Uitwerking tentamen Fundamentele Informatica 1 (I&E), donderdag 7 januari 2016

23:20

1(a)

(i) $L_1^* = \{\Lambda, a, aa, aaa, \ldots\}$

$L_2^* = \{\Lambda, bb, bbbb, bbbbbb, \ldots\}$

De eerste vijf elementen in de canonieke volgorde van $L_1^* L_2^*$ zijn:

Λ, a, aa, bb, aaa En vervolgens: $abb, aaaa, aabb, bbbb, \ldots$

23:24

(ii) $L_1 L_2 = \{\Lambda, a, bb, abb\}$

De eerste vijf elementen in de canonieke volgorde van $(L_1 L_2)^*$ zijn:

$\Lambda, a, aa, bb, aaa.$ En vervolgens: $abb, bba, aaaa, aabb, \ldots$

23:26

(b) Bij de taal $(L_1 L_2)^*$ ben je gedwongen om elementen van L_1 en L_2 af te wisselen bij de opbouw van een woord $x \in (L_1 L_2)^*$:

$x = x_{1,1} x_{2,1} x_{1,2} x_{2,2} \cdots x_{1,k} x_{2,k}$, waarbij $x_{1,i} \in L_1$ en $x_{2,i} \in L_2$ voor $i = 1, \ldots, k$

Bij $L_1^* L_2^*$ kun je elementen van L_1 achter elkaar zetten en elementen van L_2 achter elkaar zetten. Als L_1 en L_2 niet Λ bevatten, is dat niet mogelijk bij $(L_1 L_2)^*$.

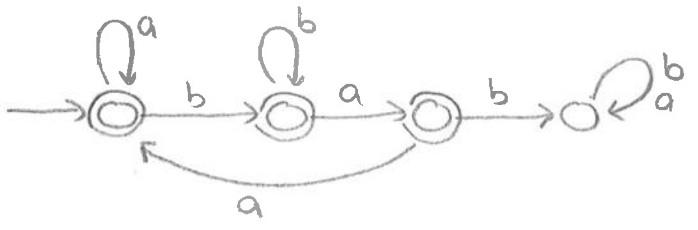
Neem daarom $L_1 = \{a\}$ en $L_2 = \{bb\}$. Dan zit $x = aabb$ wel in $L_1^* L_2^*$, maar niet in $(L_1 L_2)^*$.

Er geldt dus niet algemeen dat $L_1^* L_2^* \subseteq (L_1 L_2)^*$

23:34

2(a) We houden in de toestanden van M_1 bij hoe ver we zijn met het opbouwen van de 'verboden' substring bab:

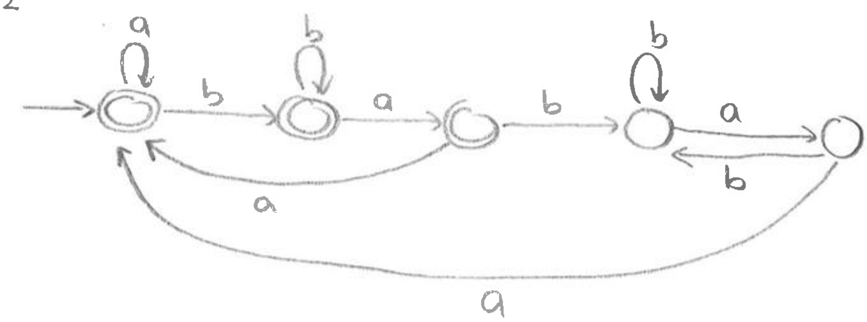
M_1



23:37

(b) We beginnen hetzelfde als bij M_1, maar bieden een ontsnappingsmogelijkheid vanuit de niet-accepterende toestand.

M_2



23:41

3) Stel dat $L_0 = \{ (ab)^i a^j \mid i > j \geq 1 \}$ wel door een eindige automaat geaccepteerd wordt, en dat het aantal toestanden van deze automaat n is.

Dan kiezen we $x = (ab)^{n+1} a^n$

Inderdaad is $x \in L_0$ (met $i = n+1 > j = n \geq 1$), en $|x| \geq n$.

Laat uvw een willekeurige opsplitsing van x zijn met $|uv| \leq n$ en $|v| > 0$. Dan bevat uv alleen letters uit $(ab)^{n+1}$, want uv is een prefix van x met lengte $\leq n$. Ook v bevat dus alleen letters uit $(ab)^{n+1}$. Bovendien maakt het laatste voorkomen van ab in $(ab)^{n+1}$ zeker geen deel uit van uv, opnieuw omdat $|uv| \leq n$.

Dit voorkomen zit dus in w en verandert niet als we v gaan oppompen of wegpompen.

We gaan v nu wegpompen, dat wil zeggen: we beschouwen de string uv^0w. Volgens het pomplemma zit uv^0w in L_0.

23:53
00:21
10:30

De string uv^0w eindigt met w, en dus met aba^n, want die suffix van x zat zeker nog in w.

Als uv^0w dan in L_0 moet zitten, en dus van de vorm $(ab)^i a^j$ moet zijn, moet $j = n$ zijn. Verder moet uv^0w vóór de suffix a^n minstens $n+1$ substrings ab bevatten. Dat laatste is echter niet mogelijk, want $x = uvw$ bevatte precies $n+1$ substrings ab, en daaruit hebben we v, met lengte $|v| > 0$, weggepompt.

Het 'linker deel' van uv^0w is dus te kort om nog $n+1$ substrings ab te bevatten. Misschien is uv^0w niet eens te schrijven als $(ab)^i a^j$ voor enige i, j. Als v oneven lengte heeft, hebben we door het wegpompen van v, in het linkerdeel namelijk twee a's of twee b's achter elkaar gekregen (of uv^0w begint met een b).

Hoe dan ook, uv^0w kan niet in L_0 zitten, wat in tegenspraak is met het pomplemma. Onze aanname dat L_0 door een eindige automaat geaccepteerd kan worden, moet dus onjuist zijn.

L_0 kan dus niet door een eindige automaat geaccepteerd worden.

10:43

4(a)

De eerste zes elementen in de taal die wordt beschreven door r_0 zijn:

Λ, a, aa, aaa, baa, bba.

En vervolgens: aaaa, abaa, abba, baaa, bbaa

10:46

(b) $r_1 \neq r_0$, want ~~bba~~ x= bba voldoet wel aan r_0, maar niet aan r_1
x=baa voldoet wel aan r_0, maar niet aan r_1

10:49

$r_2 \neq r_0$, want x=a voldoet wel aan r_0, maar niet aan r_2
x=aba voldoet wel aan r_2, maar niet aan r_0
x = aa voldoet wel aan r_0, maar niet aan r_2

10:50

$r_3 \neq r_0$, want x= babaa voldoet wel aan r_0, maar niet aan r_3

10:52

$r_4 \neq r_0$, want x = ba voldoet wel aan r_4, maar niet aan r_0.
x = babba voldoet wel aan r_4, maar niet aan r_0
x = baba voldoet wel aan r_4, maar niet aan r_0

10:54

~~10:56~~ 11:00

5 (a)

G_1 heeft S als enige variabele (en dus ook startvariabele) en producties

$$S \rightarrow aaSb \mid \Lambda$$

11:02

(b) In $L(G_2)$ mogen we per b maximaal twee a's genereren, maar we moeten minstens één keer minder dan twee a's voor een b genereren. Dat doen we met startvariabele S en de volgende producties

$$S \rightarrow aaSb \mid aTb \mid Tb$$

Genereer twee a's voor een b, of minder, maar in dat laatste geval gaan we verder met T, omdat we dan zouden mogen stoppen.

$$T \rightarrow aTb \mid Tb \mid \Lambda$$

Genereer nog meer b's (elk met minder dan twee a's) of stop

11:08

Eventueel zouden we nog de productie $T \rightarrow aaTb$ toe kunnen voegen, maar die is niet nodig. Als we twee a's willen genereren voor een extra b kunnen we dat al bij S doen

11:11

Alternatief: $S \rightarrow aaSb \mid aSb \mid Sb \mid ab \mid b$

11:19

6 a)

Een context-vrije grammatica $G = (V, \Sigma, S, P)$ is in Chomsky normaalvorm als elke productie in P van een van de volgende vormen is:

$A \rightarrow BC$ met $A, B, C \in V$ (variabelen, niet per se verschillend)

$A \rightarrow \sigma$ met $A \in V$, $\sigma \in \Sigma$

11:22

b) * Allereerst gaan we de Λ-producties wegwerken.
Daartoe bepalen we de nullable variabelen.
Dat zijn Y (vanwege $Y \rightarrow \Lambda$) en X (vanwege $X \rightarrow YY$).
Nu gaan we voor alle producties met een of meer nullable variabelen in de rechterkant producties toevoegen waarin de nullable variabele(n) is weggelaten

11:25

11:~~28~~35

Dat levert (alles bij elkaar) de volgende producties op

$S \rightarrow XYZ \mid YZ \mid XZ \mid Z$
$X \rightarrow abXa \mid aba \mid YY \mid Y \mid \cancel{\Lambda}$
$Y \rightarrow aY \mid a \mid \cancel{\Lambda}$
$Z \rightarrow bZ \mid b$

Hieruit verwijderen we vervolgens de Λ producties (doorgestreept)

11:39

* Vervolgens gaan we de unit-producties wegwerken.
Daartoe bepalen we voor elke variabele A de A-derivable variabelen

S-derivable : Z
X-derivable : Y
Y-derivable : –
Z-derivable : –

Voor elke A-derivable variabele B voegen we producties $A \rightarrow \beta$ toe voor elke bestaande productie $B \rightarrow \beta$
Dat levert (alles bij elkaar) de volgende producties op:

$S \rightarrow XYZ \mid YZ \mid XZ \mid \cancel{Z} \mid bZ \mid b$
$X \rightarrow abXa \mid aba \mid YY \mid \cancel{Y} \mid aY \mid a$
$Y \rightarrow aY \mid a$
$Z \rightarrow bZ \mid b$

Hier uit verwijderen we vervolgens de unit-producties (doorgestreept)

11:46

11:49

* we vervangen nu terminalen a en/of b die niet 'alleen' staan in de rechterkant van een productie door een speciale variabele X_a, respectievelijk X_b, en voegen producties toe die daarvan weer een terminaal maken.

Dat levert (alles bij elkaar) de volgende producties op:

$$
\begin{aligned}
S &\to XYZ \mid YZ \mid XZ \mid X_bZ \mid b \\
X &\to X_aX_bXX_a \mid X_aX_bX_a \mid YY \mid X_aY \mid a \\
Y &\to X_aY \mid a \\
Z &\to X_bZ \mid b \\
X_a &\to a \\
X_b &\to b
\end{aligned}
$$

11:54

* Ten slotte splitsen we producties wier rechterkant te lang is, op met behulp van hulpvariabelen $A_1, A_2, \ldots$

Dat levert (alles bij elkaar) de volgende producties voor G' op:

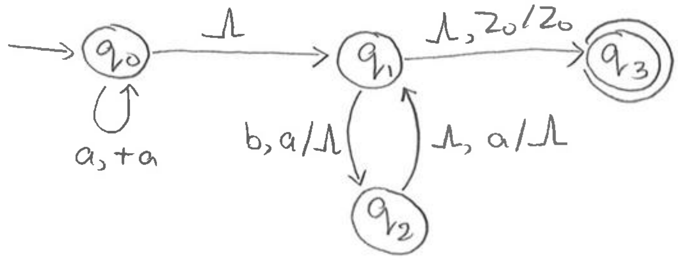
$$
\begin{aligned}
S &\to XA_1 \mid YZ \mid XZ \mid X_bZ \mid b \\
A_1 &\to YZ \\
X &\to X_aA_2 \mid X_aA_4 \mid YY \mid X_aY \mid a \\
A_2 &\to X_bA_3 \\
A_3 &\to XX_a \\
A_4 &\to X_bX_a \\
Y &\to X_aY \mid a \\
Z &\to X_bZ \mid b \\
X_a &\to a \\
X_b &\to b
\end{aligned}
$$

Met startvariabele S.

11:59.

7 (a)

M_1:



12:02

(b)

Een succesvolle berekening voor invoer $x = aaaabb$ in M_1:

$(q_0, aaaabb, Z_0) \vdash (q_0, aaabb, aZ_0) \vdash (q_0, aabb, aaZ_0) \vdash$
$(q_0, abb, aaaZ_0) \vdash (q_0, bb, aaaaZ_0) \vdash (q_1, bb, aaaaZ_0) \vdash$
$(q_2, b, aaaZ_0) \vdash (q_1, b, aaZ_0) \vdash (q_2, \Lambda, aZ_0) \vdash (q_1, \Lambda, Z_0)$
$\vdash (q_3, \Lambda, Z_0)$.

12:06

(c)

De eerste vier elementen in de canonieke volgorde van L_2 zijn

	i_1	j_1	i_2	j_2
abab	1	1	1	1
aabab	2	1	1	1
ababb	1	1	1	2
aaabab	3	1	1	1

En daarna

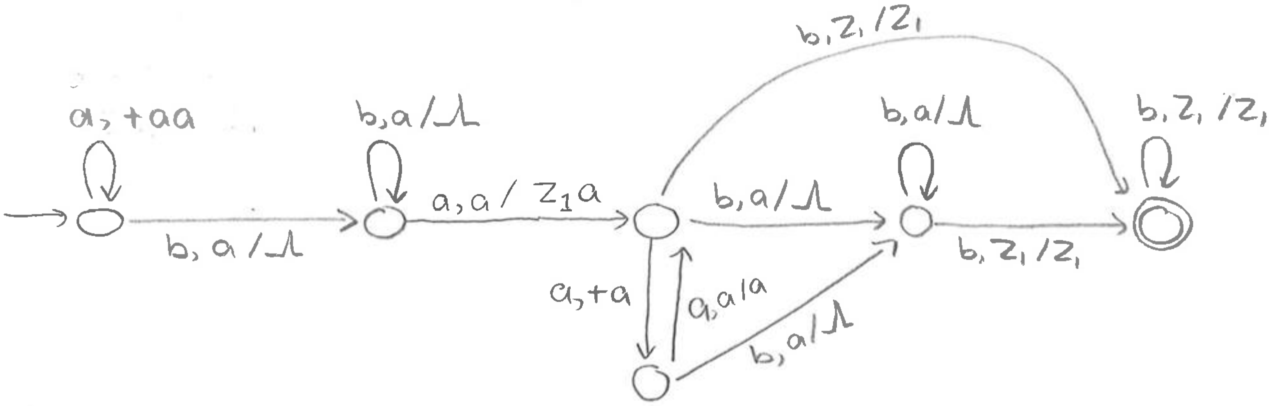
	i_1	j_1	i_2	j_2
aababb	2	1	1	2
aabbab	2	2	1	1
abaabb	1	1	2	2
ababbb	1	1	1	3

12:11

(d) We willen $2i_1 > j_1$ hebben, dus het aantal b's moet daar minder zijn dan twee keer het aantal a's. We zetten daarom per a die we lezen twee a's op de stapel, en we moeten er bij het afstapelen minstens één overhouden.

Voor vervolg uitleg, zie blz. 7

M_2



Het kan met nog een toestand minder, zie blz 8

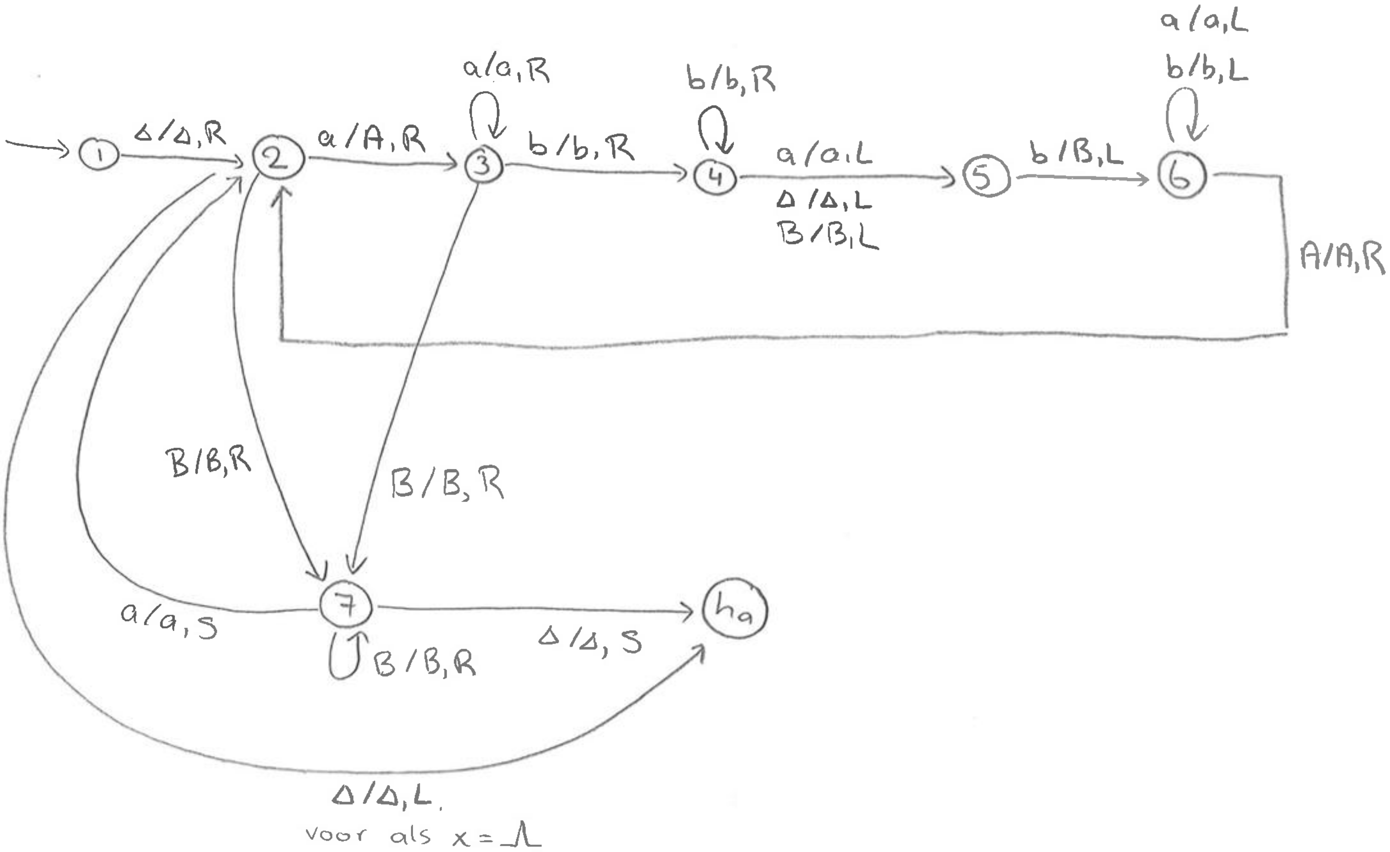
Vervolgens willen we $2j_2 > i_2$ hebben, dus het aantal a's moet daar minder zijn dan twee keer het aantal b's.
We zetten daarom per twee a's die we lezen één a op de stapel,

12:24 Bij elke b halen we vervolgens een a van de stapel.
12:48 Als de stapel 'leeg' is, moeten we nog minstens een b extra lezen.

Pas bij de tweede a zetten we een a op de stapel;
dan gaat het ook automatisch goed, als het aantal a's in de invoer hier oneven is

13:02
13:05

8 (a)



In de lus 2-3-4-5-6 wordt een substring $a^{i_k}b^{j_k}$ afgehandeld.

Iedere iteratie wordt er een a links en een b rechts gemarkeerd (hoofdletter gemaakt).

13:13
13:24
We springen uit de lus, als we geen b meer vinden,
in 2 (als $i_k = j_k$) of in 3 (als $i_k > j_k$).
In toestand 7 lopen we dan naar de volgende substring $a^{i_k}b^{j_k}$

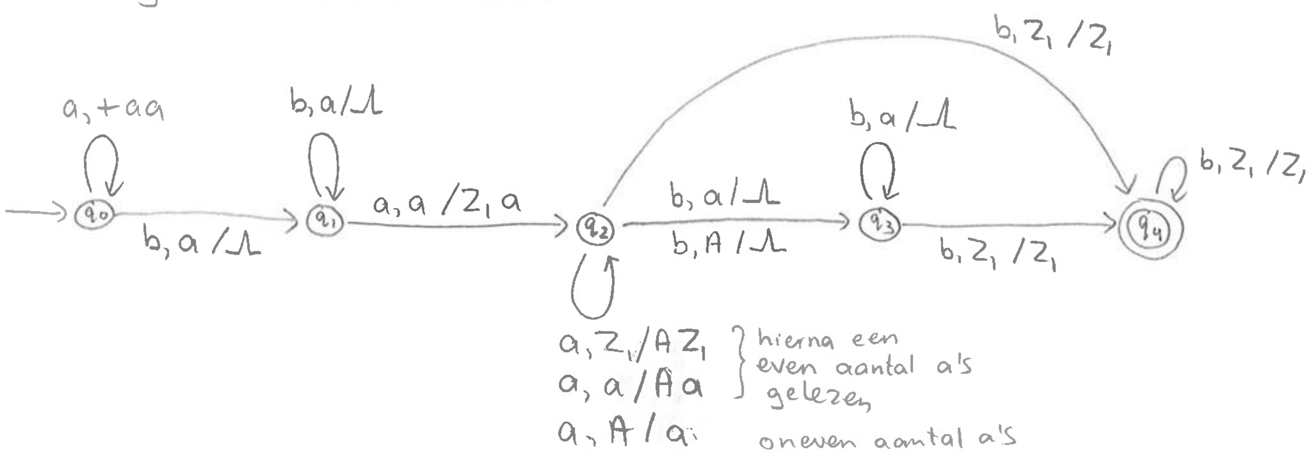
13:27. Als we die niet vinden (in plaats van een a zien we een Δ, wat betekent dat we de hele string x hebben gehad), dan accepteren we.

(b)

De eenvoudigste x die in aanmerking komt, is $x = a$ (met $m = 1$, $i_1 = 1$, $j_1 = 0$).
Na het markeren van de a, staan we in toestand 3 meteen op de Δ na de invoer. Er wordt daar echter (minstens) een b verwacht, omdat $jk \geq 1$ moet zijn. Derhalve crasht de Turing machine daar.

13:31

7 (d) Nog een toestand minder:



Hiermee geldt in q_2:

1 a gelezen	Z_1 op stapel	1 b nodig
2 a's gelezen	AZ_1 op stapel	2 b's nodig
3 a's gelezen	aZ_1 op stapel	2 b's nodig
4 a's gelezen	AaZ_1 op stapel	3 b's nodig
5 a's gelezen	aaZ_1 op stapel	3 b's nodig.