Zusammenfassung Theo-Info

Tim Baumann, http://timbaumann.info/uni-spicker

Reguläre Sprachen

Def. Ein **DFA** ist ein Tupel $(Q, \Sigma, \delta, q_0, F)$ mit

- Einer **Zustandsmenge** Q (endlich)
- Einem **Eingabealphabet** Σ (endlich)
- Einer **Übergangsfunktion** $\delta : Q \times \Sigma \to Q$
- Einem **Startzustand** $q_0 \in Q$
- Einer Menge von **akzeptierenden Zuständen** $F \subset Q$

Def. Ein **NFA** ist ein Tupel $(Q, \Sigma, \delta, q_0, F)$ mit den gleichen Komponenten wie ein DFA, aber die Übergangsfunktion hat eine andere Quellmenge: $\delta : Q \times (\Sigma \cup \{\epsilon\}) \to Q$.

Algorithmus. Umformung eines NFA $(Q, \Sigma, \delta, q_0, F)$ zu einem äquivalenten DFA

Bezeichne mit $E(P)$ für $P \subset Q$ die Menge der durch ϵ-Übergänge erreichbaren Zustände. Konstruiere dann den DFA folgendermaßen:

- Zustandsmenge: $\mathcal{P}(Q)$
- Übergangsfunktion schickt $\{q_1, \ldots, q_n\}$ bei Eingabe a auf $E(\{\delta(q_1, a), \ldots, \delta(q_n, a)\})$
- Startzustand: $E(\{q_0\})$
- Akzeptierte Zustände: alle Teilmengen $A \subset Q$ mit $A \cap F \neq \emptyset$

Def. **Reguläre Ausdrücke** sind induktiv wie folgt definiert:

Regulärer Ausdruck e	Wert $L(e)$
$\emptyset$	$\emptyset$
ϵ	$\{\epsilon\}$
a (für ein $a \in \Sigma$)	$\{a\}$
$e_1 \cdot e_2$	$L(e_1) \cdot L(e_2)$
$e_1 + e_2$	$L(e_1) \cup L(e_2)$
$(e_1)^*$	$L(e_1)^*$

Satz. Der Wert jedes regulären Ausdrucks ist eine reguläre Sprache. (Beweis durch Induktion)

Satz. Jede reguläre Sprache L ist Wert eines regulären Ausdrucks

Beweis. Sei $M = (Q, \Sigma, \delta, q_0, F)$ ein DFA, der L entscheidet. Sei o.E. $\Sigma = \{1, ..., m\}$. Bezeichne mit $L_k(p, q)$ mit $k, p, q \in \{1, ..., m\}$ die Sprache, die alle Wörter enthält, die den Automaten M von Zustand p in den Zustand q bringen nur unter Benutzung von Zwischenzuständen in $\{1, ..., k\}$. Durch Induktion über k sieht man dann, dass all diese Sprachen regulär sind und es gilt $L = \bigcup_{q \in F} L_m(q_0, q)$.
Induktionsanfang: $L_0(p, q) = \{a \in \Sigma | \delta(p, a) = q\} \cup \{\epsilon \,|\, p = q\}$
Induktionsschritt: $L_k(p, q) = L_{k-1}(p, k) \cdot L_{k-1}(k, k)^* \cdot L_{k-1}(k, q)$ □

Def. Sei L eine Sprache über einem Alphabet Σ. Dann ist die **Nerode-Relation** für L wie folgt definiert:
$(u \equiv_L v) :\Leftrightarrow (\forall\, z \in \Sigma^* \;:\; uz \in L \Leftrightarrow vz \in L)$.
Die Anzahl der Äquivalenzklassen dieser Relation wird der **Nerode-Index** von L genannt.

Algorithmus. Konstruktion des **Nerode-Automaten** zu einer Sprache L mit endlichem Nerode-Index:

- Q: Menge der Äquivalenzklassen $\{[u] \,|\, u \in \Sigma^*\}$
- $\delta([u], a) = [ua]$ für $u \in \Sigma^*$ und $a \in \Sigma$
- $q_0 = [\epsilon]$
- $F = \{[u] \,|\, u \in L\}$

Bem. Der Nerode-Automat für eine Sprache L ist minimal. Eine Sprache ist genau dann regulär, wenn ihr Nerode-Index endlich ist.

Satz (Pumping-Lemma). Sei L eine reguläre Sprache. Dann gibt es ein $t \in N$, sodass jedes Wort $w \in L$ mit Länge mindestens t als $w = uvx$ geschrieben werden kann, wobei gilt:

- $uv^ix \in L$ für alle $i \in \mathbb{N}$
- $v \neq \epsilon$
- $|uv| \leq t$

Kontextfreie Sprachen

Def. Eine **kontextfreie Grammatik** (CFG) ist ein Tupel (V, Σ, P, S) mit

- Einer endlichen Menge von Variablen bzw. Nichtterminalen V
- Einer von V disjunkten, endlichen Menge von Terminalen Σ
- Einer endlichen Menge von Produktionen P. Eine Produktion hat die Form $A \longrightarrow \alpha$, wobei $\alpha \in (V \cup \Sigma)^*$
- Einem Startsymbol $S \in V$

Def. Falls in einer Grammatik $G = (V, \Sigma, P, S)$ die Produktion $(A \longrightarrow \beta)$ vorkommt, so sagt man, dass sich $\alpha\beta\gamma$ aus $\alpha S\gamma$ **ableiten** lässt (geschrieben: $\alpha\beta\gamma \Rightarrow \alpha S\gamma$). Wenn sich ein Wort v aus einem Wort w in einem oder mehr, beliebig vielen oder genau m vielen Schritten ableiten lässt, so schreibt man $w \overset{+}{\Rightarrow} v$, $w \overset{*}{\Rightarrow} v$ bzw. $w \overset{m}{\Longrightarrow} v$.

Def. Ein Wort α über $V \cup \Sigma$ mit $S \overset{*}{\Rightarrow} \alpha$ nennt man eine **Satzform**.

Def. Ein Wort $w \in \Sigma^*$ wird von G **generiert**, wenn $S \overset{*}{\Rightarrow} w$. Die Sprache aller Wörter über Σ, die von G generiert werden, heißt die von G erzeugte Sprache. Eine Sprache heißt **kontextfrei**, wenn sie von einer kontextfreien Grammatik erzeugt wird.

Def. Eine CFG G heißt **mehrdeutig**, wenn es eine Wort in $L(G)$ gibt, das der Ertrag von zwei verschiedenen Syntaxbäumen ist. Wenn dies nicht der Fall ist, so heißt die CFG **eindeutig**.

Def. Eine CFG ist in **Chomsky-NF**, wenn jede Produktion eine der folgenden Formen hat:

- $A \longrightarrow a$ mit $a \in \Sigma$
- $A \longrightarrow BC$ mit $B, C \in V$

Algorithmus. Sei G eine CFG und $L = L(G)$. Dann gibt es eine CFG G' in Chomsky-NF, die $L \backslash \{\epsilon\}$ erzeugt. Ändere dafür G schrittweise wie folgt:

1. Elimination von ϵ-Produktionen
2. Elimination von Einheitsproduktionen (Produktionen der Form $A \longrightarrow B$ mit $A, B \in V$)
3. Auslagerung der Produktion von Terminalen (führe für jedes verwendete Terminal $a \in \Sigma$ ein neues Nichtterminal P_a und die Produktion $P_a \longrightarrow a$ ein)
4. Aufsplitten von Produktionen mit mehr als zwei Variablen auf der rechten Seite

Def. Eine CFG ist in **Greibach-NF**, wenn jede Produktion in P die Form $A \longrightarrow a\alpha$ hat, mit $a \in \Sigma$ und $\alpha \in V^*$.

Satz. Sei L eine kontextfreie Sprache. Dann wird $L \backslash \{\epsilon\}$ von einer Grammatik in Greibach-NF erzeugt.

Def. Ein **Kellerautomat** ist ein Tupel $(Q, \Sigma, \Gamma, \delta, q_0, F)$ mit

- Q, Σ, q_0 und F definiert wie beim DFA,
- einer endl. Menge Γ, dem **Kelleralphabet**, und einer
- Übergangsfunktion $\delta : Q \times (\Sigma \cup \{\epsilon\}) \times (\Gamma \cup \{\epsilon\}) \to \mathcal{P}(Q \times (\Gamma \cup \{\epsilon\}))$.

Bem. Kellerautomaten arbeiten nichtdeterministisch und müssen nicht auf jeder Eingabe halten.

Satz. Eine Sprache ist genau dann kontextfrei, wenn sie von einem Kellerautomaten akzeptiert wird.

Satz (Ogdens Lemma). Sei L eine kontextfreie Sprache. Dann gibt es ein $t \in N$, sodass jedes Wort $w \in L$, in dem mindestens t Symbole markiert sind, in der Form $w = uvxyz$ geschrieben werden kann, wobei gilt:

- $uv^ixy^iz \in L$ für alle $i \in \mathbb{N}$
- vy enthält mindestens ein markiertes Symbol.
- vxy enthält höchstens t markierte Symbole.

Turing-Maschinen

Def. Eine **Turing-Maschine** (TM) ist ein Tupel $(Q, \Sigma, \Gamma, \delta, q_0, q_a, q_r, \textvisiblespace)$ mit

- Einer Zustandsmenge Q (endlich)
- Einem Eingabealphabet Σ (endlich)
- Einem **Bandalphabet** Γ enthält Σ
- Einer Übergangsfunktion $\delta : Q \times \Gamma \to Q \times \Gamma \times \{\leftarrow, \rightarrow\}$
- Einem Startzustand $q_0 \in Q$
- Einem **akzeptierenden Zustand** $q_a \in Q$
- Einem **verwerfenden Zustand** $q_r \in Q$
- Einem Symbol $\textvisiblespace \in \Gamma \backslash \Sigma$

Def. Eine **Konfiguration** einer Turing-Maschine ist ein Wort der Form $\alpha q \beta$ mit $\alpha, \beta \in \Gamma^*$, wobei β nicht auf $\textvisiblespace$ endet, und $q \in Q$.

Def. Seien $\phi = \alpha q b \beta$ und $\phi' = \alpha' q' \beta'$ Konfigurationen einer Turing-Maschine M. Man schreibt $\phi \xrightarrow{M} \phi'$ (gesprochen: ϕ' ergibt sich aus ϕ), falls $\delta(q, b) = (q', b', D)$ und $\alpha b' \beta = \alpha' \beta'$ und der Kopf der Turing-Maschine in ϕ' gegenüber ϕ nur um eins verrückt ist gemäß der Richtung D.

Def. Eine Turingmaschine **akzeptiert** eine Eingabe $w \in \Sigma *$, wenn es eine abbrechende Folge von Konfigurationen $C_0, ..., C_t$ gibt, wobei $C_0 = q_0 w$ die Anfangskonfiguration ist, jede weiter Konfiguration sich aus der vorherigen ergibt und der Zustand in der letzten Konfiguration der akzeptierende Zustand q_a ist.

Bem. Zur Turingmaschine sind folgende Maschinen äquivalent, d.h. sie können auf TMs simuliert werden und TMs auf ihnen:

- **Mehrband-TMs:** Wie TMs, haben allerdings mehrere Bänder und je einen Lesekopf pro Band zur Verfügung, die unabhängig voneinander bewegt werden können.
- **Nichtdeterministische TMs:** Können mehrere Berechnungen gleichzeitig ausführen.
- **Random-Access-Maschinen:** Modell gängiger Computer mit CPU, Registern und RAM.

Def. Folgende (partiellen) Funktionen sind **μ-rekursiv**:

1. Die *Nachfolgerfunktion* $\sigma : \mathbb{N} \to \mathbb{N},\ x \mapsto x + 1$
2. *Die Konstantenfunktion* $\kappa_c^k : \mathbb{N}^k \to \mathbb{N},\ (x_1, ..., x_k) \mapsto c$
3. *Die Projektionsfunktion* $\pi_i^k : \mathbb{N}^k \to \mathbb{N},\ (x_1, ..., x_k) \mapsto x_i$
4. *Die Komposition von μ-rekursiven Funktionen $f : \mathbb{N}^l \to \mathbb{N}$ und $g_1, ..., g_l : \mathbb{N}^k \to \mathbb{N}$, also* $h : \mathbb{N}^k \to \mathbb{N},\ (x_1, ..., x_k) \mapsto f(g_1(x_1, ..., x_k), ..., g_l(x_1, ..., x_k))$
5. *Für $k \geq 1$ und μ-rekursive Funktionen $f : \mathbb{N}^{k-1} \to \mathbb{N}$ und $g : \mathbb{N}^{k+1} \to \mathbb{N}$ die Funktion*
$$h : \mathbb{N}^k \to \mathbb{N},\ (x_1, ..., x_k) \mapsto \begin{cases} f(x_2, ..., x_k), & \text{falls } x_1 = 0 \\ g(h(x_1 - 1, x_2, ..., x_k), x_1 - 1, x_2, ..., x_k) \end{cases}$$
6. *Für $k \geq 1$ und eine μ-rekursive Funktion $f : \mathbb{N}^{k+1} \to \mathbb{N}$ die partielle Funktion*
$$h : \mathbb{N}^k \to \mathbb{N}, (x_1, ..., x_k) \mapsto min(M),$$
$$M := \{y \in \mathbb{N} \mid f(y, x_1, ..., x_k) = 0 \text{ und } f(z, x_1, ..., x_k) \text{ def. } \forall z < y\}$$

Bem. Aus der obigen Definition ergibt sich, dass folgende Funktionen μ-rekursiv sind:

- Die *Vorgängerfunktion* $\sigma^{-1} : \mathbb{N} \to \mathbb{N},\ x \mapsto \max\{0, x - 1\}$
- *Die Monusfunktion* $\dot{-} : \mathbb{N}^2 \to \mathbb{N},\ (x, y) \mapsto \max\{0, x - y\}$
- *Seien $If, Else : \mathbb{N}^k \to \mathbb{N}^l$ μ-rekursiv, dann auch*
$$If : \mathbb{N}^{k+1} \to \mathbb{N}^l,\ (z, x_1, ..., x_k) \mapsto \begin{cases} If(x_1, ..., x_k), & \text{falls } z = 0 \\ Else(x_1, ..., x_k), & \text{sonst} \end{cases}$$
- *Die Funktionen* $\min, \text{plus}, \text{mal} : \mathbb{N}^2 \to \mathbb{N}$

Bem. Die Klasse der μ-rekursiven Funktionen stimmt mit den von RAMs berechenbaren Funktionen überein.

Def. Eine Sprache heißt **entscheidbar**, wenn sie von einer TM akzeptiert wird und die TM auf jeder Eingabe hält.

Def. Eine Sprache heißt **semi-entscheidbar** oder rekursiv aufzählbar, wenn sie von einer TM akzeptiert wird.

Satz. Eine Sprache L ist genau dann entscheidbar, wenn L und $\overline{L}$ semi-entscheidbar sind.

Satz (Rice). Sei S eine Teilmenge aller Sprachen über Σ^*. Gibt es TMs M_1 und M_2 mit $L(M_1) \in S$ und $L(M_2) \notin S$, dann ist $\{\langle M \rangle \mid L(M) \in S\}$ unentscheidbar.

Def. Eine **uneingeschränkte/allgemeine Grammatik** G ist ein Tupel (V, Σ, P, S) wie bei bei kontextfreien Grammatiken mit dem Unterschied, dass die Produktion die allgemeinere Form $\alpha \longrightarrow \beta$ mit $\alpha \in (V \cup \Sigma)^+$ und $\beta \in (V \cup \Sigma)^*$ besitzen.

Bem. Ein unerwartet unentscheidbares Problem ist das sogenannte **Postsche Korrespondenzproblem** (Domino-Problem).

Def. Sei M eine TM, die auf allen Eingaben hält. Die Funktion f, die jedem $w \in \Sigma^*$ den Bandinhalt bis zum Kopf der TM nach Halten der TM zuordnet, heißt die von M **berechnete Funktion**.

Def. Seien L_1 und L_2 Sprachen über Σ_1 bzw. Σ_2. Man sagt, dass L_1 auf L_2 **abbildungsreduzierbar** ist (notiert $L_1 \leq_m L_2$) wenn es eine berechenbare Funktion $f : \Sigma_1^* \to \Sigma_2^*$ gibt, sodass für all $x \in \Sigma^*$ gilt:
$$x \in L_1 \Leftrightarrow f(x) \in L_2$$

Satz. Gilt $L_1 \leq_m L_2$ für zwei Sprachen L_1 und L_2 und ist L_2 (semi-)entscheidbar, so ist auch L_1 (semi-)entscheidbar.

Satz. Eine Sprache L ist genau dann semi-entscheidbar, wenn sie von einer uneingeschränkten Grammatik generiert wird.

Die Chomsky-Hierarchie

Def. Eine **kontextsensitive Grammatik** ist eine allgemeine Grammatik mit der Einschränkung, dass in jeder Produktion die Länge der rechten Seite mindestens so groß ist wie die Länge der linken Seite.

Satz. Kontextsensitive Sprachen sind entscheidbar.

#	Sprachklasse	Grammatik	Automat
0	semi-entscheidbar	uneingeschränkt	Turing-Maschine (TM)
1	kontextsensitiv	kontextsensitiv	Linear beschränkte NTM
2	kontextfrei	CFG	Kellerautomat
3	regulär	(reguläre Ausdrücke)	DFA/NFA

Bem. Zwischen den semi-entscheidbaren und den kontextsensitiven Sprachen gibt es die Zwischenklasse der entscheidbaren Sprachen, hier genannt D.

Abschlusseigenschaften:

#	$L_1 \cup L_2$	$L_1 \cap L_2$	$L_1 \cdot L_2$	$\overline{L}$	L^*
0	✓	✓	✓	✗	✓
D	✓	✓	✓	✓	✓
1	✓	✓	✓	✓	✓
2	✓	✗	✓	✗	✓
3	✓	✓	✓	✓	✓

Beispielsprachen:

Regulär: $\{w \in \Sigma^* \mid |w| gerade\}$

Kontextfrei: $\{0^n 1^n \mid n \geq 0\}$, $\{w \in \{0,1\}^* \mid |w|_0 = |w|_1\}$, $\{0^n 1^m \mid 0 \leq n \leq m\}$

Kontextsensitiv: $\{0^n 1^n 2^n \mid n \in \mathbb{N}\}$, $\{a^q 0^n 1^m 2^p \mid q, n, m, p \in \mathbb{N}, q = 0 \vee n = m = p\}$

Entscheidbar: $\{w\#w \mid w \in \{0,1\}^*\}$, $\{a^n b^m c^{nm} \mid n, m \in \mathbb{N}\}$, $W_{CFG} = \{\langle G, w \rangle \mid G$ ist eine CFG und G generiert $w\}$

Semi-entscheidbar:
$W_{TM} = \{\langle M, w \rangle \mid M$ ist eine TM und M akzeptiert $w\}$,
$H_\epsilon = \{\langle M \rangle \mid M$ ist eine TM und M hält auf $\epsilon\}$

Nicht semi-entscheidbar:
$D = \{\langle M \rangle \mid M$ ist eine TM mit $\langle M \rangle \notin L(M)\}$,
$Q_{TM} = \{\langle M_1, M_2 \rangle \mid M_1$ und M_2 sind TMs mit $L(M_1) = L(M_2)\}$