Übung 4: Pumping-Lemmata, CNF und CYK

Theoretische Informatik Sommersemester 2014

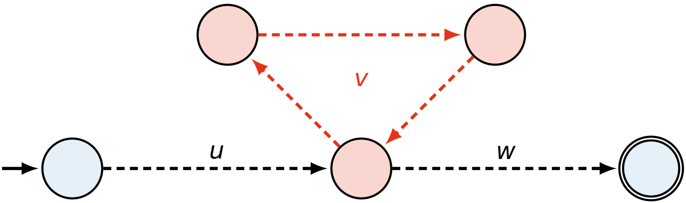
Markus Kaiser

14. Mai 2014

Satz (Pumping Lemma für reguläre Sprachen)

Sei $R \subseteq \Sigma^*$ regulär.
Dann gibt es ein $n > 0$, so dass sich jedes $z \in R$ mit $|z| \geq n$ so in $z = uvw$ zerlegen lässt, dass

- $v \neq \epsilon$
- $|uv| \leq n$
- $\forall i \geq 0 . uv^i w \in R$



Definition

Sei $G = (V, \Sigma, P, S)$ eine CFG.
Ein Symbol $X \in V \cup \Sigma$ ist

nützlich	es gibt $S \rightarrow_G^* w \in \Sigma^*$ in der X vorkommt
erzeugend	es gibt $X \rightarrow_G^* w \in \Sigma^*$
erreichbar	es gibt $S \rightarrow_G^* \alpha X \beta$

Satz

Nützliche Symbole sind erzeugend und erreichbar. Aber nicht notwendigerweise umgekehrt.

$$S \rightarrow AB \mid a, \quad A \rightarrow b$$

Definition (Chomsky-Normalform)

Eine kontextfreie Grammatik ist in Chomsky-Normalform (CNF) genau dann wenn alle Produktionen die Form

$$A \to a \quad \text{oder} \quad A \to BC$$

haben.

Satz

Zu jeder CFG G existiert eine CFG G' in Chomsky-Normalform mit

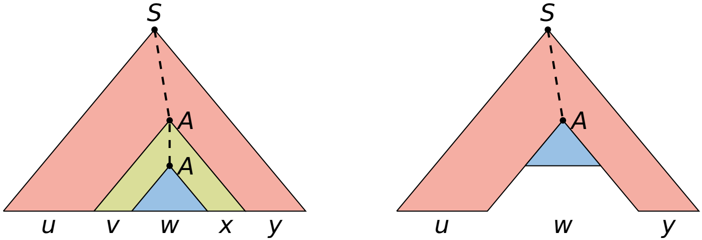
$$L(G') = L(G) \setminus \{\epsilon\}$$

Satz (Pumping Lemma für kontextfreie Sprachen)

Sei $L \subseteq \Sigma^*$ kontextfrei.
Dann gibt es ein $n > 0$, so dass sich jedes $z \in L$ mit $|z| \geq n$ so in $z = uvwxy$ zerlegen lässt, dass

- $vx \neq \epsilon$
- $|vwx| \leq n$
- $\forall i \geq 0 . uv^i wx^i y \in L$

$$S \to^* uAy \to^* uvAxy \to^* uvwxy \qquad S \to^* uAy \to^* uwy$$



CNF Konstruktion

Sei $G = (V, \Sigma, P, S)$ eine CFG.

1. Eliminiere ϵ-Produktionen
2. Eliminiere Kettenproduktionen
3. Ersetze Terminale durch Nichtterminale
4. Verkürze Ketten von Nichtterminalen der Länge ≥ 3

CNF Konstruktion

Sei $G = (V, \Sigma, P, S)$ eine CFG.

1. Eliminiere ϵ-Produktionen
2. Eliminiere Kettenproduktionen
3. Ersetze Terminale durch Nichtterminale
4. Verkürze Ketten von Nichtterminalen der Länge ≥ 3

Sind $B \to \epsilon$ und $A \to \alpha B \beta$ in P, dann füge $A \to \alpha\beta$ hinzu. Entferne danach alle ϵ-Produktionen.

$$S \to Ab, \quad A \to aAA \mid \epsilon$$

wird zu:

$$S \to Ab \mid b$$
$$A \to aAA \mid aA \mid a$$

CNF Konstruktion

Sei $G = (V, \Sigma, P, S)$ eine CFG.

1. Eliminiere ϵ-Produktionen
2. Eliminiere Kettenproduktionen
3. Ersetze Terminale durch Nichtterminale
4. Verkürze Ketten von Nichtterminalen der Länge ≥ 3

Sind $A \to B$ und $B \to \alpha$ in P, dann füge $A \to \alpha$ hinzu. Entferne danach alle Kettenproduktionen und unerreichbaren Symbole.

$$S \to A, \quad A \to a \mid B, \quad B \to bS$$

wird zu:

$$A \to a \mid bS$$
$$S \to a \mid bS$$

CNF Konstruktion

Sei $G = (V, \Sigma, P, S)$ eine CFG.

1. Eliminiere ϵ-Produktionen
2. Eliminiere Kettenproduktionen
3. Ersetze Terminale durch Nichtterminale
4. Verkürze Ketten von Nichtterminalen der Länge ≥ 3

Ersetze jedes $a \in \Sigma$ in einer rechten Seite länger als 1 durch ein neues Nichtterminal.

$$S \rightarrow aa \mid Bb \mid b, \quad B \rightarrow \dots$$

wird zu:

$$S \rightarrow X_a X_a \mid B X_b \mid b$$
$$X_a \rightarrow a, \quad X_b \rightarrow b$$

CNF Konstruktion

Sei $G = (V, \Sigma, P, S)$ eine CFG.

1. Eliminiere ϵ-Produktionen
2. Eliminiere Kettenproduktionen
3. Ersetze Terminale durch Nichtterminale
4. Verkürze Ketten von Nichtterminalen der Länge ≥ 3

Ersetze jede Produktion der Form $A \rightarrow B_1 B_2 \ldots B_k$ durch neue Nichtterminale mit Produktionen der Länge 2.

$$S \rightarrow X_a X_b B X_a, \quad X_a \rightarrow a, \quad X_b \rightarrow b, \quad B \rightarrow \ldots$$

wird zu:

$$S \rightarrow X_a T_1$$
$$T_1 \rightarrow X_b T_2, \quad T_2 \rightarrow B X_a$$

Definition (Cocke-Younger-Kasami-Algorithmus)

Der CYK-Algorithmus entscheidet das Wortproblem für kontextfreie Grammatiken in Chomsky-Normalform in $\mathcal{O}(n^3)$.
Gegeben eine Grammatik $G = (V, \Sigma, P, S)$ in CNF und ein Wort $w = a_1 \dots a_n \in \Sigma^*$. Mit

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

ist

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

$$V_{ii} = \{A \in V \mid (A \to a_i) \in P\}$$
$$V_{ij} = \{A \in V \mid \exists k, B \in V_{ik}, C \in V_{k+1,j} \,.\, (A \to BC) \in P\}$$

CYK-Algorithmus

Kombiniere Teilwörter zum ganzen Wort, wenn möglich.

1. Initialisiere mit den V_{ii}.
2. Befülle die Tabelle von unten nach oben.

$$S \to AB \mid BC, \quad A \to BA \mid a, \quad B \to CC \mid b, \quad C \to AB \mid a$$

4				
3				
2				
1				
	b	a	a	a

CYK-Algorithmus

Kombiniere Teilwörter zum ganzen Wort, wenn möglich.

1. Initialisiere mit den V_{ii}.
2. Befülle die Tabelle von unten nach oben.

$$S \to AB \mid BC, \quad A \to BA \mid a, \quad B \to CC \mid b, \quad C \to AB \mid a$$

4				
3				
2				
1	B	A, C	A, C	A, C
	b	a	a	a

CYK-Algorithmus

Kombiniere Teilwörter zum ganzen Wort, wenn möglich.

1. Initialisiere mit den V_{ij}.
2. Befülle die Tabelle von unten nach oben.

$$S \to AB \mid BC, \quad A \to BA \mid a, \quad B \to CC \mid b, \quad C \to AB \mid a$$

4				
3				
2	A, S	B	B	
1	B	A, C	A, C	A, C
	b	a	a	a

CYK-Algorithmus

Kombiniere Teilwörter zum ganzen Wort, wenn möglich.

1. Initialisiere mit den V_{ii}.
2. Befülle die Tabelle von unten nach oben.

$$S \to AB \mid BC, \quad A \to BA \mid a, \quad B \to CC \mid b, \quad C \to AB \mid a$$

4				
3	$\emptyset$	S, A, C		
2	A, S	B	B	
1	B	A, C	A, C	A, C
	b	a	a	a

CYK-Algorithmus

Kombiniere Teilwörter zum ganzen Wort, wenn möglich.

1. Initialisiere mit den V_{ii}.
2. Befülle die Tabelle von unten nach oben.

$$S \to AB \mid BC, \quad A \to BA \mid a, \quad B \to CC \mid b, \quad C \to AB \mid a$$

4	$S, \ldots$			
3	$\emptyset$	S, A, C		
2	A, S	B	B	
1	B	A, C	A, C	A, C
	b	a	a	a