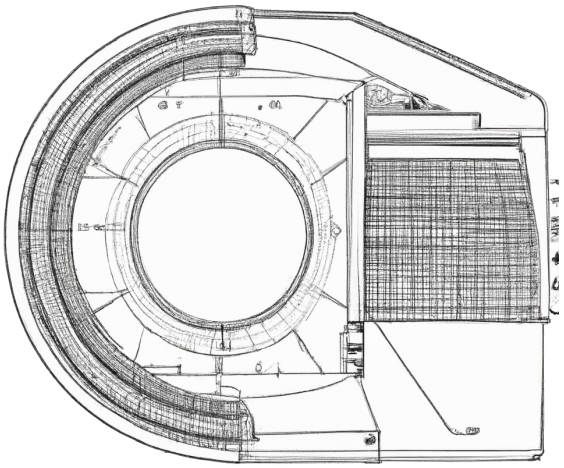
Einführung in die Theoretische Informatik

Felix Ichters*

Sommersemester 2023

Begleitmaterial zur Vorlesung 'Einführung in die Theoretische Informatik'.



*Universität Heidelberg

Inhaltsverzeichnis

Teil I.
Definitionen

Notationen

- Für $n \in \mathbb{N}_0$, sei $[n] = \{1, \ldots, n\}$ und $[n]_0 = \{0, 1, \ldots, n\}$.
- Für eine Menge A und $n \in \mathbb{N}$ ist $A^n = \{(a_1, \ldots, a_n) : a_1, \ldots, a_n \in A\}$.
- Für $n \in \mathbb{N}$ ist eine n-äre partielle Funktion $\varphi : A^n \rightsquigarrow B$ eine Funktion mit $dom(\varphi) \subseteq A^n$ und $Im(\varphi) \subseteq B$. Für $a_1, \ldots, a_n \in A$ bedeutet $\varphi(a_1, \ldots, a_n)\downarrow$, dass $a_1, \ldots, a_n \in dom(\varphi)$ gilt und $\varphi(a_1, \ldots, a_n)\uparrow$ beduetet, dass $(a_1, \ldots, a_n) \notin dom(\varphi)$. Die partielle Funktion φ ist total, wenn $dom(\varphi) = A^n$ gilt.
- Eine lineare Ordnung auch totale Ordnung, auf einer Menge A ist eine Relation $\leq \subseteq A^2$, so dass die folgenden Eigenschaften erfüllt sind

  $a \leq a \ \forall a \in A$ (Reflexivität)

  $a \leq b \wedge b \leq a \Rightarrow a = b \ \forall a, b \in A$ (Antisymmetrie)

  $a \leq b, b \leq c \Rightarrow a \leq c \ \forall a, b, c \in A$ (Transitivität)

  $a \leq b \vee b \leq a \ \forall a, b \in A$ (Totalität)

1. Grundlagen

1.1. Alphabet

Ein Alphabet ist eine nichtleere Menge Σ,die Elemente heißen Symbole.

1.2. Wörter

Ein Wort über einem Alphabet Σ ist eine endliche Folge von Symbolen aus Σ.
Für $i \in [|w|]$ bezeichnet $w(i)$ das i-te Element von w und für Symbole $a_1, \ldots, a_n \in \Sigma$ bezeichnet $a_1, a_2, \ldots, a_n$ das Wort w der Länge n mit $w(i) = a_i \ \forall i \in [n]$.

1.3. Sprache

Eine Sprache ist eine Menge von Wörtern über einem gemeinsamen Alphabet Σ.

1.4. Σ^*

Die Menge aller Wörter über Σ wird mit Σ^* bezeichnet.
Für $n \in \mathbb{N}_0$ setzen wir

$$\Sigma^{\leq n} := \{w \in \Sigma^* : |w| \leq n\}$$
$$\Sigma^{=n} := \{w \in \Sigma^* : |w| = n\}$$
$$\Sigma^{\geq n} := \{w \in \Sigma^* : |w| \geq n\}$$
$$\Sigma^+ := \Sigma^{\geq 1}$$

1.5. Verkettung

Für Wörter w_1, w_2 ist die Verkettung $w_1 w_2$ ist definiert durch

$$w_1 w_2 := w_1(1) \ldots w_1(|w_1|) w_2(1) \ldots w_2(|w_2|).$$

Für Sprachen L_1, L_2 sei $L_1 L_2$ definiert durch

$$L_1 L_2 := \{w_1 w_2 : w_1 \in L_1, w_2 \in L_2\}$$

1.6. Präfix, Infix, Suffix

Seien u,v Wörter

- u ist Präfix von v, kurz $u \sqsubseteq v$, falls es ein Wort w gibt, sodass uw=v
- u ist Infix von v, falls es Wörter w_1, w_2 gibt, sodass $v = w_1 u w_2$
- u ist Suffix von v, falls es ein Wort w gibt, sodass v=wu

1.7. Homomorphismus

Für Sprachen L, M heißt eine Funktion $\varphi : L \to M$ Homomorphismus von Sprachen, wenn gilt:

$$\varphi(uv) = \varphi(u)\varphi(v)$$

1.8. Längenlexikographische Ordnungen

Es gilt $u \leq_{llex} v$ wenn eine der folgenden Bedingungen erfüllt ist:

1. $|u| < |v|$
2. $|u| = |v|$ und ist $i \in [|u|]$ minimal mit $u(i) \neq v(i)$, so gilt $u(i) \leq v(i)$

1.9. bin(i)

$bin(i)$ ist die Funktion für das in Längenlexikopraphischer Reihnfolge $i + 1$-te Binärwort. $1bin(i)$ beschreibt die Binärdarstellung von $i + 1$.

2. Turingmaschienen

2.1. Turingmaschiene

Eine k-Band Turingmaschiene ist ein Tupel

$$M = (Q, \Sigma, \Gamma, \Delta, s, F)$$

- Q ist die endliche Zustandsmenge
- Σ ist das Eingabealphabet
- Γ das Bandalphabet mit $\Sigma \subseteq \Gamma$ und $\square \in \Gamma \backslash \Sigma$
- $\Delta \subseteq Q \times \Gamma^k \times Q \times \Gamma^k \times \{L, S, R\}^k$ die Übergangsrelation
- $s \in Q$ der Startzustand
- $F \subseteq Q$ die Menge der akzeptierten Zustände

Die Elemente von Δ heißen Instruktionen, eine Instruktion sieht wie folgt aus:

$$(q, a_1, \ldots, a_k, q', a'_1, \ldots, a'_k, B_1, \ldots, B_k)$$

Eine TM ist eine DTM, wenn es $\forall b \in Q \times \Gamma^k$ höchstens eine Instruktion $i \in \Delta$ mit Bedingungsteil b gibt

2.2. Konfiguration

Eine Konfiguration einer k-TM ist ein Tupel

$$C = (q, w_1, \ldots, w_k, p_1, \ldots, p_k) \in Q \times (\Gamma^*)^k \times \mathbb{N}^k$$

Die Startkonfiguration zur Eingabe $(u_1, \ldots, u_n) \in (\Sigma^*)^n$ mit $n \in \mathbb{N}$, ist die Konfiguration

$$start(u_1, \ldots, u_n) = (s, u_1 \square u_2 \square \ldots \square u_n, \square, \ldots, \square, 1, \ldots, 1)$$

Die Stoppkonfiguartion ist die Konfiguration zu der es keine Nachfolgekonfiguration gibt

2.3. Nachfolgekonfiguration

Für Konfigurationen $C = (q, w_1, \ldots, w_k, p_1, \ldots, p_k)$ und $C' = (q', w'_1, \ldots, w'_k, p'_1, \ldots, p'_k)$ einer k-TM,
ist die Konfiguration C' **Nachfolgekonfiguration** von C, wenn es eine Instruktion

$$(q, w_1(p_1), \ldots, w_k(p_k), q', a'_1, \ldots, a'_k, B_1, \ldots, B_k) \in \Delta$$

gibt sodass

$$w_i = \begin{cases} \square a'_i w_i(2) \ldots w_i(|w_i|) & \text{falls } p_i = 1 \text{ und } B_i = L \\ w_i(1) \ldots w_i(|w_i| - 1) a'_i \square & \text{falls } p_i = |w_i| \text{ und } B_i = R \\ w_i(1) \ldots w_i(p_1 - 1) a'_i w_i(p_i + 1) \ldots w_i(|w_i|) & \text{sonst} \end{cases}$$

und

$$w_i = \begin{cases} 1 & \text{falls } p_i = 1 \text{ und } B_i = L \\ p_i - 1 & \text{falls } p_i \geq 2 \text{ und } B_i = L \\ p_i & \text{falls } B_i = S \\ p_i + 1 & \text{falls } B_i = R \end{cases}$$

für alle $i \in [k]$ gelten.

2.4. Rechnung

Es bezeichne $\rightarrow_M$ die Relation auf der Menge der Konfigurationen einer k-TM M, sodass $C \rightarrow_M C'$
falls C, C' Konfigurationen von M sind wobei C' eine Nachfolgekonfiguration von C ist.

Eine **endliche partielle Rechnung** ist eine endliche Folge $C_1, \ldots, C_n$ von Konfigurationen von M mit $C_i \rightarrow_M C_{i+1} \forall i \in [n-1]$.
Eine **unendlich partielle Rechnung** ist eine unendliche Folge $C_1, C_2, \ldots$ von Konfigurationen von M mit $C_i \rightarrow_M C_{i+1} \forall i+1 \in \mathbb{N}$. Eine Rechnung zur Eingabe $(w_1, \ldots, w_n) \in (\Sigma^*)^n$ mit $n \in \mathbb{N}$ ist ein unendlich partielle Rechnung $start_M(w_1, \ldots, w_n) = C_1, C_2, \ldots, C_m$.

Ist M eine k-DTM, so gilt es $\forall n \in \mathbb{N}$ und $(w_1, \cdots, w_n) \in (\Sigma^*)^n$ genau eine Rechnung zur Eingabe $(w_1, \cdots, w_n)$. Ist M eine k-DTM, so gilt es $\forall n \in \mathbb{N}$ und $(w_1, \cdots, w_n) \in (\Sigma^*)^n$ genau eine Rechnung zur Eingabe $w_1, \cdots, w_n)$.

2.5. Total

Totale TMs, sind TMs die bei jeder Eingabe immer anhalten. Alle Rechnungen müssen endlich sein.

2.6. Akzeptierte Sprache

Eine Stoppkonfiguration ist akzeptiert, wenn $q \in F$.

Die akzeptierte Sprache L(M) ist die Sprache über dem Alphabet Σ, so dass $\forall w \in \Sigma^*$ genau dann $w \in L(M)$ gilt, wenn es eine endliche Rechnung $C_1, \ldots, C_n$ zur Eingabe w gibt, bei der C_n eine akzeptierte Stoppkonfiguartion ist.

Für nicht-deterministische Tms heißt das, dass es für die Wörter in der akzeptierten Sprache nur mind. eine in einer akzeptierten Stoppkonfiguartion endende endliche Rechnungen zur Eingabe w geben muss.
Für Wörter w die nicht in L(M) sind, sind alle Rechnung von M zur Eingabe am Ende nicht in einer akzeptierten Stoppkonfiguartion oder unendlich

2.7. Entscheidbar

Eine Sprache L ist genau dann entscheidbar, wenn es eine totale k-TM mit L(M) = L gibt.

2.8. Rekursiv aufzählbar

Eine Sprache L ist genau dann rekursiv aufzählbar, wenn es eine k-TM mit akzeptierter Sprache L gibt.

Wir schreiben RE für die Klasse der rekursiv aufzählbaren Sprachen. Die Aufzählbarkeit leitet sih daraus ab, dass es für eine rekuriv aufzählbare Sprache L über einem Alühabet Σ möglich ist effektive Verfahren anzugeben ,die die Wörter von L aufzählen, also dass eine endlich oder unendliche Aufzählung von $A = w_1, w_2, \cdots$ mit $w_1, w_2, \cdots = L$ existiert.

- *Jede entscheidbare Sprache ist rekuriv aufzählbar*
- *Alle endlichen Sprachen sind entscheidbar*
- *Eine Sprache L über einem Alphabet Σist genau dann entscheidbar, wenn L und $L^c := (\Sigma^*)/L$ rekursiv aufzähbar sind.*

2.9. Ausgabe

Die Ausgabe $out_M(C)$ bei Konfiguration C ist das Präfix $w \sqsubseteq w_1(p_1) \dots w_1(|w_1|)$ max. Länge mit $w \in (\Gamma \backslash \{\square\})^*$.

Also das längst mögliche Wort, dass auf Band 1 rechts vom Lesekopf steht und nicht $\square$ enthält.

2.10. Berechnete Funktion

Sei M eine k-DTM. Die von M berechnete n-äre partielle Funktion φ_M ist die partielle Funktion:

$$\varphi_M : (\Sigma^*)^n \rightsquigarrow (\Gamma \backslash \{\square\})^*$$

sodass folgendes gilt:

1. Ist die Rechnung zur Eingabe $w_1, \dots, w_n$ die endliche Rechnung $C_1, \dots, C_m$ so gilt $\varphi_M(w_1, \dots, w_n) = out_M(C_m)$
2. Ist die Rechnung zur Eingabe $w_1, \dots, w_n$ unendlich, so gilt $\varphi_M(w_1, \dots, w_n) \uparrow$

Nur deterministische TMs können Funktionen berechnen.

2.11. Partiell Berechnbar

Für Σ, Γ und eine partielle Funktion $U : \Sigma^* \rightsquigarrow \Gamma^*$ ist φ berechenbar, wenn es ein $k \in \mathbb{N}$ gibt und eine k-DTM M mit $\varphi_M = \varphi$ gibt.
Ist φ total und partiell berechenbar, so ist φ berechenbar.

2.12. Charakteristische Funktion

Sei L eine Sprache über dem Alphabet Σ.

1. Die charakteristische Funktion von L als Sprache über Σ ist die Funktion $\mathbb{1}_L : \Sigma^* \to \{0, 1\}$ mit $\mathbb{1}_L(w) = 1 \ \forall w \in L$ und $\mathbb{1}_L(w) = 0 \ \forall w \in \Sigma^* \backslash L$.
2. Die partiell charakteristische Funktion von L als Sprache über Σ ist die partielle Funktion $\chi_L : \Sigma^* \rightsquigarrow \{1\}$ mit $\chi_L(w) = 1 \ \forall w \in L$ und $\chi_L(w) \uparrow \forall w \in \Sigma^* \backslash L$.

Bemerkung

Sei L eine Sprache über einem Alphabet Σ

- L ist genau dann entscheidbar, wenn $\mathbb{1}_L$ berechenbar ist.
- L ist genau dann rekursiv aufzählbar, wenn x_L partiell berechenbar ist.

2.13. Normiet

Eine 1-DTM M heißt normiert, wenn Q=0,...,n für ein $n \in \mathbb{N}_0$, $\Sigma = \{0,1\}$, $\Gamma = \{\square, 0, 1\}$, $\Delta = 0$, $F = \{s\}$.

Alle TMs mit Eingabealphabet 0,1 lassen sich mit folgenden Schritten in eine normierte TM mit gleicher erkannter Sprache und gleicher berechneten Funktion umwandeln.

- Von nicht-Determinismus zu Determinismus

  Eine DTM kann die Rechnungen einer nicht-Deterministischen TM parallel im Sinne von abwechselnd schrittweise durchführen um schließlich das Verhalten der simulierten TM zu imitieren. Das entspricht einer Breitensuche im Rechnungsbaum.

- Von mehreren Bändern zu einem Band

  Intuitiv können k Bänder auf einem Band simuliert werden, indem die Felder des einen Bandes in k-Teilfelder unterteilt werden, die jeweils die gleichen Bandalphabetbuchstaben wie zuvor als Beschriftung zulassen und es zudem erlauben zu notieren, dass der simulierte Kopf des simulierten Bandes dort steht.

- Von beliebigem Bandalphabet zu $\{\square, 0, 1\}$

  Andere Bandalphabete können bei einem Alphabet Wechsel zum Bandalphabet $\{\square, 0, 1\}$ simuliert werden, indem mehrere nebeneinander liegende Felder verwendet werden um ein Symbol des vorigen Bandalphabets durch ein Binärwort zu beschreiben . Die TM liest stets nur ein Feld, es wird daher also nötig sein die Zustandsmenge so zu erweitern, dass angrenzende Felder im Zustand gespeichert werden können.

2.14. Code

Wir betrachten die Funktion code mit geeigneter Definitionsmenge und Zielmenge $\{0,1\}$.

- Codieren der Bewegungsrichtung

  $code(L) = 10$

  $code(S) = 00$

  $code(R) = 01$

- Codieren der einzelnen Instruktionen

  $I = (q, a, q', a', B)$

  $code(I) = 0^{|bin(q)|}1bin(q)a0^{|bin(q')|}1bin(q')a'code(B)$

- Codieren des Instruktinssatzes

  $code(\Delta) = code_1(\Delta) \dots code_{|\Delta|}(\Delta)$

- Codieren der normierten TM

  $$code(M) = 0^{|bin(n)|}1bin(n)code(\Delta)$$

Jede normierte TM hat einen Code und zwei verschiedene niemals den gleichen. Die Sprache der TMs ist entscheidbar.

2.15. Standardaufzählung

*Sei $\hat{w}_0, \hat{w}_1, \ldots$ die Aufzählung aller Codes normierter TMs in längenlexikopraphischer Ordnung.*Für $e \in \mathbb{N}$ sei $\mathcal{M}_e$ die durch $\hat{w}_e$ codierte TM und für $n \in \mathbb{N}$ sei $\Phi_e^n : \mathbb{N}_0^n \to \mathbb{N}_0$ die von

$\mathcal{M}_e$ berechnete n-äre partielle Funktion.
Für $n \in \mathbb{N}$ heißt die Folge $(\Phi_e^n)_e \in \mathbb{N}$ Standardaufzählung der n-ären partiell berechenbaren Funktion.
Für $n \in \mathbb{N}$ und eine partiell berechenbare n-äre partielle Funktion $\varphi : \mathbb{N}_0^n \to \mathbb{N}_0$ heißt jede Zahl $e \in \mathbb{N}_0$ mit $\Phi_e^n = \varphi$ Index von φ.

Ergibt sich n aus dem Kontext so schreiben wir auch Φ_e statt Φ_e^n

Für $n \in \mathbb{N}$ und eine partielle berechnbare n-äre partielle Funktion $\Phi : \mathbb{N}_0^n \to \mathbb{N}_0$ gibt es unendlich viele Indizes von φ.

2.16. $\mathcal{U}$

Es bezeichne $\mathcal{U}$ die normierte TM, bei Eingabe $(e, x_1, \ldots, x_n) \in \mathbb{N}_0^{n+1}$ wobei $n \in \mathbb{N}$ die normierte TM $\mathcal{M}_e$ bei Eingabe $x_1, \ldots, x_n$ simuliert und falls diese terminiert die Asugabe der Simulation ausgibt.

2.17. Universell

Eine DTM $\mathcal{U}$ heßt universell, wenn es für alle $n \in \mathbb{N}$ und alle partiell berechenbaren Funktionen $\varphi : \mathbb{N}_0^n \rightsquigarrow \mathbb{N}_0$ ein $e \in \mathbb{N}_0$ gibt, sodass $\mathcal{U}(e, x_1, \ldots, x_n) = \varphi(x_1, \ldots, x_n)$ für alle $x_1, \ldots, x_n \in \mathbb{N}_0$ gilt.

2.18. $s_n^m - Theorem$

Für alle $m, n \in \mathbb{N}$ existiert eine berechenbare Funktion $s_n^m : \mathbb{N}_0^{m+1} \to \mathbb{N}_0$ mit

$$\Phi_e^{m+n}(x_1, \ldots, x_m, y_1, \ldots, y_n) = \Phi^n_{s_n^m(e,x_1,\ldots,x_m)}(y_1, \ldots, y_n)$$

für alle $e, x_1, \ldots, x_m, y_1, \ldots, y_n \in \mathbb{N}_0$.

Beweis. Fixiere $m \in \mathbb{N}$. Betrachte die DTM S , die bei Eingabe $(e, x_1, \cdots, x_m) \in \mathbb{N}_0^{m+1}$ wie folgt verfährt??.

- Zunächst bestimmt S den Code von $\mathcal{M}_e$
- der Code von $\mathcal{M}_\urcorner$wird dann in einen Code einer normierten TM $\mathcal{M}$ umgewandet, die zunächst $x_1\square\cdots\square x_m\square$ neben die Eingabe schreibt, dan den Kopf auf das erste Feld des beschriebenen Bandteilsbewegt und dann wie $\mathcal{M}_\urcorner$ arbeitet.

- Es wird bestimmt an welcher Stelle der Standardaufzählung der Code von auftaucht und diese Stelle wird ausgegeben.

Sei s_n^m die von S berechnete $(m+1)$-äre partielle Funktion. Dann ist s_n^m eine Funktion wie gewünscht. □

2.19. Diagonales Halteproblem

Die Menge $H_{diag} := \{e \in \mathbb{N}_0 : \Phi_e(e) \downarrow\}$ heißt diagonales Halteproblem.
Das diagonale Halteproblem ist rekursiv aufzählbar, jedoch nicht entscheidbar.

2.20. m-Reduktion

Für eine Sprache A über einem Alphabet Σ und eine Sprache B über einem Alphabet Γ ist A genau dann many-one-reduzierbar, auch m-reduzierbar, auf B, kurs $A \leq_m B$, wenn es eine berechenbare Funktion $f : \Sigma^* \to \Gamma^*$ gibt, so dass

$$w \in A \Leftrightarrow f(w) \in B$$

für alle $w \in \Sigma^*$ gilt.
gelten $A \leq_m B$ und $B \leq_m A$, so sind A und B m-äquibalent, kurz $A =_m B$.

2.21. Postsches Korrespondenzproblem

Für ein Alphabet Σ sei eine Instanz des Postschen Korrespondenzproblem über Σ eine endliche Teilmenge $I \subseteq (\Sigma^+)^2$ Eine Lösung für eine solche Instanz ist eine endliche Folge $(u_1, v_1), \ldots, (u_n, v_n)$ von Paaren in I mit $n \geq 1$, so dass

$$u_1 \ldots u_n = v_1 \ldots v_n$$

Gibt es eine Lösung für eine Instanz des Postschen Korrespondenzproblems, so heißt diese Instanz lösbar.
Das Postsche Korrespondenzproblem über einem Alphabet Σ, kurz PCP_Σ, ist die Menge aller lösbarer Instanzen des Postschen Korrespondenzproblems über Σ.

Für ein Alphabet Σ sei eine Instanz des modifizierten Postschen Korrespondenzproblems über Σ ein Paar (p, I), wobei $I \subseteq (\Sigma^+)^2$ eine endliche Teilmenge und $p \in I$ ein Paar von Wörtern ist.
Eine Lösung für ein solche Instanz ist eine endliche Folge $(u_1, v_1), \ldots, (u_n, v_n)$ von Paaren in I, so dass

$$p = (u_1, v_1) \text{ und } u_1 \ldots u_n = v_1 \ldots v_n$$

Gibt es ein Lösung für eine Instanz des modifizierten Postschen Korrespondenzproblems, so heißt diese Instanz lösbar.
Das modifizierte Postsche Korrespondenzproblem über einem Alphabet Σ, kurz $MPCP_\Sigma$ ist die Menge aller lösbarer Instanzen des modifizierten Postschen Korrespondenzproblems über Σ.

2.22. Fixpunkt

Ein Fixpunkt einer berechenbaren Funktion $f : \mathbb{N}_0 \to \mathbb{N}_0$ ist ein $e \in \mathbb{N}_0$ mit $\Phi_{f(e)} = \Phi_e$.

2.23. Rekursionstheorem

Für alle partiell berechenbaren Funktion $\varphi : \mathbb{N}_0^2 \rightsquigarrow \mathbb{N}_0$ gibt es ein $e \in \mathbb{N}_0$ mit $\Phi_e(x) = \varphi(e, x) \forall x \in \mathbb{N}_0$.

2.24. Indexmenge

Eine Teilmenge $I \subseteq \mathbb{N}_0$ heißt Indexmenge, wenn $e \in I \Leftrightarrow e' \in I$ für alle $e, e' \in \mathbb{N}_0$ mit $\Phi_e = \Phi_{e'}$ gilt.

3. Automaten

3.1. Endlicher Automat

Ein endlicher Automat, EA, ist ein Tupel $A = (Q, \Sigma, \Delta, s, F)$.

- Q ist eine endliche Menge, die Zustandsmenge
- Σ das Eingabealphabet
- $\Delta \subseteq Q \times \Sigma \times Q$ die Übergangsrelation, so dass es für alle $q \in Q$ und $a \in \Sigma$ ein $q' \in Q$ mit (q, a, q')
- $s \in Q$ der Startzustand
- $F \subseteq Q$ die Menge der akzeptierenden Zustände

Der endliche Automat A ist ein deterministischer endlicher Automat, kurz DEA, wenn es $\forall(q, a) \in Q \times \Sigma$ *genau ein q' gibt mit* $(q, a, q') \in \Delta$. *Im Sinne der obigen Betrachtung entspricht ein EA* $A = (Q, \Sigma, \Delta, s, F)$ *der 1-TM* $M_A = (Q, \Sigma, \Sigma \cup \{\square\}, \{(q, a, q', a, R) : (q, a, q') \in \Delta\}, s, F)$.

3.2. Übergangsfunktion eines EA

Die Übergangsfunktion eines EA A ist die Funktion $\delta^*_A(q, \lambda) = \{q\}$ und

$$\delta^*_A(q, aw) = \bigcup_{q' \in \delta_A(q,a)} \delta^*_A(q', w)$$

$\forall q \in Q, a \in \Sigma, w \in \Sigma^*$
Für $Q_0 \subseteq Q$ und $w \in \Sigma^*$ schreiben wir $\delta^*_A(Q_0, w)$ statt $\bigcup_{q \in Q_0} \delta^*_A(q, w)$.

Sei A = (Q, Σ, Δ, s, F) ein EA

(i) $\forall$ $q \in Q$ *und* $a \in \Sigma$ *gilt* $\delta^*_A(q,a) = \delta_A(q, a)$.

(ii) Ist A ein DEA, $q \in Q$, $a \in \Sigma$ *und* $w \in \Sigma^*$, *und* $|\delta^*_A(q, w)| = 1??$.

(iii) Seien $u,v \in \Sigma^*$ $\forall$ $q \in Q$ *gilt* $\delta^*_A(q, uv) = \delta^*_A(\delta^*_A(Q_0, u), v)$.

3.3. Übergangsfunktion eines DEA

Sei A ein DEA. Auch die Funktion $\delta_{det,A} : Q \times \Sigma \to Q$ mit $\delta_a(q, a) = \{\delta_{drt,A}(q, a)\}$ $\forall q \in Q$ und $a \in \Sigma$ wird Übergangsfunktion von A genannt. Analoges gilt für $\delta^*_{det,A}$ und δ^*_A.
Für $Q_0 \subseteq Q$ und $w \in \Sigma^*$ schreiben wir auch $\delta^*_{det,A}(Q_0, w)$.

Sei Q eine endliche Menge, Σ *ein Alphabet,* $s \in Q$, *und* $F \subseteq Q$.

(i) $\forall$ *Funktionen* $\delta : Q \times \Sigma \to 2^Q$ *gibt es genau einen EA* $A = (Q, \Sigma, \Delta, s, F)$ *mit* $\delta_A = \delta$.

(ii) $\forall$ *Funktionen* $\delta : Q \times \Sigma \to Q$ *gibt es genau einen* $\delta_{det,A} = \delta$.

3.4. akzeptierte Sprache

Sei A ein EA. Die Sprache $L(A) := \{w \in \Sigma^* : \delta_A^*(s, w) \cap F \neq \varnothing\}$ ist die akzeptierte Sprache von A.

3.5. Regulär

Eine Sprache L heißt regulär, wenn es einen EA A mit L(A)=L gibt.
Wir schreiben REG für die Klasse der regulären Sprachen.

Beispiel

Zu jedem Zeitpunkt während der Verbindung der Eingabe durch einen endlichen Automaten höngt der restliche Bearbeitung immer nur vom gegewärtigen Zustand und dem noch einzulesenden Teil der Eingabe ab, nicht aber wie bei TM im allgemeinen von vergangenen Bandmanipulation. Interpretiert man die Eingabe als von einer äußeren Quelle kommend, so ist der Zustand des Automaten also allein durch seinen Zustand gegeben und der nächste Zustand hängt nur vom Zugeführten Symbol ab. Daher bietet sich eine Darstellung eines EA durch ein Übergangsdiagramm oder eine sogenannte Übergangstabelle an.

Sei A := ($\{q_0, q_1\}$, $\{0, 1\}$, Δ, q_0, $\{q_1\}$) mit $\Delta = \{(q_0)\}$
Übergangsdiagramm und übergangstabelle von sehen wie folgt aus:

Zustand/Symbol	0	1
q_0	q_0	q_1
q_1, *	q_1	q_0

3.6. Übergangsdiagramm

Für jeden Zustand gibt es einen Kreis. Zustände in F bekommen einen Doppelkreis. Für $(q, a, q') \in \Delta$ für wir einen Pfeil von dem Kreis von q zu dem Kreis von q' mit der Beschriftung a.
Zustätzlich gibt es einen Pfeil (ohne Beschriftung) aus dem 'Nichts' zu dem Kreis des Startzustandes.

3.7. Potenzautomat

Sei A ein EA. Der Potenzautomat von A ist der DEA $P_a = (2^Q, \Sigma, \Delta', \{s\}, \{P \subseteq Q : P \cap F \neq \varnothing\})$ mit

$$\delta_{det,P_a}(Q_0, a) = \bigcup_{q \in Q_0} \delta_A(q, a) \ \forall Q_0 \subseteq Q \ \forall a \in \Sigma$$

Satz

Eine Sprache L ist genau dann regulär, wenn es einen DEA A mit L(A) = L gibt.

Beweis. Sei A = (Q, Σ, Δ, s, F) ein EA mit Potenzautomat P_A. Es genügt zu zeigen, dass L(A) = L(P_A). Hierfür genügt es zu zeigen, dass:

$$\delta^*_{det,P} = \delta^*_A(s,w) \forall w \in \Sigma^*(*)$$

Denn damit folgt

$$w \in L(P_A) \Leftrightarrow \delta^*_{P_A}(\{s\},w) \cap \{P \subseteq Q : P \cap F \neq \varnothing\} \neq \varnothing$$

$$\Leftrightarrow \delta^*_{det,P_A}(\{s\},w) \cap F \neq \varnothing$$

$$\underset{(*)}{\Leftrightarrow} \delta^*_A(\{s\},w) \cap F \neq \varnothing$$

$$\Leftrightarrow w \in L(A)$$

Wir zeigen (*) mittels vollständiger Induktion über |w|. Es gilt $\delta^*_{det,P_A}(\{s\}, \lambda) = \delta^*_A(s, \lambda)$. Sei w $\in \Sigma^+$ mit $\delta^*_{det,A}(\{s\}, v) = \delta^*_A(s, v)$ $\forall$ v $\in \Sigma^{\leq |w|-1}$. Nun zeigen wir (*) Sei va := w mit a $\in \Sigma$ und |v| = |w| - 1.

$$\delta^*_{det,P_A}(\{s\},w) \underset{\text{Bem 4.5}}{=} \delta^*_{det,P_A}(\delta^*_{det,P_A}(\{s\},v),a)$$

$$\underset{\text{Ind. hyp}}{=} \delta^*_{det,P_A}(\delta^*_{det,P_A}(\{s\},v),a)$$

$$= \bigcup_{q \in \delta^*_{det,A}} \delta_A(q,a)$$

$$= \delta^*_A(\delta^*_A(s,v),a)$$

$$= \delta^*_A(s,va)$$

$$= \delta^*_A(s,w)$$

□

4. Reguläre Sprachen

Wir verschieben den Fokus von endlichen Automaten auf die Klasse der von diesen erkannten Sprachen. Dabei spielen endl. Automaten weiterhin eine wichtige Rolle. Wegen 4.7 beschränken wir uns auf deterministische endliche Automaten.
Sei A DEA. Die Menge der zulässigen Eingaben Σ^* ist unendlich groß, die Menge der Zustände Q is aber endlich. Zwangsläufig wir A also das Einlesen verschiedener Eingaben im gleichen Zustand abschließen (und somit gleich behandeln). Dies führt zum Begriff der A-Äquivalenz.

4.1. Äquivalenzrelation

Sei A eine Menge. Eine Äquivalenzrelation auf A ist eine Relation $\sim\subseteq A$, so dass die folgenden Eigenschaften erfüllt sind.

1. $a \sim a \ \forall a \in A$ (Reflexivität)
2. $a \sim b \Rightarrow b \sim a \ \forall a, b \in A$ (Symmetrie)
3. $a \sim b, b \sim c \Rightarrow a \sim c \ \forall a, b, c \in A$ (Transitivität)

Die Äquivalenzklasse eines Elements $a \in A$ bezüglich $\sim$ ist die Menge $[a]_\sim := \{a' \in A : a' \sim a\}$. Der Index von $\sim$ isr die Kardinalität der Menge $A/\sim := \{[a]_\sim : a \in A\}$ falls diese endlich ist und unendlich andernfalls.

4.2. A-Äquivalenz

Sei A ein DEA. mit erweiterter Übergangsfunktion $\delta^* : Q \times \Sigma \to Q$.
Die A-Äquivalenz ist die Relation $\sim_A$ auf Σ^* mit

$$u \sim_A v \Leftrightarrow \delta^*(s, u) = \delta^*(s, v) \ \forall u, v \in \Sigma^*$$

(i) Die A-Äquivalenz ist eine Äquivalenzrelation.

(ii) Der Index von $\sim_A$ ist höchstens $|Q|$.

(iii) Es gilt $L(A) = \bigcup_{w \in L(A)} [w]_{\sim A}$.

4.3. Rechtskongurenz

Sei Σ ein Alphabet. Eine Rechtskongurenz auf Σ^* ist eine Äquivalenzrelation $\sim\subseteq (\Sigma^*)^2$ mit $u \sim v \Rightarrow uw \sim vw \ \forall u, v, w \in \Sigma^*$.

Proposition

Sei $A = (Q, \Sigma, \Delta, s, F)$ ein DEA. Die A-Äquivalenz $\sim_A$ ist eine Rechtskonqruenz auf Σ^*.

Beweis. Seien $u, v, w \in \Sigma^*$ mit $u \sim_A v$. Dann gilt

$$\delta^*_{det,A}(s, uw) = \delta^*_{det,A}(\delta^*_{det,A}(s, u), w) = \delta^*_{det,A}(\delta^*_{det,A}(s, v), w)$$
$$= \delta^*_{det,A}(s, vw).$$

Dann gilt $uw \sim_A vw$. □

Zu jedem DEA A gibt es also eine dazugehärige Rechtskonguenz $\sim$ auf Σ^* mit endlichem Index so dass L(A) die Vereinigung von Äquivalenzklasse von $\sim_A$ ist. Tatsächlich gilt auch die Umkehrung: Ist L die Vereinigung von Äquivalenzklasse einer Rechtskongruenz $\sim$ mit endlichem Index, so gibt es einen DEA A mit $L(A) = L$.

4.4.

Sei Σ ein Alphabet und L Vereinigung von Äquivalenzklassen einer Rechtskongurenz $\sim$ auf Σ^* mit endlichem Index. Es bezeichne

$$A_{\sim,L} := (\Sigma^*/\sim, \Sigma, \Delta, [\lambda]_\sim, \{[w]_\sim : w \in L\})$$

den DEA mit $\delta_{det,A_{\sim,L}}([w]_\sim, a) = [wa]_\sim \ \forall w \in \Sigma^*$ und $a \in \Sigma$.

Lemma

Sei Σ ein Alphabet, L Vereinigung von Äquivalenzklassem einer Rechtskongruenz $\sim$ auf Σ^* mit endlichem Index und sei $\delta^* : \Sigma^*_{/\sim} \times \Sigma^* \to \Sigma^*_{/\sim}$ die erweiterte Übergangsfunktion von $A_{\sim,L}$. Dann gilt $\delta^*([\lambda]_\sim, w) = [w]_\sim \forall w \in \Sigma^*$.

Beweis. Wir verwenden vollständige Induktion über $|w|$. Es gilt $\delta^*([\lambda]_\sim, \lambda) = [\lambda_\sim]$. Sei nun $w \in \Sigma^+$ mit $\delta^*([\lambda]_\sim, v) \ \forall v \in \Sigma^{\leq |w|-1}$. Es genügt zu zeigen, dass $\delta^*([\lambda]_\sim, w = [w]_\sim)$.
Sei $va = w$ mit $a \in \Sigma$
Dann gilt $\delta^*([\lambda]_\sim, w) = \delta^*(\delta^*([\lambda]_\sim, v), a) = \lambda^*([v]_\sim, a) = [va]_\sim = [w]_\sim$ □

Satz

Sei L die vereinigung von Äquivalenzklasse einer Rechtskongruenz $\sim$ mit endlichem Index Es gibt $L(A_{\sim,L}) = L$

Beweis. Sei Σ das Alphabet, so dass $\sim$ eine Rechtskongruenz auf Σ^* ist. Sei $\delta^* : \Sigma^*_{/\sim} \times \Sigma^* \to \Sigma^*_{/\sim}$ die erweiterte Übergangsfunktion von $A_{\sim,L}$ und sei $w \in \Sigma^*$. Es folgt

$$w \in L(A_{\sim,L}) \Leftrightarrow \delta^*([\lambda]_\sim, w) \in [v]_\sim : v \in L$$
$$\Leftrightarrow [w]_\sim \in [v]_\sim : v \in L$$
$$\Leftrightarrow \exists v \in L : [w]_\sim = [v]_\sim$$
$$\Leftrightarrow \exists v \in L : w \sim v$$
$$\Leftrightarrow w \in L$$

□

Korollar

Eine Sprache L ist genau dann regulär, wenn sie die Verienigung von Äquivalenzklasse einer Rechtskongruenz mit endlichem Index ist.

Beweis. Folgt aus Bemerkung 5.3, Proposition 5.5 und Satz 5.8 □ □

Betrachten man nur deterministische endliche Automaten ohne unerreichbare Zustände, so entsprechen diese bis auf Unbenutzung von Zuständen sogar den Rechtskongruenz mit endlichem Index zusammen mit Vereinigung von Äquivalenzklassn dieser.

4.5. Erreichbar

Sei Σ ein Alphabet. Sei A ein EA mit erw. Übergangsfunktion δ^*. Ein Zustand $q \in Q$ heißt erreichbar in A wenn es ein Wort
$w \in \Sigma^*$ mit $q \in \delta^*(s, w)$ gibt.

4.6. Isomorph

Sei $A_i$für $i \in \{1, 2\}$ ein EA mit Übergangsfunktion δ_i.
Die endlichen Automaten A_1 und A_2 sind Isomorph, kurz $A \cong A_2$, wenn es eine Projektion $f : Q_1 \to Q_2$ gibt, so dass folgendes gilt:

1. $f(s_1) = s_2$
2. $\delta_2(f(q_1), a) = f(\delta_1(q_1, a))\ \forall q_1 \in Q, a \in \Sigma$
3. $f(F_1) = F_2$

Satz

(i) Ist A ein DEA ohne unereichbare Zustände, so gilt $A \cong A_{\sim A, L(A)}$

(ii) Ist L die Vereinuíngung von Äquivalenzklasse einer Rechtskongruenz $\sim$ mit endlichem Index, so gilt $(\sim, L) = (\sim_{A_{\sim,L}, L(A_{\sim,L})})$.

Beweis. (i) Sei $A = (Q, \Sigma, \Delta, s, F)$ eine DEA mit erweiterte Übergangsfunktion $\delta^* : Q \times \Sigma^* \to Q$ ohne unereichbare Zustände , $\sim:=\sim_A$, $A' := A_{\sim,L(A)}$ und sei $\delta' : \Sigma^*/\sim \times \Sigma^* \to \Sigma/\sim$ die erweiterte Übergangsfunktion von A'. Sei $f : Q \to \Sigma^*/\sim$ die Bijektive mit $f(q) := \{w \in \Sigma^* : \delta^*(s, w) = q\}$. Es gelte $f(s) = [\lambda]_\sim$ und $f(F) = \{[w]_\sim : w \in L(A)\}$. Es genügt somit zu zeigen , dass $\delta'(f(q), a) = f(\delta(q, a))\forall q \in Q, a \in \Sigma$. Sei $q \in Q, a \in \Sigma^*$. Es genügt $w \in \delta'(f(q), a) \Leftrightarrow \delta^*(s, w) = \delta^*(q, a)$ zu zeigen. Sei $v \in \Sigma^*$ mit $\delta^*(s, v) = q$. Nun gilt $w \in \delta'(f(q), a) \leftrightarrow w \in \delta'([v]_\sim, a) \leftrightarrow w \sim va \leftrightarrow \delta^*(s, w) = \delta^*(s, va) \leftrightarrow \delta^*(s, w) = \delta^*(q, a)$

(ii) Sei Σ ein Alphabet, $\sim$ eine Rechtskongruenz aud Σ^*, L Vereinigung von Äquivalenzklassen von $\sim$, $A' := A_{\sim,L} = (\Sigma^*/\sim, \Sigma, A', [\lambda]_\sim, \uparrow)$, $\delta'^* : \Sigma^*/\sim \times \Sigma^* \to \Sigma^*/\sim$ die erweiterterte Übergangsfunktion von A' $\{w \in \Sigma^* : w \in L\}$ und $\sim':=\sim_{A'}$. Nach Satz 5.8 gilt $L = L(A')$, es genügt also $\sim=\sim'$ zu zeigen. Sei $u, v \in \Sigma^*$. Es folgt $u \sim v \leftrightarrow [u]_\sim = [v]_\sim \leftrightarrow \delta'([\lambda]_\sim, u) = \delta'([\lambda]_\sim, v) \Leftrightarrow u \sim v$

□

4.7. L-Äquivalenz

Sei L eine Sprache über einem Alphabet Σ. Die L-Äquivalenz von L als Sprache ist die Relation $\sim_L$ auf Σ^* mit

$$u \sim_L v \Leftrightarrow (uw \in L \Leftrightarrow vw \in L \ \forall w \in \Sigma^*)$$

Bemerkung

Sei L eine Sprache über Σ.

1. Die l-Äquivalenz ist eine Rechtskongurenz
2. Es gilt $L = \bigcup_{w \in L} [w]_{\sim L}$

4.8. Partition

Sei A eine Menge. Eine Partition von A ist eine Menge $\mathcal{A} = \{A_1, \ldots, A_n\}$ paarweise disjunkte nichtgeliche Teilmengen von A mit $\bigcup_{i \in [w]} A_i = A$

4.9. Verfeinerung

Seien $\mathcal{A}_1$ und $\mathcal{A}_2$ Partitionen einer Menge A. Die Partition $\mathcal{A}_2$ verfeinert $\mathcal{A}_1$, wenn es $\forall A_2 \in \mathcal{A}_2$ ein $A_i \in \mathcal{A}_1$, mit.... *Seien $\mathcal{A}_1$ und $\mathcal{A}_2$ Partitionen eine Menge A, so dass $\mathcal{A}_2$ die Partition $\mathcal{A}_1$ verfeinert.*

1. *$\forall A' \in \mathcal{A}_1$, gibt es eine Teilmenge $\mathcal{A}'_2 \in \mathcal{A}_2$, die eine Partition von A' ist*
2. *Es gilt $|\mathcal{A}_1| \leq |\mathcal{A}_2|$*
3. *Gilt $|\mathcal{A}_1| = |\mathcal{A}_2|$ dann ist $\mathcal{A}_1 = \mathcal{A}_2$*

Proposition

Seien Σ ein Alphabet und L eine Sprache über Σ und $\sim$ eine Rechtskongurenz auf Σ^* mit $L = \bigcup_{w \in L} [w]_\sim$. Die Partition $\Sigma^* \backslash \sim$ ist eine Verfeinerung der Partition $\Sigma^* \backslash \sim_L$.

Beweis. Seien $u, v \in \Sigma^*$ mit $u \sim v$. Es genügt zu zeigen, dass $u \sim_L v$.
Sei $w \in \Sigma*$. Es genügt $uw \in L \Leftrightarrow vw \in L$ zu zeigen.
Da $\sim$ eine Rechtskongurenz ist folgt $uw \sim vw$. Ist $uw \in L$, so folgt aus $L = \bigcup_{w' \in L} [w']_\sim$ auch $vw \in L$ (analog folt auch $vw \in L \Rightarrow uw \in L$).
$\Rightarrow u \sim_L v$

□

D.h. $\sim_L$ ist die gröbste Partition, die L darstellen kann.

4.10. Minimalautomat

Sei L eine reguläre Sprache über Σ. Der Minimalautomat von L als Sprache über Σ ist der DEA $A_{\sim_{L,L}}$

Satz

Sei L eine reguläre Sprache über Σ und sei M der Minimalautomat von L. Dann gilt:

(i) $L(M) = L$

(iii) ist A ein DEA mit Zustandsmenge Q_A und $L(A) = L$, so gilt $|Q_A| \geq |Q|$

(iii) ist A ein DEA mit $|Q|$ Zuständen und $L(A) = L$, so gitl $A \cong M$

Beweis. (i) folgt aus vorigem Satz

(ii) Aus voriger Bemerkung folgt $|Q_A| \geq |\Sigma/_A|$. Nach voriger Proposition ist $\Sigma^*/_{\sim_A}$ eine Verfeinerung...

(iii) Sei A ein DEA mit $|Q|$ Zuständen und $L(A) = L$.
Hätte A unereichbare Zustände, so folgt $|\Sigma^*/_{\sim_A}| < |Q| < |\Sigma/_{\sim_L}|$...

□

4.11. Satz von Myhill und Nerode

Für eine Sprache L über einem Alphabet Σ sind die folgenden Aussagen äquivavlent:

(i) L ist regulär

(ii) Der Index von $\sim_L$ ist endlich

(iii) L ist die Vereinigung von Äquivavlenzklassen einer Rechtskongruenz mit endlichem Index

Beweis. (i) $\Leftrightarrow$ (iii) ist die Aussage von vorigem Korollar
Die Relation $\sim_L$ ist eine Rechtskongruenz und es gilt $L = \bigcup_{w \in L} [w]_{sim_L}$. Somit folgt (ii)$\Rightarrow$ (iii).
Die Implikation (iii) $\Rightarrow$ (ii) folgt.

4.12. Pumping-Lemma

Wir wollen nun ein Kriterium beschreiben das hilft nicht reguläre Sprachen zu erkennen

Sei Σ ein Alphabet. Für jede reguläre Sprache $L \subseteq \Sigma^*$ gibt es eine Konstante $k \in \mathbb{N}$, so dass folgendes gilt:
Ist $z \in L$ mit $|z| \geq k$, so gibt es Wörter $u, v, w \in \Sigma^*$ mit $z = uvw$, so dass folgendes gilt:

(i) $v \neq \lambda$

(ii) $|uv| \leq k$

(iii) $uv^i w \in L \ \forall i \in \mathbb{N}_0$

□

Beweis Sei $L \subseteq \Sigma^*$ eine reguläre Sprache und A ein DEA mit L(A)=L und erweitereter Übergangsfunktion $\delta^* : Q \times \Sigma^* \to Q$.
Sei $k := |Q|$.
Sei $z \in L$ mit $|z| \geq k$. (Falls kein solches z existiert ist nichts zu zeigen)
Die Funktion $f : \{0, \ldots, k\} \to Q$, $i \to \delta^*(s, z(1), \ldots, z(i))$ ist keine Injektion, denn es gilt $|\{0, \ldots, k\}| = k + 1 > |Q|$. Seien $j_1, j_2 \in \{0, \ldots, k\}$ mit $j_1 < j_2$ und $f(j_1) = f(j_2)$.
Sei $u := z(1) \ldots z(j_1), v = z(j_1 + 1) \ldots z(j_2), w = z(j_2 + 1) \ldots z(|z|)$.
Dann gilt $z = uvw$.
Aus $j_1 < j_2$ folgt $v \neq \lambda$.
Aus $j_2 \leq k$ folgt $|uv| \leq k$.
Es bleibt zu zeigen, dass $uv^i w \in L$ $\forall i \in \mathbb{N}_0$ gilt.
Dafür genügt es zu zeigen $\delta^*(s, uv^i) = \delta^*(s, u)$.
Denn dann gilt

$$\delta^*(s, uv^i w) = \delta^*(\delta^*(s, uv^i), w) = \delta^*(\delta^*(s, u), w) = \delta^*(\delta^*(s, uv)w) = \delta^*(s, uvw)$$

und damit $uv^i w \in L$.
Wir zeigen $\delta^*(s, uv^i) = \delta^*(s, u)$ mittels vollständiger Induktion über i.

i=0
Gelte nun $\delta^*(s, uv^{i-1}) = \delta^*(s, u)$ für ein $i \in \mathbb{N}$
Wieder folgt

$$\delta^*(s, uv^i) = \delta^*(\delta^*(s, uv^{i-1}), v) = \delta^*(\delta^*(s, u), v) = \delta^*(s, uv) = f(j_2) = f(j_1) = \delta^*(s, u)$$

Beispiel Die Sprache $L = \{0^n 1^n : U \in \mathbb{N}_0\}$ ist nicht regulär Dies lässt sich mit dem Pumping-Lemma wie folgt zeigen.

Beweis. $\Rightarrow \exists k \in \mathbb{N}_0 : \forall z \in L$ mit $|z| \geq k$ gilt, $\exists u, v, w \in \Sigma^*$ mit $z = uvw$ und (i)(ii)(iii) aus Pumping.
Sei $z := 0^k 1^k$.
Aus (i) und (ii) folgt, dass $v = 0^l$ für $l > 0$ und damit folgt
$uw = 0^{k-l} 1^k \in L$ nach Pumping-Lemma. Dies ist ein Widerspruch das $L = \{0^n 1^n : u \in \mathbb{N}_0\}$ $\square$

5. Formale Grammatiken

Idee: Konstruktion aller Wörter einer Sprache.

Beispiel: $L = \{0^n\}_{n\in\mathbb{N}_0}$
S Startsymbol
$S \to X, S \to 0S$ Regel $S \to 0S \to 00S \to 000S \to 000$

5.1. Grammatiken

Eine Grammatik ist ien Tupel $G = (N, T, P, S)$. Dabei ist

- N das Alphabet der Nichtterminalsymbole/Variablen
- T das Alphabet der Terminalsymbole mit $N \cup T = \varnothing$
- $P \subseteq ((n \cup T)^* \backslash T^*) \times (N \cup T)^*$ eine endliche Menge von Regeln
- $S \in \mathbb{N}$ das Startsymbol

EIne Satzform von G ist ein Wort $s \in (N \cup T)^*$ und eine Terminalwort von G ist ein Wort $t \in T^*$.

5.2. Ableitung

Sei $G = (N, T, P, S)$ eine Grammatik. Eine Satzform w' von G ist in einem Schritt aus einer Satzform w von G ableitbar, wenn es Satzformen u,v,x,y von G gibt, so dass $w = xuy, u \to v \in P$ und $w' = xvy$ gelten. Es bezeichne $\to_G$ die Relation auf der Menge der Satzformen von G, sodass $w \to_G w'$ genau dann für Satzformen von G gilt, wenn w' aus w in einem Schritt ableitbar ist.
Für Satzformen u,v von G ist eine Ableitung von v aus u eine Folge $u = w_1, \ldots, w_n = v$ mit $w_i \to_G w_{i+1}\ \forall i \in [n-1]$ und eine Ableitung von v in G ist eine Ableitung von v aus S in G. Für $n \in \mathbb{N}$ schreiben wir $u \to_G^n v$ wenn es eine Ableitung von v aus u der Länge n gibt und wir schreiben $u \to_G^* v$ wenn eine Ableitung von v aus u in G existiert.

5.3. Erzeugte Sprache

Sei $G = N, T, P, S$ eine Grammatik. Die von G erzeugte Sprache $L(G)$ ist die Menge aller Wörter $w \in T^*$ für die es eine Ableitung von w in G gibt.

Lemma Sei $G = (N, T, P, S)$ eine Grammatik und seien u,v,x,y Satzformen von G und seien $n, m \in \mathbb{N}$ mit $u \to_G^n v$ und $w \to_u^n xuy$.
Dann gilt $w \to_u^{m+n-1} xvy$.

Beweis Sei $\alpha_1, \ldots, \alpha_n$ ein Ableitung von xuy aus w und $\beta_1, \ldots, \beta_m$ eine Ableitung von v aus u. Dann ist
$\alpha_1, \ldots, \alpha_{n-1}, x\beta_1 y, \ldots, x\beta_m y = xvy$
eine Ableitung von xvy aus w in G der Länge n+m-1.

Im folgenden beschäftigen wir uns mit dem Thema welche Sprache Grammatiken verschiedener Komplexitätsstufen erzeugen können.

Satz Eine Sprache ist genau dann rekuriv aufzählbar, wenn sie von einer Grammatik erzeugt wird.

Beweisidee Wird eine Sprache L von einer Grammatik erzeugt, so ist L die erkannte Sprache einer TM, die in geeigneter Weise Ableitungen von G erzeugt, prüft ob diese Ableitung dem Wort der Eingabe entspricht und gegebenfalss akzeptiert. Wenn eine Ableitung der Eingabe gefunden ist.

Gegebn eine rekursiv aufzählbare Sprace L und eine TM, die L erkennt. So konstruieren wir ähnlich dem Postschen Korrespondenzproblems Regeln und Symobole, sodass wir die Arbeitsweise der TM modellieren können und entsprechend mit einem Terminalwort enden wenn dies von der TM erkannt wird.

5.4. Rechtslinear

Eine Grammatik $G = (N, T, P, S)$ ist rechtslinear, wenn alle Regeln von der Form

$$X \in uy \text{ oder } X \to u$$

mit $X, y \in \mathbb{N}$ und $u \in T^*$ sind.

Hier ist es sinnvoll endliche Automaten zu betrachten bei denen es nicht $\forall$ Zustände q und Eingabesymbole a ein Tripel (q, a, q') in der Übergangsrelation geben muss. Solche Automaten sind zwangsläufig nicht deterministisch.

Satz Eine Sprache ist genu dann regulär, wenn sie von einer rechtslinearen Grammatik erzeugt wird.

Beweisidee Zunächst überzeugt man sich davon, dass eine Sprache L genau dann von einer rechtlinearen Grammatik erzeugt wird, wenn sie von einer Grammatik $G = (N, T, P, S)$ erzeugt wird bei der alle Regeln von der Form $X \to ay$ oder $X \to \lambda$
mit $x, y \in \mathbb{N}$ und $a \in T$ sind. Eine Solche Grammatik wird als Grammatik in Simulationsform bezeichnet.

Die Sprache L die von einer rechtslinearen Grammatik (in Simulationsform) gebildet wird von dem EA

$$A = (N, T, \Delta, S, \{X \in N : X \to \lambda \in P\})$$

mit

$$\Delta = \{(X, a, y) \in N \times T \times N : X \to ay \in P\}$$

erkannt.
Umgekehrt ist es einfach zu sehen, dass jede Reguläre Sprace von einer rechtslinearen Grammatik erzeugt wird.