FRIEDRICH-ALEXANDER-UNIVERSITÄT ERLANGEN-NÜRNBERG
Professur für Didaktik der Informatik

Lösungs*vorschläge* zu den Staatsexamina:
Theoretische Informatik und Algorithmik

Frühjahr 2023

Inhaltsverzeichnis

1 Thema

1.1 Teilaufgabe 1: Algorithmik

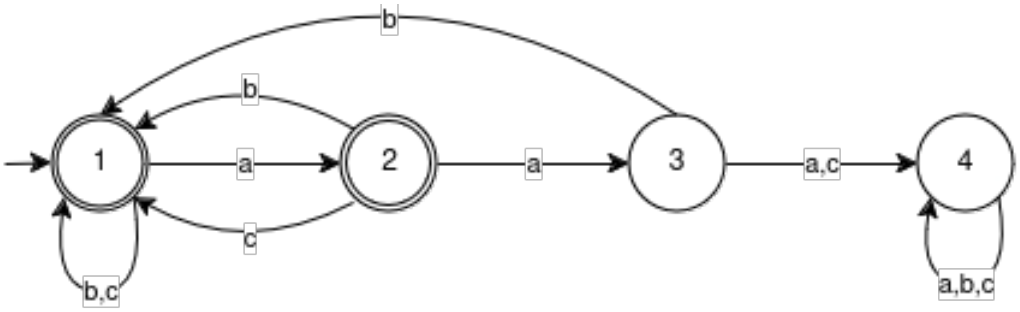
1.2 Teilaufgabe 2: Theoretische Informatik

1.2.1 Aufgabe 1: Reguläre Sprachen

(a) $DEA = (\{1,2,3,4\}, \{a,b,c\}, \delta, 1, \{1,2\})$, wobei δ gegeben ist durch:

Zustand	a	b	c
1	2	1	1
2	3	1	1
3	4	1	4
4	4	4	4

Grafische Darstellung:



(b) Die Anzahl der Äquivalenzklassen kann die Anzahl an Zuständen nicht überschreiten, die der DEA aufweist, welcher diese Sprache erkennt. Der DEA aus a) verfügt über vier Zustände, also hat L_1 maximal vier Äquivalenzklassen.

Wir finden vier unterschiedliche Äquivalenzklassen zu den Repräsentanten ε, a, aa und aaa:

- Da $\varepsilon \circ a \in L_1$, aber $a \circ a \notin L_1$ gilt $[\varepsilon] \nsim_{L_1} [a]$.
- Da $\varepsilon \circ a \in L_1$, aber $aa \circ a \notin L_1$ gilt $[\varepsilon] \nsim_{L_1} [aa]$.
- Da $\varepsilon \circ a \in L_1$, aber $aaa \circ a \notin L_1$ gilt $[\varepsilon] \nsim_{L_1} [aaa]$.
- Da $a \circ c \in L_1$, aber $aa \circ c \notin L_1$ gilt $[a] \nsim_{L_1} [aa]$.
- Da $a \circ c \in L_1$, aber $aaa \circ c \notin L_1$ gilt $[a] \nsim_{L_1} [aaa]$.
- Da $aa \circ b \in L_1$, aber $aaa \circ b \notin L_1$ gilt $[aa] \nsim_{L_1} [aaa]$.

Drei Wörter je Äquivalenzklasse:

1. $bc, ccc, bbb \in [\varepsilon]$
2. $a, aca, aba \in [a]$

3. $aa, aabaa, abaa \in [aa]$
4. $aaa, aac, aaabc \in [aaa]$

(c) **Übergangstabelle**

Zustand	a-Übergang	b-Übergang	ϵ-Übergang
1	2	1	4
2	3	$\varnothing$	$\varnothing$
3	1	2	1
4	$\varnothing$	3	$\varnothing$

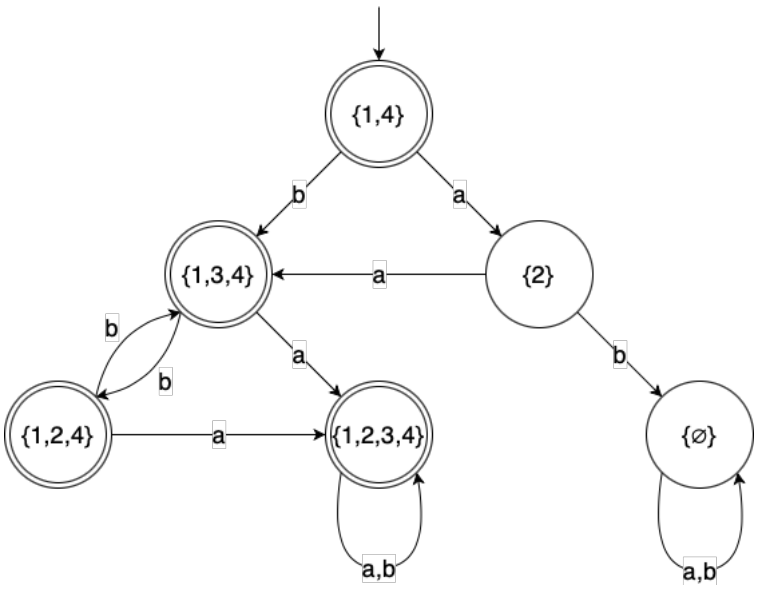
ϵ-Funktionsabschlusstabelle

Zustand	Funktionsabschluss
1	{1,4}
2	{2}
3	{1,3,4}
4	{4}

Pontenzmengenkonstruktionstabelle

Zustand	a-Übergang	b-Übergang	Endzustand?
{1,4}	{2}	{1,3,4}	Ja
{2}	{1,3,4}	{∅}	
{1,3,4}	{1,2,4}	{1,2,3,4}	Ja
{1,2,4}	{1,2,3,4}	{1,3,4}	Ja
{1,2,3,4}	{1,2,3,4}	{1,2,3,4}	Ja
{∅}	{∅}	{∅}	

Fertiger DEA



1.2.2 Aufgabe 2: Regularität und Kontextfreiheit

(a) $L_1 = \{a^m b^n c^n \mid n, m \geq 1; n, m \in \mathbb{N}\} \cup \{b^m c^n \mid n, m \geq 0; n, m \in \mathbb{N}\}$

Pumpinglemma für reguläre Sprachen
Ist L_1 regulär, so gilt:
$\exists p \geq 1 : \forall z \in L, |z| \geq p :$
$\exists u, v, w \in \Sigma^* : z = uvw$ mit

1. $|v| \geq 1$
2. $|uv| \leq p$
3. $\forall i \in \mathbb{N} : uv^i w \in L_1$

Nehmen wir an L_1 sei regulär, dann können wir das Pumpinglemma anwenden und folgern:

Sei $p \geq 1$ die Pumpingzahl. Wir wählen $z = ab^p c^p \in L_1$ mit $|z| = 2p + 1 \geq p$.
Aus $|uv| \leq p$ folgt: uv ist von der Form ab^{p-n} für ein $n \in \{1, 2, ..., p\}$ und w ist von der Form $b^n c^p$.
Aus $|v| \geq 1$ folgt: v ist von der Form ab^k oder b^{k+1} mit $k \in \{0, 1, ..., p - n\}$.
Im Fall $v = ab^k$ ist $uv^0 w = b^{p-k} c^p \notin L_1$, da $m = 0$.
Im Fall $v = b^{k+1}$ ist $uv^0 w = ab^{p-k-1} c^p \notin L_1$, da $p - k - 1 \neq p$.

Das ist ein Widerspruch zur Annahme L_1 sei regulär. $\Rightarrow L_1$ ist nicht regulär!

(b) L_2 ist regulär und wird von folgendem regulären Ausdruck erzeugt: $abc(abc)^* = (abc)^+$

Das lässt sich begründen, indem man sich die Reihenfolge der Buchstaben ansieht. L_2 ist so definiert, dass jedes Wort in der Sprache mit a beginnen muss, darauf folgt ein b und dann ein c, worauf wieder nur ein a fogen kann u.s.w.. Zudem muss jedes Wort auf ein c enden und mindestens einmal die Kombination abc enthalten (der Rest fällt durch $n = 0$ weg).

(c) Als Nachweis für die Kontextfreiheit geben wir eine kontextfreie Grammatik an, die L_3 erzeugt:

S → ASE | T | abTde
T → BTD | c
A → aa
B → bb
E → ee
D → dd

Erklärung:
Die Geradheit der Summe wird durch die Ableitungsregeln mit Doppel-Terminalen auf der rechten Seite gewährleistet. Somit können bei jeder Regel (außer für c) nur 2 Buchstaben pro Seite dazu kommen und die SUmme ist insgesamt immer ein Vielfaches von 2, also gerade.
Der Fall, dass die A's, B's und somit auch die D's, E's jeweils eine ungerade Anzahl haben,

wird durch die Regel $S \to abTde$ abgedeckt.

(d) $L_4 = \{w_1 c^n w_2 \mid n \geq 0, n \in \mathbb{N}, w_1, w_2 \in \{a,b\}^*, \#_a(w_1) > n \; und \; \#_b(w_2) < n\}$

Pumpinglemma für kontextfreie Sprachen
Ist L_4 eine kontextfreie Sprache, so gilt:
$\exists p \in \mathbb{N} : \forall z \in L, |z| \geq p :$
$\exists u,v,w,x,y \in \Sigma^* : z = uvwxy$ mit

1. $|vx| \geq 1$
2. $|vwx| \leq p$
3. $\forall i \in \mathbb{N} : uv^i wx^i y \in L_4$

Nehmen wir an L_4 sei kontextfrei, dann können wir das Pumpinglemma anwenden und folgern:

Sei $p \in \mathbb{N}$ die Pumpingzahl. Wir wählen $z = a^{p+1}c^p b^{p-1} \in L_4$ mit $|z| = 3p > p$.
Aus $|vwx| \leq p$ folgt: vwx kann nicht aus a's, c's und b's bestehen. Die Form von vwx entspricht also einem der beiden Fälle:

1. $a^i c^j$ für $i \in \{1,2,...,p+1\}$ und $j \in \{1,2,...,p\}$ mit $i+j \leq p$
2. $c^i b^j$ für $i \in \{1,2,...,p\}$ und $j \in \{1,2,...,p-1\}$ mit $i+j \leq p$

Aus $|vx| \geq 1$ ergeben sich dann für v und x folgende Fälle (mit $k+l+m \geq 1$):

- Fall 1.1: $v = a^k c^l$ und $x = c^m$
- Fall 1.2: $v = a^k$ und $x = a^l c^m$
- Fall 2.1: $v = c^k b^l$ und $x = b^m$
- Fall 2.2: $v = c^k$ und $x = c^l b^m$

In Fall 1.1 ist $uv^0wx^0y = a^{p+1-k}c^{p-l-m}b^{p-1} \notin L_4$, da entweder $p-l-m \not> p-1$ oder falls $l = m = 0 : \; k \geq 1 \Rightarrow p+1-k \not> p-l-m$ gilt.
In Fall 1.2 ist $uv^0wx^0y = a^{p+1-k-l}c^{p-m}b^{p-1} \notin L_4$, da entweder $p-m \not> p-1$ oder falls $m = 0 : \; k+l \geq 1 \Rightarrow p+1-k-l \not> p-m$ gilt.
In Fall 2.1 ist $uv^2wx^2y = a^{p+1}c^{p+k}b^{p-1+l+m} \notin L_4$, da entweder $p+1 \not> p+k$ oder falls $k = 0 : \; l+m \geq 1 \Rightarrow p+k \not> p-1+l+m$ gilt.
In Fall 2.2 ist $uv^2wx^2y = a^{p+1}c^{p+k+l}b^{p-1+m} \notin L_4$, da entweder $p+1 \not> p+k+l$ oder falls $k = l = 0 : \; m \geq 1 \Rightarrow p+k+l \not> p-1+m$ gilt.

Das ist ein Widerspruch zur Annahme L_4 sei kontextfrei. $\Rightarrow L_4$ ist nicht kontextfrei!

1.2.3 Aufgabe 3: Entscheidbarkeit

(a) $L_1 - L_2 = \{w \in L_1 | w \notin L_2\}$ Es handelt sich bei $L_1 - L_2$ also um die Sprache aller Wörter aus L_1, die in L_2 nicht enthalten sind.

Das Wortproblem für reguläre Sprachen ist entscheidbar (z.B. durch einen entsprechenden DEA) und zwar in $O(n)$, da die Eingabe lediglich einmal abgelaufen werden muss. Der CYK-Algorithmus liefert den Nachweis, dass auch das Wortproblem für kontextfreie Grammatiken mit einer Zeitkomplexität von $O(n^3)$ entscheidbar ist. Somit sind also L_1 und L_2 von einer deterministische Turingmaschine (DTM) in polynomieller Zeit entscheidbar.

Wir können also eine DTM bauen, die folgendermaßen funktioniert:

- Prüfe in polynomieller Zeit, ob die Eingabe ein Wort in L_1 ist.
- Falls diese Prüfung zutreffend war, prüfe weiter, ob die Eingabe ein Wort in L_2 ist. Auch das erfolgt nach der vorherigen Begründung in polynomieller Zeit.
- Ergab die erste Prüfung $wahr$ und die zweite Prüfung $falsch$, ist die Eingab ein Wort in L. Diese Fallunterschiedung ist problemlos in konstanter Zeit feststellbar.

Die Sprache $L_1 - L_2$ kann somit durch eine DTM in polynomieller Zeit ($O(n) + O(n^3) + O(1) = O(n^3)$) entschieden werden, also liegt $L_1 - L_2$ in $\mathcal{P}$.

(b) $L \cup \overline{L} = L \cup (\Sigma^* \setminus L) = \Sigma^*$
Also ist $L \cup \overline{L}$ die Sprache *aller* Wörter. Diese Sprache ist regulär und somit entscheidbar.
$L \cap \overline{L} = L \cap (\Sigma^* \setminus L) = \varnothing$
Also ist $L \cap \overline{L}$ die *leere* Sprache. Auch diese Sprache ist regulär und somit entscheidbar.

Somit sind $L \cap \overline{L}$ und $L \cup \overline{L}$ für alle Sprachen L entscheidbar, also insbesondere auch wenn L semi-entscheidbar ist.

(c) Da L semi-entscheidbar ist, gibt es eine Turing Maschine (TM) M_1, die genau dann auf Eingabe w hält und akzeptiert, wenn $w \in L$ gilt.
Analog gilt für $\overline{L}$, dass es eine TM M_2 gibt, die genau dann auf Eingabe w hält und akzeptiert, wenn $w \in \overline{L}$ gilt.

Man kann also eine TM M mit zwei Bändern konstruieren, welche zunächst die Eingabe w auf das zweite Band kopiert und anschließend durch abwechselnde Schritte auf dem ersten Band M_1 und auf dem zweiten Band M_2 simuliert. Falls M_1 hält und akzeptiert, tut auch M dies; falls M_2 hält und akzeptiert, hält M und akzeptiert *nicht*.
Diese TM hält auf jeder Eingabe $w \in \Sigma^*$, da entweder $w \in L$ und somit M_1 hält oder $w \in \overline{L}$ und somit M_2 hält. M akzeptiert genau die Wörter, die auch M_1 akzeptiert, also genau die Eingaben $w \in L$.
Also ist L durch M entscheidbar.

(d) Die Aussage ist korrekt. Da L in $\mathcal{NP}$ liegt, gibt es eine NTM M, die L in polynomieller Laufzeit entscheidet. Generell sind NTM und DTM aber gleichmächtig. Somit gibt es eine DTM M', welche die gleiche Sprache wie M erkennt (dabei allerdings deutlich mehr Berechnungsschritte benötigen kann), und damit auch L entscheidet.
Folglich ist L entscheidbar.

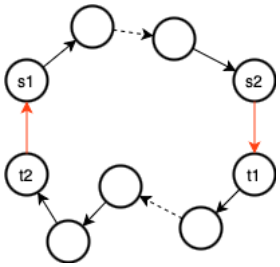
1.2.4 Aufgabe 4: Komplexität

(a) Die NP-Härte von SC wird durch eine polynomielle Reduktion auf das bereits als NP-vollständig deklarierte Problem $2VDP$ nachgewiesen. Dazu wird eine totale und berechenbare Funktion f benötigt, die Probleme aus $2VDP$ als kodierte Wörter auf Probleme aus SC abbildet:

Sei $f : \Sigma^* \to \Sigma^*$ definiert über

$$f(w) = \begin{cases} c(V, E \cup \{(s_2, t_1), (t_2, s_1)\}, (s_2, t_1), (t_2, s_1)) & , falls\ w = c(V, E, s_1, s_2, t_1, t_2) \\ & mit\ einem\ Graph\ G = (V, E) \\ & und\ s_1, s_2, t_1, t_2 \in V \\ \varepsilon & , sonst \end{cases}$$

Die hinter der Konstruktion liegende Idee wird durch folgende Grafik veranschaulicht:



f ist offensichtlich total. Außerdem lässt sich eine DTM konstruieren, die f in polynomieller Laufzeit berechnet:

- Syntaxcheck, ob $w = c(V, E, s_1, s_2, t_1, t_2)$ mit einem Graph $G = (V, E)$ und $s_1, s_2, t_1, t_2 \in V$ (in $O(|V| + |E|) = O(n)$)
- Passendes zusammensetzen der Ausgabe zu $c(V, E \cup \{(s_2, t_1), (t_2, s_1)\}, (s_2, t_1), (t_2, s_1))$ (in $O(1)$)

Es bleibt noch zu zeigen, dass $w \in 2VDP \Leftrightarrow f(w) \in SC$ gilt. Dies beweisen folgende Äquivalenzumformungen ($\forall w \in \Sigma^*$):

$$\begin{aligned} w \in 2VDP \iff & w = c(V, E, s_1, s_2, t_1, t_2)\ mit\ Graph\ G = (V, E)\ und\ s_1, s_2, t_1, t_2 \in V \\ & \wedge\ \exists\ Pfade\ p_1 = s_1 ... s_2\ und\ p_2 = t_1 ... t_2\ in\ G : \forall\ u \in p_1,\ v \in p_2 : u \neq v \\ \Leftrightarrow & f(w) = c(V, E \cup \{(s_2, t_1), (t_2, s_1)\} = E', (s_2, t_1), (t_2, s_1))\ mit\ Graph \\ & G' = (V, E')\ und\ s_1, s_2, t_1, t_2 \in V\ \wedge\ \exists\ Pfade\ p_1 = s_1 ... s_2\ und\ p_2 = t_1 ... t_2\ in\ G' : \\ & \forall\ u \in p_1,\ v \in p_2 : u \neq v \\ \Leftrightarrow & f(w) = c(V, E', (s_2, t_1), (t_2, s_1))\ mit\ Graph\ G' = (V, E')\ und\ (s_2, t_1), (t_2, s_1) \in E' \\ & \wedge\ \exists\ Pfad\ p = s_1 ... s_2 t_1 ... t_2 s_1\ in\ G' :\ \forall\ n, m \leq k := |p| : p_n \neq p_m,\ außer\ p_1 = p_k \\ \Leftrightarrow & f(w) \in SC \end{aligned}$$

Informelle Beschreibung: Zusammensetzen der zwei knoten-disjunkten Pfade über zwei neue Kanten, die jeweils die Endpunkte verbinden, zu einem einfachen Kreis.

Somit gilt $2VDP \leq_p SC$ und da $2VDP$ NP-vollständig ist, muss SC NP-hart sein.

Um noch zu zeigen, dass SC in NP liegt, muss eine NTM skizziert werden, die SC in polynomieller Zeit entscheidet:

1. Starte mit einem leeren Pfad durch den Graphen ($O(1)$).
2. Rate nun aus den vom letzten Knoten aus erreichbaren Knoten, die noch nicht im Pfad vorkommen (außer dem Startknoten), nichtdeterministisch den nächsten Knoten des Pfades, sodass am Ende (falls dieser existiert) ein Simple Circle gefunden wird ($O(n)$).
3. Wiederhole den vorherigen Schritt solange, bis der Startknoten wieder erreicht wird - in diesem Fall wurde eine Lösung gefunden $\Rightarrow$ halten und akzeptieren - oder keine weiteren Knoten hinzugefügt werden können - in diesem Fall gibt es keine Lösung $\Rightarrow$ halten und nicht akzeptieren ($O(n)$).

Es folgt, dass SC in NP liegt und somit NP-vollständig ist.

(b) Folgender Algorithmus löst USC:

1. Überprüfe, ob $e_1 = e_2$.
2. Falls $e_1 = e_2$ und $e_1 = (e_1', e_1''),\ e_2 = (e_2', e_2'')$ starte A unter Eingabe e_1', e_1'', e_2'', e_2'
   - Falls A eine Lösung ausgibt, ist der Pfad $p = p_1 \circ p_2$ eine Lösung von USC
   - Falls A keine Lösung findet, gibt es auch für USC keine Lösung
3. Falls $e_1 \neq e_2$ und $e_1 = (e_1', e_1''),\ e_2 = (e_2', e_2'')$ starte A unter Eingabe e_1'', e_2'', e_2', e_1'
   - Falls A eine Lösung ausgibt, ist der Pfad $p = p_1 \circ e_2' e_2'' \circ p_2 \circ e_1' e_1''$ eine Lösung von USC
   - Falls A keine Lösung findet, gibt es auch für USC keine Lösung

Anmerkung: $\circ$ meint hier die Aneinanderreihung oder Konkatenation von Pfaden. Dabei werden die Knoten in den Pfaden zusammen hintereinander geschrieben, wobei der letzte Knoten des ersten Pfades mit dem ersten Knoten des zweiten Pfades übereinstimmen muss und nur einmal nidergeschrieben wird. Bsp.: $abc \circ cde = abcde,\ \forall a, b, c, d, e \in V$.

Laufzeitanalyse:
Die Überprüfung in Schritt 1, das Starten von A sowie die Überprüfung und Rückgabe in Schritt 2 und 3 laufen jeweils in $O(1)$. Der Durchlauf von A in Schritt 2 und 3 hat höchstens eine polynomielle Laufzeit in der Größe der Eingabe, da U2VDP in $\mathcal{P}$ liegt.
Somit hat der beschriebene Algorithus insgesamt höchstens eine polynomielle Laufzeit in der Größe der Eingabe.

Folglich liegt USC in $\mathcal{P}$.

2 Thema

2.1 Teilaufgabe 1: Algorithmik

2.2 Teilaufgabe 2: Theoretische Informatik

2.2.1 Aufgabe 1: Reguläre Sprachen

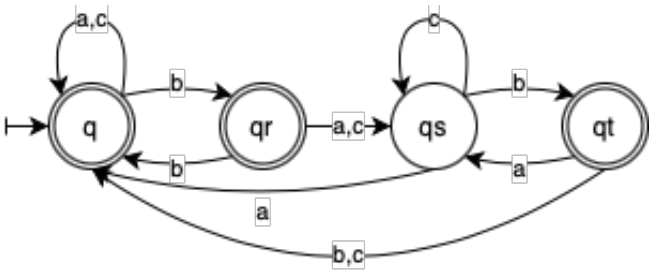
(a) $L = (a|b|c)^*b(a|c)(c|ba)^*$

(b) Schritt 1: Umwandlung des NEA zum DEA mittels Potenzmengenkonstruktion:

Zustand	Eingabe a	Eingabe b	Eingabe c	Final	Anmerkung
q	q	qr	q		Neuer Zustand qr
qr	qs	q	qs		Neuer Zustand qs
qs	q	qt	qs	✓	Neuer Zustand qt
qt	qs	q	q		

Der Tabelle können aus der ersten Spalte die für den DEA benötigten Zustände entnommen werden. Zudem werden die Übergänge nach Eingabe von a, b oder c aufgeführt und die Endzustände gekennzeichnet.

Schritt 2: Der gesuchte DEA soll aber nicht L, sondern das Komplement von L erkennen. Um das zu erreichen, werden Endzustände und Nicht-Endzustände vertauscht. Hier die graphische Darstellung des gesuchten DEA, der $\overline{L}$ akzeptiert:

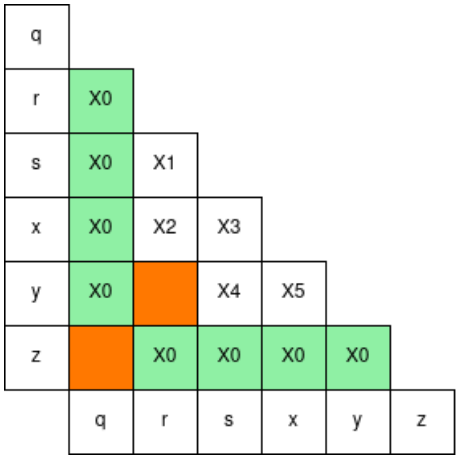


(c) Schritt 1: Angabe einer Zeugentabelle zur Überprüfung der Erreichbarkeit:

Zustand	q	r	s	t	x	y	z
Zeuge	1	11	ϵ	-	01	0	00

Der Zustand t ist also nicht erreichbar und entfällt im minimierten Automaten.

Schritt 2: Identifizieren nicht äquivalenter Zustände über Tabelle mit Zustandspaaren:



Tabellenfüllverfahren
Zunächst werden alle Zustandspaare, die aus einem Endzustand und einem Nicht-Endzustand bestehen, mit X0 markiert. Weitere Kreuze in der Tabelle findet man mittels folgender Übergangstabelle von Zustandspaaren:

Zustandspaar	0	1	Erläuterung
(s,r)	(y,q)	(q,x)	Eingabe 0 führt mit (y,q) zu X0. Ergänze X1.
(x,r)	(r,q)	(z,x)	Eingabe 0 führt mit (r,q) zu X0. Ergänze X2.
(x,s)	(r,y)	(z,q)	Eingabe 1 führt mit (z,q) zu X0. Ergänze X3.
(y,s)	(z,y)	(x,q)	Eingabe 1 führt mit (z,q) zu X0. Ergänze X4.
(y,x)	(z,r)	(z,x)	Eingabe 1 führt mit (z,x) zu X0. Ergänze X5.
(z,q)	(x,x)	(y,r)	z und q sind äquivalent.
(y,r)	(z,q)	(x,x)	y und r sind äquivalent.

Die Zustände z und q sowie y und r sind äquivalent, da auch bei wiederholter Überprüfung keine Übergänge gefunden werden, die zu einem Kreuz führen. Damit ergeben sich für den minimierten Automaten folgende Zustände, die den Äquivalenzklassen der Zustände von A entsprechen: $[z,q],[y,r],[s],[x]$

Der minimierte Automat A' lautet also:
$A' = (\{[q,z],[y,r],[s],[x]\},\{0,1\},\delta',[s],\{[q,z]\})$ mit

δ'	0	1
[s]	[y,r]	[q,z]
[x]	[y,r]	[q,z]
[y,r]	[q,z]	[x]
[q,z]	[x]	[y,r]

2.2.2 Aufgabe 2: Kontextfreie Sprachen

(a) $G = (\{S, A, B\}, \Sigma, P, S)$ mit P:

$$\begin{array}{lcl} S & \rightarrow & AB \mid \epsilon \\ A & \rightarrow & 0A1 \mid 2A \mid \epsilon \\ B & \rightarrow & 1B2 \mid B0 \mid \epsilon \end{array}$$

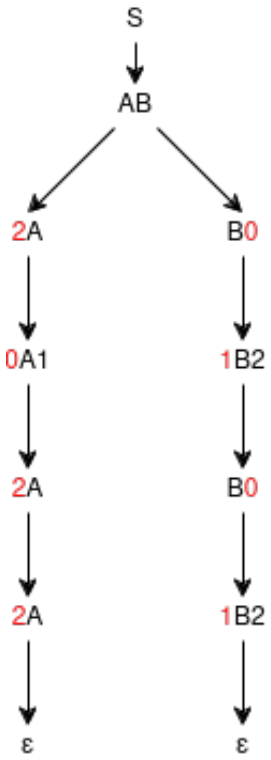
Konstruktionsidee
Im Kern beruht diese Grammatik auf der Idee, dass jedes Wort in L aus zwei Teilwörtern besteht:

- einem linken Teilwort x, dass an seinem rechten Rand (also dem Mittelteil des Gesamtwortes) so viele 1en aufweist wie es selbst 0en enthält, wobei links und rechts aller 0en beliebig viele 2en zulässig sind, und analog dazu
- einem rechten Teilwort y, dass an seinem linken Rand (also dem Mittelteil des Gesamtwortes) so viele 1en aufweist wie es selbst 2en enthält, wobei links und rechts der 2en beliebig viele 0en zulässig sind.

Somit können x und y aus den Variablen A und B der Grammatik erzeugt werden und es ist jederzeit sichergestellt, dass im Gesamtwort $n = |w|_0 + |v|_2$ gilt, denn n entspricht der Summe der 1en in x und y.

(b) Ableitung: $\underline{S} \Rightarrow \underline{A}B \Rightarrow 2\underline{A}B \Rightarrow 20\underline{A}1B \Rightarrow 202\underline{A}1B \Rightarrow 2022\underline{A}1B \Rightarrow 20221\underline{B} \Rightarrow 20221\underline{B}0 \Rightarrow 202211\underline{B}20 \Rightarrow 202211\underline{B}020 \Rightarrow 2022111\underline{B}2020 \Rightarrow 20221112020$

Ableitungsbaum:



(c) $L = \{w1^n v \mid n \in \mathbb{N}, w, v \in \{0,2\}^*, n = |w|_0 \cdot |v|_2\}$

Beweis der Nicht-Kontextfreiheit

Pumpinglemma für kontextfreie Sprachen
Ist L eine kontextfreie Sprache, so gilt:
$\exists p \geq 1 : \forall z \in L, |z| \geq p :$
$\exists u, v, w, x, y \in \Sigma^* : z = uvwxy$ mit

1. $|vx| \geq 1$
2. $|vwx| \leq p$
3. $\forall i \in \mathbb{N} : uv^i wx^i y \in L$

Nehmen wir an L sei kontextfrei, dann können wir das Pumpinglemma anwenden und folgern:

Sei $p \in \mathbb{N}$ die Pumpingzahl. Wir wählen $z = 0^p 1^{p^2} 2^p \in L$ mit $|z| = 2p + p^2 \geq p$.
Aus 1 und 2 folgt: vwx kann nicht aus 0en, 1en und 2en bestehen und vx darf nicht leer sein. Die Formen von v und x entsprechen also einem der vier Fälle (mit $1 \leq k+l+m \leq p$):

- Fall 1.1: $v = 0^k 1^l$ und $x = 1^m$
- Fall 1.2: $v = 0^k$ und $x = 0^l 1^m$
- Fall 2.1: $v = 1^k 2^l$ und $x = 2^m$
- Fall 2.2: $v = 1^k$ und $x = 1^l 2^m$

Fall 1.1:
Man erhält durch pumpen mit $i = 2$ das Wort $z' = uv^2wx^2y = 0^{p+2k}1^{p^2+2(l+m)}2^p$.
Angenommen $k \geq 1$. Damit $z' \in L$ gilt, muss $l + m = k * p$ und somit $k + l + m = k * (p+1) > p$ gelten. ↯
Folglich muss $k = 0$ sein. Aber es muss wieder $l + m = k * p = 0$ gelten und somit $k + l + m = 0$. ↯

Fall 1.2:
Man erhält durch pumpen mit $i = 2$ das Wort $z' = uv^2wx^2y = 0^{p+2(k+l)}1^{p^2+2m}2^p$.
Angenommen $k \geq 1$. Damit $z' \in L$ gilt, muss $m = (k + l) * p$ und somit $k + l + m = l * (p+1) + k * (p+1) > p$ gelten. ↯
Folglich muss $k = 0$ sein. Aber es muss wieder $m = (k + l) * p = l * p$ gelten.
Ist nun $l \geq 1$ gilt $k + l + m = l * (p+1) > p$. ↯
Falls aber $l = 0$, ist auch $k + l + m = 0$. ↯

Fall 2.1:
Man erhält durch pumpen mit $i = 2$ das Wort $z' = uv^2wx^2y = 0^p 1^{p^2+2k}2^{p+2(l+m)}$.
Angenommen $l \geq 1$. Damit $z' \in L$ gilt, muss $k = (l + m) * p$ und somit $k + l + m = l * (p+1) + m * (p+1) > p$ gelten. ↯
Folglich muss $l = 0$ sein. Aber es muss wieder $k = (l + m) * p = m * p$ gelten.
Ist nun $m \geq 1$ gilt $k + l + m = m * (p+1) > p$. ↯
Falls aber $m = 0$, ist auch $k + l + m = 0$. ↯

Fall 2.2:
Man erhält durch pumpen mit $i = 2$ das Wort $z' = uv^2wx^2y = 0^p1^{p^2+2(k+l)}2^{p+2m}$.
Angenommen $m \geq 1$. Damit $z' \in L$ gilt, muss $k + l = m * p$ und somit $l + m + k = m * (p + 1) > p$ gelten. ↯
Folglich muss $m = 0$ sein. Aber es muss wieder $k + l = m * p = 0$ gelten und somit $l + m + k = 0$. ↯

In jedem Fall ergibt sich ein Widerspruch zur Annahme L sei kontextfrei. $\Rightarrow L$ ist nicht kontextfrei.

2.2.3 Aufgabe 3: Entscheidbarkeit

(a) Formulierung des Entscheidungsproblems als Wortproblem von
$L = \{c(M) |\ M\ ist\ TM \wedge (\forall n \leq 42 : \exists w \in \Sigma^* : |w| = n \wedge w \in L(M))\}$

Hier soll nicht die Entscheidbarkeit widerlegt, sondern die Semientscheidbarkeit bewiesen werden. Daher wird nicht *von einem Problem reduziert*, sondern stattdessen *ein Entscheidungsverfahren aufgezeigt*.

Dazu wird eine *nichtdeterministische 3-Band* TM M' skizziert, die alle $w \in L$ akzeptiert:

1. Syntaxcheck, ob $w = c(M)$ mit TM M, auf Band 1
2. Simulieren von M mit Eingabe ε auf Band 2: Falls M hält und akzeptiert, fortfahren mit Schritt 3; ansonsten halten und nicht akzeptieren
3. Nichtdeterministische Auswahl eines Buchstaben $x_i \in \Sigma$ und Schreiben von x_i auf Band 2, Zeiger eins nach rechts setzen
4. n-maliges Wiederholen von Schritt 2, beginnen mit $n = 1$ auf Band 3
5. Simulieren von M mit Eingabe $x_1x_2...x_n$ auf Band 2 (Zeiger dafür zuerst auf x_1 setzen): Falls M hält und akzeptiert, fortfahren mit Schritt 6; ansonsten halten und nicht akzeptieren
6. n um eins erhöhen auf Band 3 und solange $n \leq 42$ zu Schritt 4 zurückspringen: Falls $n > 42$ halten und akzeptieren

Anmerkungen zu M': Da DTM und NTM, sowie Einband- und Mehrband-Maschinen gleich mächtig sind, spielt es keine Rolle welche Anzahl an Bändern und welchen Typ von Maschine man hier wählt. Die *nichtdeterministische* Funktionsweise von M' stellt sicher, dass die in Schritt 3 stückweise "geratenen" Wörter immer genau diejenigen von Länge n sind, welche in $L(M)$ liegen (Vorstellung vom allwissenden Orakel).

Die TM M' akzeptiert alle $w \in L$, somit ist L semi-entscheidbar.

(b) **Satz von Rice (für formale Sprachen)**
Sei $\mathcal{S}$ eine nicht leere, echte Teilmenge der Menge aller formalen Sprachen.
Dann ist die folgende formale Sprache unentscheidbar:
$L_{\mathcal{S}} = \{c(M) |\ M\ ist\ TM\ und\ L(M)\ liegt\ in\ \mathcal{S}\}$

Umschreiben von L:
$L = \{c(M) |\ M\ ist\ TM \wedge L(M) \in \mathcal{S}$ mit $\mathcal{S} := \{L \subseteq \Sigma^* | \forall n \leq 42 : \exists w \in \Sigma^* : |w| = n \wedge w \in L\}$

Nachweis der Nichttrivialität der Menge $\mathcal{S}$:

$\Sigma^* \in \mathcal{S}$, da in Σ^* Worte mit beliebiger Länge liegen $\Rightarrow \mathcal{S} \neq \varnothing$.

$\{\varepsilon\} \notin \mathcal{S}$, da in $\{\varepsilon\}$ z.B. kein Wort der Länge 3 liegt $\Rightarrow \mathcal{S}$ ist nicht die Menge aller formalen Sprachen.

Somit folgt nach dem Satz von Rice, dass L nicht entscheidbar ist.

2.2.4 Aufgabe 4: NP-Vollständigkeit

(a) Dazu wird das Vorgehen einer nichtdeterministischen Zweiband-TM skizziert, welche das Problem im Polynomieller Zeit lößt:

1. Überprüfe, ob w von der Form $c(V, E, k)$ ist mit (V, E) repräsentiert einen Graphen G und $k \in \mathbb{N}$ (auf Band 1).
2. Wähle nichtdeterministisch einen Knoten v' aus V und füge ihn der Knotenmenge V' hinzu (kodiert auf Band 2).
3. Überprüfe, ob es zu v' ein v'' in V' gibt, sodass $(v', v'') \in E$ liegt (auf Band 2).
   - Falls es kein solches v'' in V' gibt, wiederhole ab Schritt 2.
   - Falls genau ein solches v'' in V' existiert und $|V'| \geq k$ gilt, halte und akzeptiere.
   - Falls mehrere Kandidaten für v'' in V' gefunden werden und $|V'| - 1 \geq k$ gilt, halte und akzeptiere.
   - Falls mehrere Kandidaten für v'' in V' gefunden werden und $|V'| - 1 < k$ gilt, halte und akzeptiere nicht.

Anmerkung: Durch die nichtdeterministische Wahl wird stets das passende v' hinzugefügt, falls ein solches existiert (Vorstellung vom allwissenden Orakel).

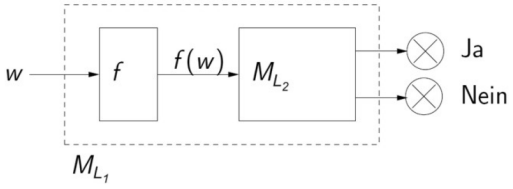
Laufzeitanalyse: Der Syntaxcheck in Schritt 1 läut mit nichtdeterministischem Raten der Übergänge im Eingabewort von V nach E und nach k in $O(|V|)+O(|E|)+O(k) \leq O(n)$. Schritt 2 läuft maximal in $O(|V'|) \leq O(|V|) \leq O(n)$. Die erste Überprüfung in Schritt 3 erfordert höchstens $O(|V'|) \cdot O(|E|) \leq O(n^2)$. Die Bestimmung der Mächtigkeit von V' und der Abgleich mit k erfolgen in $O(|V'|) + O(k) \leq O(n)$.
Die Schritte 2 und 3 werden höchstens $|V|$ mal wiederholt. Insgesamt arbeitet die NTM also in $O(n^3)$.

Daraus folgt, dass das in der Aufgabe beschriebene Graphenproblem in NP liegt.

(b) Sei $L_1 = \{c(V, E, k) \mid G = (V, E)\ ist\ Graph \wedge k \in \mathbb{N} : \exists V' \subseteq V : (\nexists e = (v_1, v_2) \in E\ mit\ v_1, v_2 \in V') \wedge |V'| \geq k\}$
die Sprache, deren Wortproblem gleich dem Entscheidungsproblem einer unabhängigen Menge in G ist.

Sei $L_2 = \{c(V, E, k) \mid G = (V, E)\ ist\ Graph \wedge k \in \mathbb{N} : \exists V' \subseteq V : |\{e = (v_1, v_2) \in E \mid v_1, v_2 \in V'\}| \leq 1 \wedge |V'| \geq k\}$
die Sprache, deren Wortproblem gleich dem Entscheidungsproblem einer fast unabhängigen Menge mit mindestens k Knoten in G ist.

Man verwende folgenden Ansatz für die polynomielle Reduktion:



Dazu wird eine totale und berechenbare Funktion f benötigt, die das Wortproblem aus L_1 auf das Wortprobleme aus L_2 abbildet:

Sei $f : L_1 \to L_2$ definiert über

$$f(w) = \begin{cases} c(\tilde{V}, \tilde{E}, \tilde{k}) & \text{falls } w = c(V, E, k) \text{ mit } G = (V, E) \text{ ist Graph und } k \in \mathbb{N} \\ \varepsilon & \text{sonst} \end{cases}$$

Dabei wird V um zwei Knoten $v_1, v_2 \notin V$ ergänzt und zum neuen $\tilde{V} = V \cup \{v_1, v_2\}$. Die neuen Knoten bilden auch eine Kante, somit ist $\tilde{G} = G \cup \{(v_1, v_2)\}$. Außerdem gilt $\tilde{k} = k + 2$.

f ist offensichtlich total. Außerdem lässt sich eine DTM konstruieren, die f in polynomieller Laufzeit berechnet:

- Syntaxcheck, ob w von der Form $c(V, E, k)$ ist, wobei (V, E) einen Graphen G repräsentiert und $k \in \mathbb{N}$ (höchstens in $O(n^2)$).
- An die Kodierung von V zwei neue Knoten v_1 und v_2 anhängen (höchstens in $O(n^2)$).
- An die Kodierung von G die neue Kante (v_1, v_2) anhängen (höchstens in $O(n)$).
- Zu k in der entsprechenden Kodierung 2 addieren (höchstens in $O(n)$).

Es bleibt noch zu zeigen, dass $w \in L_1 \Leftrightarrow f(w) \in L_2$ gilt. Dies beweisen folgende Äquivalenzumformungen ($\forall w \in L_1$):

$$\begin{aligned}
w \in L_1 &\Leftrightarrow w = c(V, E, k) \; mit \; G = (V, E) \; ist \; Graph \; und \; k \in \mathbb{N} \\
&\quad \wedge (\exists V' \subseteq V : \nexists e = (v_1, v_2) \in E \; mit \; v_1, v_2 \in V' \wedge |V'| \geq k) \\
&\Leftrightarrow f(w) = c(\tilde{V}, \tilde{E}, \tilde{k}) \; mit \; \tilde{G} = (\tilde{V}, \tilde{E}) \; ist \; Graph \; und \; \tilde{k} \in \mathbb{N} \\
&\quad \wedge (\exists V' \subseteq V : \nexists e = (v_1, v_2) \in E \; mit \; v_1, v_2 \in V' \wedge |V'| \geq \tilde{k} - 2) \\
&\Leftrightarrow f(w) = c(\tilde{V}, \tilde{E}, \tilde{k}) \; mit \; \tilde{G} = (\tilde{V}, \tilde{E}) \; ist \; Graph \; und \; \tilde{k} \in \mathbb{N} \\
&\quad \wedge (\exists \tilde{V'} \subseteq \tilde{V} : \exists! e = (v_1, v_2) \in \tilde{E} \; mit \; v_1, v_2 \in \tilde{V'} \wedge |\tilde{V'}| = |V'| + 2 \geq \tilde{k}) \\
&\Leftrightarrow f(w) = c(\tilde{V}, \tilde{E}, \tilde{k}) \; mit \; \tilde{G} = (\tilde{V}, \tilde{E}) \; ist \; Graph \; und \; \tilde{k} \in \mathbb{N} \\
&\quad \wedge (\exists \tilde{V'} \subseteq \tilde{V} : |\{e = (v_1, v_2) \in \tilde{E} \mid v_1, v_2 \in \tilde{V'}\}| \leq 1 \wedge |\tilde{V'}| \geq \tilde{k}) \\
&\Leftrightarrow f(w) \in L_2
\end{aligned}$$

Informelle Beschreibung: Man füge zwei neue Knoten mit einer gemeinsamen Kante hinzu und erhöhe k um 2. Somit kann für jede unabhängige Menge in L_1 mit Mächtigkeit k eine fast unabhängige Menge in L_2 mit Mächtigkeit $k+2$ gefunden werden, die zusätzlich genau die neu hinzugefügten Knoten enthält. Falls eine fast unabhängige Menge in L_2 nur einen der extra hinzugefügten Knoten enthält, muss eine Kante aus E enthalten sein. Allerdings kann dann einer der Knoten dieser Kante weggelassen werden und es gibt immer noch eine unabhängige Menge in L_1 mit zwei Knoten weniger (einer der hinzugefügten und einer aus der Kante). Die Rückrichtung stimmt also auch.

Somit gilt $L_1 \leq_p L_2$ und da L_1 NP-hart ist, muss L_2 auch NP-hart sein, denn die Relation $\leq_p$ ist transitiv. Da L_2 zudem in NP liegt (siehe (a)), ist L_2 und damit das entsprechende Graphenproblem NP-vollständig.

2.2.5 Aufgabe 5: Aussagen

(a) Falsch.
Unabhängig von Σ kann $L = \{\varepsilon\}$ gewählt werden. Diese Sprache wird offensichtlich durch eine DTM entschieden, die auf der leeren Eingabe hält und akzeptiert und bei nicht leerer Eingabe hält und nicht akzeptiert.

(b) Falsch.
Betrachte die Sprache $L = \{a^n | n \in \mathbb{P} \cup \{0,1\}\}$, welche wegen der Primzahleigenschaften bekanntermaßen nicht regulär ist. Allerdings ist $L^* = \{a^n | n \in \{\sum_{i=0}^{\infty} a_i | a_i \in \mathbb{P} \cup \{0,1\}\}\} = \{a^n | n \in \mathbb{N}_0\} = \Sigma^*$, da alle $n \in \mathbb{N}$ als Summe von 1en dargestellt werden können. $L^* = \Sigma^*$ ist allerdings trivialer Weise regulär.

(c) Richtig.
Beweis durch Widerspruch: Sei L unentscheidbar. Wir nehmen an, dass $\overline{L} \in \mathcal{NP}$. Demnach gibt es eine NTM M, die $\overline{L}$ in polynomieller Zeit akzeptiert. Darüber kann eine NTM M' konstruiert werden, welche M simuliert und lediglich akzeptieren und nicht-akzeptieren vertauscht. Da M in polynomieller Zeit arbeitet, tut dies auch M' und entscheidet somit L. ↯
$\Rightarrow$ Wenn L unentscheidbar ist, muss $\overline{L} \notin \mathcal{NP}$ gelten.

(d) Richtig.
Da die DTM M jedes Eingabewort in konstanter Zeit entscheidet, kann ein DEA konstruiert werden, der M simuliert und so L akzeptiert. Dafür sind höchstens $k \cdot 3 \cdot (|\Sigma| + 1) \cdot |\Gamma|$ - also endlich viele - Zustände nötig (maximal k Berechnungsschritte mit 3 möglichen Zeigerbewegungen, $|\Sigma| + 1$ möglichen Eingabesymbolen bzw. einem leeren Übergang und $|\Gamma|$ möglichen Werten auf dem Band).
Deshalb muss $L = L(M)$ regulär sein.