Kontextfreie Sprachen: Operationen und Chomsky-Normalform

Prof. Dr. David Sabel

LFE Theoretische Informatik

Letzte Änderung der Folien: 31. Mai 2022

Kontextfreie Sprachen

Zur Erinnerung:

- Kontextfreie Sprachen (CFLs) werden von einer kontextfreien Grammatik (CFG) erzeugt
- Das sind die Typ 2-Grammatiken
- Bedingung: Alle linken Seiten der Produktionen bestehen aus genau einer Variablen, d.h. sie sind von der Form $A \to r$.

Motivation

- Kontextfreie Sprachen sind insbesondere nützlich um Sprachen mit Klammerungen zu beschreiben
- Die Syntax von Programmiersprachen wird meist mit einer kontextfreien Grammatik angegeben

Beispiele:

- $G = (\{E, M, Z\}, \{+, *, (,)\} \cup \{0, \ldots, 9\}, P, E)$ mit

$$\begin{aligned} P = \{ & E \to M \mid E + M, \\ & M \to Z \mid M * Z, \\ & Z \to N \mid (E), \\ & N \to 1D \mid \ \ldots \ \mid 9D, \\ & D \to 0D \mid \ \ldots \ \mid 9D \mid \varepsilon \} \end{aligned}$$

- $L = \{a^j b^j \mid j \in \mathbb{N}\}$ ist kontextfrei: Produktionen $\{S \to \varepsilon \mid T, T \to aTb \mid ab\}$ mit S als Startsymbol erzeugen L

Einfache Operationen auf CFGs

- Wir definieren Operationen auf CFGs, welche die erzeugte Sprache unverändert lassen
- Die Operationen werden danach als Hilfsmittel wiederverwendet (insbesondere bei der Berechnung von Normalformen).

Sprechweisen

Definition (links- bzw. rechts-rekursive Produktion)

Eine Produktion nennt man **links-rekursiv**, wenn sie von der Form

$$A \to Au$$

ist, und **rechts-rekursiv**, wenn sie von der Form

$$A \to uA$$

ist, wobei in beiden Fällen u eine Satzform ist.

Elimination der Links-Rekursion

Lemma (Elimination von Links-Rekursion)

Sei $G = (V, \Sigma, P, S)$, $P = P' \cup \underbrace{\{A \to Au_1 \mid \cdots \mid Au_n \mid w_1 \mid \cdots \mid w_m\}}_{P''}$ eine CFG mit

- P'' sind alle Produktionen in P mit A als linker Seite und
- die Satzformen $w_1, \ldots, w_m$ beginnen alle nicht mit A.

Es gilt $L(G) = L(G')$ für $G' = (V \cup \{B\}, \Sigma, P' \cup P''', S)$ mit B neue Variable und
$P''' = \{A \to w_1 B \mid \cdots \mid w_m B \mid w_1 \mid \cdots \mid w_m,$
$B \to u_1 \mid \ldots \mid u_n \mid u_1 B \mid \ldots \mid u_n B\}$

Beweis: Ausführlicher Beweis im Skript
Wesentliche Idee:
Ersetze Linksableitungsschritte mit G: $x_1 A x_2 \Rightarrow_G^* x_1 w_r u_{f(1)} \cdots u_{f(k)} x_2$ durch
Rechtsableitungsschritte mit G': $x_1 A x_2 \Rightarrow_{G'}^* x_1 w_r u_{f(1)} \cdots u_{f(k)} x_2$

Beispiel

Die CFG $G = (\{A, C\}, \{b, c, d\}, \{A \to ACA \mid bb, C \to Ccc \mid d\}, A)$ hat links-rekursive Produktionen.

Entfernen der Links-Rekursion für A ergibt die CFG

$$G' = (\{A, B, C\}, \{b, c, d\}, \{A \to bbB \mid bb, B \to CA \mid CAB, C \to Ccc \mid d\}, A)$$

Anschließendes Entfernen der Links-Rekursion für C ergibt die CFG

$$G'' = (\{A, B, C, D\}, \{b, c, d\},$$
$$\{A \to bbB \mid bb, B \to CA \mid CAB, C \to d \mid dD, D \to cc \mid ccD\}, A)$$

Inlining von Produktionen

Lemma (Inlining von Produktionen)

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

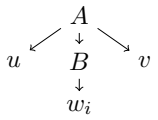
- $A \to uBv \in P$,
- $B \to w_1 \mid \ldots \mid w_n$ alle Regeln mit B als linker Seite

und sei $G' = (V, \Sigma, P \setminus \{A \to uBv\} \cup \{A \to uw_1v \mid \ldots \mid uw_nv\}, S)$.

Dann erzeugen G und G' dieselbe Sprache, d. h. $L(G) = L(G')$.

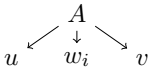
Beweis: Das folgt durch Modifizieren der Syntaxbäume zur Ableitung mit G bzw. G':

tausche alle Baumabschnitte



(im Syntaxbaum mit G)

durch



(im Syntaxbaum mit G')

Sharing von Satzformen mit neuen Produktionen

Lemma (Sharing von Satzformen mit neuen Produktionen)

Sei G eine CFG mit $G = (V, \Sigma, P \cup \{A \to w_1 \cdots w_n\}, S)$.
Seien $B_1, \ldots, B_n$ neue Variablen (d. h. $V \cap \{B_1, \ldots, B_n\} = \emptyset$) und sei
$G' = (V \cup \{B_1, \ldots, B_n\}, \Sigma, P \cup \{A \to B_1 \cdots B_n, B_1 \to w_1, \ldots, B_n \to w_n\}, S)$.
Dann gilt $L(G) = L(G')$.

Beweis:

"$\subseteq$": Konstruiere aus $S \Rightarrow_G^* w$ Ableitung $S \Rightarrow_{G'}^* w$: Übersetze jeden Schritt $uAv \Rightarrow_G uw_1 \cdots w_n v$ in $uAv \Rightarrow_{G'} uB_1 \cdots B_n v \Rightarrow_{G'}^n uw_1 \cdots w_n v$.

"$\supseteq$": Betrachte den Syntaxbaum für $S \Rightarrow_{G'}^* w$.
Identifiziere die Anwendungen der Regeln $A \to B_1 \cdots B_n$, $B_i \to w_i$, und modifziere Syntaxbaum durch Anwendung der Regel $A \to w_1 \cdots w_n$.
Lese Ableitung $S \Rightarrow_G^* w$ ab.

Normalformen

- Normalformen von Grammatiken fordern eine spezielle Form der Produktionen
- Nützlich, wenn man Grammatiken analysiert oder Algorithmen auf Grammatiken formuliert
- Man muss dann nur diese Form (statt aller erlaubten) von Produktionen betrachten
- Wir betrachten zwei Normalformen
  - Chomsky-Normalform
  - Greibach-Normalform

Die Chomsky-Normalform

Definition (Chomsky-Normalform)

Eine CFG $G = (V, \Sigma, P, S)$ mit $\varepsilon \notin L(G)$ ist in **Chomsky-Normalform**,
wenn für jede Produktion $A \to w \in P$ gilt: $w = a \in \Sigma$ oder $w = BC$ mit $B, C \in V$.

Beispiel:

- Die CFG $G = (\{A\}, \{(,), [,]\}, \{A \to (A) \mid () \mid [A] \mid [] \mid A\,A\}, A)$
  ist **nicht** in Chomsky-Normalform
  (nur die Produktion $A \to A\,A$ passt zum vorgeschriebenen Format).
- Die CFG $G' = (\{A, B, C, D, E, F, G\}, \{(,), [,]\}, P, A)$ mit

$$P = \{A \to BF \mid BC \mid DG \mid DE \mid AA,\\ B \to (, C \to), D \to [, E \to], F \to AC, G \to AE\}$$

  ist in Chomsky-Normalform (und erzeugt die gleiche Sprache wie G).

Eigenschaften der Chomsky-Normalform

Sei G eine CFG in Chomsky-Normalform, dann gilt:

- Syntaxbäume zu Ableitungen mit G sind immer **Binärbäume**
- Ableitungen eines Worts $w \in L(G)$ bestehen immer genau aus $2 \cdot |w| - 1$ Ableitungsschritten.

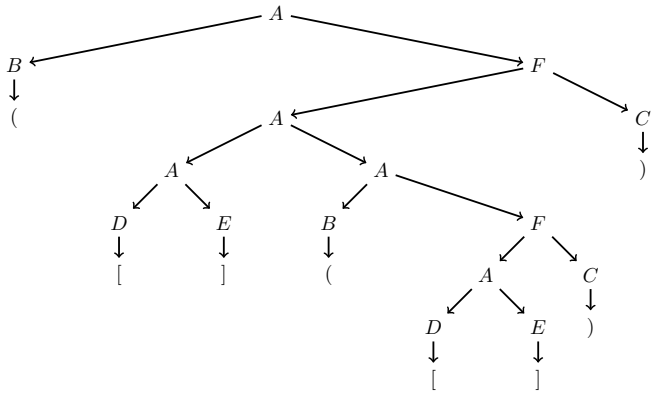
Beispiel:
$G' = (\{A, B, C, D, E, F, G\}, \{(,), [,]\}, P, A)$ mit

$$P = \{A \to BF \mid BC \mid DG \mid DE \mid AA,\\ B \to (, C \to), D \to [, E \to], F \to AC, G \to AE\}$$

Ableitung von $([]([]))$:
$A \Rightarrow BF \Rightarrow (F \Rightarrow (AC \Rightarrow (AAC \Rightarrow (DEAC \Rightarrow ([EAC$
$\Rightarrow ([]AC \Rightarrow ([]BFC \Rightarrow ([](FC \Rightarrow ([](ACC \Rightarrow ([](DECC$
$\Rightarrow ([]([ECC \Rightarrow ([]([]CC \Rightarrow ([]([])C \Rightarrow ([]([]))$

Eigenschaften der Chomsky-Normalform (2)

Der Syntaxbaum dazu:



Herstellen der Chomsky-Normalform

Theorem

Für CFGs G mit $\varepsilon \notin L(G)$ kann eine äquivalente CFG in Chomsky-Normalform berechnet werden.

Herstellen der Chomsky-Normalform

Jede CFG (mit $\varepsilon \notin L(G)$) kann in Chomsky-Normalform gebracht werden. Das Verfahren geht in mehreren Schritten vor:

1. Entfernen von ε-Produktionen (kennen wir bereits)
2. Entfernen von Einheitsproduktionen (Produktionen $A \to B$)
3. Sharen aller Terminale a in rechten Seiten, die nicht nur aus a bestehen durch neue Produktionen $A \to a$
4. Alle Produktionen $A \to B_1 \cdots B_m$ mit $m > 2$ in mehrere zerlegen: $A \to B_1 C_1, C_1 \to B_2 C_3, \ldots, C_{m-2} \to B_{m-1} B_m$

Entfernen von Einheitsproduktionen

- Intuitiv ist klar, dass $A \to B$ entfernt werden kann: Wenn erst $A \to B$, dann $B \to w$ angewendet wird, kann man auch gleich $A \to w$ anwenden.
- Algorithmisch zu beachten:
  - Eliminiere in der richtigen Reihenfolge:

    Wenn $A \to B$ und $B \to C$, dann ist Ersetzen von $A \to B$ durch $A \to C$ nicht zielführend.
  - Zyklen $A \to B$ und $B \to A$ müssen vorher entfernt werden!

Algorithmus 5: Entfernen von Einheitsproduktionen

Eingabe: Eine CFG $G = (V, \Sigma, P, S)$
Beginn
Erzeuge gerichteten Graph $D = (V, E)$, mit $(A, B) \in E$ für jede Einheitsproduktion $A \to B \in P$;
solange *es einen Zyklus* $(A_1, A_2), \ldots, (A_{n-1}, A_n), (A_n, A_1) \in E$ *gibt* **tue**
$P := P \setminus \{A_1 \to A_2, \ldots, A_{n-1} \to A_n, A_n \to A_1\}$; /* entferne zykl. Regeln */
$P := P[A_1/A_2, \ldots, A_1/A_n]$; /* ersetze alle Vorkommen von A_i durch A_1 für $i = 2, \ldots, n$ */
$V := V \setminus \{A_2, A_3, \ldots, A_n\}$; /* lösche $A_2, \ldots, A_n$ */
$S := S[A_1/A_2, \ldots, A_1/A_n]$; /* ersetze Startsymbol durch A_1, falls es A_i, $2 \leq i \leq n$ war */
$E := E \setminus \{(A_1, A_2), \ldots, (A_{n-1}, A_n), (A_n, A_1)\}$; /* entferne Zyklus aus Graph */
$E := E[A_1/A_2, \ldots, A_1/A_n]$; /* ersetze A_i durch A_1 für $i = 2, \ldots, n$ in anderen Kanten */
Sortiere D topologisch und nummeriere die Variablen in V durch (und benenne entsprechend in E, P, S um), so dass gilt: $A_i \to A_j$ impliziert $i < j$;
Sei $V = \{A_1, \ldots, A_k\}$;
für *i=k bis* 1 **tue**
wenn $A_i \to A_j \in P$ **dann**
seien $A_j \to w_1, \ldots, A_j \to w_m$ alle Produktionen mit A_j als linker Seite;
$P := P \cup \{A_i \to w_1, \ldots, A_i \to w_m\}$;
$P := P \setminus \{A_i \to A_j\}$;
Gib die so entstandene Grammatik als G' aus;

Korrektheit von Algorithmus 5 (1)

Satz

Algorithmus 5 berechnet bei Eingabe einer CFG G mit $\varepsilon \notin L(G)$ eine CFG G', die keine Einheitsproduktionen hat, sodass gilt $L(G) = L(G')$. Wenn G keine ε-Produktionen hat, dann hat auch G' keine ε-Produktionen.

Beweis: Wir zeigen:

1. Das Entfernen eines Zyklus verändert die erzeugte Sprache nicht
2. Das Entfernen einer Einheitsproduktion $A_i \to A_j$ in der rückwärts-laufenden für-Schleife ändert die erzeugte Sprache nicht.

   bereits gezeigt, da die Operation „Inlining von Produktionen" korrekt.
3. Der Algorithmus terminiert und führt nie Einheits- oder ε-Produktionen ein.

Korrektheit von Algorithmus 5: Entfernen von Zyklen ist korrekt

Beweisskizze (ausführlicher Beweis im Skript):

- Habe $G = (V, \Sigma, P, S)$ Zyklus $A_1 \to A_2, \ldots, A_{n-1} \to A_n, A_n \to A_1$
- Sei $G' = (V', \Sigma, P', S')$ die Grammatik nach Entfernen des Zyklus.
- Sei σ die Substitution $\{A_i \mapsto A_1 \mid i \in \{2, \ldots, n\}\}$.
- „$L(G) \subseteq L(G')$":
  - Zeige mit Induktion über n, dass für alle $w_1, w_2 \in (\Sigma \cup V)^*$ gilt: Wenn $w_1 \Rightarrow_G^n w_2$, dann $\sigma(w_1) \Rightarrow_{G'}^* \sigma(w_2)$.
- „$L(G') \subseteq L(G)$":
  - Zeige mit Induktion über n, dass für $w \in (\Sigma \cup V')^*$ und $w_0 \in \Sigma^*$ gilt: Wenn $w \Rightarrow_{G'}^n w_0 \in \Sigma^*$. dann $w \Rightarrow_G^* w_0$.

Korrektheit von Algorithmus 5: Terminierung

Algorithmus 5 terminiert, denn

- Die Solange-Schleife terminiert, da jede Iteration die Anzahl der Zyklen strikt verkleinert.
- Die für-Schleife terminiert offensichtlich.

Es werden keine Einheitsproduktionen eingeführt:

- Da die Produktionen topologisch sortiert behandelt werden: Wenn $A_i \to A_j$ entfernt wird, wurden **vorher** alle Einheitsproduktionen $A_j \to A_k$ entfernt. D.h. zu diesem Zeitpunkt gilt: für alle Produktionen $A_j \to w$ besteht w nicht nur aus einer Variablen.

Es werden keine ε-Produktionen eingeführt:

- offensichtlich.

Algorithmus 6: Herstellen der Chomsky-Normalform

Eingabe: CFG G mit $\varepsilon \notin L(G)$
Ausgabe: CFG G' in Chomsky-Normalform mit $L(G) = L(G')$
Beginn

Entferne die ε-Produktionen in G mit Algorithmus 1 und entferne anschließend die Einheitsproduktionen mit Algorithmus 5;
Sei $G' = (V', \Sigma, P', S')$ die entstandene Grammatik;
für alle $a \in \Sigma$ **tue**
/* Führe neue Variable A_a für a ein, und ersetze Vorkommen von a durch das Nichtterminal */
$G' := (V' \cup \{A_a\}, \Sigma, \{A \to w[A_a/a] \mid A \to w \in P' \text{ und } |w| > 1\}$
$\cup \{A \to w \mid A \to w \in P' \text{ und } |w| = 1\} \cup \{A_a \to a\}, S)$
/* Nun sind alle Regeln von der Form $A \to a$ oder $A \to B_1 \cdots B_m$ mit $m \geq 2$ */
für alle $A \to B_1 \cdots B_m \in P'$ *mit* $m > 2$ **tue**
Seien $C_1, \ldots, C_{m-2}$ neue Variablen;
$V' := V' \cup \{C_1, \ldots, C_{m-2}\}$;
/* Ersetze in P' die Produktion $A \to B_1 \cdots B_m$ durch neue Regeln */
$P' := (P' \setminus \{A \to B_1 \cdots B_m\})$
$\cup \{A \to B_1 C_1\} \cup \{C_i \to B_{i+1} C_{i+1} \mid \text{für } i = 1, \ldots, m-3\} \cup \{C_{m-2} \to B_{m-1} B_m\}$;

Theorem

Für CFGs G mit $\varepsilon \notin L(G)$ berechnet Algorithmus 6 eine äquivalente CFG in Chomsky-Normalform.

Beweis:

- Die Schritte „Entferne der ε-Produktionen und Entfernen von Einheitsproduktionen haben wir als korrekt gezeigt.
- Die Schritte „Einführen von Produktionen $A \to a$“ und Behandlung von $A \to B_1 \cdots B_m$ sind Instanzen der korrekten Operation „Sharing von Satzformen mit neuen Variablen“
- Verifiziere, dass danach alle Produktionen die gewünschte Form haben.

Beispiel: Chomsky-Normalform berechnen

$G_0 = (\{A, B, C, D, S\}, \{0, 1\}, P_0, S)$ mit

$$P_0 = \{S \to 1A,\ A \to AB,\ A \to DA,\ A \to \varepsilon,\ B \to 0,\ B \to 1,\ C \to AAA,\ D \to 1AC\}.$$

Schritt 1: Entfernen der ε-Produktionen

$G_0 = (\{A, B, C, D, S\}, \{0, 1\}, P_0, S)$ mit

$$P_0 = \{S \to 1A,\ A \to AB,\ A \to DA,\ A \to \varepsilon,\ B \to 0,\ B \to 1,\ C \to AAA,\ D \to 1AC\}.$$

- Menge W der Variablen, die ε herleiten:
  $W = \{A, C\}$ da $A \to \varepsilon$ und $C \to AAA$
- Starte mit
  $G_1 = (\{A, B, C, D, S\}, \{0, 1\}, P_1, S)$
  $P_1 = \{S \to 1A,\ A \to AB,\ A \to DA,\ B \to 0,\ B \to 1,\ C \to AAA,\ D \to 1AC\}.$
- Hinzufügen von Produktionen für Vorkommen von A und C
  $P_1 = \{S \to 1A,\ S \to 1,\ A \to AB,\ A \to B,\ A \to DA,\ A \to D,\ B \to 0,\ B \to 1,\ C \to AAA,\ C \to AA,\ C \to A,\ D \to 1AC,\ D \to 1A,\ D \to 1C,\ D \to 1\}.$

Schritt 2: Entfernen der Einheitsproduktionen (1)

$P_1 = \{S \to 1A,\ \ S \to 1,\ \ A \to AB,\ \ A \to B,\ \ A \to DA,\ \ A \to D,$
$B \to 0,\ \ B \to 1\ \ C \to AAA,\ \ C \to AA,\ \ C \to A,$
$D \to 1AC,\ \ D \to 1A,\ \ D \to 1C,\ \ D \to 1\}.$

- Der gerichtete Graph ist D $= (\{S, A, B, C, D\}, \{(A, B), (A, D), (C, A)\})$.
- Es gibt keine Zyklen, wir erhalten $P_2 = P_1$
- Topologisches Sortieren und Umbenennen der Variablen, sodass „$A_i \to A_j$ impliziert $i < j$“ gilt, erfordert eine Umbenennung, welche die Beziehungen „$A < B$“, „$A < D$“, „$C < A$“ erzeugt.
- Wir wählen Umbenennung ρ mit $\rho(C) = A_1$, $\rho(A) = A_2$, $\rho(B) = A_3$, $\rho(D) = A_4$, $\rho(S) = A_5$.
- Das liefert uns $G_3 = (\{A_1, A_2, A_3, A_4, A_5\}, \Sigma, P_3, A_5)$ mit

$P_3 = \{A_5 \to 1A_2,\ \ A_5 \to 1,\ \ A_2 \to A_2A_3,\ \ A_2 \to A_3,\ \ A_2 \to A_4A_2,$
$A_2 \to A_4,\ \ A_3 \to 0,\ \ A_3 \to 1,\ \ A_1 \to A_2A_2A_2,\ \ A_1 \to A_2A_2,$
$A_1 \to A_2,\ \ A_4 \to 1A_2A_1,\ \ A_4 \to 1A_2,\ \ A_4 \to 1A_1,\ \ A_4 \to 1\}.$

Schritt 2: Entfernen der Einheitsproduktionen (2)

$P_3 = \{A_5 \to 1A_2,\ \ A_5 \to 1,\ \ A_2 \to A_2A_3,\ \ A_2 \to A_3,\ \ A_2 \to A_4A_2,$
$A_2 \to A_4,\ \ A_3 \to 0,\ \ A_3 \to 1,\ \ A_1 \to A_2A_2A_2,\ \ A_1 \to A_2A_2,$
$A_1 \to A_2,\ \ A_4 \to 1A_2A_1,\ \ A_4 \to 1A_2,\ \ A_4 \to 1A_1,\ \ A_4 \to 1\}.$

- Nun läuft die Für-Schleife für i von 5 bis 1:
  - Für $i = 5, i = 4, i = 3$ gibt es jeweils keine Produktion der Form $A_i \to A_j$.
  - Für $i = 2$ wird $A_2 \to A_3$ ersetzt durch $A_2 \to 0,\ \ A_2 \to 1$, und es wird $A_2 \to A_4$ ersetzt durch $A_2 \to 1A_2A_1,\ \ A_2 \to 1A_2,\ \ A_2 \to 1A_1$ und $A_2 \to 1$. Danach ist

$P_4 = \{A_5 \to 1A_2,\ \ A_5 \to 1,\ \ A_2 \to A_2A_3,\ \ A_2 \to 0,\ \ A_2 \to 1,\ \ A_2 \to A_4A_2,$
$A_2 \to 1A_2A_1,\ \ A_2 \to 1A_2,\ \ A_2 \to 1A_1,\ \ A_3 \to 0,\ \ A_3 \to 1,$
$A_1 \to A_2A_2A_2,\ \ A_1 \to A_2A_2,\ \ A_1 \to A_2,\ \ A_4 \to 1A_2A_1,$
$A_4 \to 1A_2,\ \ A_4 \to 1A_1,\ \ A_4 \to 1\}.$

  - Für $i = 1$ wird $A_1 \to A_2$ ersetzt durch $A_1 \to A_2A_3$, $A_1 \to 0$, $A_1 \to 1$, $A_1 \to A_4A_2$, $A_1 \to 1A_2A_1$, $A_1 \to 1A_2$ und $A_1 \to 1A_1$.

Schritt 2: Entfernen der Einheitsproduktionen (3)

- Daher ist die Grammatik nach Entfernen der Einheitsproduktionen: $G_5 = (V_5, \Sigma, P_5, A_5)$ mit $V_5 = \{A_1, A_2, A_3, A_4, A_5\}$ und

$P_5 = \{A_5 \to 1A_2,\ \ A_5 \to 1,\ \ A_2 \to A_2A_3,\ \ A_2 \to 0,\ \ A_2 \to 1,$
$A_2 \to A_4A_2, A_2 \to 1A_2A_1,\ \ A_2 \to 1A_2,\ \ A_2 \to 1A_1,\ \ A_3 \to 0,$
$A_3 \to 1,\ \ A_1 \to A_2A_2A_2,\ \ A_1 \to A_2A_2,\ \ A_1 \to A_2A_3,\ \ A_1 \to 0,$
$A_1 \to 1,\ \ A_1 \to A_4A_2,\ \ A_1 \to 1A_2A_1,\ \ A_1 \to 1A_2,\ \ A_1 \to 1A_1,$
$A_4 \to 1A_2A_1,\ \ A_4 \to 1A_2,\ \ A_4 \to 1A_1,\ \ A_4 \to 1\}.$

Schritt 3: Terminalsymbole durch neue Produktionen darstellen

Füge $B_0 \to 0$ und $B_1 \to 1$ hinzu und ersetze in rechten Seiten mit Wortlänge > 1:

$P_6 = \{B_0 \to 0,\ \ B_1 \to 1,\ \ A_5 \to B_1A_2,\ \ A_5 \to 1,\ \ A_2 \to A_2A_3,\ \ A_2 \to 0,$
$A_2 \to 1,\ \ A_2 \to A_4A_2,\ \ A_2 \to B_1A_2A_1,\ \ A_2 \to B_1A_2,$
$A_2 \to B_1A_1,\ \ A_3 \to 0,\ \ A_3 \to 1,\ \ A_1 \to A_2A_2A_2,\ \ A_1 \to A_2A_2,$
$A_1 \to A_2A_3,\ \ A_1 \to 0,\ \ A_1 \to 1,\ \ A_1 \to A_4A_2,\ \ A_1 \to B_1A_2A_1,$
$A_1 \to B_1A_2,\ \ A_1 \to B_1A_1,\ \ A_4 \to B_1A_2A_1,$
$A_4 \to B_1A_2,\ \ A_4 \to B_1A_1,\ \ A_4 \to 1\}.$

Schritt 4: Rechte Seiten zerlegen

Zerlege rechte Seiten mit Wortlänge > 2:

Ergibt $G_7 = (V_7, \Sigma, P_7, A_5)$, wobei

$$
\begin{aligned}
V_7 &= \{A_1, A_2, A_3, A_4, A_5, B_0, B_1, C_1, C_2, C_3, C_4\} \\
P_7 &= \{B_0 \to 0,\ \ B_1 \to 1,\ \ A_5 \to B_1A_2,\ \ A_5 \to 1,\ \ A_2 \to A_2A_3,\ \ A_2 \to 0, \\
&\quad\ A_2 \to 1,\ \ A_2 \to A_4A_2,\ \ A_2 \to B_1C_1\ \ C_1 \to A_2A_1, \\
&\quad\ A_2 \to B_1A_2,\ \ A_2 \to B_1A_1,\ \ A_3 \to 0,\ \ A_3 \to 1,\ \ A_1 \to A_2C_2, \\
&\quad\ C_2 \to A_2A_2,\ \ A_1 \to A_2A_2,\ \ A_1 \to A_2A_3,\ \ A_1 \to 0,\ \ A_1 \to 1, \\
&\quad\ A_1 \to A_4A_2,\ \ A_1 \to B_1C_3,\ \ C_3 \to A_2A_1,\ \ A_1 \to B_1A_2, \\
&\quad\ A_1 \to B_1A_1,\ \ A_4 \to B_1C_4,\ \ C_4 \to A_2A_1,\ \ A_4 \to B_1A_2, \\
&\quad\ A_4 \to B_1A_1,\ \ A_4 \to 1\}.
\end{aligned}
$$

Alle Schritte beendet, G_7 ist in Chomsky-Normalform