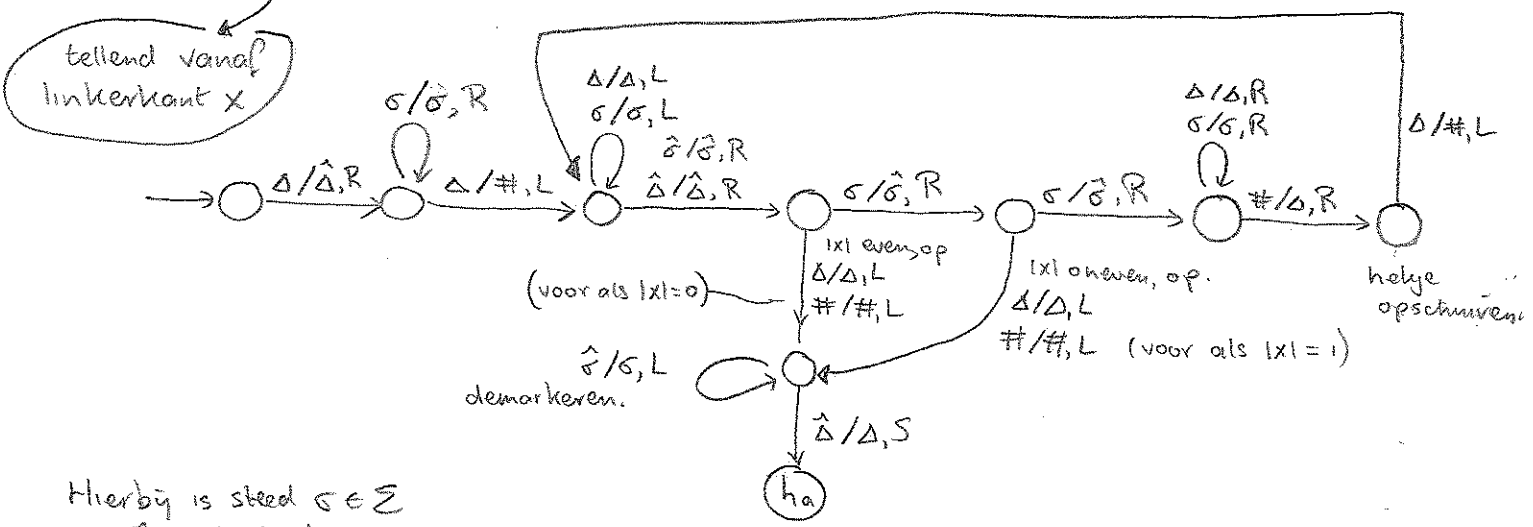
10:55 **Uitwerking tentamen Fundamentele Informatica 3, woensdag 5 juni 2013**

1(a) Het hekje moet achter x op de tape komen, met $\lfloor \frac{1}{2}|x| \rfloor$ Δ's er tussen, want x_4 bezet posities $1 \dots |x|$.

(zelf?)

In eerste instantie zetten we # direct achter x, en voor elke twee letters van x (die letters markeren we met $\hat{}$) voegen we een Δ toe.



Hierbij is steeds $\sigma \in \Sigma$ en $\hat{\sigma}$ is het daarmee corresponderende gemarkeerde symbool.

11:10

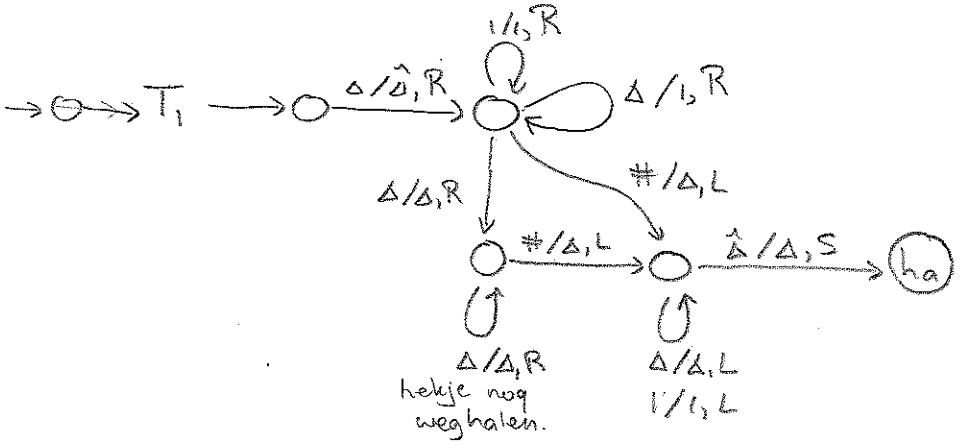
(b)

T_2 maakt gebruik van T_1

Allereerst roepen we T_1 aan om een # op positie $\lfloor \frac{3}{2}|x| \rfloor + 1$ te zetten.

Het getal y ontstaat dan door een willekeurig aantal Δ's tussen x en het # te vervangen door een 1.

Aan het eind moeten we nog wel het # weghalen:



11:18

Uitwerking tentamen Fundamentele Informatica 3 woensdag 5 juni 2013

2 (a)

We genereren eerst de random getallen $n_1 - 1$ en n_2, met A's, respectievelijk B's. Voor elke A in $n_2 - 1$ kopiëren we n_2 door een letter C van links naar rechts door n_2 heen te sturen, die bij elke B die hij tegenkomt een symbool D te produceren. Deze D loopt ook naar rechts, en verandert rechts in een 1. Om de rechterkant van de string te herkennen, zetten we daar een variabele R.

Op deze manier vermenigvuldigen we feitelijk n_1 en n_2

G heeft startsymbool S en producties:

$S \to TR$
$T \to AT \mid TB \mid ABB$
$AB \to CB$
$CB \to BCD$
$DB \to BD$
$DR \to R1$
$CR \to R$
$B \to 1$
$R \to \lambda$

S: zet R neer aan rechterkant
T: genereer $A^{n_1-1}B^{n_2}$ voor willekeurige $n_1, n_2 \geq 2$.
C: boodschapper die door n_2 loopt om n_2 te kopiëren
D: onderdeel van kopie van n_2
R: rechterkant van (niet-terminaal deel) van string

11:33

De enige manier om van de A's af te komen is dat ze tegen een B aan staan en dan C worden.
De enige manier om van de C's af te komen is dat ze langs (alleen) B's naar rechts lopen, tot R toe.
Net zo voor de D's.
Dit houdt in dat de B's en de R niet te vroeg kunnen verdwijnen, want anders houden we variabelen A/C/D over.

11:37

(b) $\underline{S} \Rightarrow \underline{T}R \Rightarrow \underline{T}BR \Rightarrow \underline{AB}BB^R \Rightarrow \underline{CB}BB^R \Rightarrow BC\underline{DB}BR \Rightarrow$
$BCB\underline{DB}R \Rightarrow BCBB\underline{DR} \Rightarrow BC\underline{BB}R1 \Rightarrow B\underline{BC}\underline{DB}R1 \Rightarrow$
$BBCB\underline{DR}1 \Rightarrow BB\underline{CB}R11 \Rightarrow BBBCD\underline{R}11 \Rightarrow BBBC\underline{R}111 \Rightarrow$
$\underline{BBB}R111 \Rightarrow^4 111111$

11:42

11:43 **Uitwerking tentamen Fundamentele Informatica 3**
woensdag 5 juni 2013

3 (a)

Met een afleiding in de grammatica willen we een berekening van T_1 voor een bepaalde invoer x simuleren.
Als deze berekening tot h_a leidt ($\Rightarrow T_1$ accepteert x), moet x ook door G_1 gegenereerd kunnen worden. We moeten dus nog weten wat de oorspronkelijke string x was.

Daarom wordt elke letter van x aan het begin dubbel aangemaakt, tussen de haken. Het eerste symbool tussen de haken verandert niet. Daar kunnen we de oorspronkelijke tape-inhoud dan altijd nog uit teruglezen (x, maar ook Δ's ervoor en erachter).
Op het tweede symbool tussen de haken gaan we de stappen van T_1 tijdens de berekening simuleren. De rij van tweede-symbolen-van-paren vormt dus de tape-inhoud tijdens de berekening.

11:53.

(b) Zoals hierboven gezegd, willen we, voor TMs ook bij LBA M_2 als de berekening van M_2 voor x naar h_a leidt, aan het eind x reconstrueren uit de gegenereerde string. Deze string bevat dan onder andere h_a.
Als de string bijvoorbeeld de deelstring $h_a(\sigma, b)$ bevat, en we zouden de productie $h_a(\sigma, b] \longrightarrow \sigma_1$ willen toepassen (wat in G_1 geen enkel probleem is), mag dat in de CSG G_2 alleen wanneer $h_a(\sigma, b)$ één symbool is. Anders zou de rechter kant van de productie korter zijn dan de linker kant, en dat mag niet in een CSG.

Daarom vormt het voorkomen van een toestand $(h_a, p, q, \ldots)$ in G_2 altijd één symbool samen met het paar $(\sigma_4 \sigma_5)$ dat erachter staat. Als we een productie van de vorm

(een string in)

$$p(\sigma, a) \longrightarrow (\sigma, b)\, q$$

zouden hebben, zou dat voor q niet meer gelden. Daarom zetten we $(\sigma_2 \sigma_3)$ er links en rechts achter:

$$\underline{p(\sigma, a)}\ \underline{(\sigma_2 \sigma_3)} \longrightarrow \underline{(\sigma, b)}\ \underline{q(\sigma_2 \sigma_3)}$$

en krijgen we een productie met zowel links als rechts twee (onderstreepte) symbolen.
In onbeperkte grammatica G_1 hoeft dat niet, want daar mag een toestand * (zoals eerder aangegeven) best een losstaand symbool zijn. B.v. bij de productie $h_a(\sigma, b] \longrightarrow \sigma_1$ mogen we best vijf symbolen links herschrijven in één symbool rechts.
Daarom hebben we daar $(\sigma_2 \sigma_3)$ niet nodig.

12:10

(c) $\sigma_1 \in \Sigma$
$\sigma_2 \in \Sigma$
$\sigma_3 \in \Gamma \cup \{\Delta\}$.

12:12

4 (a)

We noemen R een taal eigenschap als voor alle TMs T_1 en T_2 met $L(T_1) = L(T_2)$ geldt:

T_1 heeft eigenschap $R \iff T_2$ heeft eigenschap R.

Ofwel als een TM T_1 eigenschap R heeft, dan heeft elke TM T_2 die dezelfde taal accepteert als T_1 ook eigenschap R.

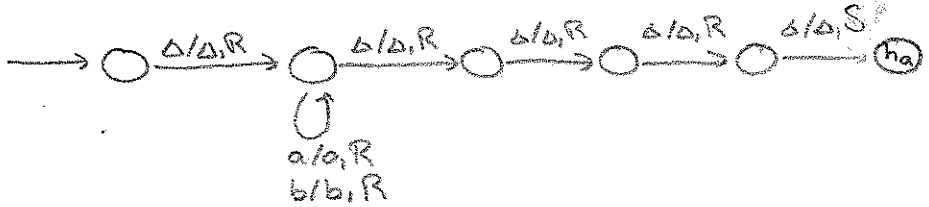
12:14

(b)

We hebben twee TMs T_1 en T_2 nodig die dezelfde taal accepteren, terwijl T_1 eigenschap R wel heeft en T_2 niet (of andersom).
Omdat T_1 en T_2 dezelfde taal moeten accepteren, moet (o.a.) wel gelden dat $ababab \in L(T_1) \iff ababab \in L(T_2)$.
Het onderscheid in wel of niet eigenschap R hebben, kan dus niet in het wel of niet accepteren van ababab zitten.
Zowel T_1 als T_2 moet ababab accepteren; alleen moet T_1 dat doen zonder ooit positie 10 op de tape te bereiken, en T_2 moet dan doen nadat hij wel positie 10 heeft bereikt.

Neem T_1:

T_2:



Er geldt $L(T_1) = L(T_2) = \{a,b\}^*$, Beide accepteren ababab.
T_1 komt voor invoer ababab niet verder dan positie 0.
→ en heeft dus eigenschap R
T_2 komt voor invoer ababab tot positie $6+4 = 10$
en heeft dus niet eigenschap R.

12:23.

(c)

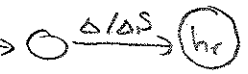
We moeten een reductie AcceptsNothing $\leq$ Subset vinden

Instanties van AcceptsNothing zijn TMs T.

Instanties van Subset zijn paren TMs (T_1, T_2)

Voor een willekeurige TM T moeten we dus TMs (T_1, T_2) vinden / construeren, zódat $L(T) = \emptyset \iff L(T_1) \subseteq L(T_2)$

We kunnen nemen $T_1 = T$

T_2 is de volgende TM. 

Duidelijk is dat $L(T_2) = \emptyset$.

Er geldt:

T is ja-instantie van AcceptsNothing $\iff L(T) = \emptyset \iff L(T) \subseteq \emptyset$

(die enige deelverzameling van $\emptyset$ is immers $\emptyset$ zelf)

$\iff L(T_1) \subseteq L(T_2) \iff (T_1, T_2)$ is ja-instantie van Subset.

Verder is de constructie van (T_1, T_2) uit T uiteraard makkelijk algoritmisch uit te voeren.

We hebben derhalve een geldige reductie van AcceptsNothing naar Subset : AcceptsNothing $\leq$ Subset.

Het beslissingsprobleem Subset is dus 'minstens zo moeilijk' als AcceptsNothing. Omdat AcceptsNothing onbeslisbaar is, is volgens een stelling uit het boek ook Subset onbeslisbaar

12:35

5 (a) Als f is ontstaan uit g en h door de operatie van primitieve recursie, heeft f $(n+1)$ argumenten en is f voor $X \in \mathbb{N}^n$ als volgt gedefinieerd

$$f(X, 0) = g(X)$$
$$f(X, k+1) = h(X, k, f(X, k))$$

12:39

(b) We kunnen de functie $Moves_T$ als volgt formeel beschrijven.

$$Moves_T(m, 0) = \begin{cases} m & \text{als } IsConfig_T(m) \\ 0 & \text{anders} \end{cases}$$

$$Moves_T(m, k+1) = \begin{cases} Move_T(Moves_T(m,k)) & \text{als } IsConfig_T(m) \\ 0 & \text{anders} \end{cases}$$

$Moves_T$ is derhalve het resultaat van primitieve recursie op functies g en h gebaseerd de primitieve recursieve functies $IsConfig_T$ en $Move_T$.
Dit betekent dat ook $Moves_T$ zelf primitief reaursief is

12:46

(c)

$NumberOfMovesToAccept_T(m)$ kun je ook zien als het kleinste aantal stappen k, zódat $Moves_T(m,k)$ het configuratiegetal van een accepterende configuratie is
Ofwel: het kleinste aantal stappen k, zódat

$$IsAccepting_T(Moves_T(m,k)) = 0$$

Dus $NumberOfMovesToAccept_T(m) = \mu k [IsAccepting_T(Moves_T(m,k)) = 0]$

Bij (b) zagen we dat $Moves_T(m,k)$ primitief recursief is
Gegeven is dat $IsAccepting_T$ primitief recursief is
Dit betekent (door compositie) dat de functie $IsAccepting_T(Moves_T(m,k))$ primitief recursief is. (en dus totaal & en μ-recursief)
Maar dan is $NumberOfMovesToAccept_T$ het resultaat van onbegrensde minimalisatie op een totale μ-recursieve functie.
Dat betekent dat $NumberOfMovesToAccept_T$ zelf μ-recursief is.

12:56

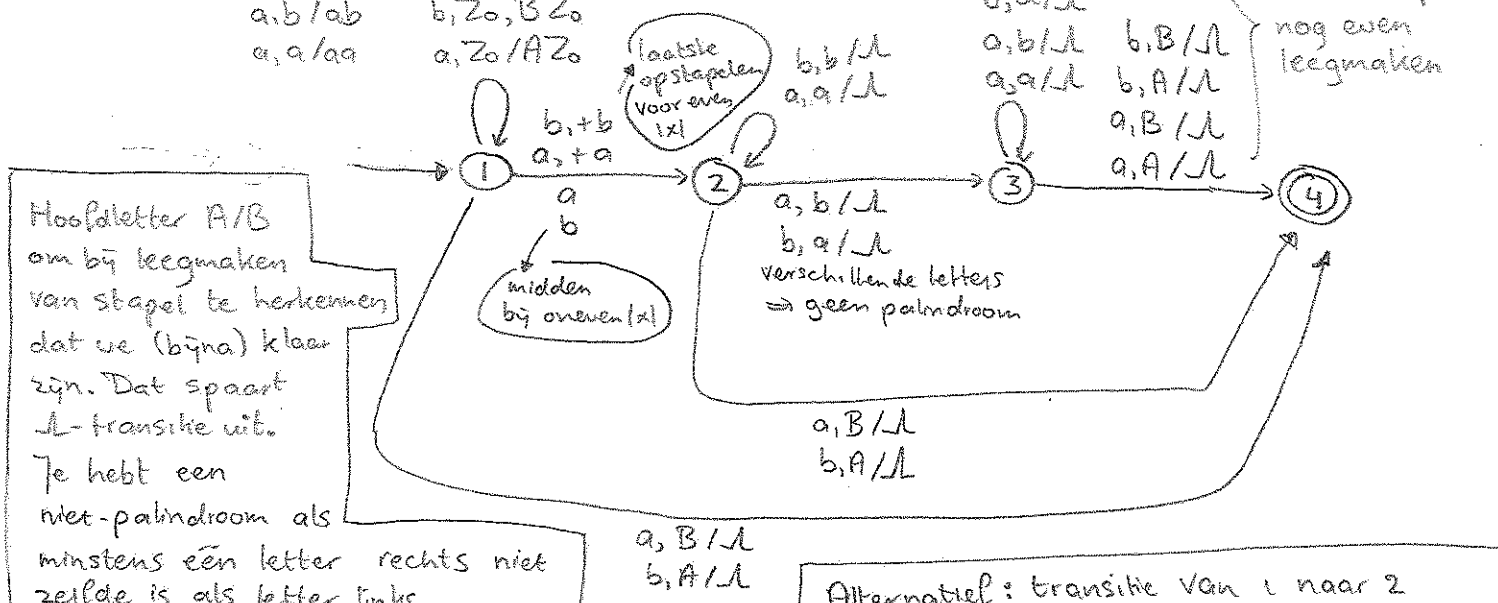
Uitwerking tentamen Fundamentele Informatica 3, woensdag 5 juni 2013

14:50

5'(a)

b,b/bb
b,a/ba
a,b/ab
a,a/aa

b,A/bA
b,B/bB
a,B/aB
a,A/aA
b,Z0,BZ0
a,Z0/AZ0



Hoofdletter A/B om bij leegmaken van stapel te herkennen dat we (bijna) klaar zijn. Dat spaart Λ-transitie uit.
Je hebt een niet-palindroom als minstens één letter rechts niet zelfde is als letter links.
Die eerste letter rechts zoeken we.

Alternatief: transitie van 1 naar 2 uitbreiden met a,Z0/AZ0 en b,Z0/BZ0. Dan kan transitie van 1 naar 4 weg.

14:57

(b)

Omdat het bij een stapelautomaat M kan voorkomen dat je met dezelfde invoerstring x in verschillende toestanden (van M) kunt eindigen, kan

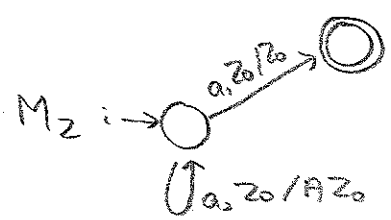
Als een van die toestanden een eindtoestand is en een ander niet, dan zou de ene toestand ervoor zorgen dat M x accepteert en de andere toestand ervoor zorgen dat M' (uit M ontstaan door eindtoestanden en niet-eindtoestanden te verwisselen) x accepteert.
⇒ $x \in L(M)$ en $x \in L(M')$ ⇒ $L(M')$ is dan niet $(L(M))'$.

Dit is mogelijk in een stapelautomaat door twee zaken (die elk op zich al voldoende zijn om dit te veroorzaken).

* een stapelautomaat is niet per se deterministisch. Je kunt dus bijvoorbeeld vanuit een bepaalde toestand p met een bepaalde invoerletter σ en een bepaalde symbool X bovenop de stapel meerdere transities hebben
* een stapelautomaat kan Λ-transities hebben: nadat je een woord x helemaal gelezen en verwerkt hebt, kun je dan misschien nog doorlopen naar een andere toestand over zulke Λ-transities

15:07

(c)



$L(M_2) = \{a\}$

15:10

7a, M_2 kan ook deterministisch zijn, vanwege (de genoemde) Λ-transities.

Bijvoorbeeld:

M_2 

$L(M_2) = \{a\}$

15:12