| Uitwerking tentamen Computability, vrijdag 26 maart 2021 |

21.02
1(a) j moet dus groter zijn dan i en dan k

De eerste zes elementen van L in de canonieke volgorde zijn:

b	bb	abb	bba	bbb	abba
i=k=0, j=1	i=k=0, j=2	i=1, j=2, k=0	i=0, j=2, k=1	i=k=0, j=3	i=k=1, j=2

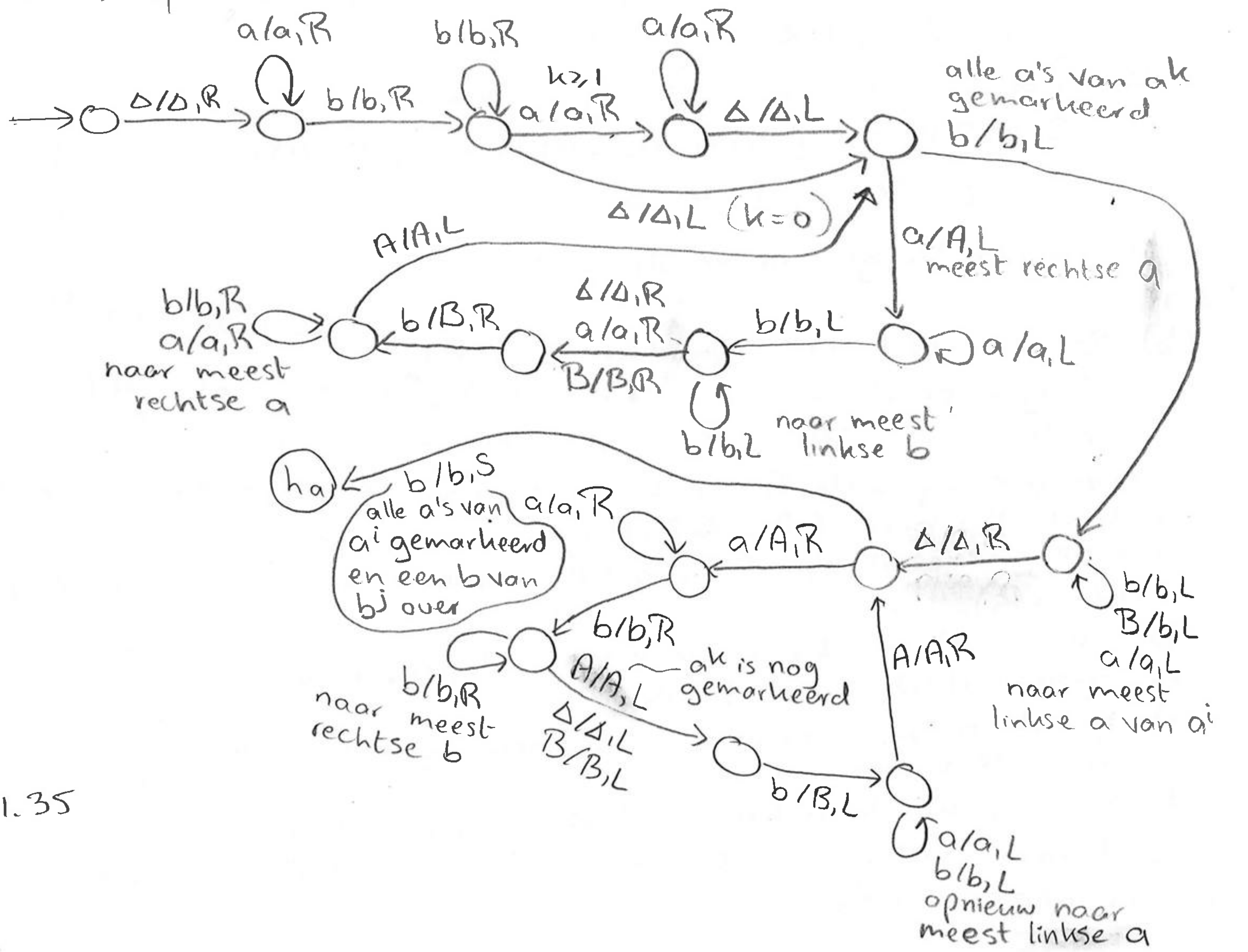
En vervolgens abbb, bbba, bbbb, aabbb, abbba, abbbb, bbbaa, bbbba, bbbbb

21.08
(b) T controleert eerst of de letters in de juiste volgorde staan: 0 of meer a's, 1 of meer b's, 0 of meer a's.

Daarna controleert T of $k < j$, door steeds de meest rechtse a van a^k en de meest linkse b van b^j te markeren als hoofdletter A, resp. B. Als alle a's van a^k gemarkeerd zijn, moet er nog minstens één b over zijn.

Vervolgens worden de B's van b^j gedemarkeerd, waarna T gaat controleren of $i < j$. T doet dit door steeds de meest linkse a van a^i en de meest rechtse b van b^j te markeren als hoofdletter A, resp. B. Als alle a's van a^i gemarkeerd zijn, moet er nog minstens één b over zijn. Als dat het geval is, kan T accepteren

21.16 / 21.17



21.35

2) Een woord x zit in L^* als x te schrijven is als $x = x_1 x_2 \dots x_k$ voor zekere $k \geq 0$, met elke $x_i \in L$, en ook is $x_i \neq \Lambda$ te 'kiezen'.
Zolang de resterende x nog niet Λ is, doet T_2 het volgende:
T_2 insert op een willekeurige plek in x (vóór de eerste, voor de tweede, .., voor de laatste letter van x) een Δ in x, om op die manier feitelijk x_k af te splitsen van x. Vervolgens wordt T_1 aangeroepen, om te testen of $x_k \in L$. T_1 wordt hiertoe zó aangepast dat hij niet links van de tape kan lopen, op x_{k-1} en dat hij het laatste gebruikte vakje op de tape bijhoudt.
Als T_1 x_k accepteert, wordt de tape van x_k' (na x_{k-1}, en tot 'het eind van de tape') schoongeveegd, waarna T_2 een nieuwe Δ in de resterende string x insert, om zo x_{k-1} af te splitsen, enzovoort.
Als x uiteindelijk Λ geworden is, en alleen dan, accepteert T_2

21.53

3) (a) Kortste string die door G gegenereerd wordt, is Λ, via

$$\underline{S} \Rightarrow \underline{T}R \Rightarrow \underline{LR} \Rightarrow \Lambda$$

Vervolgens krijgen we de string 1, via

$$\underline{S} \Rightarrow \underline{T}R \Rightarrow \underline{T}ABR \Rightarrow L\underline{AB}R \Rightarrow LB1\underline{AR} \Rightarrow \underline{LB}1R$$
$$\underline{L1}R \Rightarrow 1\underline{LR} \Rightarrow 1$$

Vervolgens krijgen we de string 111

22.01

(b) S genereert TR (productie $S \to TR$)
T genereert 0 meer substrings AB, (productie $T \to TAB$) en sluit af met L, (productie $T \to L$)
zodat we $L(AB)^k R$ krijgen voor zekere $k \geq 0$
Vervolgens lopen de A's naar rechts, naar R over de B's heen ($AB \to B1A$), over 1'en heen die gegenereerd zijn ($A1 \to 1A$), waarna de A's bij R verdwijnen ($AR \to R$).
De voorste A moet over k B's heen lopen naar R toe, en genereert daarmee k 1'en.
De tweede A moet over $k-1$ B's heen lopen naar R toe, en genereert daarmee $k-1$ 1'en
Enzovoort, tot de achterste A die nog één 1 genereert.
In totaal worden zo $k + (k-1) + (k-2) + \dots + 1 = \frac{1}{2}k(k+1)$ 1'en gegenereerd.

Ten slotte loopt L naar rechts, naar R toe, over de 1'en ($L1 \to 1L$), onderwijl de B's opruimend ($LB \to L$).
Bij R aangekomen verdwijnen L en R ($LR \to \Lambda$), waarna de $\frac{1}{2}k(k+1)$ 1'en overblijven.
Ofwel, G genereert $\{ 1^{\frac{1}{2}k(k+1)} \mid k \geq 0 \}$

22.14

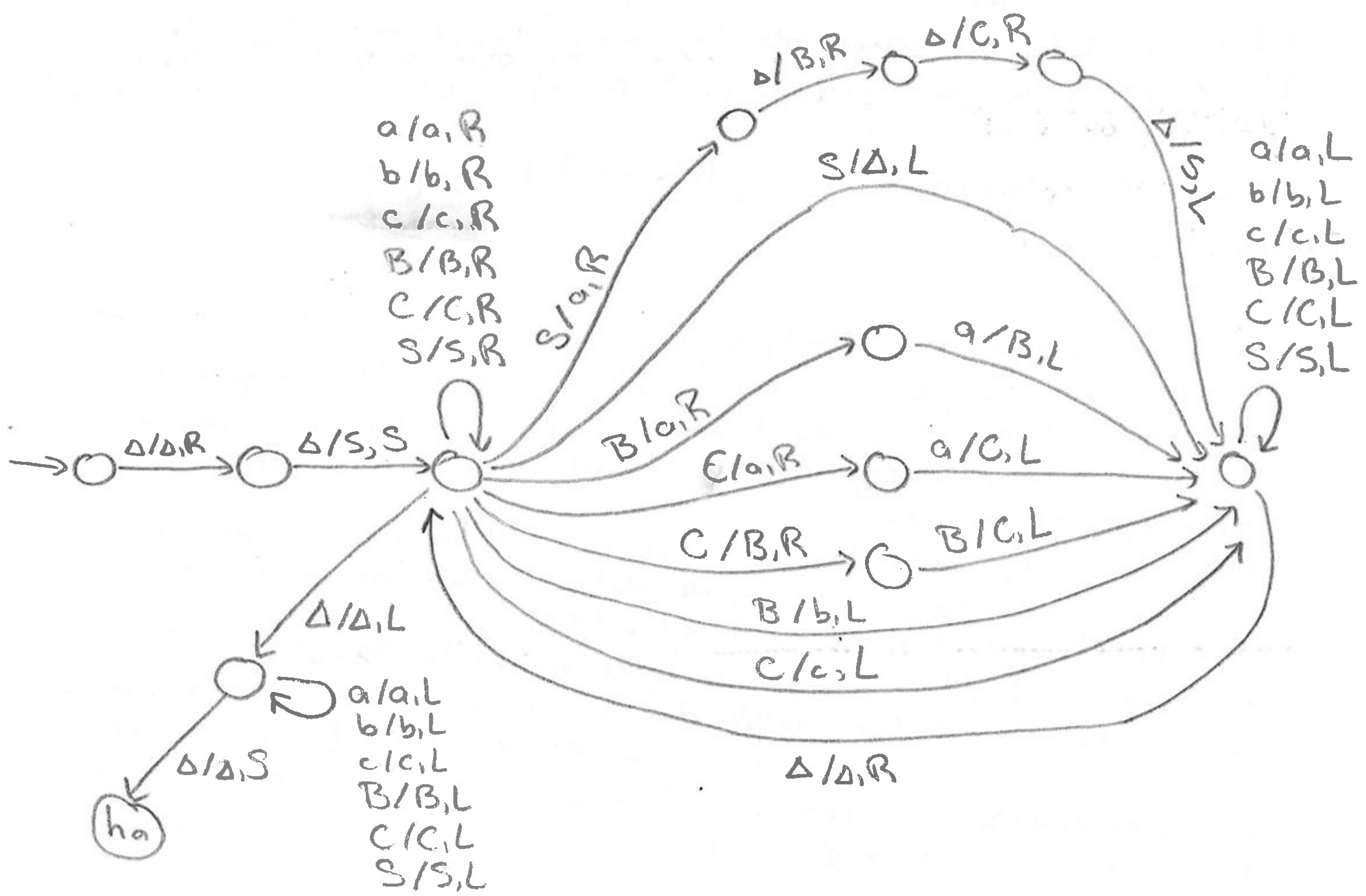
22.17

4 (a) Als invoer $x \in L(G)$, dan bestaat er een afleiding $S \Rightarrow^* x$ in G voor x. Dan is het mogelijk dat in stap 2 precies deze afleiding gesimuleerd wordt, zodat, achter de invoer x nog een keer x gegenereerd wordt, waarna in stap 3 geaccepteerd wordt.
Als $x \in L(G)$, bestaat er dus een berekening in T voor invoer x die leidt tot ha. Volgens de definitie van acceptatie door een NTM betekent dit dat $x \in L(T)$.
Als $x \notin L(G)$, dan bestaat er geen afleiding $S \Rightarrow^* x$ in G voor x. Dan is het ook niet mogelijk dat in stap 2 precies x gegenereerd wordt, waarna T in stap 3 zou accepteren. Er is dan dus geen enkele berekening van T voor invoer x die leidt tot ha.
Dat betekent dat $x \notin L(T)$
Ofwel: $x \in L(G) \iff x \in L(T)$,
ofwel: $L(T) = L(G)$.

22.28

(b) T_2 zet allereerst de startvariabele S op de tape, en daarmee de eerste string in elke afleiding in G.
Vervolgens loopt T_2 herhaaldelijk van links naar rechts naar een willekeurige positie in de tot nu toe gegenereerde string.
Daar aangekomen probeert T_2 een willekeurige productie uit G toe te passen, waarna T_2 weer teruggaat naar links.
Als T_2 bij het van links naar rechts lopen 'de string uitloopt', (bijvoorbeeld als er geen productie meer toe te passen is), stopt de simulatie, en zet T_2 zijn leeskop terug naar links

22.36



2245

5) a) Aan te tonen: AcceptsAtLeast-$\{x\} \leq$ AcceptsAtLeast-$\{y,z\}$

instantie is TM T_1 TM T_2

Laat T_1 een willekeurige instantie van AcceptsAtLeast-$\{x\}$.
Dan moeten we een TM T_2 construeren, zódat

T_1 accepteert tenminste x $\iff$ T_2 accepteert tenminste y en z.

We construeren de volgende TM T_2:

EraseInput $\rightarrow$ Write (x) $\rightarrow$ T_1

In woorden: T_2 veegt zijn invoer van de tape, schrijft daar de vaste string x voor terug, en roept dan T_1 aan.
Er geldt:

T_1 is ja-instantie van AcceptsAtLeast-$\{x\}$ $\iff$
T_1 accepteert tenminste x $\implies$
T_2 (die zijn invoer wegveegt en vervolgens T_1 op x uitvoert) accepteert elke invoer $\implies$ T_2 accepteert ook y en z $\iff$
T_2 is ja-instantie van AcceptsAtLeast-$\{y,z\}$

T_1 is nee-instantie van AcceptsAtLeast-$\{x\}$ $\Longrightarrow$
T_1 accepteert invoer x niet (de enige string in $\{x\}$) $\Longrightarrow$
T_2 accepteert geen enkele invoer $\Longrightarrow$
T_2 accepteert ook y en z niet $\Longrightarrow$
T_2 is nee-instantie van AcceptsAtLeast-$\{y,z\}$.

Uiteraard is de constructie van T_2 uit T_1 (met een vaste string x) algoritmisch uit te voeren.
Daarmee is aan alle eisen van een reductie voldaan:

$$\text{AcceptsAtLeast-}\{x\} \leq \text{AcceptsAtLeast-}\{y,z\}$$

23.01

(b) Voor een reductie van AcceptsAtLeast-$\{x,y\}$ naar AcceptsAtLeast-$\emptyset$ moeten we elke ja-instantie van AcceptsAtLeast-$\{x,y\}$ omzetten in een ja-instantie van AcceptsAtLeast-$\emptyset$, en elke nee-instantie in een nee-instantie.
Probleem is echter dat er wel nee-instanties van AcceptsAtLeast-$\{x,y\}$ zijn (b.v. de TM: $\rightarrow \bigcirc \xrightarrow{\Delta/\Delta,S} (h_r)$)
maar geen nee-instanties van AcceptsAtLeast-$\emptyset$.
Immers, elke TM T_2 (met invoeralfabet Σ) accepteert elke string in $\emptyset$, en is dus een ja-instantie.

23.08.