Aufgabe 9.1 [Kellerautomaten mit zwei Kellern] **6 Punkte**

In dieser Aufgabe betrachten wir Kellerautomaten mit zwei Kellern.
Ein 2-Kellerautomat $\mathcal{A} = (Q, \Sigma, \Gamma, \delta, s, \tau_0, F)$ ist bis auf die Transitionsrelation

$$\delta \subseteq \big(Q \times (\Sigma \cup \{\varepsilon\}) \times \Gamma \times \Gamma\big) \times \big(Q \times \Gamma^* \times \Gamma^*\big)$$

genau so definiert wie ein Kellerautomat. Das unterste Kellersymbol wird auf beiden Kellern als solches verwendet. Wir betrachten nur 2-Kellerautomaten, die mit akzeptierenden Zuständen akzeptieren.

a) Definieren Sie die Semantik von 2-Kellerautomaten (Konfiguration, Startkonfiguration, Nachfolgekonfiguration, Akzeptanz) analog zu der für Kellerautomaten aus Kapitel 9. Welche Bedingung muss für *deterministische* 2-Kellerautomaten zusätzlich gelten? **(2 Punkte)**

b) Zeigen Sie: Zu jeder Turingmaschine gibt es einen *deterministischen* 2-Kellerautomaten, der dieselbe Sprache entscheidet. Zur Vereinfachung können Sie annehmen, dass die Eingabe des deterministischen 2-Kellerautomates durch das Zeichen \$ begrenzt wird. Statt eines Eingabestrings u liest er also den String $u\$$. **(4 Punkte)**

Hinweis

Verwenden Sie $\triangleright$ als unterstes Kellersymbol, den ersten Keller für den Teil des Strings vom linken Rand bis zur aktuellen Position und den zweiten Keller für den Teil des Strings rechts von der aktuellen Position. Der 2-Kellerautomat soll zunächst die Eingabe bis zum Ende einlesen und anschließend die Turingmaschine simulieren.

Lösung:

a) Eine Konfiguration eines 2-Kellerautomaten $\mathcal{A} = (Q, \Sigma, \Gamma, \delta, s, \tau_0, F)$ ist ein Tupel (q, u, v_1, v_2) aus

- einem Zustand $q \in Q$,
- der noch zu lesenden Eingabe $u \in \Sigma^*$,
- dem Inhalt $v_1 \in \Gamma^*$ des ersten Kellers und
- dem Inhalt $v_2 \in \Gamma^*$ des zweiten Kellers.

Die Startkonfiguration bei Eingabe w ist (s, w, τ_0, τ_0). Eine Konfiguration K' ist Nachfolgekonfiguration einer Konfiguration K, in Zeichen $K \vdash K'$,

- falls $(q, \sigma, \tau_1, \tau_2, q', z_1, z_2) \in \delta$ für $K = (q, \sigma u, \tau_1 v_1, \tau_2 v_2)$ und $K' = (q', u, z_1 v_1, z_2 v_2)$;
- falls $(q, \varepsilon, \tau_1, \tau_2, q', z_1, z_2) \in \delta$ für $K = (q, u, \tau_1 v_1, \tau_2 v_2)$ und $K' = (q', u, z_1 v_1, z_2 v_2)$.

Die Notation $K \vdash^* K'$ wird wie für Kellerautomaten definiert. Eine Eingabe $w \in \Sigma^*$ wird von $\mathcal{A}$ akzeptiert, wenn $(s, w, \tau_0, \tau_0) \vdash^* (q, \varepsilon, v_1, v_2)$ für ein $q \in F$ und $v_1, v_2 \in \Gamma^*$. Die von $\mathcal{A}$ akzeptierte Sprache ist die Menge aller Wörter, die $\mathcal{A}$ akzeptiert.

Für einen deterministisch 2-Kellerautomat, muss zusätzlich gelten, dass für alle $q \in Q$, $\sigma \in \Sigma$ und $\tau_1, \tau_2 \in \Gamma$ gilt, dass $|\delta(p, \sigma, \tau_1, \tau_2)| + |\delta(p, \varepsilon, \tau_1, \tau_2)| \leq 1$, wobei

- $\delta(p, \sigma, \tau_1, \tau_2) = \{(q, z_1, z_2) \mid (p, \sigma, \tau_1, \tau_2, q, z_1, z_2)\}$ und
- $\delta(p, \varepsilon, \tau_1, \tau_2) = \{(q, z_1, z_2) \mid (p, \varepsilon, \tau_1, \tau_2, q, z_1, z_2)\}$.

b) Wir konstruieren zu einer Turingmaschine $\mathcal{M} = (Q, \Gamma, \delta, s)$ mit Eingabe-Alphabet $\Sigma = \Gamma - \{\triangleright, \sqcup\}$ einen 2-Kellerautomaten $\mathcal{A} = (Q', \Sigma, \Gamma, \delta', s', \triangleright, F)$ mit $Q' = Q \cup \{s', s_2, \text{ja}, \text{nein}\}$ und $F = \{\text{ja}\}$ entsprechend dem obigen Hinweis.

– Der 2-Kellerautomat liest zunächst die Eingabe $w \in \Sigma^*$ bis zum Ende ein und legt die Eingabe auf den ersten Keller. Danach wird der Inhalt von Keller 1 auf Keller 2 geschoben. Dieses Vorgehen ist notwendig, damit das erste Zeichen der Eingabe oben auf Keller 2 liegt. Anschließend beginnt der Automat die Simulation.

– Zunächst schreibt der Automat im Zustand s' die Eingabe auf den ersten Keller

$$(s', \sigma, \tau, \triangleright, s', \sigma\tau, \triangleright) \in \delta' \text{ für alle } \sigma \in \Sigma \text{ und } \tau \in \Sigma \cup \{\triangleright\}$$

und wechselt nach Lesen der kompletten Eingabe in den Zustand s_2:

$$(s', \$, \tau, \triangleright, s_2, \tau, \triangleright) \in \delta' \text{ für alle } \tau \in \Sigma \cup \{\triangleright\}$$

Die Konfiguration des Automaten entspricht nun der String-Zeigerkonfiguration der Turingmaschine, bei der der Schreib-/Lesekopf auf dem letzten Zeichen der Eingabe steht. Deshalb läuft der Automat zunächst zurück zum linken Rand. Entsprechend wird der Inhalt von Keller 1 in Keller 2 geschrieben. Alle folgenden Übergänge sind ε-Übergänge.

$$(s_2, \varepsilon, \tau_1, \tau_2, s_2, \varepsilon, \tau_1\tau_2) \in \delta' \text{ für alle } \tau_1 \in \Sigma \text{ und } \tau_2 \in \Sigma \cup \{\triangleright\}$$

– Anschließend wechselt der Automat in den Startzustand von $\mathcal{M}$, um die eigentliche Simulation zu beginnen.

$$(s_2, \varepsilon, \triangleright, \tau, s, \triangleright, \tau) \in \delta' \text{ für alle } \tau \in \Sigma \cup \{\triangleright\}$$

Der Automat befindet sich nun in der Konfiguration $(s, \varepsilon, \triangleright, w\triangleright)$. Da wir $\triangleright$ als unterstes Kellersymbol nutzen befindet es sich auch in Keller 2 unter unserer Eingabe w. Die folgenden Übergänge entsprechen den Transitionen der Turingmaschine $\mathcal{M}$ und das aktuelle Symbol ist immer das oberste Symbol von Keller 1.

- Für alle Transitionen der Form $\delta(q, \sigma) = (q', \tau, \leftarrow)$:

$$(q, \varepsilon, \sigma, \tau', q', \varepsilon, \tau\tau') \in \delta' \text{ für alle } \tau' \in \Gamma$$

- Für alle Transitionen der Form $\delta(q, \sigma) = (q', \tau, \downarrow)$:

$$(q, \varepsilon, \sigma, \tau', q', \tau, \tau') \in \delta' \text{ für alle } \tau' \in \Gamma$$

- Für alle Transitionen der Form $\delta(q, \sigma) = (q', \tau, \rightarrow)$:

$$(q, \varepsilon, \sigma, \tau', q', \tau'\tau, \varepsilon) \in \delta' \text{ für alle } \tau' \in \Gamma - \{\triangleright\}$$
$$(q, \varepsilon, \sigma, \triangleright, q', \sqcup\tau, \triangleright) \in \delta' \text{ für alle } \tau' \in \Gamma$$

  Hier, d.h. bei Kopfbewegung nach rechts, müssen wir den Fall gesondert behandeln, dass die simulierte Turingmaschine weiter nach rechts läuft als zuvor. In diesem Fall muss ein neues Leerzeichen eingefügt werden.

– Die Korrektheit der eigentlichen Simulation lässt sich beweisen, indem durch Induktion nach der Schrittzahl von M die folgende Korrespondenz zwischen Konfigurationen von M und A gezeigt wird. Die Konfigurationen $(q, (u, \sigma, v))$ von $\mathcal{M}$ entsprechen den Konfigurationen $(q, \sigma u^R \triangleright, v \triangleright)$ von $\mathcal{A}$.

Aufgabe 9.2 [Reduktion] **5 Punkte**

Wir betrachten die folgenden beiden Entscheidungsprobleme:

Problem: REACH
Gegeben: Gerichteter Graph $G = (V, E)$ und $s, t \in V$.
Frage: Existiert ein Weg von s nach t in G?

Problem: DG-CYCLE
Gegeben: Gerichteter Graph $G = (V, E)$ und $v \in V$.
Frage: Enthält G einen gerichteten Kreis durch v?

Zeigen Sie REACH $\leq$ DG-CYCLE durch Angabe einer passenden Reduktion f. Beweisen Sie, dass f tatsächlich eine Reduktion ist.

Lösung:

Sei G ein gerichteter Graph mit $s, t \in V$. Wir konstruieren aus dem Graphen G einen Graphen G' der genau dann einen Kreis durch s hat, wenn es in G einen Weg von s nach t gibt. Dafür ist zweierlei zu tun:

- Wir müssen garantieren, dass es einen Kreis gibt, falls es in G ein (s, t)-Weg gibt. Dazu fügen wir die Kante (t, s) ein.
- Zudem müssen wir garantieren, dass es nur dann einen Kreis gibt, falls es in G ein (s, t)-Weg gibt. Dafür entfernen wir alle Kanten, die nach s führen. Es ist leicht zu sehen, dass dadurch keine Wege von s nach t zerstört werden. Die (t, s)-Kante bleibt bestehen.

Wir definieren also folgenden gerichteten Graphen $G' = (V, E')$, wobei wir alle eingehenden Kanten von s aus E löschen. Zusätzlich fügen wir die Kante (t, s) ein:

$$E' := \{(t, s)\} \cup \big\{(u, v) \mid (u, v) \in E \text{ mit } u \in V \text{ und } v \in V - \{s\}\big\}$$

Weiterhin definieren wir die total berechenbare Funktion f durch $f(G, s, t) = (G', s)$. Wir zeigen, dass f eine Reduktion von REACH nach DG-CYCLE ist, d.h. wir müssen Totalität, Berechenbarkeit und die Reduktionseigenschaft nachweisen. Die Funktion f ist total, da G' für alle Graphen $G = (V, E)$ und beliebigen Knoten $s, t \in V$ definiert ist. Zudem ist f berechenbar, da die Mengen V und E lediglich kopiert und die neue Kantenrelation E' durch eine Schleife mit einfachen Tests bestimmt werden kann. Weiterhin zeigen wir die beiden Richtungen:

- **$(G, s, t) \in$ REACH $\Rightarrow (G', s) \in$ DG-CYCLE:**
  Sei $(G, s, t) \in$ REACH. Dann ist G ein gerichteter Graph, in welchem ein (s, t)-Weg existiert. Da in G' alle Knoten und Kanten welche auf dem Weg $s, v_1, \ldots, v_{n-1}, t$ existieren, *und zusätzlich* noch die Kante (t, s), ist $s, v_1, \ldots, v_{n-1}, t, s$ ein Kreis in G' durch s. Somit gilt $(G', s) \in$ DG-CYCLE.

- **$(G', s) \in$ DG-Cycle $\Rightarrow (G, s, t) \in$ Reach:**
  Sei $G' \in$ DG-Cycle. Dann enthält G' einen Kreis c durch den Knoten s. Der Kreis c lässt sich also in der folgenden Form schreiben:

  $$c = s, v_1, \ldots, v_{n-1}, s.$$

  Da nur die Kante (t, s) in s hineinführt, ist $v_{n-1} = t$ und damit:

  $$c = s, v_1, v_2, \ldots, v_{n-2}, t, s.$$

  Da alle Kanten von G' außer (t, s) auch in G vorkommen, existiert ein Weg von s nach t in G. Somit gilt $(G, s, t) \in$ Reach.

Insgesamt ist f damit eine Reduktion von Reach auf DG-Cycle. □

Aufgabe 9.3 **4 Punkte**

In dieser Aufgabe nutzen wir folgende Definitionen:

1. Eine Menge M heißt *gleichmächtig* zu einer Menge S, wenn eine Bijektion $f : M \to S$ existiert.
2. Eine Menge M heißt *abzählbar unendlich*, wenn sie gleichmächtig zur Menge $\mathbb{N}$ ist.
3. Eine Menge heißt *überabzählbar*, wenn sie weder endlich noch abzählbar unendlich ist.

Gegeben sei das Alphabet $\Sigma = \{0, 1\}$.

a) Zeigen Sie, dass Σ^* abzählbar unendlich ist. **(1 Punkt)**

b) Wir betrachten nun die Potenzmenge $\mathcal{P}(\Sigma^*)$. Was beschreibt diese Menge? Zeigen Sie mittels Diagonalisierung, dass die Potenzmenge $\mathcal{P}(\Sigma^*)$ überabzählbar ist. **(3 Punkte)**

Lösung:

a) Wir definieren eine Bijektion zwischen Σ^* und $\mathbb{N}$, daraus folgt dass die beiden Mengen gleichmächtig sind. Dazu nutzen wir die Funktion $Str2N$ und definieren $f : \Sigma^* \to \mathbb{N}$ mit $f(w) = Str2N(1w)$. Wir stellen eine 1 vor die Eingabe w, um Injektivität für f zu erhalten. Dies ist notwendig, da $Str2N$ führende Nullen ignoriert. Zudem bilden wir so ε auf 1 ab. Da für jedes $n \in \mathbb{N}$ ein Wort w mit $f(w) = Str2N(1w) = n$ existiert, ist f zudem surjektiv. □

Ein anderer möglicher Lösungsweg ist es, die Funktion f in Phasen zu definieren:

- In Phase 0 wird ε auf 1 abgebildet.
- In Phase $i > 0$ werden jeweils alle Wörter der Länge i auf die nächsten 2^i *freien Zahlen* abgebildet, also z.B. für $i = 1$ wird 0 auf 2 und 1 auf 3 abgebildet.

Die Funktion, die sich dabei ergibt, ist der oben definierten Funktion sehr ähnlich.

b) Die Potenzmenge $\mathcal{P}(\Sigma^*)$ ist die Menge aller Sprachen über dem Alphabet Σ.

Nehmen wir an $\mathcal{P}(\Sigma^*)$ ist abzählbar unendlich, das heißt es gibt eine surjektive Funktion $f : \mathbb{N} \to \mathcal{P}(\Sigma^*)$. Dann können wir die Elemente, d.h. Sprachen, in $\mathcal{P}(\Sigma^*)$ mit Hilfe

des folgenden Schemas auflisten. Zeile $i \in \mathbb{N}$ entspricht der Sprache $L_i = f(i)$. Spalte $j \in \mathbb{N}$ entspricht dem Wort $g(j) = w_j \in \Sigma^*$, wobei $g : \mathbb{N} \to \Sigma^*$ eine Bijektion ist, d.h. alle Wörter aus Σ^* werden aufgezählt (das eine solche Bijektion existiert haben wir in Aufgabenteil (a) gezeigt). Gilt $w_j \in L_i$ schreiben wir an die entsprechende Position (i, j) in der Tabelle ein $+$ andernfalls ein $-$, zum Beispiel

	w_1	w_2	w_3	w_4	w_5	$\dots$
L_1	$-$	$-$	$-$	$+$	$+$	$\dots$
L_2	$+$	$+$	$+$	$-$	$-$	$\dots$
L_3	$+$	$-$	$+$	$-$	$-$	$\dots$
L_4	$-$	$-$	$+$	$+$	$-$	$\dots$
L_5	$-$	$-$	$-$	$+$	$-$	$\dots$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$

Zunächst definieren wir uns eine Relation, welche die Paare (i, j) mit $i \in \mathbb{N}$ und $w_j \in L_j$ enthält, d.h. $R \subseteq \mathbb{N} \times \mathbb{N}$ mit $R := \{(i, j) \mid w_j \in L_i\}$. Nun definieren wir die Sprache $L := \{w_i \mid i \in \mathbb{N} \text{ und } (i, i) \notin R\}$. Im obigen Beispiel enthält L die Wörter w_1 und w_5 jedoch nicht die Wörter w_2, w_3 und w_4. Nach der Annahme, dass f eine surjektive Abbildung ist, muss es ein $k \in \mathbb{N}$ geben, so dass $f(k) = L_k = L$ gilt. Gilt nun $w_k \in L_k$ folgt aus der Definition von L, dass $w_k \notin L$ und falls $w_k \notin L_k$ folgt $w_k \in L$. In jedem Fall folgt daraus ein Widerspruch zu unserer Annahme, dass die Funktion f surjektiv ist, und damit auch dass $\mathcal{P}(\Sigma^*)$ abzählbar unendlich ist. □