Script generated by TTT

Title: Nipkow: Theo (23.05.2019)

Date: Thu May 23 14:14:27 CEST 2019

Duration: 91:19 min

Pages: 109

Satz 4.32 (Pumping-Lemma)

Für jede kontextfreie Sprache L gibt es ein $n \geq 1$, so dass sich jedes Wort $z \in L$ mit $|z| \geq n$ zerlegen lässt in

$$z = uvwxy,$$

$u, v, w, x, y \in \Sigma^*$

mit

- $vx \neq \epsilon$,
- $|vwx| \leq n$, *und*
- $\forall i \in \mathbb{N}.\ uv^iwx^iy \in L$.

$S \to uAy$

$A \to vAx \mid w$

$L = \{uv^iwx^iy \mid i \in \mathbb{N}\}$

$S \to uAy \to uvAxy \to uv^2Ax^2y \to uv^2wx^2y$

Beweis:
Sei $G = (V, \Sigma, P, S)$ eine CFG in Chomsky-Normalform mit $L(G) = L \setminus \{\epsilon\}$. Wähle $n = 2^{|V|}$.

4.5 Algorithmen für kontextfreie Grammatiken

Wie üblich: $G = (V, \Sigma, P, S)$ ist eine CFG.

Ein Symbol $X \in V \cup \Sigma$ ist

nützlich gdw es eine Ableitung $S \to_G^* w \in \Sigma^*$ gibt, in der X vorkommt.

4.5 Algorithmen für kontextfreie Grammatiken

Wie üblich: $G = (V, \Sigma, P, S)$ ist eine CFG.

Ein Symbol $X \in V \cup \Sigma$ ist

- nützlich gdw es eine Ableitung $S \to_G^* w \in \Sigma^*$ gibt, in der X vorkommt.
- erzeugend gdw es eine Ableitung $X \to_G^* w \in \Sigma^*$ gibt.

4.5 Algorithmen für kontextfreie Grammatiken

Wie üblich: $G = (V, \Sigma, P, S)$ ist eine CFG.

Ein Symbol $X \in V \cup \Sigma$ ist

- nützlich gdw es eine Ableitung $S \to_G^* w \in \Sigma^*$ gibt, in der X vorkommt.
- erzeugend gdw es eine Ableitung $X \to_G^* w \in \Sigma^*$ gibt.
- erreichbar gdw es eine Ableitung $S \to_G^* \alpha X \beta$ gibt.

4.5 Algorithmen für kontextfreie Grammatiken

Wie üblich: $G = (V, \Sigma, P, S)$ ist eine CFG.

Ein Symbol $X \in V \cup \Sigma$ ist

- nützlich gdw es eine Ableitung $S \to_G^* w \in \Sigma^*$ gibt, in der X vorkommt.
- erzeugend gdw es eine Ableitung $X \to_G^* w \in \Sigma^*$ gibt.
- erreichbar gdw es eine Ableitung $S \to_G^* \alpha X \beta$ gibt.

Fakt 4.35

Nützliche Symbole sind erzeugend und erreichbar.

4.5 Algorithmen für kontextfreie Grammatiken

Wie üblich: $G = (V, \Sigma, P, S)$ ist eine CFG.

Ein Symbol $X \in V \cup \Sigma$ ist

- nützlich gdw es eine Ableitung $S \to_G^* w \in \Sigma^*$ gibt, in der X vorkommt.
- erzeugend gdw es eine Ableitung $X \to_G^* w \in \Sigma^*$ gibt.
- erreichbar gdw es eine Ableitung $S \to_G^* \alpha X \beta$ gibt.

Fakt 4.35

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

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

4.5 Algorithmen für kontextfreie Grammatiken

Wie üblich: $G = (V, \Sigma, P, S)$ ist eine CFG.

Ein Symbol $X \in V \cup \Sigma$ ist

- **nützlich** gdw es eine Ableitung $S \to_G^* w \in \Sigma^*$ gibt, in der X vorkommt.
- **erzeugend** gdw es eine Ableitung $X \to_G^* w \in \Sigma^*$ gibt.
- **erreichbar** gdw es eine Ableitung $S \to_G^* \alpha X \beta$ gibt.

Fakt 4.35

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

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

Ziel: Elimination der unnützen Symbole und der Produktionen, in denen sie vorkommen.

Beispiel 4.36

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

Beispiel 4.36

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

1. Elimination nicht erzeugender Symbole:

$$S \to a, \quad A \to b$$

Beispiel 4.36

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

1. Elimination nicht erzeugender Symbole:

$$S \to a, \quad A \to b$$

2. Elimination unerreichbarer Symbole:

$$S \to a$$

Beispiel 4.36

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

1. Elimination nicht erzeugender Symbole:

$$S \to a, \quad A \to b$$

2. Elimination unerreichbarer Symbole:

$$S \to a$$

Umgekehrte Reihenfolge:

1. Elimination unerreichbarer Symbole:

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

Beispiel 4.36

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

1. Elimination nicht erzeugender Symbole:

$$S \to a, \quad A \to b$$

2. Elimination unerreichbarer Symbole:

$$S \to a$$

Umgekehrte Reihenfolge:

1. Elimination unerreichbarer Symbole:

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

2. Elimination nicht erzeugender Symbole:

$$S \to a, \quad A \to b$$

Satz 4.37

Eliminiert man aus einer Grammatik G

1. *alle nicht erzeugenden Symbole, mit Resultat G_1, und*

Satz 4.37

Eliminiert man aus einer Grammatik G

1. *alle nicht erzeugenden Symbole, mit Resultat G_1, und*
2. *aus G_1 alle unerreichbaren Symbole, mit Resultat G_2,*

Satz 4.37

Eliminiert man aus einer Grammatik G

1. *alle nicht erzeugenden Symbole, mit Resultat G_1, und*
2. *aus G_1 alle unerreichbaren Symbole, mit Resultat G_2,*

dann enthält G_2 nur noch nützliche Symbole und $L(G_2) = L(G)$.

Beweis:

Wir zeigen zuerst $L(G_2) = L(G)$.

Satz 4.37

Eliminiert man aus einer Grammatik G

1. *alle nicht erzeugenden Symbole, mit Resultat G_1, und*
2. *aus G_1 alle unerreichbaren Symbole, mit Resultat G_2,*

dann enthält G_2 nur noch nützliche Symbole und $L(G_2) = L(G)$.

Beweis:

Wir zeigen zuerst $L(G_2) = L(G)$.

Da $P_2 \subseteq P$ gilt $L(G_2) \subseteq L(G)$.

Satz 4.37

Eliminiert man aus einer Grammatik G

1. *alle nicht erzeugenden Symbole, mit Resultat G_1, und*
2. *aus G_1 alle unerreichbaren Symbole, mit Resultat G_2,*

dann enthält G_2 nur noch nützliche Symbole und $L(G_2) = L(G)$.

Beweis:

Wir zeigen zuerst $L(G_2) = L(G)$.

Da $P_2 \subseteq P$ gilt $L(G_2) \subseteq L(G)$.

Umgekehrt, sei $w \in L(G)$, dh $S \to_G^* w$.

Satz 4.37

Eliminiert man aus einer Grammatik G

1. *alle nicht erzeugenden Symbole, mit Resultat G_1, und*
2. *aus G_1 alle unerreichbaren Symbole, mit Resultat G_2,*

dann enthält G_2 nur noch nützliche Symbole und $L(G_2) = L(G)$.

Beweis:

Wir zeigen zuerst $L(G_2) = L(G)$.

Da $P_2 \subseteq P$ gilt $L(G_2) \subseteq L(G)$.

Umgekehrt, sei $w \in L(G)$, dh $S \to_G^* w$.
Jedes Symbol in dieser Ableitung ist erreichbar und erzeugend.

Satz 4.37

Eliminiert man aus einer Grammatik G

1. *alle nicht erzeugenden Symbole, mit Resultat G_1, und*
2. *aus G_1 alle unerreichbaren Symbole, mit Resultat G_2,*

dann enthält G_2 nur noch nützliche Symbole und $L(G_2) = L(G)$.

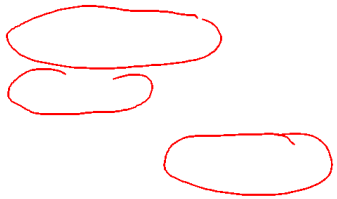
Beweis:

Wir zeigen zuerst $L(G_2) = L(G)$.

Da $P_2 \subseteq P$ gilt $L(G_2) \subseteq L(G)$.

Umgekehrt, sei $w \in L(G)$, dh $S \to_G^* w$.
Jedes Symbol in dieser Ableitung ist erreichbar und erzeugend.
Also gilt auch $S \to_{G_2}^* w$, dh $w \in L(G_2)$.

Beweis (Forts.): $[G_1$: erzeugend in $G,\ G_2$: erreichbar in $G_1]$



Beweis (Forts.): $[G_1$: erzeugend in $G,\ G_2$: erreichbar in $G_1]$

Beweis (Forts.): $[G_1$: erzeugend in $G,\ G_2$: erreichbar in $G_1]$

Wir zeigen: Alle $X \in V_2 \cup \Sigma_2$ sind nützlich in G_2, dh

$$S \quad \to_{G_2}^* \quad \ldots X \ldots \quad \to_{G_2}^* \quad \ldots \in \Sigma_2^*$$

Beweis (Forts.): [G_1 : erzeugend in G, G_2 : erreichbar in G_1]

Wir zeigen: Alle $X \in V_2 \cup \Sigma_2$ sind nützlich in G_2, dh

$$S \quad \to^*_{G_2} \quad \ldots X \ldots \quad \to^*_{G_2} \quad \ldots \in \Sigma_2^*$$

X muss in G_1 erreichbar sein, dh $S \to^*_{G_1} \alpha X \beta$.

Beweis (Forts.): [G_1 : erzeugend in G, G_2 : erreichbar in G_1]

Wir zeigen: Alle $X \in V_2 \cup \Sigma_2$ sind nützlich in G_2, dh

$$S \quad \to^*_{G_2} \quad \ldots X \ldots \quad \to^*_{G_2} \quad \ldots \in \Sigma_2^*$$

X muss in G_1 erreichbar sein, dh $S \to^*_{G_1} \alpha X \beta$.
Da alle Symbole in der Ableitung erreichbar sind: $S \to^*_{G_2} \alpha X \beta$.

Beweis (Forts.): [G_1 : erzeugend in G, G_2 : erreichbar in G_1]

Wir zeigen: Alle $X \in V_2 \cup \Sigma_2$ sind nützlich in G_2, dh

$$S \quad \to^*_{G_2} \quad \ldots X \ldots \quad \to^*_{G_2} \quad \ldots \in \Sigma_2^*$$

X muss in G_1 erreichbar sein, dh $S \to^*_{G_1} \alpha X \beta$. //
Da alle Symbole in der Ableitung erreichbar sind: $S \to^*_{G_2} \alpha X \beta$.

Alle Symbole in $\alpha X \beta$ müssen in G erzeugend sein:

Beweis (Forts.): [G_1 : erzeugend in G, G_2 : erreichbar in G_1]

Wir zeigen: Alle $X \in V_2 \cup \Sigma_2$ sind nützlich in G_2, dh

$$S \quad \to^*_{G_2} \quad \ldots X \ldots \quad \to^*_{G_2} \quad \ldots \in \Sigma_2^*$$

X muss in G_1 erreichbar sein, dh $S \to^*_{G_1} \alpha X \beta$.
Da alle Symbole in der Ableitung erreichbar sind: $S \to^*_{G_2} \alpha X \beta$.

Alle Symbole in $\alpha X \beta$ müssen in G erzeugend sein:
$\forall Y \in \alpha X \beta.\ \exists u \in \Sigma^*.\ Y \to^*_G u$

Beweis (Forts.): [G_1 : erzeugend in G, G_2 : erreichbar in G_1]

Wir zeigen: Alle $X \in V_2 \cup \Sigma_2$ sind nützlich in G_2, dh

$$S \quad \to^*_{G_2} \quad \ldots X \ldots \quad \to^*_{G_2} \quad \ldots \in \Sigma_2^*$$

X muss in G_1 erreichbar sein, dh $S \to^*_{G_1} \alpha X \beta$.
Da alle Symbole in der Ableitung erreichbar sind: $S \to^*_{G_2} \alpha X \beta$.

Alle Symbole in $\alpha X \beta$ müssen in G erzeugend sein:
$\forall Y \in \alpha X \beta.\ \exists u \in \Sigma^*.\ Y \to^*_G u$
Da alle Symbole in den Ableitungen $Y \to^*_G u$ erzeugend sind:

Beweis (Forts.): [G_1 : erzeugend in G, G_2 : erreichbar in G_1]

Wir zeigen: Alle $X \in V_2 \cup \Sigma_2$ sind nützlich in G_2, dh

$$S \quad \to^*_{G_2} \quad \ldots X \ldots \quad \to^*_{G_2} \quad \ldots \in \Sigma_2^*$$

X muss in G_1 erreichbar sein, dh $S \to^*_{G_1} \alpha X \beta$.
Da alle Symbole in der Ableitung erreichbar sind: $S \to^*_{G_2} \alpha X \beta$.

Alle Symbole in $\alpha X \beta$ müssen in G erzeugend sein:
$\forall Y \in \alpha X \beta.\ \exists u \in \Sigma^*.\ Y \to^*_G u$
Da alle Symbole in den Ableitungen $Y \to^*_G u$ erzeugend sind:
$\forall Y \in \alpha X \beta.\ \exists u \in \Sigma_1^*.\ Y \to^*_{G_1} u$

Beweis (Forts.): [G_1 : erzeugend in G, G_2 : erreichbar in G_1]

Wir zeigen: Alle $X \in V_2 \cup \Sigma_2$ sind nützlich in G_2, dh

$$S \quad \to^*_{G_2} \quad \ldots X \ldots \quad \to^*_{G_2} \quad \ldots \in \Sigma_2^*$$

X muss in G_1 erreichbar sein, dh $S \to^*_{G_1} \alpha X \beta$.
Da alle Symbole in der Ableitung erreichbar sind: $S \to^*_{G_2} \alpha X \beta$.

Alle Symbole in $\alpha X \beta$ müssen in G erzeugend sein:
$\forall Y \in \alpha X \beta.\ \exists u \in \Sigma^*.\ Y \to^*_G u$
Da alle Symbole in den Ableitungen $Y \to^*_G u$ erzeugend sind:
$\forall Y \in \alpha X \beta.\ \exists u \in \Sigma_1^*.\ Y \to^*_{G_1} u$
Also gibt es $w \in \Sigma_1^*$ mit $\alpha X \beta \to^*_{G_1} w.$

Beweis (Forts.): [G_1 : erzeugend in G, G_2 : erreichbar in G_1]

Wir zeigen: Alle $X \in V_2 \cup \Sigma_2$ sind nützlich in G_2, dh

$$S \quad \to^*_{G_2} \quad \ldots X \ldots \quad \to^*_{G_2} \quad \ldots \in \Sigma_2^*$$

X muss in G_1 erreichbar sein, dh $S \to^*_{G_1} \alpha X \beta$.
Da alle Symbole in der Ableitung erreichbar sind: $S \to^*_{G_2} \alpha X \beta$.

Alle Symbole in $\alpha X \beta$ müssen in G erzeugend sein:
$\forall Y \in \alpha X \beta.\ \exists u \in \Sigma^*.\ Y \to^*_G u$
Da alle Symbole in den Ableitungen $Y \to^*_G u$ erzeugend sind:
$\forall Y \in \alpha X \beta.\ \exists u \in \Sigma_1^*.\ Y \to^*_{G_1} u$
Also gibt es $w \in \Sigma_1^*$ mit $\alpha X \beta \to^*_{G_1} w.$

Die Ableitung $\alpha X \beta \to^*_{G_1} w$ enthält nur Symbole, die in G_1 erreichbar sind.

Beweis (Forts.): [G_1 : erzeugend in G, G_2 : erreichbar in G_1]

Wir zeigen: Alle $X \in V_2 \cup \Sigma_2$ sind nützlich in G_2, dh

$$S \quad \to^*_{G_2} \quad \ldots X \ldots \quad \to^*_{G_2} \quad \ldots \in \Sigma_2^*$$

X muss in G_1 erreichbar sein, dh $S \to^*_{G_1} \alpha X \beta$.
Da alle Symbole in der Ableitung erreichbar sind: $S \to^*_{G_2} \alpha X \beta$.

Alle Symbole in $\alpha X \beta$ müssen in G erzeugend sein:
$\forall Y \in \alpha X \beta.\ \exists u \in \Sigma^*.\ Y \to^*_G u$
Da alle Symbole in den Ableitungen $Y \to^*_G u$ erzeugend sind:
$\forall Y \in \alpha X \beta.\ \exists u \in \Sigma_1^*.\ Y \to^*_{G_1} u$
Also gibt es $w \in \Sigma_1^*$ mit $\alpha X \beta \to^*_{G_1} w$.

Die Ableitung $\alpha X \beta \to^*_{G_1} w$ enthält nur Symbole, die in G_1 erreichbar sind. Daher auch $\alpha X \beta \to^*_{G_2} w$.

Beweis (Forts.): [G_1 : erzeugend in G, G_2 : erreichbar in G_1]

Wir zeigen: Alle $X \in V_2 \cup \Sigma_2$ sind nützlich in G_2, dh

$$S \quad \to^*_{G_2} \quad \ldots X \ldots \quad \to^*_{G_2} \quad \ldots \in \Sigma_2^*$$

X muss in G_1 erreichbar sein, dh $S \to^*_{G_1} \alpha X \beta$.
Da alle Symbole in der Ableitung erreichbar sind: $S \to^*_{G_2} \alpha X \beta$.

Alle Symbole in $\alpha X \beta$ müssen in G erzeugend sein:
$\forall Y \in \alpha X \beta.\ \exists u \in \Sigma^*.\ Y \to^*_G u$
Da alle Symbole in den Ableitungen $Y \to^*_G u$ erzeugend sind:
$\forall Y \in \alpha X \beta.\ \exists u \in \Sigma_1^*.\ Y \to^*_{G_1} u$
Also gibt es $w \in \Sigma_1^*$ mit $\alpha X \beta \to^*_{G_1} w$.

Die Ableitung $\alpha X \beta \to^*_{G_1} w$ enthält nur Symbole, die in G_1 erreichbar sind. Daher auch $\alpha X \beta \to^*_{G_2} w$.

$$S \ \to^*_{G_2} \ \alpha X \beta \ \to^*_{G_2} \ w$$

□

178

Satz 4.38

Die Menge der erzeugenden Symbole einer CFG ist berechenbar.

Satz 4.38

Die Menge der erzeugenden Symbole einer CFG ist berechenbar.

Beweis:

Wir berechnen die erzeugenden Symbole induktiv:

179

Satz 4.38

Die Menge der erzeugenden Symbole einer CFG ist berechenbar.

Beweis:

Wir berechnen die erzeugenden Symbole induktiv:

- Alle Symbole in Σ sind erzeugend.
- Falls $(A \to \alpha) \in P$ und alle Symbole in α sind erzeugend, dann ist auch A erzeugend.

□

Satz 4.38

Die Menge der erzeugenden Symbole einer CFG ist berechenbar.

Beweis:

Wir berechnen die erzeugenden Symbole induktiv:

- Alle Symbole in Σ sind erzeugend.
- Falls $(A \to \alpha) \in P$ und alle Symbole in α sind erzeugend, dann ist auch A erzeugend.

□

Beispiel 4.39

$$S \to SAB, \quad A \to BC, \quad B \to C, \quad C \to c$$

Erzeugend: $\{$

Satz 4.38

Die Menge der erzeugenden Symbole einer CFG ist berechenbar.

Beweis:

Wir berechnen die erzeugenden Symbole induktiv:

- Alle Symbole in Σ sind erzeugend.
- Falls $(A \to \alpha) \in P$ und alle Symbole in α sind erzeugend, dann ist auch A erzeugend.

□

Beispiel 4.39

$$S \to SAB, \quad A \to BC, \quad B \to C, \quad C \to c$$

Erzeugend: $\{c,$

Satz 4.38

Die Menge der erzeugenden Symbole einer CFG ist berechenbar.

Beweis:

Wir berechnen die erzeugenden Symbole induktiv:

- Alle Symbole in Σ sind erzeugend.
- Falls $(A \to \alpha) \in P$ und alle Symbole in α sind erzeugend, dann ist auch A erzeugend.

□

Beispiel 4.39

$$S \to SAB, \quad A \to BC, \quad B \to C, \quad C \to c$$

Erzeugend: $\{c, C,$

Satz 4.38

Die Menge der erzeugenden Symbole einer CFG ist berechenbar.

Beweis:

Wir berechnen die erzeugenden Symbole induktiv:

- Alle Symbole in Σ sind erzeugend.
- Falls $(A \to \alpha) \in P$ und alle Symbole in α sind erzeugend, dann ist auch A erzeugend. □

Beispiel 4.39

$$S \to SAB, \quad A \to BC, \quad B \to C, \quad C \to c$$

Erzeugend: $\{c, C, B, A\}$.

Satz 4.38

Die Menge der erzeugenden Symbole einer CFG ist berechenbar.

Beweis:

Wir berechnen die erzeugenden Symbole induktiv:

- Alle Symbole in Σ sind erzeugend.
- Falls $(A \to \alpha) \in P$ und alle Symbole in α sind erzeugend, dann ist auch A erzeugend. □

Beispiel 4.39

$$S \to SAB, \quad A \to BC, \quad B \to C, \quad C \to c$$

Erzeugend: $\{c, C, B, A\}$.

Korollar 4.40

Für eine kontextfreie Grammatik G ist entscheidbar, ob $L(G) = \emptyset$.

179

Satz 4.38

Die Menge der erzeugenden Symbole einer CFG ist berechenbar.

Beweis:

Wir berechnen die erzeugenden Symbole induktiv:

- Alle Symbole in Σ sind erzeugend.
- Falls $(A \to \alpha) \in P$ und alle Symbole in α sind erzeugend, dann ist auch A erzeugend. □

Beispiel 4.39

$$S \to SAB, \quad A \to BC, \quad B \to C, \quad C \to c$$

Erzeugend: $\{c, C, B,$

179

Beispiel 4.36

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

1. Elimination nicht erzeugender Symbole:

$$S \to a, \quad A \to b$$

Satz 4.38

Die Menge der erzeugenden Symbole einer CFG ist berechenbar.

Beweis:

Wir berechnen die erzeugenden Symbole induktiv:

- Alle Symbole in Σ sind erzeugend.
- Falls $(A \to \alpha) \in P$ und alle Symbole in α sind erzeugend, dann ist auch A erzeugend. □

Beispiel 4.39

$$S \to SAB, \quad A \to BC, \quad B \to C, \quad C \to c$$

Erzeugend: $\{c, C, B, A\}$.

Korollar 4.40

Für eine kontextfreie Grammatik G ist entscheidbar, ob $L(G) = \emptyset$.

Satz 4.38

Die Menge der erzeugenden Symbole einer CFG ist berechenbar.

Beweis:

Wir berechnen die erzeugenden Symbole induktiv:

- Alle Symbole in Σ sind erzeugend.
- Falls $(A \to \alpha) \in P$ und alle Symbole in α sind erzeugend, dann ist auch A erzeugend. □

Beispiel 4.39

$$S \to SAB, \quad A \to BC, \quad B \to C, \quad C \to c$$

Erzeugend: $\{c, C, B, A\}$.

Korollar 4.40

Für eine kontextfreie Grammatik G ist entscheidbar, ob $L(G) = \emptyset$.

Beweis:

$L(G) = \emptyset \iff S$ ist nicht erzeugend. □

Satz 4.41

Die Menge der erreichbaren Symbole einer CFG ist berechenbar.

Beweis:

Satz 4.41

Die Menge der erreichbaren Symbole einer CFG ist berechenbar.

Beweis:

Wir berechnen die erreichbaren Symbole induktiv:

Satz 4.42

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

Satz 4.42

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

Satz 4.42

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

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

Satz 4.42

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 4.26).
Dann berechnen wir induktiv die Menge R aller von S ableitbaren Wörter $\in (V \cup \Sigma)^*$,

Satz 4.42

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 4.26).

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

Satz 4.42

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 4.26).

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 4.42

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 4.26).

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 4.42

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 4.26).

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 $\alpha\beta\gamma \in R$.

Es gilt: für alle $w' \in (V \cup \Sigma)^*$ mit $|w'| \leq |w|$

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

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

Satz 4.42

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 4.26).

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 $\alpha\beta\gamma \in R$.

Es gilt:

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

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 $w \in L(G)$. □

4.6 Der Cocke-Younger-Kasami-Algorithmus

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

4.6 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, $w = a_1 \dots a_n \in \Sigma^*$.

4.6 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 4.43

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

4.6 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 4.43

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

Damit gilt:

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

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

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

4.6 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 4.43

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

Damit gilt:

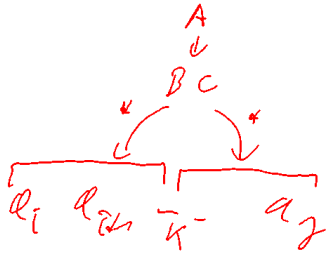
$$w \in L(G) \quad \Leftrightarrow \quad S \in V_{1n}$$

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

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

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

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



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 über $j - i$.

183

4.6 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 4.43

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

Damit gilt:

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

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

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

Korrektheitsbeweis: Induktion über $j - i$.

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

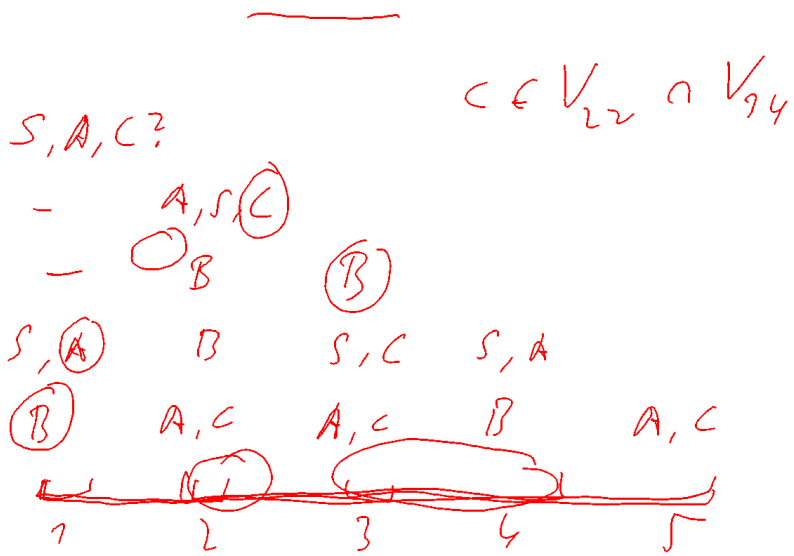
Beispiel 4.44

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

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

Satz 4.45

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



Satz 4.45

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

Satz 4.45

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 4.45

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.

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

Für jede dieser Mengen werden

Satz 4.45

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.

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

Für jede dieser Mengen werden

- $j - i < n$ Werte für k betrachtet,

Satz 4.45

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.

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

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 $C \in V_{k+1,j}$,

Satz 4.45

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.

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

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

Satz 4.45

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.

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

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}: $O(\ )$.

Satz 4.45

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.

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

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}: $O(1)$.

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 $a_i \dots a_j$.

Vorschau

Für CFGs sind folgende Probleme nicht entscheidbar:

- Äquivalenz: $L(G_1) = L(G_2)$?

4.7 Abschlusseigenschaften

Satz 4.46

Seien kontextfreie Grammatiken $G_1 = (V_1, \Sigma_1, P_1, S_1)$ und $G_2 = (V_2, \Sigma_2, P_2, S_2)$ gegeben. Dann kann man in linearer Zeit kontextfreie Grammatiken für

konstruieren.

4.7 Abschlusseigenschaften

Satz 4.46

Seien kontextfreie Grammatiken $G_1 = (V_1, \Sigma_1, P_1, S_1)$ und $G_2 = (V_2, \Sigma_2, P_2, S_2)$ gegeben. Dann kann man in linearer Zeit kontextfreie Grammatiken für

1. $L(G_1) \cup L(G_2)$ $S \to S_1 \mid S_2$

konstruieren.

4.7 Abschlusseigenschaften

Satz 4.46

Seien kontextfreie Grammatiken $G_1 = (V_1, \Sigma_1, P_1, S_1)$ und $G_2 = (V_2, \Sigma_2, P_2, S_2)$ gegeben. Dann kann man in linearer Zeit kontextfreie Grammatiken für

1. $L(G_1) \cup L(G_2)$
2. $L(G_1)L(G_2)$ $S \to S_1 S_2$

konstruieren.

4.7 Abschlusseigenschaften

Satz 4.46

Seien kontextfreie Grammatiken $G_1 = (V_1, \Sigma_1, P_1, S_1)$ und $G_2 = (V_2, \Sigma_2, P_2, S_2)$ gegeben. Dann kann man in linearer Zeit kontextfreie Grammatiken für

1. $L(G_1) \cup L(G_2)$
2. $L(G_1)L(G_2)$
3. $(L(G_1))^*$

$S \to S_1 S \mid \varepsilon$

konstruieren.

4.7 Abschlusseigenschaften

Satz 4.46

Seien kontextfreie Grammatiken $G_1 = (V_1, \Sigma_1, P_1, S_1)$ und $G_2 = (V_2, \Sigma_2, P_2, S_2)$ gegeben. Dann kann man in linearer Zeit kontextfreie Grammatiken für

1. $L(G_1) \cup L(G_2)$
2. $L(G_1)L(G_2)$
3. $(L(G_1))^*$
4. $(L(G_1))^R$

konstruieren.

4.7 Abschlusseigenschaften

Satz 4.46

Seien kontextfreie Grammatiken $G_1 = (V_1, \Sigma_1, P_1, S_1)$ und $G_2 = (V_2, \Sigma_2, P_2, S_2)$ gegeben. Dann kann man in linearer Zeit kontextfreie Grammatiken für

1. $L(G_1) \cup L(G_2)$
2. $L(G_1)L(G_2)$
3. $(L(G_1))^*$
4. $(L(G_1))^R$

konstruieren.

Backus Naur Form

Die Klasse der kontextfreien Sprachen ist also unter Vereinigung, Konkatenation, Stern und Spiegelung abgeschlossen.

$S \to a^* A (B \mid C)$

Satz 4.47

Die Menge der kontextfreien Sprachen ist nicht abgeschlossen unter Durchschnitt oder Komplement.

Satz 4.47

Die Menge der kontextfreien Sprachen ist nicht abgeschlossen unter Durchschnitt oder Komplement.

Beweis:

Satz 4.47

Die Menge der kontextfreien Sprachen ist nicht abgeschlossen unter Durchschnitt oder Komplement.

Beweis: $S \to DC \quad D \to \varepsilon \mid aDb \quad C \to \varepsilon \mid cC$

$$L_1 := \{a^i b^i c^j \mid i, j \in \mathbb{N}\} \text{ ist kontextfrei}$$
$$L_2 := \{a^i b^j c^j \mid i, j \in \mathbb{N}\} \text{ ist kontextfrei}$$

Satz 4.47

Die Menge der kontextfreien Sprachen ist nicht abgeschlossen unter Durchschnitt oder Komplement.

Beweis:

$$L_1 := \{a^i b^i c^j \mid i, j \in \mathbb{N}\} \text{ ist kontextfrei}$$
$$L_2 := \{a^i b^j c^j \mid i, j \in \mathbb{N}\} \text{ ist kontextfrei}$$
$$L_1 \cap L_2 = \{a^i b^i c^i \mid i \in \mathbb{N}\} \text{ ist nicht kontextfrei}$$

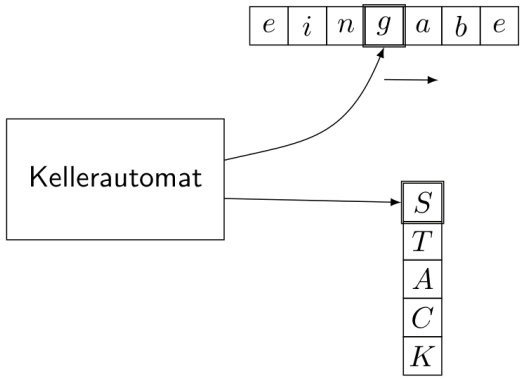
Satz 4.47

Die Menge der kontextfreien Sprachen ist nicht abgeschlossen unter Durchschnitt oder Komplement.

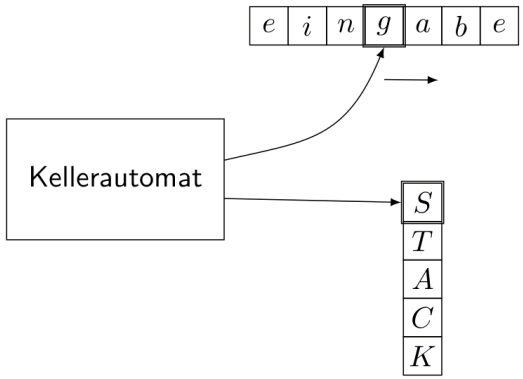
Beweis:

$$L_1 := \{a^i b^i c^j \mid i, j \in \mathbb{N}\} \text{ ist kontextfrei}$$
$$L_2 := \{a^i b^j c^j \mid i, j \in \mathbb{N}\} \text{ ist kontextfrei}$$
$$L_1 \cap L_2 = \{a^i b^i c^i \mid i \in \mathbb{N}\} \text{ ist nicht kontextfrei}$$

Wegen de Morgan (1806–1871) können die CFLs daher auch nicht unter Komplement abgeschlossen sein. □



4.8 Kellerautomaten



191

Anwendungsgebiete von Kellerautomaten:

- Syntaxanalyse von Programmiersprachen

Anwendungsgebiete von Kellerautomaten:

- Syntaxanalyse von Programmiersprachen
- Analyse von Programmen mit Rekursion

192

Definition 4.48

Ein (nichtdeterministischer) Kellerautomat (PDA = Pushdown Automaton) $M = (Q, \Sigma, \Gamma, q_0, Z_0, \delta, F)$ besteht aus

Definition 4.48

Ein (nichtdeterministischer) Kellerautomat (PDA = Pushdown Automaton) $M = (Q, \Sigma, \Gamma, q_0, Z_0, \delta, F)$ besteht aus

- Q endliche Menge von Zuständen

Definition 4.48

Ein (nichtdeterministischer) Kellerautomat (PDA = Pushdown Automaton) $M = (Q, \Sigma, \Gamma, q_0, Z_0, \delta, F)$ besteht aus

- Q endliche Menge von Zuständen
- Σ endliches Eingabealphabet
- Γ endliches Kelleralphabet

$q_0 \in Q$ Anfangszustand

$Z_0 \in \Gamma$ initialer Kellerinhalt

- δ Übergangsfunktion $\delta\colon Q \times (\Sigma \cup \{\epsilon\}) \times \Gamma \to \mathcal{P}_e(Q \times \Gamma^*)$ ($\mathcal{P}_e$ = Menge aller endlichen Teilmengen)

Definition 4.48

Ein (nichtdeterministischer) Kellerautomat (PDA = Pushdown Automaton) $M = (Q, \Sigma, \Gamma, q_0, Z_0, \delta, F)$ besteht aus

- Q endliche Menge von Zuständen
- Σ endliches Eingabealphabet
- Γ endliches Kelleralphabet

$q_0 \in Q$ Anfangszustand

$Z_0 \in \Gamma$ initialer Kellerinhalt

- δ Übergangsfunktion $\delta\colon Q \times (\Sigma \cup \{\epsilon\}) \times \Gamma \to \mathcal{P}_e(Q \times \Gamma^*)$ ($\mathcal{P}_e$ = Menge aller endlichen Teilmengen)

$F \subseteq Q$ Endzustände

Achtung: $\delta(q, a, Z) \ni (q', \alpha)$

- α kann die Länge, 0, 1, und mehr haben

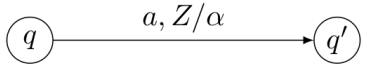
Achtung: $\delta(q, a, Z) \ni (q', \alpha)$

- α kann die Länge, 0, 1, und mehr haben
- Falls $a = \epsilon$: kein Eingabezeichen wird gelesen

Achtung: $\delta(q, a, Z) \ni (q', \alpha)$

- α kann die Länge, 0, 1, und mehr haben
- Falls $a = \epsilon$: kein Eingabezeichen wird gelesen

Eine graphische Notation für $\delta(q, a, Z) \ni (q', \alpha)$:



Kein endlicher Automat!

Definition 4.49
Eine Konfiguration eines Kellerautomaten M ist ein Tripel (q, w, α) mit $q \in Q$, $w \in \Sigma^*$ und $\alpha \in \Gamma^*$.

Definition 4.49

Eine Konfiguration eines Kellerautomaten M ist ein Tripel (q, w, α) mit $q \in Q$, $w \in \Sigma^*$ und $\alpha \in \Gamma^*$.
Die Anfangskonfiguration von M für die Eingabe $w \in \Sigma^*$ ist (q_0, w, Z_0).

Definition 4.49

Eine Konfiguration eines Kellerautomaten M ist ein Tripel (q, w, α) mit $q \in Q$, $w \in \Sigma^*$ und $\alpha \in \Gamma^*$.
Die Anfangskonfiguration von M für die Eingabe $w \in \Sigma^*$ ist (q_0, w, Z_0).

Intuitiv stellt eine Konfiguration (q, w, α) eine "Momentaufnahme" des Kellerautomaten dar:

- Der momentane Zustand ist q.
- Der noch zu lesende Teil der Eingabe ist w.
- Der aktuelle Kellerinhalt ist α
  (das oberste Kellerzeichen ganz links stehend).

Definition 4.50

Die Transitionsrelation $\to_M$ zwischen Konfigurationen:

$$(q, aw, Z\alpha) \to_M (q', w, \beta\alpha) \quad \text{falls } (q', \beta) \in \delta(q, a, Z)$$

Definition 4.50

Die Transitionsrelation $\to_M$ zwischen Konfigurationen:

$$(q, aw, Z\alpha) \to_M (q', w, \beta\alpha) \quad \text{falls } (q', \beta) \in \delta(q, a, Z)$$

$$(q, w, Z\alpha) \to_M (q', w, \beta\alpha) \quad \text{falls } (q', \beta) \in \delta(q, \epsilon, Z)$$

Definition 4.50

Die Transitionsrelation $\rightarrow_M$ zwischen Konfigurationen:

$$(q, aw, Z\alpha) \rightarrow_M (q', w, \beta\alpha) \quad \text{falls } (q', \beta) \in \delta(q, a, Z)$$

$$(q, w, Z\alpha) \rightarrow_M (q', w, \beta\alpha) \quad \text{falls } (q', \beta) \in \delta(q, \epsilon, Z)$$

Intuitive Bedeutung von $(q, w, \alpha) \rightarrow_M (q', w', \alpha')$:

> *Wenn M sich in der Konfiguration (q, w, α) befindet, dann kann er in einen Schritt in die Nachfolgerkonfiguration (q', w', α') übergehen.*

Definition 4.50

Die Transitionsrelation $\rightarrow_M$ zwischen Konfigurationen:

$$(q, aw, Z\alpha) \rightarrow_M (q', w, \beta\alpha) \quad \text{falls } (q', \beta) \in \delta(q, a, Z)$$

$$(q, w, Z\alpha) \rightarrow_M (q', w, \beta\alpha) \quad \text{falls } (q', \beta) \in \delta(q, \epsilon, Z)$$

Intuitive Bedeutung von $(q, w, \alpha) \rightarrow_M (q', w', \alpha')$:

> *Wenn M sich in der Konfiguration (q, w, α) befindet, dann kann er in einen Schritt in die Nachfolgerkonfiguration (q', w', α') übergehen.*

Achtung: eine Konfiguration kann mehrere Nachfolger haben (Nichtdeterminismus!)

Notation $\rightarrow_M^*$, $\rightarrow_M^n$, etc wie üblich

196