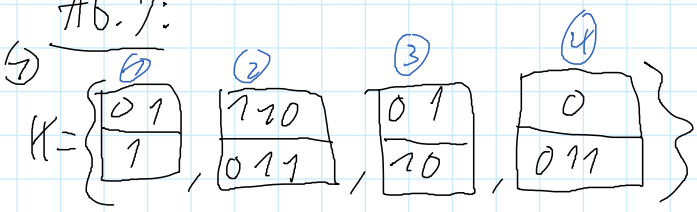
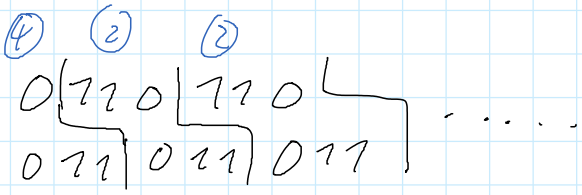
Blatt 6 w

Wednesday, May 27, 2020 10:05 AM

A6.1:

a)



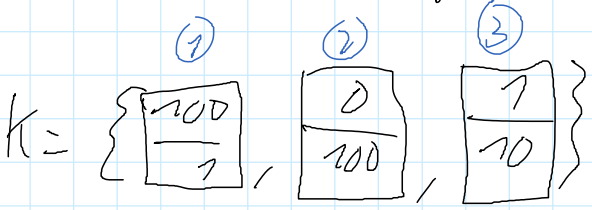


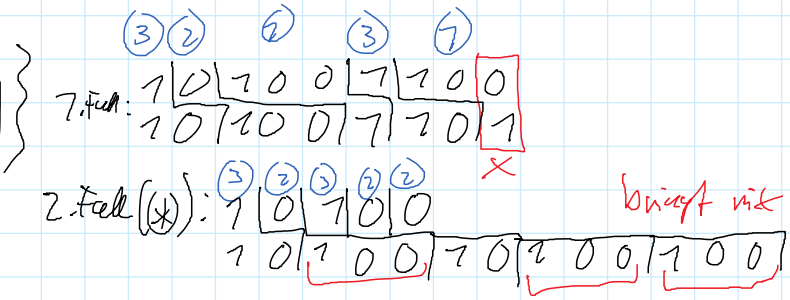
Beobachtungen:

- Muss mit (0,011) beginnen, da bei allen Anderen 1 nicht übereinstimmen
- Danach muss aber 11 folgen, was nur mit (2) erreicht wird.
- Ausschließend befinden wir uns aber wieder in der selben Situation wo nur (2) passt => Man befindet sich in einer Endlosschleife.

=> K hat keine Lösung

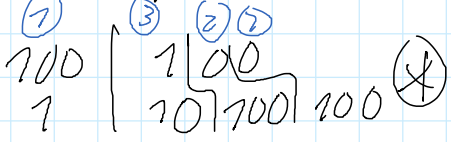
b)



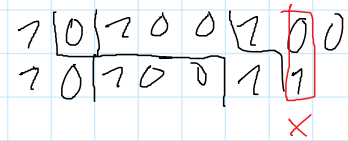


Beobacht.:

- Es gibt zwei mögl. eine 100 oben zu ergeben:



- Es gibt keine Mögl. unten eine Vorrangige 0 zu setzen. => Man kann nicht mit (1) beginnen.



Mehr wie gleich vielen gibt es nicht => es gibt keine Lösung

A6.2:

A6.2:

$H' \stackrel{!}{=} H_{\leq |w|^2} = \{\langle M\rangle \mid M$ hält auf allen w nach höchstens $|w|^2$ vielen Schritten oder gar nicht$\}$

↑ U.B. das jedes mal zu schreiben.

$\Rightarrow \overline{H'} = \{\langle M\rangle \mid M$ hält nach mindestens $|w|^2-1$ Schritten$\} \cup \{x \in \{0,1\}^* \mid x \neq \langle M\rangle\}$

Idee: $H_\varepsilon \leq H'$

$\Rightarrow \langle M\rangle \in H_\varepsilon \Leftrightarrow \langle M^*\rangle \in H'$

mit $f(\langle M\rangle) = \langle M^*\rangle$

$x \notin H_\varepsilon$: M^* nach $|w|^2-1$ Schritten halten $\notin H'$

Sei $x \in H_\varepsilon$

$\Rightarrow x = \langle M\rangle$ mit M hält auf ε

$\Rightarrow$ Wähle w das M nach i Schritten terminiert mit $i \leq |w|^2$

$\Rightarrow M^*$ macht mehr als $|w|^2$ Schritte

$\Rightarrow \langle M^*\rangle \in H'$

M^* auf Eingabe w

1) Sim. M auf ε für $|w|^2-1$ Schritte

Wenn hält $\Rightarrow$!

Wenn nicht $\Rightarrow$ okay.

0) für $x \neq \langle M\rangle$

$f(x) = x$

$x \in H_\varepsilon$: M^* macht mehr als $|w|^2$ Schritte

Sei $x \notin H_\varepsilon$

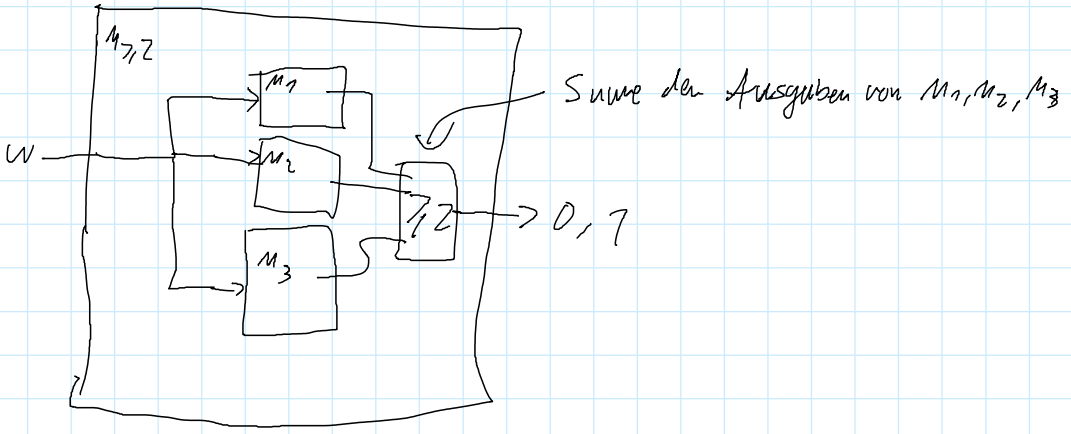
1. Fall $x \neq \langle M\rangle \Rightarrow f(x) = \langle M\rangle \notin H'$

2. Fall $x = \langle M\rangle \Rightarrow$ Wähle hier w s.d. M hält? Ka warum man sowas machen darf (Wurde auch im Beweis zu Theorem 2.18 gemacht...)

A6_3:

a) Beweise Turing-Reduktion

Seien $M_{\neq 2}, M_1, M_2, M_3$ TM die $L_{\neq 2}, L_1, L_2, L_3$ erkennen.



Sei $w \in L_{\neq 2}$.

$\Rightarrow |I(w)| \neq 2 \Rightarrow w \in L_i$ mit $i \in I(w)$

$\Rightarrow \exists$ mind. 2 $i \in I(w)$ s.d. M_i w akz.

$\Rightarrow$ $\neq 2$-Bed. akz.

$\Rightarrow M_{\neq 2}$ akz. w

Sei $w \notin L_{\neq 2} \Rightarrow |I(w)| < 2$

$\Rightarrow$ Dann $i \in I(w)$ mit M_i akz. w oder keine der 3 TM akz.

$\Rightarrow$ $\boxed{\neq 2}$-Block verwirft oder wird nicht erreicht.

$\Rightarrow M_{\neq 2}$ verw. w oder hält nicht

$\Rightarrow L_{\neq 2}$ ist rekursiv aufzählbar

b) Hier wird fast die gleiche Turing Reduktion gebaut nur wird Den $\boxed{\neq 2}$ Block mit einem $\boxed{=2}$-Block ersetzt wird, welcher 1 wiedergibt, wenn genau zweimal 1 eingegeben ist, sonst 0. ($M_{=2}$ entsch. $L_{=2}$)

Sei $w \in L_{=2}$
$\Rightarrow |\Gamma(w)| = 2$
$\Rightarrow$ zwei TM akz. w
$\Rightarrow \boxed{=2}$-Block akz.
$\Rightarrow M_{=2}$ akz. w

Sei $w \notin L_{=2}$
$\Rightarrow |\Gamma(w)| \neq 2$
$\Rightarrow$ weniger oder mehr als 2 TM akz. w
$\Rightarrow \boxed{=2}$-Block akz. nicht
$\Rightarrow M_{=2}$ akz. w nicht.

$\Rightarrow L_{=2}$ rek. aufzählbar.

AG. 4: ξ: Sprache L rek. $\Leftrightarrow \exists$ Aufzählen E zu L oder L in lexikographischer Reihenfolge auf das Ausgabeband schreibt

„$\Leftarrow$": Hierfür ist es wichtig zu erkennen, dass $\forall w \in \{0,1\}^* \exists i \in \mathbb{N}$ mit $w = w_i$ wobei w_i das i-te Wort aus $\{0,1\}^*$ in lexikographischer Reihenfolge ist.
Existiert solch ein Aufzählen E so kann man eine TM M_L bauen, die L entscheidet:
M_L berechnet für eine Eingabe w das i mit $w = w_i$ und simuliert

E so lange bis $w_i = w$ auf dem Band steht, oder ein w_j mit $j > i$ gefunden wird. Im ersten Fall akz. M_L w und im zweiten Fall verwirft M_L w.

„$\Rightarrow$" L wfk. $\Rightarrow \exists M_L$ das L entscheidet.

Dann kann ein Aufzählen E folgendermaßen konstruiert werden:

E generiert alle $w \in \{0,1\}^*$ in lexikographischer Reihenfolge und simuliert M_L auf alle diesen Eingaben.

↳ Wenn M_L w akz. schreibt E w auf das Ausgabeband gefolgt von einem #

↳ Wenn M_L w verw. schreibt E nichts und berechnet das nächste Wort.

Da M_L immer terminiert ist gewährleistet, dass E (bei einer Simulation von M_L auf eine Eingabe w) nicht in einer Endlosschleife landet.