Theoretische Informatik

Setup

- Möglichkeiten der Dokumentation:
  - Stift und Papier
  - Microsoft Word
  - Markdown (z. B. auf VSCode)
  - Latex (z. B. auf Overleaf)

Formale Sprachen

> Ein **Alphabet** Σ („Sigma“) ist eine endliche Menge von Symbolen. Ein **Wort über** Σ, $\omega\isin\Sigma^$ *("Omega"), ist eine beliebige Kombination von Zeichen des Alphabets. Das* ***leere Wort*** *wird mit ϵ ("Epsilon") bezeichnet. Eine* ***formale Sprache*** *L* ***über*** *Σ ist eine Teilmenge von Worten $L\subseteq\Sigma^$.*

> Die Regeln zur Bildung zulässiger Zeichenketten stellen die **Syntax** einer Sprache dar. Die Bedeutung eines Wortes nennt man **Semantik**.

Beispiel 1: Alle alphabetisch geordneten Worter aus a's und b's

$ \Sigma={a, b} \ \Sigma^*={\epsilon, a, b, aa, bb, ab, ba, aaa, aab, aba, baa, abb, bab, bba, bbb, aaaa, ...} \ L={\epsilon, a, b, aa, bb, ab, aaa, aab, abb, bbb, aaaa, ...} $

Beispiel 2: Palindrome

$ \Sigma={a, b, c, ..., x, y, z} \ L={\epsilon, abba, otto, reittier, ...} $

Beispiel 3: Primzahlen

$ \Sigma={0, 1, 2, 3, 4, 5, 6, 7, 8, 9} \ L={\epsilon, 2, 3, 5, 7, 11, 13, ...} $

Beispiel 4: Klammersprache (Dyck-Sprache D_2)

$ \Sigma={(,), [,]} \ L={\epsilon, (), [()], ()[()], ...} $

Grammatik

> Eine **Grammatik** G ist ein Viertupel (N, Σ, S, P). Sie besteht aus:
>
> - einer endlichen **Nonterminalmenge** N (auch Variablenmenge),
> - einem endlichen **Terminalalphabet** Σ,
> - einer **Startvariablen** $S\in N$,
> - einer endlichen Menge von **Produktionen** P. Jede Produktion hat die Form $u \rightarrow v$ mit $u \in (N \cup \Sigma)^* \setminus {\epsilon}$ und $v \in (N \cup \Sigma)^*$.

Beispiel 1:

$ G=(N, \Sigma, S, P) \ \text{mit} \ N={S} \ \Sigma={a, b} \ P={S\rightarrow aSb~|~ab} \ $

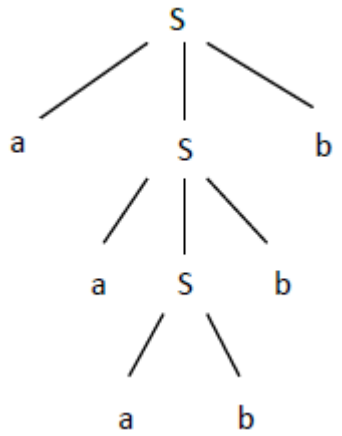
Erzeugte Sprache:

$L(G)={ab, aabb, aaabbb, ...}={a^nb^n|n\in N}$

Ableitung des Wortes "aaabbb":

$S \rightarrow aSb \rightarrow aaSbb \rightarrow aaabbb$

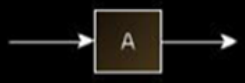
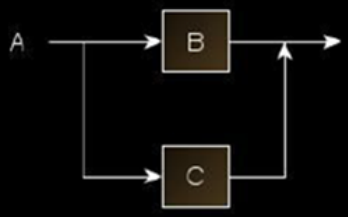
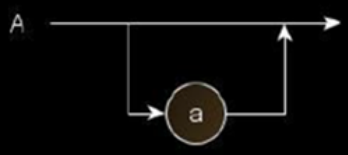
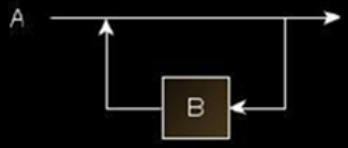
Ableitungsbaum des Wortes "aaabbb":



Chomsky-Hierarchie

> - Typ 0 (**allgemein**): beliebig
> - Typ 1 (**kontextsensitiv**): $u\rightarrow v \Rightarrow |u|\leq|v|$
> - Typ 2 (**kontextfrei**): $u\rightarrow v \Rightarrow u\in N$
> - Typ 3 (**regulär**): $u\rightarrow v \Rightarrow v=a|aB$

Syntaxdiagramme

Kontrollstruktur / Bedeutung	Grammatikregel in EBNF	Syntaxdiagramm
Terminalsymbol		
Nonterminalsymbol		
Sequenz	A → B C	
Selektion	A → B | C	
Rekursion	A → B A	
Option [Symbol] keinmal oder einmal	A → [a]	
Iteration {Symbol} keinmal oder beliebig oft	A → {B}	

Endliche Automaten

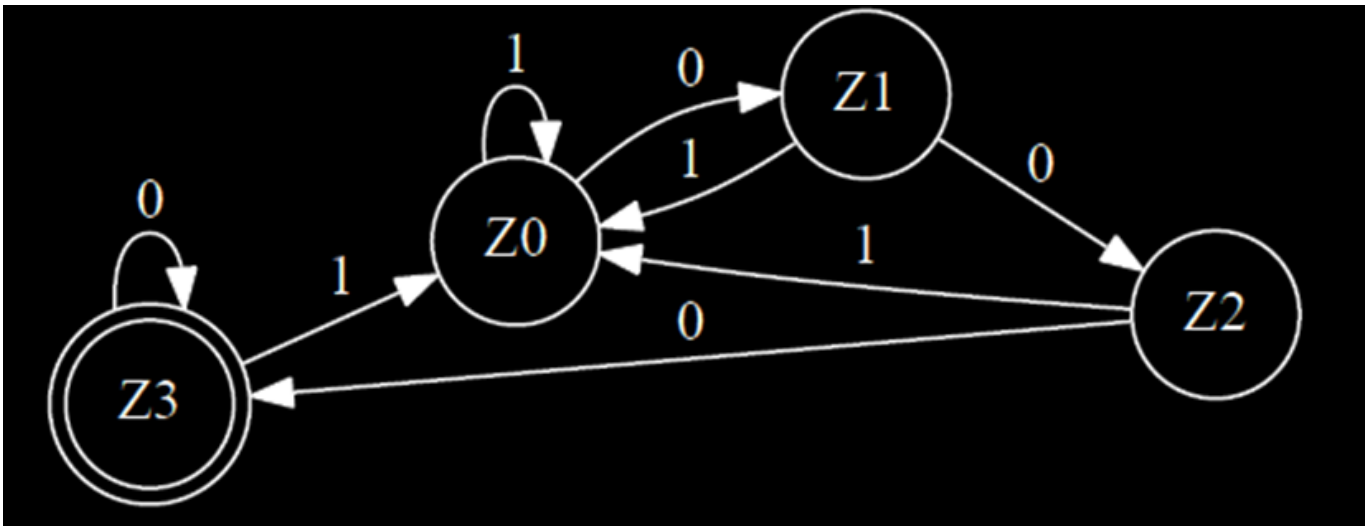
> Ein endlicher Automat ohne Ausgabe (Akzeptor) wird durch ein 5-Tupel $A=(Z,\Sigma, z_0, Z_E, \delta)$ spezifiziert.
>
> - Z ist die endliche und nichtleere Menge der **Zustände**.
> - Σ ist das **Eingabealphabet**, d.h. eine endliche, nichtleere Menge von Symbolen.
> - $z_0 \in Z$ ist der **Anfangszustand**.
> - Z_E ist die Menge der **Endzustände** mit $Z_E\subset Z$.
> - $\delta : Z \times \Sigma \rightarrow Z$ ist die **Zustandsübergangsfunktion**.
>
> Ein Wort, das den Automaten vom Start- in einen Endzustand überführt, gilt als akzeptiert.

> Ein endlicher Automat mit Ausgabe (Transduktor) wird durch ein 7-Tupel $A=(Z,\Sigma, Y, z_0, Z_E, \delta, \lambda)$ spezifiziert.
>
> - Z ist die endliche und nichtleere Menge der **Zustände**.
> - Σ ist das **Eingabealphabet**, d.h. eine endliche, nichtleere Menge von Symbolen.
> - Y ist die endliche und nichtleere Menge der möglichen **Ausgaben**.
> - $z_0 \in Z$ ist der **Anfangszustand**.

> - Z_E ist die Menge der **Endzustände** mit $Z_E\subset Z$.
> - $\delta : Z \times \Sigma \rightarrow Z$ ist die **Zustandsübergangsfunktion**.
> - λ ist die **Ausgabefunktion**, die jedem Zustand und Eingabezeichen eine Ausgabe zuordnet.
>
> Ein Wort, das den Automaten vom Start- in einen Endzustand überführt, gilt als akzeptiert.

Beispiel (ohne Ausgabe):

Zustandsübergangsdiagramm



Zustandsübergangstabelle

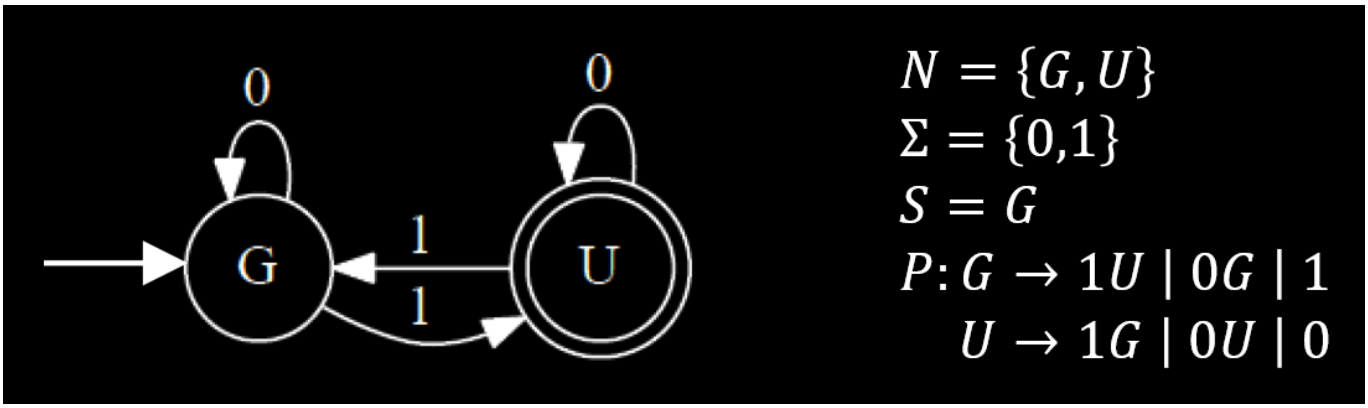
Zustand	Eingabe	Folgezustand
Z0	0	Z1
Z0	1	Z0
Z1	0	Z2
Z1	1	Z0
Z2	0	Z3
Z2	1	Z0
Z3	0	Z3
Z3	1	Z0

Zustand/Eingabe	0	1
Z0	Z1	Z0
Z1	Z2	Z0
Z2	Z3	Z0
Z3	Z3	Z0

Automaten und Grammatiken

> Zu jedem Akzeptor $A=(Z,\Sigma, z_0, Z_E, \delta)$ kann man eine reguläre Grammatik $G=(N,\Sigma, S, P)$ konstruieren und umgekehrt.
>
> - Jedes Nichtterminal ist ein Zustand
> - Die Alphabete sind identisch
> - Jede Produktion stellt einen Zustandübergang dar
> - Die Produktionen $Z\rightarrow \epsilon$ markieren Endzustände

Beispiel zu "Binärzahlen mit ungerader Parität":



Nichtdeterministische Endliche Automaten

> Ein nichtdeterministischer endlicher Automat (NEA) wird durch ein 5-Tupel $A=(Z,\Sigma, z_0, Z_E, \delta)$ spezifiziert.
>
> - Z ist die endliche und nichtleere Menge der **Zustände**.
> - Σ ist das **Eingabealphabet**, d. h. eine endliche, nichtleere Menge von Symbolen.
> - $z_0\in Z$ ist der **Anfangszustand**.
> - Z_E ist die Menge der **Endzustände** mit $Z_E\subset Z$.
> - $\delta: Z \times \Sigma\rightarrow Z^$ *ist die* ***Zustandsübergangsabbildung****, die von einem Zustand zu beliebig vielen Folgezuständen übergehen kann.* $Z^$ bezeichnet die Potenzmenge der Zustände.
>
> Ein Wort, für das es eine Möglichkeit gibt, den Automaten vom Start- in einen Endzustand zu überführen, gilt als akzeptert.

NEA $\Rightarrow$ DEA

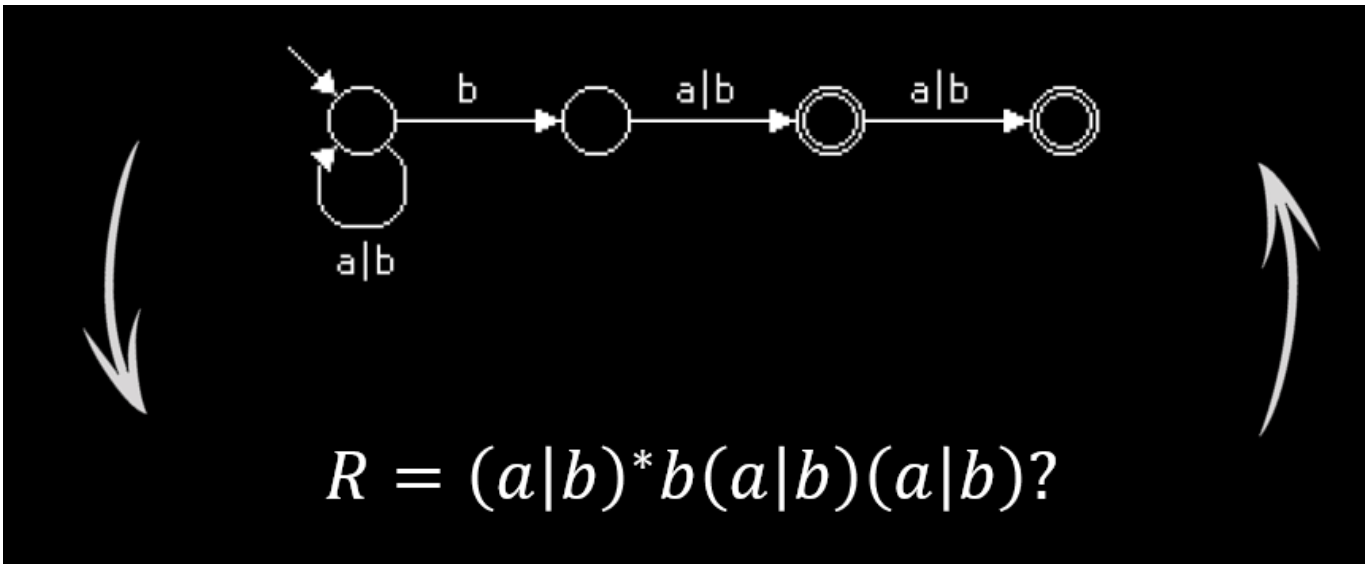
1. Man erstellt eine Zustandsübergangstabelle. Hierbei stehen in den Zellen nicht einzelne Folgezustände wie bei einem DEA, sondern Folgezustandsmengen. Ist ein Übergang nicht vorgesehen, trägt man die leere Menge ein.
2. Man erstellt die Zustandsübergangstabelle des äquivalenten DEAs indem man Zustandsmengen zu einem neuen Zustand zusammenfasst.
3. Man zeichnet den DEA entsprechend seiner Zustandstabelle, dabei werden die Zustände Endzustände, die einen Endzustand des NEAs enthalten.
4. Möchte man einen vollständigen DEA aufführen, der in allen Zuständen alle Eingaben berücksichtigt, muss man noch einen Fehlerzustand, der kein Endzustand sein darf, einführen.

Reguläre Ausdrücke

> Mit einem regulären Ausdruck wird (alternativ zur Tabelle oder zum Graphen) eine formale Sprache definiert.
>
> - Ø ist ein regulärer Ausdruck. Er beschreibt die leere Wortmenge {}.
> - $\boldsymbol{\epsilon}$ ist ein regulärer Ausdruck. Er beschreibt die Wortmenge $\{\epsilon\}$, in der nur das leere Wort vorkommt.
> - Für jedes $a \in \Sigma$ ist $\boldsymbol{a}$ ein regulärer Ausdruck. Der reguläre Ausdruck a beschreibt die Wortmenge $\{a\}$.
> - Wenn α und β reguläre Ausdrücke sind, dann ist auch die Konkatenation $\boldsymbol{\alpha\beta}$ ein regulärer Ausdruck. Wenn α die Wortmenge A und β die Wortmenge B beschreibt, dann beschreibt die Konkatenation $\alpha\beta$ die Menge $\{ab | a \in A \ und \ b \in B\}$ aller Wörter, die mit einem Wort aus A beginnen und mit einem Wort aus B enden.
> - Wenn α und β reguläre Ausdrücke sind, dann ist auch die Alternative $\boldsymbol{\alpha|\beta}$ ein regulärer Ausdruck. Wenn α die Wortmenge A und β die Wortmenge B beschreibt, dann beschreibt die Alternative $\alpha|\beta$ die Menge $\{\omega | \omega \in A \ oder \ \omega \in B\}$ aller Wörter, die in A oder in B vorkommen.
> - Wenn α ein regulärer Ausdruck ist, dann ist auch die Iteration α^* ein regulärer Ausdruck. Wenn α die Wortmenge A beschreibt, dann beschreibt die Iteration α^* die Menge A^* aller Wörter, die durch endlich häufiges Aneinanderfügen (auch keinmal) von Wörtern aus A entstehen.

Regulärer Ausdruck	Wortmenge / Formale Sprache	Beschreibung: "Die Sprache, die ..."
$\varnothing$	$\{\}$	... nichts enthält (leere Menge).
ϵ	$\{\epsilon\}$	... nur das leere Wort enthält.
0	$\{0\}$	... nur eine 0 enthält.
1	$\{1\}$	... nur eine 1 enthält.
10	$\{10\}$	... nur das Wort "10" enthält.
1|0	$\{1,0\}$	... nur aus einer 0 und einer 1 besteht.
1*	$\{\epsilon, 1, 11, 111, 1111, ...\}$	... aus allen Wörtern mit beliebig vielen (auch gar keine) 1en besteht.
01*	$\{0, 01, 011, 0111, 01111, ...\}$	... aus allen Wörtern besteht, bei denen auf eine 0 beliebig viele (auch gar keine) 1en folgen.
0*1*	$\{\epsilon, 0, 00, ..., 1, 01, 001, ..., 11, 011, 0011, ...\}$	... aus allen Wörtern besteht, bei denen auf beliebig viele (auch gar keine) 0en beliebig viele (auch gar keine) 1en folgen.
0*|1*	$\{\epsilon, 0, 00, 000, ..., 1, 11, 111, ...\}$	... aus allen Wörtern mit entweder beliebig vielen (auch gar keinen) 0en oder beliebig vielen (auch gar keinen) 1en besteht.
0|(1(0|1)*)	$\{0, 1, 10, 11, 100, 101, 110, 111, ...\}$	... die Binärzahlen darstellt.
1+	$\{1, 11, 111, 1111, ...\}$	... aus Wörtern nur aus 1en mit mindestens einer 1 besteht. (nicht verwechseln mit 1*)
1?	$\{\epsilon, 1\}$	... nur aus 1 oder dem leeren Wort besteht. Das ? bedeutet, dass das Zeichen davor optional vorkommen kann.

Akzeptor $\Leftrightarrow$ Regulärer Ausdruck



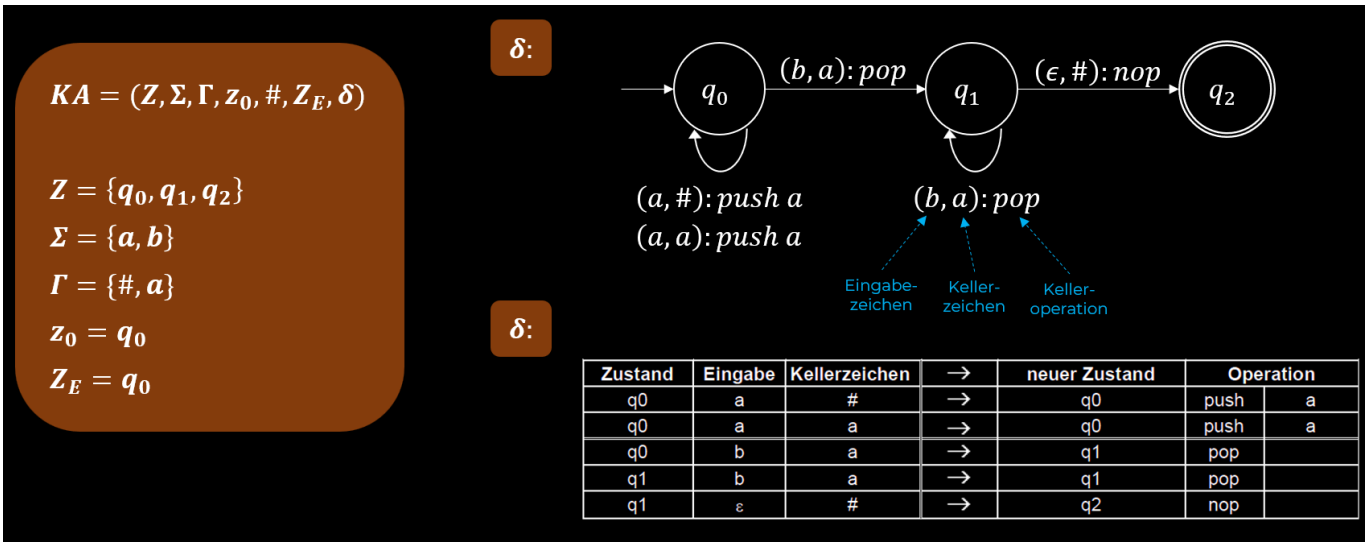
Satz über reguläre Sprachen

> Die Klasse der Sprachen, die mit einer regulären Grammatik beschrieben werden können, ist identisch mit der Klasse der Sprachen, die mit einem regulären Ausdruck beschrieben werden können. Sie ist ebenso identisch mit der Klasse der Sprachen, die von deterministischen endlichen Automaten bzw. von nichtdeterministischen endlichen Automaten erkannt werden können.

Kellerautomat

> Ein nichtdeterministischer Kellerautomat (KA) wird durch ein 7-Tupel $KA=(Z, \Sigma, \Gamma, z_0, #, Z_E, \delta)$ definiert.
>
> - Z ist eine nichtleere endliche Menge – die **Zustandsmenge**.
> - Σ ist eine nichtleere endliche Menge – das **Eingabealphabet**.
> - Γ ist eine nichtleere endliche Menge – das **Kelleralphabet**.
> - $z_0\in Z$ ist der **Anfangszustand**.
> - $# \in \Gamma$ ist das **Kellervorbelegungszeichen**.
> - Z_E ist die Menge der **Endzustände** mit $Z_E\subset Z$.
> - $\delta: Z\times (\Sigma\cup{\epsilon})\times \Gamma\rightarrow (Z\times\Gamma^*)^$ ist die **Zustandsübergangsabbildung**, die ausgehend von aktuellen Zustand, vom Kellerzeichen und vom Eingabezeichen in einen (oder mehrere) neue Zustände unter Ausführung einer Kelleroperation führt.
>
> Ein Wort, für das es eine Möglichkeit gibt, den Automaten vom Start- in einen Endzustand zu überführen, sodass der Keller am Ende leer ist, gilt als akzeptert.
>
> **Speichereinheit**: LIFO, unendlich groß, leer enthält immer # **Steuereinheit**: push, pop, nop

Beispiel:



Zustand	Eingabe	Kellerzeichen	→	neuer Zustand	Operation	
q0	a	#	→	q0	push	a
q0	a	a	→	q0	push	a
q0	b	a	→	q1	pop	
q1	b	a	→	q1	pop	
q1	ε	#	→	q2	nop	

Satz über kontextfreie Sprachen

> Die Menge der Sprachen, die ein Kellerautomat erkennt, ist gleich der Menge der Sprachen, die durch eine kontextfreie Grammatik erzeugt werden können.