Einführung

Es gibt in der Informatik, aber nicht nur da, die Frage, ob Texte, (allg.: Ketten von Zeichen) syntaktisch korrekt sind oder nicht. Auch die Frage nach der Suche bestimmter Muster (Pattern), die in allgemeiner Weise beschrieben werden müssen (z. B. durch reguläre Ausdrücke) stellt die Frage, wie man die syntaktische und inhaltliche Korrektheit von Texten zeigen kann. Wir kennen schon Automaten, die in der Lage sind, einzelne Worte auf Gültigkeit (im Sinne von Zugehörigkeit zu einer Sprache) zu prüfen. Aber wie kann man eine Sprache, die meist unendlich viele Worte beinhaltet (abzählbar unendliche ist) vollständig beschreiben?

Beispiel 1 *Ist der Text korrekt?*

```
\textbf {f geh{\"o}rt zu der Komplexit{\"a}tsklasse der Ordnung O(g)}
\newline $\mathcal{O}$(g) = \{f: $\mathbb{N}$ $\rightarrow$ $\mathbb{R^
+}$ \big| $\exists$ n\textsubscript{0} $\in$ $\mathbb{N}$ $\wedge$ c
$\in$ $\mathbb{R^+}$: $\forall$ n $\ge$ n\textsubscript{0}: f(n) $\leq$
 c$\cdot$g(n)\}\newline\newline\pause
Beispiele: $\mathcal{O}$(1), $\mathcal{O}$(n), $\mathcal{O}$$\text{(n}^2
)$, $\mathcal{O}$(n$\cdot$log n), $\mathcal{O}$($2^n$) \newline\newline
\pause
```

Hier muss man als Autor anhand der gültigen LaTeX-Regeln prüfen, ob der Text gültig ist.

Beispiel 2 *Ist der Text korrekt?*

```
zaehlen:: Eq a => a -> [a] -> [b]
zaehlen a [] = 1
zaehlen a [x:xr] | == x = 1 + zaehlen a xr
                 | otherwise = zaehlen xr
```

Hier gelingt die Überprüfung der Korrektheit schon besser, weil man als Programmierer die Syntaxregeln von Haskell kennt. Aber auch hier besteht das Problem des Baus eines Automaten (Compilers), der die Überprüfung der Haskellsyntax und -semantik übernimmt.

Eine Frage, die eng mit der Frage der Korrektheit von Texten zusammenhängt, ist die Frage der Beschreibung der Sprache $L(\Sigma)$ eines Automaten über dem Alphabet Σ.

Beispiel 3 *Die Sprache aus Ipogesien*

```
ipigisi, isipigisisi, ipisigisisi, isisipigisisisi, isipisigisisisi,
ipisisigisisisi, ...
```

Ist das Wort isipisisigisisisisisi ist Wort der Ipogesien-Sprache? Beschreibe **alle** Worte der Sprache! ⇒ Besprechung an der Tafel!

Wichtige Grundbegriffe

Definition 1: Alphabet, Wort, Formale Sprache

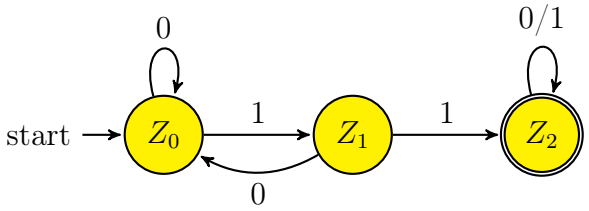
Ein **Alphabet** ist eine nicht-leere, endliche Menge Σ (deren Elemente wir Buchstaben oder Zeichen nennen). Ein Σ - **Wort** ist eine endliche Sequenz von Buchstaben aus $\Sigma, w = a_1 \dots a_n$ mit $a_i \in \Sigma$. Das **leere Wort** wird mit ϵ bezeichnet. Die Menge aller Σ - Wörter wird mit Σ^* bezeichnet. Eine **formale Sprache** bzw. Σ - **Sprache** L(Σ) ist eine Teilmenge $L \subseteq \Sigma^*$, d.h. eine Menge von Σ - Wörtern. Σ^+ steht für die Menge der von ϵ verschiedenen (also nicht-leeren) Σ - Wörter.

Mit |w| bezeichnen wir die Länge des Wortes w; für $w = a_1 \dots a_n$ ist |w| = n. Das leere Wort ϵ ist das einzige Wort der Länge 0. Σ^n steht für die Menge aller Σ - Wörter der Länge n $\in \mathbb{N}, \Sigma^n = \{w \in \Sigma^* | \, |w| = n\}$. Dabei gilt: $\Sigma^0 = \{\epsilon\}$.

Übung 1: Gegeben sei das Alphabet $\Sigma = \{x, y, z\}$. Gib alle Wörter $w \in \Sigma^2$ bzw. $w \in \Sigma^3$ an, die zu $\mathcal{L} = \{w \in \Sigma^* |$ w ist eine lexikografisch geordnete Buchstabenfolge$\}$ gehören.

Übung 2: Entwickle einen Akzeptor, der die Sprache $\mathcal{L} = \{ab^n c \,|\, n \geq 0\}$ akzeptiert.

Übung 3: Gegeben ist der abgebildete Automat $\mathcal{A}$. Gib die Sprache $\mathcal{L}(\mathcal{A})$ an.



Übung 4: Entwickle ein ZÜG eines DEAs zu folgender Sprache $\mathcal{L}(\mathcal{A}) = \{$w | w enthält gerade Anzahl von Nullen und Einsen.$\}$

Sprachen sind im allgemeinen unendlich: sie enthalten unendlich viele Wörter. Wir brauchen endliche Darstellungen → Grammatiken

Grammatiken

Als nächstes wird eine weitere Möglichkeit vorgestellt, Automaten und deren Sprache zu beschreiben. Es wird sich zeigen, dass dieser Mechanismus geeignet ist, das Problem der vollständigen Beschreibung der abzählbar unendlichen Menge der gültigen Wörter zu erfassen.

Definition 2: Grammatik

Eine **Grammatik G** wird durch ein 4-Tupel (N, T, S, P) spezifiziert:

- N ist die Menge der Nichtterminalsymbole.
- T ist die Menge der Terminalsymbole.
- S ∈ N ist das Startsymbol.
- P ist die Menge der Regeln oder Produktionen, die festlegt, welche Nichtterminalsymbole wie ersetzt werden.

Hinweis 1: Durch Umbenennung ist oft auch G ein 4-Tupel der Form (V, Σ, S, P).
Hinweis 2: Mathematisch bedeutet das Regelsystem $P \subseteq (N \cup T)^+ \times (N \cup T)^*$ und $N \cap T = \emptyset$. Die linke Seite einer Produktionsregel enthält mindestens ein Nichtterminal.

Die Anwendung von Produktionen nennt man auch *direkte Ableitung* bzw. *Ableitung*.
Seien $u \in (N \cup T)^+$ und $v \in (N \cup T)^*$ zwei Worte dann ist

$u \to v :=$ ∃ Regel in P $y \to y'$ mit $u = xyz \to xy'z = v$ mit $x, z \in (N \cup T)^*$.

Beispiel 1:

Sei $G = (N, T, S, P)$ mit

$N = \{S_0, A, B\}$

$T = \{0, 1\}$

$S = S_0$

P = { $S_0 \to 1S_0 \,|\, 0A$,
A $\to 1S_0 \,|\, 0B$,
B $\to 1S_0 \,|\, 0B \,|\, 0$}

Ableitung: $S_0 \to 1S_0 \to 10A \to 100B \to 1000$.

Hinweis: Nichtterminale werden mit großem Buchstaben A, B, X, ... angegeben, Terminale mit Kleinbuchstaben a, b, ... oder Ziffern 0, 1, ...

Mit Hilfe der Ableitung kann man die Sprache $\mathcal{L}(\mathcal{G})$ wie folgt definieren:

Definition 3: Formale Sprache

$\mathcal{L}(\mathcal{G}) = \{w \in T^* | S \to^* w\}$, wobei $\to^*$ ist eine mehrfache Ableitung.

Hinweis: Man nennt $\to^*$ auch die transitive Hülle von $\to$.

Beispiel 2: Gib die Sprache $\mathcal{L}(\mathcal{G})$ durch korrekte Ableitung an.

Sei G = (N,T,S,P) mit

N = $\{S, B, C\}$

T = $\{a, b, c\}$

P = $\{S \to aSBC, S \to aBC, CB \to BC, aB \to ab, bB \to bb, bC \to bc, cC \to cc\}$

Lösung:

Beispielableitung $S \to aSBC \to aaBCBC \to aabCBC \to aabBCC \to aabbCC \to aabbcC \to aabbcc$ also: $\mathcal{L}(\mathcal{G}) = \{a^n b^n c^n | n \in \mathbb{N}\}$

Übung 1: Gib die Sprache $\mathcal{L}(\mathcal{G})$ an.

Sei G = (N, T, S, P) mit

N = $\{S\}$

T = $\{a, b\}$

P = $\{S \to aSb | ab\}$

Lösung:

Übung 2: Gib die Sprache $\mathcal{L}(\mathcal{G})$ an.

Sei G = (N, T, S, P) mit

N = $\{S, X, Y\}$

T = $\{a, b\}$

P = $\{S \to XSY | \epsilon,$
$XY \to YX,$
$X \to a,$
$Y \to b\}$

Lösung:

Übung 3: Gib die Sprache $\mathcal{L}(\mathcal{G})$ an.

Lösung:

Sei G = (N, T, S, P) mit
N = { }
T = { }
P = $\{S \to bSb \,|\, a\}$

Übung 4: Finde zur Sprache $\mathcal{L}(\mathcal{G})$ eine passende Grammatik.
Gegeben ist die folgende Sprache $\mathcal{L}(\mathcal{G}) = \{ww^R | \, w \in \{a, b\}^*$, w^R ist das Spiegelbild von w$\}$

Hinweis: Man kann zeigen, das die von einem Automaten erkannte Sprache $\mathcal{L}(\mathcal{A})$ dieselbe Sprache ist, wie die Sprache $\mathcal{L}(\mathcal{G})$, wenn dabei G die aus dem Automaten abgeleitete Grammatik ist.

Übung 5: Himmlische Sprache *Frohlocke*. Übung siehe Tafel! (Steht noch aus!)

Eine wichtige Frage ist das **Wortproblem:** Wie prüft man, ob ein Wort w zur Sprache $\mathcal{L}(\mathcal{A})$ gehört?
Idee: Finde im zu prüfenden Wort Muster, die den rechten Seiten der Produktionen entsprechen und ersetze diese durch die linken Seiten. Tue dies solange, bis du das Startsymbol erhältst.

Beispiel 3: Gegeben sind die Produktionsregeln der Grammatik aus Beispiel 1:
P = $\{S_0 \to 1S_0 | \, 0A, A \to 1S_0 | \, 0B, B \to 1S_0 | \, 0B | \, 0\}$
Gilt w = 111000 $\in \mathcal{L}(\mathcal{A})$?
Lösung: 111000 $\leftarrow$ 11100B $\leftarrow$ 1110A $\leftarrow$ 111S_0 $\leftarrow$ 11S_0 $\leftarrow$ 1S_0 $\leftarrow$ S_0 Korrekt!
Prüfe: w = 1011 $\in \mathcal{L}(\mathcal{G})$?

Reguläre Grammatiken

Es geht nun in diesem Kapitel um spezielle Sprachen und deren Beschreibungen. Reguläre Sprachen sind die am meisten eingeschränkten Sprachen. Sie unterliegen besonderen Forderungen. Sie haben große Bedeutung in Textverarbeitungen und bei der lexikalischen Überprüfung in der Programmierung.
Die Beschreibung von *Formalen Sprachen* durch Grammatiken sind eine weitere Möglichkeit, die es gestattet unendliche Mengen von Worten einer *Formalen Sprachen* zu

beschreiben. Damit gibt es folgende Beschreibungsmöglichkeiten:

- Aufzählung der Wörter (nur bei endlichen Mengen von Wörtern vollständig),
- akzeptierende Automaten,
- Mengenausdrücke: z. B. $\{a^n \mid \text{n Primzahl}\}$,
- reguläre Ausdrücke (für reguläre Sprachen) oder
- Grammatiken

Es gibt verschiedene Grammatiken, die verschiedene *Formale Sprachen* beschreiben. Der amerikanische Linguist Noam Chomsky hat 4 Klassen von Grammatiken beschrieben.

Typ	Bezeichnung	Eigenschaften
Typ 0		keine Einschränkungen
Typ 1	kontextsensitiv	für alle Regeln $w_1 \to w_2$ gilt: $\|w_1\| \leq \|w_2\|$ bzw. auch so formuliert: Seien $x, y, z \in (N \cup T)^*$ und $A \in N$. Dann gilt $xAz \to xyz$, $\|y\| > 0$, zusätzlich: $S \to \epsilon$
Typ 2	kontextfrei	wie Typ 1 und $w_1 \in N$, d. h. links taucht nur ein Nichtterminal auf
Typ 3	(rechts)regulär	wie Typ 2 und zusätzlich $w_2 \in T \cup TN$, d.h. jede rechte Seite besteht aus genau einem Terminal und höchstens einem weiteren Nichtterminal.

Und nun sollen die einfachsten Grammatiken beschrieben werden.

Definition 4: Rechtsreguläre Grammatik

Eine **rechtsreguläre Grammatik G** wird durch ein 4-Tupel (N, T, S, P) spezifiziert:

- N ist die Menge der Nichtterminalsymbole.
- T ist die Menge der Terminalsymbole.
- S ∈ N ist das Startsymbol.
- P ist die Menge der Regeln oder Produktionen, die von der Form P = $\{A \to aB \mid B \mid a \mid \epsilon\,,\ B \to \epsilon\}$ A, B ∈ N sind.

Es sind also nur Ableitungsregeln erlaubt, die rechts höchstens ein Nichtterminal hinter einen Terminalsymbol enthalten. Man nennt diese Grammatik *rechtsregulär*.
Sind die Regeln der Form P = $\{A \to Ba \,|\, B \,|\, a \,|\, \epsilon\,,\ B \to \epsilon\}$, so ist die Grammatik *linksregulär*.

Aufgabe: Ordne in den obigen Übungen die Grammatiken entsprechend der Chomsky-Hierarchie zu.
Lösung:

Hinweis: Ist eine Sprache regulär, so existiert eine linksreguläre und rechtsreguläre Grammatik und beide beschreiben die selbe reguläre Sprachen.

Lemma (ohne Beweis): Die Sprachen $\mathcal{L}(\mathcal{G})$, die von einer regulären Grammatik erzeugt werden und die Sprachen $\mathcal{L}(\mathcal{A})$ eines endlichen Akzeptors sind äquivalent.

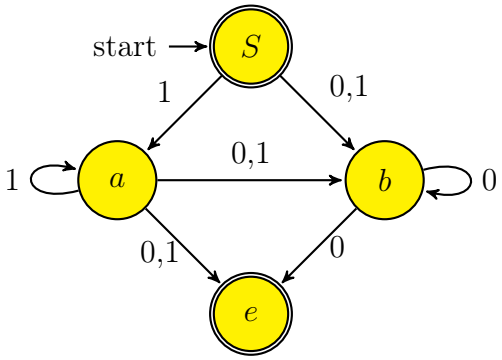
Wie kann man aus einer regulären Grammatik einen Akzeptor entwickeln und andersherum? Es lässt sich schnell herausfinden, dass die Überführung einer regulären Grammatik meist in einem NEA endet. Jedoch ist jeder NEA in einen DEA transformierbar.

Beispiel:

Sei G = (N, T, S, P) mit

$$\begin{aligned} \mathrm{N} &= \{A, B\} \\ \mathrm{T} &= \{0, 1\} \\ \mathrm{P} &= \{S \to \epsilon \,|\, 1A \,|\, 1B \,|\, 0B, \\ &\quad\ A \to 1 \,|\, 0 \,|\, 1A \,|\, 1B \,|\, 0B, \\ &\quad\ B \to 0 \,|\, 0B\} \end{aligned}$$

Der zugehörige Automat sieht wie folgt aus:



Fragen: Welcher Automatentyp liegt vor? Welche Transformationsregeln lassen sich formulieren? Entwickle einen äquivalenten DEA!
Zur Definition eines NEA siehe nächstes Kapitel.

Satz 1: Äquivalenz von DEA und zugehöriger regulären Grammatik

Jeder DEA mit $A = (X, Z, Y, g, z_0)$ bzw. $A = (Z, \Sigma, Y, \delta, z_0)$ lässt sich in eine reguläre Grammatik $G = (N, T, S, P)$ bzw. $G = (V, \Sigma, S, P)$ transformieren.
Zu jeder regulären Grammatik kann ein entsprechender DEA entwickelt werden, der die selbe Sprache $\mathcal{L}$ erkennt.

Vorgehensweise beim Übergang vom DEA zur regulären Grammatik:

1. Notiere die Terminalmenge T anhand der Eingabesymbole aus X des DEA.
2. Notiere die Nichtterminalmenge N anhand der Zustandsmenge Z des DEA. S_0 sei dabei z_0.
3. Notiere zu jedem Übergang g im Automaten eine eigene Regel für P.
4. Minimiere ggf. durch Zusammenfassen geeigneter Regeln.

Reguläre Ausdrücke

Studiere den ausgeteilten Text zu den *Regulären Ausdrücken*! Gib eine Übersicht über die wichtigsten EBNF-Operatoren! Finde ein leichtes und ein schwereres Beispiel für reguläre Ausdrücke und formuliere diese als Aufgabe deinen Mitschülern.
Dies könnte so aussehen: Welche Worten werden durch die folgenden beiden regulären Ausdrücke beschrieben?

a) ab*a
b) a(a|b|c)*

Es werden die folgenden Operatoren in der EBNF unterschieden: Seien r und s reguläre Ausdrücke.

- rs bedeutet die Konkatenation beider Elemente
- (r|s) bedeutet die Alternative r oder s mit gleichzeitiger Gruppierung durch die Klammer (wird manchmal durch + ausgedrückt)
- r* ist die mehrfache Wiederholung („gar nicht oder beliebig oft“) (Kleene-Stern)
- Optionsklammern {}, Wiederholungsklammern [], Gruppierungsklammern ()
- [a-m] bedeutet Zeichen von a bis m sind möglich
- [^a] ein beliebiges Zeichen außer a (Negation)
- + der vorangegangene Ausdruck kommt mind. einmal oder mehrfach vor, entspricht dem {1,}, z. B. [ab]+ entspricht „a“, „b“, „aa“ usw.
- ? der vorangegangene Ausdruck kommt höchstens einmal vor (kann, muss aber nicht)
- {n,} der vorangegangene Ausdruck kommt mind. n-mal vor
- {n,m} der vorangegangene Ausdruck kommt mind. n- mal aber höchstens m-mal vor
- {,m} der vorangegangene Ausdruck kommt höchstens m-mal vor

Beispiele:
1. 0*(10*10*)* - bezeichnet die Sprache aller Wörter über {0,1}, die eine gerade Anzahl von 1en enthalten.
2. 1*0* - bezeichnet die Wörter über {0,1} die nicht 01 als Unterwort enthalten.
3. (0|1(01*0)*1)* - bezeichnet die durch drei teilbaren Binärzahlen.

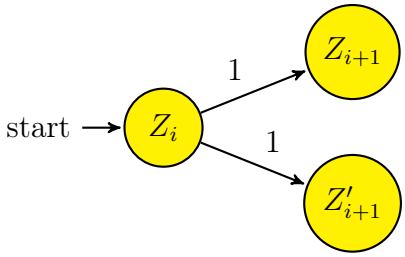
Nichtdeterministische endliche Automaten - NEA

Definition 5: Nichtdeterministische Automaten

Sei $\mathcal{P}(\mathcal{Z})$ die Potenzmenge von der Zustandsmenge Z. Ein **NEA** A ist ein 5-Tupel $A = (Z, \Sigma, \delta, z_0, Y)$ mit:

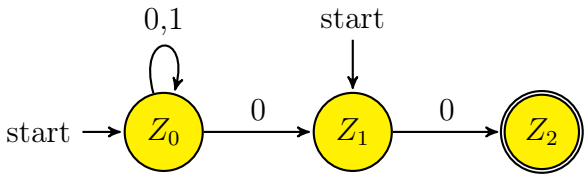
- Z ist die endliche Menge der Zustände,
- Σ, das Eingabealphabet,
- $\delta : Z \times \Sigma \to \mathcal{P}(\mathcal{Z})$
- $S \subseteq Z$, die Menge der Startzustände,
- $E \subseteq Z$, die Menge der Endzustände.

Somit können NEA's folgende Übergänge im Zustandsgraphen enthalten:



Ein Wort gilt als akzeptiert, wenn mindestens einen Pfad von einem Startzustand zu einem Endzustand gibt.

Beispiel:



Die akzeptierte Sprache $\mathcal{L}(\Sigma) = \{w \in \{0,1\}^* \mid w = 0 \text{ oder } w \text{ endet mit } 00\}$

Übungen

Die nun folgenden Übungen geben einen Überblick über die in der Klausur möglichen Aufgaben. Die Lösungen können bei mir abgeholt werden. Aber nur im Austausch zu eurem Lösungsangebot.

1. Aufgabe: Gegeben ist ein DEA $A = (\{z_0, z_1, z_2, z_3\}, \{a, b\}, \delta, z_0, \{z_0\})$ und die folgende Zustandsübergangstabelle ZÜT.
(Wir hatten diese Tabellen nicht so ausführlich besprochen, weil diese sehr intuitiv verständlich sind, wie diese Aufgabe zeigen wird.)

δ	z_0	z_1	z_2	z_3
a	z_1	z_2	z_3	z_0
b	z_3	z_0	z_1	z_2

a) Zeichne einen ZÜD
b) Gib die Sprache $L(\Sigma)$ an!

Lösung:

2. Aufgabe: Sei $A = (\{z_0, z_1, z_2, z_3\}, \{a, b\}, \delta, z_0, \{z_3\})$ ein DEA mit folgender Überführungsfunktion δ:

δ	z_0	z_1	z_2	z_3
a	z_1	z_1	z_3	z_3
b	z_0	z_2	z_0	z_3

Gib eine rechtsreguläre Grammatik G an.
Lösung:

3. Aufgabe: Gegeben ist ein DEA. Zeige durch Angabe eines Algorithmus, dass zu jedem DEA eine reguläre Grammatik existiert, die die selbe Sprache $\mathcal{L}(\Sigma)$ beschreibt.
Lösung:

4. Aufgabe: Welche Worte werden durch den regulären Ausdruck (0|(0|1)*00) beschrieben?
Lösung:

5. Aufgabe: Gegeben ist der reguläre Ausdruck ((0|(0|1)*00)). Gib mindestens zwei Beispielworte der Sprache $\mathcal{L}(\Sigma)$ an. Formuliere die Sprache $\mathcal{L}(\Sigma)$ (in mathematischer No-

tation mit Mengenschreibweise oder in Worten)!
Lösung:

6. Aufgabe: Finde eine Grammatik für die Sprache über dem Alphabet $\Sigma = \{0, 1\}$, deren Wörter gleich viele Nullen wie Einsen enthalten. Welche Grammatik liegt vor?
Lösung:

7. Aufgabe: Gegeben ist eine Grammatik mit den Produktionen
$\mathrm{P} = \{S \to 0S \,|\, 1A \,|\, 1,$
$A \to 0B \,|\, 1A \,|\, 0 \,|\, 1,$
$B \to 0S \,|\, 1A \,|\, 1\}$
Welche Sprache wird beschrieben? Entwickle einen DEA!
Lösung:

8. Aufgabe: Gib einen regulären Ausdruck für die Sprache über dem binären Alphabet an, deren Wörter als letztes oder vorletztes Zeichen eine 1 haben! Entwickle einen DEA.
Lösung:

9. Aufgabe: Entwirf einen DEA für den folgenden regulären Ausdruck $0(0|1)^\star$ sowohl zeichnerisch als auch formal. Gib auch eine Grammatik an.
Lösung:

10. Aufgabe (schwer): Gegeben sei die Sprache
$\mathrm{L} = \{w \in \{0,1\}^+ | \, \forall u, v \in \{0,1\}^* : w = u1v \Rightarrow \exists x, y \in \{0,1\}^* :$
$(u = x1 \wedge v = 00y) \vee v = 100y\}$
a) Gib eine rechtslineare Grammatik G an, so dass gilt $\mathcal{L}(\mathcal{G}) = \mathrm{L}$.
b) Konstruiere einen vollständigen DEA mit $\mathcal{L}(\Sigma) = \mathcal{L}(\mathcal{G}) = \mathrm{L}$.