10.ub5

1(a)

De eerste vijf elementen in de canonieke volgorde van L:

$i=0, k=1$:	ba	En vervolgens	
$i=0, k=2$:	bbaa	$i=0, k=4$	bbbbaaaa
$i=1, k=2$:	abbbaa	$i=2, k=3$	aabbbbbaaa
$i=0, k=3$:	bbbaaa	$i=1, k=4$	abbbbbaaaa
$i=1, k=3$:	abbbbaaa	$i=0, k=5$	bbbbbaaaaa

10.u8

(b) T_1 controleert eerst of zijn invoer x van de vorm $a^i b^j a^k$ is, met $i \geq 0$ en $j, k \geq 1$. Zo nee, dan verwerpt/crasht hij meteen.

Zo ja, dan markeert T_1 steeds (van links naar rechts) een a van a^i als A, waarna hij (eveneens van links naar rechts twee b's van b^j markeert als B, en een a van a^k als A.

Als dat niet lukt, omdat er niet voldoende b's zijn of niet voldoende a's van a^k (dus $j < 2i$ of $k < i$), crasht T_1.

Als dit wel lukt en alle a's van a^i zijn gemarkeerd, wordt gekeken of er nog b's van b^j over zijn.

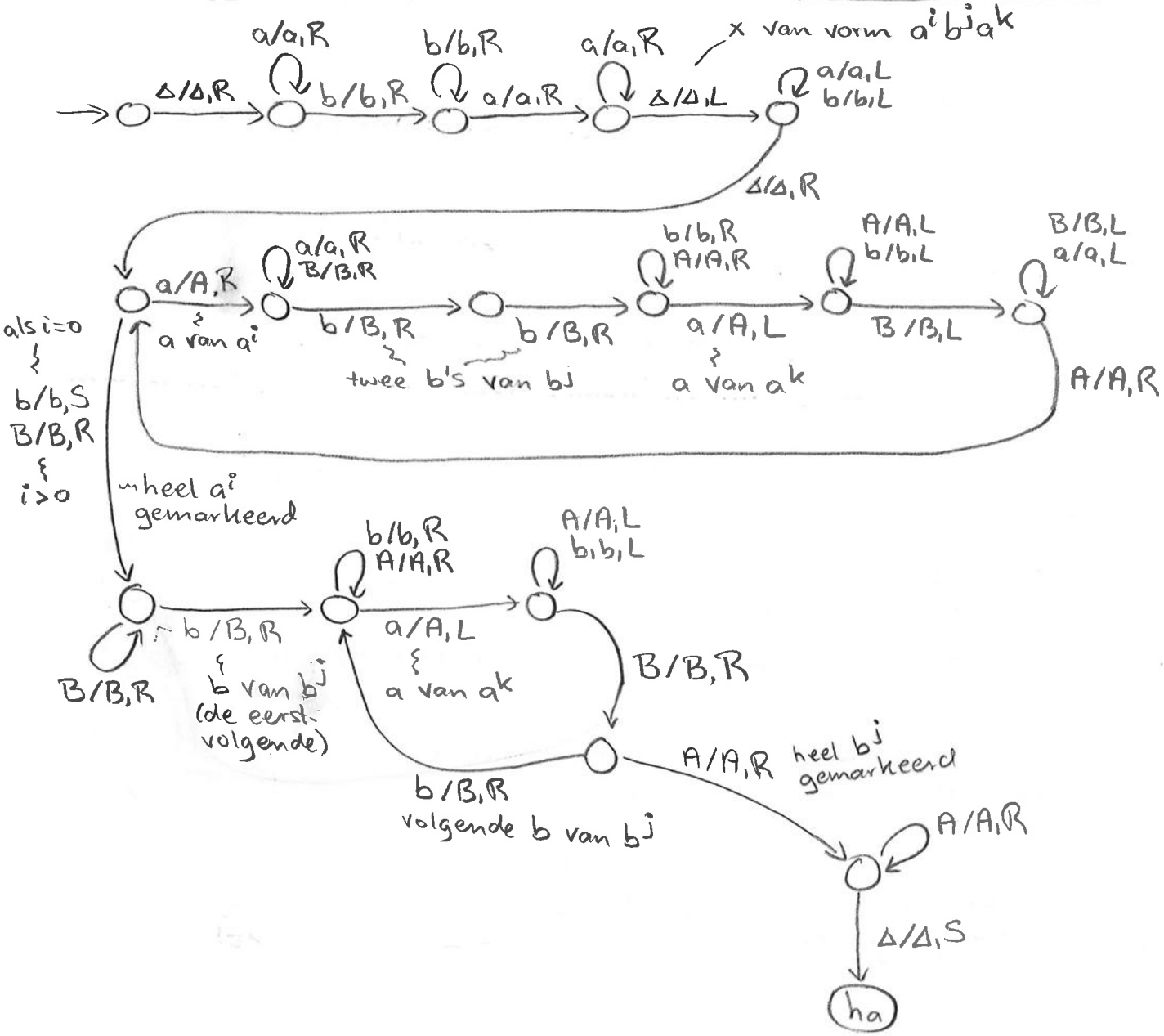
Zo nee, dan crasht de machine (want $j = 2i$, terwijl $j = i+k > 2i$ moet zijn).

Zo ja, dan markeert T_1 steeds (van links naar rechts) een resterende b van b^j, als B waarna hij (eveneens van links naar rechts) een resterende a van a^k markeert als A.

Als dit niet lukt (omdat k te klein is, dus $j > i+k$), crasht de machine.

Als dit wel lukt, dan controleert T_1 of alle a's van a^k gemarkeerd zijn als A. Zo nee, dan crasht de machine, zo ja, dan accepteert T_1 met de leeshop op de Δ na de compleet gemarkeerde invoer.

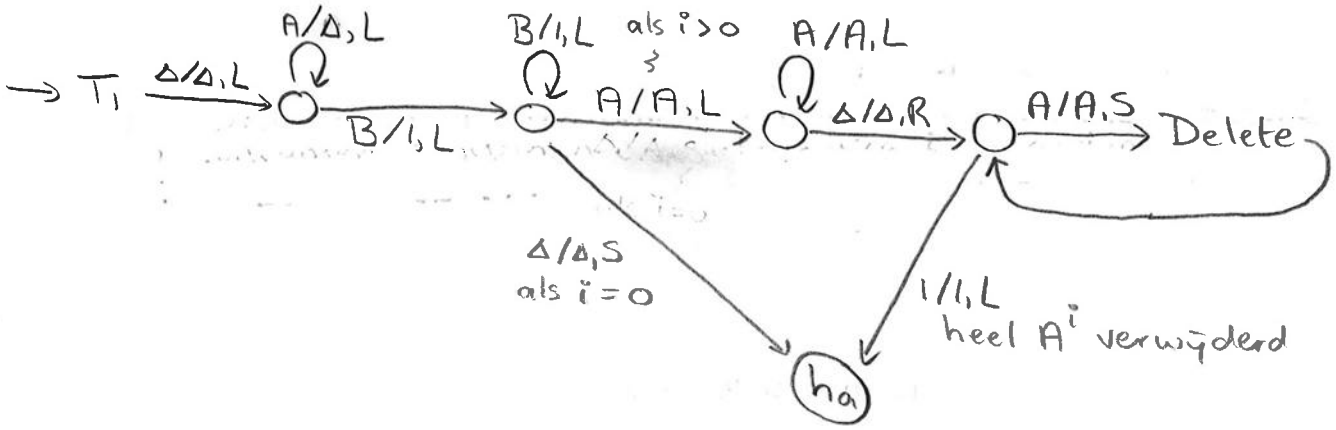
11.02



11.17

(c) T_2 roept eerst T_1 aan.
Als T_1 accepteert, dan zit de invoer x in L, en staat de leeskop op de Δ direct achter de gemarkeerde invoer.
T_2 gaat nu de achterste A's (van a^k) verwijderen, maakt van de B's (van $b^j = b^{i+k}$) 1'en voor de functiewaarde, en verwijdert de voorste A's (van a^i) met herhaaldelijk Delete.
Daarna accepteert T_2 met de leeskop op vakje 0.

11.24



11.29

2) T_2 krijgt een string $x \in \{a,b\}^*$ als invoer, en zet daar met Insert(a) en Insert(b) rond toestand q_1 een willekeurige string $y \in \{a,b\}^*$ vóór. De resulterende string yx wordt vervolgens aan T_1 gegeven.

Er geldt: $x \in L(T_2)$ $\Leftrightarrow$ als er een string $y \in \{a,b\}^*$ bestaat, zó dat $yx \in L(T_1)$ $\Leftrightarrow$ als x een suffix is van een element van L_1.

Dus $L(T_2) = \{x \in \{a,b\}^* \mid yx \in L_1 \text{ voor zekere } y \in \{a,b\}^*\}$

$\Leftrightarrow$ als er een berekening van T_2 voor invoer x bestaat die leidt tot acceptatie

11.39.

3a) G heeft startvariabele S en de volgende producties

$S \to A_1 B_1 S B_2 A_2$

A_1 wordt een a van a^i
B_1 een b van b^i
B_2 een b van b^k
A_2 een a van a^k
Hoog i en k allebei met 1 op

$S \to S B_2 A_2$

Hoog alleen k met 1 op

$S \to L_1 R_1 B_2 A_2$

L_1 gaat naar links lopen om $A_1^i B_1^i$ om te zetten in $a^i b^i$
R_1 gaat naar rechts lopen om $B_2^k A_2^k$ om te zetten in $b^k a^k$.
Hoog k nog met 1 op, om $i < k$ af te dwingen.

$B_1A_1 \to A_1B_1$	zet A_1 en B_1 in goede volgorde
$A_2B_2 \to B_2A_2$	zet B_2 en A_2 in goede volgorde
$B_1L_1 \to L_1b$	L_1 loopt naar links en verandert B_1 in b
$A_1L_1 \to L_2a$	tot L_1 bij A_1 komt. Daar maakt hij een a van, maar verandert zelf in L_2, om te voorkomen dat er nu nog B_1's in b worden veranderd (om volgorde $A_1^iB_1^i$ af te dwingen)
$A_1L_2 \to L_2a$	L_2 loopt verder naar links en verandert A_1 in a
$L_1 \to \Lambda$	L_1 (als $i=0$) en L_2 hebben hun werk gedaan en verdwijnen.
$L_2 \to \Lambda$	
$R_1B_2 \to bR_1$	analoog aan werking L_1 en L_2. (omdat $k>0$ moet zijn, is $R_1 \to \Lambda$ niet nodig)
$R_1A_2 \to aR_2$	
$R_2A_2 \to aR_2$	
$R_2 \to \Lambda$	

11.52

(b)

$$\underline{S} \Rightarrow A_1B_1\underline{S}B_2A_2 \Rightarrow A_1B_1L_1R_1B_2\underline{A_2B_2}A_2 \Rightarrow A_1\underline{B_1L_1}R_1B_2B_2A_2A_2$$

$$\Rightarrow \underline{A_1L_1}bR_1B_2B_2A_2A_2 \Rightarrow \underline{L_2}abR_1B_2B_2A_2A_2 \Rightarrow abR_1\underline{B_2B_2}A_2A_2$$

$$\Rightarrow^* abbbR_1\underline{A_2A_2} \Rightarrow abbbaR_2\underline{A_2} \Rightarrow abbbaaR_2 \Rightarrow abbbaa$$

11.58

4(a) We noemen een taal L recursief opsombaar, als er een Turingmachine T bestaat, zódat $L = L(T)$
(ofwel: T accepteert precies alle strings in L)

12.00

We noemen een taal L recursief, als er een Turingmachine T bestaat die de karakteristieke functie χ_L van L berekent.

12.01

(b) (i) is waar.
(ii) is waar.
(iii) is niet waar.

12.03.

5) (a)

Neem:

P_1 : gegeven een Turingmachine T_1, accepteert T_1 de invoer Λ binnen tien stappen ?

P_2 : gegeven een Turingmachine T_2, accepteert T_2 de invoer Λ?

(b)

We tonen aan dat $P_1 \leq P_2$

instantie: T_1 T_2

We moeten de willekeurige TM T_1 ombouwen naar een TM T_2 zó dat T_1 is ja-instantie van $P_1 \iff T_2$ is ja-instantie van P_2.

TM T_2 doet in principe hetzelfde als T_1, maar houdt in de toestanden ook bij hoeveel stappen in de berekening zijn gedaan (tot maximaal tien stappen), door van elke toestand in T_1 tien kopies te maken. Wanneer T_2 tien stappen voor zijn invoer heeft uitgevoerd, gaat hij naar hr. en nog niet is ge-halt.

Er geldt nu:

T_1 is ja-instantie van $P_1 \iff$ T_1 accepteert de invoer Λ binnen tien stappen $\Rightarrow$ T_2 accepteert de invoer Λ (eveneens binnen tien stappen) $\iff$ T_2 is ja-instantie van P_2.

T_1 is nee-instantie van $P_1 \iff$ T_1 accepteert de invoer Λ niet, of pas in meer dan tien stappen $\Rightarrow$ T_2 accepteert de invoer Λ niet (want na tien stappen verwerpt T_2 sowieso) $\iff$ T_2 is nee-instantie van P_2

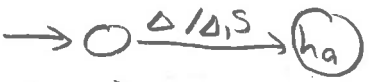
We kunnen T_2 inderdaad op algoritmische wijze uit T_1 construeren. Inderdaad is aan alle eisen van een reductie $P_1 \leq P_2$ voldaan.

Er geldt niet dat $P_2 \leq P_1$.
Immers, P_2 is een bekend niet-beslisbaar beslissingsprobleem. Als $P_2 \leq P_1$, zou ook P_1 niet beslisbaar zijn.
Echter, P_1 is beslisbaar, simpelweg door de instantie T_1 maximaal tien stappen te simuleren voor de invoer Λ. Als T_1 daarbij ha bereikt, is het een ja-instantie, en anders een nee-instantie.

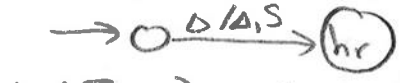
12.24.

6) We moeten aantonen dat de eigenschap "T accepteert oneindig veel strings" een niet-triviale taaleigenschap is.

* Sowieso gaat het om een eigenschap van een (één) Turingmachine.
* Het is een taaleigenschap.:
  als immers $L(T_1) = L(T_2)$ voor twee Turingmachines T_1 en T_2, dan accepteren zij dezelfde strings. Als dat er oneindig veel zijn bij T_1, zijn dat er natuurlijk ook oneindig veel bij T_2 en vice versa.
* Het is een niet-triviale eigenschap.
  - er bestaat immers (minstens) één Turingmachine die de eigenschap wel heeft, b.v.

    T_{ja}: 

    want $L(T_{ja}) = \Sigma^*$ en dat bevat oneindig veel strings
  - er bestaat ook (minstens één Turingmachine die de eigenschap niet heeft, b.v.

    T_{nee}: 

    want $L(T_{nee}) = \emptyset$ en dat bevat nul strings.

12.32