FSA - Cheat Sheet für Klausur

Komplexität-Theorie

- Algorithmen mit polynominaler Laufzeit gelten als handhabbar
- Algorithmen mit exponentieller Laufzeit gelten als nicht handhabbar

Komplexitätsklassen

P	Problem kann in polynominaler Zeit von einem deterministischem System berechnet werden
NP	Problem kann in polynominaler Zeit von einem nichtdeterministischem System berechnet werden

Reguläre Bezeichnungen

Σ (Sigma)	Alphabet, bzw. Menge an Buchstaben
Σ^*	Die Menge aller Wörter im Alphabet Σ
$L \subseteq \Sigma^*$	L ist eine Teilmenge und heißt formale Sprache über dem Alphabet von Σ
w	Ein Wort, mit Buchstaben a die aus dem Alphabet Σ sind: $a \in \Sigma$
w^R	Inverses Wort, Beispiel: Ist $w = abb$ dann ist $w^R = bba$
$\|w\|$	Betrag des Wortes, Anzahl der Buchstaben im Wort
ϵ	Ein leeres Wort, anders gesagt „NULL“

Kleenesche Hülle

Der kleenesche Stern oder auch nur Stern sagt aus, dass ein Wort oder ein Buchstabe beliebig oft wiederholt werden kann.

Beispielsweise kann der Ausdruck $R = 0^*1$ folgende Wörter bilden: $01,\ 0001,\ 000001$, usw...

Regex arbeitet z.B. mit diesem System.

Grammatik

G	Eine Grammatik bestehend aus: $G = (\Sigma, N, P, S)$
Σ	Ein Alphabet
N	Eine Menge an Variablen, die angibt, wie die Zeichen aus dem Alphabet Σ kombiniert werden dürfen, meist durch Operator | (oder) getrennt
P	Eine Menge an Produktionsregeln, welche beschreiben, wie bzw. welche Wörter aus dem Alphabet Σ gebildet werden können
S	Eine Startvariable mit $S \in N$, also eine Menge, die angibt, in welcher Reihenfolge die Variablen drankommen dürfen. Beispiel: Wenn $A \rightarrow 1 \mid 2$ und $B \rightarrow 10 \mid 100$ und $S \rightarrow A \mid BA$ sind, dann heißt das, dass ein Wort nur in der Reihenfolge z.B. $A \Rightarrow 1$ oder $BA \Rightarrow 101$ gebildet werden darf.

Veranschaulichendes Beispiel:

Römische Zahlen von 1 bis 9:	$A \rightarrow i \mid ii \mid iii \mid iv \mid v \mid vi \mid vii \mid viii \mid ix$
Römische Zahlen von 10 bis 90 (10er Schritte):	$B \rightarrow x \mid xx \mid xxx \mid xl \mid l \mid lx \mid lxx \mid lxxx \mid xc$
Startsymbol-Regel:	$S \rightarrow A \mid B \mid BA$
Bildet eine Grammatik wie folgt:	$G = (\Sigma, N, P, S)$ $G = (\{i, v, l, x, c\}, \{A, B, S\}, P, S)$ mit $P = \{A \rightarrow i \mid ii \mid \ldots, B \rightarrow x \mid xx \mid \ldots, S \rightarrow A \mid B \mid BA\}$

DEA Bezeichnungen

A	Automat bestehend aus: $A = (\Sigma, S, \delta, s_0, F)$
Σ	Eingabealphabet z.B. $\Sigma = \{0,1\}$
S	Eine endliche Menge von möglichen Zuständen z.B. $S = \{z_0, z_1, z_2\}$
δ (Delta)	Zustands-Übergangsfunktion, die einen einzelnen Zustandsübergang beschreibt, z.B.: $\delta(z_0, a) = z_2 \rightarrow$ heißt, ist man im Zustand z_0 und gibt ein a ein, so gibt die Funktion den Zustand zurück, den man nach der Eingabe erreicht, in dem Fall z_2
s_0	Startzustand, welcher Teil der Menge der Zustände sein muss, also $s_0 \in S$
F	Teilmenge an Endzuständen, z.B. $F = \{z_2\}$, wobei der Endzustand aus S sein muss

Unterschied DEA und NEA

Der DEA (Deterministischer Endlicher Automat) hat immer einen Weg, den er gehen kann, wenn ein Zeichen eingegeben wird. Es gibt keine Mengen in der Zustandstabelle, sondern nur einzelne Zeichen und Zustände (z.B. z_1 oder z_2).

Bei einem NEA (Nichtdeterministischer Endlicher Automat) kann die Eingabe eines Zeichens zu mehreren Zuständen führen, in der Zustandstabelle sind damit auch Mengen von Zuständen (z.B. $\{z_1, z_2\}$) möglich.

Erweiterte Übergangsfunktion

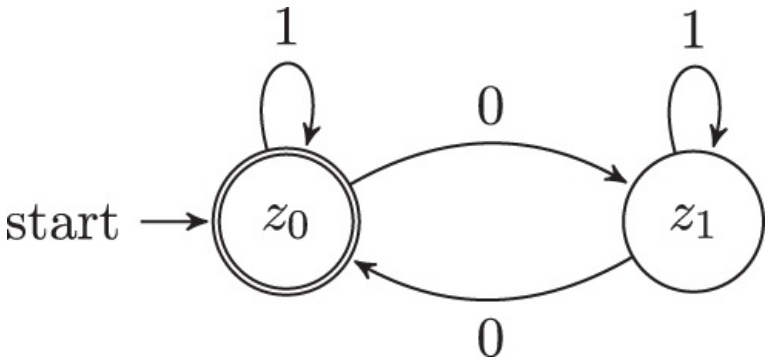
Bei der normalen Übergangsfunktion $\delta(z, a)$, wobei z der aktuelle Zustand und a ein Eingabezeichen ist, gibt diese den nächsten Zustand aus.

Die erweiterte Übergangsfunktion unterscheidet sich darin, dass aus a ein w wird, die Funktion also zurückgibt, welcher Zustand nach Eingabe eines ganzen Wortes erreicht wird. Dies sieht dann z.B. so aus: $\hat{\delta}(z, w)$.

Ist das Wort eine leere Menge, so ist der Zustand, der aus der Funktion folgt, der Eingangszustand: $\hat{\delta}(z, \epsilon) = z$.

Beispiel

Man hat einen folgenden DEA:



Beispiel aus „Theoretische Informatik für Dummies“

Die möglichen Eingaben sind also $0 \mid 1$. Möchte man nun die erweiterte Übergangsfunktion mit z.B. $\hat{\delta}(z_0, 01)$ auflösen, dann geht man folgendermaßen vor:

$$\begin{aligned}\hat{\delta}(z_0, 01) &= \delta(\hat{\delta}(z_0, 0), 1) \\ &= \delta(\delta(\hat{\delta}(z_0, \epsilon), 0), 1) \\ &= \delta(\delta(z_0, 0), 1) \\ &= \delta(z_1, 1) = z_1\end{aligned}$$

Man trennt die Zustände in der erweiterten Funktion in einzelne Delta-Übergangsfunktionen auf und löst diese dann von innen nach außen auf. Dabei ist der letzte Zustand die ϵ-Menge, denn dann löst sich die erweiterte Funktion auf und man ist im inneren in der „normalen“ Übergangsfunktion mit dem Zustand aus der erweiterten Funktion.

Anders gesagt: Man ließt von innen nach außen (wie ein Schreibkopf der von links nach rechts ließt) die einzelne Zeichen aus und schaut danach in welchen Zustand man kommt, bis man den letzten erreicht hat. Dieser Zustand ist dann die Ausgabe der erweiterten Übergangsfunktion.