Karlsruher Institut für Technologie

Theoretische Grundlagen der Informatik

Vorlesung am 16.12.2021

Torsten Ueckerdt | 16. Dezember 2021

Evaluation

Online:

- → Email mit TAN von heute.

Ende:

- Freitag, den 17. Dezember um 12:00 Uhr

- **Grammatiken**

Beispiele

Grammatiken sind Regelsysteme, mit denen sich die Wörter einer vorgegebenen Sprache erzeugen lassen.

Beispiele

Grammatiken sind Regelsysteme, mit denen sich die Wörter einer vorgegebenen Sprache erzeugen lassen.

Beispiel 1 - Cliquen

Die Sprache aller Graphen $G = (V, E)$, die eine Clique der Größe $\frac{|V|}{2}$ enthalten, lässt sich aufbauen durch:

1. Wahl der Zahl n für $|V|$
2. Wahl einer Teilmenge der Größe $\frac{|V|}{2}$
3. Zugabe aller Kanten zwischen Knoten aus dieser Teilmenge
4. Zugabe weiterer Kanten

Dieses Regelsystem ist an drei Stellen nichtdeterministisch: 1, 2 und 4.

Beispiele

Grammatiken sind Regelsysteme, mit denen sich die Wörter einer vorgegebenen Sprache erzeugen lassen.

Beispiel 2 - Arithmetische Ausdrücke

- a, $a + a$ und $a \cdot a$ sind arithmetische Ausdrücke (a Symbol aus Alphabet).
- falls A_1 und A_2 arithmetische Ausdrücke sind, so sind auch $(A_1) + (A_2)$ und $(A_1) \cdot (A_2)$ arithmetische Ausdrücke.

Die Klammern sind teilweise überflüssig.

Grammatiken

Eine Grammatik $G = (\Sigma, V, S, R)$ besteht aus vier Komponenten:

- Endliches Alphabet Σ (auch Terminalalphabet genannt);
- Endliche Menge V mit $V \cap \Sigma = \emptyset$ von Variablen (Nichtterminale);
- Startsymbol $S \in V$;
- Endliche Menge R von Ableitungsregeln (Produktionen);

  Dabei ist eine Ableitungsregel ein Paar (ℓ, r), wobei
  - $\ell \in (V \cup \Sigma)^+$
  - $r \in (V \cup \Sigma)^*$.

  Wir schreiben oft auch $\ell \to r$.

Bemerkungen

Bedeutung

- Wenn in einem Wort z das Wort ℓ Teilwort von z ist, so darf ℓ durch r in z ersetzt werden.

Notation

- Wir schreiben $w \longrightarrow z$, wenn w durch Anwendung einer Ableitungsregel in z verwandelt wird.
- Wir schreiben $w \overset{*}{\longrightarrow} z$, wenn w durch Anwendung einer oder mehrerer Ableitungsregeln in z verwandelt wird.

Erzeugte Sprache

- Die von einer Grammatik G **erzeugte Sprache** $L(G)$ ist die Menge aller Wörter $z \in \Sigma^*$, für die $S \overset{*}{\longrightarrow} z$ gilt.

Beispiel

Grammatik $G = (\Sigma, V, S, R)$ für die Menge aller arithmetischen Ausdrücke über a.

$$
\begin{aligned}
\Sigma &= \{(,), a, +, \cdot\} \\
V &= \{S\} \\
R &: \quad S \to (S) + (S) \\
&\qquad S \to (S) \cdot (S) \\
&\qquad S \to a \\
&\qquad S \to a + a \\
&\qquad S \to a \cdot a
\end{aligned}
$$

Es gilt $(a + a) \cdot (a) \in L(G)$ weil $S \to (S) \cdot (S) \to (a + a) \cdot (a)$.

Die Chomsky Hierarchie

- **Typ-0:** Grammatiken ohne weitere Einschränkungen.
- **Typ-1** / **kontextsensitiv:**
  Grammatiken, ausschließlich mit Ableitungsregeln der Form
  - $u \to v$ mit $u \in V^+, v \in ((V \cup \Sigma) \setminus \{S\})^+$ und $|u| \leq |v|$, oder
  - $S \to \varepsilon$
- **Typ-2** / **kontextfrei:**
  Grammatiken, ausschließlich mit Ableitungsregeln der Form
  - $A \to v$ mit $A \in V$ und $v \in (V \cup \Sigma)^*$
- **Typ-3** / **rechtslinear:**
  Grammatiken, ausschließlich mit Ableitungsregeln der Form
  - $A \to v$ mit $A \in V$ und $v = \varepsilon$ oder $v = aB$ mit $a \in \Sigma, B \in V$

Die Chomsky Hierarchie

- **Typ-1 / kontextsensitiv:**
  Grammatiken, ausschließlich mit Ableitungsregeln der Form
  - $u \to v$ mit $u \in V^+, v \in ((V \cup \Sigma) \setminus \{S\})^+$ und $|u| \leq |v|$, oder
  - $S \to \varepsilon$

Bemerkung.

Bei kontextsensitiven Grammatiken kann die Ableitung

$$ABC \to AXYC$$

erlaubt, aber

$$DBC \to DXYC$$

verboten sein.

Chomsky-0 Grammatiken und Semientscheidbarkeit

- **Typ-0:** Grammatiken ohne weitere Einschränkungen.

Satz.

Falls L semi-entscheidbar (rekursiv aufzählbar) ist, so gibt es eine Chomsky-0 Grammatik G mit $L(G) = L$.

Chomsky-0 Grammatiken und Semientscheidbarkeit

- **Typ-0:** Grammatiken ohne weitere Einschränkungen.

Satz.

Falls L semi-entscheidbar (rekursiv aufzählbar) ist, so gibt es eine Chomsky-0 Grammatik G mit $L(G) = L$.

Beweis:

- L semi-entscheidbar $\Rightarrow \exists$ DTM $\mathcal{M}$, die L akzeptiert.
- O.B.d.A. habe $\mathcal{M}$ genau einen akzeptierenden Endzustand q_J.
- O.B.d.A stehen auf dem Band nur Blanks wenn q_J erreicht wird.
- q_0 komme nur in der Anfangskonfiguration vor.

Wir konstruieren eine Grammatik $G = (\Sigma, V = \Gamma \cup Q \cup \{S\}, S, R)$,
die die Berechnung von $\mathcal{M} = (Q, \Sigma, \Gamma, s, \delta, F)$ rückwärts durchläuft.

Beweis - Beschreibung der Grammatik G

- **Rückwärtsrechnung**

  Falls $\delta(q, a) = (q', a', R)$, dann enthält G die Ableitungsregel

  $$a'q' \to qa.$$

  Falls $\delta(q, a) = (q', a', L)$, dann enthält G die Ableitungsregel

  $$q'ba' \to bqa \quad \text{für alle } b \in \Gamma.$$

  Falls $\delta(q, a) = (q', a', N)$, dann enthält G die Ableitungsregel

  $$q'a' \to qa.$$

Beweis - Beschreibung der Grammatik G

- **Erzeugung der akzeptierenden Schlusskonfiguration**

$$S \to q_J, \quad q_J \to \sqcup q_J, \quad q_J \to q_J \sqcup$$

- **Rückwärtsrechnung**

  Falls $\delta(q, a) = (q', a', R)$, dann enthält G die Ableitungsregel

$$a' q' \to qa.$$

  Falls $\delta(q, a) = (q', a', L)$, dann enthält G die Ableitungsregel

$$q' b a' \to bqa \quad \text{für alle } b \in \Gamma.$$

  Falls $\delta(q, a) = (q', a', N)$, dann enthält G die Ableitungsregel

$$q' a' \to qa.$$

Beweis - Beschreibung der Grammatik G

- **Erzeugung der akzeptierenden Schlusskonfiguration**

$$S \to q_J, \quad q_J \to \sqcup q_J, \quad q_J \to q_J \sqcup$$

- **Rückwärtsrechnung**

  Falls $\delta(q, a) = (q', a', R)$, dann enthält G die Ableitungsregel

$$a' q' \to qa.$$

  Falls $\delta(q, a) = (q', a', L)$, dann enthält G die Ableitungsregel

$$q' b a' \to bqa \quad \text{für alle } b \in \Gamma.$$

  Falls $\delta(q, a) = (q', a', N)$, dann enthält G die Ableitungsregel

$$q' a' \to qa.$$

- **Schlussregeln**

  Transformiere $\sqcup \sqcup \ldots \sqcup\, q_0 w_1 \ldots w_n \sqcup \sqcup \ldots \sqcup$ zu $w_1 \ldots w_n$

$$\sqcup q_0 \to q_0, \quad q_0 a \to a q_0, \quad q_0 \sqcup \to q_0, \quad q_0 \to \varepsilon$$

Beweis - Zusammenfassung

- Alle Wörter $w \in L$ können durch G erzeugt werden, indem die Berechnung von $\mathcal{M}$ rückwärts durchlaufen wird.
- Umgekehrt kann G nur Berechnungen von $\mathcal{M}$ rückwärts erzeugen.
- Daher ist $L(G) = L$.

Chomsky-0 Grammatiken und Semientscheidbarkeit

- **Typ-0:** Grammatiken ohne weitere Einschränkungen.

Satz.

Die von Typ-0 Grammatiken G erzeugten Sprachen sind semientscheidbar.

Chomsky-0 Grammatiken und Semientscheidbarkeit

- **Typ-0:** Grammatiken ohne weitere Einschränkungen.

Satz.

Die von Typ-0 Grammatiken G erzeugten Sprachen sind semientscheidbar.

Beweis:

Wir konstruieren zunächst eine NTM $\mathcal{M}$, die $L(G)$ akzeptiert:

1. Schreibe S auf das Band.
2. Wähle eine beliebige anwendbare Ableitungsregel aus.
3. Vergleiche das erzeugte Wort mit der Eingabe w.
4. Bei Gleichheit wird w akzeptiert.
5. Ansonsten springe zu Punkt 2.

Falls $w \in L(G)$, so gibt es eine akzeptierende Berechnung.

Zwischenfazit

Die Klasse der semientscheidbaren Sprachen ist genau

- die Klasse der von DTMs akzeptierten Sprachen,
- die Klasse der von NTMs akzeptierten Sprachen,
- die Klasse der von Typ-0 Grammatiken erzeugten Sprachen.

Bemerkung.

- Das Wortproblem für Typ-0 Grammatiken ist unentscheidbar.
- Als Grundlage für Programmiersprachen sind die Typ-0 Grammatiken also sicherlich zu allgemein.

Chomsky-3 Grammatiken und reguläre Sprachen

- **Typ-3 / rechtslinear:**
  Grammatiken, ausschließlich mit Ableitungsregeln der Form
  - $A \to v$ mit $A \in V$ und $v = \varepsilon$ oder $v = aB$ mit $a \in \Sigma, B \in V$

Satz.

Die Klasse der von endlichen Automaten akzeptierten Sprachen ist genau die Klasse der von Chomsky-3 Grammatiken erzeugten Sprachen.

Beweis: DEA $\Longrightarrow$ Typ-3 Grammatik

"$\Rightarrow$": Zu einem DEA $\mathcal{A}_L = (Q, \Sigma, \delta, q_0, F)$, gibt es eine Typ-3 Grammatik G_L, die $L(\mathcal{A}_L)$ erzeugt.

Beweis: DEA $\Longrightarrow$ Typ-3 Grammatik

"$\Rightarrow$": Zu einem DEA $\mathcal{A}_L = (Q, \Sigma, \delta, q_0, F)$, gibt es eine Typ-3 Grammatik G_L, die $L(\mathcal{A}_L)$ erzeugt.

G_L sei definiert durch:

- $V := Q$;
- $S := q_0$;
- R enthält die Regel
  - $q \to \varepsilon$ für alle $q \in F$,
  - $q \to aq'$, falls $\delta(q, a) = q'$.

Beweis: DEA $\Longrightarrow$ Typ-3 Grammatik

"$\Rightarrow$": Zu einem DEA $\mathcal{A}_L = (Q, \Sigma, \delta, q_0, F)$, gibt es eine Typ-3 Grammatik G_L, die $L(\mathcal{A}_L)$ erzeugt.

G_L sei definiert durch:

- $V := Q$;
- $S := q_0$;
- R enthält die Regel
  - $q \to \varepsilon$ für alle $q \in F$,
  - $q \to aq'$, falls $\delta(q, a) = q'$.

Für $w = w_1 \cdots w_n$ durchlaufe $\mathcal{A}_L$ die Zustände $q_0, \ldots, q_n \in Q$ mit $q_n \in F$.
Dann gilt:

$$q_0 \to w_1 q_1 \to w_1 w_2 q_2 \to \ldots \to w q_n \to w$$

Beweis: DEA $\Longrightarrow$ Typ-3 Grammatik

"$\Rightarrow$": Zu einem DEA $\mathcal{A}_L = (Q, \Sigma, \delta, q_0, F)$, gibt es eine Typ-3 Grammatik G_L, die $L(\mathcal{A}_L)$ erzeugt.

G_L sei definiert durch:

- $V := Q$;
- $S := q_0$;
- R enthält die Regel
  - $q \to \varepsilon$ für alle $q \in F$,
  - $q \to aq'$, falls $\delta(q, a) = q'$.

Für $w = w_1 \cdots w_n$ durchlaufe $\mathcal{A}_L$ die Zustände $q_0, \ldots, q_n \in Q$ mit $q_n \in F$.
Dann gilt:

$$q_0 \to w_1 q_1 \to w_1 w_2 q_2 \to \ldots \to w q_n \to w$$

Außerdem gibt es für alle Ableitungen von G_L eine entsprechende akzeptierende Berechnung des endlichen Automatens $\mathcal{A}_L$.

Beweis: Typ-3 Grammatik $\Longrightarrow$ NEA

"$\Longleftarrow$": Zu einer Typ-3 Grammatik G_L gibt es einen NEA $\mathcal{A}_L := (Q, \Sigma, \delta, q_0, F)$, der $L(G_L)$ akzeptiert.

Beweis: Typ-3 Grammatik $\Longrightarrow$ NEA

"$\Leftarrow$": Zu einer Typ-3 Grammatik G_L gibt es einen NEA $\mathcal{A}_L := (Q, \Sigma, \delta, q_0, F)$, der $L(G_L)$ akzeptiert.

- $Q := V$;
- $q_0 := S$;
- $F := \{A \in V : (A \to \varepsilon) \in R\}$
- $\delta(A, a) := \{B : (A \to aB) \in R\}$.

Beweis: Typ-3 Grammatik $\Longrightarrow$ NEA

"$\Longleftarrow$": Zu einer Typ-3 Grammatik G_L gibt es einen NEA $\mathcal{A}_L := (Q, \Sigma, \delta, q_0, F)$, der $L(G_L)$ akzeptiert.

- $Q := V$;
- $q_0 := S$;
- $F := \{A \in V : (A \to \varepsilon) \in R\}$
- $\delta(A, a) := \{B : (A \to aB) \in R\}$.

Für $w = w_1 \dots w_n \in L$ hat die Ableitung von w mittels G_L das Aussehen:

$$S \to w_1 A_1 \to w_1 w_2 A_2 \to \dots \to w A_n \to w$$

$\mathcal{A}_L$ kann bei Eingabe $w = w_1 \dots w_n$ folgende Abarbeitung durchlaufen:

$$S \xrightarrow{w_1} A_1 \xrightarrow{w_2} A_2 \xrightarrow{w_3} \dots \xrightarrow{w_{n-1}} A_{n-1} \xrightarrow{w_n} A_n,$$

wobei $A_n \in F$, also w akzeptiert wird.

Beweis: Typ-3 Grammatik $\Longrightarrow$ NEA

"$\Leftarrow$": Zu einer Typ-3 Grammatik G_L gibt es einen NEA $\mathcal{A}_L := (Q, \Sigma, \delta, q_0, F)$, der $L(G_L)$ akzeptiert.

- $Q := V$;
- $q_0 := S$;
- $F := \{A \in V : (A \to \varepsilon) \in R\}$
- $\delta(A, a) := \{B : (A \to aB) \in R\}$.

Für $w = w_1 \dots w_n \in L$ hat die Ableitung von w mittels G_L das Aussehen:

$$S \to w_1 A_1 \to w_1 w_2 A_2 \to \dots \to w A_n \to w$$

$\mathcal{A}_L$ kann bei Eingabe $w = w_1 \dots w_n$ folgende Abarbeitung durchlaufen:

$$S \xrightarrow{w_1} A_1 \xrightarrow{w_2} A_2 \xrightarrow{w_3} \dots \xrightarrow{w_{n-1}} A_{n-1} \xrightarrow{w_n} A_n,$$

wobei $A_n \in F$, also w akzeptiert wird.
Außerdem gibt es für alle akzeptierenden Berechnungen von $\mathcal{A}_L$ eine entsprechende Ableitung in G_L.

Bemerkung

- Wir wissen, dass die Sprache der korrekten Klammerausdrücke nicht regulär ist.
- Typ-3 Grammatiken sind also zu einschränkend, um syntaktisch korrekte Programme zu beschreiben.

Fazit: Programmiersprachen sollten echt zwischen Typ-0 und Typ-3 angesiedelt sein.

Chomsky-1 Grammatiken bzw. kontextsensitive Sprachen

Annahme:

- Die Eingabe einer Turingmaschine steht auf einem separaten Read-Only-Band.
- Der Speicherbedarf einer Turingmaschine wird nur anhand des Arbeitsbandes gemessen.

Definition.

- $\mathcal{DTAPE}(s(n))$ ist die Klasse der Sprachen L, für die es eine **DTM** mit Platzbedarf $s(n)$ (bei Eingabelänge n) gibt, die L akzeptiert.
- $\mathcal{NTAPE}(s(n))$ ist die Klasse der Sprachen L, für die es eine **NTM** mit Platzbedarf $s(n)$ (bei Eingabelänge n) gibt, die L akzeptiert.

Offensichtlich gilt

$$\mathcal{DTAPE}(s(n)) \subseteq \mathcal{NTAPE}(s(n))$$

Außerdem gilt

$$\mathcal{NTAPE}(n) = \mathcal{NTAPE}(f(n))$$

für alle $f(n) \in \theta(n)$.

Chomsky-1 Grammatiken und $\mathcal{NTAPE}(n)$

Satz.

Die Klasse der von Chomsky-1 Grammatiken erzeugten Sprachen stimmt mit der Klasse $\mathcal{NTAPE}(n)$ überein.

Ohne Beweis.

Chomsky-1 Grammatiken und $\mathcal{NTAPE}(n)$

Satz.

Die Klasse der von Chomsky-1 Grammatiken erzeugten Sprachen stimmt mit der Klasse $\mathcal{NTAPE}(n)$ überein.

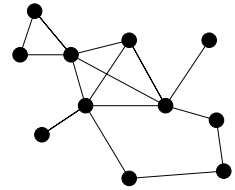
Ohne Beweis.

Bemerkung:
Es ist offen, ob $\mathcal{NTAPE}(n) = \mathcal{DTAPE}(n)$ ist.

Wiederholung: Das Problem CLIQUE

Definition.

Eine **Clique** in einem Graphen $G = (V, E)$ ist eine Menge $V' \subseteq V$ so, dass für alle $i, j \in V', i \neq j$, gilt: $\{i, j\} \in E$.



Entscheidungsproblem CLIQUE

Gegeben: Graph $G = (V, E)$
Parameter $K \in \mathbb{N}$

Frage: Gibt es in G eine Clique der Größe mindestens K?

CLIQUE ist in $\mathcal{DTAPE}(n)$

Satz.

Das Cliquen-Problem gehört zu $\mathcal{DTAPE}(n)$.

Beweis:

- Gegeben sei $G = (V, E)$ mit $V = \{1, \ldots, n\}$ und $1 \leq K \leq n$.
- Benutze n-Vektor $c \in \{0,1\}^n$ mit der Bedeutung für eine Menge C

$$c_i = 1 \quad \Longleftrightarrow \quad i \in C.$$

- Teste für jeden Vektor $c \in \{0,1\}^n$, ob die zugehörige Menge $C \subseteq V$ eine Clique der Größe K in G ist:
  - Zähle die Einsen in c.
  - Überprüfe für jedes Paar c_i, c_j mit $c_i = c_j = 1$, ob $\{i, j\} \in E$.

CLIQUE ist in $\mathcal{DTAPE}(n)$

Satz.

Das Cliquen-Problem gehört zu $\mathcal{DTAPE}(n)$.

Beweis:

Alle Vektoren $c \in \{0,1\}^n$ können nacheinander, beginnend mit $(0,0,\ldots,0)$, durchgetestet werden, wobei:

- nach einem positiven Test (G,K) akzeptiert wird;
- nach einem negativen Test der Vektor durch seinen lexikalischen Nachfolger überschrieben wird.

Dazu wird insgesamt nur linearer Speicherplatz benötigt.

Kontextsensitive Grammatiken: Bemerkungen

- Solange $\mathcal{P} \neq \mathcal{NP}$, kann das Wortproblem für kontextsensitive Grammatiken im Allgemeinen nicht in polynomialer Zeit entschieden werden.

Kontextsensitive Grammatiken: Bemerkungen

- Solange $\mathcal{P} \neq \mathcal{NP}$, kann das Wortproblem für kontextsensitive Grammatiken im Allgemeinen nicht in polynomialer Zeit entschieden werden.
- Für jede Chomsky-1 Grammatik gibt es eine äquivalente Chomsky-1 Grammatik, bei der alle Regeln folgende Form haben:
  - $A \to C$
  - $A \to CD$
  - $AB \to CD$
  - $A \to a$
  - $S \to \varepsilon$

  wobei jeweils $A, B \in V$, $C, D \in V \backslash \{S\}$ und $a \in \Sigma$.

Kontextsensitive Grammatiken: Bemerkungen

- Solange $\mathcal{P} \neq \mathcal{NP}$, kann das Wortproblem für kontextsensitive Grammatiken im Allgemeinen nicht in polynomialer Zeit entschieden werden.
- Für jede Chomsky-1 Grammatik gibt es eine äquivalente Chomsky-1 Grammatik, bei der alle Regeln folgende Form haben:
  - $A \to C$
  - $A \to CD$
  - $AB \to CD$
  - $A \to a$
  - $S \to \varepsilon$

  wobei jeweils $A, B \in V$, $C, D \in V \setminus \{S\}$ und $a \in \Sigma$.
- Statt Regeln

$$S \to \alpha \text{ und } S \to \beta$$

  schreiben wir abkürzend

$$S \to \alpha \mid \beta$$

Typ-2 / Kontextfreie Grammatiken

Grammatiken, ausschließlich mit Ableitungsregeln der Form

$$A \to v \quad \text{mit } A \in V \text{ und } v \in (V \cup \Sigma)^*$$

Typ-2 Grammatiken: Beispiel 1

Die Sprache

$$L = \{0^n 1^n : n \geq 1\}$$

Typ-2 Grammatiken: Beispiel 1

Die Sprache

$$L = \{0^n 1^n : n \geq 1\}$$

wird erzeugt durch die Grammatik

$$\begin{aligned} V &= \{S\} \\ \Sigma &= \{0, 1\} \\ R &= \{S \to 01 \mid 0S1\} \, . \end{aligned}$$

Typ-2 Grammatiken: Beispiel 2

Sei $L = \{w \in \{0,1\}^* : w = w^R\}$ die Sprache der Palindrome über $\{0,1\}$.

Es gilt:

1. $0, 1, \varepsilon$ sind Palindrome
2. falls w Palindrom, so auch $0w0$ und $1w1$
3. alle Palindrome lassen sich durch endliche viele Anwendungen von (1) und (2) erzeugen

Typ-2 Grammatiken: Beispiel 2

Sei $L = \{w \in \{0,1\}^* : w = w^R\}$ die Sprache der Palindrome über $\{0,1\}$.

Es gilt:

1. $0, 1, \varepsilon$ sind Palindrome
2. falls w Palindrom, so auch $0w0$ und $1w1$
3. alle Palindrome lassen sich durch endliche viele Anwendungen von (1) und (2) erzeugen

Zugehörige kontextfreie Grammatik:

$$\begin{aligned} V &= \{S\} \\ \Sigma &= \{0,1\} \\ R &= \{S \to \varepsilon \mid 0 \mid 1, \\ &\qquad S \to 0S0 \mid 1S1\} \end{aligned}$$

Typ-2 Grammatiken: Beispiel 3

Sei L die Sprache aller $w \in \{0,1\}^+$ bei denen die Anzahl der Nullen gleich der Anzahl der Einsen ist.

Typ-2 Grammatiken: Beispiel 3

Sei L die Sprache aller $w \in \{0,1\}^+$ bei denen die Anzahl der Nullen gleich der Anzahl der Einsen ist.

Setze

$$\begin{aligned} V &= \{S, A, B\} \\ \Sigma &= \{0, 1\} \\ R &= \{S \to 0B \mid 1A, \\ &\quad A \to 0 \mid 0S \mid 1AA, \\ &\quad B \to 1 \mid 1S \mid 0BB\} \end{aligned}$$

Durch Induktion über die Länge der durch G erzeugten Wörter lässt sich beweisen, dass $L = L(G)$.