitiv klar, da ein Lauf von q nach p – unabhängig vom Stackinhalt – nicht möglich ist.