BFS-Skript

31.07.2023 - BFS - Prof. Wanka

1. Turing-Maschinen allgemein

1.1 Entscheidbar & Rekursiv aufzählbar

Definition 1.3
M sei deterministische 1-Band-TM. $x \in \Sigma^*$ sei die Eingabe von M.

- *M* **akzeptiert** *$x \in \Sigma^*$, falls es $\alpha, \beta \in \Gamma^*$ und $q \in F$ gibt mit $q_0 x \vdash^* \alpha q \beta$.*
  (im Beispiel 1.2: $q_0 000111 \vdash^* BBBBq_7BBBB$)
  O. B. d. A. wenn eine Konfiguration mit $q \in F$ erreicht wird, gibt es keine Nachfolgekonfiguration mehr.
- *Die Sprache* $\mathbf{L(M)}$ *von M ist die Menge aller von M akzeptierten $x \in \Sigma^*$. Hier wird keine Aussage getroffen über das Verhalten von M, wenn M die Eingabe x nicht akzeptiert. M* **akzeptiert** *die Sprache $L(M)$.*
- *Ein stärkerer Begriff: Falls M die Sprache L akzeptiert und für alle Eingaben $x \in \Sigma^*$ nach endlich vielen Schritten anhält, so* **entscheidet** *M die Sprache L.*
- *$L \subseteq \Sigma^*$ heißt* **rekursiv aufzählbar** *(r.a., engl. r.e. (recursively enumerable)) genau dann, wenn es eine deterministische 1-Band-TM M gibt mit $L(M) = L$.*
- *$L \subseteq \Sigma^*$ heißt* **entscheidbar** *oder* **rekursiv** *genau dann, wenn es eine deterministische 1-Band-TM M gibt, die L entscheidet.*

Eine Sprache L ist **entscheidbar** *genau dann wenn die sogenannte charakteristische Funktion*

$$\chi_L : \{0,1\}^* \to \{0,1\} \text{ mit } \chi_L(x) = \begin{cases} 1 & \text{falls } x \in L \\ 0 & \text{falls } x \notin L \end{cases}$$

total berechenbar ist.
Eine Sprache L ist **rekursiv aufzählbar** *genau dann wenn die partielle Funktion*

$$\chi'_L : \{0,1\}^* \to \{0,1\} \text{ mit } \chi_L(x) = \begin{cases} 1 & \text{falls } x \in L \\ \textit{undefiniert} & \text{falls } x \notin L \end{cases}$$

berechenbar ist.

L ist rekursiv aufzählbar $\iff$ es gibt eine total berechenbare *surjektive Funktion $g : \{0,1\}^* \to L$.*

Entscheidbarkeit: Hält immer, aber nur akzeptierend, wenn Wort in Sprache enthalten ist.

Rekursive Aufzählbarkeit: Hält nur akzeptierend, also nur wenn Wort in Sprache enthalten ist.

Satz 1.26 *Seien L_1 und L_2 rekursiv aufzählbare Sprachen. Dann gilt:*

(1) $L_1 \cup L_2$ ist rekursiv aufzählbar.

(2) $L_1 \cap L_2$ ist rekursiv aufzählbar.

Satz 1.27 *L ist entscheidbar $\iff$ L und $\overline{L}$ sind rekursiv aufzählbar.*

2. HALTEPROBLEM

Definition 1.13 (Halteproblem) *Die Sprache*

$$H = \{\langle M\rangle w \mid M \text{ ist deterministische 1-Band-TM, die, gestartet mit Eingabe } w, \text{ hält}\}$$

heißt **(allgemeines) Halteproblem**.

H ist rekursiv aufzählbar. Eine universelle TM „zählt H auf".

Satz 1.14 (Turing, 1936) *H ist unentscheidbar. (oder alternativ: H ist nicht rekursiv)*

Satz 1.15

(a) H ist rekursiv aufzählbar.

(b) $\overline{H} = \{0,1\}^* \setminus H$ ist nicht rekursiv aufzählbar.

2.2 INITIALES HALTEPROBLEM

Definition 1.16 (Initiales Halteproblem) *Die Sprache*

$$H_\varepsilon = \{\langle M\rangle \mid M \text{ ist deterministische 1-Band-TM, die, gestartet mit Eingabe } \varepsilon \text{ (dem leeren Wort), hält}\}$$

heißt **initiales Halteproblem**.

Satz 1.17 *H_ε ist unentscheidbar.*

3. NICHT-DETERMINISTISCHE TURING-MASCHINEN

Einziger Unterschied zu deterministischen Turing-Maschinen: $\delta : Q \times \Gamma \to P(Q \times \Gamma \times \{L, R, N\})$

Es gibt also für eine Eingabe der δ-Fkt. mehrere Ausgaben, die nicht-deterministische TM probiert dann alle aus („würfelt").

Korollar 2.4 *NTMs akzeptieren genau die rekursiv aufzählbaren Sprachen.*

3.2 P=NP PROBLEM

Definition 2.5

$$\begin{aligned}
\texttt{DTIME}(t(n)) &:= \{L \mid \textit{Es gibt eine deterministische } \mathcal{O}(t(n))\textit{zeitbeschränkte (Mehrband-)TM, die } L \textbf{\textit{ entscheidet}}\}\\
\texttt{NTIME}(t(n)) &:= \{L \mid \textit{Es gibt eine } \textbf{\textit{nicht}}\textit{deterministische } \mathcal{O}(t(n))\textit{zeitbeschränkte (Mehrband-)TM, die } L \textbf{\textit{ akzeptiert}}\}\\
\mathbf{P} &:= \bigcup_{k\in\mathbb{N}} \mathit{DTIME}\left(n^k\right)\\
\mathbf{NP} &:= \bigcup_{k\in\mathbb{N}} \mathit{NTIME}\left(n^k\right)
\end{aligned}$$

Offensichtlich gilt $P \subseteq NP$.

P: Der Algorithmus hat in **p**olynomielle Laufzeit

NP: Der Algorithmus hat exponentielle (**n**icht-**p**olynomielle) Laufzeit

3.3 VERIFIZIERER

Definition 2.6 *Sei L eine Sprache über $\{0,1\}$.*
Eine deterministische *Turingmaschine V_L heißt $t(n)$-beschränkter* **Verifizierer** *für L, wenn gilt:*

(i) *Die Eingaben von V_L sind von der Form $x\#w$, $w, x \in \{0,1\}^*$*

(ii) *Die Laufzeit ist in $\mathcal{O}(t(|x|))$.*

(iii) *Für alle $x \in \{0,1\}^*$:*

$$x \in L \Leftrightarrow \exists w : |w| \leq t(|x|) \text{ und } V_L \text{ akzeptiert } x\#w.$$

Dieses w heißt **Zertifikat** *von x.*

Satz 2.7 *Sei L eine Sprache. Dann gilt:*
$L \in \texttt{NTIME}(t(n)) \Leftrightarrow$ es gibt einen $t(n)$-beschränkten Verifizierer V_L für L.

Korollar 2.8

$$NP = \{L \mid \text{es gibt einen polynomiellen Verifizierer für } L\}$$

3.4 NP-VOLLSTÄNDIGKEIT

Definition 2.9 *Seien $L_1 \subseteq \Sigma_1^*$, $L_2 \subseteq \Sigma_2^*$.*
L_1 ist genau dann ***polynomiell reduzierbar*** *auf L_2 ($L_1 \leq_p L_2$), wenn gilt:*

(i) *$L_1 \leq L_2$ mittels Reduktionsfunktion f.*

(ii) *Die Laufzeit zur Berechnung von $f(x)$ ist $\mathcal{O}\left(|x|^k\right)$ für $k \in \mathbb{N}$.*

Lemma 2.10 *„$\leq_p$" ist transitiv.*

BEWEIS: Siehe Übung □

Definition 2.11

- *L heißt* **NP-schwer** *(engl.* **NP-hard**, *auch* **NP-schwierig** *- Vorsicht vor dem false friend „NP-hart", der in der Literatur auch auftritt), wenn gilt:*

$$\forall L' \in NP : L' \leq_p L$$

- *L heißt* **NP-vollständig** *(engl.* **NP-complete**, **NPC**), *wenn gilt:*

  (i) $L \in NP$

  (ii) *L ist NP-schwer.*

Lemma 2.12

(i) *L sei NP-schwer. Dann gilt:*

  (a) $L \in P \Rightarrow P = NP$

  (b) $P \neq NP \Rightarrow L \notin P$

(ii) *L sei NP-vollständig. Dann gilt:*

  (a) $L \in P \Leftrightarrow P = NP$

  (b) $L \notin P \Leftrightarrow P \neq NP$

  (c) *$L' \in NP$ und $L \leq_p L' \Rightarrow L'$ ist NP-vollständig.*

4. Verschiedene (NP-vollständige) Probleme

CLIQUE

$CLQUE := \{\langle G, k\rangle \mid k \in \mathbb{N}, G = (V, E) \text{ ist ungerichteter Graph}, \exists U \subseteq V, |U| = k : \forall u, v \in U : \{u, v\} \in E\}$

Independent Set

$IS := \{\langle G, k\rangle \mid k \in \mathbb{N}, G = (V, E) \text{ ist ungerichteter Graph}, \exists U \subseteq V, |U| = k : \forall u, v \in U : \{u, v\} \notin E\}$

Vertex Cover

$VC := \{\langle G, k\rangle \mid k \in \mathbb{N}, G = (V, E) \text{ ist ungerichteter Graph}, \exists U \subseteq V, |U| = k : \forall \{u, v\} \in E : u \in U \vee v \in U\}$

SAT (Erfüllbarkeitsproblem)

Eine KNF Φ heißt **erfüllbar**, *wenn es eine Belegung c gibt, sodass $c(\Phi) =$ `TRUE` ist.*

$SAT := \{\langle \Phi \rangle \mid \Phi \text{ ist eine erfüllbare KNF}\}.$

kSAT

$kSAT := \{\langle \Phi \rangle \mid \Phi \text{ sei eine erfüllbare KNF,}$
$\text{in der jede Klausel aus genau } k \text{ Literalen über } k \text{ verschiedenen Variablen besteht}\}.$

- $kSat \subseteq SAT$
- $2SAT \in P$

Färbungs-Problem

Sei $k \in \mathbb{N}$. Ein ungerichteter Graph $G = (V, E)$ heißt **k-*färbbar***, wenn es eine Abbildung $c : V \to \{1, \ldots, k\}$ gibt mit der Eigenschaft, dass $\forall \{u, v\} \in E$ gilt, dass $c(u) \neq c(v)$.

$$\textsc{Col} := \{\langle G, k\rangle \mid G \text{ ist ein ungerichteter Graph und } G \text{ ist } k\text{-färbbar}\} \ .$$

$$3\textsc{Col} := \{\langle G\rangle \mid G \text{ ist ein ungerichteter Graph und } G \text{ ist 3-färbbar}\} \ .$$

Hamiltonkreis

Ein ***Hamiltonkreis*** ist ein Kreis in G, der jeden Knoten genau einmal enthält.

$$\text{HC} := \{\langle G\rangle \mid G \text{ ist eine ungerichteter Graph und enthält einen Hamilton-Kreis}\} \ .$$

Traveling Salesman

$$\text{TSP} := \{\langle G, c, k\rangle \mid \text{der Graph } G \text{ mit Kantengewichten } c : E \to \mathbb{R} \text{ (meist } c : E \to \mathbb{N})$$
$$\text{enthält eine Rundreise (Hamiltonkreis) mit Gewicht } \leq k \qquad \} \ .$$

Binary-Programming

$$\text{BP} := \{\langle A, \vec{b}\rangle \mid A \text{ ist eine } m \times n\text{-Matrix mit ganzzahligen Einträgen, } \vec{b} \text{ ist ein Vektor mit } m$$
$$\text{ganzzahligen Einträgen, und es gibt einen 0-1-Vektor } \vec{x} \in \{0, 1\}^n \text{ mit } A \cdot \vec{x} \leq \vec{b}\} \ .$$

5. Formale Sprachen

5.1 Grammatiken

Chomsky-0 Grammatik

Definition 3.1 *Eine* ***Grammatik*** G *vom Typ* ***Chomsky-0*** *ist beschrieben durch ein 4-Tupel* $(V, \Sigma, \mathcal{P}, \mathcal{S})$ *mit:*

- V*: Endliche Menge von* ***Variablen***
- Σ*: Endliche Menge der* ***Terminalsymbole*** *bzw.* ***Terminale***
- $\mathcal{S}$*:* $\mathcal{S} \in V$*:* ***Startsymbol***
- $\mathcal{P} \subseteq ((V \cup \Sigma)^+ \setminus \Sigma^*) \times (V \cup \Sigma)^*$*: Endliche Menge von* ***Produktionen****, oder auch* ***Ersetzungsregeln****, oder* ***Ableitungsregeln****, (oder ganz kurz: Regeln)*
  Statt $(u, v) \in \mathcal{P}$ *schreiben wir auch:* $u \to v$.

Direkt ableitbar: ω ist in einem Schritt erzeugbar Indirekt ableitbar: ω ist in mehreren Schritten erzeugbar

Chomsky-1 (kontextsensitiv)

Für jede Produktion $u \to v \in P$ gilt: $|u| \leq |v|$

Chomsky-2 (kontextfrei)

Für jede Produktion $u \to v \in P$ gilt: $u \in V$ (und damit $|u| = 1$)

Chomsky-3 (regulär)

$$u \in V \text{ und } v \in \{\varepsilon\} \cup \Sigma \text{ oder}$$

$u \in V$ *und* $v \in \Sigma \circ V$ *(das heißt* $v = aw$ *mit* $a \in \Sigma$ *und* $w \in V$*).*

5.2 Erzeuger und Erkenner

Sprachtyp	Menge	Erzeuger	Erkenner
rekursiv aufzählbar	$\mathcal{L}_0$	Chomsky-0	Turingmaschinen
kontextsensitiv	$\mathcal{L}_1$	Chomsky-1	Linear beschränkte Turingmaschinen
kontextfrei	$\mathcal{L}_2$	Chomsky-2	Kellerautomaten
regulär	$\mathcal{L}_3$	Chomsky-3	Automaten

Satz 3.3 *Eine Sprache* L *ist genau dann rekursiv aufzählbar, wenn es eine* Chomsky-*0-Grammatik* G *mit* $L = L(G)$ *gibt* ($\mathcal{L}_0$ *ist die Menge der rekursiv aufzählbaren Sprachen).*

Satz 3.34

$$\mathcal{L}_3 \subsetneq \mathcal{L}_2 \subsetneq \mathcal{L}_1 \subsetneq \mathcal{E} \subsetneq \mathcal{L}_0 \subsetneq \text{„Alle Sprachen“}$$

5.3 Deterministische Endliche Automaten (DFA)

Definition 3.4 *Ein* **deterministischer endlicher Automat** ***(DFA, deterministic finite automaton)*** *A ist beschrieben durch 5 Komponenten $A = (Q, \Sigma, \delta, q_0, F)$ mit:*

- *Q: Endliche Menge der Zustände*
- *Σ : Endliches Alphabet, $Q \cap \Sigma = \emptyset$*
- *$\delta : Q \times \Sigma \to Q$ Übergangsfunktion*
- *q_0 : Startzustand*
- *$F : F \subseteq Q$, akzeptierende Endzustände*

Man kann sich einen DFA als eine *Turingmaschine, die nicht schreibt und den Kopf nur nach rechts bewegt,* vorstellen.

Die von A **akzeptierte Sprache** *ist*

$$\mathbf{L(A)} = \{w \mid w \in \Sigma^*,\, \delta(q_0, w) \in F\}\,.$$

Satz 3.6 *L werde von einem deterministischen endlichen Automaten A akzeptiert. Dann gibt es eine* Chomsky*-3-Grammatik (bzw.: reguläre Grammatik) G, die L erzeugt, also $L = L(A) = L(G)$.*

5.4 Nicht-deterministische Endliche Automaten (NFA)

Definition 3.7 *Ein* **nichtdeterministischer endlicher Automat** ***(NFA, non-deterministic finite automaton)*** *N ist beschrieben durch die 5 Komponenten $N = (Q, \Sigma, \delta, q_0, F)$ mit:*

- *Q: Endliche Menge der Zustände*
- *Σ : Endliches Alphabet, $Q \cap \Sigma = \emptyset$*
- *$\delta : Q \times \Sigma_\varepsilon \to \mathcal{P}(Q)$, wobei $\mathcal{P}$ die Potenzmenge ist und $\Sigma_\varepsilon = \Sigma \cup \{\varepsilon\}$.*
- *q_0 : Startzustand*
- *$F : F \subseteq Q$, akzeptierende Endzustände*

N **akzeptiert** *$w \in \Sigma^*$, falls*

- *$w = w_1 \dots w_n \in \Sigma_\varepsilon^*$*
- *$\delta(q_0, w_1 \dots w_n) \cap F \neq \emptyset$*

Satz 3.9 *Sei N ein nichtdeterministischer, endlicher Automat. Dann gibt es einen deterministischen, endlichen Automaten A mit $L(N) = L(A)$.*

Korollar 3.10 *Sei L eine reguläre Sprache. Dann gibt es einen DFA A mit $L(A) = L$.*

Korollar 3.11 *L ist genau dann regulär, wenn es einen DFA A gibt mit $L = L(A)$.*

Umwandlung in DEA

Beispiel:

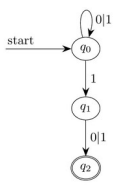
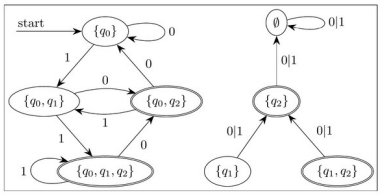


Abbildung 15: NFA N_1



$F' = \{R \in Q' \mid E(R) \cap F \neq \emptyset\} = \{\{q_2\}, \{q_0, q_2\}, \{q_1, q_2\}, \{q_0, q_1, q_2\}\}$

Abbildung 16: DFA A_1 nach Potenzmengenkonstruktion von N_1

1. Schritt: Es gibt keine ε-Übergänge.
Potenzmengenkonstruktion
(Erzeugung des deterministischen, endlichen Automaten

- $Q' = \mathcal{P}(Q)$
- $q_0' = \{q_0\}$
- $F' = \{R \in Q' \mid R \cap F \neq \emptyset\}$
- $\delta'(R, a) = \bigcup_{r \in R} \delta(r, a) = \{q \in Q \mid q \in \delta(r, a), r \in R\}\ (\in Q')$

$L(N) = L(A)$, da
$w \in L(N) \Leftrightarrow \delta(q_0, w) \cap F \neq \emptyset \Leftrightarrow \delta'(q_0', w) \in F' \Leftrightarrow w \in L(A)$

2. Schritt: Auch ε-Übergänge vorhanden.

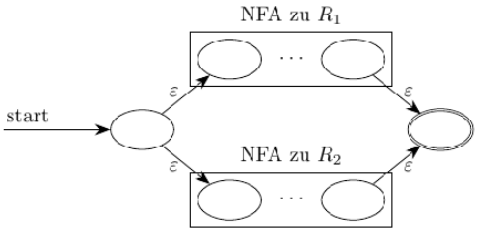
$$E(R) = \{q \in Q \mid \exists r_1, \ldots, r_k \in Q,\ r_1 \in R, r_k = q : \delta(r_i, \varepsilon) \ni r_{i+1},\ i = 1, \ldots, k-1\}$$

Da auch 0 ε-Übergänge vorkommen können, gilt: $R \subseteq E(R)$.

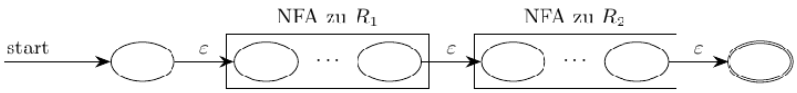
- $q_0' = E(\{q_0\})$
- $F' = \{R \in Q' \mid E(R) \cap F \neq \emptyset\}$
- $\delta'(R, a) = \bigcup_{r \in R} E(\delta(r, a)) = \{q \in Q \mid q \in E(\delta(r, a)), r \in R\}$

5.5 Mehrere Automaten verbinden

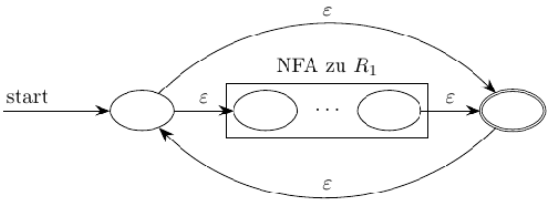
$(R_1 \cup R_2)$ wobei R_1, R_2 reguläre Ausdrücke sind



$(R_1 \cdot R_2)$ wobei R_1, R_2 reguläre Ausdrücke sind



(R_1^*) wobei R_1 regulärer Ausdruck ist



5.6 PUMPING-LEMMA

Definition 3.19 *Sei L eine Sprache über Σ.*
L hat die **reguläre Pump-Eigenschaft**, *falls gilt:*

$$\exists n_L \in \mathbb{N}\, \forall z \in L,\, |z| \geq n_L\, \exists u, v, w \in \Sigma^* : uvw = z \text{ und}$$

(i) $|uv| \leq n_L$

(ii) $v \neq \varepsilon$

(iii) $\forall i \geq 0 : uv^i w \in L$

Satz 3.20 (Pumping-Lemma für reguläre Sprachen)
Wenn eine Sprache regulär ist, dann hat sie die reguläre Pump-Eigenschaft.

Beispiele für Sprachen, die die reguläre Pump-Eigenschaft haben

- $\{0^i 1^j \mid i, j \geq 0\}$

$$L = \{a^i b^j \mid i, j \geq 0\}$$

Setze $n_L = 5$. Sei $z \in L$ beliebig, aber fest mit $|z| \geq 5$. Setze $u = \varepsilon$. v ist dann das erste Zeichen von z, w dessen Rest.

(i) $uv = v =$ erstes Zeichen von z, dessen Betrag ist $1 \leq 5$ ✓.

(ii) $v \neq \varepsilon$ ✓

(iii) $uv^i w =$ (erstes Zeichen von $z)^i w \in L$ ✓

$\Rightarrow L$ hat die reguläre Pump-Eigenschaft.

„Wording“:
$\forall x$: Sei x beliebig aber fest
$\exists y$: Setze y = ...

Negiertes Pumping-Lemma

Die **Negation der regulären Pump-Eigenschaft** kann folgendermaßen formuliert werden:
Sei L eine Sprache.

$$\forall n_L \in \mathbb{N}\, \exists x \in L,\, |x| \geq n_L\, \forall u, v, w \in \Sigma^* : uvw = x :$$
$$(\underbrace{|uv| \leq n_L}_{(i)} \wedge \underbrace{v \neq \varepsilon}_{(ii)}) \Rightarrow \underbrace{\exists i \geq 0 : uv^i w \notin L}_{\neg(iii)}$$

Beispiele für Sprachen die nicht die reguläre Pump-Eigenschaft haben

- $\{0^n 1^n \mid n \geq 1\}$
- $\{aa \mid a \in \{0,1\}^*\}$
- $\{1^p \mid p \text{ ist Primzahl}\}$

$L = \{0^n 1^n \mid n \geq 1\}$ hat nicht die reguläre Pump-Eigenschaft.
BEWEIS:
Sei $n_L \in \mathbb{N}$ beliebig, aber fest.
Setze $z = 0^{n_L} 1^{n_L} \in L$ und $|z| \geq n_L$.
Sei u, v, w mit $z = uvw$ beliebig, aber fest, wobei $|uv| \leq n_L$ und $|v| \geq 1$ gilt.
Aus $|uv| \leq n_L$ folgt: v besteht nur aus Nullen, und da $|v| \geq 1$ besteht v aus mindestens einer Null.
Setze $i := 0 \Rightarrow uw = 0^{n_L - |v|} 1^{n_L} \notin L$, da $n_l - |v| < n_L$ ist.

$L = \{aa \mid a \in \{0,1\}^*\}$ hat nicht die reguläre Pump-Eigenschaft.
BEWEIS:
Sei $n_L \in \mathbb{N}$ beliebig, aber fest.
Setze $z = 0^{n_L}1^{n_L}0^{n_L}1^{n_L} \in L$ und $|z| \geq n_L$.
Sei u, v, w mit $z = uvw$ beliebig, aber fest, wobei $|uv| \leq n_L$ und $|v| \geq 1$ gilt.
Aus $|uv| \leq n_L$ folgt: v besteht nur aus Nullen, die vor der ersten Eins stehen. Da $|v| \geq 1$ gilt, besteht v aus mindestens einer dieser Nullen.
Setze $i := 0 \Rightarrow uw = 0^{n_L-|v|}1^{n_L}0^{n_L}1^{n_L} \notin L$, da:

- Fall $|uw|$ ungerade: Somit kann es keine Aufteilung von uw in zwei gleichlange Teilworte geben.
- Fall $|uw|$ gerade: Somit gilt $|uw| = 4n_L - |v|$. Teilt man nun uw in zwei gleichlange Teilworte $uw = ab$ auf, so gilt, dass $2n_L - |v| \overset{|v|\geq 1}{<} 2n_L - \frac{1}{2}|v| = |a| \overset{1\leq|v|\leq n_L}{<} 3n_L - |v|$. Somit gilt, dass a mit einer Null endet, b jedoch mit einer Eins $\Rightarrow a \neq b$

$L = \{1^p \mid p \text{ ist Primzahl}\}$ hat nicht die reguläre Pump-Eigenschaft.
BEWEIS:
Sei $n_L \in \mathbb{N}$ beliebig, aber fest.
Sei q eine Primzahl mit $q \geq n_L + 2$.
Setze $z = 1^q \in L$ und $|z| = q \geq n_L + 2 \geq n_L$.
Sei u, v, w mit $z = uvw$ beliebig, aber fest, wobei $|uv| \leq n_L$ und $|v| \geq 1$ gilt.
Daraus folgt $|w| = |uvw| - |uv| \geq n_L + 2 - n_L \geq 2$.
Setze $i := |uw|$.
Es gilt $|uw| \geq |w| \geq 2$.
$\Rightarrow uv^iw = 1^{|uw|+|uw|\cdot|v|} = 1^{|uw|\cdot(1+|v|)} \notin L$,
denn wegen $|uw| \geq 2$ und $|v| \geq 1$ ist $|uw| \cdot (1 + |v|)$ nicht prim.

5.7 Reguläre Ausdrücke

Regulärer Ausdruck: Alle Zeichen sind $\in \varepsilon, \emptyset, \sum$

RIJK-Algorithmus

$$k = 0 \text{ und } i \neq j\text{: } R^0_{i,j} = \bigcup_{\substack{a\in\Sigma:\\ \delta(q_i,a)=q_j}} a$$

$$k = 0 \text{ und } i = j\text{: } R^0_{i,i} = \left(\bigcup_{\substack{a\in\Sigma:\\ \delta(q_i,a)=q_i}} a \right) \cup \varepsilon$$

$$k > 0 \qquad : R^k_{i,j} = R^{k-1}_{i,j} \cup (R^{k-1}_{i,k} \cdot (R^{k-1}_{k,k})^* \cdot R^{k-1}_{k,j})$$

Gesamtausdruck: $R = \bigcup_{q_j \in F} R^n_{1,j}$

Gauss-Elimination

1. Alle ausgehenden Kanten in der Schreibweise: Zustand = *Input1*Zustand1 $\cup$ *Input2*Zustand2
   Endzustand hat zusätzlich: $\cup\, \varepsilon$
2. Nacheinander alle Zustände außer den Endzustand eliminieren
   Vereinfachungen wie folgt: $\mathrm{A} = \mathrm{xA} \cup yB \quad \Rightarrow \quad x^*yB$
3. Endzustand eliminieren
   ε ist nur beim Ausklammern relevant und kann danach einfach weggelassen werden
4. Die Formel des Anfangszustands ist $\mathrm{L}_g = L(G) = L(\mathrm{Formel})$

5.8 KONTEXTFREIE GRAMMATIKEN

Chomsky-Normalform

Definition 3.24 *Eine kontextfreie Grammatik $G = (V, \Sigma, \mathcal{P}, \mathcal{S})$ ist in* **Chomsky-Normalform (ChNF)**, *wenn jede Regel in G von folgender Form ist:*

- $A \to BC$ *mit* $A \in V$, $B, C \in V \setminus \{\mathcal{S}\}$ *oder*
- $A \to a$ *mit* $A \in V$, $a \in \Sigma$ *oder*
- $\mathcal{S} \to \varepsilon$.

Nutzlose Variablen

Eine Variable A heißt ***nutzlos***, wenn es kein Wort $w \in \Sigma^*$ gibt mit $A \overset{*}{\to} w$. Sonst heißt diese Variable ***nützlich***.

Kontextfreie Grammatiken und Sprachen

Eine Sprache heißt ***kontextfrei***, wenn es eine kontextfreie Grammatik G mit $L(G) = L$ gibt

Kontextfreie Grammatiken haben Chomsky-Normalform und alle Variablen sind nützlich („Ein Automat ohne unerreichbare Zustände“).

Kontextfreie Grammatiken ohne Kettenregel

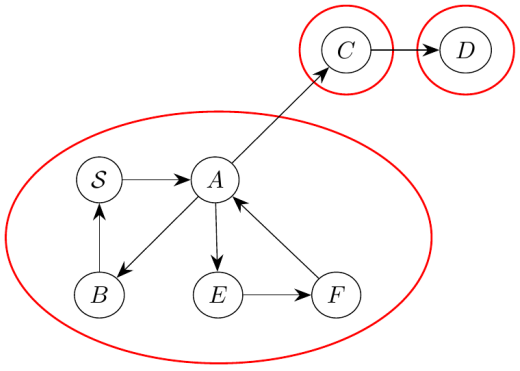
Lemma 3.27 *Zu jeder kontextfreien Grammatik $G = (V, \Sigma, \mathcal{P}, \mathcal{S})$ gibt es eine kontextfreie Grammatik $G' = (V', \Sigma, \mathcal{P}', \mathcal{S})$ ohne Kettenregeln (das heißt Regeln der Form $A \to B$, wobei A, B jeweils* eine *Variable ist) mit $L(G) = L(G')$.*

- Konstruiere einen gerichteten Graphen $G_{Kette} = (V, E_{Kette})$, in dem die Variablen die Knoten sind und die Kanten die Kettenregeln.
- Ersetze in den starken Zusammenhangskomponenten (Äquivalenzklassen) die Variablen darin durch einen Repräsentanten davon. Falls $\mathcal{S}$ in einer der starken Zusammenhangskomponente enthalten ist, so muss $\mathcal{S}$ als Repräsentant benutzt werden.
- Ersetze in allen Produktionen die ersetzten Variablen durch den entsprechenden Repräsentanten. Führe die Ersetzung ebenfalls in G_{Kette} durch.
- Ersetze nun im dem übrig gebliebenen kreisfreien Graphen (directed acyclic graph - DAG) in umgekehrter topologischer Reihenfolge („von unten nach oben“) die Kettenregeln $A \to B$ durch die Regeln $A \to$ „alle rechten Seiten von B“ und entferne jeweils im Anschluss die verarbeitete Kettenregel. □

kontextfreie Grammatik $G = (V, \Sigma, \mathcal{P}, \mathcal{S})$
$V = \{\mathcal{S}, A, B, C, D, E, F\}$
Sei $\mathcal{P}'$ die Menge der Kettenregeln aus $\mathcal{P}$

$$\mathcal{P}' = \{\mathcal{S} \to A,\ A \to B,\ B \to \mathcal{S},\ A \to E,\ E \to F,\ F \to A,\ A \to C,\ C \to D\}$$

Äquivalenzklassen (rot markiert) definiert durch gegenseitige Erreichbarkeit



CYK-Algorithmus (↑↙)

$V(i,i) = \{A \mid (A \to a_i) \in \mathcal{P}\} \ \forall i \in \{1, \ldots, n\}$
$V(i,j) = \{A \mid (A \to BC) \in \mathcal{P},\ B \in V(i,k),\ C \in V(k+1,j),\ k \in \{i, \ldots, j-1\}\}$

Pumping Lemma für kontextfreie Grammatiken

Definition 3.31 *Sei L eine Sprache über Σ.*
L hat die **kontextfreie Pump-Eigenschaft**, *falls gilt:*

$$\exists n_L \in \mathbb{N}\, \forall z \in L,\, |z| \geq n_L\, \exists u, v, w, x, y \in \Sigma^* : uvwxy = z \text{ und}$$

(i) $|vwx| \leq n_L$

(ii) $vx \neq \varepsilon$

(iii) $\forall i \geq 0 : uv^i wx^i y \in L$

Satz 3.33 (Pumping-Lemma für kontextfreie Sprachen) *Wenn eine Sprache kontextfrei ist, dann hat sie die kontextfreie Pump-Eigenschaft.*

Beispiele für Sprachen, die die kontextfreie Pump-Eigenschaft haben

- $\{a, ba\}$
- $\{a^n b^n \mid n \geq 1\}$

$L_2 = \{a^n b^n \mid n \geq 1\}$.
Setze $n_L = 4$.
Sei $z \in L_2$ *mit* $|z| \geq n_L = 4$ *beliebig aber fest.*
Es gilt $z = a^j aabbb^j \in L_2$ *mit* $j \geq 0$.
Setze $u = a^j a$, $v = a$, $w = \varepsilon$, $x = b$, $y = bb^j$.

(i) $|vwx| = 2 \leq n_L = 4$ ✓

(ii) $vx \neq \varepsilon$ ✓

(iii) $uv^i wx^i y = a^{i+j+1} b^{i+j+1} = a^k b^k \in L_2$ *für alle* $i \geq 0$, *da* $k \geq 1$ ✓

$\Rightarrow L_2$ *hat die kontextfreie Pump-Eigenschaft.*

Negiertes kontextfreies Pumping-Lemma

Die **Negation der kontextfreien Pump-Eigenschaft** kann folgendermaßen formuliert werden:
Sei L eine Sprache.

$$\forall n_L \in \mathbb{N}\, \exists z \in L,\, |z| \geq n_L\, \forall u, v, w, x, y \in \Sigma^* : uvwxy = z :$$

$$(\underbrace{|vwx| \leq n_L}_{(i)} \wedge \underbrace{vx \neq \varepsilon}_{(ii)}) \Rightarrow \underbrace{\exists i \geq 0 : uv^i wx^i y \notin L}_{\neg(iii)}$$

Beispiele für Sprachen, die die kontextfreie Pump-Eigenschaft nicht haben

- $\{a^n b^n c^n \mid n \geq 1\}$
- $\left\{a^{n^2} \mid n \geq 1\right\}$

$L_3 = \{a^n b^n c^n \mid n \geq 0\}$
Beweis der Negation der kontextfreien Pump-Eigenschaft:
Sei n_L beliebig aber fest.
Setze $z = a^{n_L} b^{n_L} c^{n_L} \in L_3$, also gilt $|z| = 3 \cdot n_L \geq n_L$.
Sei u, v, w, x, y eine beliebige Zerlegung von z mit $z = uvwxy$ und (i) und (ii).

- *Wegen (i) gilt: v und x bestehen aus maximal zwei verschiedenen Sorten von Zeichen.*
- *Wegen (ii) gilt: v und x bestehen aus mindestens einer Sorte von Zeichen.*

Setze nun in (iii) das $i = 0$.
Damit ist die Kopplung der a's an die b's und an die c's verletzt und $uv^0wx^0y = uwy \notin L_3$.
$\Rightarrow L_3$ hat die kontextfreie Pump-Eigenschaft nicht.

$L_4 = \{a^{n^2} \mid n \geq 0\}$
Beweis der Negation der kontextfreien Pump-Eigenschaft:
Sei n_L beliebig aber fest.
Setze $z = a^{n_L^2}$, $|z| = n_L^2 \geq n_L$.
Sei u, v, w, x, y eine beliebige Zerlegung von z mit $z = uvwxy$ und (i) und (ii).
Wähle $i = 2$, dann gilt $uv^iwx^iy = uv^2wx^2y = a^{n_L^2 + |vx|}$.
Damit $a^{n_L^2+|vx|} \in L_4$ muss $n_L^2 + |vx|$ eine Quadratzahl sein.
Es gilt aber $n_L^2 < n_L^2 + \overbrace{|vx|}^{\geq 1} \leq n_L^2 + n_L < n_L^2 + n_L + n_L + 1 = (n_L + 1)^2$
$\Rightarrow n_L^2 + |vx|$ ist keine Quadratzahl und damit $uv^2wx^2y \notin L_4$.
$\Rightarrow L_4$ hat die kontextfreie Pump-Eigenschaft nicht.

Fall $z \in \{c^k y \mid k \geq 1, y \in H_\varepsilon\}$:
Setze $u = \varepsilon, v = z_1 = c, w = \varepsilon, x = \varepsilon, y = z_2 z_3 \ldots z_{|z|}$.

(a) $vx \neq \varepsilon$
(b) $|vwx| \leq n_L$
(c) *Sei i beliebig aber fest:*
  - *Fall $i = 0$:*
    - *Fall $k = 1$:*
      Es gilt $uv^iwx^iy = uv^0wx^0y = uwy \in H_\varepsilon$, da das einzige c entfernt wurde.
      Daraus folgt $uwy \in \{0,1\}^*$, also $uwy \in L_5$.
    - *Fall $k > 1$:*
      Es gilt $uv^iwx^iy = uv^0wx^0y = uwy \in L_5$, da nur ein c entfernt wird und mindestens ein weiteres c übrig ist.
  - *Fall $i > 0$: Es gilt $uv^iwx^iy \in L_5$, da sich nur gleich viele oder mehr c am Anfang des Worts befinden.*

$\Rightarrow L_b$ hat die reguläre Pump-Eigenschaft.

Keller Automaten

Satz 3.36 *L ist kontextfrei $\Leftrightarrow$ Es gibt einen* nichtdeterministischen *Kellerautomaten mit $L(M) = L$.*

Abschlusseigenschaften kontextfreier Sprachen

Satz 3.37 *Die kontextfreien Sprachen sind abgeschlossen unter $\cup, \cdot, ()^*$.*

Satz 3.38 *Die kontextfreien Sprachen sind **nicht** abgeschlossen unter $\cap, \overline{()}$.*