1 Grundbegriffe

- Ein Alphabet ist eine endliche, nichtleere Menge Σ von Buchstaben (oder Symbolen).
- Ein Wort über Σ ist eine endliche Folge von Elementen aus $\sum$.
- Die Länge eines Wortes w (bezeichnet mit $|w|$) ist die Anzahl der Symbole in w.
- Das leere Wort ist das eindeutig bestimmte Wort der Länge 0 und wird mit dem griechischen Buchstaben λ bezeichnet.
- Die Menge aller Wörter über Σ bezeichnen wir mit Σ^*.
- Eine formale Sprache über Σ ist eine jede Teilmenge von Σ^*
- Die leere Sprache ist die Sprache die keine Wörter enthält, und wird mit $\emptyset$ bezeichnet.
- die Kardinalität einer Sprache L ist die Anzahl der Wörter von L und wird mit $\|L\|$ bezeichnet.

2 Operationen

- Vereinigung $\{1,2\} \cup \{2,3\} = \{1,2,3\}$.
- Durchschnitt $\{1,2\} \cap \{2,3\} = \{2\}$.
- Differenz $A - B = \{x \in A \ und \ x \notin B\}$.
- Komplement $\overline{A} = \{x \in \Sigma^* | x \notin B\}$.
- Konkatenation von Wörtern
  - Ist $u = v = \lambda$, so ist $uv = vu = \lambda$.
  - Ist $v = \Lambda$, so ist $uv = u$.
  - Ist $u = \lambda$ so ist $uv = v$.
  - Ist $u = u_1 u_2 \dots u_n undv = v_1 v_2 \dots v_m$ mit $u_i, v_i \in \Sigma$, so ist

$$uv = u_1 u_2 \dots u_n v_1 v_2 \dots \varepsilon_m.$$

- Konkatenation von Sprachen: $AB = \{ab | a \in A \ und \ b \in B\}$.
- Iteration einer Sprache: $A^0 = \{\lambda\}, \ A^n = AA^{n-1}, \ A^* = \bigcup_{n \geq 0} A^n$.
- Spiegelbildoperation von Wort $sp(u) = u_n \dot{u}_2 u_1$.
- Spiegelbildoperation von Sprache $sp(A) = \{sp(w) | w \in A\}$.
- Teilwortrelation auf Σ^*: $u \sqsupseteq v \leftrightarrow (\exists v_1, v_2 \in \Sigma^*)[v_1 u v_2 = v]$.
- Anfangswortrelation auf Σ^*: $u \sqsupseteq_a v \leftrightarrow (\exists w \in \Sigma^*)[uw = v]$.

3 Symbole

- Σ ein Alphabet von Terminalsymbolen
- N eine Endliche Menge von Nichtterminalen, $\Sigma \cap N = \emptyset$
- S Startsymbol, $S \in N$
- P Produktionsregeln, $P \subseteq (N \cup \Sigma)^+ \times (N \cup \Sigma)^*$

4 Grammatik

$G = (\Sigma, N, S, P)$

- Typ-0: Ohne Einschränkungen.
- Typ-1: $\forall p \to q \in P : |p| \leq |q|$
- Typ-2: $\forall p \to q \in P : p \in N$
- Typ-3: $\forall p \to q \in P : |p| \in N \; und \; q \in \Sigma \cup \Sigma N$

$REG \subseteq CF \subseteq CS \subseteq \mathcal{L}_0$

4.1 Sonderregelung für λ

Typ-i Grammatiken mit $i \in \{1, 2, 3\}$ sind nichtverkürzend, daher $\lambda \notin L(G)$. Daher folgende Sonderregelung:

1. Die Regel $S \to \lambda$ ist als einzige verkürzende Regel für Grammatiken vom Typ 1, 2, 3 zugelassen.
2. Tritt die Regel $S \to \lambda$ auf, so darf S auf keiner rechten Seite einer Regel vorkommen.

Dies kann für alle Fälle mit folgender Umwandlung erreicht werden:

1. In allen Regeln der Form $S \to u$ aus P mit $u \in (N \cup \Sigma)^*$ wird jedes Vorkommen von S in u durch ein neues Nichtterminal S' ersetzt.
2. Zusätzlich enthält P' alle Regeln aus P, mit S ersetzt durch S'.
3. Die Regel $S \to \lambda$ wird hinzugefügt.

5 Reguläre Sprachen

5.1 DFA

5.1.1 Definition

$M = (\Sigma, Z, \delta, z_o, F)$

- Σ: Alphabet
- Z: endliche Menge von Zuständen mit $\Sigma \cap Z = \emptyset$
- $\delta : Z \times \Sigma \rightarrow Z$ Überführungsfunktion
- $z_0 \in Z$ Startzustand
- $F \subseteq Z$ Endzustände

5.1.2 Beispiel

δ	z_o	z_1	z_2	z_3
0	z_1	z_3	z_2	z_3
1	z_3	z_2	z_2	z_3

5.1.3 $DFA \rightarrow Grammar$

- $N = Z$,
- $S = z_0$,
- P:
  - Gilt $\delta(z, a) = z'$, so ist $z \rightarrow az'$ in P.
  - Ist $z' \in F$, so ist zusätzlich $z \rightarrow a$ in P.
  - ist $\lambda \in A$ (d.h., $z_o \in F$), so ist auch $z_0 \rightarrow \lambda$ in P, und die bisher konstruierte Grammatik wird gemäß der Sonderregel für λ modifiziert.

5.2 NFA

$M = (\Sigma, Z, \delta, S, F)$

- Σ: Alphabet
- Z: endliche Menge von Zuständen mit $\Sigma \cap Z = \emptyset$
- $\delta : Z \times \Sigma \rightarrow \mathcal{P}(Z)$: Überführungsfunktionen zur Potenzmenge von Z
- $S \subseteq Z$: Menge der Startzustände
- $F \subseteq Z$ Menge der Endzustände

5.2.1 $NFA \rightarrow DFA$ (Rabin und Scott)

NFA $M = (\Sigma, Z, \delta, S, E)$ und DFA $M' = (\Sigma, \mathcal{P}(Z), \delta', z_0', F)$

- Zustandsmenge von M': $\mathcal{P}(Z)$,
- $\delta'(Z', a) = \cup_{z \in Z'} \delta(z, a) = \hat{\delta}(Z', a)$ für, all $Z' \subseteq Z$ und $a \in \Sigma$,
- $z_o' = S$,
- $F = \{Z' \subseteq Z | Z' \cap E \not\emptyset\}$

5.3 $Grammatik \rightarrow NFA$

- $Z = N \cup \{X\}$, wobei $X \notin N \cup \Sigma$ ein neues Symbol ist,
- $$F = \begin{cases} \{S, X\} & \text{falls } S \rightarrow \lambda \text{ in } P \\ \{X\} & \text{falls } S \rightarrow \lambda \text{ nicht in } P, \end{cases}$$
- $S' = \{S\}$ und
- für alle $A \in N$ und $a \in \Sigma$ sei
$$\delta(A, a) = \left(\bigcup_{A \rightarrow aB \in P} \{B\} \right) \cup \bigcup_{A \rightarrow a \in P} \{X\}.$$

5.4 Regex

- *$\emptyset$ und λ sind reguläre Ausdrücke*
- *Jedes $a \in \Sigma$ ist ein regulärer Ausdruck.*
- *Sind α und β reguläre Ausdrücke, so sind auch*
  - $\alpha\beta$
  - $(\alpha + \beta)$ *und*
  - $(\alpha)^*$

  reguläreAusdrücke
- *Nichts sonst ist ein regulärer Ausdruck*

5.5 $NFA \to L(M)$ Gleichungssysteme

Bilde ein Gleichungssystem mit n Variablen und n Gleichungen:

1. Jedes $z_i \in Z, 1 \leq i \leq n$ ist Variable auf der linken Seite einer Gleichung
2. Gilt $z_j \in \delta(z_i, a)$ für $z_i, z_j \in Z$ und $a \in \Sigma$, so ist az_j Summand auf der rechten Seite der Gleichung „$z_i = \ldots$"
3. Gilt $z_i \in F$, so ist $\emptyset^*$ Summand auf der rechten Seite der Gleichung „$z_i = \ldots$".

Todo: Die z_i werden als reguläre Sprachen interpretiert und gemäß Lemma 2.24 und Satz 2.226 ausgerechnet. Es gilt dann: $L(M) = \bigcup_{z_i \in S} z_i$ bzw. $L(;) = L(\alpha)$ für den regulären Ausdruck $\alpha = \Sigma_{z_i \in S} z_i$.

5.6 Pumping Lemma REG

Sei $L \in$ REG. Dann existiert eine (von L abhängige) Zahl $n \geq 1$, so dass sich alle Wörter $x \in L$ mit $|x| \geq n$ zerlegen lassen in $x = uvw$ wobei gilt:

1. $|uv| \leq n$,
2. $|v| \geq 1$,
3. $(\forall i \geq 0)[uv^i w \in L]$.

5.7 Mihill Nerode Minimalautomaten

5.7.1 Mihill Nerode Relation

$xR_L y$ zwichen x und y gild genau dann, wenn $(\forall z \in \Sigma^*)[xz \in L \leftrightarrow yz \in L]$. Dies induziert eine Zerlegung von Σ^* in Äquivalenzklassen:

$$[x] = \{y \in \Sigma^* | xR_L y\}$$

Die Anzahl der Äquivalenzklassen ist $\text{Index}(R_L) = \|\{[x] | x \in \Sigma^*\}\|$.

$$L \in REG \leftrightarrow \text{Index}(R_L) < \infty$$

Algorithmus
Eingabe: DFA $M = (\Sigma, Z, \delta, z_0, F)$.
Ausgabe: Ein zu M äquivalenter Minimalautomat
Schritte:

1. Entferne alle von z_0 aus nicht erreichbaren Zustände aus Z.
2. Erstelle eine Tabelle aller (ungeordneten) Zustandspaare $\{z, z'\}$ on M mit $z \neq z'$.
3. Markiere alle Paare $\{z, z'\}$ mit $z \in F \leftrightarrow z' \notin F$.

4. Seit $\{z, z'\}$ ein unmarkiertes paar. Prüfe für jedes $a \in \Sigma$, ob $\{\delta(z,a), \delta(z',a)\}$ bereits markiert ist. Ist mindestens ein Test erfolgreich, so markiere auch $\{z, z'\}$.
5. Wiederhole Schritt 4, bis keine Änderung mehr eintritt.
6. Bilde maximale Mengen paarweise nicht disjunkter unmarkierter Zustandspaare und verschmelze jeweils alle Zustände einer Menge zu einem neuen Zustand.

5.8 Abschlusseigenschaften Definitionen

1. Vereinigung, falls $(\forall A, B \subseteq \Sigma^*)[(A \in \mathcal{C} \wedge B \in \mathcal{C}) \implies A \cup B \in \mathcal{C}]$;
2. Komplement, falls $(\forall A \subseteq \Sigma^*)[A \in \mathcal{C} \implies \overline{A} \in \mathcal{C}]$;
3. Schnitt, falls $(\forall A, B \subseteq \Sigma^*)[(A \in \mathcal{C} \wedge B \in \mathcal{C}) \implies A \cap B \in \mathcal{C}]$;
4. Differenz, falls $(\forall A, B \subseteq \Sigma^*)[(A \in \mathcal{C} \wedge B \in C) \implies A \cap B \in \mathcal{C}]$;
5. Konkatenation, falls $(\forall A, B \subseteq \Sigma^*)[(A \in \mathcal{C} \wedge B \in \mathcal{C}) \implies AB \in \mathcal{C}]$;
6. Iteration (Kleene-Hülle), falls $(\forall A \subseteq \Sigma^*)[A \in \mathcal{C} \implies A^* \in \mathcal{C}]$;
7. Spiegelung, falls $(\forall A \subseteq \Sigma^*)[A \in \mathcal{C} \implies sp(A) \in \mathcal{C}]$;

5.9 Characterisierung

1. Es gibt eine rechtslineare Grammatik G mit $L(G) = L$.
2. Es gibt eine linklineare Grammatik G mit $L(G) = L$.
3. Es gibt einen DFA M mit $L(M) = L$.
4. Es gibt einen NFA M mit $L(M) = L$.
5. Es gibt einen regulären Ausdruck α mit $L(\alpha) = L$.
6. Für die Myhill-Nerode-Relation R_L gilt: $\text{Index}(R_L) < \infty$.

6 Kontextfreie Sprachen

6.1 Normalformen

Ausnahmeregel für das leere Wort muss nur für Typ-2 Grammatiken nicht genutzt werden. Eine $kfgG = (\Sigma, N, S, P)$ heißt λ-frei, falls in P keine Regel $A \to \lambda$ mit $A \neq S$ auftritt. Diese Umwandlung ist immer möglich.

6.1.1 kfG $\rightarrow$ λ-frei

Wenn $\lambda \in L(G)$ wende Sonderregelung für λ an.

1. Bestimme die Menge $N_\lambda = \{A \in N | A \vdash_G^* \lambda\}$ sukzessive wie folgt:
   (a) Ist $A \rightarrow \lambda$ eine Regel in P, so ist $A \in N_\lambda$.
   (b) Ist $A \rightarrow A_1 A_2 \dots A_k$ eine Regel in P mit $k \geq 1$ und $A_i \in N_\lambda$ für alle i, $1 \leq i \leq k$, so ist $A \in N_\lambda$.
2. Füge für jede Regel der Form
   $$B \rightarrow uAv \text{ mit } B \in N, A \in N_\lambda \text{ und } uv \in (N \cup \Sigma)^+$$
   zusätzlich die Regel $B \rightarrow uv$ zu P hinzu.
3. entferne alle Regeln $A \rightarrow \lambda$ aus P.

Schritt 2 muss auch für neu generierte Regeln iterativ angewendet werden. Dies ergibt die gesuchte λ-freie kfG G' mit $L(G) = L(G')$.

6.1.2 Einfache Regeln entfernen

Regeln $A \rightarrow B$ heißen einfach falls $A, B \in N$

1. Entferne alle Zyklen
   $$B_1 \rightarrow B_2, B_2 \rightarrow B_3, \dots, B_{k-1} \rightarrow B_k, B_k \rightarrow B_1 \text{ mit } B_i \in N$$
   und ersetze all B_i (in den verbleibenden Regelnd) durch ein neues Nichtterminal B.
2. Nummeriere die Nichtterminale als $\{A_1, A_2, \dots A_n\}$ so, dass aus $A_i \rightarrow A_j$ folgt: $i < j$.
3. Für $k = n-1, n-2, \dots, 1$ (RÜCKWÄRTS!) eliminiere die Regel $A_k \rightarrow A_l$ mit $k < l$ so: Sind die Regeln mit A_l als linker Seite gegeben durch
   $$A_l \rightarrow u_1 | u_2 | \dots | u_m,$$
   so entferne $A_k \rightarrow A_l$ und füge die folgenden Regeln hinzu:
   $$A_k \rightarrow u_1 | u_2 | \dots | u_m.$$

6.1.3 λ-frei ohne einfache Regeln $\rightarrow$ Chomsky-Normalform

Bedingung: Alle Regeln in P haben die form

- $A \rightarrow BC$ mit $A, B, C \in N$;
- $A \rightarrow a$ mit $A \in N$ und $a \in \Sigma$.

Überführung:

1. Regeln $A \rightarrow a$ mit $A \in N$ und $a \in \Sigma$ sind CNF und werden übernommen. Alle anderen Regeln sind von der Form: $A \rightarrow x$ mit $x \in (N \cup \Sigma)^*$ und $|x| \geq 2$.
2. Füge für jedes $a \in \Sigma$ ein neues Nichtterminal B_a zu N hinzu, ersetze jedes Vorkommen von $a \in \Sigma$ durch B_a und füge zu P die Regel $B_a \rightarrow a$ hinzu.
3. Nicht in CNF Sind nun nur noch Regeln der Form

   $$A \rightarrow B_1 B_2 \dots B_k, \text{ wobei } k \geq 3 \text{ und jedes } B_i \text{ein Nichtterminal ist.}$$

   Jede solche Regel wird ersetzt durch die Regeln:

   $$\begin{aligned} A &\rightarrow B_1 C_2, \\ C_2 &\rightarrow B_2 C_3, \\ &\vdots \\ C_{k-2} &\rightarrow B_{k-2} C_{k-1}, \\ C_{k-1} &\rightarrow B_{k-1} B_k, \end{aligned}$$

   wobei $C_2, C_3, \dots, C_{k-1}$ neue Nichtterminale sind.

Dies liefert die gesuchte Grammatik G' in CNF mit $L(G) = L(G')$

6.1.4 Bemerkung CNF

- Ableitung von w in G in genau $2|w| - 1$ Schritten.
- Syntaxbaum ist ein Binärbaum.

6.1.5 Greibach-Normalform

Zu jeder kfG mit $\lambda \notin L(G) gibt es eine kfG G' in GNF, sodass L(G) = L(G')$ und jede Regel in Folgender Form ist:

$$A \rightarrow aB_1 B_2 \dots B_k \text{ mit } k \geq 0 \text{ und } a \in \Sigma$$

6.1.6 Bemerkung GNF

Man unterscheidet bezüglich der Länge der rechten Seite der Produktion:

- Jede kfG kann in eine äquivalente kfG in Greibach-Normalform transformiert werden, so dass für alle Regeln $A \rightarrow aB_1 B_2 \dots B_k$ stets $k \leq 2$ gilt.
- Für den Spezialfall $k \in \{0, 1\}$ erhalten wir gerade die Definition rechtslinearer Grammatiken.
- Die Ableitung eines Wortes $w \in L(G), w \neq \lambda$ ist genau $|w|$ Schritte lang wenn G in GNF steht.

6.2 Pumping-Lemma CF

Sei L eine kontextfreie Sprache. Dann existiert eine (von L abhängige) Zahl $n \geq 1$, so dass sich alle Wörter $z \in L$ mit $|z| \geq n$ zerlegen lassen in $z = uvwxy$, wobei gilt:

1. $|vx| \geq 1$.
2. $|vwx \leq n$.
3. $(\forall i \geq 0)[uv^i wx^i y \in L]$.

6.3 Satz von Parikh

6.3.1 Parikh Abbildung

$\Psi : LL \to \mathcal{N}^n$ ist definiert durch

$$\Psi(w) = (|w|_a 1, |w|_a 2, \ldots, |w|_a n),$$

wobei $|w|_a$ die Anzahl der Vorkommen des Zeichens a in w angibt. Für Sprachen: $\Psi : \mathcal{P}(\Sigma^*) \to \mathcal{P}(\mathcal{N}^n)$ ist definiert durch

$$\Psi(L) = \{\Psi(w) | w \in L\}$$

Eine Menge ist semilinear wenn sie aus linearen Mengen zusammengesetzt ist.

6.3.2 Parikh

Für jede kontextfreie Sprache L in $\Psi(L)$ ist semilinear

6.3.3 Beispiel

$$L = \{x \in \{0,1\}^i | x = 1^k \text{ mit } k \geq 0 \text{ oder } x = 0^j 1^{k^2} mitj \geq 1 \text{ und } k \geq 1\}$$

über $\Sigma = \{0,,1\}$. Die Menge

$$\Psi(L) = \{(i,j) | (i = 0 \text{ und } j \geq 0) \text{ oder } (i \geq 1 \text{ und } j = k^2 \text{ und } k \geq 1)\}$$

ist nicht semilinear und L somit nicht kontextfrei.

6.4 Abschlusseigenschaften CF

CF ist abgeschlossen unter Vereinigung, Konkatenation, Iteration, Spiegelung.
CF ist abgeschlossen unter Schnitt mit REG.

6.5 CYK Algorithmus

Überprüfe ob Wort in Sprache. Zeile i, Spalte j der Tabelle ist

```
for k in 0..<j:
  t[i, j].add nonterminals_with_rule_to_pair( (i, k), (i+k+1, -k))
```

Wenn S in der letzten Zeile steht is $w \in L(G)$.

6.6 Kellerautomaten (PDA)

6.6.1 Definition

$M = (\Sigma, \Gamma, Z, \delta, z_0, \#)$

- Σ: Eingabe-Alphabet
- Γ: Kelleralphabet
- Z: endliche Menge von Zuständen
- $\delta : Z \times (\Sigma \cup \{\lambda\}) \times \Gamma \rightarrow \mathcal{P}_e(Z \times \Gamma^+)$ die Überführungsfunktion, $\mathcal{P}_e(Z \times \Gamma^+)$ ist die Menge aller endlichen Teilmengen von $Z \times \Gamma^+$
- $z_0 \in Z$: Startzustand
- $\# \in \Gamma$ das Bottom-Symbol im Keller

δ-Übergänge $(z', B_1, B_2 \dots B_k) \in \delta(z, a, A)$ schreiben wir kurz auch $zaA \rightarrow z'B_1B_2 \dots B_k$

- Akzeptiert wenn Keller leer
- Schritte ohne Lesen eines Eingabesymbols sind in der Form $z\lambda A \rightarrow z'B_1B_2 \dots B_k$ möglich.
- Es ist nur ein Startzustand nötig

Sprache:

- $\|_m = Z \times \Sigma^* \times \Gamma^*$ ist die Menge aller Konfigurationen von M
- ist $k = (z, \alpha, \gamma)$ eine Konfiguration aus $\|_M$, so ist im aktuallen Takt der Rechnung von M:
  - $z \in Z$ der aktuelle Zustand von M;
  - $alpha \in \Sigma^*$ der noch zu lesende Teil des Eingabeworts;
  - $\gamma \in \Gamma^*$ der aktuelle Kellerinhalt.

  Für jedes Eingabewort $w \in \Sigma^*$ ist $(z_0, w, \#)$ die entsprechende Startkonfiguration von M.
- Auf $\|_m$ definieren wir eine binäre Relation $\vdash_m \subseteq \|_M \times \|_M$ wie folgt:
  - Für $k, k' \in \|_M$ gilt $k \vdash_m k'$ genau dann, wenn k' aus k durch eine Anwendung von δ hervorgeht.
- $\vdash_M^*$ ist die reflexive und transitive Hülle von $\vdash_M$
- Die vom PDA M akzeptierte Sprache ist definiert durch

  $$L(M) = \{w \in \Sigma^* | (z_0, w, \#) \vdash_M^* (z, \lambda, \lambda) \text{ für ein } z \in Z\}$$

  (z, λ, λ) ist dann eine Endkonfiguration für PDA M.

6.6.2 Kontextfrei → PDA

Von $G = (\Sigma, N, S, P)$ zu $M = (\Sigma, N \cup \Sigma, \{z\}, \delta, z, S)$:

1. Ist $A \to q$ eine Regel in P mit $A \in N$ und $q \in (N \cup \Sigma)^+$, so sei $(z, q) \in \delta(z, \lambda, A)$.
2. Für jedes $a \in \Sigma$ sie $(z, \lambda), \in \delta(z, a, a)$.

6.6.3 PDA → Kontextfrei

Von $M = (\Sigma, \Gamma, Z, \delta, z_0, \#)$ zu $G = (\Sigma, \{S\} \cup Z \times \Gamma \times Z, S, P)$:
O.B.d.A Für alle δ-Regeln der Form $zaA \to z'B_1B_2 \dots B_k$ gelte $k \leq 2$. P besteht dann aus den folgenden Regeln:

1. $S \to (z_0, \#, z)$ für jedes $z \in Z$.
2. $(z, A, z') \to a$, falls $(z', \lambda)\ in \delta(z, a, A)$.
3. $(z, A, z') \to a(z_1, B, z')$ falls $(z_1), B) \in \delta(z, a, A)$.
4. $z, A, z') \to a(z_1, B, z_2)(z_2, C, z')$, falls $z_1, BC) \in \delta(z, a, A)$.

Wobei $z, z', z_1, z_2 \in Z, A, B, C \in \Gamma$ und $a \in \Sigma \cup \{\lambda\}$.

7 Deterministiche Kontextfreie Sprache, DCF

$M = (\Sigma, \Gamma, Z, \delta, z_0, \#, F)$

1. $M' = (\Sigma, \Gamma, Z, \delta, z_0, \#)$ ist PDA
2. $(\forall a \in \Sigma)(\forall A \in \Gamma)(\forall z \in Z)[\|\delta(z, a, A)\| + \|\delta(z, \lambda, A)\| \leq 1]$;
3. $F \subseteq Z$ ist eine ausgezeichnete Teilmenge von Endzuständen (M akzeptiert per Endzustand, nicht per leerem Keller)

Die akzeptierte Sprache ist

$$L(M) = \{x \in \Sigma^* | (z_0, x, \#) \vdash_M^* (z, \lambda, \gamma) \text{ für ein } z \in F \text{ und } \gamma \in \Gamma^*\}$$

Eine Sprache heißt Deterministisch kontextfrei wenn es einen deterministischen Kellerautomaten M gibit mit $A = L(M)$.

7.0.1 Abschlusseigenschaften DCF

DCF ist abgeschlossen unter Komplement

8 $\mathcal{L}_0$ Sprachen (Turingmaschinen)

$M = (\Sigma, \Gamma, Z, \delta, z_0, \Box, F)$
Mit k Bändern:

- Σ: Eingabe-Alphabet,
- Γ: Arbeitsalphabet mit $\Sigma \subseteq \Gamma$,
- Z: endliche Menge von Zuständen mit $Z \cap \Gamma = \emptyset$,
- $\delta : Z \times \Gamma^k \to \mathcal{P}(Z \times \Gamma^K \times \{L, R, N\}^k)$: Überführungsfunktion,
- $z_0 \in Z$: Startzustand,
- $\Box \in \Gamma - \Sigma$: „Blank"-Symbol
- $F \subseteq Z$: Menge der Endzustände

Spezialfall der deterministischen Turingmaschine mit k Bändern mit k Bändern ergibt sich, wenn δ von $Z \times \Gamma^k$ nach $Z \times \Gamma^k \times \{L, R, N\}^k$ abbildet. Für $k = 1$ ergibt sich die 1-Band-Turingmaschine die mit TM abgekürzt wird. Jede Turingmaschine kann durch eine TM simuliert werden. Statt $(z', b, x) \in \delta(z, a)$ mit $z, z' \in Z, x \in \{L, R, N\}, a, b \in \Gamma$ schreiben wir kurz: $(z, a) \to (z', b, x)$, oder $za \to z'bx$

8.1 Linear beschränkte Automaten (LBA)

1. Verdoppele das Eingabe-Alphabet Σ mit $\hat{\Sigma} = \Sigma \cup \{\hat{a} | a \in \Sigma\}$
2. Repräsentiere die Eingabe $a_1 a_2 \ldots a_n \in \Sigma^+$ durch das Wort $a_1 a_2 \ldots a_{n-1} \hat{a_n}$ über $\hat{\Sigma}$

- Eine nichtdeterministische TM M heißt linear beschränkter Automat (kurz LBA), fals für all Konfigurationen $\alpha z \beta$ und
  - für alle Wörter $x = a_1 a_2 \ldots a_{n-1} a\_n \in \Sigma^+$ mit
    $$z_0 a_1 a_2 \ldots a_{n-1} \hat{a_n} \vdash_m^* \alpha z \beta$$
    gilt: $|\alpha\beta| = n$, und
  - für $x = \lambda$ mit $z_0 \Box \vdash_M^* \alpha z \beta$ gilt: $\alpha\beta = \Box$
- Die vom LBA M akzeptierte Sprache ist definiert durch
  $$L(M) = \left\{ a_1 a_2 \ldots a_{n-1} a_n \in \Sigma^* \middle| \begin{array}{l} z_0 a_1 a_2 \ldots a_{n-1} \hat{a_n} \vdash_m^* \alpha z \beta \\ \text{mit } z \in F \text{ und } \alpha, \beta \in \Gamma^* \end{array} \right\}$$

8.1.1 CS $\rightarrow$ LBA

Von $G = (\Sigma, N, S, P)$ zu $M = (\Sigma, \Gamma, Z, \delta, z_0, \Box, F)$

1. Eingabe $x = a_1 a_2 \ldots a_n$.
2. Wähle nichtdeterministisch eine Regel $u \rightarrow v$ aus P und suche eine beliebiges Vorkommen von v in der aktuellen Bandinschrift von M.
3. Ersetzt v durch u. Ist dabei $|u| < |v|$, so verschiebe entsprechend alle Symbole rechts der Lücke um diese zu schließen.
4. Ist die aktuelle Bandinschrift nur noch das Startsymbol S, so halte im Endzustand und akzeptiere; andernfalls gehe zu (2) und wiederhole.

8.1.2 LBA $\rightarrow$ CS

Von $M = (\Sigma, \Gamma, Z, \delta, z_0, \Box, F)$ und $G = (\Sigma, \{S, A\} \cup \{\Delta \times \ \Sigma\}, S, P)$ mit

$$\Delta = \Gamma \cup (Z \times \Gamma)$$

$$\vdots$$

$$\vdots$$

8.1.3 $\mathcal{L}_{\prime} \rightarrow$ TM

1. Eingabe $x = a_1 a_2 \ldots a_n$.
2. Wähle nichtdeterministisch eine Regel $u \rightarrow v$ aus P und suche eine beliebiges Vorkommen von v in der aktuellen Bandinschrift von M.
3. Ersetzt v durch u. Ist dabei $|u| < |v|$, so verschiebe entsprechend alle Symbole rechts der Lücke um diese zu schließen.
4. Ist die aktuelle Bandinschrift nur noch das Startsymbol S, so halte im Endzustand und akzeptiere; andernfalls gehe zu (2) und wiederhole.

Wortproblem:

$$\text{Wort}_i = \{(G, X) | G \text{ist Typ-i-Grammatik und } x \in L(G)\}$$

Typ 3	reguläre Grammatik deterministhischer endlicher Automat(DFA) nichtdeterministischer endlicher Automat (NFA) regulärer Ausdruck (REG)
deterministisch	LR(1)-Grammatik deterministischer Kellerautomat (DPDA)
Typ 2	kontextfreie Grammatik Kellerautomat (PDA)
Typ 1	kontextsensitive Grammatik linear beschränkter Automat (LBA)
Typ 0	Typ-0-Grammatik Turingmaschine (NTM bzw. DTM)

Deterministischer Automat	Nichtdeterministischer Automat	äquivalent?
DFA	NFA	ja
DPDA	PDA	nein
DLBA	LBA	?
DTM	NTM	ja

	Typ 3	det.kf.	Typ2	Typ 1	Typ 0
Schnitt	ja	nein	nein	ja	ja
Vereinigung	ja	nein	ja	ja	ja
Komplement	ja	ja	nein	ja	nein
Konkatenation	ja	nein	ja	ja	ja
Iteration	ja	nein	ja	ja	ja
Spiegelung	ja	nein	ja	ja	ja

Typ 3 (DFA gegeben)	lineare Komplexität
det. kf.	lineare Komplexität
Typ 2 (CNF gegeben)	Komplexität $\mathcal{O}(n^3)$ (CYK-Algorithmus
Typ 1	exponentielle Komplexität
Typ 0	unentscheidbar (d.h. algorithmisch nicht lösbar)