SS 2005

Einführung in die Informatik IV

Ernst W. Mayr

Fakultät für Informatik
TU München

http://www14.in.tum.de/lehre/2005SS/info4/index.html.de

2. Mai 2005

Beispiel für die Anwendung des Pumping Lemmas:

Satz 47

$L = \{0^{m^2};\ m \geq 0\}$ *ist nicht regulär.*

Beweis:

Angenommen, L sei doch regulär.

Sei n wie durch das Pumping Lemma gegeben. Wähle $m \geq n$.

Dann gibt es ein r mit $1 \leq r \leq n$, so dass gilt:

$$0^{m^2+ir} \in L \text{ für alle } i \in \mathbb{N}_0 \ .$$

Aber:

$$m^2 < m^2 + r \leq m^2 + m < m^2 + 2m + 1 = (m+1)^2 \ !$$

□

Beispiel für die Anwendung des Pumping Lemmas:

Satz 47

$L = \{0^{m^2};\ m \geq 0\}$ *ist nicht regulär.*

Beweis:

Angenommen, L sei doch regulär.

Sei n wie durch das Pumping Lemma gegeben. Wähle $m \geq n$.
Dann gibt es ein r mit $1 \leq r \leq n$, so dass gilt:

$$0^{m^2+ir} \in L \text{ für alle } i \in \mathbb{N}_0 \ .$$

Aber:

$$m^2 < m^2 + r \leq m^2 + m < m^2 + 2m + 1 = (m+1)^2 \ !$$

□

Beispiel für die Anwendung des Pumping Lemmas:

Satz 47

$L = \{0^{m^2};\ m \geq 0\}$ *ist nicht regulär.*

Beweis:

Angenommen, L sei doch regulär.
Sei n wie durch das Pumping Lemma gegeben. Wähle $m \geq n$.
Dann gibt es ein r mit $1 \leq r \leq n$, so dass gilt:

$$0^{m^2+ir} \in L \text{ für alle } i \in \mathbb{N}_0 \ .$$

Aber:

$$m^2 < m^2 + r \leq m^2 + m < m^2 + 2m + 1 = (m+1)^2 \ !$$

Beispiel für die Anwendung des Pumping Lemmas:

Satz 47

$L = \{0^{m^2};\ m \geq 0\}$ *ist nicht regulär.*

Beweis:

Angenommen, L sei doch regulär.
Sei n wie durch das Pumping Lemma gegeben. Wähle $m \geq n$.
Dann gibt es ein r mit $1 \leq r \leq n$, so dass gilt:

$$0^{m^2+ir} \in L \text{ für alle } i \in \mathbb{N}_0 \ .$$

Aber:

$$m^2 < m^2 + r \leq m^2 + m < m^2 + 2m + 1 = (m+1)^2 \ !$$

□

Denkaufgabe:

$\{a^i b^i;\ i \geq 0\}$ ist nicht regulär.

Definition 48

Sei $L \subseteq \Sigma^*$ eine Sprache. Definiere die Relation $\equiv_L \subseteq \Sigma^* \times \Sigma^*$ durch

$$x \equiv_L y \Leftrightarrow (\forall z \in \Sigma^*)[xz \in L \Leftrightarrow yz \in L]$$

Lemma 49

$\equiv_L$ ist eine rechtsinvariante Äquivalenzrelation.

Dabei bedeutet rechtsinvariant:

$$x \equiv_L y \Rightarrow xu \equiv_L yu \text{ für alle } u \ .$$

Beweis:
Klar! □

Definition 48

Sei $L \subseteq \Sigma^*$ eine Sprache. Definiere die Relation $\equiv_L \subseteq \Sigma^* \times \Sigma^*$ durch

$$x \equiv_L y \Leftrightarrow (\forall z \in \Sigma^*)[xz \in L \Leftrightarrow yz \in L]$$

Lemma 49

$\equiv_L$ *ist eine rechtsinvariante Äquivalenzrelation.*

Dabei bedeutet rechtsinvariant:

$$x \equiv_L y \Rightarrow xu \equiv_L yu \text{ für alle } u \ .$$

Beweis:
Klar! □

Definition 48

Sei $L \subseteq \Sigma^*$ eine Sprache. Definiere die Relation $\equiv_L \subseteq \Sigma^* \times \Sigma^*$ durch

$$x \equiv_L y \Leftrightarrow (\forall z \in \Sigma^*)[xz \in L \Leftrightarrow yz \in L]$$

Lemma 49

$\equiv_L$ ist eine rechtsinvariante Äquivalenzrelation.

Dabei bedeutet rechtsinvariant:

$$x \equiv_L y \Rightarrow xu \equiv_L yu \text{ für alle } u \text{ .}$$

Beweis:
Klar! □

Definition 48

Sei $L \subseteq \Sigma^*$ eine Sprache. Definiere die Relation $\equiv_L \subseteq \Sigma^* \times \Sigma^*$ durch

$$x \equiv_L y \Leftrightarrow (\forall z \in \Sigma^*)[xz \in L \Leftrightarrow yz \in L]$$

Lemma 49

$\equiv_L$ ist eine rechtsinvariante Äquivalenzrelation.

Dabei bedeutet rechtsinvariant:

$$x \equiv_L y \Rightarrow xu \equiv_L yu \text{ für alle } u \text{ .}$$

Beweis:

Klar! □

Satz 50 (Myhill-Nerode)

Sei $L \subseteq \Sigma^*$. Dann sind äquivalent:

1. *L ist regulär*
2. *$\equiv_L$ hat endlichen Index (= Anzahl der Äquivalenzklassen)*
3. *L ist die Vereinigung einiger der endlich vielen Äquivalenzklassen von $\equiv_L$.*

Satz 50 (Myhill-Nerode)

Sei $L \subseteq \Sigma^*$. Dann sind äquivalent:

1. *L ist regulär*
2. *$\equiv_L$ hat endlichen Index (= Anzahl der Äquivalenzklassen)*
3. *L ist die Vereinigung einiger der endlich vielen Äquivalenzklassen von $\equiv_L$.*

Satz 50 (Myhill-Nerode)

Sei $L \subseteq \Sigma^*$. Dann sind äquivalent:

1. *L ist regulär*
2. *$\equiv_L$ hat endlichen Index (= Anzahl der Äquivalenzklassen)*
3. *L ist die Vereinigung einiger der endlich vielen Äquivalenzklassen von $\equiv_L$.*

Satz 50 (Myhill-Nerode)

Sei $L \subseteq \Sigma^*$. Dann sind äquivalent:

1. *L ist regulär*
2. *$\equiv_L$ hat endlichen Index (= Anzahl der Äquivalenzklassen)*
3. *L ist die Vereinigung einiger der endlich vielen Äquivalenzklassen von $\equiv_L$.*

Beweis:

(1)⇒(2):

Sei $L = L(A)$ für einen DFA $A = (Q, \Sigma, \delta, q_0, F)$.

□

Beweis:

(1)⇒(2):

Sei $L = L(A)$ für einen DFA $A = (Q, \Sigma, \delta, q_0, F)$.

Dann gilt

$$\hat{\delta}(q_0, x) = \hat{\delta}(q_0, y) \quad \Rightarrow \quad x \equiv_L y \; .$$

□

Beweis:

(1)⇒(2):

Sei $L = L(A)$ für einen DFA $A = (Q, \Sigma, \delta, q_0, F)$.

Dann gilt

$$\hat{\delta}(q_0, x) = \hat{\delta}(q_0, y) \quad \Rightarrow \quad x \equiv_L y \ .$$

Also gibt es höchstens so viele Äquivalenzklassen, wie der Automat A Zustände hat.

□

Beweis:

(2)⇒(3):

Sei $[x]$ die Äquivalenzklasse von x, $y \in [x]$ und $x \in L$.

□

Beweis:
(2)⇒(3):

Sei $[x]$ die Äquivalenzklasse von x, $y \in [x]$ und $x \in L$.

Dann gilt nach der Definition von $\equiv_L$:

$$y \in L$$

□

Beweis:

(3)⇒(1):

Definiere $A' = (Q', \Sigma, \delta', q_0', F')$ mit

$$
\begin{aligned}
Q' &:= \{[x];\ x \in \Sigma^*\} \qquad (Q' \text{ endlich!}) \\
q_0' &:= [\epsilon] \\
\delta'([x], a) &:= [xa] \quad \forall x \in \Sigma^*, a \in \Sigma \qquad (\text{konsistent!}) \\
F' &:= \{[x];\ x \in L\}
\end{aligned}
$$

□

Beweis:

(3)⇒(1):

Definiere $A' = (Q', \Sigma, \delta', q_0', F')$ mit

$$
\begin{aligned}
Q' &:= \{[x];\ x \in \Sigma^*\} && (Q' \text{ endlich!}) \\
q_0' &:= [\epsilon] \\
\delta'([x], a) &:= [xa] \quad \forall x \in \Sigma^*, a \in \Sigma && (\text{konsistent!}) \\
F' &:= \{[x];\ x \in L\}
\end{aligned}
$$

Dann gilt:

$$L(A') = L$$

□

3.9 Konstruktion minimaler endlicher Automaten

Satz 51

Der nach dem Satz von Myhill-Nerode konstruierte deterministische endliche Automat hat unter allen DFA's für L eine minimale Anzahl von Zuständen.

Beweis:
Sei $A = (Q, \Sigma, \delta, q_0, F)$ mit $L(A) = L$. Dann liefert

$$x \equiv_A y :\Leftrightarrow \delta(q_0, x) = \delta(q_0, y)$$

eine Äquivalenzrelation, die $\equiv_L$ verfeinert.
Also gilt: $|Q| = \text{index}(\equiv_A) \geq \text{index}(\equiv_L) =$ Anzahl der Zustände des Myhill-Nerode-Automaten. □

3.9 Konstruktion minimaler endlicher Automaten

Satz 51

Der nach dem Satz von Myhill-Nerode konstruierte deterministische endliche Automat hat unter allen DFA's für L eine minimale Anzahl von Zuständen.

Beweis:

Sei $A = (Q, \Sigma, \delta, q_0, F)$ mit $L(A) = L$. Dann liefert

$$x \equiv_A y :\Leftrightarrow \delta(q_0, x) = \delta(q_0, y)$$

eine Äquivalenzrelation, die $\equiv_L$ verfeinert.

Also gilt: $|Q| = \text{index}(\equiv_A) \geq \text{index}(\equiv_L) =$ Anzahl der Zustände des Myhill-Nerode-Automaten. □

3.9 Konstruktion minimaler endlicher Automaten

Satz 51

Der nach dem Satz von Myhill-Nerode konstruierte deterministische endliche Automat hat unter allen DFA's für L eine minimale Anzahl von Zuständen.

Beweis:

Sei $A = (Q, \Sigma, \delta, q_0, F)$ mit $L(A) = L$. Dann liefert

$$x \equiv_A y :\Leftrightarrow \delta(q_0, x) = \delta(q_0, y)$$

eine Äquivalenzrelation, die $\equiv_L$ verfeinert.
Also gilt: $|Q| = \text{index}(\equiv_A) \geq \text{index}(\equiv_L) =$ Anzahl der Zustände des Myhill-Nerode-Automaten. □

Algorithmus zur Konstruktion eines minimalen FA

Eingabe: $A(Q, \Sigma, \delta, q_0, F)$ DFA $\quad (L = L(A))$

Ausgabe: Äquivalenzrelation auf Q.

1. Entferne aus Q alle überflüssigen, d.h. alle von q_0 aus nicht erreichbaren Zustände. Wir nehmen nun an, dass Q keine überflüssigen Zustände mehr enthält.
2. Markiere alle Paare $\{q_i, q_j\} \in Q^2$ mit

$$q_i \in F \text{ und } q_j \notin F \text{ bzw. } q_i \notin F \text{ und } q_j \in F\,.$$

Algorithmus zur Konstruktion eines minimalen FA

Eingabe: $A(Q, \Sigma, \delta, q_0, F)$ DFA $\quad (L = L(A))$

Ausgabe: Äquivalenzrelation auf Q.

0. Entferne aus Q alle überflüssigen, d.h. alle von q_0 aus nicht erreichbaren Zustände. Wir nehmen nun an, dass Q keine überflüssigen Zustände mehr enthält.
1. Markiere alle Paare $\{q_i, q_j\} \in Q^2$ mit

$$q_i \in F \text{ und } q_j \notin F \text{ bzw. } q_i \notin F \text{ und } q_j \in F \ .$$

Algorithmus zur Konstruktion eines minimalen FA

Eingabe: $A(Q, \Sigma, \delta, q_0, F)$ DFA $\quad (L = L(A))$

Ausgabe: Äquivalenzrelation auf Q.

0. Entferne aus Q alle überflüssigen, d.h. alle von q_0 aus nicht erreichbaren Zustände. Wir nehmen nun an, dass Q keine überflüssigen Zustände mehr enthält.
1. Markiere alle Paare $\{q_i, q_j\} \in Q^2$ mit

$$q_i \in F \text{ und } q_j \notin F \text{ bzw. } q_i \notin F \text{ und } q_j \in F \ .$$

2. **for** alle unmarkierten Paare $\{q_i, q_j\} \in Q^2$, $q_i \neq q_j$ **do**
   if $(\exists a \in \Sigma)[\{\delta(q_i, a), \delta(q_j, a)\}$ ist markiert$]$ **then**
   markiere $\{q_i, q_j\}$;
   markiere alle $\{q, q'\}$ in $\{q_i, q_j\}$'s Liste und
   rekursiv alle Paare in der Liste von $\{q, q'\}$ usw.
   else
   for alle $a \in \Sigma$ **do**
   if $\delta(q_i, a) \neq \delta(q_j, a)$ **then**
   trage $\{q_i, q_j\}$ in die Liste von $\{\delta(q_i, a), \delta(q_j, a)\}$ ein
   fi
   od
   fi
   od
3. Ausgabe: q äquivalent zu $q' \Leftrightarrow \{q, q'\}$ *nicht* markiert.

2. **for** alle unmarkierten Paare $\{q_i, q_j\} \in Q^2$, $q_i \neq q_j$ **do**
   if $(\exists a \in \Sigma)[\{\delta(q_i, a), \delta(q_j, a)\}$ ist markiert$]$ **then**
   markiere $\{q_i, q_j\}$;
   markiere alle $\{q, q'\}$ in $\{q_i, q_j\}$'s Liste und
   rekursiv alle Paare in der Liste von $\{q, q'\}$ usw.
   else
   for alle $a \in \Sigma$ **do**
   if $\delta(q_i, a) \neq \delta(q_j, a)$ **then**
   trage $\{q_i, q_j\}$ in die Liste von $\{\delta(q_i, a), \delta(q_j, a)\}$ ein
   fi
   od
   fi
   od
3. Ausgabe: q äquivalent zu $q' \Leftrightarrow \{q, q'\}$ *nicht* markiert.

2. **for** alle unmarkierten Paare $\{q_i, q_j\} \in Q^2$, $q_i \neq q_j$ **do**
   if $(\exists a \in \Sigma)[\{\delta(q_i, a), \delta(q_j, a)\}$ ist markiert$]$ **then**
   markiere $\{q_i, q_j\}$;
   markiere alle $\{q, q'\}$ in $\{q_i, q_j\}$'s Liste und
   rekursiv alle Paare in der Liste von $\{q, q'\}$ usw.
   else
   for alle $a \in \Sigma$ **do**
   if $\delta(q_i, a) \neq \delta(q_j, a)$ **then**
   trage $\{q_i, q_j\}$ in die Liste von $\{\delta(q_i, a), \delta(q_j, a)\}$ ein
   fi
   od
   fi
   od
3. Ausgabe: q äquivalent zu $q' \Leftrightarrow \{q, q'\}$ *nicht* markiert.

Satz 52

Obiger Algorithmus liefert einen minimalen DFA für $L(A)$.

Beweis:

Sei $A' = (Q', \Sigma', \delta', q_0', F')$ der konstruierte Äquivalenzklassenautomat.

Offensichtlich ist $L(A) = L(A')$.

Es gilt: $\{q, q'\}$ wird markiert gdw

$$(\exists\, w \in \Sigma^*)[\delta(q, w) \in F \wedge \delta(q', w) \notin F \text{ oder umgekehrt}],$$

wie man durch einfach Induktion über $|w|$ sieht.

Also: Die Anzahl der Zustände von A' (nämlich $|Q'|$) ist gleich dem Index von $\equiv_L$. $\square$

Satz 52

Obiger Algorithmus liefert einen minimalen DFA für $L(A)$.

Beweis:

Sei $A' = (Q', \Sigma', \delta', q_0', F')$ der konstruierte Äquivalenzklassenautomat.

Offensichtlich ist $L(A) = L(A')$.

Es gilt: $\{q, q'\}$ wird markiert gdw

$$(\exists\, w \in \Sigma^*)[\delta(q, w) \in F \wedge \delta(q', w) \notin F \text{ oder umgekehrt}],$$

wie man durch einfach Induktion über $|w|$ sieht.

Also: Die Anzahl der Zustände von A' (nämlich $|Q'|$) ist gleich dem Index von $\equiv_L$. $\square$

Satz 52

Obiger Algorithmus liefert einen minimalen DFA für $L(A)$.

Beweis:

Sei $A' = (Q', \Sigma', \delta', q_0', F')$ der konstruierte Äquivalenzklassenautomat.
Offensichtlich ist $L(A) = L(A')$.
Es gilt: $\{q, q'\}$ wird markiert gdw

$$(\exists\, w \in \Sigma^*)[\delta(q, w) \in F \wedge \delta(q', w) \notin F \text{ oder umgekehrt}],$$

wie man durch einfach Induktion über $|w|$ sieht.
Also: Die Anzahl der Zustände von A' (nämlich $|Q'|$) ist gleich dem Index von $\equiv_L$. □

Satz 52

Obiger Algorithmus liefert einen minimalen DFA für $L(A)$.

Beweis:

Sei $A' = (Q', \Sigma', \delta', q_0', F')$ der konstruierte Äquivalenzklassenautomat.
Offensichtlich ist $L(A) = L(A')$.
Es gilt: $\{q, q'\}$ wird markiert gdw

$$(\exists\, w \in \Sigma^*)[\delta(q, w) \in F \wedge \delta(q', w) \notin F \text{ oder umgekehrt}],$$

wie man durch einfach Induktion über $|w|$ sieht.
Also: Die Anzahl der Zustände von A' (nämlich $|Q'|$) ist gleich dem Index von $\equiv_L$. □

Satz 52

Obiger Algorithmus liefert einen minimalen DFA für $L(A)$.

Beweis:

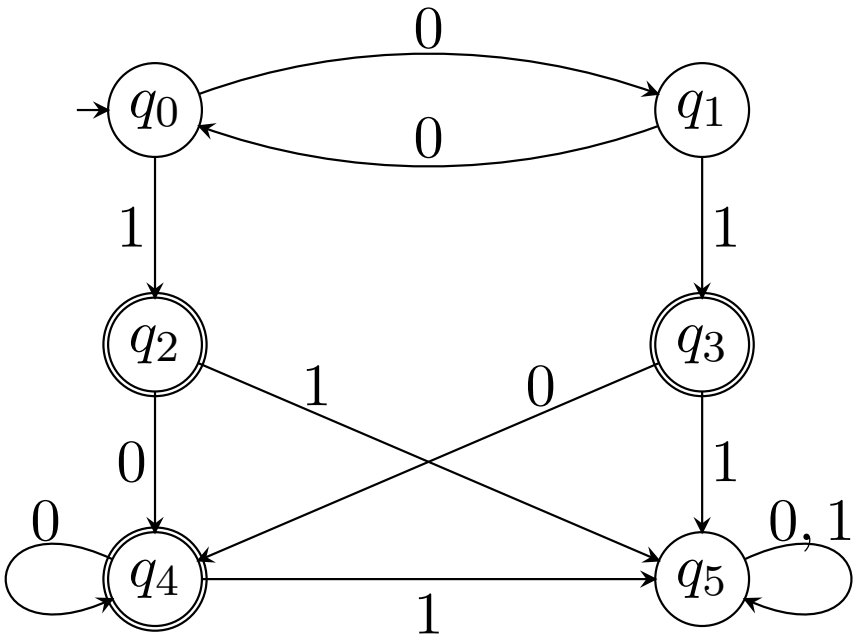
Sei $A' = (Q', \Sigma', \delta', q_0', F')$ der konstruierte Äquivalenzklassenautomat.
Offensichtlich ist $L(A) = L(A')$.
Es gilt: $\{q, q'\}$ wird markiert gdw

$$(\exists\, w \in \Sigma^*)[\delta(q, w) \in F \wedge \delta(q', w) \notin F \text{ oder umgekehrt}],$$

wie man durch einfach Induktion über $|w|$ sieht.
Also: Die Anzahl der Zustände von A' (nämlich $|Q'|$) ist gleich dem Index von $\equiv_L$. □

Beispiel 53

Automat A:



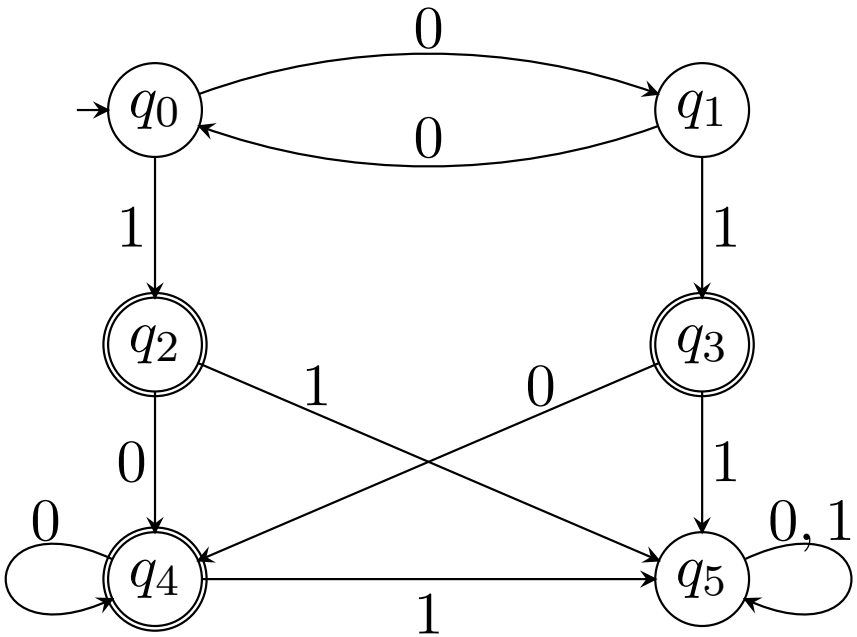
	q_0	q_1	q_2	q_3	q_4	q_5
q_0	/	/	/	/	/	/
q_1		/	/	/	/	/
q_2	×	×	/	/	/	/
q_3	×	×		/	/	/
q_4	×	×			/	/
q_5	×	×	×	×	×	/

Automat A'

$L(A') = 0^*10^*$

Beispiel 53

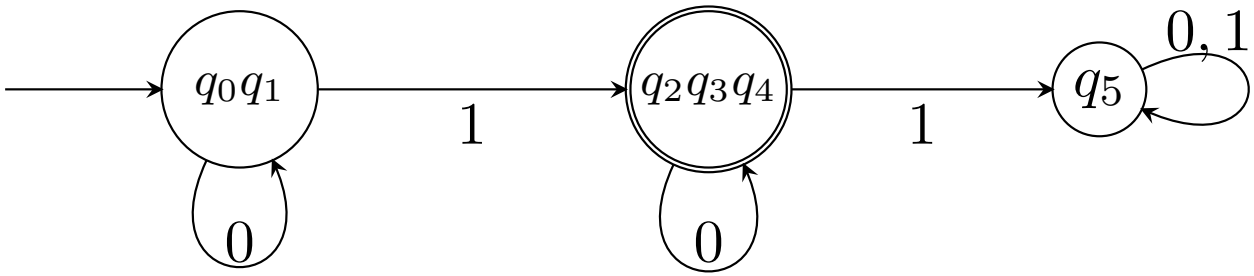
Automat A:



	q_0	q_1	q_2	q_3	q_4	q_5
q_0	/	/	/	/	/	/
q_1		/	/	/	/	/
q_2	×	×	/	/	/	/
q_3	×	×		/	/	/
q_4	×	×			/	/
q_5	×	×	×	×	×	/

Automat A'

$L(A') = 0^*10^*$



Satz 54

Sei $A = (Q, \Sigma, \delta, q_0, F)$ ein DFA. Der Zeitaufwand des obigen Minimalisierungsalgorithmus ist $O(|Q|^2|\Sigma|)$.

Beweis:

Für jedes $a \in \Sigma$ muss jede Position in der Tabelle nur konstant oft besucht werden. □

Satz 54

Sei $A = (Q, \Sigma, \delta, q_0, F)$ ein DFA. Der Zeitaufwand des obigen Minimalisierungsalgorithmus ist $O(|Q|^2|\Sigma|)$.

Beweis:

Für jedes $a \in \Sigma$ muss jede Position in der Tabelle nur konstant oft besucht werden. □

3.10 Entscheidbarkeit

Beispiel 55

Wie wir bereits wissen, ist das Wortproblem für reguläre Sprachen L entscheidbar. Wenn L durch einen deterministischen endlichen Automaten gegeben ist, ist dies sogar in linearer Laufzeit möglich. Allerdings gilt, dass bei der Überführung eines nichtdeterministischen endlichen Automaten in einen deterministischen endlichen Automaten die Komplexität exponentiell zunehmen kann.

Die folgenden Probleme sind für Chomsky-3-Sprachen (also die Familie der regulären Sprachen) entscheidbar:

3.10 Entscheidbarkeit

Beispiel 55

Wie wir bereits wissen, ist das Wortproblem für reguläre Sprachen L entscheidbar. Wenn L durch einen deterministischen endlichen Automaten gegeben ist, ist dies sogar in linearer Laufzeit möglich. Allerdings gilt, dass bei der Überführung eines nichtdeterministischen endlichen Automaten in einen deterministischen endlichen Automaten die Komplexität exponentiell zunehmen kann.

Die folgenden Probleme sind für Chomsky-3-Sprachen (also die Familie der regulären Sprachen) entscheidbar:

Wortproblem: Ist ein Wort w in $L(G)$ (bzw. $L(A)$)?

Das Wortproblem ist für alle Sprachen mit einem Chomsky-Typ größer 0 entscheidbar. Allerdings wächst die Laufzeit exponentiell mit der Wortlänge n. Für Chomsky-2- und Chomsky-3-Sprachen gibt es wesentlich effizientere Algorithmen.

Wortproblem: Ist ein Wort w in $L(G)$ (bzw. $L(A)$)?

Das Wortproblem ist für alle Sprachen mit einem Chomsky-Typ größer 0 entscheidbar. Allerdings wächst die Laufzeit exponentiell mit der Wortlänge n. Für Chomsky-2- und Chomsky-3-Sprachen gibt es wesentlich effizientere Algorithmen.

Leerheitsproblem: Ist $L(G) = \emptyset$?

Wortproblem: Ist ein Wort w in $L(G)$ (bzw. $L(A)$)?

Das Wortproblem ist für alle Sprachen mit einem Chomsky-Typ größer 0 entscheidbar. Allerdings wächst die Laufzeit exponentiell mit der Wortlänge n. Für Chomsky-2- und Chomsky-3-Sprachen gibt es wesentlich effizientere Algorithmen.

Leerheitsproblem: Ist $L(G) = \emptyset$?

Das Leerheitsproblem ist für Sprachen vom Chomsky-Typ 2 und 1 entscheidbar. Für andere Sprachtypen lassen sich Grammatiken konstruieren, für die nicht mehr entscheidbar ist, ob die Sprache leer ist.

Wortproblem: Ist ein Wort w in $L(G)$ (bzw. $L(A)$)?

Das Wortproblem ist für alle Sprachen mit einem Chomsky-Typ größer 0 entscheidbar. Allerdings wächst die Laufzeit exponentiell mit der Wortlänge n. Für Chomsky-2- und Chomsky-3-Sprachen gibt es wesentlich effizientere Algorithmen.

Leerheitsproblem: Ist $L(G) = \emptyset$?

Das Leerheitsproblem ist für Sprachen vom Chomsky-Typ 2 und 3 entscheidbar. Für andere Sprachtypen lassen sich Grammatiken konstruieren, für die nicht mehr entscheidbar ist, ob die Sprache leer ist.

Wortproblem: Ist ein Wort w in $L(G)$ (bzw. $L(A)$)?

Das Wortproblem ist für alle Sprachen mit einem Chomsky-Typ größer 0 entscheidbar. Allerdings wächst die Laufzeit exponentiell mit der Wortlänge n. Für Chomsky-2- und Chomsky-3-Sprachen gibt es wesentlich effizientere Algorithmen.

Leerheitsproblem: Ist $L(G) = \emptyset$?

Das Leerheitsproblem ist für Sprachen vom Chomsky-Typ 2 und 3 entscheidbar. Für andere Sprachtypen lassen sich Grammatiken konstruieren, für die nicht mehr entscheidbar ist, ob die Sprache leer ist.

Wortproblem: Ist ein Wort w in $L(G)$ (bzw. $L(A)$)?

Das Wortproblem ist für alle Sprachen mit einem Chomsky-Typ größer 0 entscheidbar. Allerdings wächst die Laufzeit exponentiell mit der Wortlänge n. Für Chomsky-2- und Chomsky-3-Sprachen gibt es wesentlich effizientere Algorithmen.

Leerheitsproblem: Ist $L(G) = \emptyset$?

Das Leerheitsproblem ist für Sprachen vom Chomsky-Typ 2 und 3 entscheidbar. Für andere Sprachtypen lassen sich Grammatiken konstruieren, für die nicht mehr entscheidbar ist, ob die Sprache leer ist.

Wortproblem: Ist ein Wort w in $L(G)$ (bzw. $L(A)$)?

Das Wortproblem ist für alle Sprachen mit einem Chomsky-Typ größer 0 entscheidbar. Allerdings wächst die Laufzeit exponentiell mit der Wortlänge n. Für Chomsky-2- und Chomsky-3-Sprachen gibt es wesentlich effizientere Algorithmen.

Leerheitsproblem: Ist $L(G) = \emptyset$?

Das Leerheitsproblem ist für Sprachen vom Chomsky-Typ 2 und 3 entscheidbar. Für andere Sprachtypen lassen sich Grammatiken konstruieren, für die nicht mehr entscheidbar ist, ob die Sprache leer ist.

Wortproblem: Ist ein Wort w in $L(G)$ (bzw. $L(A)$)?

Das Wortproblem ist für alle Sprachen mit einem Chomsky-Typ größer 0 entscheidbar. Allerdings wächst die Laufzeit exponentiell mit der Wortlänge n. Für Chomsky-2- und Chomsky-3-Sprachen gibt es wesentlich effizientere Algorithmen.

Leerheitsproblem: Ist $L(G) = \emptyset$?

Das Leerheitsproblem ist für Sprachen vom Chomsky-Typ 2 und 3 entscheidbar. Für andere Sprachtypen lassen sich Grammatiken konstruieren, für die nicht mehr entscheidbar ist, ob die Sprache leer ist.

Endlichkeitsproblem: Ist $|L(G)| \leq \infty$?

Das Endlichkeitsproblem ist für alle regulären Sprachen lösbar.

Lemma 56

Sei n die Pumping-Lemma-Zahl, die zur Sprache L gehört. Dann gilt:

$$|L| = \infty \text{ gdw } (\exists z \in L)[n \leq |z| < 2n] \ .$$

Endlichkeitsproblem: Ist $|L(G)| \leq \infty$?

Das Endlichkeitsproblem ist für alle regulären Sprachen lösbar.

Lemma 56

Sei n die Pumping-Lemma-Zahl, die zur Sprache L gehört. Dann gilt:

$$|L| = \infty \text{ gdw } (\exists z \in L)[n \leq |z| < 2n] \ .$$

Endlichkeitsproblem: Ist $|L(G)| \leq \infty$?

Das Endlichkeitsproblem ist für alle regulären Sprachen lösbar.

Lemma 56

Sei n die Pumping-Lemma-Zahl, die zur Sprache L gehört. Dann gilt:

$$|L| = \infty \text{ gdw } (\exists z \in L)[n \leq |z| < 2n] \; .$$

Endlichkeitsproblem: Ist $|L(G)| \leq \infty$?

Das Endlichkeitsproblem ist für alle regulären Sprachen lösbar.

Lemma 56

Sei n die Pumping-Lemma-Zahl, die zur Sprache L gehört. Dann gilt:

$$|L| = \infty \text{ gdw } (\exists z \in L)[n \leq |z| < 2n] \; .$$

Beweis:

Wir zeigen zunächst $\Leftarrow$:
Aus dem Pumping-Lemma folgt: $z = uvw$ für $|z| \geq n$ und $uv^iw \in L$ für alle $i \in \mathbb{N}_0$. Damit erzeugt man unendlich viele Wörter.

Nun wird $\Rightarrow$ gezeigt:
Dass es ein Wort z mit $|z| \geq n$ gibt, ist klar (es gibt ja unendlich viele Wörter). Mit Hilfe des Pumping-Lemmas lässt sich ein solches Wort auf eine Länge $< 2n$ reduzieren. □

Damit kann das Endlichkeitsproblem auf das Wortproblem zurückgeführt werden.

Beweis:

Wir zeigen zunächst $\Leftarrow$:
Aus dem Pumping-Lemma folgt: $z = uvw$ für $|z| \geq n$ und $uv^iw \in L$ für alle $i \in \mathbb{N}_0$. Damit erzeugt man unendlich viele Wörter.

Nun wird $\Rightarrow$ gezeigt:
Dass es ein Wort z mit $|z| \geq n$ gibt, ist klar (es gibt ja unendlich viele Wörter). Mit Hilfe des Pumping-Lemmas lässt sich ein solches Wort auf eine Länge $< 2n$ reduzieren. □

Damit kann das Endlichkeitsproblem auf das Wortproblem zurückgeführt werden.

Beweis:

Wir zeigen zunächst $\Leftarrow$:
Aus dem Pumping-Lemma folgt: $z = uvw$ für $|z| \geq n$ und $uv^iw \in L$ für alle $i \in \mathbb{N}_0$. Damit erzeugt man unendlich viele Wörter.

Nun wird $\Rightarrow$ gezeigt:
Dass es ein Wort z mit $|z| \geq n$ gibt, ist klar (es gibt ja unendlich viele Wörter). Mit Hilfe des Pumping-Lemmas lässt sich ein solches Wort auf eine Länge $< 2n$ reduzieren. □

Damit kann das Endlichkeitsproblem auf das Wortproblem zurückgeführt werden.