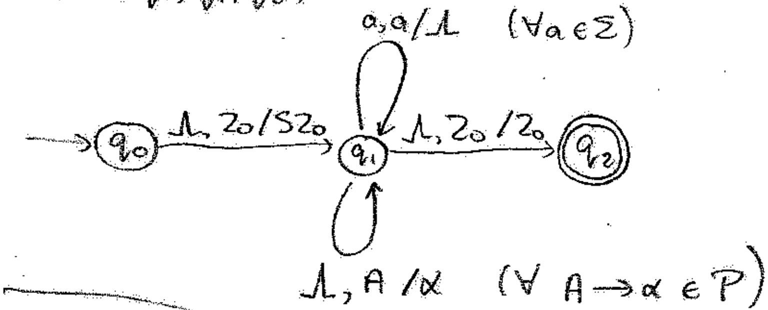
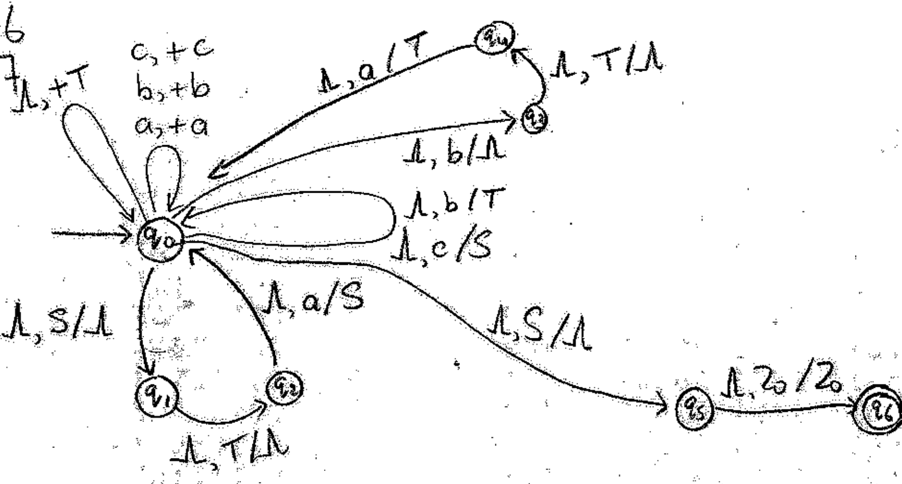
11:32 Uitwerking tentamen Fundamentele Informatica 3, maandag 11 juni 2012

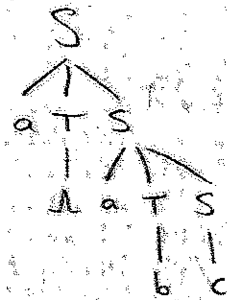
1(a) $Q = (q_0, q_1, q_2)$



11:36
11:37
(b)



Alternatief 1c):

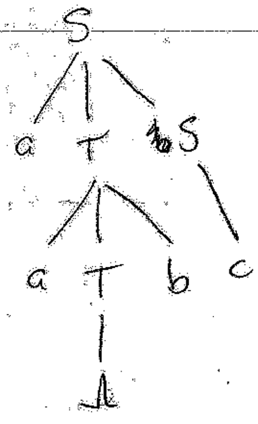


Linkspref: $S \Rightarrow aTS \Rightarrow aS \Rightarrow aaTS \Rightarrow aabS \Rightarrow aabc$

Rechtspref: $S \Rightarrow aTS \Rightarrow aTaTS \Rightarrow aTaTc \Rightarrow aTabc \Rightarrow aabc$

RvV, 21-06-2012

11:43
(c)



Afleiding (links-preferent) is:

$S \Rightarrow aTS \Rightarrow aaTbS \Rightarrow aabS \Rightarrow aabc$

Afleiding (rechts-preferent) is:

$S \Rightarrow aTS \Rightarrow aTc \Rightarrow aaTbc \Rightarrow aabc$

RvV, 21-06-2012

11:45
(d) $(q_0, aabc, Z_0) \vdash (q_0, abc, aZ_0) \vdash (q_0, bc, aaZ_0) \vdash (q_0, bc, TaaZ_0)$

$\vdash (q_0, c, bTaaZ_0) \vdash (q_3, c, TaaZ_0) \vdash (q_4, c, aaZ_0) \vdash (q_0, c, TaZ_0)$

$\vdash (q_0, \Lambda, cTaZ_0) \vdash (q_0, \Lambda, STaZ_0) \vdash (q_1, \Lambda, TaZ_0) \vdash (q_2, \Lambda, aZ_0)$

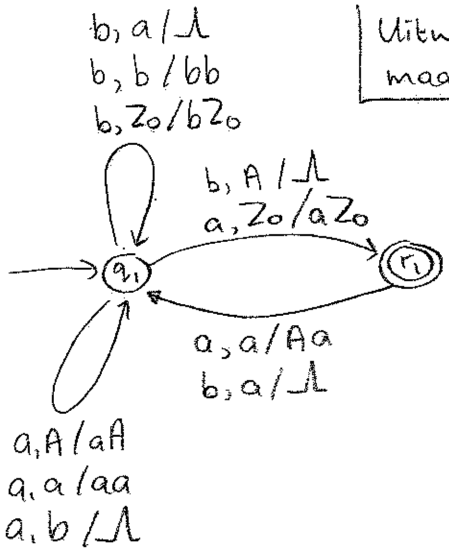
$\vdash (q_0, \Lambda, SZ_0) \vdash (q_5, \Lambda, Z_0) \vdash (q_6, \Lambda, Z_0)$

11:50

Uitwerking tentamen Fundamentele Informatica 3
maandag 11 juni 2012

11:50

2 a)



11:54

Pieter loopt 2b

b) De toestanden Q zijn alle paren van toestanden uit M_1 en M_2:

$$Q = Q_1 \times Q_2$$

De begintoestand is het paar van de begintoestanden uit M_1 en M_2

$$q_0 = (q_1, q_2)$$

De eindtoestanden zijn alle paren van eindtoestanden uit M_1 en uit M_2

$$A = A_1 \times A_2$$

Als M_1 een transitie

$$(p_1) \xrightarrow{a,\ A/\alpha} (q_1) \quad \text{bevat}$$

en M_2 een transitie $(p_2) \xrightarrow{a} (q_2)$ bevat,

bevat M een transitie

$$(p_1, p_2) \xrightarrow{a,\ A/\alpha} (q_1, q_2)$$

Als M_1 een transitie

$$(p_1) \xrightarrow{\Lambda,\ A/\alpha} (q_1) \quad \text{bevat,}$$

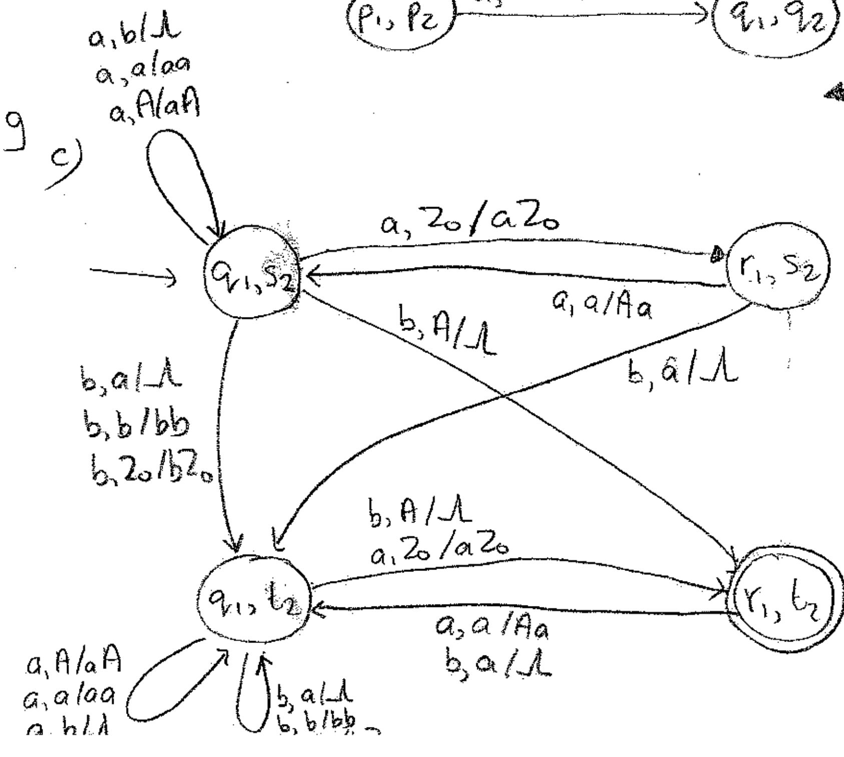
bevat M transities

$$(p_1, p_2) \xrightarrow{\Lambda,\ A/\alpha} (q_1, p_2)$$

voor alle $p_2 \in Q_2$

11:59

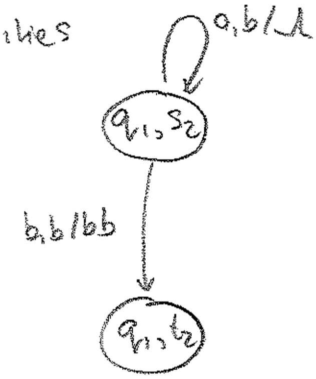
c)



d) Alle vier toestanden kunnen ooit bereikt worden,
bijvoorbeeld bij invoer aba

In de 'bovenste twee' toestanden blijf je slechts zolang je a's leest.
Zo gauw je een b leest, ga je naar beneden, en daar blijf je dan
⟹ bovenin zul je nooit een b op de stapel hebben staan,
want die zet je erop als je een b leest (en er geen a/A op de stapel staat)

⟹ de transities

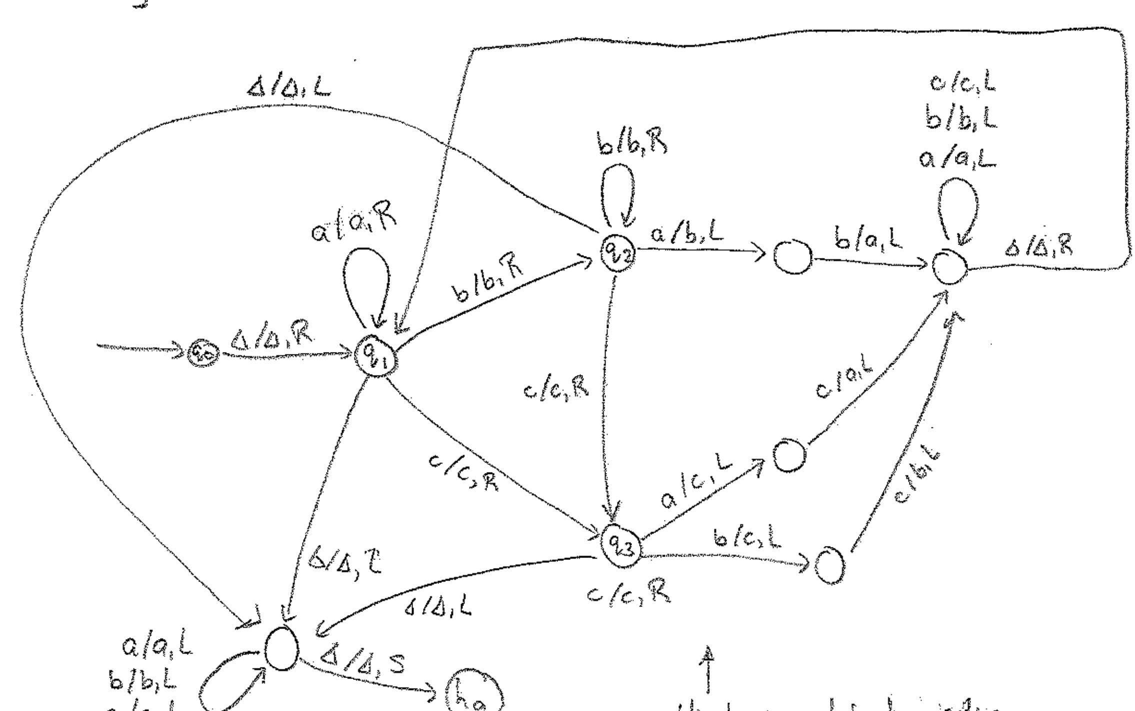


zul je nooit gebruiken
Alle andere transities kunnen wel ooit gebruikt worden.

12:20
13:46

3) ~~T loopt de string van links naar rechts door,~~
en onthoudt wat de 'grootste' letter is die hij tot nu toe heeft gezien:
alleen a's, (q_1) ook b's, (q_2) of ook c's. (q_3)
Zo gauw T een letter tegenkomt die kleiner is dan de grootste tot nu toe (dat is tevens de vorige gelezen letter), verwisselt hij deze twee letters en begint van voren af aan (weer aan de linker kant dus).

Als we aan rechterkant op Δ stuiten, staat kennelijk hele woord 'in de goede volgorde'.



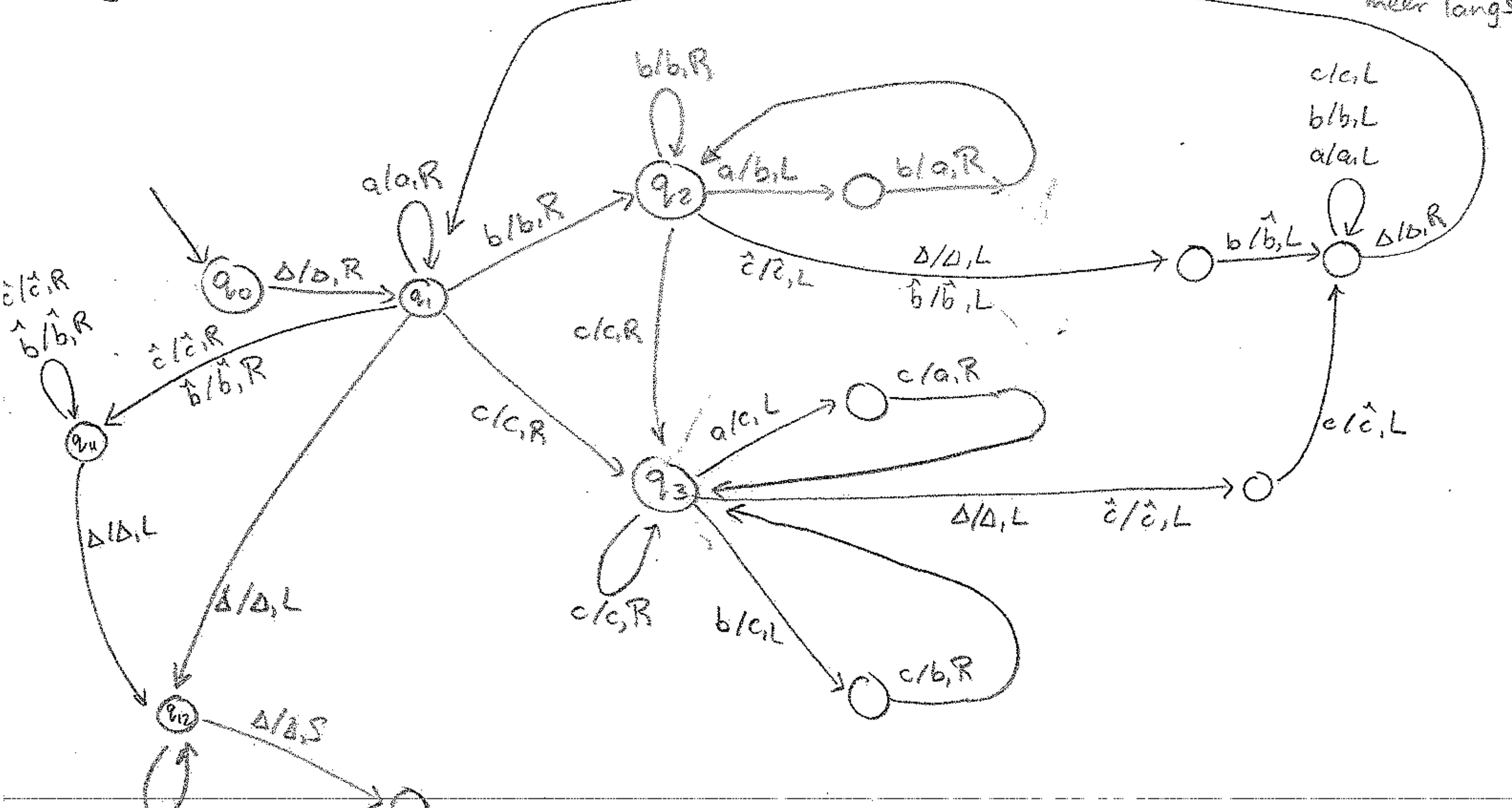
Hierboven drie transities omdat we een kleinere letter tegenkomen dan de grootste tot nu toe

14:01

14:01 Uitwerking tentamen Fundamentele Informatica 3, maandag 11 juni 2012

Alternatief: à la BubbleSort (hebben we in HW3 bekeken)

Als we twee letters moeten verwisselen, lopen we door naar rechts (na het verwisselen). Zo komt in één iteratie grootste element in ieder geval rechts. Dat grootste element markeren we met een ^. Daar hoeven we volgende iteratie niet meer langs



Als we in q_1 op Δ of $\hat{b}$ of $\hat{c}$ stuiten, weten we dat alles in goede volgorde staat
$\Rightarrow$ alleen (zonodig) nog ^'s verwijderen, in q_{11} en q_{12}

14:16

14:20

4 (a) $e(q_0) = 111$ $e(q_1) = 1111$ $e(h_a) = 1$ $e(h_r) = 11$

$e(\Delta) = 1$ $e(a) = 11$ $e(b) = 111$ $e(c) = 1111$

$e(R) = 1$ $e(L) = 11$ $e(S) = 111$

We hoeven alleen de vier transities te coderen.

Als $\delta(p,a) = (q,b,D)$, wordt de codering: $e(p)0e(a)0e(q)0e(b)0e(D)0$

En dan ná iedere transitie een extra 0

$\underbrace{1110101111010100}$ $\underbrace{11110110111101101 00}$ $\underbrace{1111011101010111 00}$ $\underbrace{111101011011110110 0}$

14:28

Evt ook begintoestand coderen, want moet volgens derde editie.

(b) * x moet van de vorm $(11^*0)^5 0 ((11^*0)^5 0)^*$ zijn,
ofwel minstens een vijftal strings van minstens een 1, gescheiden door 0'en

* (de (combinaties van) de eerste twee componenten van de vijftallen moeten allemaal verschillend zijn,
ofwel iedere transitie wordt maar een keer gecodeerd,
en er zijn niet twee verschillende transities voor een combinatie (toestand, letter)

* de eerste component van elk vijftal bevat minstens drie 1'en,
ofwel er zijn geen transities vanuit h_a en h_r

* de laatste component van elk vijftal bevat hoogstens drie 1'en,
want er zijn maar drie mogelijke richtingen

14:36

5 (a) 'Het grote doel' van deze producties is om een willekeurige beginconfiguratie van de Turingmachine te genereren, inclusief een begintoestand en voldoende vakjes met Δ achter de invoer om de berekening van de Turing machine voor zijn invoer helemaal te kunnen simuleren (althans voor een invoer die geaccepteerd wordt)

- symbolen worden steeds dubbel gegenereerd, omdat we met de tweede kopie de Turing machine gaan simuleren, terwijl we in de eerste kopie de oorspronkelijke invoer onthouden. Die eerste kopie hebben we nodig als we in stap 3 deze oorspronkelijke invoer willen reconstrueren.
- Het doel van de productie $S \to S(\Delta\Delta)$ is om voldoende vakjes met Δ na de invoer te genereren (zie 'grote doel') in de beginconfiguratie
- Het doel van de productie $T \to q_0(\Delta\Delta)$ is om ~~de~~ stap 1 (het genereren van een beginconfiguratie) af te ronden: we zetten nog een Δ voor de invoer, en zetten daarvoor de begintoestand q_0.

14:46

(b) Tweede soort: simuleren van de Turing machine

$q_0(\sigma\Delta) \longrightarrow (\sigma\Delta)q_1 \qquad \sigma \in \{a,b,\Delta\}$

$q_1(\sigma a) \longrightarrow (\sigma a)q_1 \qquad$ " "

$q_1(\sigma b) \longrightarrow h_a(\sigma\Delta) \qquad \sigma$ " "

$(\sigma_1\sigma_2)q_1(\sigma_3\Delta) \longrightarrow h_r(\sigma_1\sigma_2)(\sigma_3 c) \qquad \sigma_1,\sigma_3 \in \{a,b,\Delta\}$, $\sigma_2 \in \{a,b,c,\Delta\}$.

14:50

~~Derde soort~~: reconstrueren van oorspronkelijke invoer

$h_a(\sigma_1\sigma_2) \longrightarrow h_a(\sigma_1\sigma_2)h_a \qquad \sigma_1 \in \{a,b,\Delta\} \quad \sigma_2 \in \{a,b,c,\Delta\}$

$(\sigma_1\sigma_2)h_a \longrightarrow h_a(\sigma_1\sigma_2)h_a \qquad$ " "

$h_a(\Delta\sigma_2) \longrightarrow \Lambda \qquad \sigma_2 \in \{a,b,c,\Delta\}$.

$h_a(\sigma_1\sigma_2) \longrightarrow \sigma_1 \qquad \sigma_1 \in \{a,b\} \quad \sigma_2 \in \{a,b,c,\Delta\}$

14:53

(c) $S \Rightarrow I \Rightarrow T(aa) \Rightarrow T(bb)(aa) \Rightarrow T(aa)(bb)(aa) \Rightarrow$

$q_0(\Delta\Delta)(aa)(bb)(aa) \Rightarrow (\Delta\Delta)\,q_1(aa)(bb)(aa) \Rightarrow (\Delta\Delta)(aa)\,q_1(bb)(aa) \Rightarrow$

$(\Delta\Delta)(aa)\,h_a(b\Delta)(aa) \Rightarrow (\Delta\Delta)\,h_a(aa)\,h_a(b\Delta)(aa) \Rightarrow h_a(\Delta\Delta)\,h_a(aa)\,h_a(b\Delta)(aa)$

$\Rightarrow h_a(\Delta\Delta)\,h_a(aa)\,h_a(b\Delta)\,h_a(aa) \Rightarrow h_a(aa)\,h_a(b\Delta)\,h_a(aa) \Rightarrow$

$a\,h_a(b\Delta)\,h_a(aa) \Rightarrow ab\,h_a(aa) \Rightarrow aba.$

15:00.

6] 20:24

a] AcceptsAllLengths

Stelling van Rice direct toepasbaar, want

- * er staat: gegeven een Turing machine T
- * er staat feitelijk: bevat $L(T)$ woorden van alle lengtes ≥ 0
  - $\Rightarrow$ dat is een taal eigenschap
- * dit is een niet-triviale taal eigenschap,
  - want er is een TM T_1 met $L(T_1) = \Sigma^* \Rightarrow T_1$ heeft de eigenschap
  - er is een TM T_2 met $L(T_2) = \emptyset \Rightarrow T_2$ heeft de eigenschap niet

AcceptsLengthN

~~zelfde antwoord, behalve bij tweede * moet staan:~~

- * er staat feitelijk: bevat $L(T)$ een woord van lengte N
  - $\Rightarrow$ dat is een taal eigenschap.

AcceptsLength

Stelling van Rice is niet direct toepasbaar, want

- * er staat niet (slechts): gegeven een Turing machine T

20:34

b] Dat moet dus wel AcceptsLength worden.

Een instantie van: AcceptsLength is een tweetal (T_2, N_2) met T_2 een TM en N_2 een geheel getal ≥ 0

We kunnen AcceptsLengthN hiernaar reduceren

Een instantie van AcceptsLengthN is een TM T_1

Neem als $T_2 = T_1$ en als $N_2 = N$ van het beslissingsprobleem AcceptsLengthN.

Zo'n instantie $(T_2, N_2) = (T_1, N)$ is eenvoudig, en dus zeker algoritmisch te construeren uit T_1.

Inderdaad geldt:

T_1 ja-instantie van AcceptsLengthN $\Leftrightarrow$ T_1 accepteert een woord van lengte N $\Leftrightarrow$
$T_2 = T_1$ accepteert een woord van lengte $N_2 = N$ $\Leftrightarrow$ (T_2, N_2) is ja-instantie van AcceptsLength

We hebben dus inderdaad een reductie: AcceptsLengthN $\leq$ AcceptsLength.

Omdat AcceptsLengthN volgens de stelling van Rice onbeslisbaar is (zie (a)) is AcceptsLength ook onbeslisbaar.

20:46.