GTI Übungsblatt 10
Tutor: Marko Schmellenkamp
ID: MS1
Übung: Mi 16-18

Max Springenberg, 177792

10.1

Siehe ausgedrucktes Blatt vom Aufgabenzettel.

10.2

10.2.1

$L_a = L_1 \cup L_2$

Wir wissen, dass L_n von einer TM $M_n, n \in \{1, 2\}$ entschieden wird.
Ferner muss unsere TM nach Aufgabenstellung nicht für alle Eingaben terminieren, wenn diese nicht in der Sprache sind.
Damit ergäbe sich M_a aus den Turingmaschinen M_1, M_2 wie folgt:

1. Zustände aus M_1, M_2 werden so umgennant, dass sie nicht konkurierend sind
2. Der Starzustand aus M_a sei der Startzustand aus M_1
3. M_a simuliert zunächst M_1, bis entweder akzeptiert wird, oder nicht. wenn nicht wird der String wieder auf die initiale Eingabe gesetzt und in den Startzustand von M_2 gewechselt.

10.2.2

$L_b = L_1 \circ L_2$

Wir wissen, dass L_n von einer TM $M_n, n \in \{1, 2\}$ entschieden wird.
Wir können die TM M_b wie folgt konstruieren.

1. Zustände aus M_1, M_2 werden so umgennant, dass sie nicht konkurierend sind
2. Es wird mit der TM M_1 begeonnen. Zunächst wird getestet, ob die ganze Eingabe in L_1 ist. Wenn nicht wird das letzte Symbol $\sigma \in \Sigma$ der Eingabe markiert (durch z.B. $\underline{\sigma}$) und erneut getest ob der Teilstring aller nicht markierten Zeichen in L_1 ist, solange bis der erste Teilstring aus L_1 gefunden wurde.
3. Nun wird das letzte Zeichen des Teilstring aus L_1 markiert und alle zuvor markierten Zeichen demarkiert. M_2 wird ab dem markiertem Zeichen simuliert.
4. Wenn der Verbliebene String nicht in L_2 ist werden alle Zeichen rechts vom markierten letzten Symbol, dass noch zu dem Teilstring aus L_1 markiert und auf allen unmarkierten Zeichen wieder, wie zuvor durch sequentielles suchen und markieren des letzten Zeichens vestgestellt, welches der nächst längste Teilstring aus L_1 ist.
5. weiter bei 3

10.3

Problem: **A**
Gegeben: **TM** M**,** $k \in \mathbb{N}_0$
Frage: **Erzeugt** M **bei Eingabe** 0^k **die Ausgabe 1?**

Nach der Vorlesung existiert keine TM M_{hw}, die testet, ob eine TM bei einer Eingabe I 'hello world' ausgibt.
Der Beweis, dass es keine solche Turingmaschine M_1 für das Problem A gibt kann analog gezeigt werden.
In der Vorlesung wurde angenommen, dass eine TM H existiert die testen kann ob eine TM M bei einer Eingabe I, 'hello world' ausgibt, und dem entsprechend selbst 'ja', oder 'nein' ausgibt. Anschließend wurde die TM H_1 betrachtet, die 'ja' ausgibt, wenn eine TM M bei eingabe I 'hello world' ausgibt und 'hello world' sonst. Abschließend wurde eine TM H_2 eingeführt, die 'ja' ausgibt wenn M unter der Eingabe M 'hello world' aus gibt und 'hello world' sonst.
Es wurde gezeigt, dass H unter der Eingabe H_2, H_2 nicht korrekt ist und es ferner keine TM gibt, die 'hello world' unter Eingabe testet.
Wenn man nun im Beweis 1 anstatt 'hello world' betrachtet, so erkennt man, dass es auch keinen 1-Tester gibt.

Da es keine Möglichkeit gibt mit einer Turingmaschine zu testen, ob eine andere TM M bei Eingabe 0^k 1 ausgibt, kann es auch keine Turingmaschine M' mit $L(M') = \{k | M$ gibt bei Eingabe 0^k 1 aus$\}$ geben.
Ferner ist A damit auch nicht semientscheidbar.

10.3.1

Problem: **B**
Gegeben: **TM** M
Frage: **Erzeugt** M **bei keiner Eingabe die Ausgabe 1?**

Damit M semientscheidbar wäre müsste es möglich sein zu testen, ob M bei einer Eingabe 1 ausgibt. Nach Aufgabenteil a) ist dies nicht möglich.
Wenn es also nicht möglich ist zu testen, ob eine TM eine bestimmte Ausgabe macht, so ist es auch nicht möglich zu testen, ob eine TM diese ausgabe nicht macht. Ferner ist es nicht möglich zu testen, ob eine TM diese Ausgabe nie macht.

10.4

Konstante Funktionen sind primitiv rekursiv.
$g(x) = 1, h(x) = 0$ sind solche konstanten Funktionen.

$even(x)$ lässt sich nun auch als:
$$even(x) = \begin{cases} g(x) & \text{, x ist gerade} \\ h(x) & \text{, x ist ungerade} \end{cases}$$
schreiben.

Primitive Rekursionen mit primitiv rekursiven Funktionen sind auch primitiv rekursiv.
Wir stellen fest *even* kann auch wie folgt definiert werden:
$even(0) = g(0)$
$even(1) = h(1)$
$even(x + 2) = even(x)$
da g, h primitiv rekursiv sind, ist *even* nach der Definition von primitv rekursiven Funktionen aus der Vorlesung auch primitiv rekursiv.