Uitwerking tentamen Fundamentele Informatica 3, maandag 29 juni 2020

10.57

2 (a)

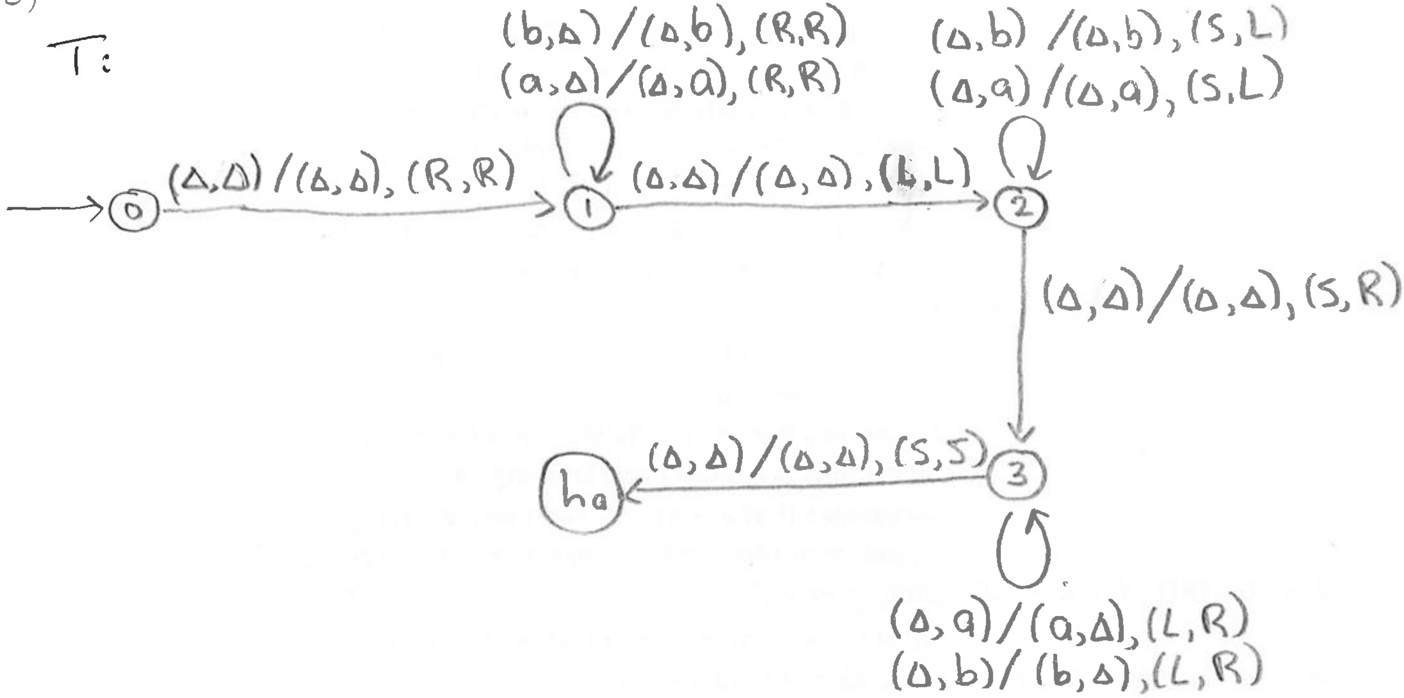
Laat T een 2-tapes Turingmachine zijn met starttoestand q_0.
In zijn initiële configuratie voor invoer x

* bevindt T zich in toestand q_0
* met beide leeskoppen op vakje 0 van de respectievelijke tape
* met op tape 1: ΔxΔΔ...
* met op tape 2: ΔΔΔΔ... (leeg dus)

11.02

(b)

T:



11.11

(c)

In toestand 1 loopt T over zijn invoer x naar rechts, en verplaatst die invoer van tape 1 naar tape 2.
In toestand 2 staat T met zijn leeskop op tape 1 op de Δ overeenkomend met de laatste letter van x, terwijl hij op tape 2 terugloopt naar links, naar het begin van x.
In toestand 3 verplaatst T de invoer x weer van tape 2 naar tape 1. Omdat hij daarbij op tape 1 naar links loopt, terwijl hij op tape 2 naar rechts loopt, komt x in omgekeerde vorm op tape 1 terecht.
Wanneer x accepteert staat zijn leeskop op tape 1 keurig op vakje 0, met x^r daarachter, terwijl zijn leeskop op tape 2 achter 'de weer verwijderde x' staat. Dat laatste maakt niet uit.

11.20/1

11.29

3(a) Geldige combinaties van i en j zijn

$(i,j) = (1,1), (2,2), (2,3), (3,3), (3,4), (3,5), (4,4), \dots$

$i+j+2i = 4 \quad 8 \quad 9 \quad 12 \quad 13 \quad 14 \quad 16$

De eerste vier elementen in de canonieke volgorde zijn dus
abaa, aabbaaaa, aabbbaaaa, aaabbbaaaaaa.

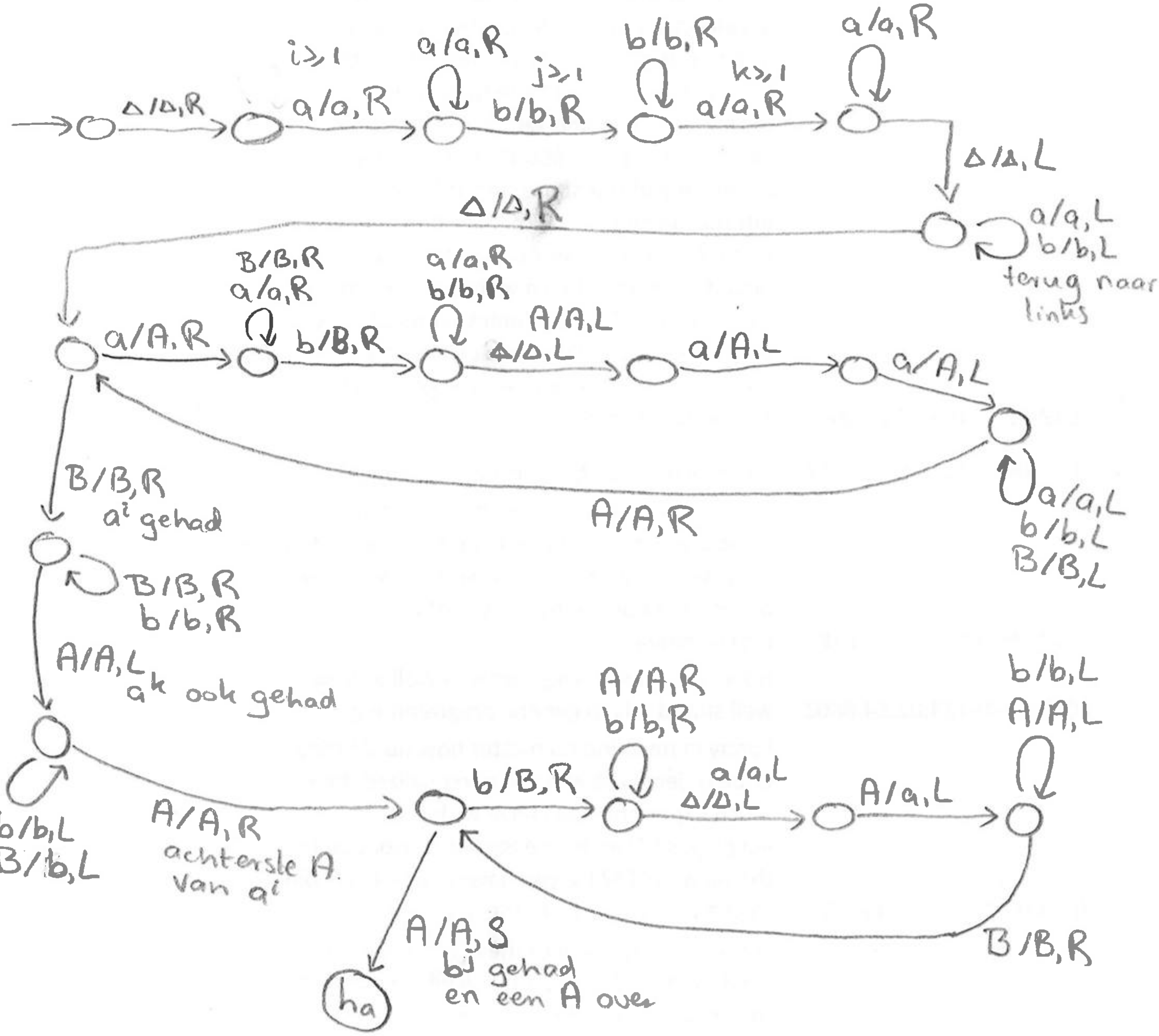
11.34

(b)

T_1 controleert eerst of zijn invoer van de vorm $a^i b^j a^k$ is met $i,j,k \geq 1$. Zo nee, dan verwerpt / crasht T_1.
Zo ja, dan kijkt T eerst of $i \leq j$ en of $k = 2i$.
Dat doet hij door steeds de eerste a van a^i, de eerste b van b^j en de laatste twee a's van a^k te veranderen in een hoofdletter. Tot de a's van a^i op zijn.
Als dit lukt, en na afloop zijn alle a's van a^k inderdaad veranderd in hoofdletters, moeten we alleen nog controleren dat $j < 2i$. Dat doen we door de B's van b^j weer te veranderen in b's, en vervolgens steeds de eerste b van b^j te veranderen in B en de laatste A van a^k weer in a. Tot de b's van b^j op zijn.
Als dit lukt, en na afloop is er nog minstens een A van a^k over, is $j < 2i$ en kunnen we accepteren.

11.48

(b)


Δ/Δ,R
i≥1
a/a,R
a/a,R
j≥1
b/b,R
b/b,R
k≥1
a/a,R
a/a,R
Δ/Δ,L
Δ/Δ,R
a/a,L
b/b,L
terug naar links
a/A,R
B/B,R
a/a,R
b/B,R
a/a,R
b/b,R
A/A,L
Δ/Δ,L
a/A,L
a/A,L
A/A,R
a/a,L
b/b,L
B/B,L
B/B,R
a^i gehad
B/B,R
b/b,R
A/A,L
a^k ook gehad
b/b,L
B/b,L
A/A,R
achterste A van a^i
b/B,R
A/A,R
b/b,R
a/a,L
Δ/Δ,L
A/a,L
b/b,L
A/A,L
B/B,R
A/A,S
b^j gehad en een A over
ha

11.59

12.13 Uitwerking tentamen Fundamentele Informatica 3, maandag 29 juni 2020

4(a)

We kiezen voor L_1

De grammatica G kent de volgende producties, met startvariabele S:

$S \to aBBSaa$ genereer een a voor a^i, twee b's voor b^j en twee a's voor a^{2i}

$S \to aBTaa$ genereer een a voor a^i, één b voor b^j en twee a's voor a^{2i}

$T \to aBTaa$ "

Door de overgang naar T dwingen we af dat $j < 2i$. Minstens één keer genereren we één b voor b^j en twee a's voor a^{2i}
Door alle drie de producties dwingen we af dat $i \leq j$, en dat we a^i links en a^{2i} rechts krijgen.

$T \to M$ klaar met genereren letters
Nu staan alleen de a's van a^i en de B's van b^j nog door elkaar, links van M.

$Ba \to aB$ B loopt naar rechts, naar M toe.

$BM \to Mb$ daar aangekomen wordt het b

$M \to \Lambda$ als alle B's over M heen gesprongen zijn en b geworden zijn, zijn a^i en b^j gesorteerd en kan M verdwijnen.

12.25 / 13.28

(b)

Een afleiding in G voor aabbbaaaa:

$$\underline{S} \Rightarrow aBB\underline{S}aa \Rightarrow aBBaB\underline{T}aaaa \Rightarrow a\underline{BBa}BMaaaa$$
$$\Rightarrow^2 aa\underline{BBBM}aaaa \Rightarrow^3 aa\underline{M}bbbaaaa \Rightarrow aabbbaaaa.$$

13.32

Uitwerking tentamen Fundamentele Informatica 3, maandag 29 juni 2020

12.36

5(a)

L_1 is recursief opsombaar, dus er bestaat een Turingmachine die L_1 accepteert. Laat T_1 zo'n Turingmachine zijn.

De Turingmachine T_2 voor L_2 werkt als volgt:
* controleer of zijn invoer x soms gelijk is aan x_1
* zo ja, accepteer
  zo nee, voer T_1 uit op invoer x.

T_2 accepteert dan x_1, en ook alle strings die T_1 accepteert, ofwel $\{x_1\} \cup L_1 = L_1 \cup \{x_1\} = L_2$.
L_2 is derhalve recursief opsombaar.

12.42/47

(b)

De Turingmachine T_3 voor L_3 werkt als volgt:
* controleer of zijn invoer x soms gelijk is aan $x_1, x_2, .., $ of x_n.
  Omdat dit eindig veel (eindige) strings zijn, is dit uit te voeren.
* als x inderdaad gelijk is aan $x_1, x_2, ...$ of x_n, verwerp
  anders, accepteer.

T_3 accepteert alle invoeren x behalve $x_1, x_2, .., x_n$, ofwel T_3 accepteert L_3.
L_3 is derhalve recursief opsombaar.

12.52

Als volgt:
* controleer of invoer x gelijk is aan x_1
* zo nee, controleer of invoer x gelijk is aan x_2
* zo nee, controleer of invoer x gelijk is aan x_3
* enzovoort

6 (a)

De stelling van Rice is rechtstreeks toepasbaar op Accepts-ab.
Dat is immers een beslissingsprobleem

* waarvan de instanties (losse) Turingmachines T_1 zijn
* waarvan de eigenschap "accepteert T_1 de string ab" een niet-triviale taaleigenschap is.

Immers:

- als T_1 de eigenschap heeft, en $L(T_1) = L(T_2)$, is kennelijk $ab \in L(T_1)$ en dus ook $ab \in L(T_2)$, zodat T_2 de eigenschap heeft (het is een taaleigenschap)
- de eigenschap is niet-triviaal, want de Turingmachine $\rightarrow \bigcirc \xrightarrow{\Delta/\Delta,S} (h_a)$ (die alles accepteert) heeft de eigenschap wel; de Turingmachine $\rightarrow \bigcirc \xrightarrow{\Delta/\Delta,S} (h_r)$ (die niets accepteert) heeft de eigenschap niet

13.01

(b) Accepts-ab is dus niet beslisbaar vanwege de stelling van Rice. Om aan te tonen dat ook Accepts-Prefix niet beslisbaar is, moeten we Accepts-ab reduceren naar Accepts-Prefix:

$$\begin{array}{ccc} \text{Accepts-ab} & \le & \text{Accepts-Prefix} \\ T_1 & & (T_2, x_2) \end{array}$$

Laat T_1 een willekeurige instantie van Accepts-ab zijn, dat wil zeggen: een Turingmachine met invoeralfabet $\{a, b\}$.
We kiezen nu $x_2 = \Lambda$ (de lege string), en construeren een Turingmachine T_2 die als volgt werkt: vervang de invoer door ab, en voer T_1 daar op uit.

Er geldt nu:

× als T_1 een ja-instantie van Accepts-ab is,
dan accepteert T_1 de string ab,
dan accepteert T_2 elke invoer,
dan accepteert T_2 ook de invoer Λ,
dan accepteert T_2 dus een prefix van $x_2 = \Lambda$,
dan is (T_2, x_2) een ja-instantie van Accepts-Prefix.

* als T_1 een nee-instantie van Accepts-ab is,
dan accepteert T_1 de string ab niet,
dan accepteert T_2 geen enkele invoer,
dan accepteert T_2 ook de invoer Λ niet,
dan accepteert T_2 geen enkele prefix van $x_2 = \Lambda$,
dan is (T_2, x_2) een nee-instantie van Accepts-Prefix.

De constructie van (T_2, x_2) uit T_1 is algoritmisch uit te voeren.
We hebben derhalve een geldige reductie.
Omdat Accepts-ab niet beslisbaar is, is ook Accepts-Prefix niet beslisbaar.

13.17