Script generated by TTT

Title: seidl: Theoretische_Informatik (24.05.2012)

Date: Thu May 24 16:01:37 CEST 2012

Duration: 91:49 min

Pages: 60

Satz 3.44
Das Wortproblem ($w \in L(G)?$) ist für eine CFG G entscheidbar.

Satz 3.44
Das Wortproblem ($w \in L(G)?$) ist für eine CFG G entscheidbar.

Beweis:
OE sei $w \neq \epsilon$.

$S \to^* \varepsilon$

Satz 3.44
Das Wortproblem ($w \in L(G)?$) ist für eine CFG G entscheidbar.

Beweis:
OE sei $w \neq \epsilon$. Wir eliminieren zuerst alle ϵ-Produktionen aus G (wie in Lemma 3.26).

Satz 3.44

Das Wortproblem ($w \in L(G)$?) ist für eine CFG G entscheidbar.

Beweis:

OE sei $w \neq \epsilon$. Wir eliminieren zuerst alle ϵ-Produktionen aus G (wie in Lemma 3.26).

Dann berechnen wir induktiv die Menge R aller von S ableitbaren Wörter $\in (V \cup \Sigma)^*$,

Satz 3.44

Das Wortproblem ($w \in L(G)$?) ist für eine CFG G entscheidbar.

Beweis:

OE sei $w \neq \epsilon$. Wir eliminieren zuerst alle ϵ-Produktionen aus G (wie in Lemma 3.26).

Dann berechnen wir induktiv die Menge R aller von S ableitbaren Wörter $\in (V \cup \Sigma)^*$, die nicht länger als w sind:

- $S \in R$

Satz 3.44

Das Wortproblem ($w \in L(G)$?) ist für eine CFG G entscheidbar.

Beweis:

OE sei $w \neq \epsilon$. Wir eliminieren zuerst alle ϵ-Produktionen aus G (wie in Lemma 3.26).

Dann berechnen wir induktiv die Menge R aller von S ableitbaren Wörter $\in (V \cup \Sigma)^*$, die nicht länger als w sind:

- $S \in R$

- Wenn $\alpha B \gamma \in R$ und $(B \to \beta) \in P$ und $|\alpha\beta\gamma| \leq |w|$, dann auch $\alpha\beta\gamma \in R$.

Satz 3.44

Das Wortproblem ($w \in L(G)$?) ist für eine CFG G entscheidbar.

Beweis:

OE sei $w \neq \epsilon$. Wir eliminieren zuerst alle ϵ-Produktionen aus G (wie in Lemma 3.26).

Dann berechnen wir induktiv die Menge R aller von S ableitbaren Wörter $\in (V \cup \Sigma)^*$, die nicht länger als w sind:

- $S \in R$
- Wenn $\alpha B \gamma \in R$ und $(B \to \beta) \in P$ und $|\alpha\beta\gamma| \leq |w|$, dann auch $\alpha\beta\gamma \in R$.

Man zeigt:

$$w \in L_V(G) \quad \Leftrightarrow \quad w \in R$$

wobei $L_V(G) := \{w \in (V \cup \Sigma)^* \mid S \to_G^* w\}$.

Satz 3.44

Das Wortproblem ($w \in L(G)$?) ist für eine CFG G entscheidbar.

Beweis:

OE sei $w \neq \epsilon$. Wir eliminieren zuerst alle ϵ-Produktionen aus G (wie in Lemma 3.26).

Dann berechnen wir induktiv die Menge R aller von S ableitbaren Wörter $\in (V \cup \Sigma)^*$, die nicht länger als w sind:

- $S \in R$
- Wenn $\alpha B \gamma \in R$ und $(B \to \beta) \in P$ und $|\alpha\beta\gamma| \leq |w|$, dann auch $\alpha\beta\gamma \in R$.

Man zeigt:

$$w \in L_V(G) \quad \Leftrightarrow \quad w \in R$$

wobei $L_V(G) := \{w \in (V \cup \Sigma)^* \mid S \to_G^* w\}$.

Da R endlich ist ($|R| \leq |V \cup \Sigma|^{|w|}$), ist $w \in R$ entscheidbar, und damit auch $w \in L_V(G)$, und damit auch $w \in L(G)$. □

Der CYK-Algorithmus entscheidet das Wortproblem für kontextfreie Grammatiken in Chomsky-Normalform.

3.7 Der Cocke-Younger-Kasami-Algorithmus

Der CYK-Algorithmus entscheidet das Wortproblem für kontextfreie Grammatiken in Chomsky-Normalform.

Eingabe: Grammatik $G = (V, \Sigma, P, S)$ in Chomsky-Normalform, $w = a_1 \dots a_n \in \Sigma^*$.

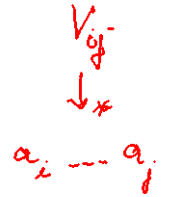
3.7 Der Cocke-Younger-Kasami-Algorithmus

Der CYK-Algorithmus entscheidet das Wortproblem für kontextfreie Grammatiken in Chomsky-Normalform.

Eingabe: Grammatik $G = (V, \Sigma, P, S)$ in Chomsky-Normalform, $w = a_1 \dots a_n \in \Sigma^*$.

Definition 3.45

$$V_{ij} := \{A \in V \mid A \to_G^* a_i \dots a_j\}$$



3.7 Der Cocke-Younger-Kasami-Algorithmus

Der CYK-Algorithmus entscheidet das Wortproblem für kontextfreie Grammatiken in Chomsky-Normalform.

Eingabe: Grammatik $G = (V, \Sigma, P, S)$ in Chomsky-Normalform, $w = a_1 \dots a_n \in \Sigma^*$.

Definition 3.45

$$V_{ij} := \{A \in V \mid A \to_G^* a_i \dots a_j\} \quad \text{für } i \leq j$$

Damit gilt:

$$w \in L(G) \quad \Leftrightarrow \quad S \in V_{1n}$$

Der CYK-Algorithmus berechnet die V_{ij} rekursiv nach wachsendem $j - i$:

$$V_{ii} = \{A \in V \mid (A \to a_i) \in P\}$$

Der CYK-Algorithmus berechnet die V_{ij} rekursiv nach wachsendem $j - i$:

$$V_{ii} = \{A \in V \mid (A \to a_i) \in P\}$$
$$V_{ij} = \left\{ A \in V \mid \begin{array}{c} \exists i \leq k < j,\ B \in V_{ik},\ C \in V_{k+1,j} \cdot \\ (A \to BC) \in P \end{array} \right\} \quad \text{für } i < j$$

Der CYK-Algorithmus berechnet die V_{ij} rekursiv nach wachsendem $j - i$:

$$V_{ii} = \{A \in V \mid (A \to a_i) \in P\}$$
$$V_{ij} = \left\{ A \in V \mid \begin{array}{c} \exists i \leq k < j,\ B \in V_{ik},\ C \in V_{k+1,j} \cdot \\ (A \to BC) \in P \end{array} \right\} \quad \text{für } i < j$$

Korrektheitsbeweis: Induktion nach $j - i$.

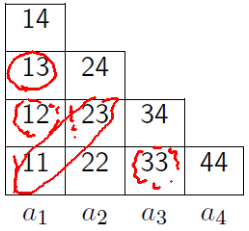
Der CYK-Algorithmus berechnet die V_{ij} rekursiv nach wachsendem $j - i$:

$$V_{ii} = \{A \in V \mid (A \to a_i) \in P\}$$

$$V_{ij} = \left\{ A \in V \mid \begin{array}{c} \exists i \le k < j,\ B \in V_{ik},\ C \in V_{k+1,j} \cdot \\ (A \to BC) \in P \end{array} \right\} \quad \text{für } i < j$$

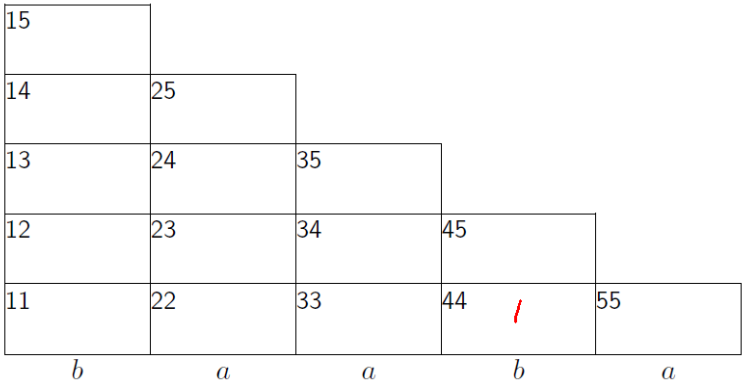
Korrektheitsbeweis: Induktion nach $j - i$.

Die V_{ij} als Tabelle (mit ij statt V_{ij}):



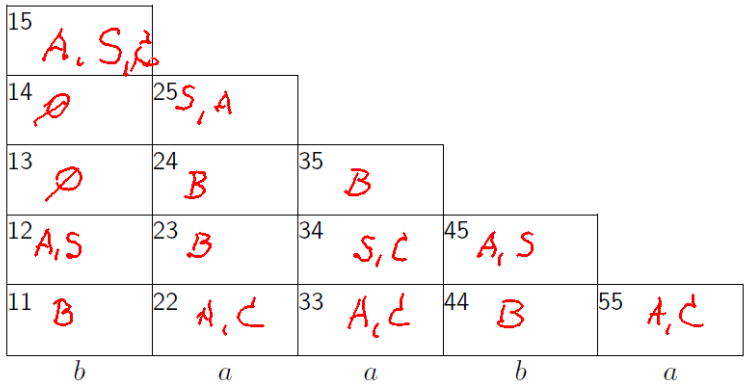
Beispiel 3.46

$$\begin{array}{lcl} S & \to & AB \mid BC \\ A & \to & BA \mid a \\ B & \to & CC \mid b \\ C & \to & AB \mid a \end{array}$$



Beispiel 3.46

$$\begin{array}{lcl} S & \to & AB \mid BC \\ A & \to & BA \mid a \\ B & \to & CC \mid b \\ C & \to & AB \mid a \end{array}$$



Satz 3.47

Der CYK-Algorithmus entscheidet das Wortproblem $w \in L(G)$ für eine fixe CFG G in Chomsky-Normalform in Zeit $O(|w|^3)$.

Satz 3.47
Der CYK-Algorithmus entscheidet das Wortproblem $w \in L(G)$ für eine fixe CFG G in Chomsky-Normalform in Zeit $O(|w|^3)$.

Beweis:
Sei $n := |w|$. Es werden $\frac{n(n-1)}{2} \in O(n^2)$ Mengen V_{ij} berechnet.

Satz 3.47
Der CYK-Algorithmus entscheidet das Wortproblem $w \in L(G)$ für eine fixe CFG G in Chomsky-Normalform in Zeit $O(|w|^3)$.

Beweis:
Sei $n := |w|$. Es werden $\frac{n(n-1)}{2} \in O(n^2)$ Mengen V_{ij} berechnet.

$$V_{ij} = \left\{ A \in V \mid \begin{array}{c} \exists i \le k < j,\ B \in V_{ik},\ C \in V_{k+1,j} \cdot \\ (A \to BC) \in P \end{array} \right\} \quad (i < j)$$

Für jede dieser Mengen werden

- $j - i < n$ Werte für k betrachtet,
- für jedes k wird für alle Produktionen $A \to BC$ untersucht, ob $B \in V_{ik}$ und $C \in V_{k+1,j}$,
- wobei $|V_{ik}|, |V_{k+1,j}| \le |V|$.

Satz 3.47

Der CYK-Algorithmus entscheidet das Wortproblem $w \in L(G)$ für eine fixe CFG G in Chomsky-Normalform in Zeit $O(|w|^3)$.

Beweis:

Sei $n := |w|$. Es werden $\frac{n(n-1)}{2} \in O(n^2)$ Mengen V_{ij} berechnet.

$$V_{ij} = \left\{ A \in V \mid \begin{array}{c} \exists i \leq k < j,\ B \in V_{ik},\ C \in V_{k+1,j} \cdot \\ (A \to BC) \in P \end{array} \right\} \quad (i < j)$$

Für jede dieser Mengen werden

- $j - i < n$ Werte für k betrachtet,
- für jedes k wird für alle Produktionen $A \to BC$ untersucht, ob $B \in V_{ik}$ und $C \in V_{k+1,j}$, wobei $|V_{ik}|, |V_{k+1,j}| \leq |V|$.

Gesamtzeit: $O(n^3)$

Satz 3.47

Der CYK-Algorithmus entscheidet das Wortproblem $w \in L(G)$ für eine fixe CFG G in Chomsky-Normalform in Zeit $O(|w|^3)$.

Beweis:

Sei $n := |w|$. Es werden $\frac{n(n-1)}{2} \in O(n^2)$ Mengen V_{ij} berechnet.

$$V_{ij} = \left\{ A \in V \mid \begin{array}{c} \exists i \leq k < j,\ B \in V_{ik},\ C \in V_{k+1,j} \cdot \\ (A \to BC) \in P \end{array} \right\} \quad (i < j)$$

Für jede dieser Mengen werden

- $j - i < n$ Werte für k betrachtet,
- für jedes k wird für alle Produktionen $A \to BC$ untersucht, ob $B \in V_{ik}$ und $C \in V_{k+1,j}$, wobei $|V_{ik}|, |V_{k+1,j}| \leq |V|$.

Gesamtzeit: $O(n^3)$

Denn $|P|$ und $|V|$ sind Konstanten unabhängig von n.

[Konstruktion jeder Menge V_{ii}: $O(1)$. Für alle V_{ii} also $O(n)$.] □

Erweiterung

Der CYK-Algorithmus kann so erweitert werden, dass er nicht nur das Wortproblem entscheidet, sondern auch die Menge der Syntaxbäume für die Eingabe berechnet.

Erweiterung

Der CYK-Algorithmus kann so erweitert werden, dass er nicht nur das Wortproblem entscheidet, sondern auch die Menge der Syntaxbäume für die Eingabe berechnet.

Realisierung:

- V_{ij} ist die Menge der Syntaxbäume mit Rand $a_i \dots a_j$.

Erweiterung
Der CYK-Algorithmus kann so erweitert werden, dass er nicht nur das Wortproblem entscheidet, sondern auch die Menge der Syntaxbäume für die Eingabe berechnet.

Realisierung:

- V_{ij} ist die Menge der Syntaxbäume mit Rand $a_i \dots a_j$.
- Statt A enthält V_{ij} einen Syntaxbaum, dessen Wurzel mit A beschriftet ist.

Erweiterung
Der CYK-Algorithmus kann so erweitert werden, dass er nicht nur das Wortproblem entscheidet, sondern auch die Menge der Syntaxbäume für die Eingabe berechnet.

Realisierung:

- V_{ij} ist die Menge der Syntaxbäume mit Rand $a_i \dots a_j$.
- Statt A enthält V_{ij} einen Syntaxbaum, dessen Wurzel mit A beschriftet ist.

$S \to^* BC \to^* bC \to^* bAB \to^* baB \to^* ba\,CC \to^* baAB\,a \to^* baaba$

Vorschau
Für CFGs sind folgende Probleme nicht entscheidbar:

Vorschau
Für CFGs sind folgende Probleme nicht entscheidbar:

- Äquivalenz: $L(G_1) = L(G_2)$?

Vorschau

Für CFGs sind folgende Probleme nicht entscheidbar:

- Äquivalenz: $L(G_1) = L(G_2)$?
- Schnittproblem: $L(G_1) \cap L(G_2) = \emptyset$?

Vorschau

Für CFGs sind folgende Probleme nicht entscheidbar:

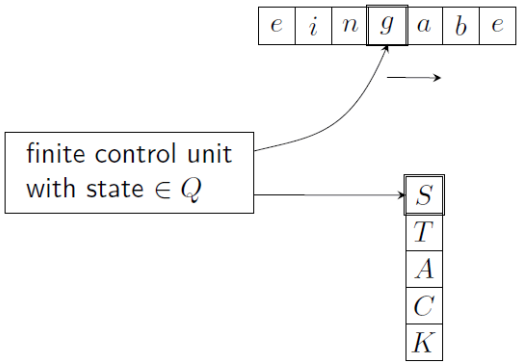
- Äquivalenz: $L(G_1) = L(G_2)$?
- Schnittproblem: $L(G_1) \cap L(G_2) = \emptyset$?
- Regularität: $L(G)$ regulär?

Vorschau

Für CFGs sind folgende Probleme nicht entscheidbar:

- Äquivalenz: $L(G_1) = L(G_2)$?
- Schnittproblem: $L(G_1) \cap L(G_2) = \emptyset$?
- Regularität: $L(G)$ regulär?
- Mehrdeutigkeit: Ist G mehrdeutig?

3.8 Kellerautomaten



Anwendungsgebiete von Kellerautomaten:

- Syntaxanalyse von Programmiersprachen

Anwendungsgebiete von Kellerautomaten:

- Syntaxanalyse von Programmiersprachen
- Analyse von Programmen mit Rekursion

Definition 3.48 (PDA)

Ein (nichtdeterministischer) Kellerautomat ((N)PDA = (Nondeterministic) Pushdown Automaton) besteht aus:

Q endliche Zustandsmenge

Definition 3.48 (PDA)

Ein (nichtdeterministischer) Kellerautomat ((N)PDA = (Nondeterministic) Pushdown Automaton) besteht aus:

Q endliche Zustandsmenge

Σ endliches Eingabealphabet

Γ endliches Kelleralphabet

Definition 3.48 (PDA)

Ein (nichtdeterministischer) Kellerautomat ((N)PDA = (Nondeterministic) Pushdown Automaton) besteht aus:

- Q endliche Zustandsmenge
- Σ endliches Eingabealphabet
- Γ endliches Kelleralphabet
- $q_0 \in Q$ Anfangszustand
- $Z_0 \in \Gamma$ Anfangskellerinhalt

Definition 3.48 (PDA)

Ein (nichtdeterministischer) Kellerautomat ((N)PDA = (Nondeterministic) Pushdown Automaton) besteht aus:

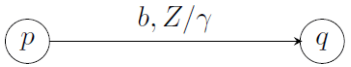
- Q endliche Zustandsmenge
- Σ endliches Eingabealphabet
- Γ endliches Kelleralphabet
- $q_0 \in Q$ Anfangszustand
- $Z_0 \in \Gamma$ Anfangskellerinhalt
- δ Übergangsfunktion $Q \times (\Sigma \cup \{\epsilon\}) \times \Gamma \to \mathcal{P}(Q \times \Gamma^*)$ wobei $|\delta(q, b, Z)| < \infty$ für $q \in Q, b \in \Sigma \cup \{\epsilon\}, Z \in \Gamma$.

Definition 3.48 (PDA)

Ein (nichtdeterministischer) Kellerautomat ((N)PDA = (Nondeterministic) Pushdown Automaton) besteht aus:

- Q endliche Zustandsmenge
- Σ endliches Eingabealphabet
- Γ endliches Kelleralphabet
- $q_0 \in Q$ Anfangszustand
- $Z_0 \in \Gamma$ Anfangskellerinhalt
- δ Übergangsfunktion $Q \times (\Sigma \cup \{\epsilon\}) \times \Gamma \to \mathcal{P}(Q \times \Gamma^*)$ wobei $|\delta(q, b, Z)| < \infty$ für $q \in Q, b \in \Sigma \cup \{\epsilon\}, Z \in \Gamma$.
- $F \subseteq Q$ akzeptierende Zustände oder Endzustände

Eine graphische Notation für $\delta(p, b, Z) \ni (q, \gamma)$:



Achtung: Kein endlicher Automat!

Die schrittweise Verarbeitung der Eingabe wird als Relation $\to$ zwischen *Konfigurationen* (= Gesamtzuständen) des Systems beschrieben:

Definition 3.49
Eine Konfiguration eines PDA $M = (Q, \Sigma, \Gamma, q_0, Z_0, \delta, F)$ ist ein Tripel (q, w, γ) mit

- $q \in Q$ (Zustand),
- $w \in \Sigma^*$ ((Rest)Eingabe),
- $\gamma \in \Gamma^*$ (Keller).

Die schrittweise Verarbeitung der Eingabe wird als Relation $\to$ zwischen *Konfigurationen* (= Gesamtzuständen) des Systems beschrieben:

Definition 3.49
Eine Konfiguration eines PDA $M = (Q, \Sigma, \Gamma, q_0, Z_0, \delta, F)$ ist ein Tripel (q, w, γ) mit

- $q \in Q$ (Zustand),
- $w \in \Sigma^*$ ((Rest)Eingabe),
- $\gamma \in \Gamma^*$ (Keller).

Die Ersetzungsrelation $\to_M$ ergibt sich wie folgt aus δ:
Falls $\delta(q, b, Z) \ni (q', \gamma')$ (wobei $b \in \Sigma \cup \{\epsilon\}$) dann

Die schrittweise Verarbeitung der Eingabe wird als Relation $\to$ zwischen *Konfigurationen* (= Gesamtzuständen) des Systems beschrieben:

Definition 3.49
Eine Konfiguration eines PDA $M = (Q, \Sigma, \Gamma, q_0, Z_0, \delta, F)$ ist ein Tripel (q, w, γ) mit

- $q \in Q$ (Zustand),
- $w \in \Sigma^*$ ((Rest)Eingabe),
- $\gamma \in \Gamma^*$ (Keller).

Die Ersetzungsrelation $\to_M$ ergibt sich wie folgt aus δ:
Falls $\delta(q, b, Z) \ni (q', \gamma')$ (wobei $b \in \Sigma \cup \{\epsilon\}$) dann

$$(q, bw, Z\gamma) \to_M (q',$$

Die schrittweise Verarbeitung der Eingabe wird als Relation $\to$ zwischen *Konfigurationen* (= Gesamtzuständen) des Systems beschrieben:

Definition 3.49
Eine Konfiguration eines PDA $M = (Q, \Sigma, \Gamma, q_0, Z_0, \delta, F)$ ist ein Tripel (q, w, γ) mit

- $q \in Q$ (Zustand),
- $w \in \Sigma^*$ ((Rest)Eingabe),
- $\gamma \in \Gamma^*$ (Keller).

Die Ersetzungsrelation $\to_M$ ergibt sich wie folgt aus δ:
Falls $\delta(q, b, Z) \ni (q', \gamma')$ (wobei $b \in \Sigma \cup \{\epsilon\}$) dann

$$(q, bw, Z\gamma) \to_M (q', w,$$

Die schrittweise Verarbeitung der Eingabe wird als Relation $\to$ zwischen *Konfigurationen* (= Gesamtzuständen) des Systems beschrieben:

Definition 3.49

Eine Konfiguration eines PDA $M = (Q, \Sigma, \Gamma, q_0, Z_0, \delta, F)$ ist ein Tripel (q, w, γ) mit

- $q \in Q$ (Zustand),
- $w \in \Sigma^*$ ((Rest)Eingabe),
- $\gamma \in \Gamma^*$ (Keller).

Die Ersetzungsrelation $\to_M$ ergibt sich wie folgt aus δ:
Falls $\delta(q, b, Z) \ni (q', \gamma')$ (wobei $b \in \Sigma \cup \{\epsilon\}$) dann

$$(q, bw, Z\gamma) \to_M (q', w, \gamma'\gamma)$$

Die schrittweise Verarbeitung der Eingabe wird als Relation $\to$ zwischen *Konfigurationen* (= Gesamtzuständen) des Systems beschrieben:

Definition 3.49

Eine Konfiguration eines PDA $M = (Q, \Sigma, \Gamma, q_0, Z_0, \delta, F)$ ist ein Tripel (q, w, γ) mit

- $q \in Q$ (Zustand),
- $w \in \Sigma^*$ ((Rest)Eingabe),
- $\gamma \in \Gamma^*$ (Keller).

Die Ersetzungsrelation $\to_M$ ergibt sich wie folgt aus δ:
Falls $\delta(q, b, Z) \ni (q', \gamma')$ (wobei $b \in \Sigma \cup \{\epsilon\}$) dann

$$(q, bw, Z\gamma) \to_M (q', w, \gamma'\gamma)$$

Achtung: $b = \epsilon$ und $\gamma' = \epsilon$ und $|\gamma'| > 1$ möglich!

Definition 3.50

Ein PDA M akzeptiert $w \in \Sigma^*$ gdw

$$(q_0, w, Z_0) \to_M^* (f, \epsilon, \gamma)$$

für ein $f \in F$, $\gamma \in \Gamma^*$.

Beispiel 3.51

Die Sprache $L = \{ww^R \mid w \in \{0,1\}^*\}$ wird vom PDA

$$M = (\{p, q, r\}, \{0, 1\}, \{0, 1, Z_0\}, p,$$

Beispiel 3.51

Die Sprache $L = \{ww^R \mid w \in \{0,1\}^*\}$ wird vom PDA

$$M = (\{p,q,r\},\ \{0,1\},\ \{0,1,Z_0\},\ p,\ Z_0,$$

Beispiel 3.51

Die Sprache $L = \{ww^R \mid w \in \{0,1\}^*\}$ wird vom PDA

$$M = (\{p,q,r\},\ \{0,1\},\ \{0,1,Z_0\},\ p,\ Z_0,\ \delta,$$

Beispiel 3.51

Die Sprache $L = \{ww^R \mid w \in \{0,1\}^*\}$ wird vom PDA

$$M = (\{p,q,r\},\ \{0,1\},\ \{0,1,Z_0\},\ p,\ Z_0,\ \delta,\ \{r\})$$

Beispiel 3.51

Die Sprache $L = \{ww^R \mid w \in \{0,1\}^*\}$ wird vom PDA

$$M = (\{p,q,r\},\ \{0,1\},\ \{0,1,Z_0\},\ p,\ Z_0,\ \delta,\ \{r\})$$

$$\delta(p,a,Z) \quad = \quad \{(p,\ aZ)\} \quad \text{für } a \in \{0,1\},\ Z \in \{0,1,Z_0\}$$

Beispiel 3.51

Die Sprache $L = \{ww^R \mid w \in \{0,1\}^*\}$ wird vom PDA

$$M = (\{p,q,r\}, \{0,1\}, \{0,1,Z_0\}, p, Z_0, \delta, \{r\})$$

$$\begin{array}{lcll} \delta(p,a,Z) & = & \{(p, aZ)\} & \text{für } a \in \{0,1\},\ Z \in \{0,1,Z_0\} \\ \delta(p,\epsilon,Z) & = & \{(q, Z)\} & \text{für } Z \in \{0,1,Z_0\} \end{array}$$

Beispiel 3.51

Die Sprache $L = \{ww^R \mid w \in \{0,1\}^*\}$ wird vom PDA

$$M = (\{p,q,r\}, \{0,1\}, \{0,1,Z_0\}, p, Z_0, \delta, \{r\})$$

$$\begin{array}{lcll} \delta(p,a,Z) & = & \{(p, aZ)\} & \text{für } a \in \{0,1\},\ Z \in \{0,1,Z_0\} \\ \delta(p,\epsilon,Z) & = & \{(q, Z)\} & \text{für } Z \in \{0,1,Z_0\} \end{array}$$

$$\begin{array}{lcll} \delta(q,a,a) & = & \{(q, \epsilon)\} & \text{für } a \in \{0,1\} \\ \delta(q,\epsilon,Z_0) & = & \{(r, \epsilon)\} & \end{array}$$

Beispiel 3.51

Die Sprache $L = \{ww^R \mid w \in \{0,1\}^*\}$ wird vom PDA

$$M = (\{p,q,r\}, \{0,1\}, \{0,1,Z_0\}, p, Z_0, \delta, \{r\})$$

$$\begin{array}{lcll} \delta(p,a,Z) & = & \{(p, aZ)\} & \text{für } a \in \{0,1\},\ Z \in \{0,1,Z_0\} \\ \delta(p,\epsilon,Z) & = & \{(q, Z)\} & \text{für } Z \in \{0,1,Z_0\} \\ \delta(q,a,a) & = & \{(q, \epsilon)\} & \text{für } a \in \{0,1\} \\ \delta(q,\epsilon,Z_0) & = & \{(r, \epsilon)\} & \end{array}$$

akzeptiert.

Hauptresultate:

- Kellerautomaten akzeptieren genau die kontextfreien Spachen.
- Nichtdeterministische Kellerautomaten (PDAs) sind mächtiger als deterministische Kellerautomaten (DPDAs).

3.9 Tabellarischer Überblick

Abschlusseigenschaften

	Schnitt	Vereinigung	Komplement	Produkt	Stern