Vorlesungsscript Theoretische Informatik II

Marco Kuhrmann, 702089

15. November 2000

Inhaltsverzeichnis

1 Motivation **1**

2 Definition O **1**

3 Lemma P **2**

4 Lemma Q **3**

4.1 Vorbereitung des Beweises 3

1 Motivation

Im Vergangenen wurde bereits gezeigt, dass CF $\subseteq$ NPDA ist. Nun ist zu zeigen, dass auch die umgekehrte Richtung CF $\supseteq$ NPDA gilt. Diesen Beweis führt man, indem man eine Teilklasse von NPDA's konstruiert. Man kann nämlich jeden NPDA so konstruieren, dass dieser nach dem Akzeptieren einen eventuell nicht leeren Kellerspeicher bereinigt. Es sollen also NPDA's betrachtet werden, die nur akzeptieren, wenn das gelesene Wort in der Sprache enthalten ist und der Stack leer ist. Beweist man dann die Gültigkeit der Inklusion für diese spezielle Teilklasse der Push-Down-Akzeptoren, so hat man automatisch auch die Gültigkeit für alle NPDA's gezeigt.
Dafür führt man im ersten Schritt dann eine Einschränkung ein. Es sollen für die weiteren Betrachtungen nur diejenigen NPDA's von Interessen sein, die nur akzeptieren, wenn ihr Kellerspeicher leer ist. Dazu folgende Definition:

2 Definition O

Für einen NPDA $\rho = (T, \Gamma, S, \delta, s_0, b_0, F)$ sei $\text{Null}(\rho) = \left\{ w \in T^\star | \, \exists i \in \mathbb{N}_0 \, \exists s \in F \left((s, \varepsilon, b_0) \in \widehat{\delta}^i (s_0, w, b_0) \right) \right\}$ die Menge aller Wörter, die ρ erkennt und die folgenden Anforderungen genügen:

Diese Sprache $\text{Null}(\rho)$ definiert die Menge der Wörter, die ρ akzeptiert, genau dann wenn auch sein Keller leer ist. Aus dieser zusätzlichen Bedingung folgt dann entsprechend die Einschränkung der Sprache, da ja i.A. beim NPDA's der Kellerinhalt für das Akzeptieren nicht weiter von Interesse ist, sofern bereits ein Endzustand erreicht wurde.

Basierend auf dieser Feststellung muß nun im folgenden noch genau definiert werden, wie ein solcher Akzeptor funktioniert. Es muß ferner klargestellt werden, wie er zu reagieren hat, falls er in einem Endzustand landet, der Stack jedoch nicht leer ist. Man muß also quasi "Buchhaltungsschritte" in die Zustandsüberführungsfunktion mit einbinden und deren Arbeitsweise spezifizieren. Dieses geschiet in folgendem Lemma:

3 Lemma P

Zu jedem NPDA $\rho = (T, \Gamma, S, \delta, s_0, b_0, F)$ gibt es einen NPDA $\rho' = (T, \Gamma, S', \delta', s_0, b_0, F')$ mit $L(\rho) = \text{Null}(\rho)$.

Wie man der Definition des Automaten entnehmen kann ist er lediglich in der Zustandsmenge und der Endzustandsmenge und dem zufolge auch in der Überführungsfunktion leicht modifiziert. Er unterscheidet sich sonst nicht von einem "herkömmlichen" NPDA. Also arbeitet ρ' genau wie ρ mit dem Unterschied, dass ρ' nach dem Akzeptieren des Wortes noch den Keller aufräumen muß. Der Beweis für die Existenz erfolgt über die Konstruktion dieses Automaten:

Beweis. Sei ein Akzeptor wie im Lemma gegeben, dann definiere die neuen Mengen wie folgt:

1. S' die neue Zustandsmenge definiert durch $S' = S \cup \{\xi\}$ und die beiden Mengen sind disjunkt.
2. F' sei die neue Menge der Endzustände, die durch $\{\xi\}$ definiert ist.
3. δ' sei die neue Überführungsfunktion, die wie folgt definiert ist:

$$\delta'(s,x,c) = \begin{cases} \delta(s,\varepsilon,c) \cup \{(\xi,c)\}, \forall s \in F \forall c \in \Gamma \text{ falls } x = \varepsilon \\ \{(\xi,\varepsilon)\}, \forall c \in \Gamma \setminus \{b_0\} \text{ falls } s = \xi \wedge x = \varepsilon \\ \delta(s,x,c), \forall c \in \Gamma \forall x \in T \cup \{\varepsilon\}, \text{ falls } s \neq \xi \\ \emptyset, \text{ falls } s = \xi \text{ und } x \neq \varepsilon \vee c = b_0 \end{cases}$$

Hierbei sind die einzelnen Möglichkeiten für δ' wie folgt beschrieben:

1. Die erste Möglichkeit beschreibt den Übergang zum Kellerabbau.
2. Die zweite Möglichkeit beschreibt dann eben diesen oben bereits erwähnten "Buchhaltungsschritt", also den Kellerabbau.
3. Die dritte Möglichkeit beschreibt gerade das Originalverhalten des Akzeptors bis zu dem Punkt, wo er entscheiden muß, ob der Keller noch geleert werden muß oder nicht.
4. Der vierte Entscheidungspfad beschreibt, wie der Automat sich bei einem "Fehler" verhält. Dies ist notwendig, da die Entscheidung was zu tun ist nichtdeterministisch erfolgt. Es kann also durchaus passieren, dass der Akzeptor ungünstig den nächsten Schritt bestimmt. Sollte er dadurch in einen Zustand geraten, indem den Bedingungen des letzten Pfades Genüge getan wird, so sperrt der Akzeptor. Dies hat natürlich zur Folge, dass er auf der einen Seite nicht mehr akzeptieren kann, auch wenn das Wort zu Sprache gehören sollte. Auf der anderen Seite kann der Automat aber auch schon akzeptiert haben und gelangt bei einem Buchhaltungsschritt in diese Situation. Das heißt also, das Wort an sich ist bereits akzeptiert, der Automat sperrt sich jedoch gegen jegliche weitere Aktion, kommt also auch nicht in den eigentlich beabsichtigten Endzustand ξ.

   Wie man der Konstruktion der Überführungsfunktion entnehmen kann bestätigt sich die im Lemma gemachte Aussage, dass ρ' genau wie ρ arbeitet. Allerdings muß dieser um endgültig zu akzeptieren dann im neuen Endzustand ξ ankommen. In diesem Zustand wird der Keller dann gelöscht.

□

Selbsverständlich muß zu der Sprache $\text{Null}(\rho)$ auch eine kontextfreie Grammatik existieren, die eben diese Sprache erzeugen kann. Um dieses zu zeigen soll der nächste Hilfssatz dienen:

4 Lemma Q

Zu jedem NPDA $\rho = (T, \Gamma, S, \delta, s_0, b_0, F)$ gibt es eine kontextfreie Grammatik $G = (N, T, P, \sigma)$ mit $L(G) = \text{Null}(\rho)$.

Um den Beweis für diesen Hilfssatz zu führen bedarf es einiger Vorbereitung. Diese soll dann noch durch ein Beispiel ergänzt werden.

4.1 Vorbereitung des Beweises

Für die Definition der Grammtik setze man nun wie folgt: $N = (S \times \Gamma \times S) \cup \{\sigma\}$ sei die Menge der Nonterminals. Für die Menge der Produktionen ergibt sich:

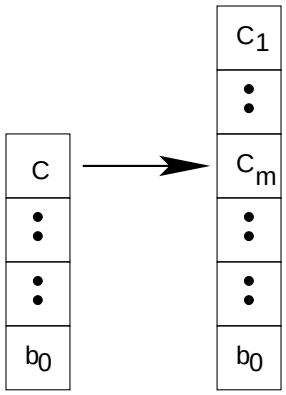
$$
\begin{array}{lcl}
P & = & \{\sigma \to (s_0, b_0, s) \,|\, s \in F\} \\
 & \cup & \{(s, b_0, s) \to \varepsilon | s \in F\} \\
 & \cup & \left\{ \begin{array}{l} (s, c, s') \to a\,(s_1, c_1, s_2)\,(s_2, c_2, s_3) \ldots (s_m, c_m, s') \mid \\ \forall s, s', s_1, \ldots, s_m \in S \forall c, c_1, \ldots, c_m \in \Gamma \forall a \in T \cup \{\varepsilon\}\,((s_1, c_m \ldots c_1) \in \delta(s, a, c)) \end{array} \right\} \\
 & \cup & \{(s, c, s') \to a | \, \forall s, s' \in S \forall c \in \Gamma \forall a \in T \cup \{\varepsilon\}\,((s', \varepsilon) \in \delta(s, a, c))\}
\end{array}
$$

Durch diese Produktionenmenge sollen die Ableitungen erfolgen. Dabei erfüllen die Regeln folgende Funktionen.

Die erste Regel definiert den Startzustand von ρ durch eine Grammatikregel. Die Startsituation, die durch diese Regel beschrieben wird, heißt nichts anderes als: Es ist s_0 der Startzustand und der Kellerspeicher ist leer (b_0). Daraus soll ein Endzustand s erreicht werden. Durch diese Regel wird der Ableitungsprozess, also die Tätigkeit des Automaten angestoßen.

Die zweite Regel soll ausschließlich dazu dienen, überflüssige Nonterminals zu beseitigen. Was dies bedeutet ist später aus der Skizze zur Kellerhöhe relativ gut ersichtlich.

Die dritte Regel erzeugt nun die Ableitungskette, deren Abarbeitung durch Regel 1 angestoßen wurde. Hierbei muß jedes Symbol in einem Zustand bearbeitet werden. Diese Simulation durch eine Produktion entspricht dann dem Zustandsübergang im Automaten. In Regel 1 ist ersichtlich, dass ein Endzustand erreicht werden soll. Dieses geschieht, wie man in Regel 3 sehen kann, über mehrere Schritte, die nötig sind um die Ableitung $s \to s'$ zu erhalten. Die somit erzeugten Übergänge (s_i, c_i, s_{i+1}) müssen existieren, und zwar genau m Stück. Nur so ist es möglich, die Kellersymbole $c_1, \ldots, c_m$, die anstelle von c auf den Stack geschrieben werden, irgendwann zu ersetzen. Dies gilt im Speziellen auch für das Ersetzen durch ε. Die Ersetzung durch ε ist hierbei ein Zwang der Sprache $\text{Null}(\rho)$. Das Tripel (s, c, s') hat folgende Bedeutung in der Abarbeitung: Es ist s der Zustand, in dem der Automat vor einer Folge von Rechenschritten ist. c ist das Symbol, welches als Oberstes auf dem Keller abgelegt ist (*ontop*). Dieses Symbol wird dann durch eine Folge von $c_1, \ldots, c_m$ nach folgendem Schema ersetzt:



Ersetzung von c durch $c_1, \ldots, c_m$

s' kennzeichnet als letzte Komponenete des Tripels den Zustand des Automaten, wenn c bzw. $c_1, ..., c_m$ gepopt sind, also alles was durch c erzeugt wurde wieder vom Stack entfernt worden ist.

Die vierte Regel definiert den Abbau des Kellerspeichers ohne neue Symbole zu schreiben. Es ist also egal was auf dem Speicher liegt, es wird alles entfernt. Dies entspricht der Regel 3 mit dem Spezialfall $m = 0$.

Weitere Erläuterungen

1. Von (s, c, s') aus sollen in G alle Wörter $w \in T^\star$ erzeugt werden, die ρ von der Konfiguration (s, c), also im Zustand s mit c on top startet und in s' mit leerem Keller endet, und dann akzeptiert - und nur dann. Also

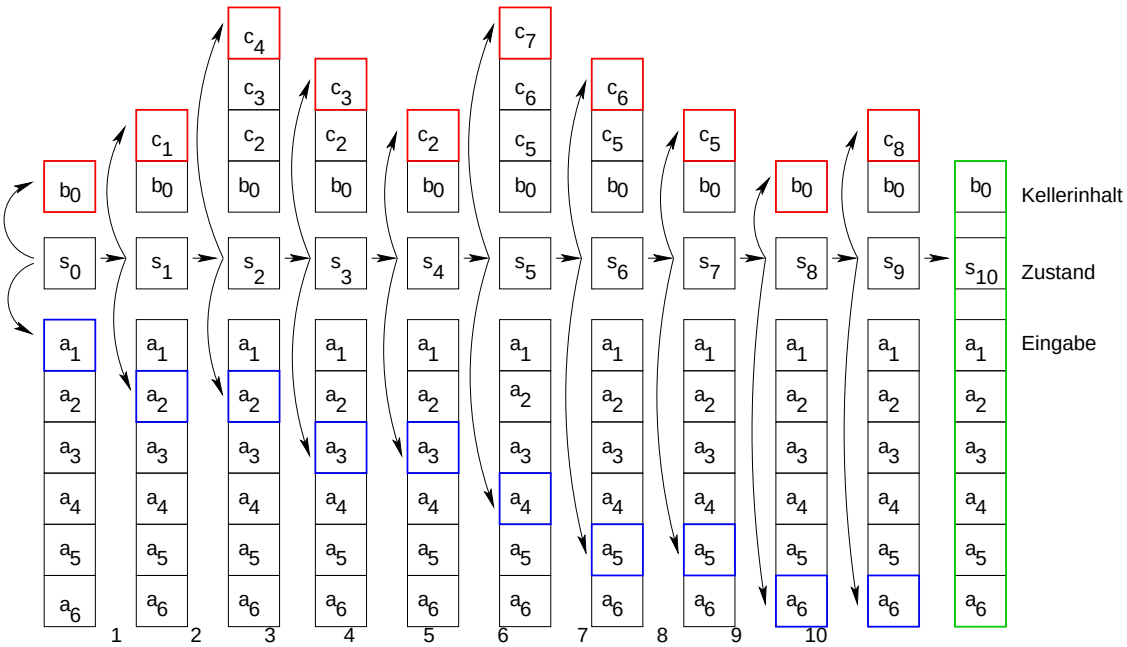
$$(s, c) \overset{w}{\to} (s', b_0)$$

2. Ein Konfigurationsübergang $(s, c) \overset{a}{\to} (s_1, c_m, ..., c_1)$ von ρ wird durch folgende Regel simuliert:

$$(s, c, s') \to a\,(s_1, c_1, s_2)\,(s_2, c_2, s_3)\,...\,(s_m, c_m, s')$$

Von (s_1, c_1, s_2) werden alle Wörter erzeugt, die von ρ bis zum Abbau von $c_1$vom Stack akzeptiert werden. Dies setzt sich analog für die anderen Tupel fort.
Die Zwischenzustände, im Konkreten $s_2, ..., s_k$ sind diejenigen, die ρ unmittelbar nach Abbau der $c_1, ..., c_{k-1}$ erreicht.

Hierzu jedoch ein konkretes Beispiel:

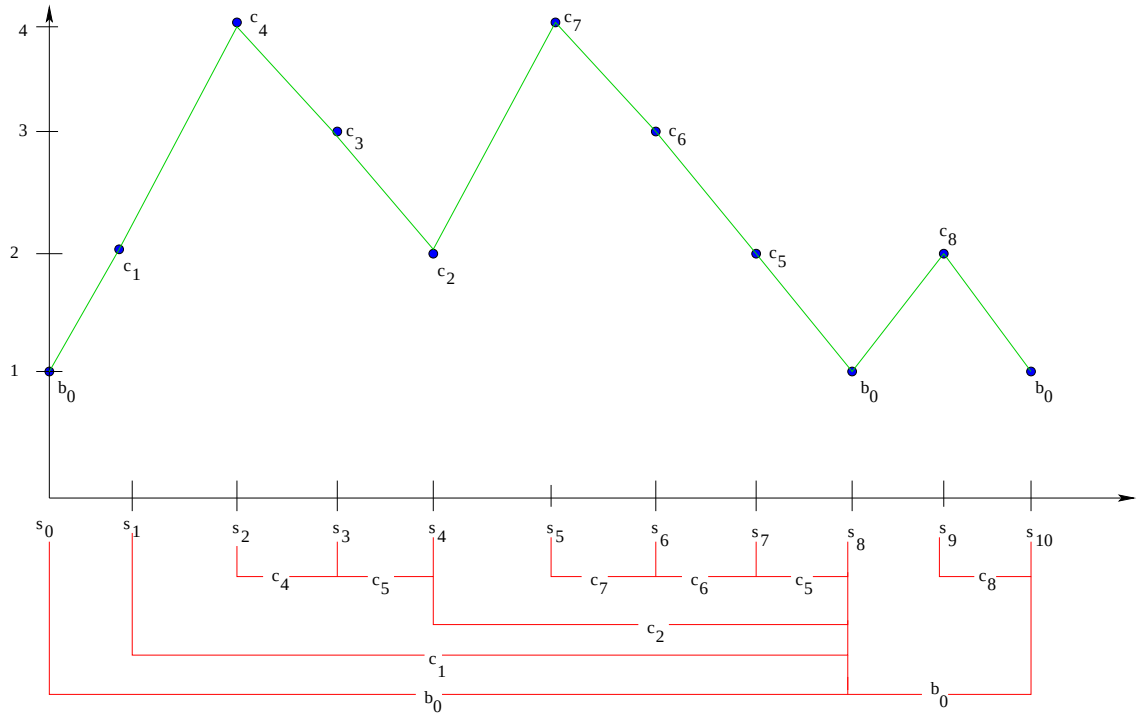


Graphische Darstellung Abarbeitung

Die Überführungen in diesem Beispiel werden alle durch die Funktion $\widehat{\delta}$ realisiert. Die einzelnen Überführungen sind mit 1 bis 10 durchnummeriert und sehen wie folgt aus:

1	$(s_1, b_0, c_1) \in \delta(s_0, a_1, b_0)$
2	$(s_2, c_2 c_3 c_4) \in \delta(s_1, \varepsilon, c_1)$
3	$(s_3, \varepsilon) \in \delta(s_2, a_2, c_4)$
4	$(s_4, \varepsilon) \in \delta(s_3, \varepsilon, c_3)$
5	$(s_5, c_5 c_6 c_7) \in \delta(s_4, a_3, c_2)$
6	$(s_6, \varepsilon) \in \delta(s_5, a_4, c_7)$
7	$(s_7, \varepsilon) \in \delta(s_6, \varepsilon, c_6)$
8	$(s_8, \varepsilon) \in \delta(s_7, a_5, c_5)$
9	$(s_9, c_8) \in \delta(s_8, \varepsilon, b_0)$
10	$(s_{10}, \varepsilon) \in \delta(s_9, a_6, c_8)$

Man kann sich dieses durch eine Skizze der Kellerhöhe noch deutlicher vor Augen führen:



Grafik der Kellerhöhe

In dieser Kellerhöhenskizze steckt die Idee der Konstruktion. Die durch rot verbundenen Tripel beschreiben die für die weitere Berechnung notwendigen Informationen. Die Tripel werden zu Nonterminals.
Man kann ebenfalls die Abarbeitungsreihenfolge und somit auch die Ableitungsfolge erkennen. So sieht man z.B., dass der nächste Zustand nach s_1, bei dem die Kellerhöhe um eins niedriger ist s_8 ist. Das Gleiche gilt, wie man sehen kann, für s_4 und s_8 sowie für s_6 und s_7.

Mit hilfe der Kellerhöhenskizze kann man jetzt den genauen Ableitungsweg nachvollziehen. Dieser lautet wie folgt:

$$
\begin{aligned}
&\sigma \to (s_0, b_0, s_{10}) \\
&\to a_1 (s_1, c_1, s_8)(s_8, b_0, s_{10}) \\
&\to a_1 (s_2, c_4, s_3)(s_3, c_3, s_4)(s_4, c_3, s_8)(s_8, b_0, s_{10}) \\
&\to a_1 a_2 \underbrace{(s_3, c_3, s_4)}_{\varepsilon} (s_4, c_2, s_8)(s_8, b_0, s_{10}) \\
&\to a_1 a_2 (s_4, c_2, s_8)(s_8, b_0, s_{10}) \\
&\to a_1 a_2 a_3 (s_5, c_7, s_6)(s_6, c_6, s_7)(s_7, c_5, s_8)(s_8, b_0, s_{10}) \\
&\to a_1 a_2 a_3 a_4 \underbrace{(s_6, c_6, s_7)}_{\varepsilon} (s_7, c_5, s_8)(s_8, b_0, s_{10}) \\
&\to a_1 a_2 a_3 a_4 \underbrace{(s_7, c_5, s_8)}_{\varepsilon} (s_8, b_0, s_{10}) \\
&\to a_1 a_2 a_3 a_4 (s_8, b_0, s_{10}) \\
&\to a_1 a_2 a_3 a_4 a_5 \underbrace{(s_9, c_8, s_{10})}_{a_6} (s_{10}, b_0, s_{10}) \\
&\to a_1 a_2 a_3 a_4 a_5 a_6 \underbrace{(s_{10}, b_0, s_{10})}_{\varepsilon} \\
&\to a_1 a_2 a_3 a_4 a_5 a_6 \in L(G)
\end{aligned}
$$

Damit ist die Vorbereitung für den Beweis abgeschlossen, dieser findet dann in der nächsten Vorlesung statt.