

Theoretisch Kurz (Theoretische Informatik in 10 Seiten)

Niko Efthymiou

November 17, 2008

Index

Def. 2.1	Alphabet	1
Def. 2.2	Wort	1
Def. 2.3	Sprache	1
Def. 2.4	Konkatenation	1
Def. 2.5	Präfix/Teilwort/Suffix	1
Def. 2.6	Sprachenoperationen	1
Def. 2.7	Relation	1
Def. 2.8	reflexiv	1
Def. 2.9	transitiv	1
Def. 2.10	symmetrisch	1
Def. 2.11	antisymmetrisch	1
Def. 2.12	total	1
Def. 2.13	Äquivalenzrelation	1
Def. 2.14	Äquivalenzklasse	1
Def. 2.15	Partielle Ordnung	1
Def. 2.16	Lineare Ordnung	1
Def. 2.17	Partielle Ordnung	1
Def. 2.18	Σ^* Induktiv	1
Def. 2.19	Wortlänge	1
Def. 2.20	Konkatenation	1
Lemma 2.21	Länge der Konkatenation	1
Def. 3.1	Reguläre Sprachen	2
Def. 3.2	Semantik regulärer Ausdrücke	2
Def. 3.3	Syntaktischer Zucker	2
Lemma 3.4	Äquivalenz von regulären Ausdrücken und Sprachen	2
Def. 3.5	Endlicher Automat	2
Satz 3.6	EA und regulären Sprachen	2
Def. 3.7	NEA	2
Satz 3.8	NEA zu EA	2
Def. 3.9	NEA mit ϵ -Übergängen	2
Satz 3.10	NEA+ ϵ zu EA	2
Satz 3.11	RA zu NEA+ ϵ	2
Satz 4.1	Pumping-Lemma	3
Satz 4.2	von Myhill und Nerode	3
Alg. 4.3	Markierungsalgorithmus	3
Alg. 4.4	Minimierungsalgorithmus	3
Satz 4.5	Abschluss unter booleschen Operationen	3
Def. 4.6	Homomorphismus	3

Satz 4.7	Abschluss unter homomorphismen	3
Def 1	Strings und Logik	3
Satz 4.8	von Büchi	4
Def 2	MSO-Formeln	4
Def. 4.9	Existenzielle mon. Logik 2. Stufe	4
Fol. 4.10	aus Büchi	4
Satz 4.11	Äquivalenzen für RA	4
Def. 5.1	Kontext frei Gramatik	4
Def 3	Ableitung	4
Def. 5.2	Begriffe	4
Def. 5.3	Kellerautomat	5
Lemma 5.4		5
Lemma 5.5		5
Satz 5.6	Leere Keller zu Akzeptierenden Zuständen	5
Satz 5.7	Akzeptierenden Zuständen zu Leere Keller	5
Satz 5.8	Grammatik zu Kellerautomat	5
Satz 5.9	Kellerautomat zu Grammatik	5
Satz 1	Äquivalenzen für kontextfreie Sprachen	5
Kor. 1	Kellerautomat mit einem Zustand	5
Def 4	Deterministischer Kellerautomat	5
Lemma 5.10	Regulär \subseteq deterministisch kontextfrei	5
Lemma 5.11	Regulär \subset det. kontextfrei \subset kontextfrei	5
Lemma 5.12	det. kontextfrei \Rightarrow eindeutige Grammatik	5
Def 5	erreichbar, erzeugend, nützlich	5
Def 6	Chomsky-Normalform	5
Satz 6.1	Kontextfrei \Rightarrow Chomsky-Sormalform	6
alg. 1	Chomsky	6
Def 7	Kreibach-Normalform	6
Satz 2		6
Satz 6.2	Pumping-Lemma 2	6
Prop. 6.3		6
Def 8	Substitution	6
Satz 6.4	Abgeschlossenheit unter Substitution	6
Satz 6.5	Weiter Abschlusseigenschaften	6
Satz 6.6	... und noch ein paar mehr	6
Satz 6.7	Nichtabgeschlossenheit	6
Satz 6.8	Schnitt mit regulären Sprachen	6
Satz 6.9	det. kontextfreie abg. unter Komplement	6
Satz 6.10	det. kontextfreie abg. unter Vereinigung und Schnitt	6
Def. 6.11	Lerheitstest, Wortproblem	6
Alg. 6.12	CYK-Algorithmus	6
Def. 6.13	Chomsky Grammatik	7
Def. 6.14	Syntaxanalyse	7
Def. 6.15	$LL(k)$ -Grammatik	7
Def 9	$FIRST/FOLLOW$	7
Lemma 6.16	$LL(1)$	7
Def. 6.17	$FIRST_k$	7
Def. 6.18	$LR(k)$ -Grammatik	7
Satz 6.19	det. kontextfrei $\Leftrightarrow LR(k)$	7
Def 10	Berechenbarkeit	7
Def. 7.1	Semantik von Turing-Maschinen	7
Def. 7.2	Weiter def.	7
Def. 7.3	LOOP-Programme	7
Def. 7.4	Semantik von LOOP-Programmen	8

Def. 7.5	WHILE-Programme	8
Def. 7.6	GOTO-Programme	8
Satz 7.7	WHILE \equiv GOTO	8
Lemma 7.8	k -String \equiv 1-String TM	8
Satz 7.9	WHILE \equiv TM	8
Satz 7.10	TM \equiv GOTO	8
Def. 7.11	Primitiv rekursive Fkt.	8
Def. 7.12	μ -rekursiv	8
Satz 7.13	Universelle TM	8
Def. 7.14	Partielle Funktionen	8
Def. 7.15	Entscheidbar	8
Def. 7.16	Charakteristische Funktion	8
Lemma 7.17	Charakteristische Funktion und entscheidbarkeit	8
Def. 7.18	Rekursiv aufzählbar	9
Lemma 7.19	Aufzählbar \equiv semi-entscheidbar	9
Def. 7.20	Spezielles Halteproblem	9
Satz 7.21		9
Def. 7.22	reduzierbar	9
Lemma 7.23	entscheidbarkeit und Reduktionen	9
Def. 7.24	Halteproblem	9
Satz 7.25	H und H_0 sind nicht entscheidbar	9
Satz 7.26	Rice	9
Def. 7.27	Postisches Korrespondenzproblem (PCP)	9
Satz 7.28	Unentscheidbarkeit von PCP	9
Def. 7.29	Schnittproblem	9
Def. 7.30	Eindeutigkeitsproblem	9
Satz 7.31		9
Def. 8.1	MinSpanningTree	9
Def. 8.2	TravelingSalesman	9
Def. 8.3	Schrittzahl	9
Def. 8.4	Zeitschranke	9
Lemma 8.5	alles das selbe	9
Def. 8.6	Rucksack	10
Def. 8.7	HamiltonKreis	10
Def. 8.8	SAT	10
Def. 8.9	Cligue	10
Lemma 8.10	NP-Probleme	10
Satz 8.11	$P \subseteq NP \subseteq EXP$	10
Def. 8.12		10
Satz 8.13	Cook	10
Satz 8.14		10
Lemma 8.15		10
Lemma 8.16		10
Lemma 8.17		10

2 Grundlegende Begriffe

Def. 2.1 (Alphabet) Ein Alphabet Σ ist eine endliche, nicht-leere Menge.

Def. 2.2 (Wort) Ein Wort w über einem Alphabet Σ ist eine endliche Folge $\sigma_1 \dots \sigma_n$ von Zeichen aus Σ

Def. 2.3 (Sprache) Eine Sprache über einem Alphabet Σ ist eine (endliche oder unendliche) Menge von Wörtern über Σ

Def. 2.4 (Konkatenation) Sind u und v Wörter, so bezeichnet man $u \circ v$ die Konkatenation von u und v , d.h. das Wort, das entsteht, wenn v hinter u geschrieben wird.

Def. 2.5 (Präfix/Teilwort/Suffix) Sind x, y, z Wörter und ist $w = xyz$, so heißt x ein Präfix von w , y ein Teilwort von w und z ein Suffix von w

Def. 2.6 (Sprachenoperationen) Seien $A, B \subseteq \Sigma^*$ Sprachen. Dann bezeichnet man: $A \cup B$ Vereinigung, $A \cap B$ Durchschnitt, $A - B$ Differenz, $\bar{A} = \Sigma^* - A$, $AB = \{uv | u \in A, v \in B\}$, $A^0 = \{\epsilon\}$, $A^{n+1} = A^n A$, $A^* = \bigcup_{n \geq 0} A^n$, $A^+ = \bigcup_{n \geq 1} A^n$

Def. 2.7 (Relation) Sei A Menge, A^n Menge der n -Tupel aus A . n -Stellige Relation R über A : Teilmenge von A^n

Def. 2.8 (reflexiv) $\forall x \in A$ gilt $(x, x) \in R$

Def. 2.9 (transitiv) $\forall x, y, z \in A$ gilt $(x, y) \in R$ und $(y, z) \in R$, so auch $(x, z) \in R$.

Def. 2.10 (symmetrisch) $\forall x, y \in A: (x, y) \in R \Rightarrow (y, x) \in R$

Def. 2.11 (antisymmetrisch) $\forall x, y \in A: (x, y) \in R \wedge (y, x) \in R \Rightarrow x = y$

Def. 2.12 (total) $\forall x, y \in A: (x, y) \in R \vee (y, x) \in R$

Def. 2.13 (Äquivalenzrelation) Reflexive, symmetrische und transitive Relation. Oft mit \sim dargestellt

Def. 2.14 (Äquivalenzklasse) Die Äquivalenzklasse $[x]$ von x ist die Menge aller y mit $x \sim y$

Def. 2.15 (Partielle Ordnung) Reflexive, antisymmetrische und transitive Relation.

Def. 2.16 (Lineare Ordnung) Reflexive, antisymmetrische, transitive und totale Relation.

Def. 2.17 (Partielle Ordnung) Wie im Wörterbuch

Def. 2.18 (Σ^* Induktiv) $\epsilon \in \Sigma^*$ und $w \in \Sigma^* \wedge \sigma \in \Sigma \Rightarrow w \cdot \sigma \in \Sigma^*$

Def. 2.19 (Wortlänge) $|\epsilon| = 0$ und $|w\sigma| = |w| + 1$

Def. 2.20 (Konkatenation) $u \circ \epsilon = u$ und $u \circ (w\sigma) = (u \circ w) \cdot \sigma$

Lemma 2.21 (Länge der Konkatenation) $|u \circ v| = |u| + |v|$

3 Reguläre Sprachen: Grundlagen

Def. 3.1 (Reguläre Sprachen) Sei Σ ein Alphabet. Die folgenden Regeln definieren die regulären Sprachen über Σ . \emptyset , $\{\epsilon\}$ und $\{\sigma\}$, $\forall \sigma \in \Sigma$ sind regulär. Sind A und B regulär, so auch AB , $A \cup B$ und A^* .

Def. 3.2 (Semantik regulärer Ausdrücke) Wir definieren für jeden regulären Ausdruck α eine Sprache $L(\alpha)$ wie folgt: $L(\emptyset) = \emptyset$, $L(\epsilon) = \{\epsilon\}$ und $L(\sigma) = \{\sigma\}$, $\forall \sigma \in \Sigma$ sind reguläre Ausdrücke. Sind α und β reguläre Ausdrücke, so auch $L(\alpha\beta) = L(\alpha)L(\beta)$, $L(\alpha + \beta) = L(\alpha) \cup L(\beta)$ und $L(\alpha^*) = L(\alpha)^*$.

Def. 3.3 (Syntaktischer Zucker) $[a - z] = a + \dots + z$, $\alpha^? = (\alpha + \epsilon)$, $\alpha^n = \alpha \dots \alpha$ (n mal α), $\alpha^{\{m,n\}} = \alpha^m(\alpha^?)^{\{n-m\}}$ min. m , max. n mal

Lemma 3.4 (Äquivalenz von regulären Ausdrücken und Sprachen) Für jeden regulären Ausdruck α ist $L(\alpha)$ regulär. Für jede reguläre Sprache L , existiert α mit $L = L(\alpha)$.

Def. 3.5 (Endlicher Automat) Ein Endlicher Automat \mathcal{A} besteht aus: Menge Q von Zuständen, Eingabealphabet Σ , Überföhrungsfunktion $\delta : Q \times \Sigma \rightarrow Q$, Startzustand s und Akzeptierenden Zuständen $F \subseteq Q$. Wir schreiben $\mathcal{A} = (Q, \Sigma, \delta, s, F)$.

- $\delta^* : Q \times \Sigma^* \rightarrow Q$
- $\delta^*(q, \epsilon) = q$
- $\delta^*(q, u\sigma) = \delta(\delta^*(q, u), \sigma)$, $u \in \Sigma^*, \sigma \in \Sigma$

Satz 3.6 (EA und reguläre Sprachen) Für jeden endlichen Automaten \mathcal{A} ist $L(\mathcal{A})$ regulär.

Def. 3.7 (NEA) Ein nicht deterministischer endlicher Automat $\mathcal{A} = (Q, \Sigma, \delta, s, F)$ besteht aus Menge Q von Zuständen, Eingabealphabet Σ , Überföhrungsfunktion $\delta : Q \times \Sigma \rightarrow \mathcal{P}(Q)$, Startzustand s und Akzeptierenden Zuständen $F \subseteq Q$.

- $\delta^* : Q \times \Sigma^* \rightarrow \mathcal{P}(Q)$
- $\delta^*(q, \epsilon) = \{q\}$
- $\delta^*(q, u\sigma) = \bigcup_{p \in \delta^*(q, u)} \delta(p, \sigma)$

Satz 3.8 (NEA zu EA) Zu jedem nichtdeterministischen endlichen Automaten \mathcal{A} gibt es einen deterministischen endlichen Automaten \mathcal{A}' mit $L(\mathcal{A}') = L(\mathcal{A})$.

Def. 3.9 (NEA mit ϵ -Übergängen) Ein nicht deterministischer endlicher Automat $\mathcal{A} = (Q, \Sigma, \delta, s, F)$ mit ϵ -Übergängen besteht aus einer Zustandsmenge Q , einem Eingabealphabet Σ , einem Anfangszustand $s \in Q$, einer Akzeptierenden Menge F von Zuständen, sowie einer Überföhrungsfunktion $\delta : Q \times (\Sigma \cup \{\epsilon\}) \rightarrow \mathcal{P}(Q)$.

- $\delta^*(q, \epsilon) = \epsilon\text{-closure}(q)$
- $\delta^*(q, u\sigma) = \bigcup_{q' \in \delta^*(q, u)} \epsilon\text{-closure}(\delta(q', \sigma))$

Satz 3.10 (NEA+ ϵ zu EA) Zu jedem nichtdeterministischen Automaten \mathcal{A} mit ϵ -Übergängen gibt es einen deterministischen endlichen Automaten \mathcal{A}' mit $L(\mathcal{A}) = L(\mathcal{A}')$.

Satz 3.11 (RA zu NEA+ ϵ) Zu jedem regulären Ausdruck α gibt es einen nichtdeterministischen Automaten \mathcal{A} mit ϵ -Übergängen, so dass $L(\alpha) = L(\mathcal{A})$ gilt.

4 Reguläre Sprachen: Eigenschaften

Satz 4.1 (Pumping-Lemma) Sei L regulär. Dann gibt es ein n , so dass jeder String $w \in L$ mit $|w| \leq n$ als $w = xyz$ geschrieben werden kann, so dass die folgenden Aussagen gelten.

- $y \neq \epsilon$
- $|xy| \leq n$
- für alle $k \geq 0$ ist $xy^kz \in L$

Anwendung: Angenommen, L regulär. Sei n die Zahl aus dem Pumping-Lema. Wähle $w \in L$ geschickt (abhängig von n) so, dass $|w| \geq n$. Sein x, y, z wie im Pumping-Lema, dann wissen wir das die Bedingungen erfüllt sind. Zeige, dass, egal wie w zerlegt wurde, sich ein Widerspruch ergibt.

Satz 4.2 (von Myhill und Nerode) Eine Sprache L ist regulär, genau dann, wenn \sim_L endlich viele Äquivalenzklassen hat.

Alg. 4.3 (Markierungsalgorithmus) Eingabe: $\mathcal{A} = (Q, \Sigma, \delta, s, F)$ Ausgabe: Relation N

1. Markiere alle Paare $(p, q), (q, p)$ mit $p \in F, q \notin F$
2. Für jedes unmarkierte Paar (p, q) und jedes $\sigma \in \Sigma$ markiere (p, q) , falls $\forall \sigma \in \Sigma, (\delta(p, \sigma), \delta(q, \sigma))$ markiert ist.
3. Wiederhole 2. bis sich keine neuen markierten Paare ergeben
4. $N =$ Menge aller markierten Paare

Aufwand $O(|Q|^2|\Sigma|)$

Alg. 4.4 (Minimierungsalgorithmus) Eingabe: $\mathcal{A} = (Q, \Sigma, \delta, s, F)$ Ausgabe: minimierter Automat \mathcal{A}' mit $L(\mathcal{A}) = L(\mathcal{A}')$

1. Entferne alle Zustände von \mathcal{A} , die von s nicht erreichbar sind.
2. berechne die Relation N mit dem Markierungsalgorithmus.
3. Verschmelze sukzessive alle nicht markierten Zustandspaare zu jeweils einem Zustand

Aufwand ist $O(|Q|^2|\Sigma|)$

Satz 4.5 (Abschluss unter booleschen Operationen) Sind L und L' reguläre Sprachen, so sind $L \cap L', \Sigma^* - L$ und $L + L'$ regulär

Def. 4.6 (Homomorphismus) Seien Σ, Γ Alphabete und $h : \Sigma \rightarrow \Gamma^*$. So heist h Homomorphismus. h lässt sich auf Σ^* erweitern: $h(\epsilon) = \epsilon$ und $h(u\sigma) = h(u)h(\sigma)$. $h(uv) = h(u)h(v)$ nennt man die Homomorphismus eigenschaft

Satz 4.7 (Abschluss unter homomorphismen) Ist L reguläre Sprache über Σ und Γ ein Alphabet, so sind $h(L)$ und $h^{-1}(L)$, für jeden Homomorphismus $h : \Gamma^* \rightarrow \Sigma^*$ regulär

Def (Strings und Logik) Sei $w = \sigma_1 \dots \sigma_n$ Wort über $\Sigma = \{\tau_1, \dots, \tau_n\}$. Die Struktur $S_w = (\{1, \dots, n\}, succ, \leq, P_{\tau_1}, \dots, P_{\tau_n})$ ist definiert durch: $succ = \{(i, i+1) | 1 \leq i \leq n\}$, $\leq = \{(i, j) | 1 \leq i \leq j \leq n\}$, $\forall i : P_{\tau_i} = \{j | \sigma_j = \tau_i\}$.

Variablen x_i , deren Elemente aus der Grunmenge zugeordnet werden (Positionen in w). $S_w \models \phi$ falls ϕ in S_w gilt.

Satz 4.8 (von Büchi) Eine Sprache L ist regulär, genau dann, wenn es eine Formel ϕ der Monadischen Logik erster Stufe gibt, so dass für alle nicht-leeren Strings w gilt: $w \in L \Leftrightarrow S_w \models \phi$.

Insbesondere: sei $\mathcal{A} = (Q, \Sigma, \delta, s, F)$ EA für L , $Q = q_1, \dots, q_n$ und $s = q_1$, dann beschreibt folgende Formel alle Strings aus L :

$$\begin{aligned} \phi = & \exists X_1 \dots \exists X_k \\ & \forall x \bigvee_{i=1}^k X_i(x) \wedge \bigwedge_{i \neq j} \neg (X_i(x) \wedge X_j(x)) \wedge \\ & \exists x \phi_{first}(x) \wedge \bigvee_{\substack{i, \sigma \\ \delta(s, \sigma) = q_i}} (X_i(x \wedge P\sigma(x)) \wedge \\ & \exists x \phi_{last}(x) \wedge \bigvee_{q_i \in F} X_i(x) \wedge \\ & \forall x \forall y \text{ succ}(x, y) \rightarrow \bigvee_{\substack{i, j, \sigma \\ \delta(q_i, \sigma) = q_j}} X_i(x) \wedge X_j(y) \wedge P\sigma(y) \end{aligned}$$

Def (MSO-Formeln) $P_\sigma(x), X(x), x \leq y, x = y$ sind MSO-Formeln, $\forall \sigma, x, y, X$. Sind ϕ, ψ MSO-Formeln, so auch $\phi \wedge \psi, \neg \phi, \exists x \phi$ und $\exists X \phi$

Def. 4.9 (Existenzielle mon. Logik 2. Stufe) Die existenzielle monadische Logik zweiter Stufe besteht aus allen MSO-Formeln der Form $\exists X_1 \dots \exists X_k \psi$, wobei ψ Formel der 1. Stufe.

Fol. 4.10 (aus Büchi) Zu jeder MSO-Formel ϕ (ohne frei Variablen) über Σ -Strings, gibt es eine existenzielle MSO-Formel ϕ' , so dass für alle Σ -Strings w gilt: $w \models \phi \Leftrightarrow w \models \phi'$

Satz 4.11 (Äquivalenzen für RA) Es gelten folgende Äquivalenzen:

Assoziativität $\alpha + (\beta + \gamma) \equiv (\alpha + \beta) + \gamma$ bzw. $\alpha(\beta\gamma) \equiv (\alpha\beta)\gamma$

Kommutativität $\alpha + \beta \equiv \beta + \alpha$

Neutrale Elemente $\emptyset + \alpha \equiv \alpha \equiv \alpha + \emptyset$ bzw. $\epsilon\alpha \equiv \alpha \equiv \alpha\epsilon$

Nullelement $\emptyset\alpha \equiv \emptyset \equiv \alpha\emptyset$

Distributivität $\alpha(\beta + \gamma) \equiv \alpha\beta + \alpha\gamma$ bzw. $(\alpha + \beta)\gamma \equiv \alpha\gamma + \beta\gamma$

Stern $(\alpha^*)^* \equiv \alpha^*, \emptyset^* = \epsilon$ und $\epsilon^* = \epsilon$

5 Kontextfreie Sprachen: Grundlagen

Def. 5.1 (Kontext frei Grammatik) Eine kontextfreie Grammatik $G = (V, \Sigma, S, P)$ besteht aus einer Menge von Variablen V , einem Alphabet Σ , einem Startsymbol $S \in V$ und einer Menge von Regeln $P \subseteq V \times (V \cup \Sigma)^*$. Die Sprache von G ist dann $L(G) = \{w \in \Sigma^* \mid S \Rightarrow_G^* w\}$

Def (Ableitung) Falls, $\alpha, \beta, \gamma \in (V \cup \Sigma)^*, X \in V$ und $X \rightarrow \gamma$ in P , so gilt $\alpha X \beta \rightarrow_G \alpha \beta \gamma$

Def. 5.2 (Begriffe) Linksableitung Ableitung, in der in jedem Schritt die am weitesten links stehende Variable ersetzt wird.

Rechtsableitung analog

Satzform String $w \in (V \cup \Sigma)^*$ mit $S \Rightarrow^* w$

Links-Satzform String $w \in (V \cup \Sigma)^*$ mit $S \Rightarrow_l^* w$

Rechts-Satzform String $w \in (V \cup \Sigma)^*$ mit $S \Rightarrow_r^* w$

Def. 5.3 (Kellerautomat) Ein Kellerautomat besteht aus einer Zustandsmenge Q , einem Eingabealphabet Σ , einem Kelleralphabet Γ , einer Überföhrungsfunktion $\delta : Q \times (\Sigma \cup \{\epsilon\}) \times \Gamma \rightarrow \mathcal{P}(Q \times \Gamma^*)$, einem Startzustand s , einem untersten kellersymbol τ_0 und einer Menge von akzeptierenden Zuständen F . $\mathcal{A} = (Q, \Sigma, \Gamma, \delta, s, \tau_0, F)$

Lemma 5.4 Sei $\mathcal{A} = (Q, \Sigma, \Gamma, \delta, s, \tau_0, F)$ ein Kellerautomat. Sein $x, y, w \in \Sigma^*, \alpha, \beta, \gamma \in \Gamma^*, p, q \in Q$, dann gilt: $(p, x, a) \vdash^* (p, y, \beta) \Rightarrow (p, xw, \alpha\gamma) \vdash^* (q, yw, \beta\gamma)$

Lemma 5.5 Sei $\mathcal{A} = (Q, \Sigma, \Gamma, \delta, s, \tau_0, F)$ ein Kellerautomat. Sein $x, y, w \in \Sigma^*, \alpha, \beta \in \Gamma^*, p, q \in Q$, dann gilt: $(p, xw, a) \vdash^* (p, yw, \beta) \Rightarrow (p, x, \alpha) \vdash^* (q, y, \beta)$

Satz 5.6 (Leere Keller zu Akzeptierenden Zuständen) Für jeden Kellerautomaten \mathcal{A} , der mit leerem Keller akzeptiert gibt es einen Kellerautomaten \mathcal{B} der mit akzeptierenden Zuständen akzeptiert und $L(\mathcal{A}) = L(\mathcal{B})$

Satz 5.7 (Akzeptierenden Zuständen zu Leere Keller) Für jeden Kellerautomaten \mathcal{A} , der mit akzeptierenden Zuständen akzeptiert, gibt es einen Kellerautomaten \mathcal{B} der mit Leere Keller akzeptiert und $L(\mathcal{A}) = L(\mathcal{B})$

Satz 5.8 (Grammatik zu Kellerautomat) Zu jeder Grammatik G gibt es einen Kellerautomaten \mathcal{A} mit $L(\mathcal{A}) = L(G)$

Satz 5.9 (Kellerautomat zu Grammatik) Zu jedem Kellerautomaten \mathcal{A} , der mit leerem Keller akzeptiert, gibt es eine Grammatik G mit $L(G) = L(\mathcal{A})$

Satz (Äquivalenzen für kontextfreie Sprachen) Für eine Sprache L sind äquivalent: L ist kontextfrei, L wird von einem Kellerautomaten mit akzeptierenden Zuständen erkannt und L wird von einem Kellerautomaten mit leerem Keller erkannt.

Kor. (Kellerautomat mit einem Zustand) Zu jedem Kellerautomaten gibt es einen äquivalenten Kellerautomaten mit nur einem Zustand

Def (Deterministischer Kellerautomat) Ein Kellerautomat heißt deterministisch, falls $\forall q \in Q, \sigma \in \Sigma, \tau \in \Gamma$ gilt: $|\delta(q, \sigma, \tau)| \leq 1$, $|\delta(q, \epsilon, \tau)| \leq 1$ und $\delta(q, \sigma, \tau) \neq \emptyset \Leftrightarrow \delta(q, \epsilon, \tau) = \emptyset$. Eine Sprache L heißt deterministisch kontextfrei, falls es einen deterministischen Kellerautomaten \mathcal{A} gibt mit $L(\mathcal{A}) = L$.

Lemma 5.10 (Regulär \subseteq deterministisch kontextfrei) Jede reguläre Sprache ist deterministisch kontextfrei.

Lemma 5.11 (Regulär \subset det. kontextfrei \subset kontextfrei) Es gibt deterministisch kontextfreie Sprachen, die nicht regulär sind. Es gibt kontextfreie Sprachen, die nicht deterministisch kontextfrei sind.

Lemma 5.12 (det. kontextfrei \Rightarrow eindeutige Grammatik) Ist L det. kontextfrei, so gibt es eine eindeutige Grammatik für L

6 Kontextfreie Sprachen: Eigenschaften

Def (erreichbar, erzeugend, nützlich) Eine Variable X einer Grammatik $G = (V, \Sigma, S, P)$ heißt: erreichbar, falls es eine Ableitung $S \Rightarrow_G^* \alpha X \beta$ gibt. erzeugend, falls es eine Ableitung $X \Rightarrow_G^* w$ mit $w \in \Sigma^*$ gibt. nützlich, falls es eine Ableitung $S \Rightarrow_G^* \alpha X \beta \Rightarrow_g^* w$ mit $w \in \Sigma^*$ gibt.

Def (Chomsky-Normalform) $G = (V, \Sigma, S, P)$ ist in Chomsky-Normalform wenn G nur nützliche Variablen enthält und alle Regeln von G von der Form $X \rightarrow YZ$ und $X \rightarrow \sigma$ enthält mit $X, Y, Z \in V, \sigma \in \Sigma$. Falls S in keiner rechten Regelseite vorkommt, ist zusätzlich $S \rightarrow \epsilon$ erlaubt.

Satz 6.1 (Kontextfrei \Rightarrow Chomsky-Normalform) Ist L kontextfrei, so gibt es eine Grammatik G in Chomsky-Normalform, so dass $L(G) = L$.

alg. (Chomsky) Sei $G_0 = (V, \Sigma, S, P)$ Grammatik für L

1. Entfernen von nutzlosen Variablen
2. Umwandeln aller Regeln, deren rechte Seite sowohl Variablen als auch Terminalsymbole enthalten
3. Ersetzen aller Regeln $X \rightarrow \beta$ mit $|\beta| > 2$
4. Ersetzen aller ϵ -Regeln
5. Ersetzen aller Regeln der Art $X \rightarrow Y$
6. Falls $\epsilon \in L$ füge neues Startsymbol $S' \rightarrow S|\epsilon$ hinzu

Def (Kreibach-Normalform) $G = (V, \Sigma, S, P)$ ist in Greibach-Normalform wenn: G nur nützliche Variablen enthält, und alle Regeln von G von der Form $X \rightarrow \sigma Y_1 \dots Y_k$ mit $X, Y_1, \dots, Y_k \in V, \sigma \in \Sigma$ sind. Falls S in keiner rechten Regelseite vorkommt ist auch $S \rightarrow \epsilon$ erlaubt

Satz Ist L kontextfrei, so gibt es eine Grammatik G in Greibach-Normalform, so dass $L(G) = L$.

Satz 6.2 (Pumping-Lemma 2) Ist L kontextfrei, so gibt es ein $n \in \mathbb{N}$, so dass sich jeder String $z \in L$ so in $z = uvwxy$ zerlegen lässt, dass (1) $|vx| \geq 1$, (2) $|vwx| \leq n$ und (3) $uv^iwx^iy \in L, \forall i \geq 0$

Prop. 6.3 Die folgenden Sprachen sind nicht kontextfrei: $L_{abc} = \{a^n b^n c^n | n \geq 1\}$ und $L_Q = \{0^{n^2} | n > 0\}$

Def (Substitution) Eine Substitution s ist eine Abbildung, die jedem Symbol $\sigma \in \Sigma$ eine Sprache $s(\sigma)$ zuordnet. Fortgesetzt auf Strings $s(\sigma_1 \dots \sigma_n) = s(\sigma_1) \dots s(\sigma_n)$. Fortgesetzt auf Sprachen via $s(L) = \bigcup_{w \in L} s(w)$

Satz 6.4 (Abgeschlossenheit unter Substitution) Ist $L \subseteq \Sigma^*$ kontextfrei und ist $\forall \sigma \in \Sigma, s(\sigma)$ kontextfrei, dann ist auch $s(L)$ kontextfrei.

Satz 6.5 (Weiter Abschlusseigenschaften) Die Klasse der kontextfreien Sprachen ist abgeschlossen unter Vereinigung, Konkatenation, dem $*$ -Operator, dem $+$ -Operator und Homomorphismen

Satz 6.6 (... und noch ein paar mehr) Ist L kontextfrei, so auch $L' = \{reverse(w) | w \in L\}$, als auch $L'' = h^{-1}(L)$ für Homomorphismus h .

Satz 6.7 (Nichtabgeschlossenheit) Die Klasse der kontextfreien Sprachen ist *nicht* unter Durchschnitt, Komplement, Mengendifferenz abgeschlossen.

Satz 6.8 (Schnitt mit regulären Sprachen) Ist L_1 kontextfrei und L_2 regulär so ist $L_1 \cap L_2$ kontextfrei.

Satz 6.9 (det. kontextfreie abg. unter Komplement) Die Klasse der deterministisch kontextfreien Sprachen ist abgeschlossen unter Komplementbildung.

Satz 6.10 (det. kontextfreie abg. unter Vereinigung und Schnitt) Die Klasse der deterministisch kontextfreien Sprachen ist abgeschlossen unter Vereinigung und Durchschnitt.

Def. 6.11 (Lerheitstest, Wortproblem) Gegeben G , ist $L(G) \neq \emptyset$?
Sei L kontextfreie Sprache. Gegeben $w \in \Sigma^*$, ist $w \in L$?

Alg. 6.12 (CYK-Algorithmus) Sei Grammatik G für L in Chomsky Normalform.

1. Für $w = a_1 \dots a_n \in \Sigma^*$ sei $w[i, j] = w_i \dots w_j$

2. Für jede Wahl von $i, j, 1 \leq i \leq j \leq n$ sei $V_{i,j} = \{X \in V \mid X \Rightarrow^* w[i, j]\}$. Da G in CNF ist, ist X genau dann in $V_{i,j}$, falls $\exists Y, Z, k$ mit $X \rightarrow YZ \in G$, $Y \in V_{i,k}$ und $Z \in V_{k+1,j}$.

Def. 6.13 (Chomsky Grammatik) Eine Chomsky-Grammatik ist ein 4-Tupel (V, Σ, P, S) mit V, Σ, S wie zuvor und $P \subseteq (V \cup \Sigma)^* V (V \cup \Sigma)^* \times (V \cup \Sigma)^*$

Def. 6.14 (Syntaxanalyse) Gegeben $w \in \Sigma^*$, ist ein Ableitungsbaum für w gesucht, falls $w \in L(G)$, sonst hinweis warum $w \notin L(G)$.

Def. 6.15 ($LL(k)$ -Grammatik) Sei $k > 0$. Eine Grammatik G heißt $LL(k)$ -Grammatik, falls für $u, x, y \in \Sigma^*, X \in V$ und $\alpha, \beta, \gamma \in (\Sigma \cup V)^*$, $S \Rightarrow_l^* uX\alpha \Rightarrow u\beta\alpha \Rightarrow_l^* ux$, $S \Rightarrow_l^* uX\alpha \Rightarrow u\gamma\alpha \Rightarrow_l^* uy$ und $p_k(x) = p_k(y) \Rightarrow \beta = \gamma$ gelten.

Def ($FIRST/FOLLOW$) $FIRST(\alpha) = \{\sigma \in \Sigma \mid \alpha \Rightarrow^* \sigma v, v \in \Sigma^*\} \cup \{\epsilon \mid \alpha \Rightarrow^* \epsilon\}$. $FOLLOW(X) = \{\sigma \in \Sigma \mid S \Rightarrow^* uX\sigma v, u, v \in \Sigma^*\}$.

Lemma 6.16 ($LL(1)$) Eine kontextfrei Grammatik G ohne nutzlose Variablen ist genau dann eine $LL(1)$ -Grammatik, wenn für alle variablen X von G und alle Strings $\alpha \neq \beta$, für die $X \rightarrow \alpha$ und $X \rightarrow \beta$ produktionen sind, gilt $FIRST(\alpha) \cap FIRST(\beta) = \emptyset$, falls $\alpha \Rightarrow^* \epsilon$ so ist $FOLLOW(X) \cap FIRST(\beta) = \emptyset$

Def. 6.17 ($FIRST_k$) Für $\alpha \in (\Sigma \cup V)^*$ sei $FIRST_k(\alpha) = \{p_k(x) \mid \alpha \Rightarrow^* x \in \Sigma^*\}$.

Def. 6.18 ($LR(k)$ -Grammatik) Sei $k > 0$. Ein Grammatik G heißt $LR(k)$ -grammatik, falls für $X, Y \in V, uxy \in \Sigma^*$ und $\alpha, \beta, \gamma \in (\Sigma \cup V)^*$, $S \Rightarrow_r^* \alpha Xu \Rightarrow_r \alpha \beta u$, $S \Rightarrow_r^* \gamma Yx \Rightarrow_r \alpha \beta y$ und $FIRST_k(u) = FIRST_k(y) \Rightarrow \alpha = \gamma, X = Y, x = y$

Satz 6.19 (det. kontextfrei $\Leftrightarrow LR(k)$) Sei $k > 1$ L deterministisch kontextfrei $\Leftrightarrow L = L(G)$ für eine $LR(k)$ -Grammatik G

7 Berechenbarkeit

Def (Berechenbarkeit) $f : \Sigma^* \rightarrow \Sigma^*$ berechenbar \Leftrightarrow es giebt eine Algorithmus, der bei Eingabe $w \in \Sigma^*$ die Ausgabe $f(w)$ erzeugt

Def. 7.1 (Semantik von Turing-Maschinen) Eine Konfiguration von M ist ein Tupel (q, u, σ, v) mit $q \in Q \cup \{ja, nein, h\}$, u der string links vom Kopf, σ das zeichen unter dem Kopf und v der String rechts vom Kopf.

Ist $K = (q, u, \sigma, v)$ ein Konfiguration, so ist $K' = (q', u', \sigma', v')$ die Nachfolgekongfiguration von K ($K \vdash_M K'$) falls $\delta(q, \sigma) = (q', \tau, d)$.

Def. 7.2 (Weiter def.) Sei $\Sigma \subseteq \Gamma - \{\triangleright, \sqcup\}$ ein Ein-/Ausgabe-Alphabet

M akzeptiert w , falls $K_0(w) \vdash_M^* (ja, u, \sigma, v)$ für irgentwelche $u, v \in \Gamma^*$ und $\sigma \in \Gamma$

M lehnt w ab, falls $K_0(w) \vdash_M^* (nein, u, \sigma, v)$ für irgentwelche $u, v \in \Gamma^*$ und $\sigma \in \Gamma$

$f_M(w) = u \in \Sigma^*$, falls $K_0(w) \vdash_M^* (h, \epsilon, \triangleright, u)$ oder $K_0(w) \vdash_M^* (h, \epsilon, \triangleright, u \sqcup v)$

Ene Funktion $f : \Sigma^* \rightarrow \Sigma^*$ heißt Turing-berechenbar, falls $f = f_M$ für eine Turing-Mashine M gilt.

Def. 7.3 (LOOP-Programme) Ein LOOP-Program besteht aus Variablen x_1, x_2, x_3, \dots , Konstanten $0, 1, 2, \dots$, Trennsymbolen $:=$, Operatoren $+-$ und denn Schlüsselwortern LOOP, DO, END. LOOP-Programme sind: $x_j := x_j + c$ und $x_i := x_j - c$ sind, $P_1; P_2$ für LOOP-Programme P_1, P_2 , LOOP x_i DO P END.

Def. 7.4 (Semantik von LOOP-Programmen) Eine Folge von $S = s_1, s_2, \dots$ von natürlichen Zahlen, wobei nur endlich viele $s_i \neq 0$ heist Speicherinhalt. Ist S ein Speicherinhalt und P ein LOOP-Programm, so ist $S' = P(S)$ der Speicherinhalt nach Bearbeitung von P . Der Speicherinhalt von $S(x)$ bei Eingabe von x ist die Folge $x, 0, 0, \dots, f_P = s'_1$, wobei $S' = P(S(x))$, ist die von P berechnete Funktion. $F : \mathbb{N} \rightarrow \mathbb{N}$ heist LOOP-Berechenbar falls $\exists P$ mit $f_P = f$

Def. 7.5 (WHILE-Programme) Erweiterung von LOOP-Programmen durch das Schlüsselwort WHILE, ergibt WHILE-Programme. Bedingte wiederholung: Ist P ein WHILE-Programm, so auch $WHILE\ x_i \neq 0\ DO\ P\ END$. Ist P' von der Form $WHILE\ x_i \neq 0\ DO\ P\ END$ und $S = s_1, s_2; \dots$ ein Speicherinhalt, so ist $P'(S) = S$ falls $s_i = 0$, $P'(P(S))$ sonst. $F : \mathbb{N} \rightarrow \mathbb{N}$ ist WHILE-Berechenbar, falls $f = f_P$

Def. 7.6 (GOTO-Programme) Ein GOTO-Programm besteht aus einer Folge $M_1 : A_1; \dots; M_k : A_k$ von Anweisungen A_i mit Marken M_i . Anweisung kann $x_i := x_j + c$ oder $x_i := x_j - c$, $IF\ x_i = c\ THEN\ GOTO\ M_j$ und $HALT$ sein. (l, S) ist eine Konfiguration, wobei M_l Marke und S ein Speicherinhalt ist. Die Nachfolge-Konfiguration (l', S') einer Konfiguration (l, S) ist ...

Satz 7.7 (WHILE \equiv GOTO) Jede WHILE-Berechenbare Funktion ist auch GOTO-Berechenbar

Lemma 7.8 (k -String \equiv 1-String TM) Zu jeder k -String TM M giebt es eine 1-String TM M' , so dass $\forall w \in \Sigma^* : f_M(w) = f_{M'}(w)$

Satz 7.9 (WHILE \equiv TM) Jede WHILE-Berechenbare Funktion ist auch Turing-berechenbar. $(f(n) = Z(f_M(S(n))))$

Satz 7.10 (TM \equiv GOTO) Jede Turing-berechenbare Funktion ist auch GOTO-Berechenbar

Def. 7.11 (Primitiv rekursive Fkt.) Primitiv Rekursiv sind alle Fkt. $c^k : \mathbb{N}^k \rightarrow \mathbb{N}$ mit $c^k(x_1, \dots, x_k) = c$, $\pi_i^k : \mathbb{N}^k \rightarrow \mathbb{N}$ mit $\pi_i^k(x_1, \dots, x_k) = x_i$ und $s : \mathbb{N} \rightarrow \mathbb{N}$ mit $s(x) = x + 1$. Weiterhin sind $g : \mathbb{N}^l \rightarrow \mathbb{N}$ und $g_1, \dots, g_l : \mathbb{N} \rightarrow \mathbb{N}$ primitiv Rekursiv, so auch $f : \mathbb{N}^k \rightarrow \mathbb{N}$ mit $f(x_1, \dots, x_k) = g(g_1(x_1, \dots, x_k), \dots, g_l(x_1, \dots, x_k))$ Primitiv rekursiv. Sind g, h primitiv rekursiv so auch f gegeben durch $f(0, x_2, \dots, x_k) = g(x_2, \dots, x_k)$ und $f(x + 1, x_2, \dots, x_k) = h(f(x, x_2, \dots, x_k), x, x_2, \dots, x_k)$

Def. 7.12 (μ -rekursiv) Ist $f : \mathbb{N}^{k+1} \rightarrow \mathbb{N}$ eine Funktion, so bezeichne μf die Funktion $g : \mathbb{N}^k \rightarrow \mathbb{N}$ mit $g(x_1, \dots, x_k) = \min\{n \mid f(n, x_1, \dots, x_k) = 0 \text{ und } f(m, x_1, \dots, x_k) \neq \perp, \forall m < n\}$. Jede primitiv rekursive Funktion ist μ -rekursiv. Ist f μ -rekursiv, so auch μf .

Satz 7.13 (Universelle TM) Sei $\Sigma = \{0, 1\}$. Es giebt eine universelle TM M , für die gilt: $f_M(w \# x) = f_{M_w}(x)$, für alle Strings $w \in \Sigma^*$, die eine TM kodieren, und alle $x \in \Sigma^*$, für die M_w anhält. $M(w \# x)$ hält nicht an, falls w keine TM ist oder M_w bei eingabe von x nicht anhält.

Def. 7.14 (Partielle Funktionen) Bei Partiellen Funktionen f ist es erlaubt das $f(w)$ nicht für alle w definiert ist. Wir schreiben $f(w) = \perp$. Der definitionsbereich von f ist $D(f) := \{w \mid f(w) \neq \perp\}$. Wertebereich von f ist $D(f) = \{w \in \Sigma^* \mid \exists u \in \Sigma^* f(u) = w\}$

Def. 7.15 (Entscheidbar) Partielle Funktion f heist entscheidbar, falls es eine TM M giebt, die $\forall w \in D(f)$ mit ausgabe $f(w)$ anhält und $\forall w \notin D(f)$ nicht anhält. Eine Menge L heist entscheidbar, falls TM M giebt, die für alle $w \in L$ im zustand *ja* und für alle $w \notin L$ in *nein* anhält. Eine Menge L heist semi-entscheidbar, falls es eine TM giebt die in *ja* anhält falls $w \in L$ und sonst nicht

Def. 7.16 (Charakteristische Funktion) Sei $L \subseteq \Sigma^*$ Sprache, dann ist $\chi_L(w) = 1$ falls $w \in L$, 0 sonst und $\chi'_L(w) = 1$ falls $w \in L$, undefiniert sonst

Lemma 7.17 (Charakteristische Funktion und entscheidbarkeit) L entscheidbar $\Leftrightarrow \chi_L$ berechenbar. L semi-entscheidbar $\Leftrightarrow \chi'_L$ berechenbar. L entscheidbar $\Leftrightarrow L$ und \bar{L} semi-entscheidbar

Def. 7.18 (Rekursiv aufzählbar) L heißt rekursiv aufzählbar, falls L der Wertebereich einer totalen, berechenbaren Funktion ist

Lemma 7.19 (Aufzählbar \equiv semi-entscheidbar) L rekursiv aufzählbar $\Leftrightarrow L$ semi-entscheidbar.

Def. 7.20 (Spezielles Halteproblem) $K = \{w \in \{0,1\}^* \mid \text{w kodiert eine TM und } M_w \text{ h\ae}lt \text{ bei Eingabe } w\}$

Satz 7.21 K ist semi-entscheidbar aber nicht entscheidbar

Def. 7.22 (reduzierbar) Seien $L, L' \subseteq \Sigma^*$. L heißt auf L' reduzierbar $/L \leq L'$, falls es eine totale, berechenbare Funktion f gibt, so dass $\forall w \in \Sigma^*$ gilt: $w \in L \Leftrightarrow f(w) \in L'$

Lemma 7.23 (entscheidbarkeit und Reduktionen) Falls $L \leq L'$, dann gelten: L' entscheidbar $\Rightarrow L$ entscheidbar. Ist L unentscheidbar $\Rightarrow L'$ ist unentscheidbar.

Def. 7.24 (Halteproblem) $H = \{w\#x \mid \text{w kodiert eine TM und } M_w \text{ h\ae}lt \text{ bei Eingabe } w\}$. $H_0 = \{w\#x \mid \text{w kodiert eine TM und } M_w \text{ h\ae}lt \text{ bei Eingabe } w\}$

Satz 7.25 (H und H_0 sind nicht entscheidbar) n.c.

Satz 7.26 (Rice) Sei S eine Menge berechenbarer Funktionen mit $\emptyset \neq S \neq \mathcal{R}$. Dann ist die Sprache $C(S) = \{w \mid M_w \text{ berechnet eine Funktion aus } S\}$ nicht entscheidbar.

Def. 7.27 (Postsches Korrespondenzproblem (PCP)) Gegeben einer Folge $((x_1, y_1)); \dots, (x_k, y_k)$ von String-Paaren, ist eine Folge i_1, \dots, i_n gesucht so dass $x_{i_1} \dots x_{i_n} = y_{i_1} \dots y_{i_n}$ gilt.

Satz 7.28 (Unentscheidbarkeit von PCP) PCP ist semi-entscheidbar aber nicht entscheidbar

Def. 7.29 (Schnittproblem) Gegeben kontextfreier Sprachen G_1, G_2 , ist die Antwort auf $L(G_1) \cap L(G_2) \neq \emptyset$ gesucht

Def. 7.30 (Eindeutigkeitsproblem) Ist eine kontextfrei Grammatik G eindeutig?

Satz 7.31 Das Schnittproblem und das Eindeutigkeitsproblem sind für Kontextfreie Grammatiken unentscheidbar

8 Komplexitätstheorie

Def. 8.1 (MinSpanningTree) Gegeben einem zusammenhängendem Graphen $G = (V, E)$ und einer Gewichtsfunktion $L : E \rightarrow \mathbf{Q}^+$, ist ein aufspannender Baum $T \subseteq E$ von G mit minimaler Gesamtlänge $\sum_{e \in T} L(e)$ gesucht

Def. 8.2 (TravelingSalesman) Gegeben einem ungerichteten, zusammenhängendem Graphen $G = (V, E)$ und einer Gewichtsfunktion $l : E \rightarrow \mathbf{Q}^+$, ist ein Kreis $K \subseteq E$, der durch alle Knoten mit minimaler Gesamtlänge $\sum_{e \in K} l(e)$ gesucht.

Def. 8.3 (Schrittzahl) Bei TM: Ist $K_0(x) \vdash_M K_1 \vdash_M \dots \vdash_M K_m$ mit K_m Haltekonfiguration nach Eingabe von x , so setzen wir $t_M(x) = m$. Falls keine solche Folge existiert $t_M(x) = \perp$. analog bei GOTO- und WHILE-Programmen.

Def. 8.4 (Zeitschranke) Bei TM: Eine Funktion $T : \mathbf{N} \rightarrow \mathbf{N}$ heißt Zeitschranke für M falls $\forall x \in \Sigma^*$ gilt: $t_M(x) \leq T(|x|)$. Analog für *-Programme

Lemma 8.5 (alles das selbe) Alle Berechnungsmodelle sind bis auf einen polynomiellen Faktor äquivalent. Das gilt auch für viele andere Modelle (Vorsicht bei $x_i := x_j \times x_l$)

Def. 8.6 (Rucksack) Gegeben m Gegenstände mit Werten w_1, \dots, w_m und Gewichten g_1, \dots, g_m und einer Gewichtschränke G ist $I \subseteq \{1, \dots, m\}$ gesucht, so dass $\sum_{i \in I} g_i \leq G$ und $\sum_{i \in I} w_i$ maximal.

Def. 8.7 (HamiltonKreis) Gegeben einem Ungewichtetem Graphen G , ist ein Geschlossener Weg, in dem jeder Knoten genau ein mal vorkommt gesucht.

Def. 8.8 (SAT) Gegeben aussagenlogische Formel ϕ in konjunktiver Normalform, ist eine Belegung der Variable von ϕ , gesucht, die ϕ erfüllt.

Def. 8.9 (Clique) Gegeben eines Graphen $G = (V, E)$, wird die maximale Menge K von Knoten, gesucht, die alle mit einander verbunden sind.

Lemma 8.10 (NP-Probleme) Clique, Rucksack, HamiltonKreis, TravelingSalesman und SAT sind in NP

Satz 8.11 ($P \subseteq NP \subseteq EXP$)

Def. 8.12 Seien $L, L' \subseteq \Sigma^*$. L heißt auf L' polynomial reduzierbar ($L \leq L'$) falls es eine in Polynomialzeit berechenbare Funktion f , so dass $\forall e \in \Sigma^*$ gilt: $w \in L \Leftrightarrow f(w) \in L'$

Satz 8.13 (Cook) SAT ist NP -Vollständig

Satz 8.14 $SAT \leq_p 3SAT$

Lemma 8.15 Clique auch

Lemma 8.16 Hamiltonkreis auch

Lemma 8.17 Rucksack auch