Script generated by TTT

Title: Seidl: Theoretische_Informatik (03.06.2013)

Date: Mon Jun 03 10:15:22 CEST 2013

Duration: 92:26 min

Pages: 47

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 $w \in L(G)$. □

3.7 Der Cocke-Younger-Kasami-Algorithmus

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

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

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 $j - i$:

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

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:

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

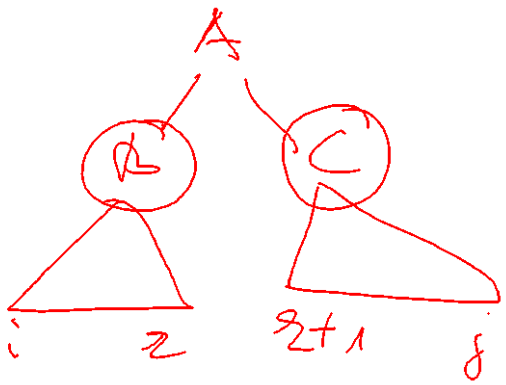
Der CYK-Algorithmus berechnet die V_{ij} rekursiv nach wachsendem $j - i$:

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

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

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

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



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

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

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

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}):

14			
13	24		
12	23	34	
11	22	33	44
a_1	a_2	a_3	a_4

Beispiel 3.46

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

15 S				
14	25 S, C, A			
13	24 B	35 B		
12 A, S	23 B	34 S, C	45 S, A	
11 B	22 A, C	33 A, C	44 B	55 A, C
b	a	a	b	a

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.

Beispiel 3.46

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

15				
14	25			
13	24	35		
12	23	34	45	
11	22	33	44	55
b	a	a	b	a

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.

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

Für jede dieser Mengen werden

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.

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

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 $|V_{ik}|, |V_{k+1,j}| \le |V|$.

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.

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}: $O(\ )$.

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.

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:

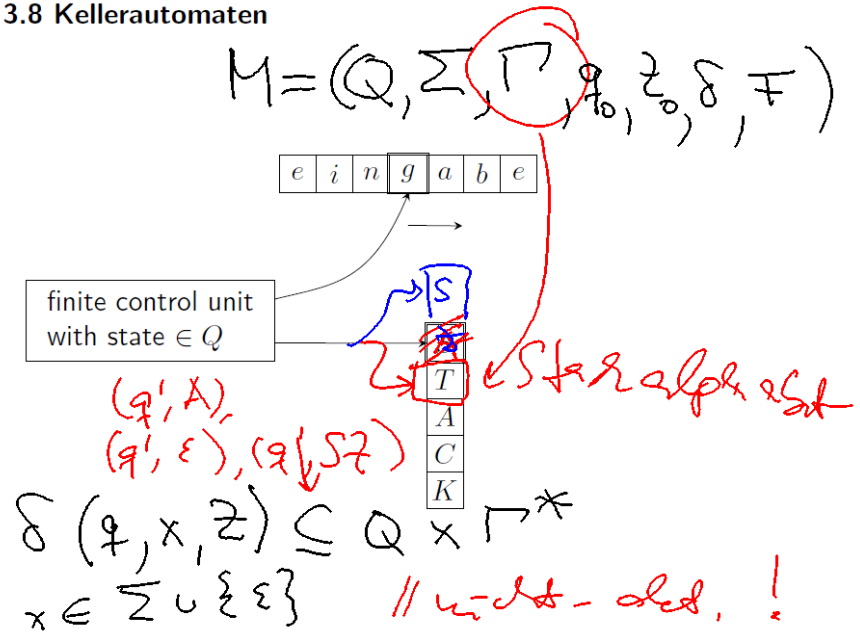
- Äquivalenz: $L(G_1) = L(G_2)$?
- Schnittproblem: $L(G_1) \cap L(G_2) = \emptyset$?
- Regularität: $L(G)$ regulär?
- Mehrdeutigkeit: Ist G mehrdeutig?

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



Konfiguration

$(q, w, \gamma) \in Q \times \Sigma^* \times \Gamma^*$

Start: (q_0, w, Z_0)

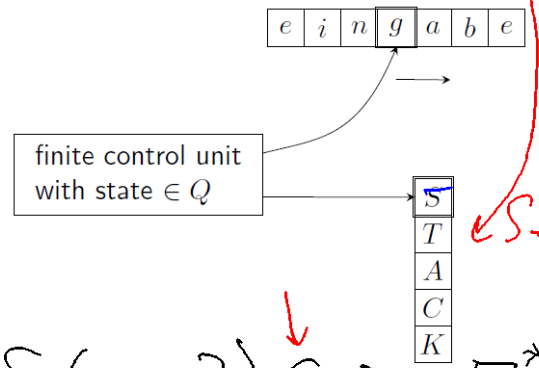
End: // Kopf ist leer

(q, ε, γ), $q \in F$

$\vdash_M$, $\vdash_M^*$

3.8 Kellerautomaten

$M = (Q, \Sigma, \Gamma, q_0, Z_0, \delta, F)$



Stapelalphabet

$\delta(q, x, Z) \subseteq Q \times \Gamma^*$

$x \in \Sigma \cup \{\varepsilon\}$ // nicht-det.!

Anwendungsgebiete von Kellerautomaten:

- Syntaxanalyse von Programmiersprachen
- Analyse von Programmen mit Rekursion

Compilerbau

Definition 3.48

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

$$(q_0, w, Z_0) \to_M^* (f, \epsilon, \gamma) \text{ für ein } f \in F, \gamma \in \Gamma^*.$$

Definition 3.48

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

$$(q_0, w, Z_0) \to_M^* (f, \epsilon, \gamma) \text{ für ein } f \in F,\ \gamma \in \Gamma^*.$$

$$L_F(M) := \{w \mid \exists f \in F, \gamma \in \Gamma^*.\ (q_0, w, Z_0) \to_M^* (f, \epsilon, \gamma)\}$$

Definition 3.48

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

$$(q_0, w, Z_0) \to_M^* (f, \epsilon, \gamma) \text{ für ein } f \in F,\ \gamma \in \Gamma^*.$$

$$L_F(M) := \{w \mid \exists f \in F, \gamma \in \Gamma^*.\ (q_0, w, Z_0) \to_M^* (f, \epsilon, \gamma)\}$$

- Ein PDA M akzeptiert $w \in \Sigma^*$ mit leeren Keller gdw

$$(q_0, w, Z_0) \to_M^* (q, \epsilon, \epsilon) \text{ für ein } q \in Q.$$

Definition 3.48

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

$$(q_0, w, Z_0) \to_M^* (f, \epsilon, \gamma) \text{ für ein } f \in F,\ \gamma \in \Gamma^*.$$

$$L_F(M) := \{w \mid \exists f \in F, \gamma \in \Gamma^*.\ (q_0, w, Z_0) \to_M^* (f, \epsilon, \gamma)\}$$

- Ein PDA M akzeptiert $w \in \Sigma^*$ mit leeren Keller gdw

$$(q_0, w, Z_0) \to_M^* (q, \epsilon, \epsilon) \text{ für ein } q \in Q.$$

$$L_\epsilon(M) := \{w \mid \exists q \in Q.\ (q_0, w, Z_0) \to_M^* (q, \epsilon, \epsilon)\}$$

Definition 3.48

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

$$(q_0, w, Z_0) \to_M^* (f, \epsilon, \gamma) \text{ für ein } f \in F,\ \gamma \in \Gamma^*.$$

$$L_F(M) := \{w \mid \exists f \in F, \gamma \in \Gamma^*.\ (q_0, w, Z_0) \to_M^* (f, \epsilon, \gamma)\}$$

- Ein PDA M akzeptiert $w \in \Sigma^*$ mit leeren Keller gdw

$$(q_0, w, Z_0) \to_M^* (q, \epsilon, \epsilon) \text{ für ein } q \in Q.$$

$$L_\epsilon(M) := \{w \mid \exists q \in Q.\ (q_0, w, Z_0) \to_M^* (q, \epsilon, \epsilon)\}$$

Konvention: Wir blenden die F-Komponente von M aus, wenn wir nur an $L_\epsilon(M)$ interessiert sind.

Beispiel 3.49

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

$M = (\{p, q, r\},$

Beispiel 3.49

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

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

Beispiel 3.49

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 3.49

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

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

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

Beispiel 3.49

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 3.49

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

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

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

Beispiel 3.49

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

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

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

Beispiel 3.49

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

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

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

sowohl mit Endzustand als auch mit leerem Keller akzeptiert.