Gruppe 8
Loris Reiff

Midterm Vorbereitung – Lösungen

1. (a) Sei $w = (10)^{2^{2^{n^2+1}}} \in \{0,1\}^*$ für alle $n \in \mathbb{N}$. Zeige, dass eine Konstante $d \in \mathbb{N}$ existiert, so dass für alle $n \in \mathbb{N}$ gilt, dass

$$K(w_n) \leq \frac{1}{2}\log_2(\log_2(\log_2|w_n| - 1) - 1) + d$$

Lösung: Wir geben zunächst für jedes $n \in \mathbb{N}$ ein Programm an, welches w_n erzeugt

```
begin
  x := n;
  x := 2^(2^(x*x + 1));
  for i:=1 to x do
    write(10);
end;
```

Der einzige Teil des Maschinencodes dieses Programms, der von w_n abhängt, ist die Darstellung von n in der zweiten Zeile. Der restliche Programmcode hat eine konstante Länge. Also ist die binäre Länge dieses Programms $\lceil\log_2(n+1)\rceil + c$ für eine Konstante c.
Damit lässt sich die Kolmogorov-Komplexität von w_n von oben abschätzen durch

$$K(w_n) \leq \lceil\log_2(n+1)\rceil + c \leq \log_2(n) + c + 1$$

Die Länge von w_n ist:

$$\begin{aligned}
|w_n| = 2 \cdot 2^{2^{n^2+1}} &\iff \log_2|w_n| - 1 = 2^{n^2+1} \\
&\iff \log_2(\log_2|w_n| - 1) = n^2 + 1 \\
&\iff \sqrt{\log_2(\log_2|w_n| - 1) - 1} = n
\end{aligned}$$

somit

$$\begin{aligned}
K(w_n) &\leq \log_2\sqrt{\log_2(\log_2|w_n| - 1) - 1} + c + 1 \\
&\leq \frac{1}{2}\log_2(\log_2(\log_2|w_n| - 1) - 1) + d
\end{aligned}$$

mit $d = c + 1$.

(b) Wir betrachten die Sprache

$$L_1 = \{101^i0^j1^k \mid i + k = j, \text{mit } i, j, k \in \mathbb{N}\}$$

Sei w_n das kanonisch n-te Wort in L_1. Zeige, dass es eine Konstante $c \in \mathbb{N}$ gibt, so dass für alle $n \in \mathbb{N}$ gilt:

$$K(w_n) \leq 2 \cdot \log_2(|w_n|) + c$$

Lösung: Es gibt offensichtlich ein Programm A, welches für ein Wort $w \in \Sigma_{\text{bool}}^*$ entscheidet, ob $w \in L_1$ oder $w \notin L_1$. Nach Satz 2.2 aus dem Buch gilt für das n-te kanonische Wort der Sprache L_1 somit

$$K(w_n) \leq \lceil \log_2(n+1) \rceil + c$$

für eine von n unabhängiger Konstante. Bemerke jedes Wort in L_1 hat eine gerade Länge und ist länger als 1. Des weiteren sehen wir, dass es $j+1$ verschiedene Wörter der Länge $2j+2$ in L_1 gibt mit $j \in \mathbb{N}$. Wir können also die Anzahl aller Wörter mit Länge $\leq 2j+2$, wie folgt abschätzen

$$\begin{aligned}\sum_{i=0}^{j}(i+1) &= j + 1 + \sum_{i=1}^{j} i \\ &= j + 1 + \frac{j(j+1)}{2} \leq (2j)^2 - 1\end{aligned}$$

für alle $j \geq 1$. Das heisst, es gibt maximal $(2j)^2 - 1$ Wörter der Länge höchstens $2j+2$. Wenn wir $n = (2j)^2$ setzen, folgt, dass das die Wörter w_n eine Länge grösser $2j+3$ haben. Somit $|w_n| \geq 2j + 3 = \sqrt{n} + 3 \geq \sqrt{n}$. Nun können wir die Behauptung einfach folgern

$$\begin{aligned}K(w_n) &\leq \lceil \log_2(n+1) \rceil + c \\ &\leq \log_2(n+1) + c' \\ &\leq \log_2(|w_n|^2) + c' \\ &\leq 2\log_2(|w_n|^2) + c'\end{aligned}$$

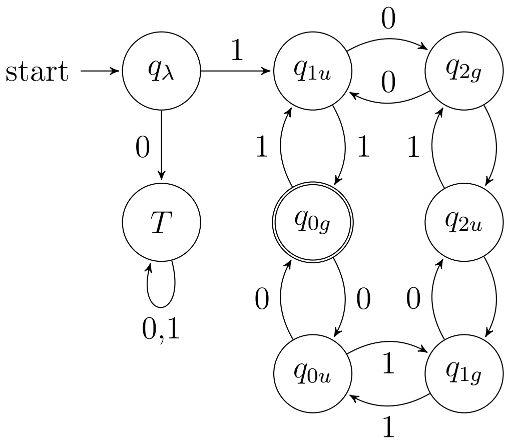
für eine Konstante c'.

2. (a) Entwerfe einen endlichen Automaten (in Diagrammdarstellung) für die Sprache

$$L_2 = \{w \in \{0,1\}^* \mid |w_n| \text{ ist gerade und Nummer}(w) \equiv_3 0 \text{ und } w \text{ ist die kürzeste Binärdarstellung für Nummer}(w)\}$$

Begründe deinen Entwurf.

Lösung: Die Sprache wird von folgendem Automaten akzeptiert



Da $w \in L_2$ eine gerade Zahl zeichen haben muss, wird 0 nicht akzeptiert, des Weiteren muss w die kürzeste Binärdarstellung sein, somit kann w nicht mit 0 beginnen.

Der Automate speichert in seinen weiteren Zuständen, ob das eingelesene Wort gerade ist oder nicht und den Wert für Nummer(w) mod 3. Wörter die in den Zustand q_{iu} mit $i \in \{0,1,2\}$ führen, gilt also, dass das Wort ungerade (u) ist und Nummer(w) $\equiv_3 i$. Bzw. q_{ig} für gerade.

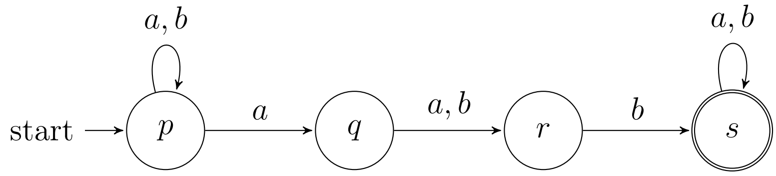
Die Transitionen können wir wie folgt herleiten. Für ein Wort $x_1 x_2 \ldots x_n \in \{0,1\}^*$ gilt

$$\text{Nummer}(x_1 \ldots x_n) = 2 \cdot \text{Nummer}(x_1 \ldots x_{n-1}) + \text{Nummer}(x_{n-1})$$

somit

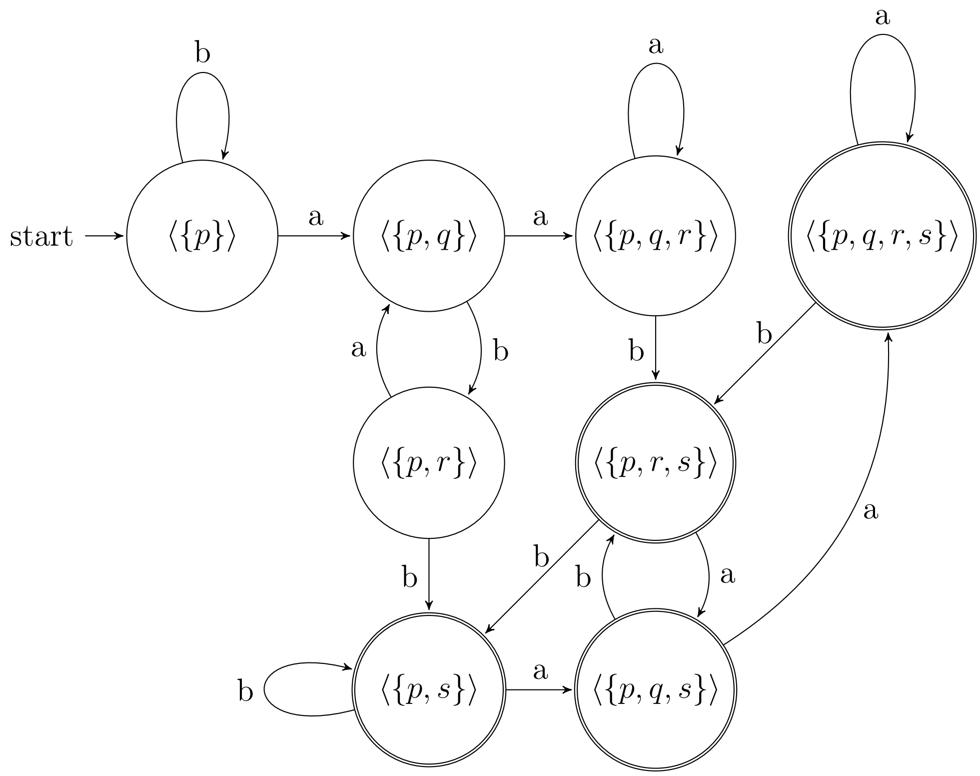
$$\text{Nummer}(x_1 \ldots x_n) \bmod 3 = 2 \cdot \text{Nummer}(x_1 \ldots x_{n-1}) \bmod 3 + \text{Nummer}(x_{n-1}) \bmod 3$$

(b) Verwende die Potenzmengenkonstruktion, um den folgenden nichtdeterministischen endlichen Automaten in einen äquivalenten deterministischen Automaten umzuwandeln.



Nicht erreichbare Zustände können weggelassen werden.

Lösung:



Die akzeptierenden Zustände könnten noch zusammengefasst werden (kurz begründen).

(c) Zeige, dass die folgende Sprache nicht regulär ist. Verwende eine beliebige Methode, die in der Vorlesung vorgestellt wurde.

$$L_3 = \{0^{n(n+1)} \mid n \in \mathbb{N}\}$$

Lösung:

Beweis mittels Lemma 3.3.
Angenommen L_3 sei regulär. Es gibt also einen EA $A = (Q, \{0,1\}, \hat{\delta}_A, q_0, F)$ mit $L(A) = L_3$. Beachten wir die Wörter

$$\lambda, 0, 0000, \ldots, 0^{|Q|^2}$$

Es existieren also $i, j \in \{1, 2, \ldots, |Q|+1\}$ mit $i < j$ und

$$\hat{\delta}_A(q_0, 0^{i^2}) = \hat{\delta}_A(q_0, 0^{j^2})$$

(Schubfachprinzip)
Gemäss Lemma 3.3 im Buch gilt somit

$$0^{i^2}z \in L_3 \iff 0^{j^2}z \in L_3$$

für alle $z \in \{0,1\}^*$. Für $z = 0^i$ haben wir aber einen Widerspruch, denn $0^{i^2}0^i = 0^{i^2+i} \in L_3$ und $0^{j^2}0^i \notin L_3$, da $(j-1)j < j^2 \leq j^2 + i < j(j+1)$. Das heisst also, dass L_3 nicht regulär ist. $\square$

Mittels Pumping-Lemma:
Angenommen L_3 sei regulär. Betrachten wir nun das Wort

$$w = 0^{n_0(n_0+1)}$$

Offensichtlich gilt $|w| = n_0^2 + n_0 \geq n_0$. Folglich gibt es gemäss dem Pumping-Lemma, für das Wort w, eine Zerlegung $w = yxz$, wobei

(i) $|yx| \leq n_0$

(ii) $|x| \geq 1$

(iii) entweder $\{yx^kz \mid k \in \mathbb{N}\} \subseteq L_3$ oder $\{yx^kz \mid k \in \mathbb{N}\} \cap L_3 = \emptyset$

Nach (i) gibt es $y = 0^l$ und $x = 0^m$ für $l, m \in \mathbb{N}$ mit $l + m \leq n_0$. Nach (ii) gilt $m \geq 1$. Und weil $w = 0^{n_0(n_0+1)} \in L_3$ ist, muss also $\{yx^kz \mid k \in \mathbb{N}\} \subseteq L_3$ gelten. Dies ist aber ein Widerspruch, da $yx^2z = 0^{n_0(n_0+1)+m} \notin L_3$, weil $n_0(n_0+1) < n_0(n_0+1) + m < (n_0+1)(n_0+2)$. Somit ist $L_3 \notin \mathcal{L}_{\text{EA}}$ $\square$

Beweis mittels der Methode der Kolmogorov-Komplexität.
Angenommen, L_3 sei regulär. Für jedes $m \in \mathbb{N}$ ist 0^{2m+1} das erste Wort in der Sprache

$$L_{0^{m(m+1)+1}} = \{y \mid 0^{m(m+1)+1}y \in L_3\}$$

da $0^{m(m+1)+1}0^{2m+1} = 0^{m^2+3m+2} = 0^{(m+1)(m+2)}$ Nach Satz 3.1 aus dem Buch existiert eine Konstante $c \in \mathbb{N}$, unabhängig von m, so dass

$$K(0^{2m+1}) \leq \lceil \log 2(1+1) \rceil + c = 1 + c$$

Da es nur endlich viele Programme der konstanten Länge kleiner gleich $1+c$ gibt, aber unendlich viele Wörter der Form 0^{2m+1}, ist dies ein Widerspruch. Also ist die Annahme falsch und L_3 ist nicht regulär. □

3. (a) Zeige $L_{\mathrm{H}}^{C} \leq_{\mathrm{R}} L_{\mathrm{diag}}$
Zur Erinnerung:
Sei w_i das i-te Wort über $\{0,1\}$ und M_i die i-te Turing-Maschine in kanonischer Ordnung.

$$\begin{aligned} L_{\mathrm{diag}} &= \{w_i \in \{0,1\}^* \mid M_i \text{akzeptiert } w_i \text{ nicht}\} \\ L_{\mathrm{H}}^{C} &= \{\mathrm{Kod}(M)\#w \in \{0,1,\#\}^* \mid M \text{ hält nicht auf } w\} \cup \\ &\qquad \{x \in \{0,1,\#\}^* \mid x \text{ hat nicht die Form } \mathrm{Kod}(M)\#w\} \end{aligned}$$

Lösung: Wir zeigen $L_{\mathrm{H}}^{C} \leq_{\mathrm{EE}} L_{\mathrm{diag}}$ was $L_{\mathrm{H}}^{C} \leq_{\mathrm{R}} L_{\mathrm{diag}}$ impliziert.
Wir beschreiben eine TM M, die L_{H}^{C} auf L_{diag} reduziert. Für eine eingabe $x \in \{0,1,\#\}^*$ arbeit M wie folgt:

1. Prüfe, ob x die Form $\mathrm{Kod}(M')\#w$ für eine TM M' und ein Wort $w \in \{0,1\}^*$ hat.
   (i) Falls nein: Konstruiere eine TM M_∞ die immer in eine Endlosschleife geht.
   (ii) Falls ja: Modifiziere eine TM M' zu einer TM $\overline{M}$ welche alle Transitionen von q_{reject} nach q_{accept} umleitet. Und konstruiere eine TM $\widehat{M}$, welche die Eingabe ignoriert und auf w simuliert.
2. Berechne w_i so, dass M_i die konstruierte TM ist.

   Nun zeigen wir:

$$x \in L_{\mathrm{H}}^{C} \iff x \in L_{\mathrm{Diag}}$$

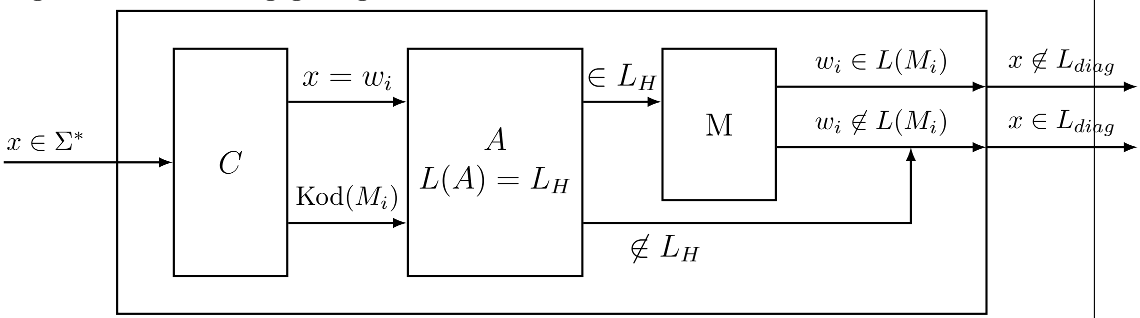
Falls x nicht die Form $\mathrm{Kod}(M')\#w$ hat, gilt $x \in L_{\mathrm{H}}^{C}$ und
$M_i = M_\infty$ hält nicht $\implies$ M_i akzeptiert w_i nicht $\iff$ $w_i \in L_{\mathrm{Diag}}$

und sonst

$$\begin{aligned} x \notin L_{\mathrm{H}}^C &\iff M \text{ hält auf } w \\ &\iff \overline{M} \text{ akzeptiert } w \\ &\iff \widehat{M} = M_i \text{ akzeptiert alles, insbesonder } w_i \\ &\iff M(x) = w_i \notin L_{\mathrm{Diag}} \end{aligned}$$

(b) Zeige, dass $L_{\mathrm{diag}} \leq_{\mathrm{R}} L_{\mathrm{H}}$ gilt

Lösung:
Um $L_{\mathrm{diag}} \leq_{\mathrm{R}} L_{\mathrm{H}}$ zu zeigen, nehmen wir an, A sei ein Algorithmus, der L_H entscheidet. Dann konstruieren wir einen Algorithmus B, der mit Hilfe von A die Sprache L_{diag} entscheidet. Der Algorithmus B ist so strukturiert wie in der folgenden Abbildung gezeigt:



Für eine Eingabe $x \in \Sigma_{\mathrm{bool}}^*$ berechnet das Teilprogramm C das $i \in \mathbb{N}$, so dass $x = w_i$ das i-te Wort über Σ_{bool} in kanonischer Ordnung ist, und die Kodierung Kod(M_i) der i-ten TM. Das Teilprogramm A für L_{H} bekommt Kod(M_i) und $x = w_i$ als Eingabe in der Form Kod$(M_i)\#x$. Falls A die Eingabe Kod$(M_i)\#x$ verwirft, dann hält M_i nicht auf w_i, also akzeptiert M_i das Wort w_i auch nicht. Also gilt $w_i \in L_{\mathrm{diag}}$ und B akzeptiert seine Eingabe $x = w_i$. Falls A die Eingabe Kod$(M_i)\#x$ akzeptiert, dann hält M_i auf w_i. In diesem Fall simuliert das Teilprogramm M die Arbeit von M_i auf w_i. Diese Simulation endet auf jeden Fall in endlicher Zeit. Falls die Simulation ergibt, dass M_i das Wort w_i akzeptiert, dann gilt $w_i \in L_{diag}$ und B verwirft seine Eingabe $x = w_i$. Sonst verwirft M_i das Wort w_i, es gilt also $w_i \in L_{\mathrm{diag}}$ und B akzeptiert seine Eingabe $x = w_i$.