Karlsruher Institut für Technologie
Institut für Theoretische Informatik

Prof. Dr. Peter Sanders
L. Hübschle-Schneider, T. Maier

Weihnachtsblatt zu Theoretische Grundlagen der Informatik im WS 2015/16

http://algo2.iti.kit.edu/TGI2015.php
{sanders,huebschle,t.maier}@kit.edu

Musterlösungen

Aufgabe 1 (*Reguläre Sprachen,* $2+2+3+3$ *Punkte*)

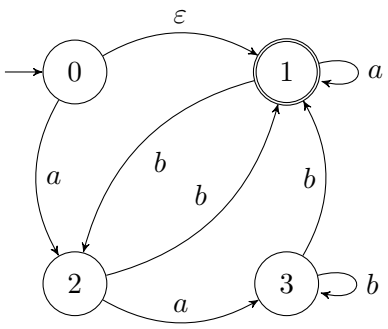
Konstruieren Sie einen deterministischen Automaten, einen regulären Ausdruck, und eine rechtsreguläre Grammatik für jede der folgenden regulären Sprachen.

a) $L_a = \{w \in \{a,b\}^* \mid \exists_{X \in \{a,b\}} \ w \text{ endet auf } aXa\}$

b) $L_b = \{w \in \{a,b\}^* \mid n_b \mod 3 = 1 \ \vee \ n_a \mod 2 = 1\}$

Wenden Sie die Potenzmengenkonstruktion an, um aus dem folgenden NEA den zugehörigen DEA zu konstruieren. Verwenden Sie dann den aus der Vorlesung bekannten Algorithmus zur Automatenminimierung, um die entstandenen Automaten zu minimieren. Füllen sie dabei die untenstehenden Tabellen aus und zeichnen sie die entstandenen Automaten.

c)



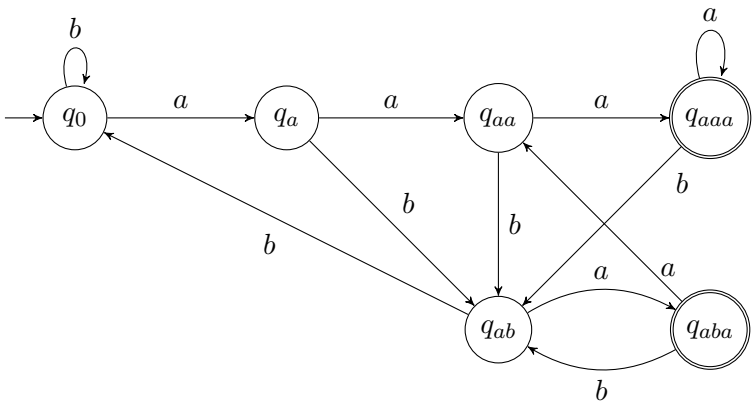
Zustände	a	b
$A =$		
$B =$		
$C =$		
$D =$		
$E =$		
$F =$		
$G =$		
$H =$		

Zustände	A	B	C	D	E	F	G	H
A	$=$							
B		$=$						
C			$=$					
D				$=$				
E					$=$			
F						$=$		
G							$=$	
H								$=$

d) Zeigen sie mit Hilfe des Pumpinglemmas, dass die Sprache $L = \{a^n b^n c^m \mid n, m \in \mathbb{N}\} \cup \{a^n b^m c^m \mid n, m \in \mathbb{N}\}$ nicht regulär ist.

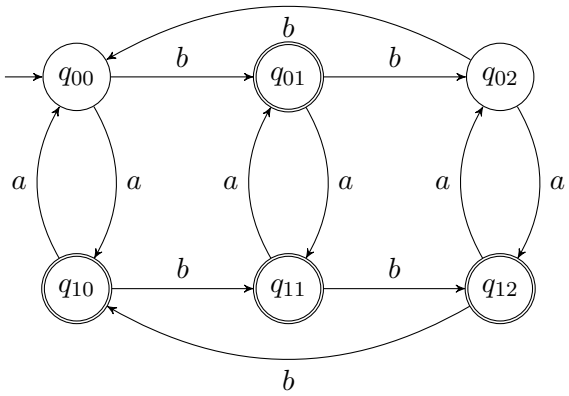
Musterlösung:

a) Automat:



Grammatik: $G_a = (\{a, b\}, \{S, X, Y\}, \{S \to aS \mid bS \mid aX, \quad X \to aY \mid bY, \quad Y \to a\}, S)$
RegEx: `(a∪b)*a(a∪b)a`

b) Automat:



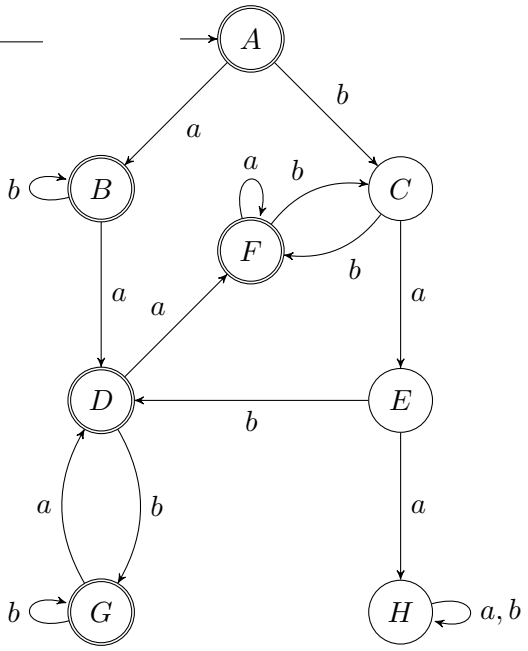
Grammatik: $G_b = (\{a, b\}, \{Q_{00}, Q_{01}, Q_{02}, Q_{10}, Q_{11}, Q_{12}\}, P_b, Q_{00})$

$P_b = \{Q_{00} \to aQ_{10} \mid bQ_{01}, \quad Q_{01} \to aQ_{11} \mid bQ_{02} \mid \varepsilon, \quad Q_{02} \to aQ_{12} \mid bQ_{00},$
$Q_{10} \to aQ_{00} \mid bQ_{11} \mid \varepsilon, \quad Q_{11} \to aQ_{01} \mid bQ_{12} \mid \varepsilon, \quad Q_{12} \to aQ_{02} \mid bQ_{10} \mid \varepsilon\}$

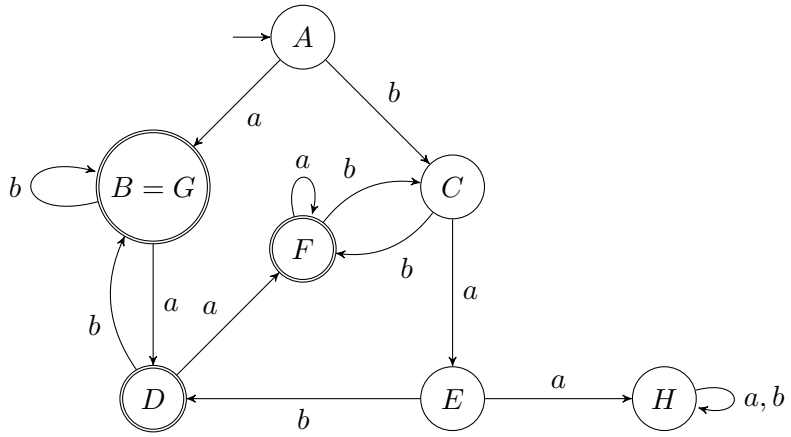
RegEx: `((b*ab*ab*)*b*ab*)∪((a*ba*ba*ba*)*a*ba*)`

c) Automat:

Zustände	a	b
$A = \{0, 1\}$	$\{1, 2\}$	$\{2\}$
$B = \{1, 2\}$	$\{1, 3\}$	$\{1, 2\}$
$C = \{2\}$	$\{3\}$	$\{1\}$
$D = \{1, 3\}$	$\{1\}$	$\{1, 2, 3\}$
$E = \{3\}$	$\emptyset$	$\{1, 3\}$
$F = \{1\}$	$\{1\}$	$\{2\}$
$G = \{1, 2, 3\}$	$\{1, 3\}$	$\{1, 2, 3\}$
$H = \emptyset$	$\emptyset$	$\emptyset$



Zustände	$\{0,1\}$	$\{1,2\}$	$\{2\}$	$\{1,3\}$	$\{3\}$	$\{1\}$	$\{1,2,3\}$	$\emptyset$
$\{0,1\}$	$=$							
$\{1,2\}$	b	$=$						
$\{2\}$	ε	ε	$=$					
$\{1,3\}$	b	ab	ε	$=$				
$\{3\}$	ε	ε	ab	ε	$=$			
$\{1\}$	ab	b	ε	b	ε	$=$		
$\{1,2,3\}$	b	$-$	ε	ab	ε	b	$=$	
$\emptyset$	ε	ε	b	ε	b	ε	ε	$=$



d) z.Z.: Bereits negiertes Pumpinglemma:
$\forall_{n\in\mathbb{N}}\ \ \exists_{w\in L}$ mit $|w| \geq n\ \ \forall_{uvx=w}$ mit $|uv| \leq n\ \wedge\ |v| > 0\ \ \exists_{i\in\mathbb{N}_0}$ mit $uv^ix \notin L$

Sei n die Zahl aus dem Pumpinglemma. Wir wählen das Wort a^nb^nc für jede Aufteilung mit obigen Einschränkungen gilt: $v = a^k$ $(k \in \mathbb{N})$ daraus folgt $uv^2x = a^{n+k}b^nc$ liegt nicht in L.

Aufgabe 2 (*Kontextfreie Sprachen,* $2 + 4 + 2$ *Punkte*)

a) Bringen Sie die folgende kontextfreie Grammatik in Chomskynormalform.

$$
\begin{aligned}
G_a &= (\{a, b\}, \{S, C, D, E\}, P_a, S) \\
P_a &= \{S \to C \mid DE \mid abD, \\
&\qquad C \to aCb \mid ba, \\
&\qquad D \to aD \mid bDa \mid \varepsilon, \\
&\qquad E \to S \mid abE\}
\end{aligned}
$$

b) Gegeben sei die Grammatik $G_b = (\{a, b, c\}, \{A, B, C, S\}, P_b, S)$ mit

$$
\begin{aligned}
P_b = \{&S \to AB \mid CS \mid BS \mid AA, \\
&A \to a \mid SS \mid BB \mid BC, \\
&B \to b \mid CC \mid BS, \\
&C \to c\}
\end{aligned}
$$

Prüfen Sie mit Hilfe des CYK-Algorithmus (von Cocke, Younger, Kazami) ob die Wörter *abcabb* und *acbabcb* in der von G_b erzeugten Sprache liegen. Füllen Sie hierzu die folgenden Tabellen aus.

a	b	c	a	b	b

a	c	b	a	b	c	b

c) Konstruieren Sie einen Kellerautomaten, der die Sprache $L = \{a^n b^n c^m \mid m, n \in \mathbb{N}\} \cup \{a^n b^m c^m \mid m, n \in \mathbb{N}\}$ erkennt. Ist der von Ihnen konstruierte Automat deterministisch? Kann es einen deterministischen Kellerautomat geben, der diese Sprache erkennt? Begründen Sie kurz (kein Beweis).

d) Verwenden Sie das Pumpinglemma für kontextfreie Sprachen um zu zeigen, dass die Sprache $L = \{a^n w a^n w^R \mid w \in \{b, c\}^*, n \in \mathbb{N}\}$ nicht kontextfrei ist.

Musterlösung:

a) 1.) Elimination von ε-Übergängen:

$$
\begin{aligned}
P'_a = \{&S \to C \mid DE \mid E \mid abD \mid ab, \\
&C \to aCb \mid ba, \\
&D \to aD \mid a \mid bDa \mid ba, \\
&E \to S \mid abE\}
\end{aligned}
$$

2.) Elimination zyklischer Einheitsabbildungen ($S \leftrightarrow E$):

$$
\begin{aligned}
P''_a = \{&S \to C \mid DS \mid abD \mid ab \mid abS, \\
&C \to aCb \mid ba, \\
&D \to aD \mid a \mid bDa \mid ba\}
\end{aligned}
$$

3.) Elimination weiterer Einheitsabbildungen ($S \to C$):

$$\begin{aligned} P_a''' = \{ & S \to DS \mid abD \mid ab \mid abS \mid aCb \mid ba, \\ & C \to aCb \mid ba, \\ & D \to aD \mid a \mid bDa \mid ba\} \end{aligned}$$

4.) Entfernen zu langer und gemischter rechter Seiten:

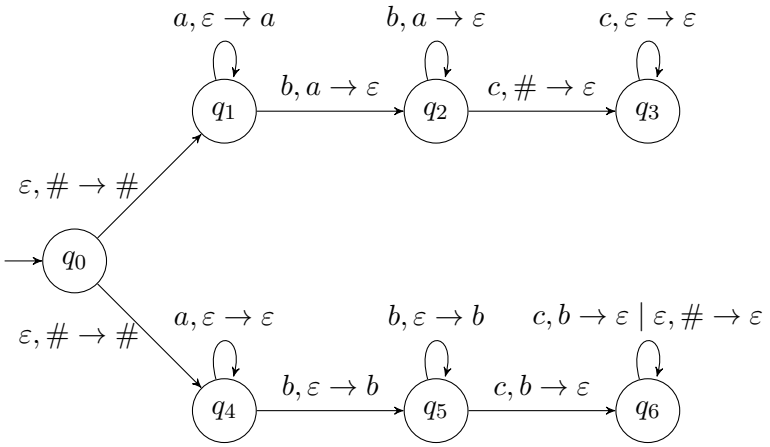
$$\begin{aligned} P_a'''' = \{ & S \to DS \mid A_bD \mid AB \mid A_bS \mid A_CB \mid BA, \\ & C \to A_CB \mid BA, \\ & D \to AD \mid a \mid B_DA \mid BA, \\ & A_B \to AB, \\ & A_C \to AC, \\ & B_D \to BD, \\ & A \to a \quad B \to b\} \end{aligned}$$

b)

a	b	c	a	b	b
A	B	C	A	B	B
S	A	$-$	S	A	
S	S	S	S		
$-$	S, B	S			
S, A	S,A,B				
S, A					

a	c	b	a	b	c	b
A	C	B	A	B	C	B
$-$	$-$	$-$	S	A	$-$	
$-$	$-$	S, B	S	S		
$-$	S	S,A,B	$-$			
$-$	S	S, A				
$-$	S					
$-$						

c)



Der Automat akzeptiert durch einen leeren Stack nachdem das vollständige Wort eingelesen wurde. Er ist nichtdeterministisch, da zu Beginn der Berechnung nichtdeterministisch entschieden wird, zu welcher Teilmenge das Wort gehört.

Es kann keinen deterministischen Kellerautomaten für diese Sprache geben. Zunächst muss die Anzahl as gezählt werden und mit der Anzahl bs verglichen werden, um zu entscheiden ob die Eingabe zur ersten Teilsprache gehört ($a^n b^n c^m$). Nach dieser Überprüfung ist keine Information mehr über die tatsächliche Anzahl von as und bs vorhanden. Die Anzahl der bs kann dementsprechend nicht mehr mit der Anzahl der cs verglichen werden (was nötig wäre, um zu entscheiden ob die Eingabe in der 2. Teilsprache liegt $a^n b^m c^m$).

d) z.Z. Negiertes Pumpinglemma für kontextfreie Sprachen:
$\forall_{n \in \mathbb{N}} \ \exists_{w \in L}$ mit $|w| \geq n \ \ \forall_{uvxyz=w}$ mit $|vxy| \leq n \ \wedge \ |vy| > 0 \ \ \exists_{i \in \mathbb{N}_0}$ mit $uv^i xy^i z \notin L$

Sei n die Zahl aus dem Pumpinglemma. Wir wählen das Wort $a^n b^n a^n b^n$. Wir betrachten die folgenden zwei Fälle (zusammen decken sie jede mögliche Zerlegung ab):

- vy enthält zumindest ein a
  $\Rightarrow$ alle as in vy kommen aus dem selben a Block (dazwischen liegen n Zeichen)
  $\Rightarrow$ in $uv^0xy^0z = uxz$ sind die zwei a-Blöcke unterschiedlich lang
  $\Rightarrow uxz \notin L$

- vy enthält kein a
  $\Rightarrow vy$ enthält nur bs $\Rightarrow vy$ kann nicht bs aus beiden b-Blöcken enthalten (n Zeichen dazwischen)
  $\Rightarrow$ in $uv^0xy^0z = uxz$ sind die zwei b Blöcke unterschiedlich lang
  $\Rightarrow uxz \notin L$

Hiermit haben wir gezeigt, dass L nicht kontextfrei sein kann, da das Pumping Lemma nicht gilt. Egal wie das Wort $a^nb^na^nb^n$ aufgeteilt wird, es kann nicht korrekt gepumpt werden.

Aufgabe 3 (*Chomsky 1/0 ,* $3+3$ *Punkte*)

a) Warum ist $L_1 = \{n\#w \mid n_a(w) = n^2, \text{ mit } n \text{ binär kodiert}\}$ vom Chomsky-Typ 1? Beweisen Sie, dass L_1 nicht kontextfrei ist und begründen Sie, warum es kontextsensitiv ist.

b) Zeigen Sie: Die Sprache aller Turingmaschinen, die Weihnachtsbäumchen malen und dann halten, ist vom Chomsky-Typ 0 und *nicht* vom Chomsky-Typ 1.

Musterlösung:

a) L_1 ist kontextsensitiv, da die Sprache äquivalent zu einer linear beschränkten Turingmaschine und nicht kontextfrei ist. Ersteres ist mit dem Pumpinglemma für kontextfreie Sprachen leicht zu sehen. Für Pumplänge n wähle das Wort $z = 10^n\#a^{2^{n+1}}$. Da die Binärzahl 10^n gerade 2^n entspricht und $(a^{2^n})^2 = a^{2^{n+1}}$, liegt z offensichtlich in L_1 und ist auch ausreichend lang. Wir betrachten jetzt jede konforme Zerlegung $w = uvwxy$ mit $|vwx| < n$ und $|vx| \geq 1$.

   - Wenn $\# \notin vwx$, können wir trivial abpumpen – entweder wird die Zahl dadurch zu klein oder die Anzahl as zu gering, das abgepumpte Wort $z' = uwy$ ist aber definitiv nicht in L_1.
   - Wenn das Trennzeichen $\#$ in v oder x liegt, ist das Wort $z' = uwy \notin L_1$, da das Trennzeichen fehlt.
   - Wenn das Trennzeichen $\#$ in w liegt, unterscheiden wir zwei Fälle. Wenn v oder x leer ist, ist das abgepumpte Wort offensichtlich nicht in L_1, da entweder die Zahl durch das abpumpen zu klein wurde oder das Wort nun mehr as enthält als erlaubt. Andernfalls, das heißt wenn v Zeichen der Zahl und x Zeichen des Wortes enthält, können wir aufpumpen. Die Zahl wächst dadurch exponentiell, das Wort aber nur linear, daher kann das Wort $z' = uv^nwx^ny$ nicht in L_1 liegen (die Zahl ist nun mindestens 2^{2n}, das Wort aber höchstens $2^{n+1} + n^2 \ll (2^{2n})^2$ as lang).

   Die Äquivalenz zu einer linear beschränkten Turingmaschine ist relativ leicht zu sehen, auch ohne die Turingmaschine explizit zu konstruieren. Dazu müssen wir zuerst die Zahl n quadrieren (arithmetische Operationen sind mit Turingmaschinen einfach). Um die lineare Platzbeschränkung einzuhalten, müssen wir dazu das Bandalphabet erweitern und können Beispielsweise zur Basis 4 rechnen. Dann dekrementieren wir n in jedem Schritt und suchen ein passendes a in w, das wir durch ein anderes Zeichen ersetzen (beispielsweise ein b). Wir akzeptieren, falls am Ende $n = 0$ ist und keine weiteren as in w sind.

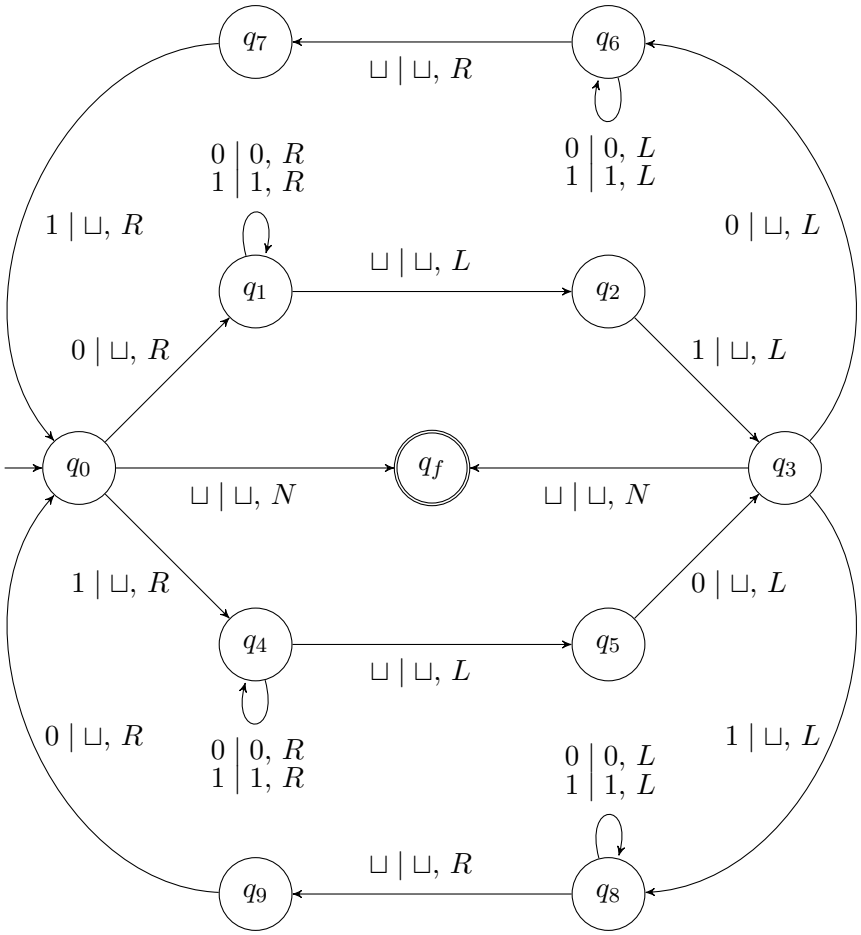
b) Die Aufgabenstellung lässt sich so umformulieren, dass man zeigen soll, dass die Sprache aller Turingmaschinen, die Weihnachtsbäumchen malen, semientscheidbar (aber nicht entscheidbar) ist. Es ist leicht zu sehen, dass das der Fall ist: wir können die Maschine simulieren und sobald sie hält überprüfen, dass sie ein Weihnachtsbäumchen gemalt hat. Andersrum ist die Erkennung von Weihnachtsbäumchen quasi identisch zur Erkennung des Vortes `VIRUS` im Beispiel aus Übung 7, weshalb die Menge unentscheidbar ist.

Aufgabe 4 (*Turingmaschinen,* $4 + (1 + 2 + 1)$ *Punkte*)

a) Konstruieren Sie eine linear beschränkte Turingmaschine $\mathcal{M} = (\mathcal{Q}, \Sigma, \Sigma \cup \{\sqcup\}, \delta, q_0, \mathcal{F})$ mit Eingabealphabet $\Sigma = \{0, 1, a, b, \#\}$, Startzustand $q_0 \in \mathcal{Q}$ und $|\mathcal{Q}| \leq 8$, welche die Sprache $L = \{n\#w \mid w \in \{a, b\}^* \wedge n_a(w) = n, \text{ mit } n \text{ binär kodiert}\}$ erkennt (Beachten Sie, dass sich die Sprache von der aus Aufgabe 3 unterscheidet: n wird nicht quadriert). Dabei darf die Binärzahl wie auf Blatt 7 führende Nullen enthalten. Ihre Maschine darf die Eingabe völlig zerfleddern. Beschreiben Sie die Funktionsweise Ihrer Turingmaschine und umreißen Sie die Aufgaben der Zustände. Sie dürfen ausnahmsweise annehmen, dass die Eingabe $n\#w$ die Bedingung $w \in \{a, b\}^*$ erfüllt.

Hinweis: Können Sie Ihre Subtraktionsmaschine vom 7. Übungsblatt teilweise wiederverwenden? Schaffen Sie es, mit 7 Zuständen auszukommen?

b) Gegeben sei die Turingmaschine $\mathcal{M} = (\{q_0, \ldots, q_9, q_f\}, \{0, 1\}, \{0, 1, \sqcup\}, \delta, q_0, \{q_f\})$, deren Zustandsübergangsfunktion δ durch den untenstehenden Graphen gegeben ist.



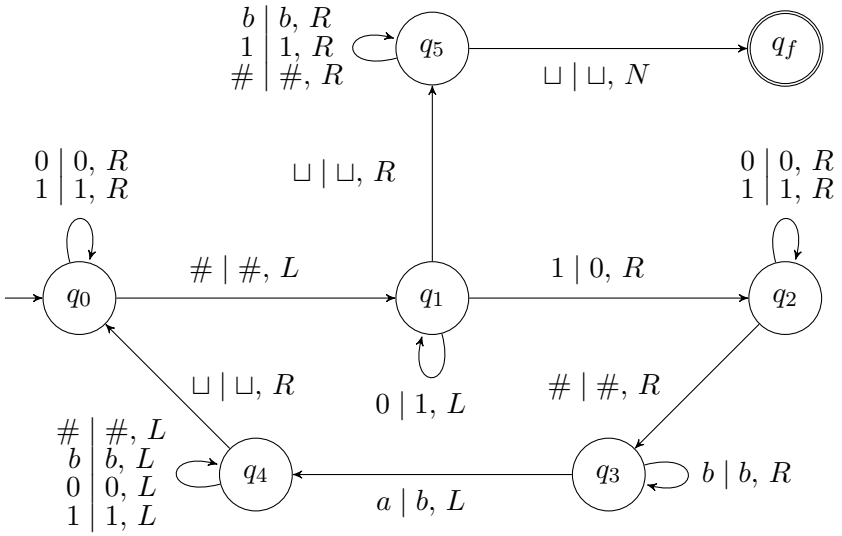
1.) Simulieren Sie die Berechnung dieser Turingmaschine auf der Eingabe $w = 01101001$. Geben Sie *alle* durchlaufenen Konfigurationen an.

2.) Welche Sprache erkennt diese Turingmaschine?

3.) Geben Sie eine Grammatik maximalen Chomsky-Typs an, welche diese Sprache erzeugt. Welchen (höchsten) Chomsky-Typ hat Ihre Grammatik?

Musterlösung:

a) $\mathcal{Q} = \{q_0, \ldots, q_5, q_f\}$, der Zustandsübergangsgraph ist durch untenstehende Zeichnung gegeben.



Die Maschine subtrahiert zuerst eins von der Binärzahl n (Zustände q_0, q_1 und q_2, ähnlich zu Aufgabe 1a auf Blatt 7) und sucht dann das erste a im Wort w (Zustand q_3), welches sie durch ein b substitutiert. Dann geht sie nach links durch (Zustand q_4) und fängt wieder von vorne an. Sollte die Zahl Null sein, geht sie das restliche Wort durch und prüft, ob noch irgendwo ein a kommt (Zustand q_5). Sollte das nicht der Fall sein, hält sie nichtakzeptierend durch fehlenden Übergang, ansonsten akzeptierend in q_f. Wenn $n > n_a(w)$ hält die Maschine nichtakzeptierend durch fehlenden Übergang aus q_3 mit $\sqcup$. Falsch formatierte Eingaben werden ebenfalls durch fehlende Zustandsübergänge abgelehnt.

Sollte die Bedingung $w \in \{a, b\}^*$ auch noch geprüft werden, würde dies zwei zusätzliche Zustände zwischen q_2 und q_3 erfordern, von denen der erste ganz nach rechts durchläuft und überprüft, dass nur as und bs folgen, und der zweite wieder bis zum # nach links läuft und dann mit Kopfbewegung nach rechts in q_3 übergeht.

b) Turingmaschinenerkennung

1.) Die Turingmaschine durchläuft folgende Konfigurationen:

$(q_0)01101001$	$1101001(q_1)\sqcup$	$(q_6)\sqcup 11010$	$0(q_3)1$
$(q_1)1101001$	$110100(q_2)1$	$(q_7)11010$	$(q_8)0$
$1(q_1)101001$	$11010(q_3)0$	$(q_0)1010$	$(q_8)\sqcup 0$
$11(q_1)01001$	$1101(q_6)0$	$(q_4)010$	$(q_9)0$
$110(q_1)1001$	$110(q_6)10$	$0(q_4)10$	$(q_0)\sqcup$
$1101(q_1)001$	$11(q_6)010$	$01(q_4)0$	$(q_f)\sqcup .$
$11010(q_1)01$	$1(q_6)1010$	$010(q_4)\sqcup$	
$110100(q_1)1$	$(q_6)11010$	$01(q_5)0$	

2.) $L = \{w\overline{w} \mid w \in \{0,1\}^*, \overline{w}$ ist w in umgekehrter Reihenfolge mit 0 und 1 vertauscht$\}$

3.) $G = (\{0,1\}, \{S\}, S, P)$ mit $P = \{S \to \varepsilon \mid 0S1 \mid 1S0\}$ ist kontextfrei.

Aufgabe 5 (*Entscheidbarkeit,* $(1+2+2+1+1)+1+2+(2+2)$ *Punkte*)

a) Sind die folgenden Mengen entscheidbar, semientscheidbar oder unentscheidbar? Beweisen Sie!

1.) $L_1 = \{1 \mid \text{Der Weihnachtsmann existiert}\}$
2.) $L_2 = \{\langle M\rangle \mid \text{Für alle Worte } w \in \Sigma^* : M \text{ akzeptiert } w \Leftrightarrow M \text{ akzeptiert } w^R\}$
3.) $L_3 = \{\langle M\rangle \mid M(x) = x^2\}$
4.) $L_4 = \{\langle M\rangle \mid M \text{ ist eine Turingmaschine und } L(M) \text{ ist semi-entscheidbar}\}$
5.) $L_5 = \{\langle M\rangle w \mid M \text{ ist eine Turingmaschine und } M \text{ hält auf der Eingabe } w\}$

b) Zeigen Sie, dass das Komplement $\mathcal{H}^c$ des Halteproblems nicht semientscheidbar ist.

c) Zeigen Sie, dass das Komplement L_d^c der Diagonalsprache semientscheidbar ist.

d) Analog zu Turingmaschinen lassen sich auch andere Maschinenmodelle gödelisieren, beispielsweise deterministische endliche Automaten. Wir nehmen wieder an, dass ungültige Automatengödelnummern den Automaten beschreiben, der die leere Sprache akzeptiert. Sei Σ ein endliches Alphabet. Zeigen Sie, dass die folgenden Sprachen entscheidbar sind.

1.) $\text{INFINITE}_{\text{DFA}} = \{\langle \mathcal{M}\rangle \mid \mathcal{M} \text{ ist ein endlicher Automat und } L(\mathcal{M}) \text{ ist unendlich}\}$
2.) $\text{ALL}_{\text{DFA}} = \{\langle \mathcal{M}\rangle \mid \mathcal{M} \text{ ist ein endlicher Automat und } L(\mathcal{M}) = \Sigma^*\}$

Musterlösung:

a) 1.) Entscheidbar. Es gibt zwei Möglichkeiten: Entweder der Weihnachtsmann existiert und $L_1 = \{1\}$, oder der Weihnachtsmann existiert nicht und $L_1 = \emptyset$. Beide Mengen sind offensichtlich entscheidbar. Dass wir nicht wissen, welche jetzt die "richtige" ist, ändert an der Entscheidbarkeit von L_1 nichts.

2.) Unentscheidbar, denn damit ließe sich das Halteproblem lösen: Sei $\langle M\rangle w$ eine Instanz des Halteproblems. Mithilfe dieser Instanz konstruieren wir eine neue Turingmaschine M', die bei Eingabe v folgendes Verhalten aufweist:

- Falls $v = 01$, akzeptiere
- Ansonsten lösche die Eingabe vom Band, simuliere M bei Eingabe w und akzeptiere dann.

Wir beobachten: M' akzeptiert 01 immer. Außerdem akzeptiert M' ganz Σ^*, falls M auf der Eingabe w hält. In diesem Fall gilt also für alle $w \in \Sigma^*$: $w \in L(M') \Leftrightarrow w^R \in L(M')$. Falls M bei Eingabe w nicht hält, erkennt M' nur die Sprache $\{01\}$, und in diesem Fall ist $(01)^R = 10 \notin L(M')$. Es gibt also ein $w \in \Sigma^*$ sodass $w \in L(M')$ und $w^R \notin L(M')$. Eine Turingmaschine, die L_2 erkennt, muss also das Halteproblem lösen, um zu entscheiden, ob $\langle M'\rangle$ in der Sprache liegt. Daher kann L_2 nicht entscheidbar sein.

3.) Unentscheidbar. Mit dem Satz von Rice können wir das leicht erkennen und mit Beweis über den Rekursionssatz zeigen.

Sei M' eine Turingmaschine, die L entscheidet, also gegeben einer Turingmaschine M entscheidet, ob diese für all x den Wert x^2 ausgibt:

$$M'(\langle M\rangle) = \begin{cases} \text{👍} & \text{wenn } \forall_x M(x) = x^2, \\ \text{👎} & \text{wenn } \exists_x M(x) \neq x^2. \end{cases}$$

Wir konstruieren nun eine Turingmaschine D wie folgt: D holt sich seine eigene Beschreibung $\langle D\rangle$ (Rekursionssatz) und führt M' darauf aus. Akzeptiert M', lehnt D ab, und andersrum:

$$D(x) = \begin{cases} x^2 & \text{wenn } M'(\langle D\rangle) = \text{👎}, \\ \bot & \text{wenn } M'(\langle D\rangle) = \text{👍}. \end{cases}$$

Das ist ein Widerspruch.

4.) Entscheidbar. Per Definition ist eine Sprache genau dann semientscheidbar, wenn eine Turingmaschine existiert, die sie akzeptiert. Daher ist jede Sprache, die von einer Turingmaschine akzeptiert wird, semientscheidbar. Insgesamt ist L also entscheidbar.

5.) Semientscheidbar. Wir geben eine Turingmaschine M' an, die genau dann akzeptiert, wenn die gegebene Maschine M auf w hält. Dazu simuliert M' die Maschine M auf w und akzeptiert dann.

b) Wir wissen bereits, dass das Halteproblem $\mathcal{H}$ nicht entscheidbar ist, und außerdem, dass L und L^c genau dann semientscheidbar sind, wenn L entscheidbar ist. Außerdem wissen wir aus Teil 4 des vorherigen Aufgabenteils, dass das Halteproblem semientscheidbar ist. Daher kann $\mathcal{H}^c$ nicht semientscheidbar sein.

c) Die Definition der Diagonalsprache war $L_d = \{w_i \mid M_i \text{ akzeptiert } w_i \text{ nicht}\}$. Wir wissen aus der Vorlesung bereits, dass weder L_d noch L_d^c entscheidbar sind. Wir können aber eine neue Turingmaschine M' konstruieren, die für Eingabe $(\langle M_i \rangle v)$ die Turingmaschine M_i auf v ausführt:

- Falls M_i nach endlich vielen Schritten akzeptiert, akzeptiere
- Falls M_i nicht akzeptiert, lehne ab.
- Ob die Simulation stoppt oder nicht ist uns egal, da es nur um Semientscheidbarkeit geht.

M' semientscheidet das Komplement L_d^c der Diagonalsprache.

d) Wir konstruieren Entscheider $\mathcal{T}$, die als Eingabe $\langle \mathcal{M} \rangle$ erhalten, wobei $\mathcal{M} = (\mathcal{Q}, \Sigma, \delta, q_0, \mathcal{F})$.

1.) Ausgehend von allen Endzuständen $q \in \mathcal{F}$ durchlaufen wir den Rückwärtsgraphen des Zustandsübergangsgraphen (dort sind alle Kanten genau umgekehrt) in einer Breitensuche und geben allen erreichten Zuständen eine Markierung f. Dies sind die Zustände, von denen aus ein Endzustand erreicht werden kann.

Ausgehend vom Startzustand q_0 durchlaufen wir nun den (ursprünglichen) Zustandsübergangsgraphen mit einer Tiefensuche und markieren alle erreichten Zustände mit einer anderen Markierung s. Sobald die Tiefensuche auf einen bereits mit s markierten Zustand q trifft, überprüfen wir, ob q auch eine f-Markierung trägt. Falls ja, akzeptieren wir den Automaten, denn er enthält einen Kreis, der von q_0 aus erreichbar ist, beliebig oft durchlaufen werden kann und zu einem Endzustand führt. Andernfalls gehen wir die Kante, über die q erreicht wurde, wieder zurück, entfernen die Markierung von q aber nicht, und fahren normal mit der Tiefensuche fort. Beim Rückschreiten (Backtracking) in der Tiefensuche entfernen wir s-Markierungen. Sollte die Tiefensuche enden, lehnen wir die Eingabe ab, da kein nutzbarer Kreis gefunden wurde.

Vergleiche den gefundenen Kreis mit dem Aufpumpen des Mittelteils im Pumpinglemma. Die Maschine entscheidet die Sprache, da sowohl Breiten- als auch Tiefensuche nach endlich vielen Schritten alle zyklenfreien Pfade durchlaufen haben und terminieren.

2.) Wir konstruieren einen äquivalenten Entscheider, der überprüft ob $L(\mathcal{M})^c = \emptyset$. Dazu vertauschen wir zuerst die Endzustände von $\mathcal{M}$ mit den Nichtendzuständen, d.h. $\mathcal{F}' = \mathcal{Q} \setminus \mathcal{F}$. Falls $\mathcal{F}' = \emptyset$ können wir sofort akzeptieren. Andernfalls durchlaufen wir den Zustandsübergangsgraphen von $\mathcal{M}' = (\mathcal{Q}, \Sigma, \delta, q_0, \mathcal{F}')$ mit einer Breitensuche beginneng bei q_0 und markieren alle erreichten Zustände. Falls dabei ein Endzustand $q \in \mathcal{F}'$ erreicht wird, lehnen wir die Eingabe ab. Falls die Suche terminiert, ohne dass en Endzustand $q \in \mathcal{F}'$ gefunden wurde, akzeptiere.

$\mathcal{T}$ überprüft also, ob es einen Endzustand gibt, der vom Startzustand aus erreichbar ist. Da $\mathcal{M}'$ genau die Komplementsprache $L(\mathcal{M})^c$ von $L(\mathcal{M})$ erkennt, genügt es also zu entscheiden, ob $\mathcal{M}'$ die leere Sprache erkennt. Dies ist genau dann der Fall, wenn kein Endzustand vom Startzustand aus erreichbar ist. Da die Breitensuche immer nach endlich vielen Schritten terminiert, ist $\mathcal{T}$ ein Entscheider für ALL_{DFA}.

Aufgabe 6 (*Nicht-Entscheidbarkeit,* $3+1+1+1+1+1+1+1$ *Punkte*)

a) Beweisen Sie: Die Sprache $L = \{\langle M\rangle \mid M$ der Kopf von M besucht immer jedes Symbol der Eingabe$\}$ ist nicht entscheidbar.

Geben Sie für jedes der folgenden Post'schen Korrespondenzprobleme entweder eine Lösung an oder einen Beweis, dass keine Lösung existiert.

b) $t_1 = (b, ab), t_2 = (ba, abb), t_3 = (ab, abb)$

c) $t_1 = (ca, bac), t_2 = (b, ab), t_3 = (aba, a)$

d) $t_1 = (bab, bb), t_2 = (ba, abb), t_3 = (ab, ba)$

e) $t_1 = (aaaaa, a), t_2 = (aa, aaaaa)$

f) $t_1 = (abc, ab), t_2 = (abba, aabb), t_3 = (c, ccc), t_4 = (bbba, cbbb), t_5 = (abcc, aab)$

g) $t_1 = (ba, bab), t_2 = (abb, bb), t_3 = (bab, abb)$

h) $t_1 = (b, ca), t_2 = (a, ab), t_3 = (ca, a), t_4 = (abc, c)$

Musterlösung:

a) Reduktion mit Hilfe des Halteproblems:
Gegeben einer Instanz des Halteproblems $IN = \langle M_{IN}\rangle w_{IN}$ konstruieren wir die folgende Turingmaschine T_{IN}

- Schreibe IN links neben die Eingabe (mit einem Trennsymbol)
- Simuliere M_{IN} auf der Eingabe w_{IN} (dabei muss darauf geachtet werden, dass die Simulation den Kopf nicht über die ursprüngliche Eingabe bewegt. Hierzu muss das simulierte Band potentiell verschoben werden (ähnlich der Simulation einer Zweibandturingmaschine auf einer Einbandturingmaschine). Dies führt zu einer quadratischen Laufzeitverschlechterung (kein Problem).
- Nun bewegt sich der Kopf zurück zur Eingabe und besucht jedes Symbol.

Es ist klar, dass eine Turingmaschine die für jedes T_{IN} entscheiden kann, ob es jedes Zeichen der Eingabe besucht, auch das Halteproblem lösen kann.

Alternativ, mit Hilfe des Rekursionssatzes: Sei T_E eine Turingmaschine, die L entscheidet. Wir konstruieren die Turingmaschine T_H die von T_E nicht korrekt kategorisiert wird.

- T_H schreibt sowohl $\langle T_H\rangle$ als auch $\langle T_E\rangle$ auf das Band.
- Nun simuliert T_H die Turingmaschine T_E auf der Eingabe $\langle T_H\rangle$. Hierbei müssen wir erneut sicherstellen, dass die Eingabe nicht besucht wird (s.o.).
  - Falls T_E akzeptiert hält T_H
  - sonst besucht T_H die gesamte Eingabe und hält danach.

Es ist offensichtlich, dass T_H die Eingabe T_E falsch kategorisiert hat. Dementsprechend kann T_E die Sprache L nicht korrekt entscheiden.

b) Es ist keine Lösung möglich, da für jedes $t_i = (x_i, y_i)$ gilt, dass $|x_i| < |y_i|$. Dementsprechend können nie zwei gleich lange Wörter erreicht werden.

c) Die Folge $t_3t_1t_2$ ergibt das Wort *abacab*.

d) Es gibt keine Lösung, da kein Tupel als Anfang geeignet ist (x_i muss Präfix von y_i sein oder andersherum).

e) Eine Folge aus $3 \times t_1, 4 \times t_2$ liefert das Wort a^{23} (die Ordnung ist egal)

f) Die Folge $t_1t_4t_5t_3$ ergibt das Wort *abcbbbaabccc*.

g) Es gibt nur ein mögliches Starttupel, ($t_1 = (ba, bab)$). Dieses erzeugt einen Überschuss von genau einem b im Wort y, weshalb auf das Tupel t_1 nur t_3 folgen kann. Dadurch entsteht erneut ein b Überschuss. Hierdurch gerät man in einen Zyklus (es können nur t_3s angehängt werden, diese verändern jedoch nichts an der Wort-Differenz).

h) Die Folge $t_2t_1t_3t_2t_4$ liefert das Wort *abcaaabc*

Aufgabe 7 (*Wahr-Falsch-Block,* 10 *Punkte*)

Sind die folgenden Aussagen wahr oder falsch? Begründen Sie kurz.

Aussage	Wahr	Falsch	Begründung
$\{p \mid p \text{ ist prim}\}$ ist entscheidbar	×		Wir können alle möglichen Teiler von 2 bis $\sqrt{n}$ durchprobieren und dadurch immer in endlicher Zeit zu einem Ergebnis kommen
$\emptyset$ und $\mathbb{N}_0$ sind die einzigen entscheidbaren Teilmengen von $\mathbb{N}_0$		×	z.B. $\{n \in \mathbb{N}_0 \mid n \bmod 2 = 0\}$ ist auch entscheidbar
Die Vereinigung kontextfreier Sprachen ist kontextfrei	×		Nichtdeterministischer Kellerautomat leicht konstruierbar (vgl Konstruktion für NFA)
Zu jeder endlichen Sprache L gibt es eine Chomsky-3-Grammatik G mit $L(G) = \overline{L}$ (Komplementsprache)	×		Jede endliche Sprache ist regulär und reguläre Sprachen sind unter Komplementbildung abgeschlossen
Wenn $uv \to ux$ dann $v \to x$		×	Gilt nur für kontextfreie und reguläre Grammatiken und ist daher im Allgemeinen falsch
Wenn $v \to x$ dann $uv \to ux$	×		Nach Definition der Ableitung
Es gibt `loop`-Programme, die nicht terminieren		×	`loop`-Programme dürfen nur eine *vor* Beginn der Schleife bekannte Anzahl Durchläufe machen → Nichtterminierung unmöglich
Wenn eine Menge entscheidbar ist, dann ist ihr Komplement rekursiv aufzählbar	×		Die entscheidbaren Mengen sind unter Komplementbildung abgeschlossen und Entscheidbarkeit impliziert rekursive Aufzählbarkeit.
Jede Teilmenge einer entscheidbaren Menge ist entscheidbar		×	Menge aller gültigen Gödelnummern ist entscheidbar aber nahezu alle ihrer interessanten Teilmengen unentscheidbar, z.B. Halteproblem.
Das Wortproblem für kontextsensitive Sprachen ist entscheidbar	×		Vorlesung: Konstruktion, um für kontextsensitive Sprache eine linear beschränkte Turingmaschinen zu konstruieren, die diese entscheidet.