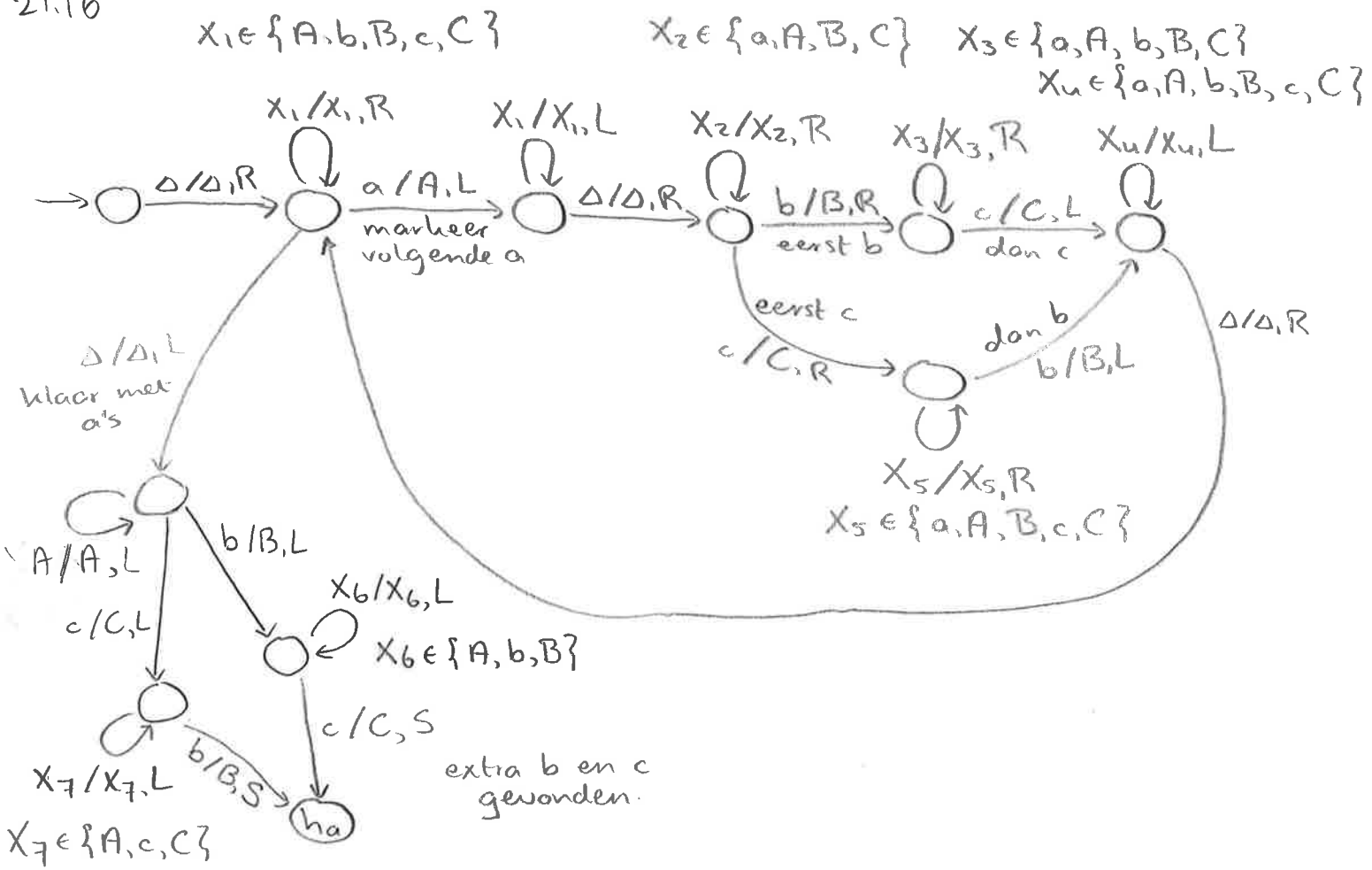
21.08

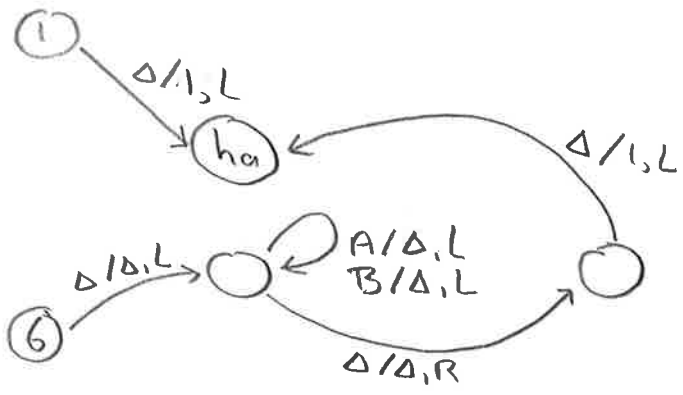
1) T gaat herhaaldelijk van links naar rechts op zoek naar de eerste a. Als hij die vindt, markeert hij die als A. (op zoek) Vervolgens gaat hij van links (vanaf het begin van de string) naar rechts naar een b en een c (in een of andere volgorde), en markeert die als B en C. Elke a in x moet namelijk zowel door b als door c gematched worden.
Als er geen a meer over is, gaat T van rechts naar links op zoek naar nog een b en een c, want het aantal b's en het aantal c's moet strict groter zijn dan het aantal a's. Als T die extra b en c vindt, accepteert T.

21.16



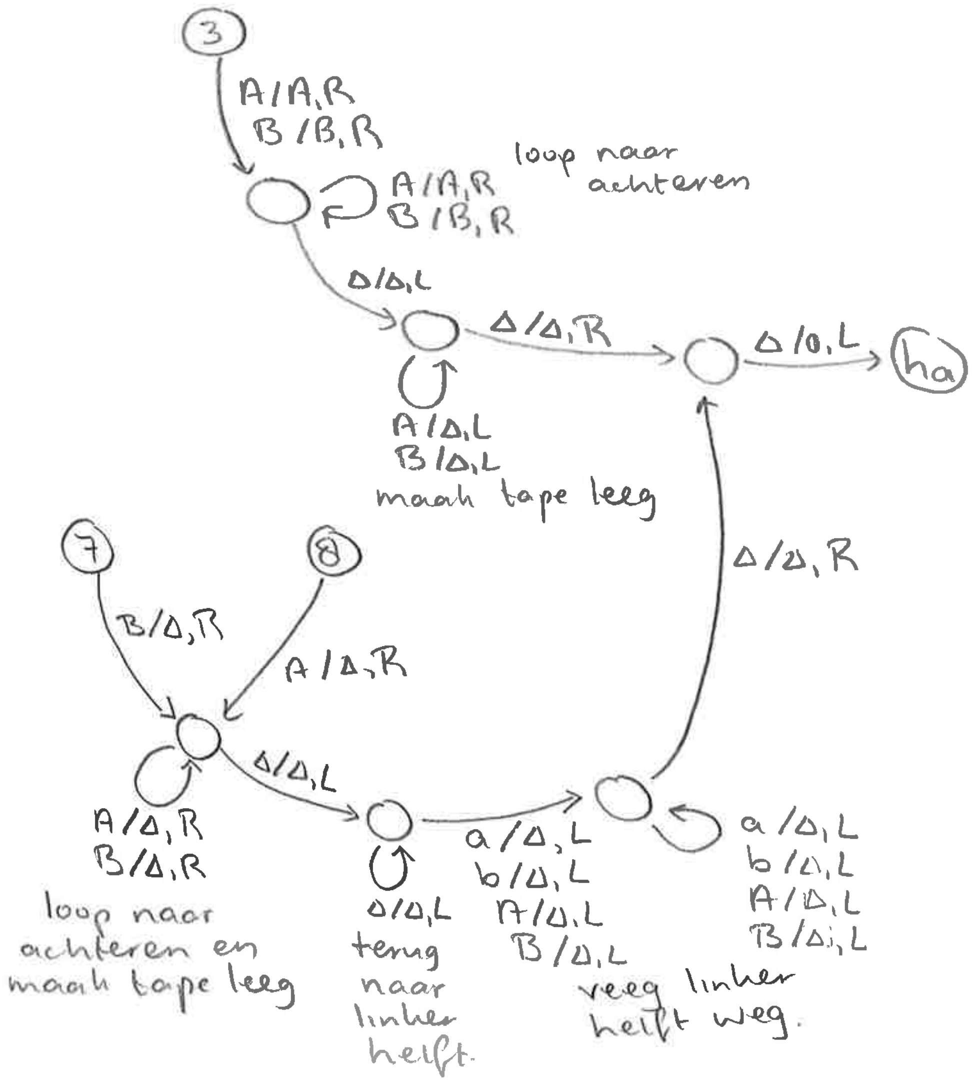
21.29 / 21.44

2] Als T_1 accepteert moet T_2 een 1 achterlaten op de tape. Hiertoe veranderen we transities van toestanden 1 en 6 naar ha als volgt:



Als T_1 verwerpt, moet T_2 een 0 achterlaten op de tape.
Dat is aan de orde als we
* in toestand 3 een hoofdletter zien (als de invoer oneven lengte heeft
* in toestand 7 een B zien } als de rechter helft verschilt
* in toestand 8 een A zien } van de linker helft.

We voegen de volgende transities en toestanden toe



22.04

3) We nummeren
toestanden: ha = 1, $q_0 = 3$, $q_1 = 4$
tapesymbolen: Δ = 1, a = 2
richtingen: R = 1, L = 2, S = 3

Een transitie

$$(p) \xrightarrow{\sigma/\tau,\,D} (q)$$

$$1^{n(p)}0\,1^{n(\sigma)}0\,1^{n(q)}0\,1^{n(\tau)}0\,1^{n(D)}0$$

waarbij n(..) het nummer van de toestand, tapesymbool, richting is.

We krijgen dan voor T:

1110 10 11110 10 10 0 1111 0 11 0 1111 0 11 0 110 0 1111 0 10 10 10 110 0 0
q_0 Δ q_1 Δ R q_1 a q_1 a L q_1 Δ ha Δ L

22.16

4)

$S \rightarrow LaR$	kortst mogelijke string a, met linker en rechter grens L en R
$L \rightarrow LaMb$	genereer een nieuwe eerste reeks a, en een boodschapper M die alle reeksen hier achter 1 langer maakt.
$Mb \rightarrow baM$	voeg bij passeren van b een extra a toe voor de reeks achter de b.
$Ma \rightarrow aM$	loop naar rechts
$MR \rightarrow R$	M heeft zijn werk gedaan en verdwijnt
$L \rightarrow \Lambda$, $R \rightarrow \Lambda$	L en R hebben hun werk gedaan en verdwijnen

22.24

5) (a)
L_1 en L_2 zijn recursief.
Er bestaan dus Turingmachines T_1 en T_2 die de karakteristieke functies van respectievelijk L_1 en L_2 berekenen.
T werkt als volgt:
* kopieer invoer x
* voer T_1 uit op de achterste kopie van x
* als T_1 een 0 achterlaat op de tape ($x \notin L_1$), veeg dan de tape schoon en laat zelf ook een 0 achter op de tape
* als T_1 een 1 achterlaat op de tape ($x \in L_1$), verwijder dan deze 1 en voer T_2 uit op de voorste kopie van x.
* de uitvoer van T_2 (0 of 1 op de tape) is gewoon de uitvoer van T

22.32

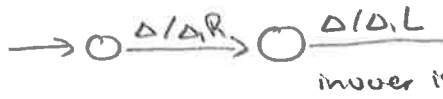
(b) Uit de eerste eigenschap van een redelijke coderingsfunctie volgt dat $E = \{e(I) \mid I \text{ is instantie van } P\}$ een recursieve taal is.
Nu geldt dat $N(P) = Y(P)' \cap E$.
Omdat $Y(P)$ recursief is, is ook het complement $Y(P)'$ recursief. (want gesloten onder complement)
Omdat $Y(P)'$ en E recursief zijn, is $N(P) = Y(P)' \cap E$ ook recursief (want gesloten onder doorsnede)

22.40

6) $\underset{\text{instanties}}{\text{Accepts}_{T_1}\text{-}\Lambda} \leq \underset{T_3}{\text{AcceptsSomethingEvenSteps}}$

Laat Turingmachine T_1 een willekeurige instantie van $\text{Accepts}_{T_1}\text{-}\Lambda$ zijn.
T_3 werkt dan als volgt:

T_3 controleert of zijn invoer Λ is.
Zo niet, dan verwerpt T_3
Zo wel, dan voert T_3 een variant T_1' van T_1 uit op invoer Λ:



(dus eerst twee stappen, en dan T_1')

In T_1' is elke transitie van T_1 vervangen door twee transities, met een hulptoestand voor elke transitie. B.v.:

Er geldt:

T_1 is ja-instantie van Accepts-Λ $\Leftrightarrow$ T_1 accepteert Λ $\Rightarrow$ T_3 accepteert Λ, en doet dat in een even aantal stappen $\Rightarrow$ er is een string $x \in \Sigma^*$ (namelijk $x = \Lambda$) die door T_3 in een even aantal stappen wordt geaccepteerd $\Leftrightarrow$ T_3 is ja-instantie van AcceptsSomethingEvenSteps.

T_1 is nee-instantie van Accepts-Λ $\Leftrightarrow$ T_1 accepteert Λ niet $\Rightarrow$ T_3 accepteert Λ niet en T_3 accepteert sowieso andere strings ook niet $\Rightarrow$ T_3 accepteert geen enkele string $\Rightarrow$ T_3 accepteert zeker geen enkele string in een even aantal stappen $\Leftrightarrow$ T_3 is nee-instantie van AcceptsSomethingEvenSteps.

De constructie van T_3 uit T_1 is uiteraard algoritmisch uit te voeren, zodat we een geldige reductie hebben.
Omdat dus Accepts-$\Lambda \leq$ AcceptsSomethingEvenSteps, en gegeven is dat Accepts-Λ niet beslisbaar is, is AcceptsSomethingEvenSteps ook niet beslisbaar.

23.03