Mitschrift zur Vorlesung

Automatentheorie und formale Sprachen

Tim Niemueller

30. Oktober 2003

Vorwort

Dieses Dokument ist eine Mitschrift der Vorlesung "Automatentheorie und formale Sprache", die im Sommersemester 2003 and der RWTH Aachen von Prof. Dr. Indermark gehalten wurde.

Dieser Text spiegelt die von mir aufgezeichneten Unterlagen wider, die ich während der Vorlesung angelegt habe. Ich kann daher weder die Vollständigkeit noch die Korrektheit dieser Mitschrift gewährleisten.

Ich halte mich an die Notation von Prof. Indermark. Das bedeutet, dass für die "daraus folgt" Beziehung der $\curvearrowright$ Pfeil verwendet wird. Für die "genau dann, wenn" Beziehung wird der $\rightleftarrows$ Pfeil verwendet.

Dieses Dokument unterliegt der GNU Free Documentation License 1.2 (siehe Anhang A).

Für Kritik, Verbesserungsvorschläge und insbesondere Korrekturen bin ich jederzeit offen.

Tim Niemueller
Sommer 2003 Aachen

Download und Kontakt
Web: http://www.niemueller.de/
Mail: atfs-skript@niemueller.de

©2003 by Tim Niemueller (made with LaTeX 2ε and Vim)

Inhaltsverzeichnis

Kapitel 1

Alphabete, Wörter, Sprachen

Zeichenreihen als Grundobjekte der Informatik.

- Präzisierung des Algorithmusbegriffs durch Turingmaschinen
- Kommunikation mit Rechnern über Tastatur
- Informationsverarbeitung: Transformation von Bit-Strings

1.1 Grundbegriff Σ Alphabet

Nicht-leere endliche Menge.
$a \in \Sigma$ Buchstabe, Zeichen, Character, Symbol

1.2 Grundbegriff Σ^*

Menge aller Wörter über Σ.
$\Sigma^* := \{a_1 a_2 ... a_n | n \in \mathbb{N}\}$

1.2.1 Operationen auf Σ^*

- Verkettung (Konkatenation)
  $a \cdot b : \Sigma^* \times \Sigma^* \to \Sigma^*$ mit $a, b \in \Sigma^*$

  i. $\cdot$ ist assoziativ

  ii. ε ist $\cdot$-neutral

  Sprechweise: $< \Sigma^*; \cdot; \varepsilon >$ ist ein
  Bemerkung: Freie Erzeugung, d.h. $a_1 ... a_n = b_1 ... b_m \curvearrowright n = m, a_i = b_i (i = 1...n)$
- Länge eines Wortes
  $w = a_1 a_2 ... a_n \in \Sigma^*$, dann ist $|a_1 ... a_n| := n$, falls alle $a_i \in \Sigma$, also $|\varepsilon| = 0$ und $|wv| = |w| + |v|$ für $w, v \in \Sigma^*$
- Potenzen eines Wortes
  $w \in \Sigma^*$
  $w^0 := \varepsilon$, $w^{n+1} := w^n w$

- Spiegelbild eines Wortes
  $\varepsilon^R := \varepsilon$
  $(wa)^R := aw^R$ mit $w \in \Sigma^*$ und $a \in \Sigma$

1.3 Grundbegriff $\wp(\Sigma^*)$

$\wp(\Sigma^*)$ ist die Menge der formalen Sprachen über Σ und ist definiert als $\wp(\Sigma^*) := \{L | L \subseteq \Sigma^*\}$.
Beispiel: $\emptyset$, $\{\varepsilon\}$

1.3.1 Operationen auf $\wp(\Sigma^*)$

- Boolesche Operationen
  $L_1 \cup L_2$, $L_1 \cap L_2$, $\overline{L} := \Sigma^* \setminus L$
- Komplexprodukt
  $L_1 L_2 := \{wv \mid w \in L_1, v \in L_2\}$
- Potenzen einer Sprache
  $L^0 := \{\varepsilon\}$
  $L^{n+1} := L^n L$
- Stern einer Sprache
  $L^* := \bigcup_{n \in \mathbb{N}} L^n$
  $\emptyset^* = \{\varepsilon\}$
- Spiegelbild einer Sprache
  $L^R := \{w^R \mid w \in L\}$
- Reguläre Operationen
  $L_1 \cup L_2$, $L_1 L_2$, L^*

Kapitel 2

Reguläre Ausdrücke und endliche Automaten

2.1 Reguläre Ausdrücke

Eine regulärer Ausdruck beschreibt eine formale Sprache, die sich mit Hilfe regulärer Operationen (siehe1.3.1 auf Seite 2) aus einfachen Sprachen erzeugen lässt.

2.1.1 Definition der Syntax von RegE(Σ)

Sei Σ ein Alphabet.
Die Menge der RegE(Σ)der *regulären Ausdrücke über* Σ ist induktiv definiert durch

i. $\Lambda \in RegE(\Sigma)$

ii. $a \in RegE(\Sigma)$ für alle $a \in \Sigma$

iii. $(\alpha \vee \beta) \in RegE(\Sigma)$
"Alternative"

iv. $(\alpha \cdot \beta) \in RegE(\Sigma)$
"Konkatenation"

v. $(\alpha^*) \in RegE(\Sigma)$
"Repetition"

Vereinfachte Schreibweise

- Präzedenzregeln, um Klammern zu sparen:
  $*$ bindet stärker als $\cdot$
  $\cdot$ bindet stärker als $\vee$

- Der Punkt $\cdot$ wird weggelassen

Beispiel: $a \vee b^*c = (a \vee ((b^*) \cdot c))$

2.1.2 Definition der Semantik von RegE(Σ)

Ein regulärer Ausdruck α beschreibt eine formale Sprache $L(\alpha)$ (auch geschrieben als $[|\ \alpha\ |]$). Die Semantik ist definiert durch

i. $L(\Lambda) := \emptyset$

ii. $L(a) := \{a\}$

iii. $L(\alpha \vee \beta) := L(\alpha) \cup L(\beta)$

iv. $L(\alpha \cdot \beta) := L(\alpha) \cdot L(\beta)$

v. $L(\alpha^*) := L(\alpha)^*$

Sprechweise: $w \in L(\alpha) \curvearrowright$ w ist ein Match für α, α ein Muster (Pattern).

2.1.3 Klasse der regulären Sprachen

Die Klasse RegL(Σ)der regulären Sprachen über Σ ist induktiv definiert durch

- $\emptyset$, $\{a\} \in RegL(\Sigma)$ für alle $a \in \Sigma$
- $L, L' \in RegL(\Sigma) \curvearrowright L \cup L'$, LL', $L^* \in RegL(\Sigma)$

Folgerung: $RegL(\Sigma) = \mathfrak{L}(\Sigma, RegE)$

$L(\Lambda^*) = L(\Lambda)^* = \emptyset^* = \bigcup_{n=0}^{\infty} \emptyset^n = \{\varepsilon\}$ wegen $\emptyset^0 = \{\varepsilon\}$.
$L^+ = \bigcup_{n=1}^{\infty} L^n$

2.2 Deterministische endliche Automaten

2.2.1 Definition

Seien Q und Σ nicht-leere, endliche Mengen. $q_0 \in Q$, $F \subset Q$ und $\delta : Q \times \Sigma \to Q$.
Dann heißt

$\mathfrak{A} = \langle Q, \Sigma, \delta, q_0, F \rangle$

ein *deterministischer endlicher Automat* über Σ mit der *Zustandsmenge* Q, dem *Eingabealphabet* Σ, der *Transitionsfunktion* δ, dem *Anfangszustand* q_0 und der *Endzustandsmenge* F.

2.2.2 Bezeichnung: $\mathfrak{L}(\Sigma, DFA)$

Klasse der von endlichen Automaten erkennbaren Sprachen über Σ.

2.2.3 Ziel

Beweise, dass gilt: $\mathfrak{L}(\Sigma, DFA) = RegL(\Sigma)$
Hilfsmittel: nicht-deterministischer Automat
Hinweis: Scanner, Suchmaschinen, SW-Tools

2.3 Nicht-Deterministische endliche Automaten

2.3.1 Definition

Seien Q, B, q_0, F wie bei einem DFA definiert.
Ferner $\Sigma_\varepsilon := \Sigma \cup \{\varepsilon\}$ und $\delta := Q \times \Sigma_\varepsilon \to \wp(Q)$.
Dann heißt

$\mathfrak{A} = \langle Q, \Sigma, \delta, q_0, F \rangle$

ein *nicht-deterministischer endlicher Automat über* Σ.
Bezeichnung: $NFA(\Sigma)$, NFA.
Es folgt: $DFA(\Sigma) \subseteq NFA(\Sigma)$.

2.3.2 Semantik eines $\mathfrak{A} \in NFA(\Sigma)$

Erweiterung der Semantik von DFAs:
Für $T \subseteq Q$ ist die ε-*Hülle von* T – geschrieben $\varepsilon(T)$ – induktiv definiert durch:

- $T \subseteq \varepsilon(T)$
- $q \in \varepsilon(T) \curvearrowright \delta(q, \varepsilon) \subseteq \varepsilon(T)$

Die erweiterte Transitionsfunktion $\overline{\delta} : \wp(Q) \times \Sigma^* \to \wp(Q)$ ist induktiv definiert durch:

- $\overline{\delta}(T, \varepsilon) := \varepsilon(T)$
- $\overline{\delta}(T, wa) := \varepsilon(\bigcup_{q \in \overline{\delta}(T,w)} \delta(q, a)$

Damit ist die durch $\mathfrak{A}$ erkannte Sprache definiert als:

$\mathfrak{L}(\mathfrak{A}) := \{w \in \Sigma^* \mid \overline{\delta}(\{q_0\}, w) \cap F \neq \emptyset\}$

Beachte: Für DFAs stimmen beide Semantikdefinitionen überein:

- $|\delta(q, a)| = 1 \forall q \in Q, a \in \Sigma$
- $|\delta(q, a)| = 0$

2.3.3 Potenzmengenautomat

Sei $\mathfrak{A} = \langle Q, \Sigma, \delta, q_0, F \rangle \in NFA(\Sigma)$

Der *Pontenzmengenautomat* $\mathfrak{A} := \langle \widehat{Q}, \Sigma, \widehat{\delta}, \widehat{q_0}, \widehat{F} \rangle \in DFA(\Sigma)$ ist definiert durch

- $\widehat{Q} := \{T \subseteq Q \mid T = \widehat{\delta}(\{q_0\}, w), w \in \Sigma^*\}$
- $\widehat{\delta} : \widehat{Q} \times \Sigma \to \widehat{Q}$ mit $\widehat{\delta}(T, a) := \overline{\delta}(T, a)$
- $\widehat{q_0} := \varepsilon(\{q_0\})$
- $T \in \widehat{F} \rightleftarrows T \in \widehat{Q}$ und $T \cap F \neq \emptyset$

$T \in \widehat{Q} \curvearrowright T = \overline{\delta}(\{q_0\}, w)$
ferner: $\overline{\delta}(T, \varepsilon) = \varepsilon(T) =$ Transitionsfunktion

$$\begin{aligned}\delta(T,a) &= \varepsilon(\bigcup_{q\in\delta(T,\varepsilon)} \delta(q,a)) \\ &= \varepsilon(\bigcup_{q\in T} \delta(q,a)) \\ &= \varepsilon(\bigcup_{q\in\overline{\delta}(\{q_0\},w)} \delta(q,a)) \\ &= \overline{\delta}(\{q_0\}, wa) \in \widehat{Q}\end{aligned}$$

Lemma

Für $\mathfrak{A} \in NFA(\Sigma)$ gilt:

$L(\mathfrak{A}) = L(\mathfrak{A}^P)$

Beweis

$$\begin{aligned}w \in L(\mathfrak{A}) &\rightleftarrows \delta(\{q_0\}, w) \cap F \neq \emptyset \\ &\rightleftarrows \overline{\delta}(\varepsilon(\{q_0\}, w) \cap F \neq \emptyset \\ &\rightleftarrows \widehat{\overline{\delta}}(\widehat{\delta_0}, w) \in \widehat{F} \\ &\rightleftarrows w \in L(\mathfrak{A}^P)\end{aligned}$$

□

2.4 Synthese und Analyse endlicher Automaten

2.4.1 Synthese

Konstruiere für $\alpha \in RegE(\Sigma)$ einen äquivaltenen Automaten $\mathfrak{A} \in NFA(\Sigma)$, d.h.

$L(\alpha) = L(\mathfrak{A}(\alpha))$

2.4.2 Der Algorithmus von Thompson

Idee

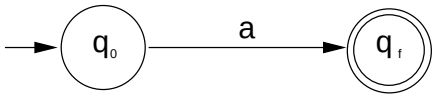
$\mathfrak{A}$ hat genau einen Endzustand $q_f \neq q_0$.

Algorithmus

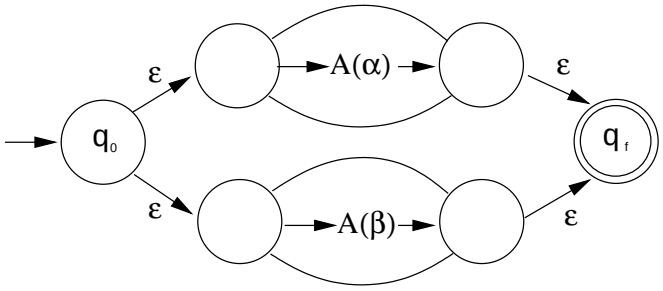
q_0 Quelle, q_f Senke (im Zustandsgraphen)

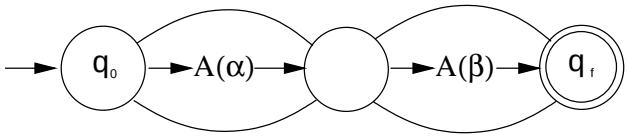
$\mathfrak{A}(\Lambda) :=$ 

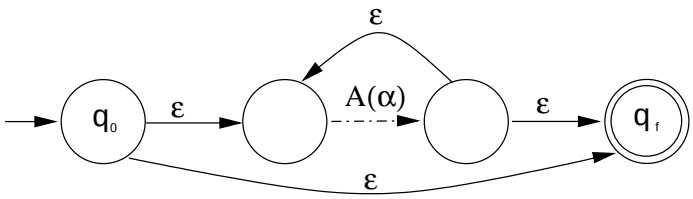
(keine Transition)

$\mathfrak{A}(a) :=$ 

$a \in \Sigma$

$\mathfrak{A}(\alpha \vee \beta) :=$ 

$\mathfrak{A}(\alpha \cdot \beta) :=$ 

$\mathfrak{A}(\alpha^*) :=$ 

Offentsichtlich gilt für jedes $\alpha \in RegE(\Sigma)$: $L(\alpha) = L(\mathfrak{A}(\alpha))$

Korollar

$RegL(\Sigma) \subseteq \mathfrak{L}(\Sigma, DFA)$

2.4.3 Analyse

Konstruiere für $\mathfrak{A} \in DFA(\Sigma)$ einen $\alpha(\mathfrak{A}) \in RegE(\Sigma)$ mit $L(\alpha(\mathfrak{A})) = L(\mathfrak{A})$.
Oder kurz: $\alpha(\mathfrak{A}) \sim \mathfrak{A}$

$\mathfrak{A} = \langle Q, \Sigma, \delta, q_1, F \rangle \in DFA(\Sigma)$ und

$Q = \{q_1, ..., q_n\}$

Für $i, j \in \{1, ..., n\}$ und $k \in \{0, ..., n\}$ definieren wird

$W_{ij}^k := \{w \in \Sigma^* \mid w$ überführt q_i in q_j ohne Benutzung von $q_{k+1}, ..., q_n$ als Zwischenzustände $\}$

Dann gilt: $L(\mathfrak{A}) = \bigcup_{q_j \in F} W_{ij}^k$

Somit genügt der Nachweis der Regularität der Sprachen W_{ij}^k.

Induktion über k:

1. $k = 0:\ W_{ij}^0 \subseteq \Sigma \cup \{\varepsilon\}$
   $\curvearrowright W_{ij}^0$ regulär, beachte: $L(\Lambda^*) = \{\varepsilon\}$
2. $k = 1 \rightsquigarrow k$:
   $W_{ij}^k = W_{ij}^{k-1} \cup W_{ik}^{k-1}(W_{kk}^{k-1})^* W_{kj}^{k-1}$

Korollar (Satz von Kleene)

$\mathfrak{L}(\Sigma, DFA) = \mathfrak{L}(\Sigma, RegE) = RegL(\Sigma)$ =Typ-3-Sprachen nach Chomsky

2.4.4 Wortsuche in Texten

Problem

Bestimme alle Dokumente, in denen ein Wort einer gegebenen Wortmenge vorkommt.

Beispiel: Der Suchautomat von {web, ebay}

Es gibt zwei Implementierungstechniken:

1. NFA-Methode
   Berechne für ein $w \in \Sigma_{ASCII}^*$ *einen* Lauf durch den Suchautomaten (erreichbare Zustandsmenge). Siehe hierzi Folie 13 aus der Vorlesung.
2. DFA-Methode
   Konstruiere den Potenzmengenautomaten des Suchautomaten.

Bemerkung: Implementierung von egrep und fgrep verwenden beide Techniken.

2.5 Pumping Lemma

Das Pumping Lemma ist ein Hilfsmittel zum Nachweis nicht-regulärer Sprachen.

2.5.1 Satz (Pumping Lemma, Iterationslemma)

Sei $L \in RegL(\Sigma)$. Dann existiert $k \in \mathbb{N}$, so dass für jedes $x \in L$ mit $|x| \geq k$ eine Zerlegung

$x = uvw$

mit folgenden Eigenschaften:

i. $|v| \geq 1$

ii. $|uv| \leq k$

iii. $uv^i w \in L$ für jedes $i \in \mathbb{N}$ (auch $i = 0$)

Beweis

Sei $\mathfrak{A} \in DFA(\Sigma)$ mit $L(\mathfrak{A}) = L$ und $K := |Q|$. Sei $x \in L$ mit $|x| \geq k$. Dann wird beim "Erkennungslauf" von x in $\mathfrak{A}$ mindestens ein Zustand mehr als einmal besucht.

Sei q_i der erste solche Iterationszustand und v Teilwort von dort bis zur ersten Wiederholung.

Dann gilt $|uv| - 1 \leq k - 1 \curvearrowright (ii)$.
(i) und (iii) klar. □

Beachte: Das Pumping-Lemma bescheribt eine notwendige, aber nicht eine hinreichende Eigenschaft.

Beispiel

$L = \{a^n b^n \mid n \geq 1\} \notin RegL(\{a, b\})$

Beweis

Angenommen, $L \in RegL$.
Dann existiert $k \in \mathbb{N}$ mit (i)-(iii) aus Pumping-Lemma.

Für $x = a^k b^k$ muss eine Zerlegung existieren der Form

$a^k b^k = uvw$

mit $v \neq \varepsilon$ und $|uv| \leq k$, also $v \in \{a^i \mid i > 0\}$ und $uw = a^{k-|v|} b^k \in L$. Widerspruch. □

2.6 Zustandsreduktion endlicher Automaten

Ziel: Konstruktion endlicher Automaten mit minimaler Zustandszahl.

2.6.1 Minimalautomat

Definition Minimalautomat

Für $L \subseteq \Sigma^*$ und $w \in \Sigma^*$ heißt $d_w(L) := \{v \in \Sigma^* \mid wv \in L\}$ die Ableitung von L nach w.

Lemma über minimale Zustandmenge

Sei $L = L(\mathfrak{A})$ für $\mathfrak{A} = \langle Q, \Sigma, \delta, q_0, F\rangle \in DFA(\Sigma)$.

Dann gilt: $D(L) = \{d_w(L) \mid w \in \Sigma^*\}$ mit $|D(L)| \leq |Q|$

Beweis der Minimalität

Für $q \in Q$ bezeichne $L(q) := L(\langle Q, \Sigma, \delta, q, F\rangle)$.

Dann gilt: $d_w(L) = d_w(L(q_0)) = L(\delta(q_0, w))$

Einen schöneren Beweis gibt es in der Vorlesung "Berechenbarkeit und Komplexität". □

2.6.2 Ableitungsautomat

Definition

Sei $L \in RegL(\Sigma)$. Der *Ableitungsautomat* $\mathfrak{A}_L = \langle D(L), \Sigma, \delta, q_0, F \rangle \in DFA(\Sigma)$ ist definiert durch:

- $q_0 := D_\varepsilon(L) = L$
- $F := \{d_w(L) \mid w \in L\}$, wg. $w \in L$ also $\varepsilon \in d_w(L)$
- $\delta(d_w(L), a) := d_{wa}(L)$

Beachte: $d_w(L) = d_v(L) \curvearrowright d_{wa}(L) = d_{va}(L)$. Also ist die Definition unabhängig vom "Repräsentanten" w.

Lemma

$L(\mathfrak{A}_L) = L$ (d.h. $\mathfrak{A}_L$ erkennt L).

Beweis

$$\begin{aligned} w \in L(\mathfrak{A}_L) &\rightleftarrows \delta(q_0, w) \in F \\ &\rightleftarrows d_w(L) \in F \\ &\rightleftarrows w \in L \end{aligned}$$

□

Korollar

i. Der Ableitungsautomat ist zustandsminimal

ii. Für $L \in \Sigma^*$ gilt: $L \in RegL(\Sigma) \rightleftarrows D(L) < \infty$

2.6.3 Zustandsreduktion

Konstruktion eines Zustands-minimalen Automaten aus einem gegebenen Automaten durch:

- weglasssen nicht-erreichbarer Zustände und
- verschmelzen äquivalenter Zustände

Dann heißt

- $q \in Q$ *erreichbar* $\rightleftarrows \exists w \in \Sigma^* : \overline{\delta}(q_0, w) = q$
- $q_1 \sim q_2$ (äquivalent) $\rightleftarrows L(q_1) = L(q_2)$

2.6.4 Definition des Faktorautomaten

Für $\mathfrak{A} = \langle Q, \Sigma, \delta, q_0, F\rangle \in DFA(\Sigma)$ist der *Faktorautomat*

$\mathfrak{A}/\sim := \langle \widetilde{Q}, \Sigma, \widetilde{\delta}, \widetilde{q_0}, \widetilde{F}\rangle \in DFA(\Sigma)$

wie folgt definiert:

- für $q \in Q$ sei $[q] := \{q' \mid q' \sim q\}$
- $\widetilde{Q} := \{[\overline{\delta}(q_0, w)] \mid w \in \Sigma^*\}$
- $\widetilde{q} := [q_0]$
- $\widetilde{F} := \{[q] \mid q \in F\}$
- $\widetilde{\delta}([q], a) := [\delta(q, a)]$

Beachte

$q \in F, q \sim q' \curvearrowright q' \in F$ und $q \sim q' \curvearrowright \delta(q,a) \sim \delta(q',a)$ zeigt die Unabhängigkeiten der Definition von den Repräsentanten.

Lemma

Für $\mathfrak{A} \in DFA(\Sigma)$ gilt:

$\mathfrak{A}/\sim = \mathfrak{A}_L(\mathfrak{A})$ (bis auf Zustandsnamen)

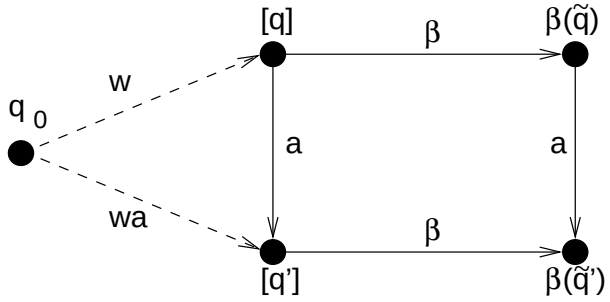
Insbesondere ist $\mathfrak{A}/\sim$ äquivalent zu $\mathfrak{A}$ und es folgt: Zustandsminimale Automaten sind bis auf Isomorphie eindeutig bestimmt.

Beweis

Sei $\beta : \widetilde{Q} \mapsto D(L(\mathfrak{A}))$ definiert durch

$\beta([\overline{\delta}(q_0, v)]) := d_w(L(\mathfrak{A}))$

- β ist unabhängig vom Repräsentanten $w \in \Sigma^*$:
  $\overline{\delta}(q_0, w) \rightleftarrows L(\overline{\delta}(q_0, w)) = L(\overline{\delta}(q_0, v)) \curvearrowright d_w(L(\mathfrak{A})) = d_v(L(\mathfrak{A}))$ $(*)$
- β ist bijektiv, weil die Folgerung $(*)$ umkehrbar ist
- β ist selbsterhaltend
- $\beta([q_0]) = \beta([\overline{\delta}(q_0, \varepsilon)]) = d_\varepsilon(L(\mathfrak{A})) = L(\mathfrak{A})$ = Anfangszustand des Ableitungsautomaten
- $\overline{\delta}(q_0, w) = q \in F$
  $\beta([\overline{\delta}(q_0, w)]) = d_w(L(\mathfrak{A}))$ ist ebenfalls Endzustand im Ableitungsautomaten



$$\bar{\delta}(\underbrace{[q]}_{\underbrace{[\bar{\delta}(q_0,w)]}_{\delta(d_w(L(\mathfrak{A})),a)}},a) = \underbrace{[\delta(q,a)]}_{\underbrace{[\bar{\delta}(q_0,wa)]}_{d_{wa}(L(\mathfrak{A}))}}$$

$$\bar{\delta}(\delta(d_w(L(\mathfrak{A})),a),a) = d_{wa}(L(\mathfrak{A}))$$

□

2.6.5 Verfahren der Zustandsminimierung

- Weglassen nicht erreichbarer Zustände
- Verschmelzen äquivalenter Zustände

Definition

Sei $\mathfrak{A} = \langle Q, \Sigma, \delta, q_0, F\rangle \in DFA(\Sigma)$und jeder Zustand erreichbar, ferner $k \in \mathbb{N}, q_1, q_2 \in Q$.

Dann sagt man

- q_1 *k-äquivalent* q_2 ($q_1 \sim^k q_2$)
  $\rightleftarrows \forall w \in \Sigma^*, |w| \leq k : w \in L(q_1) \rightleftarrows w \in L(q_2)$
- $\mathfrak{A}$ induziert die Abbildung
  $r^k : Q^2 \mapsto \{0,1\}$
  mit $r^k(q_1,q_2) = 1 \rightleftarrows q_1 \sim^k q_2$
  Darstellung als Matrizen $R^k = (r^k(q_i,q_j))_{i,j=1}^n$ mit $n = |Q|$.

Beachte: $r^k(q_i,q_j) = r^k(q_j,q_i)$

Berechnung der k-Äquivalenz-Matrizen

- $q_1 \sim^0 q_2 \rightleftarrows (q_1 \in F \rightleftarrows q_2 \in F)$
- $q_1 \sim^{k+1} q_2 \rightleftarrows q_1 \sim^0 q_2$ und
  $\delta(q_1,a) \sim^k \delta(q_2,a) \forall a \in \Sigma$

Lemma

Es gibt ein $k \in \mathbb{N}$, so dass für alle $n \in \mathbb{N}$

$R^k = R^{k+n}$

Beweis

Da $r^k(q_i,q_j) = 0 \curvearrowright r^{k+1}(q_i,q_j) = 0$, muss es ein k geben mit $r^k = r^{k+1}$.
Dann muss auch $r^k = r^{k+1} \forall n \in \mathbb{N}$ gelten.

Zu zeigen: $r^{k+1} = r^{k+2}$.

Sei $r^k(q_1,q_2) = r^{k+1}(q_1,q_2) = 1$, also $q_1 \sim^k q_2$ und $g_1 \sim^{k+1} q_2$. Sei ferner $a_1...a_{k+2} \in L(q_1)$.

$\delta(q_1,a_1) \sim^k \delta(q_2,a_1)$ und nach $(*)$ auch:
$\delta(q_1,a_1) \sim^{k+1} \delta(q_2,a_1)$.

Somit:
$a_2...a_{k+2} \in L(\delta(q_2,a_1))$
$a_1...a_{k+2} \in L(q_2)$ □

2.6.6 Markierungsalgorithmus

Eingabe: $\mathfrak{A} = \langle Q,\Sigma,\delta,q_0,F\rangle \in DFA(\Sigma)$wobei jedes $q \in Q$ erreichbar ist.

Ausgabe: äquivalente Zustände

Verfahren

- Tabelle aller Zustandspaare $\{q,q'\}$ mit $q \neq q'$
- Markiere alle Paare $\{q,q'\}$ mit $q \in F$ und $q' \notin F$ oder umgekehrt
- Für jedes unmarkierte Paar $\{q,q'\}$ und $a \in \Sigma$ teste, ob $\{\delta(q,a),\delta(q',a)\}$ markiert ist.
  Wenn ja: Markiere $\{q,q'\}$
- Wiederhole letzten Schritt bis keine Änderung mehr erfolgt.
- Unmarkierte Paare repräsentieren äquivalente Zustände.

Bemerkung

Implementierung mit $O(|Q|^2)$ Zeitkomplexität möglich.

2.7 Entscheidbare Eigenschaften

a. Deterministische Automaten

- **Wortproblem:** $w \in L(\mathfrak{A})$?
  Für $w \in \Sigma^*$ und $\mathfrak{A} \in DFA(\Sigma)$ durch Eingabe in $|w|+1$ Schritten entscheidbar
- **$\emptyset$-Problem (Leerheitsproblem):** $L(\mathfrak{A}) = \emptyset$?
  Für $\mathfrak{A} \in DFA(\Sigma)$: Durch Eingabe aller Wörter w mit $|w| < |Q|$ entscheidbar.
  Beachte: $w \in L(\mathfrak{A}), |w| \geq |Q| \curvearrowright \exists v \in L(\mathfrak{A}), |v| < |w|$ entscheidbar
  Verfahren: Testen, ob F von q_0 erreichbar.
  Durchführung in $O(|Q^2|)$-Zeit.

- $\sim$**-Problem:** $L(\mathfrak{A}) = L(\mathfrak{A}')$?
  Für $\mathfrak{A}, \mathfrak{A}' \in DFA(\Sigma)$:
  Reduktion auf $\emptyset$-Problem:
  $L(\mathfrak{A}) = L(\mathfrak{A}') \rightleftarrows L(\mathfrak{A}) \subseteq L(\mathfrak{A}')$ und $L(\mathfrak{A}') \subseteq L(\mathfrak{A}) \rightleftarrows L(\mathfrak{A}) \cap \overline{L(\mathfrak{A}')} = \emptyset$
  Zwei Produktautomaten mit $|Q| \cdot |Q'|$ Zustände, teste auf Leerheit ($\emptyset$)
  Problem in $O(|Q|^2 \cdot |Q|^2)$-Zeit entscheidbar.

b. Reguläre Ausdrücke, Nicht-deterministische Automaten
   Durch Transformation in DFA's sind alle 3 Probleme entscheidbar. Aber der Aufwand steigt.

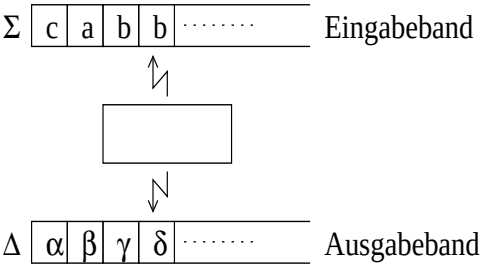
   - **Wortproblem:** $O(|\alpha||w|)$-Zeit bzw. $O(|Q|^4|w|)$
   - $\emptyset$**-Problem:** NFA wie DFA behandeln. RegE $\mapsto$ NFA in $O(|\alpha|)$-Zeit mit $|Q| \leq 2|\alpha|$
   - $\sim$**-Problem:** NP-hart (exponentieller Aufwand, Potenzmengenkonstruktion)

2.8 Endliche Automaten mit Ausgabe

2.8.1 Idee und Ziel

Idee: Erweiterung eines DFA um ein Ausgabeband

Ziel: Nicht Wörter erkennen, sondern Wörter transformieren, Berechnung von Wortfunktion.



2.8.2 Definition: Allgemein-sequentieller Automat

Sei $\langle Q, \Sigma, \delta, q_0, F \rangle \in DFA(\Sigma)$, Δ ein Ausgabealphabet und

$\lambda : Q \times \Sigma \mapsto \Delta^*$ eine *Ausgabefunktion* und
$\delta : Q \times \Sigma \mapsto Q$ eine *Übergangsfunktion*.

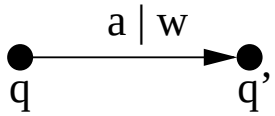
Dann heißt

$\mathfrak{A} = \langle Q, \Sigma, \Delta, q_0, \delta, \lambda \rangle$

ein allgemein-sequentieller Automat.

Bezeichnung: $\mathfrak{A} \in GSM(\Sigma, \Delta)$ ("generalised sequential automat")
Beachte: Die Endzustände werden nicht berücksichtigt.

Graphische Darstellung



falls $\delta(q, a) = q'$ und $\lambda(q, a) = w$

$\mathfrak{A}$ berechnet eine Wertefunktion $f_a : \Sigma^* \mapsto \Delta^*$.
Wir erweitern λ zu $\overline{\lambda} : Q \times \Sigma^* \mapsto \Delta^*$ durch
$\overline{\lambda}(q,\varepsilon) := \varepsilon$ und $\overline{\lambda}(q,wa) := \overline{\lambda}(q,w) \cdot \lambda(\overline{\delta}(q,w),a)$.

Dann ist $f_a(w) := \overline{\lambda}(q_0,w)$.

$f : \Sigma^* \mapsto \Delta^*$ heißt *allgemein-sequentiell*, wenn es $\mathfrak{A} \in GSM(\Sigma,\Delta)$ gibt mit $f = f_a$.

2.8.3 Spezialfall: Mealy-Automaten

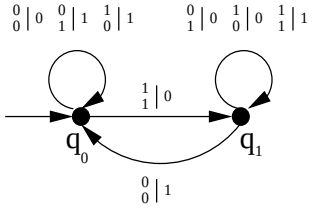
Sei $\mathfrak{A} \in GSM(\Sigma,\Delta)$. Falls $\lambda : Q \times \Sigma \mapsto \Delta$, so heißt $\mathfrak{A}$ auch *Mealy-Automat* und $f_{\mathfrak{A}}$ *sequentiell*.

Was können diese Automaten?

Binäre Addition Addition gespiegelter Binärzahlen (mit wenigstens einer Führungsnull).

$\Sigma = \mathbb{B}$
$\Delta = \wp$ mit $\mathbb{B} = \{0,1\}$

	0110101
+	0101101
=	1100010



2.8.4 Satz von Ginsberg und Rose

Sei $f : \Sigma^* \mapsto \Delta^*$. Dann gilt:

f ist allgemein-sequentiell $\rightleftarrows$ f erfüllt i. - iv.

i. $f(\varepsilon) = \varepsilon$

ii. f ist *präfixtreu*, d.h.:
$\forall u,v \in \Sigma^* \exists w \in \Delta^*$ mit $f(uv) = f(u)w$

iii. f ist *längenbeschränkt*, d.h.:
$\exists k \in \mathbb{N}$ mit $|f(w)| \leq k|w| \forall w \in \Sigma^*$

iv. f erhält *Regularität*, d.h.:

(a) $L \in RegL(\Sigma) \curvearrowright f(L) := \{f(w) \mid w \in L\} \in RegL(\Delta)$
(b) $L \in RegL(\Delta) \curvearrowright f^{-1}(L) := \{w \in \Sigma^* \mid f(w) \in L\} \in RegL(\Sigma)$

Beweis

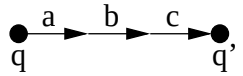
"$\curvearrowright$"

i. - iii. folgen aus der Definition von $f_{\mathfrak{A}} = f$.

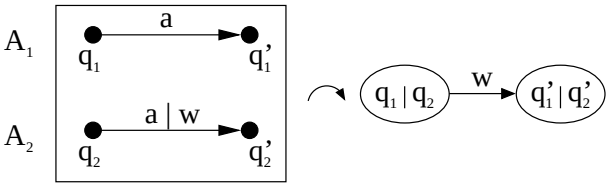
iv. a) $L \in RegL(\Sigma) : \mathfrak{A}_1 = \langle Q_1, \Sigma, \delta_1, q_0^1, F_1 \rangle \in DFA(\Sigma)$. und f ist allgemein-sequentiell: $\mathfrak{A}_2 = \langle Q_2, \Sigma, \Delta, \delta_2, q_0^2, \lambda_2 \rangle$

Zu zeigen: $f(L) \in RegL(\Sigma)$

Konstruktion von $\mathfrak{A} \in NFA(\Delta)$ mit $L(\mathfrak{A}) = f(L)$ durch Parallelkonstruktion und Selektion der Ausgabe.

Wortübergänge können weggelassen werden: Zwischenzustände ausführen 

$\mathfrak{A} = \langle Q, \Delta, \delta, q_0, F\rangle$ mit $\delta : Q \times \Delta^* \mapsto \wp(Q)$ mit $|\delta| < \infty$, $Q := {Q_1}^\circ \times Q_2$, $q_0 := (q_0^1, q_0^2)$, $F := F_1 \times Q$.
$\delta((q_1,q_2),w) \ni (q_1',q_2') :\curvearrowleft \exists a \in \Sigma$ mit $\delta_1(q,a) = q_1'$ und $\delta_2(q_2,a) = q_2'$ und $\lambda_2(q_2,a) = w$.



Es folgt: $w \in L(\mathfrak{A}) \rightleftarrows \exists v \in L$ mit $f(v) = w$. □

Folgerungen

1. Die Spiegelfunktion $\cdot^R : \Sigma^* \mapsto \Sigma^*$ ist nicht allgemein-sequentiell.
2. Die serielle Multiplikation von Binärzahlen ist nicht allgemein-sequentiell.

Beweis

1. $\cdot^R$ erfüllt nicht die Präfixtreue.
2. Die serielle Multiplikation von Zahlen sei definiert durch $f(\mathbb{B}^2)^* \mapsto \mathbb{B}^*$ mit
   $f(a_1,b_1)(a_2,b_2)...(a_n,b_n)) = a_1a_2...a_n \cdot b_1...b_n$ ($\cdot$ sei binär und a und b haben durch Führungsnullen die gleiche Länge).
   Dann erfüllt f nicht iv.a), d.h. f erhält nicht die Regularität.
   Sei $L \subseteq (\mathbb{B}^2)^*$ gegeben durch $(1,0)(0,1)^+ \in RegE(\mathbb{B}^2)$. $w = (1,0)(0,1)^k \in L$ stellt das Binärpaar $(10^k, 01^k)$ dar, welches die Zahlen 2^k und $2^k - 1$ repräsentiert,da $2^k - (2^k - 1) = 2^{2k} - 2^k$ die Binärdarstellung 1^k0^k hat.
   Also gilt: $f(L) = \{1^k0^k \mid k \geq 1\} \in RegL(\mathbb{B})$.

□

Kapitel 3

Kontextfreie Sprachen und Kellerautomaten

3.1 Kontextfreie Grammatiken

3.1.1 Definition: Kontextfreie Grammatiken

Seien N und Σ nicht-leere endliche Mengen mit $N \cap \Sigma = \emptyset$.
Sei $p \subseteq N \times (N \cup \Sigma)^*$ mit $|P| < \infty$ und $\rho \in N$.

dann heißt $G = \langle N, \Sigma, P, S \rangle$eine *kontextfreie Grammatik*.

Bezeichnung: $G \in CFG(\Sigma)$ oder $G \in CFG$.

Bezeichnung und Konvention

A, B, C...	$\in N$	Nichtterminalsymbole
a, b, c...	$\in \Sigma$	Terminalsymbole
$S \in N$		Startsymbol
X, Y, Z	$\in \chi := N \cup \Sigma$	Symbole
$\alpha, \beta, \gamma, ...$	$\in \chi^*$	Satzformen
u, v, w	$\in \Sigma^*$	Terminalwörter
$A \mapsto \alpha := (A, \alpha)$	$\in D$	Regel, Produktion

3.1.2 Ableitungsrelation

Definition

Sei $G = \langle N, \Sigma, P, S \rangle \in CFG$ und $\pi = A \mapsto \alpha \in P$.

π bestimmt eine *Ableitungsrelation* $\Rightarrow_\pi \subseteq \chi^* \times \chi^*$ mit $\beta_1 \Rightarrow_\pi \beta_2 \rightleftarrows \exists Kontext \gamma_1, \gamma_2 \in \chi^*$, so dass $\beta_1 = \gamma_1 \alpha \gamma_2$ und $\beta_2 = \gamma_1 \alpha \gamma_2$.

Dann heißt das Tripel $(\gamma_1, \pi, \gamma 2)$ auch Ableitungsschritt.

G bestimmt die Ableitungsrelation $\Rightarrow_G \subseteq \chi^* \times \chi^*$ durch $\Rightarrow_G := \bigcup_{\pi \in P} \Rightarrow_\pi$ und damit die von G erzeugte Sprache $L(G) := \{w \in \Sigma^* \mid S \Rightarrow_g^* w\}$.

$\Rightarrow_G^* := \bigcup_{n=0}^{\infty} \Rightarrow_G$ (Reflexive, transitive Hülle von $\Rightarrow_G$).

Es folgt: $w \in L(G) \rightleftarrows \exists \alpha_0, \alpha_1, ..., \alpha_n \in \chi^*$ und $\pi_1, ..., \pi_n \in P : S = \alpha_0 \Rightarrow_{\pi_1} \alpha_1 ... \Rightarrow_{\pi_n} \alpha_n = w$.

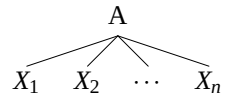
Bezeichnung: $CFL(\Sigma) = \{L \subseteq \Sigma^* \mid \exists G \in CFG(\Sigma) : L(G) = L\}$

3.1.3 Ableitungsbäume, Rechts- und Linksableitungen, eindeutigkeit/Mehrdeutigkeit

Sei $G = \langle N, \Sigma, P, S \rangle \in CFG(\Sigma)$.

Darstellung von Ableitungen durch Bäume

1. Eine Regel $\pi = A \mapsto X_1 ... X_n$ bestimmt den Regelbaum.

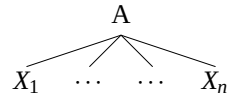
   

   Spezialform $A \mapsto \varepsilon$:

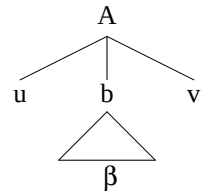
   

2. Eine Ableitung $A \Rightarrow \alpha_1 ... \Rightarrow \alpha_n$ bestimmt den Ableitungsbaum durch entpsrechendes Verkleben der Regelbäume. Es lässt sich durch Induktion über n definieren.

   $n = 1$: Regelbaum
   $n \rightsquigarrow n+1 : A \Rightarrow \alpha_1 \Rightarrow ... \Rightarrow \alpha_n$ haben den Ableitungsbaum:

   $\alpha_n \Rightarrow \alpha_{n+1}$ sei gegeben durch den Ableitungsschritt (u, π, v). Dann entsteht der Ableitungsbaum von $A \Rightarrow \alpha_1 ... \Rightarrow \alpha_n \Rightarrow \alpha_{n+1}$ durch Anhängen des Regelbaums von π zwischen u und v:
   Regel $\pi = B \mapsto \beta$

Folgerung

Ein Ableitungsbaum repräsentiert eine Klasse von Ableitungen, die sich nur in der Reihenfolge von Ableitungsschritten unterscheiden.

3.1.4 Rechts- und Linksableitung

Definition

Eine Ableitung heißt *Rechtsableitung* (bzw. *Linksableitung*), wenn jeder Ableitungsschritt (α, π, β) ein Rechtsableitungsschritt ist, d. h. $\beta \in \Sigma^*$ (bzw. ein Linksableitungsschritt ist, d.h. $\alpha \in \Sigma^*$).

Schreibweise

$\Rightarrow_l$ bzw. $\Rightarrow_r$ für Links- bzw. Rechtsableitungsschritte.

Folgerung

Ein Ableitungsbaum hat genau eine Rechts- und genau eine Linksableitung.

3.1.5 Eindeutigkeit und Mehrdeutigkeit

Definition

Sei $G \in CFG(\Sigma)$ *eindeutig* $\rightleftarrows$ zu jedem $w \in L(G)$ gibt es genau eine Rechtsableitung (Linksableitung) $G \Rightarrow_r \alpha_1 \Rightarrow_r ... \Rightarrow_r w$.

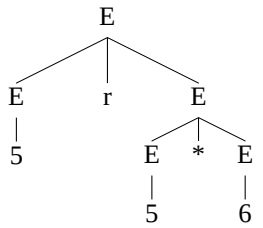
G mehrdeutig $\rightleftarrows$ G nicht eindeutig.

Beispiel

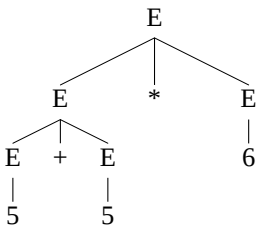
$E \mapsto E+E \mid E*E \mid 5 \mid 6$ ist mehrdeutig.

$5+5*6$.

$E \Rightarrow E+E \Rightarrow E+E*E \Rightarrow^3 5+5*6$ (Rechtsableitung)



$E \Rightarrow E*E \Rightarrow E+E*E \Rightarrow^3 5+5*6$ (Linksableitung)
$E \Rightarrow E*E \Rightarrow E*6 \Rightarrow E+E*6 \Rightarrow E+5*6 \Rightarrow 5+5*6$ (Rechtsableitung, verschieden von der ersten)



3.2 Einseitig-lineare Grammatiken

Definition

Sei $G = \langle N, \Sigma, P, S \rangle \in CFG(\Sigma)$.

- Dann heißt L *linkslinear*, wenn für jedes $\pi = A \mapsto \alpha \in P$ gilt: $a = Bw$ oder $\alpha = w$ für ein $B \in N$ und $w \in \Sigma^*$
- Gilt stattdessen $\alpha = wB$ oder $a = w$ für jedes $\pi \in P$, so heißt G *rechtslinear*.
- G einseitig linear $\rightleftarrows$ G rechtslinear oder G linkslinear

Ziel

$\{L(G) \mid \text{G linkslinear}\} = \{L(G) \mid \text{G rechtslinear}\} = \{L(G) \mid \text{G einseitig linear}\} = RegL(\Sigma)$.

Satz

$\mathfrak{L}(\Sigma, NFA) = \{L \subseteq \Sigma^* \mid L = L(G), \text{ G rechtslinear }\}$.

Beweis

"$\supseteq$"

Sei $G = \langle N, \Sigma, P, S\rangle$ rechtslinear.
Auffassung von G als $\mathfrak{A} = \langle Q, \Sigma, \delta, q_0, F\rangle \in NFA(\Sigma)$:

$Q := N \cup \{q_f\} (q_f neu)$
$q_0 := S$
$F := \{q_f\}$
$\delta(A, w) \ni B$ falls $A \mapsto wB$ in G
$\delta(A, w) \ni q_f$ falls $A \mapsto w$
offensichtlich: $L(G) = L(\mathfrak{A})$. □

"$\subseteq$"

$\mathfrak{A} = \langle Q, \Sigma, \delta, q_0, F\rangle \in NFA(\Sigma)$ kann als rechtslineare Grammatik aufgefasst werden (ähnlich wie oben).
Unterschied: $q \in F \mapsto$ neue Regel $q \mapsto \varepsilon$ □

Korollar

$RegL(\Sigma) \subseteq CFL(\Sigma)$

Für $|\Sigma| \geq 2$ ist diese Inklusion echt, weil $\{a^n b^n \mid n \in \mathbb{N}\} = L(G)$ mit $G = (S \mapsto aSb \mid \varepsilon)$.

Lemma

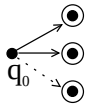
Zu $\mathfrak{A} \in DFA(\Sigma)$ konstruieren wir $\mathfrak{A}^R \in NFA(\Sigma)$ mit $L(\mathfrak{A}^R) = L(\mathfrak{A})^R$.

Idee

Ersetze q —a→ q' durch q ←a— q'

Ersetze —→ q_0 durch ◉

und ergänze diesen NFA durch:



□

Korollar

$\{L \subseteq \Sigma^* \mid L = L(G), \text{ G rechtslinear }\}$ = $\{L \subseteq \Sigma^* \mid L = L(G), \text{ G linkslinear }\}$

Beweis

L rechtslinear erzeugbar $\rightleftarrows L \in RegL \rightleftarrows L^R \in RegL \rightleftarrows L^R$ rechtslinear erzeugbar $\rightleftarrows L \in L^R$ linkslinear erzeugbar (Regeln spiegeln) □

3.3 Normalform von kontextfreien Grammatiken

Hilfsmittel: Vorgänger von Satzformen $\alpha, \beta \in \chi^*$

$\alpha \Rightarrow \beta$ α direkter Vorgänger von β
$\alpha \Rightarrow^* \beta$ α Vorgänger von β

3.3.1 Vorgängermenge

Definition

Sei $G = \langle N, \Sigma, P, S \rangle \in CFG$ und $L \subseteq \chi^*$.
Dann ist

i. die direkte *Vorgängermenge von L* definiert durch $\underline{Pre}_G(L) := \{\alpha \in \chi^* \mid \exists \beta \in L : \alpha \Rightarrow_G \beta\}$
ii. der *Vorgängerabschluss von L* definiert durch $\underline{Pre}^*_G(L) := \{\alpha \in \chi^* \mid \exists \beta \in L : \alpha \Rightarrow^*_G \beta\}$

Lemma

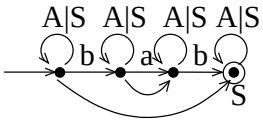
Sei $G = \langle N, \Sigma, P, S \rangle \in CFG$ und $L \subseteq \chi^*$.
Dann gilt: $L \in RegL(\chi) \curvearrowright \underline{Pre}^*_G(L) \in RegL(\chi)$.

Beweis

Sei $\mathfrak{A} = \langle Q, \chi, \delta, q_0, F \rangle \in NFA$ mit $L(\mathfrak{A}) = L$.
Konstruiere $\mathfrak{A}' = \langle Q, \chi, \delta', q_0, F \rangle \in NFA$ durch

i. $\delta(q,a) \subseteq \delta'(q,a) \forall (q,a) \in Q \times \chi_\varepsilon$
ii. Wenn $A \mapsto \alpha \in P$ und $\overline{\delta}'(\{q\}, \alpha) \ni q'$ so $\delta'(q,A) \ni q'$ hinzufügen.

Ergänze δ' um solche Transitionen, solange dies möglich ist.
Dieser Erweiterungsautomat terminiert, weil Q und χ endlich sind.



□

Bemerkung

Die Zeitkomplexität der Automatenkonstruktion liegt in $O(|Q|^6)$.

3.3.2 Erreichbarkeit

Definition

Sei $G \in CFG(\Sigma)$ und $A \in N$:

i. A heißt produktiv $\rightleftarrows \exists w \in \Sigma^* : A \Rightarrow^* w$
ii. A heißt erreichbar $\rightleftarrows \exists \alpha, \beta \in \chi^* : S \Rightarrow \alpha A \beta$

Folgerung

A produktiv $\rightleftarrows A \in \underline{Pre}^*_G(\Sigma^*)$
A erreichbar $\rightleftarrows S \in \underline{Pre}^*_G(\chi^* A \chi^*)$

3.3.3 Reduzierte Grammatik

Definition

$G \in CFG(\Sigma)$ heißt *reduziert* gdw. entweder $P = \emptyset$ oder jedes $A \in N$ produktiv und erreichbar ist.

Lemma

Jede Grammatik $G \in CFG(\Sigma)$ lässt sich in eine äquivalente reduzierte Grammatik $G' \in CFG(\Sigma)$ transformieren.

Beweis

Stelle für jedes $A \in N$ fest, ob A produktiv und erreichbar ist (Vorgängerabschluss).

Fall 1: Wenn S nicht produktiv dann wähle $P := \emptyset$.

Fall 2: Wenn S produktiv, lasse alle $A \in N$, die nicht produktiv oder erreichbar sind weg, sowie alle Regeln, in denen solches A's vorkommen.

Korollar

Das $\emptyset$-Problem von CFG's ist entscheidbar.

3.3.4 Elimination von ε-Regeln

Definition

$G \in CFG(\Sigma)$ heißt ε-frei gdw. $A \mapsto \varepsilon \in P \curvearrowright A = S$ und S auf keiner rechten Regelseite steht.

Lemma

$X \Rightarrow \varepsilon \rightleftarrows X \in \underline{Pre}^*_G(\{\varepsilon\})$

Satz

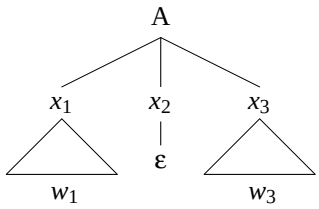
Jedes $G \in CFG(\Sigma)$ lässt sich in eine äquivalente ε-freie Grammatik $G' \in CFG(\Sigma)$ transformieren.

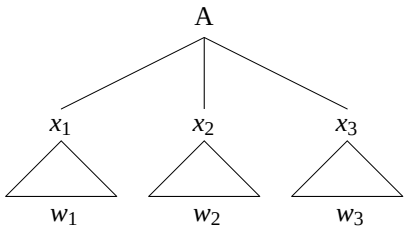
Beweis

Konstruiere mit $\underline{Pre}^*(\{\varepsilon\})$ die Menge $N_\varepsilon := \{A \in N \mid A \Rightarrow^* \varepsilon\}$ und damit $G' = \langle N', \Sigma, P', S' \rangle \in CFG(\Sigma)$.

- $N' := N \cup \{S'\}$ (neues Startsymbol)
- $P' := \{S \mapsto S'\} \cup \{S' \mapsto \varepsilon \mid S \in N_\varepsilon\} \cup \{A \mapsto x_1, ..., x_k \mid k \geq 1, x_i \in (N \cup \Sigma), \exists \alpha_0, ..., \alpha_k \in N_\varepsilon^* : A \mapsto \alpha_0 X_1 \alpha_1 ... \alpha_k, X_k \subset P\}$.

Beachte: Wegen $k \geq 1$ entfallen ε-Regeln und G' ist ε-frei.





Bleibt zu zeigen: $L(G') = L(G)$

1. $w = \varepsilon$
   $\varepsilon \in L(G') \rightleftarrows S' \mapsto \varepsilon \in P \rightleftarrows S \in N_\varepsilon \Rightarrow_G \varepsilon \rightleftarrows \varepsilon \in L(G)$
2. $w \neq \varepsilon$

   (i) $w \in L(G') \curvearrowright S \Rightarrow_{G'}^* w \curvearrowright S \Rightarrow_{G'}^* w \curvearrowright w \in L(G)$ weil Ableitung in G' mit $\alpha_i \Rightarrow^* \varepsilon$ aufgefüllt werden kann.

   (ii) $w \in L(G)$
   Wir zeigen durch Induktion über die Ableitungslänge, dass gilt:
   $A \Rightarrow_G^r w \in \Sigma^+ \curvearrowright A \Rightarrow_G^* w$
   $\mathbf{r} = \mathbf{1} : A \Rightarrow_G, w \in \Sigma \curvearrowright A \mapsto w \in P \curvearrowright A \mapsto w \in P' \curvearrowright A \Rightarrow_G^* w$
   $\mathbf{r} \mapsto \mathbf{r} + \mathbf{1} : A \Rightarrow x_1 ... x_k \Rightarrow_G^r w$
   Dann existiert eine Ableitung $x_i \Rightarrow_G^{r_i} w_i$ mit $w = w_1 ... w_k$ mit $r_i \leq r$.
   Falls $w_i \neq \varepsilon$ dann $x_i \Rightarrow_G^* w_i$ nach Induktionsvoraussetzung.
   Falls $w_i = \varepsilon$ dann $x_i \in N_\varepsilon$. Durch entsprechendes Löschen entsteht $A \Rightarrow x'_1 ... x'_k \Rightarrow^* w'_1 ... w'_k$

□

3.3.5 Elimination von Kettenregeln

Definition

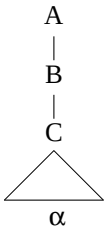
Eine Regel der Form $A \mapsto B$ heißt *Kettenregel*.

Satz

Jedes $G = \langle N, \Sigma, P, S \rangle \in CFG(\Sigma)$ lässt sich in eine äquivalente ε-freie Grammatik $\overline{G} = \langle \overline{N}, \Sigma, \overline{P}, \overline{S} \rangle \in CFG(\Sigma)$ ohne Kettenregeln transformieren.

Beweis

Idee: Eliminiere $A \mapsto B$ und $B \mapsto C$ und ergänze $C \mapsto \alpha$ um $B \mapsto \alpha$ und $A \mapsto \alpha$.



Transformiere G in äquivalente ε-freie Grammatik $G' = \langle N', \Sigma, P', S' \rangle$. Dann definieren wir $\overline{G} = \langle \overline{N}, \Sigma, \overline{P}, \overline{S} \rangle$ durch
$\overline{N} := N'$
$\overline{S} := S'$
$A \mapsto \alpha \in \overline{P} \rightleftarrows \alpha \ni N'$ und $\exists B \mapsto \alpha P'$ mit $A \in \underline{Pre}^*(B)$.
Die Korrektheit folgt, da wegen der ε-Freiheit von G gilt $A \in \underline{Pre}^*(B) \rightleftarrows A \Rightarrow A_1 \Rightarrow A_2 ... \Rightarrow A_n = B$. □

3.3.6 Die Chomsky-Normalform

Definition

$G = \langle N, \Sigma, P, S \rangle \in CFG(\Sigma)$ ist in *Chomsky-Normalform* ($G \in CNF$) genau dann wenn jede Regel von P folgende Form hat:

- $A \mapsto BC$ mit $B, C \in N$ oder
- $A \mapsto \alpha$ mit $\alpha \in \Sigma$ oder
- $S \mapsto \varepsilon$.

Satz

Jedes $G = \langle N, \Sigma, P, S \rangle \in CFG(\Sigma)$ lässt sich in eine äquivalente Grammatik $G' \in CNF$ transformieren.

Beweis

O.B.d.A. sei G ε-frei und ohne Kettenregel. Außerdem können wir annehmen, dass für
$K \geq 2 : A \mapsto x_1 ... x_k \in P \curvearrowright x_i \in N$.
Denn $x_i = a$ kann ersetzt werden durch neues $C_a \in N$ unter Hinzufügen von $C_a \mapsto a$.
Bleibt die Elimination von Regeln der Form $A \mapsto B_1 ... B_k$ mit $k \geq 3$. Ersetzen durch:
$A \mapsto B_1 C_1$
$C_1 \mapsto B_2 C_2$
$C_2 \mapsto B_3 C_3$
$\vdots$
$C_{k-2} \mapsto B_{k-1} B_k$
mit neuen N-Symbolen $C_1, ..., C_{n-2}$. □

3.3.7 Die Greibach-Normalform, Linksrekursion

Definition: Greibach-Normalform

$G = \langle N, \Sigma, P, S \rangle \in CFG(\Sigma)$ ist in *Greibach-Normalform* ($G \in GNF$) genau dann, wenn jede Regel von P folgende Form hat:

- $A \mapsto aB_1 ... B_n$ oder
- $A \mapsto a$ oder
- $S \mapsto \varepsilon$ und S auf keiner rechten Regelseite.

Satz

Jedes $G \in CFG(\Sigma)$ lässt sich in eine äquivalente $G' \in GNF$ transformieren.

Beweisidee

Elimination linksrekursiver N-Symbole.

Definition: Linksrekursiv

$A \in N$ heißt *linksrekursiv* genau dann, wenn $A \Rightarrow^+ A\alpha$ für $\alpha \in \chi^*$.

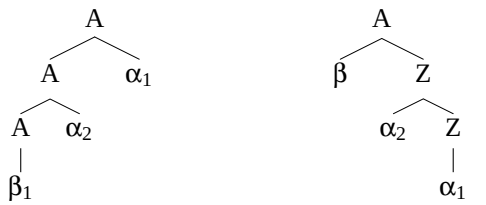
Spezialfall: Direkte Linksrekursion

Seien $A \mapsto A\alpha_1 \mid A\alpha_2 \mid ... \mid A\alpha_r$ alle Regeln der Form $A \mapsto A\alpha$
und $A \mapsto \beta_1 \mid ... \mid \beta_s$.

dann lässt sich dieser Regelsatz äquivalent ersetzen durch

$A \mapsto \beta_1 Z \mid ... \mid \beta_s Z \mid \beta_1 \mid ... \mid \beta_s$ (Z ist neues N-Symbol)
$Z \mapsto \alpha_1 Z \mid ... \mid \alpha_r Z \mid \alpha_1 \mid ... \mid \alpha_r$

Grund für Korrektheit:



Bemerkung: Linksrekursion stört die Syntaxanalyse.

3.4 Abschlusseigenschaften von CFL

3.4.1 Substitution

Definition

Sei Σ ein Alphabet und für jedes $a \in \Sigma$ sei Σ_a ein weiteres Alphabet.
$\Delta = \bigcup_{a \in \Sigma} \Sigma_a$
dann heißt $\varphi : P(\Sigma^*) \mapsto P(\Delta^*)$ eine *Substitutionsabbildung*, falls

- $\varphi(\{a\}) \subseteq \Sigma_a^*$
- $\varphi(\{\varepsilon\}) = \{\varepsilon\}$
- $\varphi(\{a_1 ... a_k\}) = \varphi(\{a_1\}) \cdot ... \cdot \varphi(\{a_k\})$
- $\varphi(L) = \bigcup_{w \in L} \varphi(\{w\})$

Schreibweise: $\varphi(w) := \varphi(\{w\})$

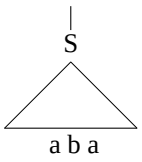
Substitutionssatz

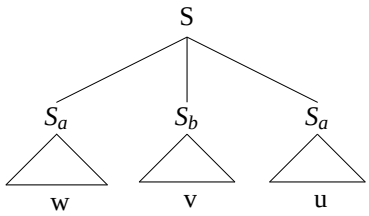
$L \in CFL(\Sigma), \varphi(a) \in CFL(\Sigma_a)$.
Dann ist $\varphi(L) \in CFL(\Delta)$.

Beweisidee

Kombination der Grammatiken für L und $\varphi(a)$ mit disjunkten N-Symbolen und Ersetzen der $a \in \Sigma$ durch S_a (Startsymbol der Grammatik für $\varphi(a)$).

$$(aba \in L) \quad \mapsto \quad (wvu \in \varphi(L))$$





□

Korollar

Die Klasse der kontextfreien Sprachen (CFL) ist abgeschlossen unter regulären Operationen.

Beweis

Sei $L_1, L_2 \in CFL(\Sigma), \Sigma' := \{a,b\}, \varphi(\{a,b\}) := L_1$ und $\varphi(b) := L_2$. Dann gilt:

- $\varphi(\{a,b\} = L_1 \cup L_2$
- $\varphi(ab) = L_1 L_2$
- $\varphi(\{a\}^*) = L_1^*$

Da $\{a,b\}, \{ab\}$ und $\{a\}^*$ in $CFL(\{a,b\})$ liegen, folgt: $L_1 \cup L_2, L_1 L_2, L_1^* \in CFL(\Sigma)$. □

3.4.2 Pumping Lemma

Sei $L \in CFL(\Sigma)$.

Dann existiert $k \geq 1$ so, dass für alle Worte $z \in L$ mit $|z| \geq k$ gilt:

Es existiert eine Zerlegung der Form $z = uvwxy$ mit

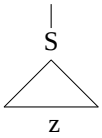
- $|vwx| \leq k$
- $vx \neq \varepsilon$
- $uv^i wx^i y \in L$ für alle $i \in \mathbb{N}$.

Das Pumping Lemma wird z.B. verwendet, um zu zeigen, dass CFL unter $\cap$ und - nicht abgeschlossen ist.

Beweis

O.B.d.A. sei $G = \langle N, \Sigma, P, S \rangle \in CNF$ mit $L(G) = L$. Es sei weiterhin $n := |N|$, $k := 2^n$ und $z \in L$ mit $|z| \geq k$.

Ein Ableitungsbaum t: hat mindestens 2^n Blätter.



Da $G \in CNF$, muss ein Pfad maximaler Länge mindestens $n+1$ Kanten, also $n+2$ Knoten besitzen. Der Pfad p besitzt daher mindestens $n+1$ Knoten markiert durch N-Symbole.
Also: 2 Kanten mit demselben N-Symbol.
Wähle auf Pfad p den letzten Knoten, dessen Marke $A \in N$ sich wiederholt. Dann hat sein Teilbaum höchstens $2^n = k$ Blätter.

- $|vwx| \leq k$
- $vx \neq \varepsilon$ (Binärdarstruktur: Verzweigung bei A)
- $uv^2wx^2y \in L$ ("Schnippeln")

□

Lemma

$L = \{a^n b^n c^n \mid n \geq 1\} \notin CFL(\{a,b,c\})$

Beweis

Angenommen L sei kontextfrei, so existiert ein Pumpingindex $k \in \mathbb{N}$ und $w = a^k b^k c^k$ besitzt eine Zerlegung $w = uvwxy$.

$$\left.\begin{array}{l} |vwx| \leq k \\ vx \neq \varepsilon \\ uwy \in L \end{array}\right\} \text{vx kann nicht gleichzeit ein a und ein c enthalten}$$

$|vwx| < 3k$

Entweder $uwy = a^k \ldots$ oder $uwy = \ldots c^k \rightsquigarrow$ Widerspruch zu $L \in CFL \curvearrowright L \notin CFL$. □

Satz

CFL ist weder unter Durchschnitt noch unter Komplement abgeschlossen.

Beweis

$$\left\{ \begin{array}{lcl} S & \mapsto & Sc \mid Ac \\ A & \mapsto & aAb \mid ab \end{array} \right\} \quad \text{und} \quad \left\{ \begin{array}{lcl} S & \mapsto & aS \mid aA \\ A & \mapsto & bAc \mid bc \end{array} \right\}$$

$$L = \{a^n b^n c^p \mid n, p \geq 1\} \qquad L' = \{a^p b^n c^n \mid n, p \geq 1\}$$

$L, L' \in CFL$
$L \cap L' = \{a^n b^n c^n \mid n \geq 1\} \notin CFL.$

Da CFL unter $\cup$ abgeschlossen ist, kann CFL unter dem Komplement nicht abgeschlossen sein, denn: $L_1 \cap L_2 = \overline{\overline{L_1} \cup \overline{L_2}}$. □

3.5 Entscheidbare Eigenschaften von CFG

Satz

Das Wortproblem und das $\emptyset$-Problem sind für CFG entscheidbar.

Beweis

$w \in L(G) \rightleftarrows S \in \underline{Pre}^*(\{w\})$
$L(G) = \emptyset \rightleftarrows S \notin \underline{Pre}^*(\Sigma^*)$ □

Bemerkung

1. Das Wortproblem ist in $O(n^3)$ entscheidbar
2. Das Äquivalenzproblem ist nicht entscheidbar. Die Mehrdeutigkeit ist ebenfalls nicht entscheidbar.

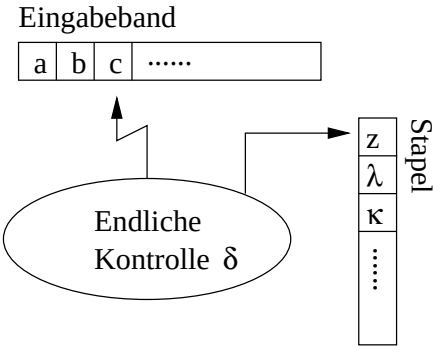
3.6 Kellerautomaten

- Der Kellerspeicher (Stack, Stapel, push down store) ist eine wichtige Datenstruktur
- Bindeglied zwischen Wärtern und Bäumen
- Charakterisierung der kontextfreien Sprachen durch nicht-deterministische Kellerautomaten (keine Äquivalenz zu deterministischen Kellerautomaten).
- Bedeutung der Kellerautomaten für den Compilerbau:
  - Syntaxanalyse
  - Datenkeller (Auswertung von Rechenausdrücken)
  - Prozedurkeller (Speichertechnik für rekursive Prozeduren und Methoden)

3.6.1 Definition eines Kellerautomaten

Seien Q, Σ, Γ nicht-leere endliche Mengen von Zuständen, Eingabesymbolen und Kellersymbolen; ferner : $q_0 \in Q$ Anfangszustand, $F \subseteq Q$ Endzustandsmenge, $Z_0 \in \Gamma$ Kellerautomatsymbol und $\delta : Q \times \Sigma_\varepsilon \times \Gamma \mapsto P_f(Q \times \Gamma^*)$ mit $\Sigma_\varepsilon := \Sigma \cup \{\varepsilon\}$ Transitionsfunktion. P_f sei eine endliche Teilmenge.

Dann heißt $\mathfrak{A} = \langle Q, \Sigma, \Gamma, \delta, q_0, Z_0, F \rangle$ ein Kellerautomat über Σ. Bezeichnung: $\mathfrak{A} \in PDA(\Sigma)$.



3.6.2 Semantik eines Kellerautomaten

Konfigurationsmenge $Q \times \Sigma^* \times \Gamma^*$ (Kellerspitze links)

Einzelschrittrelation $(q, w, \alpha) \vdash (q', w', \alpha')$

- Σ-Schritt: $(q, aw, z\alpha) \vdash (q', w, \beta\alpha)$ falls $\delta(q, a, z) \ni (q', \beta)$
- ε-Schritt $(q, w, z\alpha) \vdash (q', w, \beta\alpha)$ falls $\delta(q, \varepsilon, z) \ni (q', \beta)$

Beachte: ε-Schritte lassen das Eingabeband unverändert.

3.6.3 Die vom Automaten $\mathfrak{A}$ erkannte Sprache

Es gibt drei Möglichkeiten der Erkennung:

1. mit Endzustand
   $L(\mathfrak{A}, F) := \{w \in \Sigma^* \mid (q_0, w, z_0) \vdash^* (q_f, \varepsilon, \alpha), q_f \in F\}$
2. mit leerem Keller
   $L(\mathfrak{A}, \varepsilon) := \{w \in \Sigma^* \mid (q_0, w, z_0) \vdash^* (q, \varepsilon, \varepsilon)\}$
3. mit beidem
   $L(\mathfrak{A}, F, \varepsilon) := \{w \in \Sigma^* \mid (q_0, w, z_0) \vdash^* (q_F, \varepsilon, \varepsilon), q_f \in F\}$

Im allgemeinen sind diese Sprachen verschieden.

Lemma

$$\mathfrak{L}(\Sigma, PDA, F) = \mathfrak{L}(\Sigma, PDA, \varepsilon) = \mathfrak{L}(\Sigma, PDA, F, \varepsilon) =: \mathfrak{L}(\Sigma, PDA)$$

Diese Sprachen nennt man auch "Kellersprachen".

3.6.4 Nulldeterminismus

$\delta(q,a,A) = \{(q_1,\alpha_1),(q_2,\alpha_2)\}$
$\delta(q,\varepsilon,A) = \{(q'_1,\alpha'_1),(q'_2,\alpha'_2)\}$

Folgezustände von $(q,aw,A\beta) \vdash (q1,w,\alpha_1\beta)$
$\vdash (a_1,aw\alpha'_1\beta)$
$\vdash \ldots$

Ziel: Zeige, dass Kellersprachen = CFL

Satz

$CFL(\Sigma) \subseteq \mathfrak{L}(\Sigma,PDA)$

Beweis

$L = L(G), G = \langle N,\Sigma,P,S\rangle$. Konstruiere einen Automaten $\mathfrak{A}_G \in PDA(\Sigma)$ mit $L(\mathfrak{A}_G,\varepsilon) = L(G)$.

Idee:

1. Simulation der Linksableitungen auf Keller.
   "Rate" dabei den richtigen Ableitungsschritt (Nicht-Determinismus).
   $\mathfrak{A} = \langle Q,\Sigma,\Gamma,\delta,q_0,z_0\rangle \in PDA(\Sigma)$ mit $Q := \{q\}, \Gamma := N \cup \Sigma, z_0 := S$.
   Ableitungsschritte: $\delta(q,\varepsilon,A) := \{(q,\beta) \mid A \mapsto \beta \in P\} \forall A \in N$.
2. Vergleich der Terminalsymbole des Kellers mit der Eingabe.
   Vergleichsschritte: $\delta(q,a,a) = \{(q,\varepsilon)\} \forall a \in \Sigma$.

Zum Nachweis von $L(G) = L(\mathfrak{A}_G)$ zeigen wir, dass für alle $A \in N$ und $w \in \Sigma^*$ gilt:

$$A \Rightarrow_G^* \rightleftarrows (q,w,A) \vdash^* (q,\varepsilon,\varepsilon)$$

Beachte: Für alle $v \in \Sigma^*, \alpha \in \chi^* : (q,wv,A\alpha) \vdash^* (q,v,\alpha)$.

Beweis

"$\curvearrowright$": Induktion über die Ableitungslänge:
$n = 1 : A \mapsto w \curvearrowright (q,w,A) \vdash (q,w,w) \vdash^* (q,\varepsilon,\varepsilon)$.
$n \rightsquigarrow n+1 : A \Rightarrow v_0A_1v_1 \ldots A_rv_r \Rightarrow^n w$.
Dann muss $A_i \Rightarrow^{n_i} w_i, w = v_0w_1v_1 \ldots w_rv_r, n_i \leq n$, also gilt dafür die Induktionsvoraussetzung.
$(q,w,A) \vdash (q,w,v_0A_1v_1 \ldots A_rv_r) \vdash^* (q,w_1v_1 \ldots w_rv_r,A_1v_1 \ldots A_rw_r) \vdash_{IV}^* (q,v_1 \ldots w_rv_r,v_1 \ldots A_rv_R) \vdash^* (q,\varepsilon,\varepsilon)$.

"$\curvearrowleft$": Induktion übern die Länge der Berechnung:
$i = 1 : (q,w,A) \vdash (q,\varepsilon,\varepsilon) \curvearrowleft A \mapsto \varepsilon, w = \varepsilon \curvearrowleft A \Rightarrow^* w$.
$i \rightsquigarrow i+1$: Ähnlich wie für "$\curvearrowright$". □

Satz

$\mathfrak{L}(\Sigma,PDA) \subseteq CFL(\Sigma)$

Beweis Literatur

Korollar

CFL ist unter Schnitt mit regulären Sprachen abgeschlossen.

Beweis

Für $L \in CFL(\Sigma)$ und $R \in RegL(\Sigma)$ existiert ein Automat $\mathfrak{A}_L \in PDA(\Sigma)$ und $\mathfrak{A}_R \in DFA(\Sigma)$ mit $L = L(\mathfrak{A}_L, F_L)$ und $R = L(\mathfrak{A}_R)$.

Konstruiere $\mathfrak{A}_L \| \mathfrak{A}_R \in PDA(\Sigma)$ durch *Parallelkonstruktion*.

Also: $Q := Q_L \times Q_R$. $\delta((q_1,q_2),a,Z) \ni ((q_1',q_2'),\alpha) \rightleftarrows \delta_L(q_1,a,Z) \ni (q_1',\alpha)$ *und* $\delta_R(q_2,a) = q_2'$, $f := F_L \times F_R$. □

Ziel: Der Deklarationszwang von Variablen ist keine kontextfreie Eigenschaft.

Lemma

$L = \{ww \mid w \in \{a,b\}^+\} \notin CFL$

Beweis

Zunächst zeigen wir: $L' = \{a^m b^n a^m b^n \mid m,n \geq 1\} \notin CFL$.

Wäre $L' \in CFL$, so ergäbe sich mit dem Pumping-Index k:
$a^k b^k a^k b^k = uvwxy$ mit $|vwx| \leq k, vx \neq \varepsilon$ also auch $uwy \in L'$. Dies ist ein Widerspruch, weil die Zahl der vorderen und hinteren a's bzw. b's nicht mehr gleich sind.

$L' = L \cap \{a\}^+\{b\}^+\{a\}^+\{b\}^+$.

Wäre $L \in CFL$, so auch L'. Dies ist ein Widerspruch. □

Korollar

Die Menge P der Pascal-Programme ist nicht kontextfrei.

Beweis

$L := \{\texttt{program T (output);var w : integer;begin w := 1 end.} \mid w \in \{a,b\}^+\}$

Also: $L \subseteq P$
R sei bestimmt durch den regulären Ausdruck

$$\texttt{program T (output); var (a} \vee \texttt{b)}^+ \texttt{ : integer; begin (a} \vee \texttt{b)}^+ \texttt{ := 1 end.}$$

Dann gilt: $L = P \cap R$.

h sei eine Substitutionsabbildung mit $h(a) = a, h(b) = b, h(x) = \varepsilon$ wenn $x \notin \{a,b\}$.

Dann: $h(L) = \{ww \mid w \in \{a,b\}^+\} \notin CFL \curvearrowright L \notin CFL \curvearrowright P \in CFL$. □

3.6.5 Deterministische Kellerautomaten

Definition

$\mathfrak{A} = \langle Q, \Sigma, \Gamma, \delta, q_0, Z_0, F \rangle$ heißt *deterministisch*, in Zeichen $\mathfrak{A} \in DPDA(\Sigma)$, wenn gilt:

i. $|\delta(q,a,Z)| \leq 1 \quad \forall (q,a,Z) \in Q \times \Sigma_\varepsilon \times \Gamma$

ii. $|\delta(q,\varepsilon,Z) = 1 \curvearrowright |\delta(q,a,z)| = 0 \quad \forall a \in \Sigma$

Folgerung

Zu jeder Konfiguration (a,w,α) gibt es höchstens eine Folgekonfiguration.

3.6.6 Erkennung durch Endzustände

$L(\mathfrak{A}) := L(\mathfrak{A},F) = \{w \in \Sigma^* \mid (q_0,w,Z_0) \vdash^* (q,\varepsilon,\alpha)$ mit $q \in F$ und $\alpha \in \Gamma^*\}$.

Es gibt nicht die Äquivalenz zur Erkennung mit leerem Keller bzw. leerem Keller und Endzuständen. Es gilt:

$\mathfrak{L}(\Sigma, DPDA, \varepsilon) \subset \mathfrak{L}(\Sigma, DPDA, F)$ und $\mathfrak{L}(\Sigma, DPDA, \varepsilon, F) \subset \mathfrak{L}(\Sigma, DPDA, F) =: \mathfrak{L}(\Sigma, DPDA)$.

3.6.7 Hiobsbotschaft

Satz: $\mathfrak{L}(\Sigma, DPDA)$ ist unter Komplement abgeschlossen.

Beweisidee

Zu jedem $\mathfrak{A} \in DPDA(\Sigma)$ ist ein äquivalenter $\tilde{\mathfrak{A}} \in DPDA(\Sigma)$ konstruierbar, so dass gilt:

Für jedes $w \in \Sigma^*$ gibt es $q \in \tilde{Q}$ und $\alpha \in \tilde{\Gamma}^*$ mit $(\tilde{q_0},w,\tilde{Z_0}) \vdash^* (q,\varepsilon,\alpha)$ Endkonfiguration mit $w \in L(\mathfrak{A}) \rightleftarrows q \in F$.

Dami ist die Komplementabbildung wie bei DFAs möglich.

Kern des Beweises

Elimination von *Schleifenkonfigurationen*, d.h. $(q,w,\alpha) in Q \times \Sigma^* \times \Gamma^*$ heißt Schleifenkonfiguration, falls für jedes $i \in \mathbb{N}$ ein $q_i \in Q$ und ein $\alpha_i \in \Gamma^*$ existiert, so dass $|\alpha_i| \geq |\alpha|$ und $(q,w,\alpha) \vdash^i (q_i,w,\alpha_i)$.

Diese Eigenschaft ist entscheidbar $\curvearrowright$Elimination. □

Folgerung

$\mathfrak{L}(\Sigma, DPDA) \subset \mathfrak{L}(\Sigma, PDA) = CFL$.

3.7 Der Algorithmus von Cocke, Younger und Kasami

Der Algorithmus ist eine effiziente Lösung des Wortproblems für beliebiege CFG. Er wird mittels dynamischer Programmierung implementiert. Er läuft in $O(n^3)$-Zeit und in $O(n^2)$-Platz.

Sei $G \in CNF$ (auf beliebige Grammatiken übertragbar).

Für $G = \langle N, \Sigma, P, S \rangle \in CNF$ $(A \mapsto BC, A \mapsto A), w = a_1 \dots a_n, n > 0$ definieren wir:

$w_{ij} = a_i \dots a_j$ für $1 \leq i \leq j \leq n$ (Teilwort)

$N_{ij} := \{A \in N \mid A \Rightarrow_G^* w_{ij}\}$

Es gilt: $w \in L(G) \rightleftarrows S \in N_{1n}$

i. $A \in N_{ii} \rightleftarrows A \mapsto a_i \in P$
   Berechne: $N_{11}N_{22} \dots N_{mn}$
   Dann: $N_{12}N_{23} \dots N_{m-1n}$
   $N_{13}N24 \dots N_{m-2n}$ usw.

ii. $A \in N_{ij}$ mit $i < j \rightleftarrows \exists k, i \leq k < j, \exists A \mapsto BC \in P : B \subset N_{ik}$ und $C \in N_{k+1j}$

3.7.1 Komplexität

Zeit

"for k": $c \cdot d$
"for i": $c \cdot d(n-d)$
"for d": $c \sum_{d=1}^{n-1} d(n-d) = c(\sum_{d=1}^{n} (nd - d^2)) = c(n \underbrace{\sum_{d=1}^{n} d}_{\frac{n}{2}(n+1)} - \underbrace{\sum_{d=1}^{n} d^2}_{\frac{n \cdot (n+1)(2n+2)}{6}} = c\frac{m^3-n}{6} = O(n^3)$

Speicherplatz

Die Speicherplatzkomplexität ist $O(n^2)$ wie man sich leicht überlegen kann, da eine Tabelle mit $n \cdot n$ Einträgen angelegt wird.

3.8 Erweiterte kontextfreie Grammatiken (ECFG)

Erweiterte kontextfreie Grammatiken bieten einen höheren Beschreibungskomfort durch reguläre Ausdrücke als rechte Regelseite. Es handelt sich um eine zur nicht erweiterten kontextfreien Grammatik äquivalente Darstellung.

Definition

$G = \langle N, \Sigma, P, S \rangle$ ist *erweiterte CFG*, in Zeichen $G \in ECFG$, wenn $\langle N, \Sigma, \emptyset, S \rangle \in CFG$ und $P \mapsto RegE(N \cup \Sigma)$.

Schreibweise $A \mapsto \alpha$, falls $P(A) = \alpha$.

Semantik P repräsentiert die Regelmenge $\tilde{P}$ mit $A \mapsto \tilde{\alpha} \in \tilde{P} \rightleftarrows \tilde{\alpha} \in L(P(A))$.

Beachte $\tilde{P}$ ist im allgemeinen unendlich. $L(G) := L(\langle N, \Sigma, \tilde{P}, S\rangle)$.

Satz

$CFL(\Sigma) = \mathfrak{L}(\Sigma, ECFG)$.

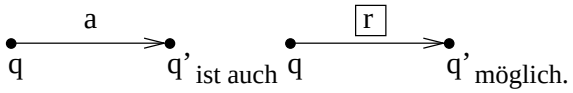
Beweisskizze

"$\subseteq$" nach Definition: $A \mapsto \alpha_1 \vee \alpha_2 \vee \alpha_3$, falls $A \mapsto \alpha_1, A \mapsto \alpha_2, A \mapsto \alpha_3$.

"$\supseteq$": Transformation von ECFG nach CFG.
Idee: Elimination regulärer Ausdrücke.
$A \mapsto \alpha^*$ ersetzen durch $A \mapsto \alpha A \mid \varepsilon$
$A \mapsto \alpha \vee \beta$ ersetzen durch $A \mapsto \alpha \mid \beta$
$A \mapsto \alpha \cdot \beta$ ersetzen durch $A \mapsto BC$ mit neuen $BC, B \mapsto \alpha$ und $C \mapsto \beta$ □

3.9 Rekursive endliche Automaten (Syntaxdiagramme)

Syntaxdiagramme entsprechen endlichen Automaten, die sich rekursiv aufrufen können.



Neben ist auch möglich.

Definition

$\mathfrak{A} = \langle Q, \Sigma, \delta, q_0, F\rangle$ heißt *rekursiver endlicher Automat* über Σ, falls $\delta : Q \times (\Sigma_\varepsilon) \cup Q) \mapsto P(Q)$ und sonst wie ein endlicher Automat.
Bezeichnung: $\mathfrak{A} \in RFA(\Sigma)$.

Semantik Konfigurationen: $Q \times \Sigma^*$
Transitionsrelation: $\vdash \subseteq (Q \times \Sigma^*)^2$ definiert durch

$$\frac{\delta(q,a) \ni P}{(q,aw) \vdash (p,w)} \qquad \frac{\delta(q,\varepsilon) \ni P}{(q,w) \vdash (p,w)}$$

$$\frac{\delta(q,r) \ni P, r \in Q \qquad (r,w) \vdash^* (s,v), s \in F}{(q,w) \vdash (p,v)}$$

$L(\mathfrak{A}) = \{w \in \Sigma^* \mid (q_0, w) \vdash^* (p, \varepsilon), p \in F\}$

Satz

$\mathfrak{L}(\Sigma, RFA) = \mathfrak{L}(\Sigma, PDA)$

Beweis

"$\subseteq$" $\mathfrak{A} = \langle Q, \Sigma, \delta, q_0, F\rangle \in RFA(\Sigma)$
Simulation von $\mathfrak{A}$durch $\mathfrak{A}' = \rangle Q, \Sigma, Q, \delta', q_0, q_0\rangle \in PDA$.

Idee: Keller für "Rücksprungadressen" (Zustände). $\delta'(q,a,s) = \{(p,s) \mid p \in \delta(q,a)\}$
$\delta'(q,\varepsilon,s) = \{(p,s) \mid p \in \delta(q,\varepsilon)\} \cup \{(r,ps) \mid p \in \delta(q,r)\} \cup \{(s,\varepsilon) \mid q \in F\}$.

"$\supseteq$" Für $L \in \mathfrak{L}(\Sigma, PDA)$ existiert $G = \langle N, \Sigma, P, S\rangle \in CNF$ mit $L = L(G)$. Konstruiere $\mathfrak{A} = \langle Q, \Sigma, \delta, q_0, F\rangle \in RFA(\Sigma)$ durch $Q := N \cup \{q_f\}$, $q_0 := S$, $F := \{q_f\}$ und

$$\delta: \begin{array}{ll} \delta(A,B) \ni C & \text{falls } A \mapsto BC \\ \delta(A,a) \ni q_f & \text{falls } A \mapsto A \\ \delta(S,\varepsilon) \ni q_f & \text{falls } S \mapsto \varepsilon \end{array}$$

□

Kapitel 4

Turingmaschinen und aufzählbare Sprachen

4.1 Chomsky-Grammatiken

Verallgemeinerung kontex-freier Grammatiken. Es handel sich um eine kontextabhängige Ersetzung.

Definition

Seien N, Σ, χ und S wie bei CFG definiert. Sei ferner $P \subseteq \chi^* N \chi^* \times \chi^*$, P endlich. Dann heißt $G = \langle N, \Sigma, P, S \rangle$ eine *Chomsky-Grammatik.*

Semantik $\pi : \alpha_1 \mapsto \alpha_2 \in P$ bestimmt die Ableitungsrelation $\Rightarrow_\pi \subseteq \chi^* \times \chi^*$ durch $\beta_1 \Rightarrow_\pi \beta_2 \rightleftarrows \exists \gamma\delta \in \chi^*$ mit $\beta_1 = \gamma\alpha_1\delta$ und $\beta_2 = \gamma\alpha_2\delta$.
Ableitungsrelation von G: $\Rightarrow_G := \bigcup_{\pi \in P} \Rightarrow_\pi$
G erzeugt $L(G) = \{w \in \Sigma^* \mid S \Rightarrow_G^* w\}$.

4.1.1 Klassifikation von Chomsky-Grammatiken und Sprachfamilien

Sei $G = \langle N, \Sigma, P, S \rangle$ eine Chomsky-Grammatik und $i \in \{0, 1, 2, 3\}$. Dann heißt G *vom Typ i* oder eine *Typ i Grammatik*, wenn P die Eigenschaft (i) besitzt mit

(0) keine Einschränkung
(1) Jedes $\pi \in P$ hat die Form $\gamma A \delta \mapsto \gamma B \delta$
(2) Jedes $\pi \in P$ hat die Form $A \mapsto x$.
(3) Entweder: Jedes $\pi \in P$ hat die Form $A \mapsto Bw$ oder $A \mapsto w$ oder $A \mapsto wB$ oder $a \mapsto W$.

Es folgt: Grammatiken vom Typ 0 sind ganu die Chomsky-Grammatiken.
Grammatiken vom Typ 1 heißen auch Kontextsensitiv.
Grammatiken vom Typ 2 sind genau die kontextfreien Grammatiken.
Grammatiken vom Typ 3 sind genau die einseitig linearen Grammtiken.

$L \subseteq \Sigma^*$ heißt *vom Typ i* $\rightleftarrows \exists G$ vom Typ i mit $L(G) = L$.

Beispiel: $L_i(\Sigma) := \{L \subseteq \Sigma^* \mid L vom Typ i\}$

4.1.2 Chomsky-Hierarchie

$L_3(\Sigma) \subsetneqq L_2(\Sigma) \subsetneqq L_1(\Sigma) \subsetneqq L_1(\Sigma)$ falls $|\Sigma| > 1$. $|\Sigma| = 1 \curvearrowright L_3(\Sigma) = L_2(\Sigma)$.

Bereits gezeigt: $L_3(\Sigma) = RegL(\Sigma) \subsetneqq CFL(\Sigma) = L_2(\Sigma)$.
Außerdem: $L_3(\Sigma) \subseteq L_1(\Sigma) \subseteq L_0(\Sigma)$. ε-freie CFG!

Definition

Sei $G = \langle N, \Sigma, P, S \rangle$ vom Typ 0. G *normiert*, wenn gilt:

- Für jedes $\pi = \alpha_1 \mapsto \alpha_2 \in P$ gibt es $\gamma, \delta \in N^*, A \in N, \beta \in \chi^*$ mit $\alpha_1 = \gamma A \delta$ und $\alpha_2 = \gamma\beta\delta$
- Σ-Symbole nur in Regeln der Form $A \mapsto a$.

Satz

Zu jedem $G = \langle N, \Sigma, P, S \rangle$ vom Typ 0 lässt sich eine äquivalente normierte Grammatik $G' = \langle N', \Sigma, P', S' \rangle$ konstruieren.

Beweis

1. Terminalsymbbol-Bedingung:
   $a \in \Sigma \mapsto A_a$ ist neues N-Symbol. a durch A_a ersetzen. $A_a \mapsto a$ hinzufügen.

2. Symbolersetzungsregeln:
   Simulation von $A_1 \dots A_n \mapsto B_1 \dots B_m (n \geq 1, m \geq 0)$ durch Symbolersetzungsregeln: Wähle neue N-Symbole A'. $A_1 \dots A_n \mapsto A'_1 A'_2 \dots A_n$
   $A'_1 \dots A_n \mapsto A'_1 A'_2 \dots A_n$
   $\vdots$
   $A'_1 \dots A'_{n-1} A_n \mapsto A'_1 \dots A'_n$
   $A'_1 \dots A'_n \mapsto B_1 A'_2 A'_n$

   Fall 1 $n \leq m$: $B_1 \dots B_{n-1} A'_n \mapsto B_1 \dots B_m$
   Fall 2 $n > m$: $B_1 \dots B_m A'_{m+1} \dots A'_1 \mapsto B_1 \dots B_m A'_{m+2} \dots A'_n$
   $\vdots$
   $B_1 \dots B_m A'_n \mapsto B_1 \dots B_m$

□

Satz

$\mathfrak{L}_0(\Sigma)$ ist unter Substitution abgeschlossen, d.h. wenn $\varphi : P(\Sigma^*) \mapsto P((\bigcup_{a \in \Sigma} \Sigma_a)^*)$ eine Substutitionsabbildung ist, so gilt:
$L \in \mathfrak{L}_0(\Sigma), \varphi(a) \in \mathfrak{L}_0(\Sigma_a)$ für alle $a \in \Sigma$. $\curvearrowright \varphi(L) \in \mathfrak{L}_0(\bigcup_{a \in \Sigma} \Sigma_a)$.

Beweis

Konstruktion von CFG ist nicht anwendbar, denn neue Ableitungen wären möglich wegen unzulässiger Satzform.

Idee: Sequentialisierung der Ableitungen mit Kontrollsymbol.

$L = L(G) mit G = \langle N, \Sigma, P, S\rangle$ vom Typ 0. $\varphi(a)0 : L(G_a)$ mit $G = \langle , N_a, \Sigma_a, P_a, S_a\rangle$ vom Typ 0.

O.B.d.A. seien G und G_a normiert, sowie N, N_a disjunkt.

Konstruktion von $\overline{G}$ mit $L(\overline{G}) = \varphi(L)$.

$\overline{N} := N \cup (\bigcup_{a\in\Sigma} N_a) \cup \{A_a \mid a \in \Sigma\} \cup \{\overline{S}, C\}$
$\overline{\Sigma} := \bigcup_{a\in\Sigma} \Sigma_a$
$\overline{P} := P_0 \cup P_1 \cup P_2 \cup (\bigcup_{a\in\Sigma} P_a) \cup \{\overline{S} \mapsto CS, C \mapsto \varepsilon\}$
P_0 entstehe aus P durch Ersetzen von a durch A_a $(a \in \Sigma)$.
$P_1 := \{CA_a \mapsto CS_a \mid a \in \Sigma\}$
$P_2 := \{Ca \mapsto ac \mid a \in \overline{\Sigma}\}$ □

Korollar $\mathfrak{L}_0(\Sigma)$ ist unter regulären Operationen abgeschlossen.
$L, L' \in \mathfrak{L}_0(\Sigma) \rightleftarrows L \cup L', LL', L^* \in \mathfrak{L}(\Sigma)$.

Satz

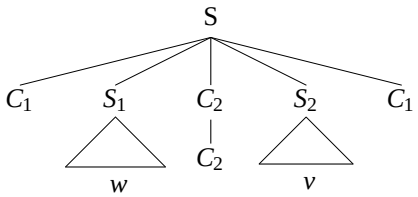
$\mathfrak{L}_0(\Sigma)$ ist unter Durschnitt abgeschlossen.

Beweis

$L_i = L(G_i)$ und $G_i = \langle N_i, \Sigma, P_i, S_i\rangle$ vom Typ 0 für $i \in \{1, 2\}$. O.B.d.A. sei $N_1 \cap N2 = \emptyset$.

Konstruktion von $G = \langle N, \Sigma, P, S\rangle$:
$N := N_1 \cup N_2 \cup \{A_a \mid a \in \Sigma\} \cup \{S, C_1, C_2\}$.

Idee:



$P := P_1 \cup P_2 \cup Q$
$Q := \{S \mapsto C_1S_1C_2S_2C_1, C_2a \mapsto A_aC_2, bA_a \mapsto A_ab, C_1A_aa \mapsto aC_1, C_1C_2C_1 \mapsto \varepsilon \mid a, b \in \Sigma\}$ □

Bemerkung Da $\mathfrak{L}_0(\Sigma) = \mathfrak{L}(\Sigma, TM)$ (siehe unten), ist $\mathfrak{L}_0(\Sigma)$ nicht unter Komplement abgeschlossen.

4.1.3 Abschlusseigenschaften von $\mathfrak{L}_0(\Sigma)$

Satz

$\mathfrak{L}_0(\Sigma)$ ist unter der Substitution abgeschlossen, d.h.: ist

$\varphi : P(\Sigma^*) \mapsto P((\bigcup_{\varphi \in \Sigma} \Sigma_a)')$

eine Substitutionsabbildung, so gilt:

$L \in \mathfrak{L}_0(\Sigma), \varphi(a) \in \mathfrak{L}_0(\Sigma_a) \in Sigma \curvearrowright \varphi() \in \mathfrak{L}_0(\bigcup_{a \in \Sigma} \Sigma_a)$

Beweis

Die Konstruktion von CFG ist nicht anwendbar, denn neue Ableitungen sind nicht möglich wegen unzulässiger Satzformen.
Idee: Sequentialisierung der Ableitungen mit Kontrollsymbol.

$L = L(G)$ mit $G = \langle N, \Sigma, P, S \rangle$ vom Typ 0.
$\varphi(a) =: L_a = L(G_a)$ mit $G_a = \langle N_a, \Sigma_a, P_a, S_a \rangle$ vom Typ 0.
O.B.d.A. sind G, G_a normiert und N, N_a disjunkt.
Konstruktion von $\overline{G}$ mit $L(\overline{G}) = \varphi(L)$:

$\overline{N} := N \uplus (\biguplus_{a \in \Sigma} N_a) \uplus \{A_a \mid a \in \Sigma\} \uplus \{\overline{S}, C\}$

$\overline{\Sigma} := \bigcup_{a \in \Sigma} \Sigma_a$

$\overline{P} := P_0 \cup P_1 \cup P_2 \cup (\bigcup_{a \in \Sigma} P_a) \cup \{\overline{S} \mapsto CS, C \mapsto \varepsilon\}$

P_0 entstehe aus P durch Ersetzen von a durch $A_a (a \in \Sigma)$.
$P_1 := \{CA_a \mapsto CS_a \mid a \in \Sigma\}$
$P_2 := \{C_a \mapsto aC \mid a \in \Sigma\}$ □

Korollar

$\mathfrak{L}_0(\Sigma)$ ist unter regulären Operationen abgeschlossen.

$L, L' \in \mathfrak{L}(\Sigma) \curvearrowright L \cup L', LL', L' \in \mathfrak{L}_0(\Sigma)$

Satz

$\mathfrak{L}_0(\Sigma)$ ist unter Durchschnitt abgeschlossen.

Beweis

$L_i = L(G_i)$ und $G_i = \langle N_i, \Sigma, P_i, S_i \rangle$ für $i = 1, 2$.
O.B.d.A. sei $N_1 \cap N_2 = \emptyset$.
Konstruktion von $G = \langle N, \Sigma, P, S \rangle$:

$N := N_1 \uplus N_2 \uplus \{A_a \mid a \in \Sigma\} \uplus \{S, C_1, C_2\}$

$P := P_1 \cup P_2 \cup Q$

$Q := \{S \mapsto C_1 S_1 C_2 S_2 C_1, C_2 a \mapsto A_a C_2, bA_a \mapsto A_a b, C_1 A_a a \mapsto a C_1, C_1 C_2 C_1 \mapsto \varepsilon \mid a, b \in \Sigma\}$ □

Bemerkung

Da $\mathfrak{L}_0(\Sigma) = \mathfrak{L}(\Sigma, TM)$ (siehe unten), ist $\mathfrak{L}_0(\Sigma)$ nicht unter Komplement abgeschlossen.

4.1.4 Kontextsensitive Grammatiken und Sprachen

Definition

$G = \langle N, \Sigma, P, S \rangle$ heißt von *wachsender Länge* genau dann, wenn für jede Regel $\alpha \mapsto \beta \in P$ gilt: $|\alpha| \leq |\beta|$, es sei denn, dass $S \mapsto \varepsilon \in P$ und S auf keiner rechten Regelseite.

Korollar

Eine Typ 1-Grammatik ist von wachsender Länge.

Satz

Zu jeder Grammatik von wachsender Länge lässt sich eine äquivalente Typ 1-Grammatik konstruieren.

Beweis Normierung ohne Fall 2 im Beweis.

Definition des Platzbedarfs

Sei $G = \langle N, \Sigma, P, S \rangle$ eine Typ 0-Grammatik, $\delta = (S \Rightarrow \alpha_0 \Rightarrow \alpha_1 \ldots \Rightarrow \alpha_n)$ eine Ableitung von G und $w \in L(G)$. Dann heißt

- $\underline{pl}_G(\delta) := max\{\alpha_i \mid 0 \leq i < n)$ der *Platzbedarf von* δ und
- $\underline{pl}_G(w) := min\{\underline{pl}_G(\delta) \mid \delta = (S \Rightarrow \ldots \Rightarrow w)\}$ der *Platzbedarf von w bezüglich G*.

Folgerung

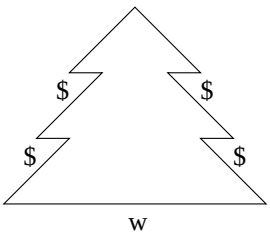
1. $\underline{pl}_g(w) \geq |w|$
2. G von wachsender Länge, $w \neq \varepsilon \curvearrowright \underline{pl}_G(w) = |w|$

4.1.5 Platzbedarfssatz

Sei G vom Typ 0 und $p \in w$, $p > 0$, so dass für alle $w \in L(G) \backslash \{\varepsilon\} : \underline{pl}_G(w) \leq p|w|$.
dann ist $L(G) \in \mathfrak{L}_1$.

Beweis

Sei $p = 1$ also für $w \in L(G) \backslash \{\varepsilon\}$ $\underline{pl}_G(w) = |w|$.



Idee:

1. $\alpha \mapsto \beta$ mit $|\alpha| = |\beta| + i(i > 0) \curvearrowright \alpha \mapsto \$^i\beta$
2. $\alpha \mapsto \beta \in Pmit|\alpha| \leq |\beta| \curvearrowright \$^+\alpha \mapsto \beta \in P'$ für alle $j = 0,1,\ldots,|\beta| - |\alpha|$

□

4.1.6 Abschlusseigenschaften von $\mathfrak{L}_1(\Sigma)$

Satz

$\mathfrak{L}_1(\sigma)$ ist unter ε-freier Substitution ($\varepsilon \neq \varphi(a)$) abgeschlossen.

Beweis

Platzbedarf der Ausgangsgrammatik mit $p = 1 \curvearrowright$Platzbedarf bei Substutitionsgrammatik $|w| + 1 < 2|w|$ wegen zusätzlicher Kontrolle C. □

Korollar

$\mathfrak{L}_1(\Sigma)$ ist unter regulären Operationen abgeschlossen.
$\mathfrak{L}_1(\Sigma)$ ist unter Durschschnitt abgeschlossen.

Beweis

$L_1, L_2 \in \mathfrak{L}_1(\Sigma)$ sind mit linearem Platzbedarf ($p = 1$) erzeugbar. Die $\cap$-Konstruktion für Typ 0-Grammatiken hat dann einen Platzbedarf von $2|w| + 3 \leq 5|w|$. □

Korollar

$\mathfrak{L}_2(\Sigma) \subsetneqq \mathfrak{L}_1(\Sigma)$.

Ein lange ungelöstes Problem: Abschluss von $\mathfrak{L}_1$ unter Komplement. Wurde 1987 positiv beantwortet.

4.2 Turingmaschinen, linear beschränkte Automaten

Ziel

$\mathfrak{L}_0(\Sigma) = \mathfrak{L}(\Sigma, TM)$ und $\mathfrak{L}_1(\Sigma) = \mathfrak{L}(\Sigma, LBA)$.

4.2.1 Definition der nicht-deterministisch erkennenden Turingmaschine

$\mathfrak{A} = \langle Q, \Sigma, \Gamma, q_0, \square, F, \delta \rangle \in TM$, wenn Q Zustandsmenge, Σ Eingabealphabet, Γ Arbeitsalphabet mit $\Sigma \subseteq \Gamma$, $q_0 \in Q$ Anfangszustand, $\square \in F \setminus \Sigma$ Blank, $F \subseteq Q$ Endzustandsmenge und $\delta : Q \times \Gamma \mapsto \wp(Q \times \Gamma \times \{L, N, R\})$ Transitionsfuntion.
$\mathfrak{A}$ heißt *(nicht-deterministische) erkennende Turingmaschine*

Schreibweise

Falls $\delta(q,a) \ni (q',b,L)$, so schreibt man $qaq'bL$ (Quintupel).

Semantik

Konfigurationsmenge: $Q \times \Gamma^* \times \Gamma \times \Gamma^* =: \underline{Conf}(\mathfrak{A})$

Anfangskonfiguration von $w \in \Sigma^*$:
$\kappa_w = \begin{cases} (q_0,\varepsilon,a,w') & ,w = aw' \\ (q_0,\varepsilon,\square,\varepsilon) & ,w = \varepsilon \end{cases}$
Beispiel für Folgekonfigurationen: $\delta(q_0,a) \ni (q_1,b,L)$
Endkonfiguration: (q,α,X,β), falls $q \in F$
Die von $\mathfrak{A}$erkannte Sprache:

$$L(\mathfrak{A}) = \{w \in \Sigma^* \mid \kappa_w \vdash^* (q,\alpha,X,\beta), q \in F\}$$

Satz

$\mathfrak{L}(\Sigma, TM) = \mathfrak{L}_0(\Sigma)$.

Linear beschränkte TM: TM mit Platzbedarf $p \cdot |w|$. LBA: TM mit Platzbedarf $|w|$.

Satz

$\mathfrak{L}_1\Sigma = \mathfrak{L}(\Sigma, LBA)$.

4.2.2 Definition der deterministischen Turingmaschine

$\mathfrak{A} \in TM(\Sigma)$ ist *deterministisch* genau dann, wenn $\delta : Q \times \Gamma \mapsto Q \times \Gamma \times \{L,N,R\}$.

Satz

Zu jedem $\mathfrak{A} \in TM(\Sigma)$ existiert ein äquivalentes $\mathfrak{A}' \in det.TM(\Sigma) : L(\mathfrak{A}) = L(\mathfrak{A}')$.

4.3 Aufzählbare und entscheidbare Sprachen

Intuitiv

$L \subseteq \Sigma^*$ ist aufzählbar, wenn es ein effektives (algorithmisches) Verfahren gibt, welcher genau die Elemente von L erzeugt (rekursiv aufzählbar).

Präzisierung durch TM mit Ausgabe

$\mathfrak{A} = \langle Q,\Sigma,\Gamma,q_0,\square,F,\delta\rangle \in DTM(\Sigma)$
$\delta : Q \times \Gamma \dashrightarrow Q \times \Gamma \times \{L,N,R\}$.
Anmerkung: $\dashrightarrow$steht in etwa für "Nicht jedes $x \in Q \times \Gamma$ muss einen Folgezustand haben".

Schreibweise für Konfigurationen:
statt (q,α,X,β) (gut für Platzbedarf) einfach $\alpha qX\beta$
$\mathfrak{A}$berechnet $f_{\mathfrak{A}} : \Sigma^* \dashrightarrow \Sigma^*$ mit $g_a(w) = v :\rightleftarrows q_0w\square \vdash^* \alpha qv\square\beta \nvdash$
wobei $\kappa \nvdash$ bedeutet: κ hat keine Folgekonfiguration.

Definition

$L \subseteq \Sigma^*$ heißt aufzählbar, wenn $\mathfrak{A} \in DTM(\Sigma)$ existiert mit $Def(f_{\mathfrak{A}}) = \Sigma^*$ und $Bild(f_{\mathfrak{A}}) = L$ oder wenn $L = \emptyset$.

Satz

Für $L \subseteq \Sigma^*$ gilt: L aufzählbar $\rightleftarrows L \in \mathfrak{L}(\Sigma, TM)$.

Beweisskizze

Sei zuerst $L \neq \emptyset$.

"$\curvearrowright$" $\mathfrak{A} \in DTM(\Sigma)$ mit $f_{\mathfrak{A}}$ total (Eingabealphabet = Definitionsbereich) und $L = Bild(f_{\mathfrak{A}})$.
Konstruktion von $\mathfrak{A}' \in DTM(\Sigma)$ mit $L(\mathfrak{A}') = L$.
Idee: Eingabe $W \in \Sigma^*$ speichern, L aufzählen und edes aufgezählte Wort mit w vergleichen, bei Übereinstimmung Endzustand.

"$\curvearrowleft$" $\mathfrak{A} \in DTM(\Sigma)$ mit $L(\mathfrak{A}) = L$: Konstruktion von $\mathfrak{A}'' \in DTM(\Sigma)$ mit $Bild(f_{\mathfrak{A}''}) = L$ und $Def(f_{\mathfrak{A}''} = \Sigma^*$.
Idee:

1. $\mathfrak{A}$lso modifizieren, dass Eingabe bei Erkennung ausgegeben wird.
   Dann: $Def(f_{\mathfrak{A}'}) = Bild(f_{\mathfrak{A}'}) = L$.

2. Damit $\mathfrak{A}'$ bei *jeder* Eingabe ein Wort aus L ausgibt, verwendet man folgenden Trick:
   Teil der Eingabe als Zähler zur Beschränkung der Rechenlänge $\Sigma = \{a_1, \ldots, a_r\}$ mit $r \geq 2$.
   Eingabe hat immer die Form $a_1^n a_i w$ bzw. a_1^n mit $i \neq 1$ und $n \in \mathbb{N}$.
   $\mathfrak{A}''$ simuliert dann $\mathfrak{A}'$ mit Eingabe von w bzw ε und höchstens n Schritten. Eine Ausgabe von $\mathfrak{A}'$ ist auch eine Ausgabe von $\mathfrak{A}''$.
   Wurde nach n Schritten keine Ausgabe berechnet, so wird irgendein $v \in L$ als Ausgabe berechnet:
   $\mathfrak{A}'$ mit Längenbeschränkung und Eingabe von $\varepsilon, a_1, a_2, \ldots$ bis zur ersen Ausgabe laufen lassen.
   Es folgt: $f_{\mathfrak{A}''}$ total mit $Bild(f_{\mathfrak{A}}) = L$i

Für $L = \emptyset$ gilt nach Definition: L aufzählbar. $\emptyset$ ist auch durch eine TM erkennbar: $q_1 aaNq_0(\{q_1\} = F), q_0 bbNq_0$.

Satz

$\mathfrak{L}_0(\Sigma) \subsetneqq \wp(\Sigma^*)$

Beweis

Die Menge der Typ 0-Grammatiken über Σ lässt sich nach ihrer Größe abzählen. Also: $\mathfrak{L}_0(\Sigma)$abzählbar, d.h. gleichmächtig zu $\mathbb{N}$.

$\wp(\Sigma^*)$ ist jedoch überabzählbar.

4.3.1 Diagonalverfahren nach Cantor

$|\Sigma| = 1, \Sigma^* = \mathbb{N}, L \subseteq \Sigma^* \mapsto \underline{char_L} : \mathbb{N} \mapsto \{0,1\}, n \in L \rightleftarrows \underline{char_L}(n) = 1.$

Angenommen, $\wp(\mathbb{N})$ wäre abzählbar. Dann gäbe es $f : \mathbb{N}^2 \mapsto \{0,1\}$

	0	1	2	3	4	$\cdots$
0	0	1	0	1	1	$\cdots$
0	1	0	1	0	0	$\cdots$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\cdots$

Definiere $L_{dia} \subseteq \mathbb{N}$ durch $\underline{char_{L_{dia}}}(j) := 1 - f(j,j)$ (flippe Diagonalelemente). Dann kann L_{dia} nicht in der Abzählung vorkommen.

4.3.2 Entscheidbare Sprachen

Intuitiv

$L \subseteq \Sigma^*$ entscheidbar, wenn es ein effektives Verfahren gibt, welches für jedes Wort $w \in \Sigma^*$ feststellt, ob $w \in L$ oder ob $w \notin L$ gilt.

Präzisierung durch TM mit Ausgabe

$L \in \Sigma^*$ heißt entscheidbar $\rightleftarrows \exists \mathfrak{A} \in DTM(\Sigma)$ mit $Def(f_{\mathfrak{A}}) = \Sigma^*$ und $L = f_{\mathfrak{A}}^{-1}(\varepsilon)$.

Satz

Für $L \in \Sigma^*$ gilt: L entscheidbar $\rightleftarrows$L und $\Sigma^* \setminus L$ aufzählbar.

Beweis

"$\curvearrowright$" L entscheidbar $\mathfrak{A} \in DTM(\Sigma)$ mit $f_{\mathfrak{A}}$ total und $L = f_{\mathfrak{A}}^{-1}(\Sigma)$. Konstruiere $\mathfrak{A}'$ mit $L(\mathfrak{A}') = L$ und $\mathfrak{A}''$ mit $L(\mathfrak{A}'') = \Sigma^* \setminus L$ durch passende Wahl der erkennenden Zustände.

"$\curvearrowleft$" Reißverschlussverfahren zweier Aufzählungsmaschinen.

Kapitel 5

Unentscheidbare Probleme

Bekannt:

- Entscheidbar (ATFS):
  - Wortproblem für Typ 1, 2 und 3 Grammatiken ("$w \in L(G)$?")
- Unentscheidbar (BuK):
  - Halteproblem für TM ("Hält $\mathfrak{A}$auf Eingabe w?")
  - Postsches Korrespondenzproblem (PCP)

Ziel: Unentscheidbarkeit

- Unentscheidbarkeit des Schnittproblems für (D)PDA
- Unentscheidbarkeit des Äquivalenzproblems für CFG

5.1 Satz: Schnittproblem für CFG

Es ist nicht entscheidbar, ob für $G_1, G_2 \in CFG$ gilt:

$L(G_1) \cap L(G_2) = \emptyset$

Beweis

Beweisführung durch Reduktion des PCP auf das Schnittproblem.

Seien $v = (v_1, \ldots, v_n), w = (w_1, \ldots, w_n) \in (\Sigma^*)^n$. Wir konstruieren $G_v, G_w \in CFG$ so, dass $PCP(v,w)$ lösbar $\rightleftarrows L(G_1) \cap L(G_2) = \emptyset$. Sei dazu $\tilde{\Sigma} = \Sigma \cup \{a_1, \ldots, a_n\}$ und $G_v = \langle \{S_v\}, \tilde{\Sigma}, P_v, S_v \rangle$ mit den Produktionen $P_v := \bigcup_{i=1}^{n} \{S_v \mapsto a_i S_v v_i \mid a_i v_i\}$ (G_w analog).

Es folgt:

$L_v := L(G_v) = \{a_{i_1} \ldots a_{i_k} v_{i_k} \ldots v_{i_1} \mid i \geq 1, 1 \leq h \leq n\}$ (L_w analog).

Somit gilt:

$$\begin{aligned} L_v \cap L_w \neq \emptyset &\rightleftarrows \text{ es ex. } k \geq 1, i_1, \ldots, i_k \in \{1, \ldots, n\} \text{ mit } a_{i_1} \ldots a_{i_k} v_{i_k} \ldots v_{i_1} = a_{i_1} \ldots a_{i_k} w_{i_k} \ldots w_{i_1} \\ &\rightleftarrows \text{ es ex. } k \geq 1, i_1, \ldots, i_k \in \{1, \ldots, n\} \text{ mit } v_{i_k} \ldots v_{i_1} = w_{i_k} \ldots w_{i_1} \\ &\rightleftarrows PCP(v,w) \text{ lösbar} \end{aligned}$$

Die Entscheidbarkeit des Schnittproblems würde aber die Entscheidbarkeit des PCP implizieren. □

Korollar

Das Schnittproblem für DPDA ist nicht entscheidbar.

Beweis

L_v und L_w sind offensichtlich deterministisch kontextfrei. □

5.1.1 Satz: Äquivalenzproblem für CFG

Es ist nicht entscheidbar, ob für $G_1, G_2 \in CFG$ gilt: $L(G_1) = L(G_2)$.

Beweis

Beweis durch Reduktion des Schnittproblems auf das Äquivalenzproblem für CFG.

1. Zu $\mathfrak{A} \in DPDA(\Sigma)$ lässt sich $\overline{\mathfrak{A}} \in DPDA(\Sigma)$ mit $L(\overline{\mathfrak{A}}) = \overline{L(\mathfrak{A})}$ konstruieren.
2. Zu einem $\mathfrak{A} \in PDA(\Sigma)$ kässt sich $G_{\mathfrak{A}}$ konstruiren mit $L(G_{\mathfrak{A}}) = L(\mathfrak{A})$.
3. Zu $G_1, G_2 \in CFG(\Sigma)$ lässt sich $G_\cup \in CFG(\Sigma)$ konstruiren mit $L(G_\cup) = L(G_1) \cup L(G_2)$. Damit folgt für $\mathfrak{A}_1, \mathfrak{A}_2 \in DPDA$:

$$
\begin{array}{lll}
L(\mathfrak{A}_1) \cap L(\mathfrak{A}_2) = \emptyset & \rightleftarrows & L(\mathfrak{A}_1) \subseteq \overline{L(\mathfrak{A}_2)} \\
& \rightleftarrows(1) & L(\mathfrak{A}_1) \subseteq L(\overline{\mathfrak{A}_2}) \\
& \rightleftarrows & L(\mathfrak{A}_1) \cup L(\mathfrak{A}_2) = L(\overline{\mathfrak{A}_2}) \\
& \rightleftarrows(2) & L(G_{\mathfrak{A}_1}) \cup L(G_{\overline{\mathfrak{A}_2}}) = L(G_{\overline{\mathfrak{A}_2}} \\
& \rightleftarrows(3) & L(G_\cup) = L(G_{\overline{\mathfrak{A}_2}})
\end{array}
$$

Die Entscheidbarkeit des Äquivalenzproblems würde somit die Entscheidbarkeit des Schnittproblems implizieren. □

Anhang A

GNU Free Documentation License

Version 1.2, November 2002

Copyright © 2000,2001,2002 Free Software Foundation, Inc.
59 Temple Place, Suite 330, Boston, MA 02111-1307 USA
Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

PREAMBLE

The purpose of this License is to make a manual, textbook, or other functional and useful document “free” in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondarily, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of “copyleft”, which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work, in any medium, that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. Such a notice grants a world-wide, royalty-free license, unlimited in duration, to use that work under the conditions stated herein. The “Document”, below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as “you”. You accept the license if you copy, modify or distribute the work in a way requiring permission under copyright law.

A “Modified Version” of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A “Secondary Section” is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document’s overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (Thus, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The “Invariant Sections” are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License. If a section does not fit the above definition of Secondary then it is not allowed to be designated as Invariant. The Document may contain zero Invariant Sections. If the Document does not identify any Invariant Sections then there are none.

The “Cover Texts” are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License. A Front-Cover Text may be at most 5 words, and a Back-Cover Text may be at most 25 words.

A “Transparent” copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, that is suitable for revising the document straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup, or absence of markup, has been arranged to thwart or discourage subsequent modification by readers is not Transparent. An image format is not Transparent if used for any substantial amount of text. A copy that is not “Transparent” is called “Opaque”.

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML, PostScript or PDF designed for human modification. Examples of transparent image formats include PNG, XCF and JPG. Opaque formats include proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML, PostScript or PDF produced by some word processors for output purposes only.

The “Title Page” means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, “Title Page” means the text near the most prominent appearance of the work’s title, preceding the beginning of the body of the text.

A section “Entitled XYZ” means a named subunit of the Document whose title either is precisely XYZ or contains XYZ in parentheses following text that translates XYZ in another language. (Here XYZ stands for a specific section name mentioned below, such as “Acknowledgements”, “Dedications”, “Endorsements”, or “History”.) To “Preserve the Title” of such a section when you modify the Document means that it remains a section “Entitled XYZ” according to this definition.

The Document may include Warranty Disclaimers next to the notice which states that this License applies to the Document. These Warranty Disclaimers are considered to be included by reference in this License, but only as regards disclaiming warranties: any other implication that these Warranty Disclaimers may have is void and has no effect on the meaning of this License.

VERBATIM COPYING

You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section A.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

COPYING IN QUANTITY

If you publish printed copies (or copies in media that commonly have printed covers) of the Document, numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a computer-network location from which the general network-using public has access to download using public-standard network protocols a complete Transparent copy of the Document, free of added material. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

MODIFICATIONS

You may copy and distribute a Modified Version of the Document under the conditions of sections A and A above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

1. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.
2. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has fewer than five), unless they release you from this requirement.
3. State on the Title page the name of the publisher of the Modified Version, as the publisher.
4. Preserve all the copyright notices of the Document.
5. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.
6. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.
7. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.
8. Include an unaltered copy of this License.
9. Preserve the section Entitled "History", Preserve its Title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section Entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.
10. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.
11. For any section Entitled "Acknowledgements" or "Dedications", Preserve the Title of the section, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.
12. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.
13. Delete any section Entitled "Endorsements". Such a section may not be included in the Modified Version.
14. Do not retitle any existing section to be Entitled "Endorsements" or to conflict in title with any Invariant Section.
15. Preserve any Warranty Disclaimers.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version's license notice. These titles must be distinct from any other section titles.

You may add a section Entitled "Endorsements", provided it contains nothing but endorsements of your Modified Version by various parties–for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

COMBINING DOCUMENTS

You may combine the Document with other documents released under this License, under the terms defined in section A above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice, and that you preserve all their Warranty Disclaimers.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections Entitled "History" in the various original documents, forming one section Entitled "History"; likewise combine any sections Entitled "Acknowledgements", and any sections Entitled "Dedications". You must delete all sections Entitled "Endorsements".

COLLECTIONS OF DOCUMENTS

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

AGGREGATION WITH INDEPENDENT WORKS

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, is called an “aggregate” if the copyright resulting from the compilation is not used to limit the legal rights of the compilation’s users beyond what the individual works permit. When the Document is included in an aggregate, this License does not apply to the other works in the aggregate which are not themselves derivative works of the Document.

If the Cover Text requirement of section A is applicable to these copies of the Document, then if the Document is less than one half of the entire aggregate, the Document’s Cover Texts may be placed on covers that bracket the Document within the aggregate, or the electronic equivalent of covers if the Document is in electronic form. Otherwise they must appear on printed covers that bracket the whole aggregate.

TRANSLATION

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section A. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License, and all the license notices in the Document, and any Warranty Disclaimers, provided that you also include the original English version of this License and the original versions of those notices and disclaimers. In case of a disagreement between the translation and the original version of this License or a notice or disclaimer, the original version will prevail.

If a section in the Document is Entitled “Acknowledgements”, “Dedications”, or “History”, the requirement (section A) to Preserve its Title (section A) will typically require changing the actual title.

TERMINATION

You may not copy, modify, sublicense, or distribute the Document except as expressly provided for under this License. Any other attempt to copy, modify, sublicense or distribute the Document is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

FUTURE REVISIONS OF THIS LICENSE

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See http://www.gnu.org/copyleft/.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License “or any later version” applies to it, you have the option of following

the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this

License, you may choose any version ever published (not as a draft) by the Free Software Foundation.

Index