Im folgenden einige Beispielaufgaben und Verdeutlichungen aus meinem Tutorium zur Theoretischen Informatik. Es werden nicht alle Themen der Vorlesung abgedeckt, dementsprechend dient dies nur als zusätzliche Vorbereitungs- bzw. Wiederholungsmöglichkeit für die Klausur. Die Aufgaben wurden möglichst einfach gehalten, um eine schnelle Wiederholung des jeweiligen Konzeptes zu ermöglichen.

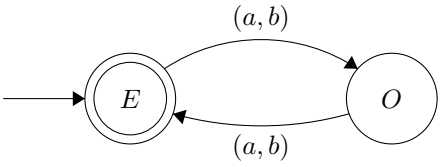
1 Formale Sprachen

1.1 Endliche Automaten

Beispiele zu endlichen Automaten, Determinisierung, Minimierung, Pumping-Lemmata, CYK, Kellerautomaten:

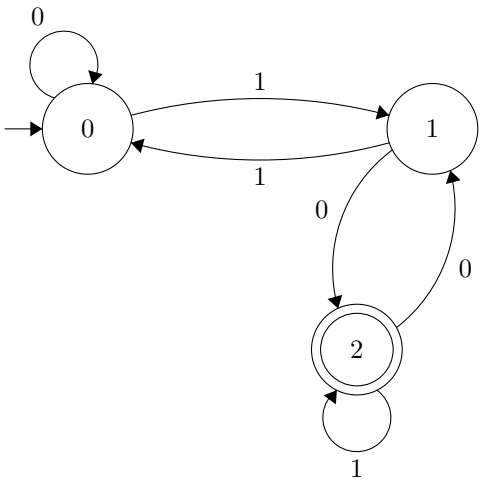
1. Alle Wörter mit gerader Wortlänge

   $\Sigma = \{a, b\}$
   $\mathrm{M} = \{\{E, O\}, \Sigma, \delta, E, E\}$



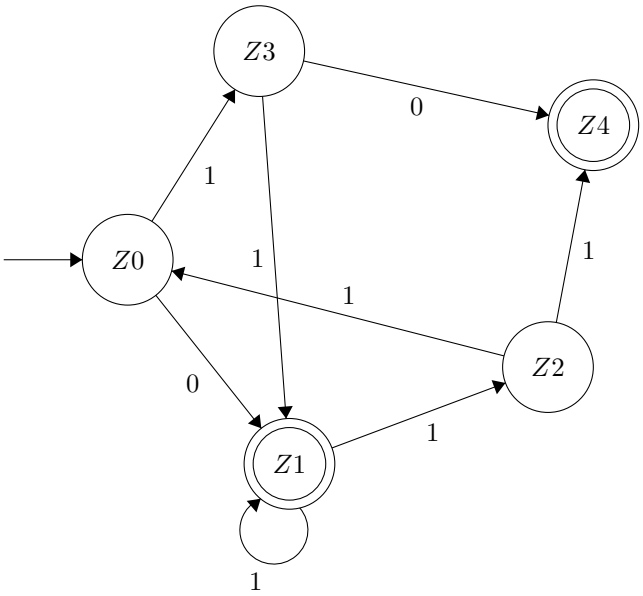
2. $\Sigma = \{0, 1\}$
   $\mathrm{w} \in \Sigma^*$
   Interpretiere w als Binärzahl (MSB links, aufbauend von MSB zu LSB).
   $\mathrm{L} = \{\ \mathrm{w} \in \Sigma^* \mid \mathrm{w} \bmod 3 = 2\ \}$

   $\mathrm{M} = \{\{0, 1.2\}, \Sigma, \delta, 0, 2\}$



1.2 Determinisierung

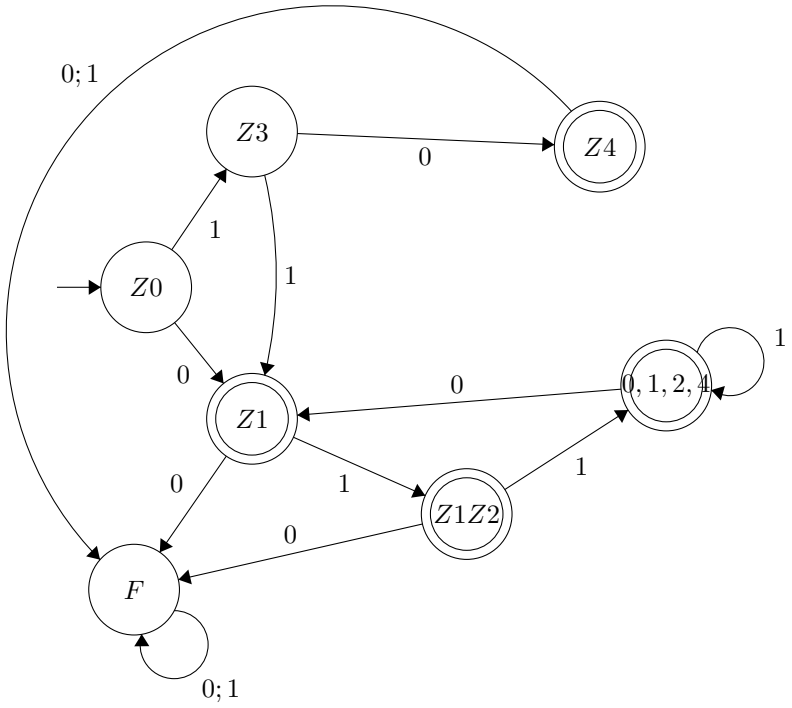
Ausgangsautomat (Tupel analog zu obigen Beispielen):



Anwendung des Determinisierungsalgorithmus':

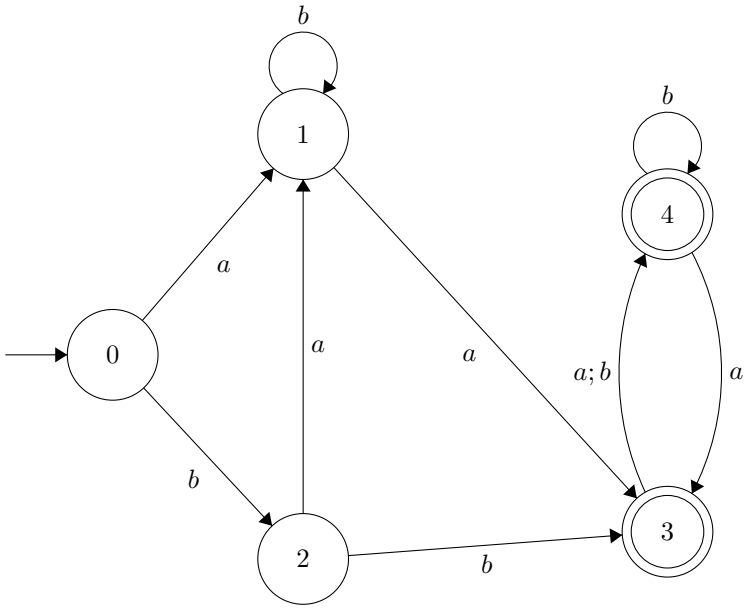
Z	0	1
Z0	Z1	Z3
Z1	F	Z1Z2
Z1Z2	F	Z0Z1Z2Z4
Z0Z1Z2Z4	Z1	Z0Z1Z2Z4
Z3	Z4	Z1
Z4	F	F

Korrekter Automat:



1.3 Minimierung

Der folgende Automat (Tupel analog zu 1.1) sei zu minimieren.
Zu beachten ist hierbei, dass geg. ein nicht deterministischer Automat vorliegt, welcher determinisert werden muss!



Anwenden des Minimierungsalgorithmus':

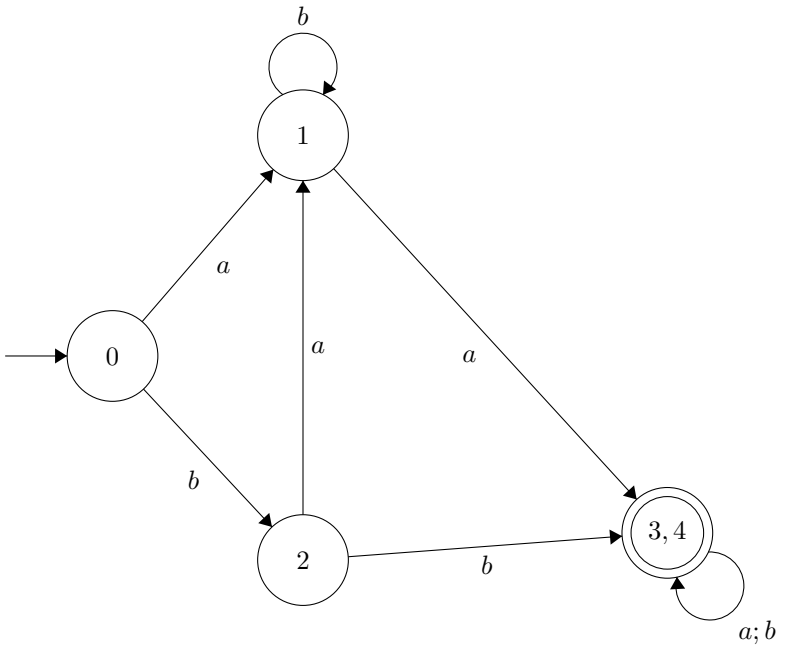
1 Markieren aller Paare (Endzustand, nicht Endzustand)

	0	1	2	3	4
0	-			E	E
1	-	-		E	E
2	-	-	-	E	E
3	-	-	-	-	

2 Iteratives Markieren gemäß Minimierungsalgorithmus:

	0	1	2	3	4
0	-	a	b	E	E
1	-	-	a	E	E
2	-	-	-	E	E
3	-	-	-	-	$\emptyset$

Zustände 3 und 4 können verschmolzen werden:
Der Minimalautomat sieht also folgendermaßen aus:



1.4 Pumping Lemma

z.Z: L= $\{w\$w^R | w \in \Sigma^*\}$ ist nicht regulär.

Anwenden des Pumping Lemmas für reguläre Sprachen:

1. Suchen eines Wortes x in der Sprache L
   x = $1^n\$1^n$ $|x| = 2n+1$ Anmerkung: Es kann auch jedes beliebig andere Wort verwendet werden, so lange es die Bedingung der Sprache erfüllt. Z.B $x = a^n bba^n$. Allerdings: Je einfacher das Wort desto einfach die Durchführung!

2. Aufgrund Bedinung 2 ($|uv| \leq n$) folgt, dass uv nur 1en enthalten kann. Die 1en entstammen dabei alle aus dem Bereich vor dem "$"
   Da v nicht leer sein darf enthält uv mindestens eine 1.

3. $uv^i w \in L \forall i \in \mathbb{N}$

   Sei i=0. $\rightarrow uv^0 w = uw$. Da v nicht leer sein kann entfernen wir vor dem "$" mindestens eine 1. Die rechte Seite bleibt unberührt. Dies stellt einen

Widerspruch dar, da nun eine unterschiedliche Anzahl an 1en vorliegt. Die Sprache ist somit nicht regulär!

Ein formalerer Ansatz wäre:
Zerlege x in:
u=1^l v=1^m w=$1^{n-(l+m)}\$1^n$

Sei nun i=2 $\rightarrow uvvw = 1^{l+2m+(n-(l+m))}\$1^n = 1^{n+m}\$1^n$. Da $m \geq 1 \rightarrow uv^iw \notin L$. Also ist L nicht regulär!

1.5 CYK

G= ($\{A, B, S\}$, $\{+, -, d\}$, P, S) (in Chomsky Normalform)

P:

S → AB
B → SS
A → +
A → -
S → d

Kann durch die gegebne Grammatik folgendes Wort abgeleitet werden?:

++ddd

+	+	d	d	d
A	A	S	S	S
-	-	B	B	
-	S	-		
-	B			
S				

Da im unteren Feld die Startvariable steht, kann das Wort also von der gegebenen Grammatik abgeleitet werden.

1.6 Pumping Lemma (kontextfreie Sprachen)

z.Z: $L = \{a^j b^j c^k | i > j > k\}$ nicht kontextfrei
x = $a^{n+2}b^{n+1}c^n$
$|x|$ = 3n+3
uv^iwx^iy
$|vx| > 0$
$|vwx| \leq n$

Fälle:
$|vx|$ besteht nur aus a's → Sei i=0,vx nicht leer. Es fällt min. 1 a raus. Somit #a=#b $\notin L$

$|vx|$ besteht nur aus b's → Sei i=2,vx nicht leer. Es wird min.1 b gepumpt. Somit #a $\leq$ #b $\notin L$

$|vx|$ besteht nur aus c's → Sei i=2,vx nicht leer. Es wird min.1 c gepumpt. Somit #b $\leq$ #c $\notin L$

Das waren die "einfacheren" Fälle

vx besteht aus a's und b's:

(a) v enthält nur a's und x enthält nur b's.
Sei i=0 → $\#b \leq \#c \notin L$

(b) v enthält bereits a's und b's
Sei i=2. Dies führt zu einem Wort der Form aaaa...aba.... $\notin L$

(c) vx aus a's und c's nicht möglich aufgrund: $|vwx| \leq n$

vx aus b's und c's analog

1.7 Kellerautomaten

z.Z: L=$\{w\$w^R|w \in \Sigma^*\}$ ist kontextfrei

Zeigen durch Konstruktion eines (deterministischen) Kellerautomaten

M = $(\{Z_0, Z_1\}, \{a, \$, b\}, \delta, \{A, B\}, Z_0, \#, E)$

$\delta = \{$
$(Z_0, a, \#) \to (Z_0, A\#)$
$(Z_0, b, \#) \to (Z_0, B\#)$
$(Z_0, \$, \#) \to (E, \epsilon)$
$(Z_0, a, A) \to (Z_0, AA)$
$(Z_0, b, A) \to (Z_0, BA)$
$(Z_0, a, B) \to (Z_0, AB)$
$(Z_0, b, B) \to (Z_0, BB)$
$(Z_0, \$, A) \to (Z_1, A)$
$(Z_0, \$, B) \to (Z_1, B)$
$(Z_1, a, A) \to (Z_1, \epsilon)$
$(Z_1, b, B) \to (Z_1, \epsilon)$
$(Z_1, \epsilon, \#) \to (Z_1, \epsilon)\}$

Damit ist L kontextfrei.

WICHTIG!: Bei Kellerautomaten ist die, durch die deterministische Variante erkannte Menge **echt kleiner**, als die der nicht-deterministischen Version!
Die Sprache L=$\{ww^R|w \in \Sigma^*\}$ ist **nicht** durch einen deterministischen Kellerautomaten erkennbar, da die Wortmitte nicht bestimmt ist.

2 Berechenbarkeit

2.1 Turingmaschine

Addieren zweier unärer Zahlen. Die Zahlen sind durch ein "$" getrennt.

M=$(\{Z_0, Z_1\}, \{1, \$\}, \{1, \$, \square\}, \delta, Z_0, \square, E)$

$\delta(Z_0, 1) = \delta(Z_0, 1, R)$
$\delta(Z_0, \$) = \delta(Z_0, 1, R)$
$\delta(Z_0, \square) = \delta(Z_1, \square, L)$
$\delta(Z_1, 1) = \delta(E, \square, N)$

Ablauf: Durchgehen bis zum "$". Ersetzen des "$" durch eine 1. Löschen der letzten 1.

Multiplikation einer Binärzahl mit 8 (ohne führende Nullen):

M=$(\{Z_0, Z_1, Z_2\}, \{0, 1\}, \{0, 1, \square\}, \delta, Z_0, \square, E)$

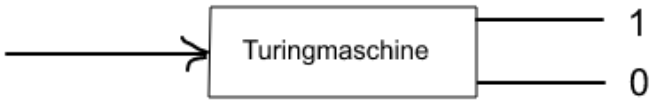
$\delta(Z_0, 1) = \delta(Z_0, 1, R)$
$\delta(Z_0, 0) = \delta(Z_0, 0, R)$
$\delta(Z_0, \square) = \delta(Z_1, 0, R)$
$\delta(Z_1, \square) = \delta(Z_2, 0, R)$
$\delta(Z_2, \square) = \delta(E, 0, N)$

Ablauf: Durchgehen bis an das Ende der Binärzahl. Rechtsshift entspricht Multiplikation mit 2. 3 Shifts (Anhängen von drei Nullen) entspricht Multiplikation mit 8.

2.2 Entscheidbarkeit

Definition Entscheidbarkeit:

$$\chi_A(w) = \begin{cases} 1 \text{ falls w} \in \mathsf{A} \\ 0 \text{ falls w} \notin \mathsf{A} \end{cases} \tag{1}$$



Sind folgende Probleme entscheidbar?

Eine Turingmaschine hält (auf dem leeren Band) innerhalb maximal 20 Schritte
Dieses Problem ist entscheidbar!
Simuliere diese Turingmaschine(Lasse sie laufen). Hält sie innerhalb der Schritte gebe 1 aus. Ansonsten gebe 0 aus.

Eine Turingmaschine hält nach mindestens 20 Schritten. Sie muss also mindestens 20 Schritte durchführen und darf dann erst halten.

Dieses Problem ist nicht entscheidbar!
Um dies zu zeigen, können wir das Halteproblem auf leerem Band auf das Problem reduzieren. Das Halteproblem (auf leerem Band) ist nicht entscheidbar.Hierfür definieren wir uns zuerst die Sprache

$$\text{L}=\{w \in \{0,1\}^* | M_w \text{hält nach mehr als 20 Schritten }\}$$

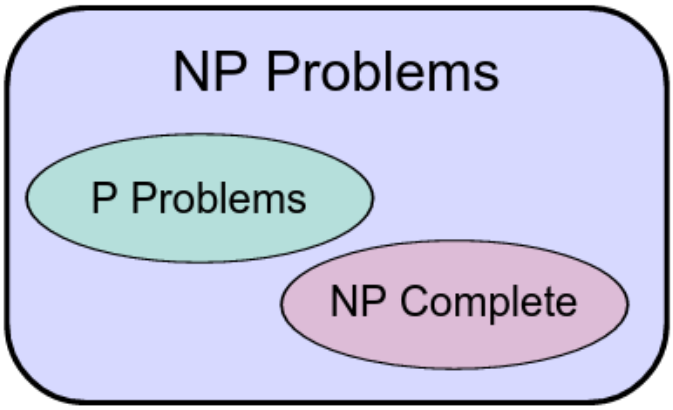
z.Z $H_0 \leq L$:
Die Funktion $f\{0,1\}^* \to \{0,1\}^*$ wird so definiert, dass $\text{M}_{f(w}$ die Eingabe auf ein anderes Band schiebt und dann 20 Schritte (auf dem nun leeren Band)ausführt. Dann startet sie M_w auf dem leeren Band. Nun gilt es noch zu zeigen:

$w \in H_0$: Dann hält M_w auf dem leeren Band. $M_{f(w)}$ hält also nach mindestens 20 Schritten. Also: $f(w) \in L$
$w \notin H_0$: Dann hält M_w nicht auf dem leeren Band. $M_{f(w)}$ hält nicht nach mindestens 20 Schritten. Also: $f(w) \notin L$
Somit ist das Problem nicht entscheidbar!

3 Komplexitätstheorie

(Basis-)Komplexitätsklassen:

P: Es existiert eine deterministische TM, welche das Problem in polyn. Zeit löst.
NP: Es existiert eine **nicht-deterministische TM**, welche das Problem in **polyn. Zeit** löst.
NP-Hart: Mindestens so schwer wie alle Probleme in NP. Das bedeutet: Alle Probleme in NP können auf ein NP-hartes Problem reduziert werden.
NP-Vollständig: Damit ein Problem als NP-Vollständig gilt, muss es 1: in NP liegen und 2: NP-hart sein.

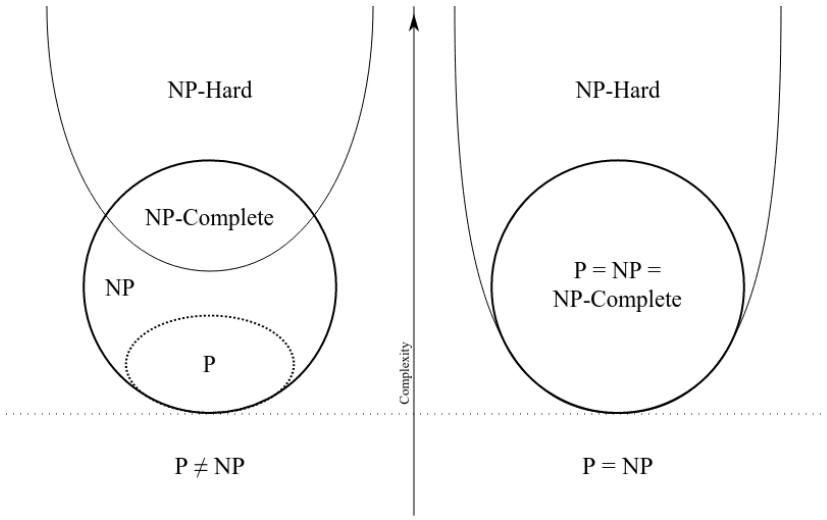


[1]

Frage: Ist eine Reduktion von einem Problem in P auf ein NP-hartes Problem bedeutungsvoll?
Da sich alle Probleme in NP(also auch alle Probleme in P) auf die NP-harten Probleme reduzieren lassen, ist mit einer solchen Reduktion im Allg. nichts von Bedeutung für die PNP Problematik gezeigt.

[1] https://upload.wikimedia.org/wikipedia/commons/b/bc/Complexity_classes.svg

Das folgende Schaubild zeigt die Verhältnisse von P-NP-NP-Vollst.-NP-Hart im Vergleich der beiden Möglichkeiten P=NP und P≠NP



[2]

Um zu zeigen, dass ein Problem in NP liegt, genügt es eine mögliche Lösung zu erraten und diese Lösung in Polynomzeit zu überprüfen. Diese Zeitabschätzung muss aber gezeigt werden.

Um zu zeigen, dass ein Problem NP-hart ist, können wir ein anderes NP-hartes Problem auf das Problem reduzieren. Also $NP-Hart \leq Problem$. Diese Reduktion muss allerdings ebenfalls in Polynomzeit durchgeführt werden können.

Mit Hilfe dieser Reduktion zeigen wir, dass unser Problem mindestens genauso schwer wie das NP-Harte Problem ist. Es kann allerdings auch deutlich schwerer sein.
Die Polynomzeitreduktion $P \leq NP$ ist also machbar, da NP mindestens genauso schwer(oder schwerer). Die Reduktion $NP \leq P$ hingegen ist ein Widerspruch und macht Sie zum Millionär!

[2] `https://upload.wikimedia.org/wikipedia/commons/a/a0/P_np_np-complete_np-hard.svg`