

GBI Definitionen

Formale Sprachen

Eine formale Sprache L ist eine Menge aus Wörtern.

zB.: $L_{ab} = \{a, b\}$ ist die Sprache mit den Wörtern a und b

- Die Potenz einer Sprache ist definiert als:

$$\begin{aligned} L^0 &= \{\epsilon\} \\ L^{n+1} &= L \cdot L^n \end{aligned}$$

- Sprache konkateniert mit einer Sprache ist einfach jede mögliche Kombination der Wörter. Als Beispiel nehmen wir wieder $L_{ab} = \{a, b\}$

$$L_{ab} \cdot \{aa, bb\} = \{aaa, bbb, baa, abb\}$$

$$\begin{aligned} L_{ab}^3 &= L_{ab} \cdot L_{ab} \cdot L_{ab} \\ &= \{a, b\} \cdot \{a, b\} \cdot \{a, b\} \\ &= \{aa, bb, ab, ba\} \cdot \{a, b\} \\ &= \{aaa, aab, bba, bbb, aba, abb, baa, bab\} \end{aligned}$$

RegEx

Wissenswertes:

- Hilfssymbole := $\{ |, (,), *, \emptyset \}$
- " * vor \cdot (Konkatenation) "
- " \cdot vor Strich "—" (Oder)
- $\langle R \rangle$ ist die formale Sprache ist, welche mit R gebildet werden kann
- $\langle \emptyset \rangle = \{ \}$
- $\langle R_1 | R_2 \rangle = \langle R_1 \rangle \cup \langle R_2 \rangle$
- $\langle R_1 \cdot R_2 \rangle = \langle R_1 \rangle \cdot \langle R_2 \rangle$

- $\langle R^* \rangle = \langle R \rangle^*$
- Es gibt **kein** R^+ sondern RR^* Bsp.: Statt $(ab)^+$ einfach $ab(ab)^*$

Bsp.:

$R = a|b$ dann ist:

$$\langle R \rangle = \langle a|b \rangle = \langle a \rangle \cup \langle b \rangle = \{a\} \cup \{b\} = \{a, b\}$$

$R = (a|b)^*$ dann ist:

$$\langle R \rangle = \langle (a|b)^* \rangle = \langle a|b \rangle^* = \{a, b\}^*$$

$R = (a * b)^*$ dann ist:

$$\begin{aligned} \langle R \rangle &= \langle (a * b)^* \rangle = \langle a * b \rangle^* \\ &= (\langle a \rangle \langle b \rangle)^* = (\langle a \rangle^* \langle b \rangle^*)^* = (\{a\}^* \{b\}^*)^* \\ &= \{a, b\}^* \end{aligned}$$

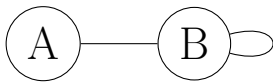
Graphen

- Ein gerichteter Graph ist das Paar $G = (V, E)$
 - **Knotenmenge** V ist endlich und nichtleer (V für engl. vertex)
 - **Kantenmenge** $E \subseteq V \times V$ (E für engl. edge)
 - * muss damit auch endlich sein, darf aber leer sein
- **Pfade** können über mehrer Kanten führen
- $V^{(+)}$: Menge der nichtleeren Listen von Elementen aus V
- Ein Pfad ist $p = (v_0, \dots, v_n) \in V^{(+)}$ wenn für jedes $i \in \mathbb{Z}_n$ gilt: $(v_i, v_{i+1}) \in E$
- Die Länge eines Pfades ist die Anzahl der Kanten
- v_n von v_0 ist erreichbar, wenn ein Pfad $p = (v_0, \dots, v_n)$ existiert
- Wenn der start und endpunkt identisch sind heißt der Pfad **geschlossen**

- Wenn der geschlossene Pfad größer gleich 1 ist, heißt er **Zyklus**
- Pfad heißt **wiederholungsfrei**, wenn
 - der erste bis zum vorletzten Knoten verschieden sind (v_0, \dots, v_{n-1})
 - der zweite bis zum letzten Knoten verschieden sind (v_1, \dots, v_n)
 - der erste und letzte Knoten dürfen gleich sein (v_0 und v_n)
 - Einfach: Außer der letzte und erste darf jeder Knoten nur einmal "betreten" werden
- **azyklischer Graph**: kein Teilgraph ist zyklisch
- Ein Graph ist **streng zusammenhängend** wenn
 - zwischen jeden beliebigen zwei Knoten (Knotenpaar) aus dem Graphen ein Pfad existiert. Also jeder Punkt von jedem anderen Punkt (sich eingeschlossen) erreichbar ist.
- Ein Graph ist ein **gerichteter Baum** wenn:
 - es eine **Wurzel** $r \in V$ gibt, für die gilt:
 - * zu jedem Knoten existiert **genau** ein Pfad
 - * Wurzel ist immer **eindeutig**
- Der **Eingangsgrad** eines Knoten ist die Anzahl aller Kanten die zu dem Knoten hinführen
- Der **Ausgangsgrad** eines Knoten ist die Anzahl aller Kanten die von den Knoten wegführen
- Der **Grad** eines Knoten ist die Anzahl der Kanten des Knotens (Also Ausgangsgrad + Eingangsgrad)
- Knoten eines Baumes werden **Blätter** genannt, wenn Sie das Ende des Baumes sind, also Ausgangsgrad = 0
- **innere Knoten** sind dann alle mit Ausgangsgrad > 0
- E^n ist ein Pfad der Länge n . Bsp.: E^2 ist ein Pfad der Länge 2

- $(x, y) \in E^2 \Leftrightarrow$ es existiert ein Pfad der Länge 2 von x nach y
- Ein ungerichteter Graph hat einfach nur Kanten und keine "Richtungs" Pfeile
- Knotengrad für ungerichtete Graphen: man zählt alle "Kantenenden"

Beispiel:



$$d(B) = 3$$

Kontextfreie Grammatik

- N sind alle Nichtterminalsymbole
- T sind alle Terminalsymbole
- $N \cap T = \emptyset$
- S ist der Start und $S \in N$
- P Produktionen, endliche Menge und $P \in N \times V^*$
 - $V = N \cup T$ die Menge aller Symbole
 - Für jeder $(X, w) \in P$ schreibt man $X \rightarrow w$
 - Man "ersetzt" P durch w

Bei einem **Ableitungsschritt** wird ein Terminalsymbol durch abgeleitet. Dieser wird dann mit " \Rightarrow " dargestellt, nicht mit Implikations verwechseln!

Bsp.:

$$G = (\{X\}, \{a, b\}, X, P) \text{ mit } P = \{X \rightarrow \epsilon, X \rightarrow aXb\}$$

Dann gilt zB.: $abaXbaXXXX \Rightarrow abaXbaaXbXXXX$ als Ableitungsschritt. (Man ersetzt das X nach ba mit aXb)

Man kann auch einfach mit einem Index angeben, wie viele Ableitungsschritt getätigt werden.

\Rightarrow^2 aber auch $u \Rightarrow^* v$ wenn v aus u ableitbar ist. \Rightarrow^0 ist einfach wieder das selbe.

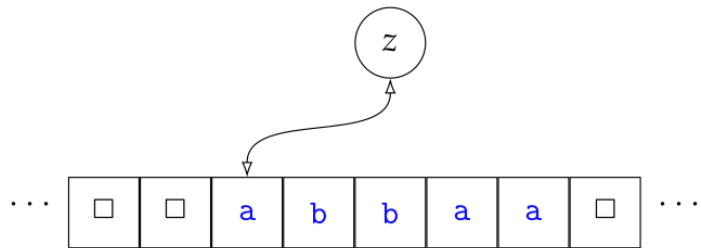
- Eine **Grammatik** erzeugt eine **formale Sprache**, also einfach alle Wörter die man aus einer Grammtik ableiten kann.
- $G = (N, T, S, P)$ erzeugt die formale Sprache $L(G) = \{w \in T^* | S \Rightarrow^* w\}$
- Diese formalen Sprachen heißen **kontextfrei**

Turing-Maschinen

Eine Turingmaschine ist im Grunde einfach nur ein spezieller Automat. Die Besonderheit der Turingmaschinen ist das "Speicherband", der "Schreib-Lese-Kopf" und die Anweisungen für den Schreib-Lese-Kopf in der Ausgabe.

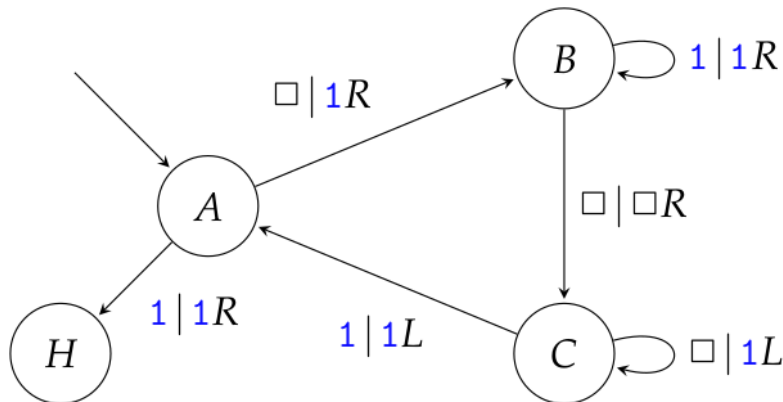
- Das Speicherband ist in einzelne Zellen eingeteilt (Wie ein Array) das nach links und rechts "unendlich" ist.
- Eine leere Zelle wird mit dem Symbol \square markiert.
- Der Schreib-Lese-Kopf zeigt immer auf eine Zelle im Speicherband und auf einen Zustand in der Turingmaschine.
- Wie auch bei Automaten gibt es akzeptierende und nicht akzeptierende Zustände.
- Die Eingabe ist der Inhalt der Zelle, auf welche der Schreib-Lese-Kopf zeigt, und die Ausgabe wird eben in diese nun geschrieben. Dabei gibt es in der Ausgabe zwei Anweisungen:
 - L Der Schreib-Lese-Kopf wird um eins nach links verschoben.
 - R Der Schreib-Lese-Kopf wird um eins nach rechts verschoben.
 - 0 Der Schreib-Lese-Kopf bleibt einfach in der Zelle.

- Eine Turingmaschine muss nicht jedes Szenario abhandeln können. Im Falle eines unbehandelten Falles bleibt die Turingmaschine einfach stehen.
- Wenn man zB. nun ein Wort in die Turingmaschine geben möchte, muss man dieses Wort einfach auf das Speicherband schreiben.



Hier sieht man eine simple Turingmaschine mit einem einzigen Zustand Z . Die Zellen unter dem Zustand sind das Speicherband mit der Eingabe. Der Pfeil symbolisiert den Schreib-Lese-Kopf. Aktuell befindet sich die Turingmaschine im Zustand Z und liest/schreibt in die erste Zelle des Speicherbandes.

Jede Turingmaschine kann zu einer Tabelle umgeschrieben, das Beispiel sollte fürs Verständnis ausreichen:



| | A | B | C | H |
|---|---------|---------|---------|---|
| □ | 1, R, B | □, R, C | 1, L, C | |
| 1 | 1, R, H | 1, R, B | 1, L, A | |

Die rechte Spalte repräsentiert die mögliche Eingabe (1 oder \square). Die obere Zeile sind die verschiedenen Zustände (A, B, C, H). In die Tabelle wird dann eingetragen was welcher Zustand für welche Eingabe macht. zB. für die Eingabe \square in Zustand A , schreibt der Schreib-Lese-Kopf eine 1 in die Zelle des Speicherbandes, das R bewegt den Schreib-Lese-Kopf eins nach rechts und der Zustand wechselt zu Zustand B .