Vorlesungsmodul Automaten und formale Sprachen - VorlMod AtmFrmSpr -

Matthias Ansorg

16. September 2005 bis 5. Oktober 2005

Zusammenfassung

Studentische Mitschrift zur Vorlesung Automaten und formale Sprachen (Modulnummer IS006 nach PO 2002) bei Prof. Dr. Ernst Kausen (Sommersemester 2005) im Studiengang Informatik an der Fachhochschule Gießen-Friedberg. Anrechnung nach PO 1991 als »Theoretische Informatik (Automaten und formale Sprachen)« (Wahlpflichtmodul im Schwerpunkt Systemtechnik) oder »Mathematische Verfahren für Informatik (Automaten und formale Sprachen)« (Wahlpflichtmodul im Schwerpunkt Technisch-Wissenschaftliche Anwendungen. Anrechnung nach PO 2002 als Schwerpunktmodul für einen beliebigen Schwerpunkt. Es scheint dass Prof. Kausen diese Veranstaltung zum ersten Mal angeboten hat, denn es gibt noch keine alten Klausuren. Es gibt also wohl auch noch keine studentischen Mitschriften, aber vermutlich bietet [15] genügend Hilfen. Die Gliederung dieser Mitschrift ist identisch zur Gliederung dieses Buches; der Stoff der Veranstaltung überdeckt sich dabei mit der Einführung und Teil 1 dieses Buches.

- **Bezugsquelle:** Die vorliegende studentische Mitschrift steht im Internet zum Download bereit. Quelle: Persönliche Homepage Matthias Ansorg http://matthias.ansorgs.de/
- **Lizenz:** Diese studentische Mitschrift ist public domain, darf also ohne Einschränkungen oder Quellenangabe für jeden beliebigen Zweck benutzt werden, kommerziell und nichtkommerziell; jedoch enthält sie keinerlei Garantien für Richtigkeit oder Eignung oder sonst irgendetwas, weder explizit noch implizit. Das Risiko der Nutzung dieser studentischen Mitschrift liegt allein beim Nutzer selbst. Einschränkend sind außerdem die Urheberrechte der angegebenen Quellen zu beachten.
- **Korrekturen und Feedback:** Fehler zur Verbesserung in zukünftigen Versionen, sonstige Verbesserungsvorschläge und Wünsche bitte dem Autor per e-mail mitteilen: Matthias Ansorg <mailto:matthias@ansorgs.de
- **Format:** Die vorliegende studentische Mitschrift wurde mit dem Programm LyX (graphisches Frontend zu LaTeX) unter Linux geschrieben und mit pdfLaTeXals pdf-Datei erstellt. Grafiken wurden mit dem Programm xfig unter Linux erstellt und als pdf-Dateien exportiert.
- **Dozent:** Prof. Dr. Ernst Kausen.
- **Verwendete Quellen:** <quelle> {<quelle>}.
- **Klausur:**
  - Dauer 90 Minuten.
  - Es sind keine Prüfungsvorleistungen notwendig; an der Klausur darf auch teilnehmen, wer die Vorlesung nicht besucht hat.
  - Die Klausur hat zu Anfang 2-3 sehr leichte Testaufgaben, die man zu mindestens 80% richtig lösen muss damit man besteht. Das soll unnötigen Zeitaufwand für die weitere Korrektur verhindern, der sonst entsteht weil sich etliche zur Klausur angemeldet haben die die Vorlesung nicht besucht haben. Mögliche Testaufgabe: Konstruktion eines DEA zu einer informal gegebenen Sprache L

- Hilfsmittel bei Prof. Kausen: erlaubt ist nur ein DIN A4 Blatt, beidseitig handschriftlich beschrieben. Dies soll eine Formelsammlung für Definitionen und Konstruktionen sein. Deshalb kommen natürlich keine Aufgaben vor, in denen nur Definitionen abgefragt werden.
- Hilfsmittel bei Prof. Metz: keine Unterlagen zugelassen.
- Man sollte sich das Skript der Veranstaltung bei Prof. Dr. Kausen kopieren, um in der Klausur dieselbe Notation wie in der Vorlesung verwenden zu können; andernfalls erhält man möglicherweise nicht die volle Punktzahl. Die in der Vorlesung verwendete Notation sollte man mit in die Formelsammlung aufnehmen.
- Die Konstruktionsaufgaben in der Klausur werden von recht einfachen Beispielen (2-3 Zustände) ausgehen, weil größere Beispiele zu viel Zeit brauchen würden.
- Wie bei allen Klausuren meldet man sich zur Klausur »Automaten und formale Sprachen« über das Online-System an. Im Notfall kann aber auch jemand mitschreiben, der sich nicht angemeldet hat.
- Prof. Kausen hat angegeben, dass von den Übungsblättern [2]r die Klausur zum Sommersemester 2005 die Blätter 1-5 und 7-9 besonders relevant sind. Von Blatt 5 sei besonders Aufgabe 1 wichtig; Aufgabe 2 wurde vom Typ her nicht direkt in der Veranstaltung behandelt.

Inhaltsverzeichnis

1 Einfhrung

1.1 Automaten, Berechenbarkeit und Komplexität

1.2 Mathematische Notation und Terminologie

Definitionen: Alphabet, Zeichen, Wort, Leerwort

- Ein Alphabet Σ ist eine endliche, nicht-leere Menge.
- Ein $a \in \Sigma$ heißt Zeichen.
- $w = a_1 a_2 a_3 \dots a_n$ ist Wort über $\Sigma \Leftrightarrow a_i \in \Sigma$
- $|w| = n$ heißt Länge des Wortes w
- ε heißt Leerwort ($|\varepsilon| = 0$
- $\Sigma^* = \{w \mid w \text{ ist Wort über } \Sigma\}$
- Jede Teilmenge $L \subset \Sigma^*$ heißt formale Sprache über Σ
- Die von einem Automaten erkannte Sprache ist $T(A) = \{w \in \Sigma^* \mid w \text{ wird akzeptiert}\}$

Kleene'sche Hülle Die Kleene'sche Hülle L^* einer Sprache L ist definiert durch[1]

$$\begin{aligned} L^* &= \bigcup_{n=0}^{\infty} L^i \\ &= L^0 \cup L^1 \cup L^2 \cup L^3 \cup \ldots = \{\varepsilon\} \cup L \cup LL \cup LLL \cup \ldots \end{aligned}$$

Weiter ist definiert:

$$\begin{aligned} L^+ &= \bigcup_{n=1}^{\infty} L^i \\ &= L^1 \cup L^2 \cup L^3 \cup \ldots = L \cup LL \cup LLL \cup \ldots \\ &= L^* \setminus \{\varepsilon\} \end{aligned}$$

Man beachte, dass es hierbei um eine Sprache L geht, nicht um ein Alphabet Σ Bei Alphabeten gilt, etwas analog:

- Σ^* ist die Menge aller Wörter über dem Alphabet Σ
- $\Sigma^+ = \Sigma^* \setminus \{\varepsilon\}$ ist die Menge aller Wörter über dem Alphabet Σ, die aus endlicher Aneinanderreihung (Konkatenation) von Symbolen aus Σ gebildet werden können.

So ist eine Sprache L zwar das »semantisch höhere Konstrukt«, aber nicht die mächtigere Menge: im Allgemeinen ist $L^* \subset \Sigma^*$ wegen der Definition $L \subset \Sigma^*$

Abzählbarkeit Es sei $|\mathbb{N}| = \omega$ Warum ist dann aber $|\{g : \mathbb{N} \to \{0,1\}\}| > \omega$ wie in [8, S. 6 (+10)] angegeben? Es ist ja $|\{g : \mathbb{N} \to \{0,1\}\}| = 2^{|\mathbb{N}|}$für jede natürliche Zahl in der Definitionsmenge $\mathbb{N}$ verdoppelt sich die Zahl der möglichen Funktionen, eine Gruppe hat hier den Funktionswert 0, eine 1

Chomsky-Hierarchie Die Beziehungen zwischen den einzelnen Typen von Grammatiken in der Chomsky-Hierarchie sind echte Inklusionen. In der Veranstaltung »Automaten und formale Sprachen« wurde die Teilmengenbeziehung gezeigt, aber nicht immer die echte Teilmengenbeziehung. Denn ersteres lässt sich einfach zeigen, indem man zeigt dass jede Typ$n$ Grammatik auch eine Typ $n-1$ Grammatik ist. Letzteres ist aber schwierig zu zeigen, entsprechende Beispiele sind nicht trivial zu finden.

1.3 Definitionen, Theoreme und Beweise

1.4 Beweistypen

Teil I
Automaten und formale Sprachen

2 Reguläre Sprachen

[1] [8, S. 4 (+10)] hat an dieser Stelle mehrere Fehler

2.1 Endliche Automaten

Definition: Endlicher Automat Ein Quintupel

$$A = (S, \Sigma, \delta, s_0, F)$$

heißt endlicher Automat[2] wenn gilt:

- Die Übergangsfunktion δ ist $\delta : \quad S \times (\Sigma \cup \{\varepsilon\}) \rightarrow p(S)$ Dabei ist $p(S) = \{M \mid M \subset S\}$ die Potenzmenge von S

Wenn man einschränkt auf $\delta : \quad S \times \Sigma \rightarrow S$ ist der Automat ein deterministischer endlicher Automat.δ kann bei NEAs und DEAs durch folgende rekursive Definition auf Eingabeworte erweitert werden - sie werden von links nach rechts »abgearbeitet« Es sei $s \in Sa \in \Sigma w \in \Sigma^*$

$$\begin{aligned} \delta(s, \varepsilon) &= s \\ \delta(s, aw) &= \delta\left(\underbrace{\delta(s, a)}_{s'}, w\right) \end{aligned}$$

Pfade durch einen endlichen Automaten Notation für Pfade durch einen endlichen Automaten (f$w \in \Sigma^* s_i \in S$

$$\begin{aligned} s_1 \xrightarrow{w} s_2 \quad &:\Leftrightarrow \quad \delta(s_1, w) = s_2 \\ s_1 \longrightarrow s_2 \quad &:\Leftrightarrow \quad \exists w \in \Sigma^* : \; s_1 \xrightarrow{w} s_2 \end{aligned}$$

Deterministischer endlicher Automat Die Übergangsrelation δ ist allgemein $\delta \subseteq S \times \Sigma^* \times S$ Bei deterministischen Automaten kann man δ statt als Relation auch als Abbildung (Funktion) formalisieren: $\delta : S \times \Sigma \rightarrow S$ Man nennt δ dann auch Übergangsfunktion. Ist der Automat vollständig, so ist es eine totale Abbildung ($\mathbb{D} = S \times \Sigma$), ansonsten eine partielle ($\mathbb{D} \subset S \times \Sigma$ Diese alternativen Formalisierungen sind nicht verwunderlich, da jede Funktion eine spezielle Relation ist: eine Relation Rfür die gilt $(p, r) \in R$, $(q, s) \in R$, $r \neq s \Rightarrow p \neq q$ ist eine Funktion. Auch die Schreibweise $p \rightarrow q(p, q) \in \rightarrow$ bei einer Relation$\rightarrow \subseteq A \times A$ zeigt schon die Verwandtschaft zwischen Funktionen und Relationen.

Ein DEA ist ein endlicher Automat, der buchstabierend ist, genau einen Startzustand besitzt und bei dem δ eine Funktion ist. Ein DEA muss also nicht vollständig sein.

Nichtdeterministischer endlicher Automat Ein nichtdeterministischer endlicher Automat (NEA) wird formalisiert als5-Tupel $(S, \Sigma, \delta, s_0, F)$ Statt der Menge F der Finalzustände kann man auch fordern, dass ein NEA genau einen Finalzustand$f$ haben muss. Jeder NEA lässt sich entsprechend transformieren, indem man εÜbergänge von allen bisherigen Finalzuständen zum neu eingeführten Finalzustand$f$ einführt.εÜbergänge erlauben Spontanübergänge.

Um zu prüfen ob ein NEA ein Wort akzeptiert, muss man alle vom Startzustand zur gegebenen Eingabe möglichen Pfade gleichzeitig verfolgen und dabei insbesondere auch Spontanübergänge berücksichtigen. Geschickt ist es dabei, diese möglichen Pfade durch den Automaten als verzweigten Graphen zu notieren. Erreicht man auf einem der Pfade einen Endzustand, ist es nicht mehr notwendig für die restlichen Pfade zu notieren ob auf ihnen ein Endzustand erreicht werden kann.

[2] $EA = NEA \cup DEA$

Operationen auf Sprachen Eine Operation hat die »Abschlusseigenschaft«, wenn ihr Ergebnis Element derselben mathematischen Struktur ist. Welche Operationen auf von endlichen Automaten erkannten Sprachen haben die Abschlusseigenschaft, d.h. werden wieder von endlichen Automaten erkannt? Nach [8, S. 22-23 (+10)] sind das die folgenden, sog.»regul« Operationen:

Schnittmenge

$$L_1 \cap L_2$$

Vereinigungsmenge

$$L_1 + L_2 = L_1 \cup L_2$$

Komplementbildung

$$\overline{L} = \Sigma^* \setminus L$$

Konkatenation

$$L_1 L_2 = L_1 \circ L_2 = \{vw \mid v \in L_1, w \in L_2\}$$

$L = L_1 = L_2$ entsteht der Spezialfall $L_1 L_2 = L^2$ bzw. bei mehrfacher Anwendung allgemeiner L^n

Kleene'sche Hülle

$$\begin{aligned} L^* &= \bigcup_{i \in \mathbb{N}_0} L^i \\ &= L^0 \cup L^1 \cup L^2 \cup L^3 \cup \ldots = \{\varepsilon\} \cup L \cup LL \cup LLL \cup \ldots \end{aligned}$$

Daneben gibt es weitere Operationen auf Sprachen ([8, S. 24 (+10)][8, S. 35ff (+10)]), die jedoch in dieser Veranstaltung nicht behandelt wurden:

- Homomorphismen
- inverse Homomorphismen
- Substitution
- Quotient
- Spiegelung

Algorithmus zur Konstruktion eines NEA-quivalenten DEA[3]

1. Das Zustandsübergangsdiagramm von A ist gegeben.
2. Wandle den NEA in einen NEA mit genau einem Startzustand.
3. Mache den NEA alphabetisch.

[3] Dieser Algorithmus ist äquivalent zu dem in der Veranstaltung »Automaten und formale Sprachen« im SS 2005 bei Prof. Dr. Ernst Kausen. Er verwendet jedoch etwas andere Schritte und Schreibweisen (in Anlehnung an [8, S.18 (+10)]) wo dies zu mehr Klarheit führt.

4. Mache den NEA buchstabierend (ε-Elimination).[4]

5. Die einzelnen Komponenten von $A = (S, \Sigma, \delta, s_0, F)$ ermitteln und aufschreiben. Dabei δ tabellarisch notieren: Funktionswerte $\delta(q, a)$ befinden sich am Kreuzungspunkt einer Zeile (aktueller Zustand) und einer Spalte (Eingabezeichen).

6. Die einzelnen Komponenten von $A' = (S', \Sigma', \delta', s'_0, F')$ ermitteln und aufschreiben:

   - $S' := \mathcal{P}(S) = \{M \mid M \subset S\}$ Beachte $\emptyset \in S'$
   - $\Sigma' := \Sigma$
   - $s'_0 := \{s_0\}$
   - $F' := \{R \in S' \mid R \cap F \neq \emptyset\}$
   - $\delta' : \ S' \times \Sigma \to S', \quad \delta'(R, a) := \{\delta(s, a) \mid s \in R\}$[5] In Worten: man bilde einen Folgezustand $R' = \delta'(R, a)$ in A' als Menge der Folgezustände $\delta(s, a)$ aller $s \in R$ in A
     - δ' notiere man tabellarisch wie schon δ und führe dabei zur Übersichtlichkeit neue Namen für die $R \in S'$ ein. Zum Beispiel: $\{s_0, s_1, s_3\} = s'_{013}$
     - Die Folgezustände einelementiger $R \in S'$ lassen sich direkt aus der tabellarischen Notation von δ ablesen.
     - Die restlichen Folgezustände ergeben sich durch Vereinigungsmengen mehrere Folgezustände einelementiger $R \in S'$
     - Wenn man in der Tabelle für δ' zuerst nur die Zustände einelementiger $R \in S'$ als Ausgangszustände (linke Spalte) notiert und dort nur noch solche ergänzt, in die diese Ausgangszustände übergehen, ergibt sich schon sofort der Minimalautomat.
   - Damit ist $\delta'(R, a)$ das $R' \in S'$, das aus all den $s \in S$ besteht die in A von einem $r \in R$ aus durch Eingabe von a erreichbar sind.

7. Das Zustandsübergangsdiagramm von A' zeichnen.

8. Vollständiger DEA oder Minimalautomat? Wenn bei der Konstruktion von δ' der Automat nicht minimiert wurde, ist der DEA nun ein vollständiger DEA. Aber auch jetzt noch kann der Minimalautomat erstellt werden. Dazu alle Zustände eliminieren, die vom Startzustand s'_0 nicht erreicht werden können.

Reduzierter endlicher Automat Sei ein deterministischer endlicher Automat $A = (S, \Sigma, \delta, s_0, F)$ Sei eine quivalenzrelation $s, s' \in S$ mit$s \ s' :\Leftrightarrow \forall w \in \Sigma^* : \ \delta(s, w) \in F \Leftrightarrow \delta(s', w) \in FA$ heißt reduzierter endlicher Automat $:\Leftrightarrow \forall s, s' \in S : \ s \ s' \Rightarrow s = s'$ In Worten: man kann von einem beliebigen Zustand jeweils nur einen Endzustand erreichen, unabhängig vom Eingabewort.

[4] Dieser Schritt kann durch Änderungen in der Definition von A'äquivalent ersetzt werden (was den Algorithmus jedoch verkompliziert). Dann ist:

$$\begin{aligned} s'_0 &:= \{s_0\} \cup \left\{\delta\left(s_0, \varepsilon^k\right) \mid k \geq 1\right\} \\ F' &:= \left\{R \in S' \mid R \cap \left(F \cup \left\{\delta\left(s_f, \varepsilon^k\right) \mid s_f \in F, \, k \geq 1\right\}\right) \neq 0\right\} \\ \delta'(R, a) &:= \left\{\delta\left(s, \varepsilon^k a \varepsilon^k\right) \mid s \in R, \, k \geq 1\right\} \end{aligned}$$

[5] Achtung: hier wurden δ und δ' als Funktionen formalisiert und nicht wie sonst manchmal als Relationen.

äquivalenter reduzierter endlicher Automat Sei ein Automat $A = (S, \Sigma, \delta, s_0, F)$ und eine Funktion $B(s) = \{s' \in S \mid s\ s'\}$[6] Dann ist der reduzierte endliche Automat A' mit $T(A) = T(A')$

$$\begin{aligned} A' &= (S', \Sigma, \delta', s_0', F') \\ S' &= \{B(s) \mid s \in S\} \\ s_0' &= B(s_0) \\ F' &= \{B(s) \mid s \in F\} \\ \delta'(B(s), a) &= B(\delta(s, a)) \end{aligned}$$

Erklärung zur Definition für δ'Üblicherweise würde man definieren $\delta'(B(s), a) = \{\delta(t, a) \mid t \in B(s)\}$ Nun gilt aber $s\ s' \Leftrightarrow \delta(s, a)\ \ \delta(s', a)$[7]: sind zwei Zustände äquivalent, so sind auch ihre Folgezustände äquivalent. Weil alle $t \in B(s)$ per Definition äquivalent sind, sind also auch alle Folgezustände $\delta(t, a) \mid t \in B(s)$äquivalent und sind zu ermitteln als Äquivalenzklasse zu einem Folgezustand$\delta(s, a)$ So erhält man obige Definition:

$$\delta'(B(s), a) = \{\delta(t, a) \mid t \in B(s)\} = B(\delta(s, a))$$

Konstruktion des äquivalenten reduzierten endlichen Automaten: state merging method Idee: S' ist die Partition von S unter der Äquivalenzrelation Ermittle also immer feinere Partitionen $\Pi_0 \Pi_1 \Pi_2, \ldots$ bis das Verfahren nicht mehr zu einer feineren Partition führt: $\Pi_k = \Pi_{k+1} = S'$ Das Verfahren ist:

1. Basispartition $\Pi_0 = \{F, S \setminus F\} = \{S_{01}, S_{02}\}$ Sie $S_i \in \Pi_0$ heißen Blöcke.
2. Blockteilung: untersuche für jeden Block B und jedes Eingabesymbol $a \in \Sigma$, ob die $\delta(s, a)$ im selben Zielblock landen[8] Falls nicht, muss B entsprechend aufgeteilt werden und es ergibt sich die feinere Partition$\Pi_1 = \{S_{11}, S_{12}, \ldots, S_{1n_1}\}$
3. Wiederhole Schritt 2 bis $\Pi_k = \Pi_{k+1} =: S'$

2.2 Nichtdeterminiertheit

Die »Determiniertheit« des DEA zeigt sich darin, dass zu einem Tupel $a \in S \times \Sigma$ genau ein Folgezustand S existiert - der Folgezustand ist damit eindeutig bestimmt, »determiniert« Und δ ist dann eine echte Funktion. Die »Nichtdeterminiertheit« des NEA dagegen besteht darin dass er für einen Zustand bei derselben Eingabe mehrere mögliche Folgezustände erlaubt. Der Folgezustand ist damit nicht durch aktuellen Zustand und Eingabe eindeutig festgelegt, d.h. er ist nicht »determiniert« Auch der aktuelle Zustand ist nicht determiniert: der Automat befindet sich im Endzustand des letzten Übergangs oder einem der Zustände, die damit durch εÜbergänge verbunden sind. Der Automat befindet sich in durch εÜbergänge verbundenen Zuständen »gleichzeitig«, zumindest logisch gesehen. Erst nach der nächsten Eingabe kann man eingrenzen, in welchen Zuständen der Automat sich zuletzt vor der Eingabe befunden hat. Das erinnert irgendwie an Unschärfe und Nichtdeterminiertheit in der Quantenphysik.

Nichtdeterminiertheit bei Automaten kann man sich etwas genauer so vorstellen: gibt man dem Automaten ein Wort stückweise als Eingabe, so durchläuft er alle Pfade und deren Folgepfade, die aufgrund der Eingabe denkbar sind, gleichzeitig. Er akzeptiert ein Wort, wenn er auf mindestens einem dieser Pfade einen Endzustand erreicht. Dass die Eingabe »stückweise« erfolgt bedeutet, dass man dem Automaten in jedem Schritt

[6] Es ist $B(s) = [s]$, d.h.$B(s)$ ermittelt zu einem s seine Äquivalenzklasse unter

[7] Beweisbar ist durch Induktion über die Definition der Äquivalenzrelation

[8] Dies ist gefordert durch $s\ s' \Leftrightarrow \delta(s, a)\ \ \delta(s', a)$für den Fall dass Block B tatsächlich nur äquivalente Zustände enthält

alternative Eingaben zur Verfügung stellt: ein Zeichen, zwei Zeichen, drei Zeichen usw. bis zum aktuellen Rest des Wortes. Denn der Automat akzeptiert möglicherweise auch Zeichenketten als Eingaben.

So ist zwar das Ergebnis des Automaten determiniert (»akzeptiert« oder »nicht akzeptiert«) und auch reproduzierbar. Aber der Weg zu diesem Ergebnis ist nicht determiniert, es werden alle möglichen Wege gleichzeitig verfolgt.

2.3 Reguläre Ausdrücke

Äquivalenz mit rechtslinearen Grammatiken Reguläre Sprachen können äquivalent durch endliche Automaten, reguläre Ausdrü oder reguläre Grammatiken charakterisiert werden. Eine Grammatik $G = (V, \Sigma, P, S)$ heißt rechtslinear, wenn jede Produktion eine der folgenden Formen hat (mit $A, B \in V w \in \Sigma^*$

$$\begin{aligned} A &\rightarrow wB \\ A &\rightarrow w \end{aligned}$$

Will man mit rechtslinearen Grammatiken auch das Leerwort ε erzeugen können, sind ein paar Erweiterungen nötig. Rechtslineare und (analog definierte) linkslineare Grammatiken fasst man zur Klasse der regulären Grammatiken zusammen (das sind Chomsky Typ 3-Grammatiken Rechtslineare und linkslineare Grammatiken charakterisieren beide die regulären Sprachen, da diese Sprachklasse unter Spiegelung abgeschlossen ist.

Axiome zum Rechnen mit formaler Sprache Seien $L, R, S, T \subset \Sigma^*$

$$\begin{aligned} L + L &= L \\ L + \emptyset &= L \\ R + S &= S + R \\ (R + S) + T &= R + (S + T) = R + S + T \\ (RS)\,T &= R\,(ST) \\ R\,\{\varepsilon\} &= \{\varepsilon\}\,R = R \\ R\emptyset &= \emptyset R = \emptyset \\ R\,(S + T) &= (RS) + (RT) = RS + RT \\ L^*L^* &= L^* \\ (L^*)^* &= L^* \end{aligned}$$

Axiome zum Rechnen mit regulären Ausdrücken Diese Axiome sind äquivalent zu den Axiomen zum Rechnen mit formaler Sprache. In [8, S.29 (+10)] finden sich folgende 14 Axiome zum Rechnen mit regulären Ausdrücken, hier mit Anmerkungen zum Verständnis:

$$\varepsilon = \emptyset^* \tag{1}$$

$$r = r \tag{2}$$

$$r + \emptyset = r \tag{3}$$

$$r + r = r \tag{4}$$

$$(r+s)+t=r+(s+t) \tag{5}$$

$$(rs)\,t=r\,(st) \tag{6}$$

$$r+s=s+r \tag{7}$$

$$r\,(s+t)=rs+rt \tag{8}$$

$$(r+s)\,t=rt+st \tag{9}$$

$$r\varepsilon=r \tag{10}$$

$$r\emptyset=\emptyset \tag{11}$$

Verständnis: $\emptyset$ ist ein regulärer Ausdruck, der keine Eingaben akzeptiert, auch nicht ε (den »leeren String« Die Konkatenation $r\emptyset$ muss diese Anforderung »nichts akzeptieren« erfüllen, und damit ist der Teil $r$llig unerheblich für das, was $r\emptyset$ akzeptiert. Diese Tatsache wird durch $r\emptyset=\emptyset$ ausgedrückt.

$$r^*=rr^*+\varepsilon \tag{12}$$

$$r^*=(r+\varepsilon)^* \tag{13}$$

Zwei weitere Axiome sind praktisch, aber zur Vollständigkeit entbehrlich:

$$r^*r^*=r^*$$

$$(r^*)^*=r^*$$

Rechenregel »Gleichungsauflösung«für reguläre Ausdrücke [8, S.30 (+10)] wird sie so formuliert: »Wenn die Gleichung $r=sr+t$ gilt und s die Leerworteigenschaft *nicht* erfüllt, dann gilt auch die Gleichung $r=s^*t$« Unter der Nebenbedingung, dass s die Leerworteigenschaft nicht erfüllt, wird also $r=s^*t$ als einzige Lösung der Gleichung $r=sr+t$ angesehen. Die Probe zeigt, dass es tatsächlich eine Lösung ist: setze $r=s^*t$ ein in $r=sr+t$

$$s^*t\overset{!}{=}s\,(s^*t)+t$$

Mit Gleichung 12 gilt:

$$s^*t=(ss^*+\varepsilon)\,t$$

Mit Gleichung 9 gilt weiter:

$$s^*t=(ss^*+\varepsilon)\,t=(ss^*)\,t+\varepsilon t$$

Und mit Gleichungen 6 und 10 gilt schlielich wie zu zeigen war:

$$s^*t=(ss^*+\varepsilon)\,t=s\,(s^*t)+t$$

Äquivalenzen ausrechnen durch das Rechnen mit regulären Ausdrücken Mit den Rechenregeln für reguläre Ausdrücke ist es möglich zu zeigen dass zwei reguläre Ausdrücke äquivalent sind, jedoch liefern sie kein effektives Konstruktionsverfahren dazu. Mit den Rechenregeln kann also auch nicht gezeigt werden dass zwei reguläre Ausdrücke nicht quivalent sind [8, S.31 (+10)] Will man durch die Rechenregeln die Äquivalenz zweier regulärer Ausdrücke r_1 und r_2 beweisen, muss man insgesamt einen Term der Form $r_1 = \ldots = r_2$ aufstellen, es bietet sich also folgendes Beweisverfahren an (in dem jedoch immer noch eine Beweisidee nötig ist):

1. Schreibe $r_1 =$ auf.
2. Forme r_1 schrittweise durch die Axiome zu einem längeren regulären Ausdruck r_1' um.
3. Fasse r_1' schrittweise durch die Axiome zum kürzeren regulären Ausdruck r_2 zusammen.

Erzeugte reguläre Sprache zu einem regulären Ausdruck ermitteln Das Prinzip ist, die Operationen im regulären Ausdruck von außen nach innen in die gleichlautenden Operationen auf regulären Sprachen umzuformen. An Beispielen:

$$L(a(b+c)) = L(a)L(b+c) = L(a)(L(b)+L(c)) = \{a\}(\{b\}\cup\{c\}) = \{a\}\times\{b,c\} = \{ab,ac\}$$

$$L(a^*) = L(a)^* = \{a\}^* = \bigcup_{n=0}^{\infty}\{a^n\} = \{\varepsilon, a, aa, aaa, aaaa, \ldots\} = \{a^n \mid n \geq 0\}$$

Endlichen Automaten zu einem regulären Ausdruck konvertieren Das Verfahren folgt der rekursiven Definition der regulären Ausdrücke und beweist den Satz »L regul⇒∃ endlicher Automat A mit $L = T(A)$« Sei nun:

- $a \in \Sigma$
- $A = (S_1, \Sigma, \delta_1, s_a, F_1) B = (S_2, \Sigma, \delta_2, s_b, F_2)$
- α, β sind reguläre Ausdrücke mit $L(\alpha) = T(A)$ und $L(\beta) = T(B)$

Dann gilt:

1. Der endliche Automat zu $L(a) = \{a\}$ ist $C = (S, \Sigma, \delta, s_0, F) = (\{s_0, s_f\}, \{a\}, \delta : \delta(s_0, a) = s_f, s_0, \{s_f\})$
2. Der endliche Automat zu $L(\varepsilon) = \{\varepsilon\}$ ist $C = (S, \Sigma, \delta, s_0, F) = (\{s_0\}, \emptyset, \delta, s_0, \{s_0\})$
3. Der endliche Automat zu $L(\emptyset) = \emptyset$ ist $C = (S, \Sigma, \delta, s_0, F) = (\{s_0\}, \emptyset, \delta, s_0, \emptyset)$
4. Der endliche Automat zu $L(\alpha+\beta)$ ist $C = (S, \Sigma, \delta, s_0, F)$ mit

$$\begin{aligned} S &= S_1 \cup S_2 \cup s_0 \\ F &= F_1 \cup F_2 \\ \delta(s,a) &= \begin{cases} \delta_1(s,a) & \text{f r } s \in S_1 \\ \delta_2(s,a) & \text{f r } s \in S_2 \\ \{s_a, s_b\} & \text{f r } s = s_0, a = \varepsilon \\ \emptyset & \text{f r } s = s_0, a \neq \varepsilon \end{cases} \end{aligned}$$

5. Der endliche Automat zu $L(\alpha\beta)$ ist $C = (S, \Sigma, s_a, \delta, F_2)$ mit

$$\begin{aligned} S &= s_1 \cup s_2 \\ \delta(s,a) &= \begin{cases} \delta_2(s,a) & \text{f r } s \in S_2 \\ \delta_1(s,a) & \text{f r } s \in S_1, s \notin F_1 \\ \delta_1(s,a) \cup & \text{f r } s = s_0, a = \varepsilon \\ \emptyset & \text{f r } s = s_0, a \neq \varepsilon \end{cases} \end{aligned}$$

2.4 Nichtreguläre Sprachen

Endlicher Index von R_L Der Satz von Myhill und Nerode besagt: Eine Sprache ist genau dann regulär wenn der Index der Relation R_L endlich ist. Vgl.[8, S.31 (+10)] und [8, S.33 (+10)] In der Veranstaltung bei Prof. Kausen wurde dieser Stoff nicht behandelt.

Pumping-Lemma Das Pumping-Lemma[9] wird verwendet zum Nachweis dass eine Sprache nicht mehr regulär ist, also zum Test der Regularität

Nichtregularität der Sprache der Palindrome Ein Palindrom ist ein Wort, das vorwärts und rückwärts gelesen gleich lautethttp://de.wikipedia.org/wiki/Palindrom In [8, S.33 (+10)] wird bewiesen dass die Sprache L der Palindrome über $\Sigma = \{a, b\}$ nicht regulär ist. Die Idee dabei:

1. Pumping-Lemma fordert, dass für jedes $x \in L$ einer regulären Sprache mit $|x| \geq n$ eine Zerlegung $x = uvw$ (mit $|uv| \leq n|v| \geq 1$) existiert so dass $uv^iw \in L$für jedes $i \in \mathbb{N}$
2. wähle also geschickt ein konkretes $x \in L$ Und wähle eine konkrete Zerlegung $x = uvw$ so dass $|uv| \leq n|v| \geq 1$ Da n (ab dem Pumping-Lemma gilt) i.A. unbekannt ist, wähle ein $x \in L$ mit n-parametrisierbarer Länge und eine Zerlegung $x = uvw$ mit n-parametrisierten Längen der Bestandteile so dass $|uv| \leq n|v| \geq 1$ erfüllt ist.
3. Wähle ein $i \in \mathbb{N}$ so dass $uv^iw \notin L$ Damit ist die Nichtregularität von L nachgewiesen. Oft bietet sich wohl $i = 0$ an.

3 Kontextfreie Sprachen

3.1 Kontextfreie Grammatiken

Die Definition für deterministisch kontextfreie Sprachen lautet: »Eine kontextfreie Sprache L heit deterministisch, wenn es einen deterministischen Kellerautomaten A gibt, der die Sprache L über Endzustände akzeptiert (d.h.$L = L(A)$«[8, S.90 (+10)] Dabei gilt: Es muss ber Akzeptierung ber Endzustände definiert werden, denn eine Definition über Akzeptierung mit leerem Keller wäre echt schwächer - denn die dadurch akzeptierten Sprachen sind präfixfrei [8, S.90 (+10)] Dabei bedeutet »präfixfrei«»Eine Sprache L heißt präfixfrei, wenn für kein Paar $u \sqsubseteq v$ mit $u \neq v$ sowohl $u \in L$ als auch $v \in L$ gilt.«[8, S.98 (+10)]

3.2 Kellerautomaten

Englisch »Pushdown-Automaten«, deshalb in [15] auch abgekürzt mit »PDA«. Kellerautomaten sind die Akzeptoren für kontextfreie Sprachen. Kellerautomaten haben ein unbeschränktes Register mit dem man sich das Abarbeiten eines Zustandes merken kann; die Unbeschränktheit des unbeschränkten Register durchbricht dabei die endliche Beschränktheit bei endlichen Automaten. Ein Zustandsübergang eines Kellerautomaten enthält die Änderung des Kellerkopfes.

Zu jeder kontextfreien Sprache kann ein nichtdeterministischer Kellerautomat konstruiert werden. Jedoch gibt es nicht zu jeder kontextfreien Sprache einen determinierten

[9] wörtlich bersetzt »Aufblassatz«, deutsche Bezeichnung »uvw-Theorem«

Kellerautomaten; ein Beispiel für eine solche kontextfreie Sprache ist aber schwierig zu finden.

3.3 Nicht-kontextfreie Sprachen

Teil II
Berechenbarkeit

4 Church-Turing These

4.1 Turingmaschinen

4.2 Varianten von Turingmaschinen

4.3 Definition des Algorithmus

5 Entscheidbarkeit

5.1 Entscheidbare Sprachen

5.2 Das Halteproblem

6 Reduzierbarkeit

6.1 Unentscheidbare Probleme aus der Theorie der formalen Sprachen

6.2 Ein einfaches unentscheidbares Problem

9 Ortskomplexität

9.1 Savitch'sches Theorem

9.2 Die Klasse PSPACE

9.3 PSPACE-Vollständigkeit

9.4 Die Klassen L und NL

9.5 NL-Vollständigkeit

9.6 NL ist CONL

10 Widerspenstigkeit (Intractability)

10.1 Hierarchietheoreme

10.2 Relativierung

10.3 Schaltungskomplexit

11 Fortgeschrittene Aspekte der Komplexit

11.1 Approximierende Algorithmen

11.2 Probabilistische Algorithmen

11.3 Alternierung

11.4 Interaktive Beweissysteme

11.5 Parallelcomputing

11.6 Kryptografie

Teil IV
Anhang

A Aufgabensammlung

Die folgenden Aufgaben sind zusammengenommen gleichzeitig:

- Die gesamte Menge der Übungsaufgaben zur Veranstaltung »Automaten und formale Sprachen« bei Prof. Dr. Hans-Rudolf Metz. Siehe [3]. Die Reihenfolge hier ist identisch zu der in [3].
- Fast die gesamte Menge der Übungsaufgaben zur Veranstaltung »Automaten und formale Sprachen« bei Prof. Dr. Ernst Kausen. Siehe [2]. Ausgenommen sind nur die ersten beiden Übungsblätter mit mathematischen Grundlagen, die nicht speziell klausurrelevant sind. Die Reihenfolge hier ist identisch zu der in [2].
- Eine kleine Auswahl der Übungsaufgaben aus dem Buch Michael Sipser: »Introduction to the Theory of Computation« (Boston 1997). Siehe [15]. Die Reihenfolge hier folgt nicht der in [15].

A.1 Zustandsübergangsdiagramm eines DEA

Aufgabenstellung Ein DEA sei formal gegeben durch $(\{q_1, q_2, q_3, q_4, q_5\}, \{\mathtt{u}, \mathtt{d}\}, \delta, q_3, \{q_3\})$, wobei δ durch die folgende Tabelle gegeben ist. Geben Sie das Zustandsübergangsdiagramm des DEA an.

	u	d
q_1	q_1	q_2
q_2	q_1	q_3
q_3	q_2	q_4
q_4	q_3	q_5
q_5	q_4	q_5

Quelle: [2, Bl.1 Aufg.1] bzw.[15, exercise 1.3]

Lösung

A.2 DEAs für Sprachen ber $\Sigma = \{0, 1\}$

Aufgabenstellung Geben Sie Zustandsübergangsdiagramme für DEAs an, die die folgenden Sprachen erkennen. In jedem Fall ist das Alphabet $\{0, 1\}$

1. $\{w \mid w$ beginnt mit 1 und endet mit $0\}$
2. $\{w \mid w$ enthält mindestens dreimal $1\}$
3. $\{w \mid w$ enthält den Substring 0101, also $w = x0101y$ für beliebige x und $y\}$
4. $\{w \mid w$ hat Länge mind. 3 und das dritte Symbol ist $0\}$
5. $\{w \mid w$ beginnt mit 0 und hat ungerade Länge, oder beginnt mit 1 und hat gerade Länge$\}$
6. $\{w \mid w$ enthält nicht den Substring $110\}$
7. $\{\varepsilon, 0\}$
8. $\{w \mid w$ enthält gerade Anzahl 0en oder genau zwei 1en$\}$
9. Die leere Menge.
10. Alle Zeichenketten auer der leeren Zeichenkette.

Quelle: [2, Bl.1 Aufg.2] bzw.[15, exercise 1.4, part a-f.k-n].

Lösung

A.3 Sprachen unter regulren Operationen

Aufgabenstellung Es sei $\Sigma = \{0, 1\}$und AB seien Sprachen über dem Alphabet Σ mit

$$A = \{w \mid w \text{ endet mit } 1\}$$

$$B = \{w \mid w \text{ beginnt mit } 1\}$$

Beschreiben Sie, welche Sprachen sich mit den regulären Operationen $A \cup BA \circ BA^*$ und B^* ergeben. Quelle: [2, Bl.1 Aufg.3]

Lösung

$$A \cup B = \{w \mid w = 1x \vee w = x1,\ x \in \Sigma^*\}$$

$$A \circ B = \{w \mid w = x11y,\ x, y \in \Sigma^*\}$$

$$A^* = \{w \mid w = x^n,\ n \in \mathbb{N}_0,\ x = y1,\ y \in \Sigma^*\}$$

$$B^* = \{w \mid w = x^n,\ n \in \mathbb{N}_0,\ x = 1y,\ y \in \Sigma^*\}$$

A.4 NEAs zu gegebenen Sprachen konstruieren (1)

Aufgabenstellung Konstruieren Sie NEAs mit der angegebenen Anzahl von Zuständen, die jeweils die angegebene Sprache akzeptieren. Es sei jeweils $\Sigma = \{0, 1\}$

1. Die Sprache $\{w \mid w \text{ endet mit } 00\}$ mit drei Zuständen.
2. Die Sprache $\{w \mid w \text{ enthält den Substring } 0101, \text{ also } w = x0101y \text{ für beliebige } x \text{ und } y\}$ mit fünf Zustnden.
3. Die Sprache $\{w \mid w \text{ enthält gerade Anzahl 0en oder genau zwei 1en}\}$ mit sechs Zuständen.
4. Die Sprache $\{0\}$ mit zwei Zuständen.
5. Die Sprache $\{\varepsilon\}$ mit einem Zustand.

Quelle: [2, Bl.2 Aufg.1] bzw.[15, exercise 1.5 part a-d.f].

Lösung

A.5 NEA in DEA umwandeln

Aufgabenstellung In der Vorlesung wurde hergeleitet, dass es zu jedem NEA einen äquivalenten DEA gibt. Verwenden Sie die im Beweis benutzte Methode, um die folgenden beiden nichtdeterministischen endlichen Automaten in äquivalente deterministische endliche Automaten zu konvertieren. Zeichnen Sie zunächst die Zustandsdiagramme.

1. Es sei $N = \{\{1, 2\}, \{\mathtt{a}, \mathtt{b}\}, \delta, 1, \{1\}\}$, wobei δ durch die folgende Tabelle gegeben ist.

	a	b	ε
1	$\{1, 2\}$	$\{2\}$	$\emptyset$
2	$\emptyset$	$\{1\}$	$\emptyset$

2. Es sei $N = \{\{1, 2, 3\}, \{\mathtt{a}, \mathtt{b}\}, \delta, 1, \{2\}\}$, und δ ist durch die folgende Tabelle gegeben.

	a	b	ε
1	$\{3\}$	$\emptyset$	$\{2\}$
2	$\{1\}$	$\emptyset$	$\emptyset$
3	$\{2\}$	$\{2, 3\}$	$\emptyset$

Quelle: [2, Bl.2 Aufg.2] bzw.[15, exercise 1.12]

Lösung

A.6 NEA zur Konkatenation regulärer Sprachen

Aufgabenstellung In der Vorlesung wurde gezeigt, da die Klasse der regulären Sprachen unter der Operation der Konkatenation abgeschlossen ist. Verwenden Sie die für die Herleitung benutzte Konstruktion, um das Zustandsdiagramm eines nichtdeterministischen endlichen Automaten anzugeben, der die Konkatenation der Sprachen

- $\{w \mid \text{die Länge von } w \text{ ist höchstens } 5\}$ und
- $\{w \mid \text{jede ungerade Position von } w \text{ ist eine } 1\}$

erkennt. Es sei $\Sigma = \{0, 1\}$ Quelle: [2, Bl.3 Aufg.1] bzw.[15, exercise 1.7 part a].

Lösung Die Konstruktion besteht darin, die Endzustände des Automaten A_1durch ε-Übergänge mit den Startzuständen des Automaten A_2 zu verbinden. Die Anfangszustände des kombinierten Automaten sind die Anfangszustände von A_1, seine Endzustände sind die Endzustände von A_2. Als Formel nach [8, S.23 (+10)]

$$\begin{aligned} A &= (Q, \Sigma, \delta, I, F) \\ &= (Q_1 \cup Q_2, \Sigma, \delta_1 \cup (F_1 \times \{\varepsilon\} \times I_2) \cup \delta_2, I_1, F_2) \end{aligned}$$

A.7 NEAs zu gegebenen Sprachen konstruieren (2)

Aufgabenstellung Konstruieren Sie NEAs mit der angegebenen Anzahl von Zuständen, die die jeweils angegebene Sprache erkennen.

1. Die Sprache $0^*1^*0^*0$ mit drei Zuständen.
2. Die Sprache 0^* mit einem Zustand.

Quelle: [2, Bl.3 Aufg.2] bzw.[15, exercise 1.5 part e.g]

Lösung Sipser unterscheidet in [15] fast nicht zwischen regulären Ausdrcken und regulären Sprachen. Jedoch sind reguläre Ausdrücke ein syntaktisches Konzept, und reguläre Sprachen sind das durch reguläre Ausdrcke repräsentierbare semantische Konzept. Reguläre Sprachen sind auch durch weitere syntaktische Konzepte repräsentierbar, z.B. durch endliche Automaten. Exakter wrde die Aufgabenstellung zum ersten Teil also z.B. lauten: »Konstruieren sie einen NEA mit drei Zuständen, der die vom regulären Ausdruck$0^*1^*0^*0$ erzeugte Sprache $L\left(0^*1^*0^*0\right)$ erkennt.«

Zur Lösung hilft es hier nicht weiter, NEAs für die Teilausdrcke zu kombinieren (wie in [8, S.23 (+10)]), da dann wesentlich mehr als drei Zustände benötigt werden.

A.8 Reguläre Ausdrcke zu gegebenen Sprachen konstruieren

Aufgabenstellung Geben Sie reguläre Ausdrcke an, die die folgenden Sprachen erzeugen:

1. $\{w \mid w$ beginnt mit 1 und endet mit $0\}$
2. $\{w \mid w$ enthält mindestens dreimal $1\}$
3. $\{w \mid w$ enthält den Substring 0101, also $w = x0101y$ für beliebige x und $y\}$
4. $\{w \mid w$ hat Länge mind. 3 und das dritte Symbol ist $0\}$
5. $\{w \mid w$ beginnt mit 0 und hat ungerade Länge, oder beginnt mit 1 und hat gerade Länge$\}$
6. $\{w \mid w$ enthält nicht den Substring $110\}$
7. $\{w \mid$ die Länge von w ist höchstens $5\}$
8. $\{w \mid w$ ist ein beliebiger String au er 11 und $111\}$
9. $\{w \mid$ jede ungerade Position von w ist eine $1\}$
10. $\{w \mid w$ enthält mindestens zwei 0en und höchstens eine $1\}$
11. $\{\varepsilon, 0\}$
12. $\{w \mid w$ enthält gerade Anzahl 0en oder genau zwei 1en$\}$
13. Die leere Menge.
14. Alle Zeichenketten außer der leeren Zeichenkette.

Quelle: [2, Bl.3 Aufg.3] bzw.[15, exercise 1.13]

Lösung Es wird stets zuerst die Beschreibung der Sprache und dann der reguläre Ausdruck genannt.

$\{w \mid w$ beginnt mit 1 und endet mit $0\}$

$$1\,(0+1)^*\,0$$

$\{w \mid w$ enthält mindestens dreimal $1\}$

$$(0+1)^*\,1\,(0+1)^*\,1\,(0+1)^*\,1\,(0+1)^*$$

$\{w \mid w$ enthält den Substring 0101, also $w = x0101y$ für beliebige x und $y\}$

$$(0+1)^*\,0101\,(0+1)^*$$

$\{w \mid w$ hat Länge mind. 3 und das dritte Symbol ist $0\}$

$$(0+1)\,(0+1)\,0\,(0+1)^*$$

$\{w \mid w$ beginnt mit 0 und hat ungerade Länge, oder beginnt mit 1 und hat gerade Länge$\}$

$$0\,((0+1)\,(0+1))^* + 1\,(0+1)\,((0+1)\,(0+1))^*$$

$\{w \mid w$ enthält nicht den Substring $110\}$

$$\begin{array}{rl} & \varepsilon \\ + & (0+1) \\ + & (0+1)\,(0+1) \\ + & 0^*\,(10+0^*)^*\,111^* \end{array}$$

$\{w \mid$ die Länge von w ist höchstens $5\}$

$$(0+1+\varepsilon)\,(0+1+\varepsilon)\,(0+1+\varepsilon)\,(0+1+\varepsilon)\,(0+1+\varepsilon)$$

$\{w \mid w$ ist ein beliebiger String außer 11 und $111\}$

$$\begin{array}{rl} & \varepsilon \\ + & 0\,(0+1)^* \\ + & 10\,(0+1)^* \\ + & 110\,(0+1)^* \\ + & 111\,(0+1)\,(0+1)^* \end{array}$$

$\{w \mid$ jede ungerade Position von w ist eine $1\}$

$$(1\,(0+1))^*$$

$\{w \mid w$ enthält mindestens zwei 0en und höchstens eine $1\}$

$$\begin{array}{rl} & 000^*\,(1+\varepsilon)\,0^* \\ + & 0\,(1+\varepsilon)\,00^* \\ + & (1+\varepsilon)\,000^* \end{array}$$

$\{\varepsilon, 0\}$

$$\varepsilon + 0$$

$\{w \mid w \text{ enthält gerade Anzahl 0en oder genau zwei 1en}\}$

$$(1^*01^*01^*)^* + 0^*10^*10^*$$

Die leere Menge.

$$\emptyset$$

Alle Zeichenketten außer der leeren Zeichenkette.

$$(0+1)(0+1)^*$$

A.9 Beispiele und Gegenbeispiele für Wörter einer Sprache

Aufgabenstellung Geben Sie für jede der folgenden Sprachen zwei Wörter an, die zu dieser Sprache gehören, und zwei Wörter, die nicht zu dieser Sprache gehören. (Also insgesamt 4 Wörter pro Sprache.) Nehmen Sie das Alphabet $\Sigma = \{a, b\}$für alle diese Sprachen an.

1. a^*b^*
2. $a(ba)^*b$
3. $a^* \cup b^*$
4. $(aaa)^*$
5. $\Sigma^*a\Sigma^*b\Sigma^*a\Sigma^*$
6. $aba \cup bab$
7. $(\varepsilon \cup a)b$
8. $(a \cup ba \cup bb)\Sigma^*$

Quelle: [2, Bl.4 Aufg.1] bzw.[15, exercise 1.15]

Lösung

A.10 Reguläre Ausdrücke in NEAs konvertieren

Aufgabenstellung Verwenden Sie das in der Vorlesung beschriebene Verfahren, um die folgenden regulären Ausdrücke in nichtdeterministische endliche Automaten zu konvertieren.

1. $(0 \cup 1)^*000(0 \cup 1)^*$
2. $\emptyset^*$

Quelle: [2, Bl.5 Aufg.1] bzw.[15, exercise 1.14 part a.c]

Lösung

A.11 DEAs in reguläre Ausdrücke konvertieren

Aufgabenstellung Verwenden Sie das in der Vorlesung beschriebene Verfahren, um die folgenden endlichen Automaten in reguläre Ausdrücke zu konvertieren. Zeichnen Sie zunächst die Zustandsdiagramme der Automaten.

1. Es sei $M = \{\{1,2\}, \{\mathtt{a},\mathtt{b}\}, \delta, 1, \{2\}\}$, wobei δ durch die folgende Tabelle gegeben ist.

	a	b
1	1	2
2	2	1

2. Es sei $M = \{\{1,2,3\}, \{\mathtt{a},\mathtt{b}\}, \delta, 1, \{1,3\}\}$, und δ ist durch die folgende Tabelle gegeben.

	a	b
1	2	2
2	2	3
3	1	2

Quelle: [2, Bl.5 Aufg.2] bzw.[15, exercise 1.16]

Lösung

A.12 Pumping Lemma für reguläre Sprachen

Aufgabenstellung Benutzen Sie das Pumping Lemma (d.i., den uvw-Satz) um nachzuweisen dass die folgenden Sprachen nicht regulär sind.

1. $A_1 = \{0^n 1^n 2^n \mid n \geq 0\}$
2. $A_2 = \{www \mid w \in \{\mathtt{a},\mathtt{b}\}^*\}$
3. $A_3 = \{\mathtt{a}^{2^n} \mid n \geq 0\}$ Dabei bedeutet $\mathtt{a}^{2^n}$ eine Zeichenkette von $2^n\mathtt{a}$

Quelle: [2, Bl.6 Aufg.1] bzw.[15, exercise 1.17]

Lösung

A.13 Parsebäume und Ableitungen für eine kontextfreie Grammatik

Aufgabenstellung Es sei $G = (V, \Sigma, R, E)$ eine kontextfreie Grammatik mit $V = \{E, T, F\}$, $\Sigma = \{\mathtt{a}, +, \times, (,)\}$ und den folgenden Regeln:

$$\begin{aligned} E &\rightarrow E{+}T \mid T \\ T &\rightarrow T{\times}F \mid F \\ F &\rightarrow (E) \mid \mathtt{a} \end{aligned}$$

Geben Sie Parsebäume und Ableitungen für jeden der folgenden Strings an.

1. $\mathtt{a}$
2. $\mathtt{a} + \mathtt{a}$
3. $\mathtt{a} + \mathtt{a} + \mathtt{a}$
4. $((\mathtt{a}))$

Quelle: [2, Bl.7 Aufg.1] bzw.[15, exercise 2.1]

Lösung

A.14 Fragen zu einer kontextfreien Grammatik

Aufgabenstellung Beantworten Sie die folgenden Fragen in Bezug auf die folgende kontextfreie Grammatik G

$$\begin{aligned} R &\rightarrow XRX \mid S \\ S &\rightarrow \mathtt{a}T\mathtt{b} \mid \mathtt{b}T\mathtt{a} \\ T &\rightarrow XTX \mid X \mid \varepsilon \\ X &\rightarrow \mathtt{a} \mid \mathtt{b} \end{aligned}$$

1. Was sind die Nichtterminalsymnbole und Terminalsymbole von G? Welches ist das Startsymbol?
2. Geben Sie drei Beispiele für Strings die in $L(G)$ enthalten sind.
3. Geben Sie drei Beispiele für Strings die *nicht* in $L(G)$ enthalten sind.
4. Wahr oder falsch: $T \Rightarrow \mathtt{aba}$
5. Wahr oder falsch: $T \stackrel{*}{\Rightarrow} \mathtt{aba}$
6. Wahr oder falsch: $T \Rightarrow T$
7. Wahr oder falsch: $T \stackrel{*}{\Rightarrow} T$
8. Wahr oder falsch: $XXX \Rightarrow \mathtt{aba}$
9. Wahr oder falsch: $X \stackrel{*}{\Rightarrow} \mathtt{aba}$
10. Wahr oder falsch: $T \stackrel{*}{\Rightarrow} XX$
11. Wahr oder falsch: $T \stackrel{*}{\Rightarrow} XXX$
12. Wahr oder falsch: $S \stackrel{*}{\Rightarrow} \varepsilon$
13. Beschreiben Sie $L(G)$ in natürlicher Sprache.

Quelle: [2, Bl.7 Aufg.2] bzw.[15, exercise 2.3]

Lösung

A.15 Kontextfreie Grammatiken zu gegebenen Sprachen konstruieren (1)

Aufgabenstellung Geben Sie kontextfreie Grammatiken an, die die folgenden Sprachen erzeugen. Das Alphabet sei jeweils $\Sigma = \{0, 1\}$

1. $\{w \mid w \text{ enthält mindestens drei 1en}\}$
2. $\{w \mid w \text{ beginnt und endet mit demselben Symbol}\}$
3. $\{w \mid w \text{ hat ungerade Länge}\}$

Quelle: [2, Bl.7 Aufg.3] bzw.[15, exercise 2.4 part a-c]

Lösung

A.16 Umwandlung in Chomsky-Normalform

Aufgabenstellung Die kontextfreie Grammatik

$$\begin{aligned} A &\rightarrow BAB \mid B \mid \varepsilon \\ B &\rightarrow 00 \mid \varepsilon \end{aligned}$$

soll mit der in der Vorlesung besprochenen Methode in eine äquivalente kontextfreie Grammatik in Chomsky-Normalform umgewandelt werden.

Quelle: [2, Bl.8 Aufg.1] bzw.[15, exercise 2.14]

Lösung

A.17 Zustandsdiagramme für Kellerautomaten zu gegebenen Sprachen (2)

Aufgabenstellung Geben Sie Zustandsdiagramme von Kellerautomaten für die folgenden Sprachen an.

1. Das Komplement der Sprache $\{\mathtt{a}^n\mathtt{b}^n \mid n \geq 0\}$
2. $\{w\#x \mid w^{\mathcal{R}}\text{ist ein Substring von } x\text{, wobei } w, x \in \{0,1\}^*\}$

Quelle: [2, Bl.8 Aufg.2] bzw.[15, exercise 2.7]

Lösung

A.18 Kontextfreie Grammatiken zu gegebenen Sprachen konstruieren (2)

Aufgabenstellung Geben Sie kontextfreie Grammatiken an, die die folgenden Sprachen erzeugen. Das Alphabet sei jeweils $\Sigma = \{0,1\}$

1. $\{w \mid w \text{ hat ungerade Länge und } 0 \text{ als mittleres Symbol}\}$
2. $\{w \mid w \text{ enth lt mehr } 1\text{en als } 0\text{en}\}$
3. $\{w \mid w = w^{\mathcal{R}}\text{, d.i., } w \text{ ist ein Palindrom}\}$
4. Die leere Menge.

Quelle: [2, Bl.8 Aufg.3] bzw.[15, exercise 2.4 part d-g]

Lösung

A.19 Zustandsdiagramme für Kellerautomaten zu gegebenen Sprachen (2)

Aufgabenstellung Geben Sie Zustandsdiagramme von Kellerautomaten für die folgenden Sprachen an.

1. $\{w \mid w \text{ hat ungerade Länge und } 0 \text{ als mittleres Symbol}\}$
2. $\{w \mid w \text{ enthält mehr } 1\text{en als } 0\text{en}\}$
3. $\{w \mid w = w^{\mathcal{R}}\text{, d.i., } w \text{ ist ein Palindrom}\}$
4. Die leere Menge.

Quelle: [2, Bl.8 Aufg.4] bzw.[15, exercise 2.5]

Lösung

A.20 Kontextfreie Grammatik in Kellerautomaten konvertieren

Aufgabenstellung Konvertieren Sie mit der in der Vorlesung vorgestellten Methode die kontextfreie Grammatik $G = (V, \Sigma, R, E)$ mit $V = \{E, T, F\}\Sigma = \{\mathtt{a}, +, \times, (,)\}$ und den Regeln

$$\begin{aligned} E &\rightarrow E{+}T \mid T \\ T &\rightarrow T{\times}F \mid F \\ F &\rightarrow (E) \mid \mathtt{a} \end{aligned}$$

in einen äquivalenten Kellerautomaten (Pushdown-Automaten).

Quelle: [2, Bl.9 Aufg.1] bzw.[15, exercise 2.11]

Lösung

A.21 Abschlusseigenschaften kontextfreier Sprachen

Aufgabenstellung Es soll gezeigt werden, dass der Durchschnitt zweier kontextfreier Sprachen nicht immer kontextfrei ist, und dass auch das Komplement einer kontextfreien Sprache nicht kontextfrei sein muss.

1. Verwenden Sie die Sprache $A = \{\mathtt{a}^m\mathtt{b}^n\mathtt{c}^n \mid m, n \geq 0\}$ und die Sprache $B = \{\mathtt{a}^n\mathtt{b}^n\mathtt{c}^m \mid m, n \geq 0\}$ zusammen mit dem in der Vorlesung besprochenen ersten Beispiel zum Pumping-Lemma kontextfreie Sprachen, um zu zeigen, dass die Klasse der kontextfreien Sprachen nicht abgeschlossen unter dem Durchschnitt ist.
2. Zeigen Sie mit dem Ergebniss des ersten Teils der Aufgabe und mit einer Regel von De Morgan (siehe Formelsammlung), dass die Klasse der kontextfreien Sprachen nicht abgeschlossen unter dem Komplement ist.

Quelle: [2, Bl.10 Aufg.1] bzw.[15, exercise 2.2]

Lösung

A.22 Nachweis dass Sprachen nicht kontextfrei sind über Pumping-Lemma

Aufgabenstellung Verwenden Sie das Pumping-Lemma um zu zeigen dass die folgenden Sprachen nicht kontextfrei sind.

1. $\left\{0^n\#0^{2n}\#0^{3n} \mid n \geq 0\right\}$
2. $\left\{w\#x \mid w \text{ ist ein Substring von } x, \text{ wobei } w, x \in \{\mathtt{a}, \mathtt{b}\}^*\right\}$

Quelle: [2, Bl.10 Aufg.2] bzw.[15, exercise 2.18 part b-c]

Lösung

A.23 Zahl der Ableitungsschritte für kontextfreie Grammatiken in Chomsky-Normalform

Aufgabenstellung Zeigen Sie: Wenn G eine kontextfreie Grammatik in Chomsky-Normalform ist, dann sind für jeden String $w \in L(G)$ mit Länge $n \geq 1$ genau $2n - 1$ Schritte notwendig für jede Ableitung von w.

Quelle: [2, Bl.10 Aufg.3] bzw.[15, exercise 2.19]

Lösung

A.24 Turing-Entscheidbarkeit

Aufgabenstellung In der Vorlesung wurde gezeigt, da eine nichtdeterministische Turing-Maschine durch eine deterministische Turing-Maschine simuliert werden kann, dass also eine Sprache genau dann Turing-erkennbar ist, wenn sie von einer nichtdeterministischen Turing-Maschine erkannt wird. Ändern Sie den Beweis ab, um zu zeigen, da eine Sprache genau dann entscheidbar ist, wenn eine nichtdeterministische Turing-Maschine sie entscheidet. (Sie können den folgenden Satz über Bäume benutzen: Wenn jeder Knoten in einem Baum endlich viele Kinder und jeder Zweig des Baumes endlich viele Knoten hat, dann hat der Baum endlich viele Knoten.)

Quelle: [2, Bl.11 Aufg.1] bzw.[15, exercise 3.3]

Lösung

A.25 Turing-Maschinen zu gegebenen Sprachen beschreiben

Aufgabenstellung Geben Sie informelle Beschreibungen von Turing-Maschinen an (entsprechend den Beschreibungen in der Vorlesung; Zustandsdiagramme sollen nicht gezeichnet werden), die die folgenden Sprachen über dem Alphabet $\{0, 1\}$ entscheiden.

1. $\{w \mid w \text{ enthält gleich viele } \texttt{0}\text{en und } \texttt{1}\text{en}\}$
2. $\{w \mid w \text{ enthält doppelt so viele } \texttt{0}\text{en wie } \texttt{1}\text{en}\}$
3. $\{w \mid w \text{ enthält nicht doppelt so viele } \texttt{0}\text{en wie } \texttt{1}\text{en}\}$

Quelle: [2, Bl.11 Aufg.2] bzw.[15, exercise 3.8]

Lösung

A.26 Mächtigkeit von Kellerautomaten mit k Kellern

Aufgabenstellung Sei ein k-Kellerautomat ein Kellerautomat mit k Kellern. Somit ist ein 0-Kellerautomat ein nichtdeterministischer endlicher Automat und ein 1-Kellerautomat ist ein konventioneller Kellerautomat.1-Kellerautomaten sind bekanntlich mächtiger als 0-Kellerautomaten, d.h. sie erkennen eine mächtigere Klasse von Sprachen.

1. Zeigen Sie dass 2-Kelleautomaten mchtiger als 1-Kellerautomaten sind.
2. Zeigen Sie dass 3-Kellerautomaten nicht mächtiger als 2-Kellerautomaten sind.

Hinweis: Simulieren sie das Band einer Turing-Maschine durch zwei Kellern.

Quelle: [2, Bl.11 Aufg.3] bzw.[15, exercise 3.9]

Lösung

B Klausurrelevanter Stoff

Die folgende erschöpfende Zusammenstellung des klausurrelevanten Stoffes hat Prof. Dr. Ernst Kausen in der Klausurvorbesprechung 2005-09-20 gemacht, also für die Prfung zum Sommersemester 2005. Darin nicht genannter Stoff (z.B. der Stoff der in Vorträgen vermittelt wurde) ist nicht klausurrelevant. Die Veranstaltung zeichnete sich besonders dadurch aus dass die verschiedenen Konstruktionsverfahren an ausführlichen Beispielen behandelt wurden. Die Turing-Maschine wurde aus Zeitgründen nicht mehr behandelt.

B.1 Formale Sprache

- Alphabet
- Σ^*
- formale Sprache $L \subset \Sigma^*$
- Operationen auf formalen Sprachen: $L_1 + L_2 L_1 L_2 L^n L^*$

B.2 Endliche Automaten

- Ein DEA $(S, \Sigma, \delta, s_0, F)$ wobei δ eine Abbildung (Funktion) ist: $\delta : S \times \Sigma \to S$
  - Zustandsdiagramm eines DEA zeichnen können
  - Bedeutung von $\delta(s, w)$ bzw.$s \xrightarrow{w} s'$ kennen
  - Konzept der akzeptierten Sprache $T(A)$
  - Konstruktion eines DEA zu einer Sprache L, die durch informale Beschreibung gegeben ist. In der Vorlesung wurden dazu um 10 Beispiele durchgenommen.
- NEA $(S, \Sigma, \delta, s_0, F)$ wobei δ im Unterschied zum DEA eine Abbildung $\delta : S \times \Sigma_\varepsilon \to p(S)$ ist[10]
  - Konzepte wie beim DEA
  - Vorteil: erlaubt flexibleren Umgang mit Automaten als bei DEAs, geringere Zustandsmengen.
  - Nachteil: das »Wortproblem« ist wesentlich schwieriger zu lsen als mit einem DEA. Denn man muss alle Verzweigungen untersuchen, es gibt keinen determinierten Pfad durch den Automaten wie beim DEA.
  - Äquivalenz von NEA und DEA. Bewiesen wird das durch ein konstruktives Verfahren, das man können muss: die Konstruktion eines DEA zu einem gegebenen NEA. Diese Aufgabe wird in der Klausur von einem einfachen Beispiel ausgehen, weil dieses Verfahren einige Zeit braucht.

B.3 Reguläre Ausdrücke

- rekursive Definition regulärer Ausdrücke
- die regulären Operationen $+ \cdot *$
- Michael Sipser macht in [15] kaum Unterschiede zwischen regulären Sprachen und regulären Ausdrcken. In dieser Veranstaltung wurde aber genau ein regulärer Ausdruck α unterschieden von $L(\alpha)$, d.i. die von α erzeugte reguläre Sprache.
- regulären Ausdruck konstruieren können, der eine (informal beschriebene) Sprache L erzeugt
- Äquivalenzkonzept: reguläre Sprache $\cong$[11] zur Menge der Sprachen die von endlichen Automaten erkannt werden. Also gibt es zu jedem regulren Ausdruck einen endlichen Automaten, der diese Sprache akzeptiert und andersherum:
  - Konstruktion »regulärer Ausdruck $\to$ NEA« für die Klausur können)

[10] $\Sigma_\varepsilon = \Sigma \cup \{\varepsilon\}$, was Spontanbergnge erlaubt; $p(S)$ ist die Potenzmenge von S, also die Menge aller mglichen Teilmengen von S

[11] »ist quivalent«

  - Konstruktion »NEA → regulärer Ausdruck« (kommt nicht in der Klausur vor)

- Pumping Lemma (uvw-Theorem) für reguläre Ausdrücke. Der Beweis des Pumping Lemma wird in der Klausur nicht gefordert. Auch eine Anwendung des Pumping Lemma zum Test der Regularität einer Sprache wird in der Klausur nicht gefordert sein, weil dies zu theoretisch sein.

B.4 Grammatiken

- Hier geht der Stoff weit über Michael Sipsers Buch [15] hinaus. Sipser behandelt nur kontextfreie Grammatiken, in der Vorlesung wurde aber das allgemeine Chomsky-Konzept einer Grammatik eingeführt: $G = (N, T, R, S)$ Das Regelsystem ist $R \subset \Sigma_e^* \times \Sigma^* \Sigma = N \cup T$ und $\Sigma_e^* = \Sigma^* \setminus \{\varepsilon\} = \Sigma^+$
- direkte Ableitbarkeit: $w \overrightarrow{G} w'$ Bedeutung: $x\alpha y \overrightarrow{G} x\beta y \Leftrightarrow (\alpha, \beta) \in R$
- Ableitbarkeit: $w \overset{*}{\overrightarrow{G}} w'$
- Die Sprache $L = L(G)$ zu einer gegebenen Grammatik G erkennen können; das ist, informal beschreiben können, welche Art Wörter man aus einer Grammatik ableiten kann.
- Chomsky-Hierarchie verstehen (es geht jeweils um echte Inklusionen)
  - Typ 0: die allgemeine Chomsky-Grammatik $G = (N, T, R, S)$
  - Typ 1: kontextsensitive Grammatiken
  - Typ 2: kontextfreie Grammatiken
  - Typ 3: rechtslineare Grammatiken (auch genannt: reguläre Grammatiken)
- Wichtige Äquivalenz, eigentlich der Kernsatz der Vorlesung:

$$\text{rechtslineare Grammatiken} \cong \text{regulre Sprachen} \cong \text{endliche Automaten}$$

- Diese Äquivalenz (zwischen rechtslinearen Grammatiken und regulren Sprachen) wird durch zwei Konstruktionsverfahren bewiesen, die man beide anwenden können muss[12]
  - vom DEA zu einer rechtslinearen Grammatik
  - von einer rechtslinearen Grammatik zu einem NEA

B.5 Kontextfreie Sprachen

- Konstruktion beherrschen: kontextfreie Grammatik zu einer informal gegebenen Sprache L konstruieren.
- Die Chomsky-Normalform der kontextfreien Grammatik (CNF) wurde behandelt, man muss also wissen was das ist. Ihre Konstruktion kommt in der Klausur jedoch nicht dran weil sie in der Vorlesung nicht in voller Tiefe behandelt wurde und weil sie zu lange dauert (mindestens 30 Minuten).
- Kellerautomaten (mit Kelleralphabet, Kellerstartzustand, komplizierteres Akzeptieren)

[12] In der Vorlesung wurden dazu einige Beispiele behandelt, die man sich ansehen sollte.

- determinierte Kellerautomaten
- nichtdeterminierte Kellerautomaten
- verstehen: wie funktioniert eine Ableitung in einem Kellerautomaten
- Konzept der vom Kellerautomaten erkannten Sprache $T(A)$ Die Konfiguration zu einem Wort w ist (s, w, k); man kann von einer Konfiguration zu einer anderen durch einen geeigneten Zustandsübergang kommen.»Akzeptieren« bedeutet bei einem Kellerautomaten entweder »Automat in einem Endzustand« oder »leerer Keller«
- Konstruieren können: nichtdeterminierten Kellerautomaten zu einer informal gegebenen kontextfreien Sprache L. Die Umkehrung (Sprache an einem nichtdeterminierten Kellerautomaten erkennen) wurde in der Vorlesung teilweise behandelt, ist aber nicht klausurrelevant.

C Schnellreferenz zur Klausur

Literatur

[1] Prof. Dr. Ernst Kausen: Persönliche Homepage. Mit Material zur Veranstaltung »Automaten und formale Sprachen« inkl. einer Inhaltsangabe zur Veranstaltung. Quelle: http://homepages.fh-giessen.de/~hg8429/

[2] Prof. Dr. Ernst Kausen: Übungen zur Veranstaltung »Automaten und formale Sprachen«, Sommersemester 2005. Die Übungen zu Abbildungen und Relationen stammen aus einer Mathematikvorlesung von Prof. Kausen aus dem WS 2003/2004 und sind deshalb nicht speziell klausurrelevant. Die restlichen Übungsblätter sind die Übungsblätter identisch mit [3]. Quelle: http://homepages.fh-giessen.de/~hg8429/uebungen.html

[3] Prof. Dr. Hans-Rudolf Metz: Übungen zur Veranstaltung »Automaten und formale Sprachen«, Sommersemester 2004. Die Übungen stammen fast alle aus [15]. Quelle: http://hera.mni.fh-giessen.de/~hg8070/afs04/afs04.html

[4] MNI-Wiki: Kausen; Wiki der Fachschaft MNI der FH Gießen-Friedberg. Bisher keine Informationen zur Veranstaltung »Automaten und formale Sprachen« (Stand 2005-09-17). Quelle: http://www.fh-giessen.de/fachschaft/mni/mediawiki/index.php/Kausen

[5] Lena-Luisa Heß: Mitschrift der Vorlesung Automaten und formale Sprachen; 2003-06-30. Studentischer Mitschrieb zu dieser Veranstaltung bei Prof. Dr. Lutz Eichner, FH Gießen-Friedberg. Quelle: http://members.tripod.com/lena_hess/data/skripte/AutomatenUndFormaleSprachen.pdf

[6] Prof. Dr. Lutz Eichner: Klausuren in Automaten und formale Sprachen. Mit diesen Klausuren zu üben ist auch für die Veranstaltung bei Prof. Kausen sinnvoll: ähnliche Aufgabentypen und klare Aufgabenstellungen.

[7] Eitan M. Gurari: An Introduction to the Theory of Computation; Ohio State University; Computer Science Press, 1989, ISBN 0-7167-8182-4. Frei online verfügbar. Quelle: http://www.cse.ohio-state.edu/~gurari/theory-bk/theory-bk.html

[8] Ekkart Kindler, Steffen Manthey: Automaten, Formale Sprachen und Berechenbarkeit I; Skript zur Vorlesung im WS 2001/02 an der TU München; Version: 1.30 vom 30. April 2002. 167 Seiten. Quelle: http://wwwcs.uni-paderborn.

de/cs/kindler/Lehre/Skripte/Automaten.pdf oder http://wind.in.tum.de/lehre/fospr/WS0102/Skript/SkriptIII.pdf. Ergänzend gibt es auch das handschriftliche Skript in gescannter Form und weitere Unterlagen auf http://wind.in.tum.de/lehre/fospr/WS0102/unterlagen.html.

[9] Birgit Pfitzmann: Informatik 3 - Ergänzungen zum Skript=Sipser-Buch; Version 11.2.2001. 60 Seiten Ergänzungen zu [15]. Quelle: http://www-krypt.cs.uni-sb.de/download/scripts/vorlesung_info3.pdf

[10] Herbert S. Wilf: Algorithms and Complexity; 1994. 139 Seiten. Zum freien Download verfügbar. Quelle: http://www.cis.upenn.edu/~wilf/AlgComp.html

[11] Leonid A. Levin: Fundamentals of Computing. Quelle: http://www.cs.bu.edu/fac/lnd/toc/

[12] CT 313: Theory of Computation 2001. Quelle: http://www.u-friend.com/lectures/CT313/

[13] Susan H. Rodger: CPS 140, Spring 1998, Lecture Notes. Quelle: http://www.cs.duke.edu/~rodger/courses/cps140/lecture.html

[14] Hans-Peter Bischof: Foundations of Computer Science – CSC 221 – Spring 1998; 1998. Quelle: http://www.cs.rit.edu/~hpb/Lectures/98_221/all.html

[15] Michael Sipser: Introduction to the Theory of Computation; Boston 1997. Das Inhaltsverzeichnis befindet sich auf http://www-math.mit.edu/~sipser/itoc-1and2.html. Einige Lösungen zu Übungsaufgaben befinden sich unter http://www.cas.mcmaster.ca/~soltys/cas705-f04/ und http://www.cas.mcmaster.ca/~soltys/cas705-f03/.

[16] MIT OpenCourseWare: 18.404J / 6.840J Theory of Computation, Fall 2002. Dieser Veranstaltung liegt [15] zugrunde. Hier werden online Hausaufgaben-Problemstellungen und eine Klausur angeboten. Dieses Material wurde auch von Prof. Metz für seine Veranstaltung »Automaten und formale Sprachen« genutzt. Quelle: http://ocw.mit.edu/OcwWeb/Mathematics/18-404JTheory-of-ComputationFall2002/CourseHome/index.htm.

[17] MIT OpenCourseWare: 6.045J / 18.400J Automata, Computability, and Complexity, Spring 2002. Enthält Aufgabenstellungen und eine Klausursammlung. Dieses Material wurde auch von Prof. Metz für seine Veranstaltung »Automaten und formale Sprachen« genutzt. Quelle: http://ocw.mit.edu/OcwWeb/Electrical-Engineering-and-Computer-Science/6-045JAutomata--Computability--and-ComplexitySpring2002/CourseHome/index.htm

[18] Alexander Asteroth, Christel Baier: Theoretische Informatik: eine Einführung in Berechenbarkeit, Komplexität und formale Sprachen. Pearson Studium. Auch als eBook verfügbar (http://www.ciando.com/shop/book/index.cfm/fuseaction/show_book/bok_id/1084/cat_id/68/cat_nav/65 oder http://www.bol.de/shop/home/artikeldetails/ID4153439/), etwa 30 EUR. Vorteil: sofort zum Download verfügbar. Prof. Dr. Metz hat dieses Buch in den Literaturangaben zu seiner Veranstaltung »Automaten und formale Sprachen«.