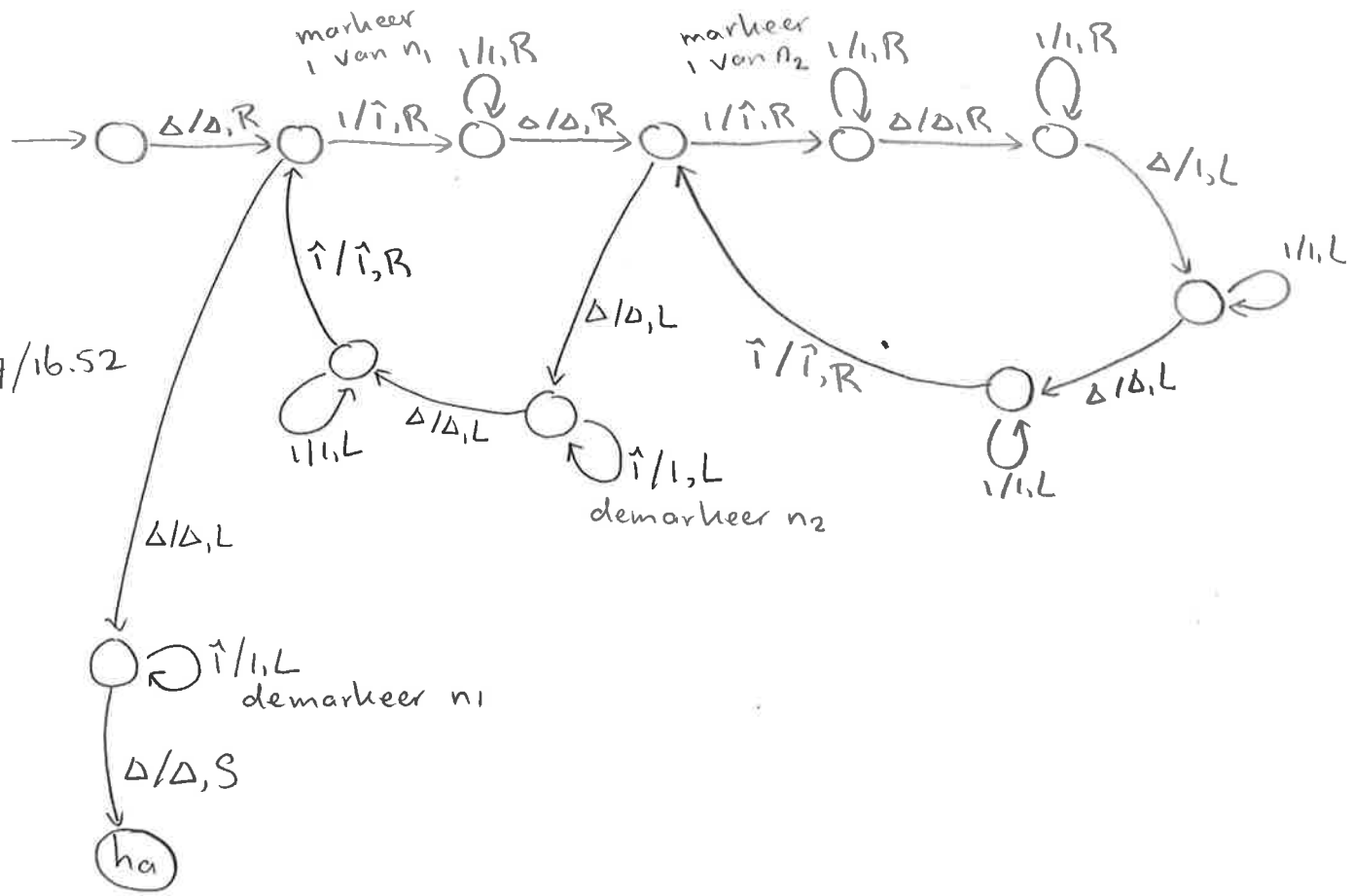
16.22

1a) M_1 markeert de eerste 1 van n_1 als $\hat{1}$, en kopieert vervolgens de 1'en van n_2 één voor één naar de tape achter n_2. De reeds gekopieerde 1'en van n_2 worden gemarkeerd als $\hat{1}$.
Als alle 1'en van n_2 gemarkeerd, en dus gekopieerd zijn, worden deze 1'en gedemarkeerd. We gaan dan naar de tweede 1 van n_1, markeren die als $\hat{1}$ en kopieren opnieuw de 1'en van n_2 naar de tape achter n_2.
Enzovoort tot en met de laatste 1 van n_1. Dan demarkeren we de 1'en van n_1 en accepteren met de leeshop op vakje 0.
Dit alles gaat automatisch ook goed als n_1 en/of $n_2 = 0$.

16.29
16.32

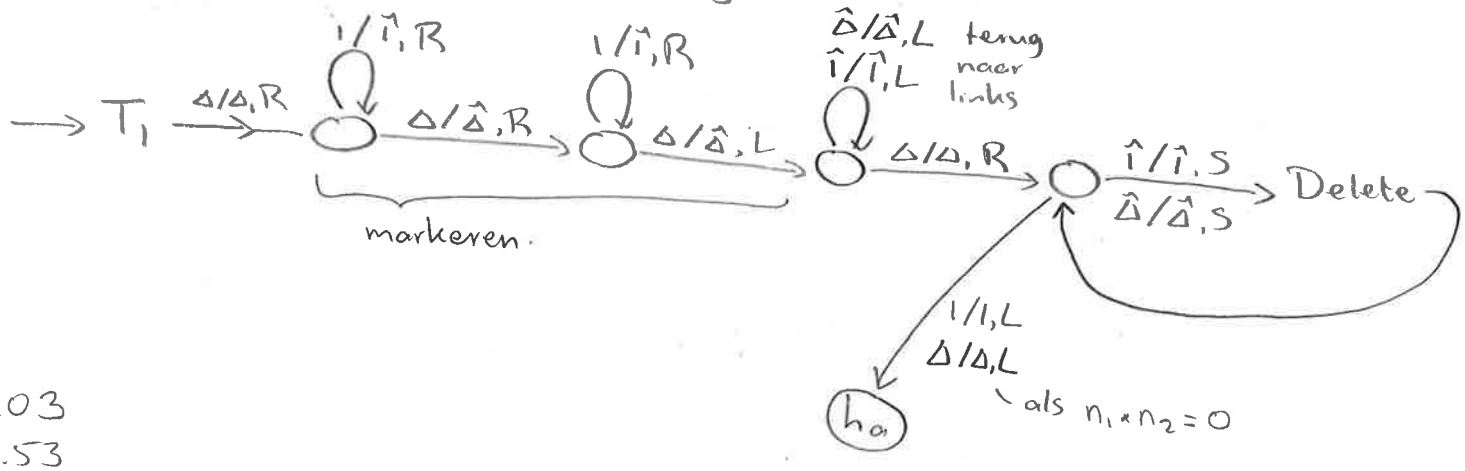


16.37/16.52

16.56

b)

We passen eerst component T_1 toe.
Het enige dat we daarna nog moeten doen is de 1'en van n_1 en n_2 en de Δ's na n_1 en na n_2 weghalen. Dat kan met herhaaldelijk toepassen van de component Delete. Om die component goed en gemakkelijk te laten werken, markeren we de 1'en van n_1 en n_2 en de twee Δ's die we willen weghalen met $\hat{\ }$.



17.03
20.53

2)

Voor een invoer $\underline{x} \in \{1\}^*$ begint T met tape-inhoud $\underline{\Delta} x$
Met NB komen we bij $\Delta x \underline{\Delta}$.
Met G wordt een (willekeurige) string $y \in \{1\}^*$ gegenereerd, wat geeft: $\Delta x \underline{\Delta} y$
Met Copy krijgen we $\Delta x \underline{\Delta} y \Delta y$
Met T_2 krijgen we $\Delta x \underline{\Delta}(y \cdot y)$ (het kwadraat van y)
Met PB komen we bij $\underline{\Delta} x \Delta (y \cdot y)$
Equal accepteert, dan en slechts dan als $x = y^2$.
Omdat y een willekeurig natuurlijk getal voorstelt, accepteert T niet-deterministisch alle kwadraten x

21.03

3)
* De aard van het model. We hebben met de Turingmachine de belangrijkste aspecten van een menselijke computer proberen te modelleren.
* Verschillende varianten / uitbreidingen van de Turingmachine (meerdere tapes, niet-determinisme) resulteerden niet in machines die meer konden
* Andere, compleet verschillende formalismen (unrestricted grammars, mu-recursieve functies) bleken net zo krachtig als de Turingmachine

* Niemand heeft tot nu toe een algoritmische procedure kunnen bedenken die niet door een Turingmachine uit te voeren is.

21.10

4) a) De eerste vijf elementen in de canonieke volgorde:
baa, baaa, baaaa, bbaaa, abbaaa,

21.11 En vervolgens: baaaaa, bbaaaa, abbaaaa, baaaaaa

b) $S \to SABC \mid SBC \mid SC \mid LBCC$

A voor a in a^i
B voor b in b^j
C voor a in a^k

$S \to SABC$: als we een A toevoegen dan ook een B en een C, want $i \leq j, k$.

$S \to SBC$: als we een B toevoegen, dan ook een C, want $j \leq k$

$S \to SC$: we mogen extra C's toevoegen

$S \to LBCC$: zet L neer links van de string, en dan ook BCC, zodat j echt groter wordt dan A en k, groter wordt dan j.
echt

$BA \to AB$ $CA \to AC$ $CB \to BC$
geef hoofdletters kans om zich te sorteren.

$LA \to A'$ $A'A \to A'A'$ $A'B \to A'B'$ $B'B \to B'B'$
zet hoofdletters van links naar rechts om in hoofdletters met accent, als ze in goede volgorde staan.
$B'C \to B'C'$ $C'C \to C'C'$

$A' \to a$ $B' \to b$ $C' \to a$
zet hoofdletters met accent om in de juiste kleine letters.

Hoofdletters met accent zijn dus om goede volgorde af te dwingen.

21.36

5 a) L_1 en L_2 zijn recursief opsombaar.
Er zijn dus Turingmachines T_1 en T_2, zodat $L_1 = L(T_1)$ en $T_2 = L(T_2)$.
Laat nu T een Turingmachine met twee tapes zijn, die zijn invoer x kopieert van tape 1 naar tape 2, en vervolgens
op tape 1 T_1 simuleert voor invoer x
op tape 2 T_2 simuleert voor invoer x.
Als T_1 <u>of</u> T_2 hierbij accepteert, accepteert T zelf.
Als een van beide T_1 of T_2 verwerpt, gaat T gewoon verder met de andere (of links van tape loopt)
Als T_1 en T_2 allebei verwerpen, verwerpt T ook.
Als T_1 of T_2 in oneindige lus raakt, en de ander verwerpt of raakt ook in oneindige lus, dan raakt ook T in een oneindige lus. (dus accepteert niet)
Er geldt nu: T accepteert invoer x $\iff$
T_1 of T_2 accepteert invoer x $\iff$
$x \in L(T_1) = L_1$ of $x \in L(T_2) = L_2$ $\iff$
$x \in L_1 \cup L_2$

21.4b Het antwoord is nee.
b) Neem $L = NSA$. L is niet recursief opsombaar.
L is wel aftelbaar oneindig, dus $L = \{a_0, a_1, a_2, \dots,\}$
$= \{a_0\} \cup \{a_1\} \cup \{a_2\} \cup \dots$
Elk van de talen $L_i = \{a_i\}$ is eindig, en dus zeker recursief opsombaar. Maar $L = \bigcup_{i=0}^{\infty} L_i = NSA$ is dus niet recursief opsombaar.

21.5 2

6 a) Als we simpelweg $T_2 = T_1$ nemen, kan een ja-instantie van AcceptsAtleastEven een nee-instantie van AcceptsEverything geven, b.v. als T_1 een Turingmachine is die alle strings van even lengte accepteert en alle strings van oneven lengte verwerpt.

21.5 g

22.04

b) AcceptsAtleastEven $\leq$ AcceptsEverything
instantie T_1 T_2

Laat T_1 een willekeurige instantie van AcceptsAtleastEven

Instantie T_2 van AcceptsEverything heeft hetzelfde invoer-alfabet Σ als T_1 en werkt als volgt:

* Als de invoer van T_2 oneven lengte heeft, accepteert T_2
* Als de invoer van T_2 even lengte heeft, simuleert hij T_1 op deze invoer

Er geldt nu:

T_1 is een ja-instantie van AcceptsAtleastEven $\Leftrightarrow$
T_1 accepteert (in ieder geval) alle strings over Σ van even lengte $\Leftrightarrow$ T_2 accepteert (in ieder geval) alle strings over Σ van even lengte $\Leftrightarrow$ T_2 accepteert alle strings over Σ $\Leftrightarrow L(T_2) = \Sigma^*$

(omdat T_2 sowieso alle strings over Σ van oneven lengte accepteert.)

$\Leftrightarrow$
T_2 is een ja-instantie van AcceptsEverything

Uiteraard is T_2 op algoritmische wijze uit T_1 te verkrijgen

22.13.