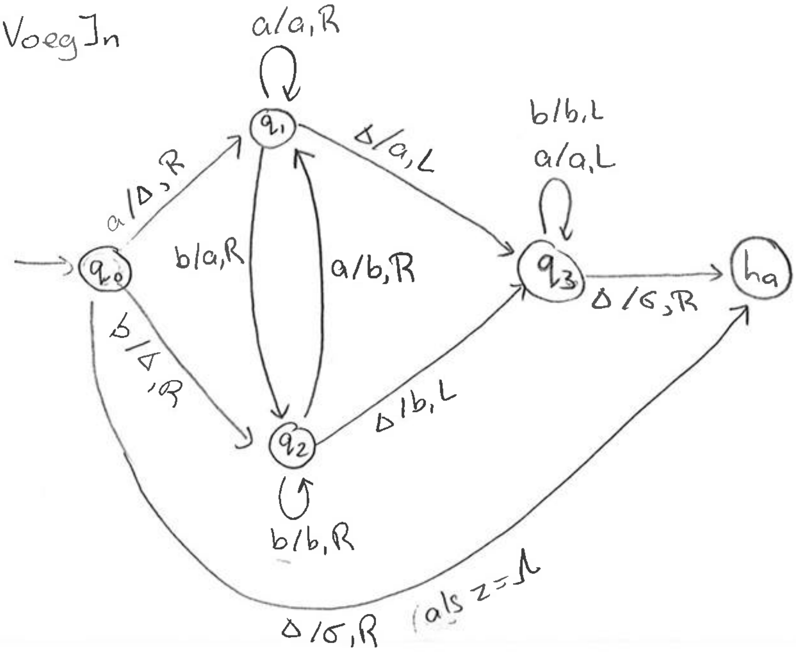
Uitwerking tentamen Fundamentele Informatica 3, vrijdag 13 januari 2017

15.29

1(a)

VoegIn



naar rechts op de tape

15.34

VoegIn schuift de string z letter voor letter, van links naar rechts een positie. Daartoe vervangt hij het huidige symbool op de leeskop door Δ, zodat hij aan het eind, in toestand q_3 weet hoe ver hij terug moet lopen. VoegIn onthoudt in toestanden q_1 en q_2 het symbool dat hij als laatste heeft verwijderd: a, respectievelijk b. Dat symbool schrijft hij terug op de volgende tape-positie, waarmee hij het daar aanwezige symbool overschrijft. Enzovoort, tot hij op de Δ direct achter z terechtkomt. Dan gaat hij naar toestand q_3, waar hij terugloopt naar de neergezette Δ, waar hij σ neerzet, en accepteert op de tapepositie rechts daarvan

15.43

(b)

De berekening van T_0 voor invoer ab:

$$q_0\Delta ab \vdash \$q_1ab \vdash \$\Delta q_2ab \vdash \$\Delta\Delta q_3ab \vdash \$\Delta\Delta 1q_1b \vdash \$\Delta\Delta 1\Delta q_2b$$
$$\vdash \$\Delta\Delta 1\Delta\Delta q_3b \vdash \$\Delta\Delta 1\Delta\Delta 1q_4\Delta \vdash \$\Delta\Delta 1\Delta\Delta 11q_2\Delta \vdash \$\Delta\Delta 1\Delta\Delta 11\Delta q_3\Delta$$
$$\vdash^* q_3\$\Delta\Delta 1\Delta\Delta 11\Delta\Delta \vdash h_a\Delta\Delta\Delta 1\Delta\Delta 11\Delta\Delta$$

15.50

(c) In het voorbeeld hierboven zagen we dat T_0 de tape inhoud Δab verving door ΔΔΔ1ΔΔ11Δ. De Δ werd op het pad $q_0 \to q_1 \to q_2 \to q_3$ vervangen door \$ΔΔ. Vervolgens werd de a op het pad $q_3 \to q_1 \to q_2 \to q_3$ vervangen door 1ΔΔ. Daarna werd de b op het pad $q_3 \to q_4 \to q_2 \to q_3$ vervangen door 11Δ. Tenslotte werd de \$ weer Δ.

Voor willekeurige $x \in \{a,b\}^*$ zal de configuratie $q_0\Delta x$ door T_0 worden omgezet in $h_a\Delta\Delta\Delta y$, waarbij in y elke a van x vervangen is door 1ΔΔ en elke b in x vervangen is door 11Δ.

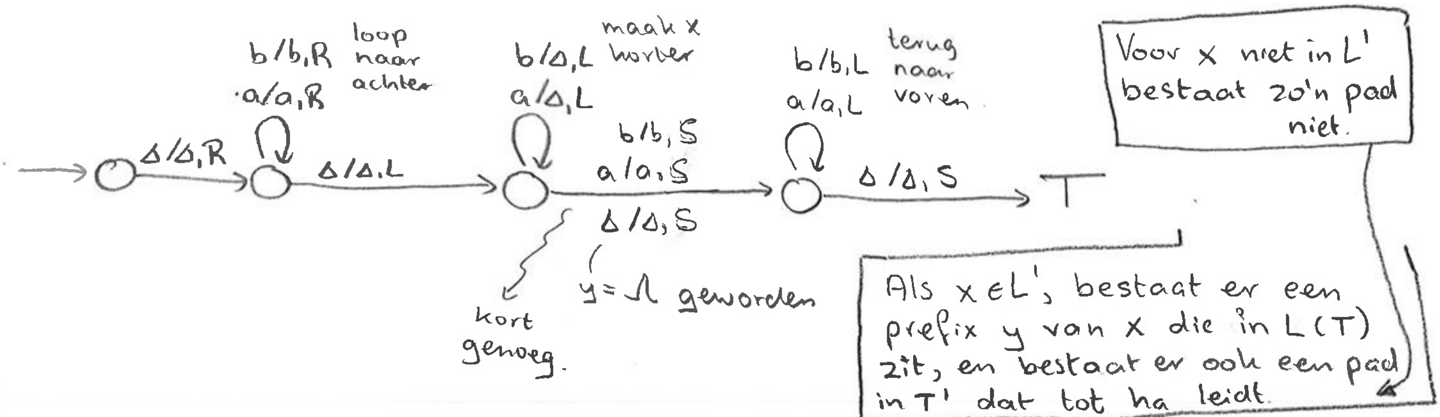
We hebben de string x als het ware gecodeerd met Δ's en 1'en, steeds drie symbolen voor elk symbool van x, net als in huiswerkopgave 3.

15.58

2 (a)

Een string $x \in \{a,b\}^*$ moet door T' geaccepteerd worden, als hij te schrijven is als $x = yz$, met $y \in L$ en $z \in \{a,b\}^*$. Met andere woorden: als een prefix y van x door T geaccepteerd wordt.

T' maakt zijn invoer daartoe niet-deterministisch korter aan de achterkant, en roept voor het resultaat y vervolgens T aan:



16.08

(b) De functie van het speciale symbool \$ op de derde tape van T_2 is om te voorkomen dat T_2 links van deze tape zou aflopen.

We komen het symbool tegen

* wanneer T_1 op het gesimuleerde pad / bitstring links van de tape zou lopen
* wanneer we na het simuleren van een pad / bitstring de derde tape schoon willen vegen, om x daar vervolgens weer naartoe te kopiëren voor de volgende simulatie.

Als dat namelijk zou gebeuren, zou T_2 crashen en zijn invoer x dus niet accepteren, terwijl er misschien nog een niet-gesimuleerd pad / bitstring is die T_2 voor invoer x naar h_a zou voeren.

16.18

(c) De functie van de bitstrings op de vierde tape van T_2 is om te herkennen wanneer we kunnen stoppen met het simuleren van paden / bitstrings, omdat we toch nooit zullen accepteren, ook niet bij langere bitstrings. We bekijken deze bitstrings wanneer we juist de laatste bitstring van een bepaalde lengte $(11\dots1)$ hebben gesimuleerd en ook naar die vierde tape hebben geschreven. We controleren dan of misschien alle bitstrings van die lengte op de vierde tape staan. Als dat het geval is, kunnen we stoppen met simuleren en x verwerpen. Immers, elk langere pad / bitstring heeft een prefix waarop T_1 al vastloopt / verwerpt. Er kan dan geen pad bestaan dat T_1 voor invoer x naar h_a leidt.

16.26

Uitwerking tentamen Fundamentele Informatica 3, vrijdag 13 januari 2017

17.07

3 (a) We noemen een unrestricted grammatica context-gevoelig wanneer iedere productie in P van de vorm $\alpha \to \beta$ is met $|\alpha| \leq |\beta|$. De rechterkant van een productie is dus altijd minstens zo lang als de linkerkant.

17.10

17.49

(b)(i)

De eerste vijf elementen van L in de canonieke volgorde:

ba, bbaa, abbbaa, bbbaaa, abbbbaaa

ba	bbaa	abbbaa	bbbaaa	abbbbaaa
$i=0$	$i=0$	$i=1$	$i=0$	$i=1$
$k=1$	$k=2$	$k=2$	$k=3$	$k=3$

17.51 En vervolgens bbbbaaaa ($i=0, k=4$), aabbbbbaaa ($i=2, k=3$), abbbbbaaaa ($i=1, k=4$), bbbbbaaaaa ($i=0, k=5$).

(b)(ii)

Een context-gevoelige grammatica voor L met de volgende producties:

$S \to A_1 B_1 B_1 C_1 S$ — A_1 voor a links, B_1 voor b, C_1 voor a rechts; bouw i op: evenveel a's links als rechts

$S \to T$ — i is klaar, ga nu verder met k

$T \to B_1 C_1 T$ — nog een B_1 voor b en een C_1 voor a rechts

$T \to B_1 R$ — k klaar. Ook R staat voor een a rechts, maar dan voor de meest rechtse a. Vanaf R naar links gaan we straks de volgorde van de hoofdletters A_1, B_1, C_1 controleren.

$B_1 A_1 \to A_1 B_1$
$C_1 A_1 \to A_1 C_1$
$C_1 B_1 \to B_1 C_1$

om de A_1's links te kunnen krijgen en de C_1's rechts.

$C_1 R \to C_2 C_2$
$C_1 C_2 \to C_2 C_2$
$B_1 C_2 \to B_2 C_2$
$B_1 B_2 \to B_2 B_2$
$A_1 B_2 \to A_2 B_2$
$A_1 A_2 \to A_2 A_2$

controleer van rechts naar links de volgorde van de A_1's, B_1's en C_1's. Met hulpvariabelen A_2, B_2 en C_2 om te voorkomen dat de a's links en de a's rechts door elkaar mogen blijven

$B_1 R \to B_2 C_2$ — voor geval $i=0, k=1$.

$A_2 \to a$
$B_2 \to b$
$C_2 \to a$

rond af naar de gewenste kleine letters.

18.04

Voor ~~twee~~ elegante alternatieve oplossingen, zie blz 7

~~11.29~~ 32

(b) iii)

Een afleiding in G van het woord abbbaa:

$$S \Rightarrow A_1 B_1 B_1 C_1 S \Rightarrow A_1 B_1 B_1 C_1 T_1 \Rightarrow A_1 B_1 B_1 C_1 B_1 R \Rightarrow A_1 B_1 B_1 B_1 C_1 R$$
$$\Rightarrow A_1 B_1 B_1 B_1 C_2 C_2 \Rightarrow^* A_2 B_2 B_2 B_2 C_2 C_2 \Rightarrow^* abbbaa.$$

11.36

11.50

5 (a)

We noemen een eigenschap R van Turingmachines een niet-triviale taaleigenschap, als

* R een taaleigenschap is. Dat wil zeggen: voor elke twee Turingmachines T_1 en T_2 met $L(T_1) = L(T_2)$ geldt: T_1 en T_2 hebben eigenschap R allebei wel of allebei niet
* en R niet-triviaal is. Dat wil zeggen dat er een Turingmachine T_1 bestaat die R wel heeft en een Turingmachine T_2 bestaat die R niet heeft.

11.55

(b) Beschouw de volgende twee Turingmachines T_1 en T_2, allebei met hetzelfde invoeralfabet Σ:

T_1 $\quad \rightarrow \bigcirc \xrightarrow{\Delta/\Delta,S} (h_a)$

T_2 $\quad \rightarrow \bigcirc \xrightarrow{\Delta/\Delta,S} \bigcirc \xrightarrow{\Delta/\Delta,S} (h_a)$

Er geldt $L(T_1) = L(T_2) = \Sigma^*$, maar T_1 heeft eigenschap R niet (hij accepteert Λ in één stap), terwijl T_2 eigenschap R wel heeft (hij accepteert Λ in twee stappen).

R is dus geen taaleigenschap.

~~11.59~~ 12.02

(c)

Laat P_2 het volgende beslissingsprobleem zijn.

$$P_2 = \text{Accepts-}\Lambda: \text{ gegeven een Turing machine } T, \text{ is } \Lambda \in L(T)?$$

We hebben de volgende reductie nodig:

$$P_2 \leq P_1$$

Laat T_2 een willekeurige instantie van P_2 zijn. We construeren uit T_2 een instantie T_1 van P_1 door elke transitie

$(p) \xrightarrow{X/Y,D} (q)$ in T_2 te vervangen door $(p) \xrightarrow{X/X,S} \bigcirc \xrightarrow{X/Y,D} (q)$

met in het midden een nieuwe hulptoestand die nergens anders gebruikt wordt.

12.11

Het is duidelijk dat T_1 algoritmisch uit T_2 te verkrijgen is.

Verder geldt:

T_2 is een ja-instantie van P_2 $\Leftrightarrow$ T_2 accepteert Λ $\Leftrightarrow$

T_1 accepteert Λ in een even aantal stappen $\Leftrightarrow$

T_1 is een ja-instantie van P_1

We hebben dus inderdaad met een reductie te maken.

Vanwege de stelling van Rice is P_2 niet-beslisbaar. Omdat $P_2 \leq P_1$ is dan ook P_1 niet beslisbaar.

12.15

T_1 accepteert precies dezelfde strings als T_2, maar heeft daar voor elke string (precies) twee keer zoveel stappen voor nodig

13.09

4) Stel dat er maar eindig veel strings $x_1, x_2, \ldots, x_n$ zijn waarvoor T oneindig loopt. Dan kunnen we een nieuwe Turing machine T' maken, die bij een invoer x eerst kijkt of x soms gelijk is aan $x_1, x_2, \ldots,$ of x_n. Als dit het geval is, verwerpt T'. Als dit niet het geval is, wordt Turingmachine T op de invoer x losgelaten. Merk op dat de controle of x gelijk is aan $x_1, x_2, \ldots,$ of x_n uit te voeren is, omdat dit maar eindig veel strings zijn.

Er geldt nu dat $L(T') = L(T) = L$ en dat T' stopt voor elke invoer x. Volgens de genoemde stelling zou L dan recursief zijn. Als L niet recursief is, moeten er dus oneindig veel strings zijn waarvoor T oneindig loopt.

16.27
6(a)

Er geldt:

$gn'(0) = 2^0 = 1. = C_1^0() =: g()$ is primitief recursief en de constante functie 2 ook primitief recursief is

$gn'(k+1) = 2^{k+1} = 2 * 2^k = 2 * gn'(k) = Mul(2, gn'(k)) \quad \forall k \geq 0$

Omdat volgens aanname de vermenigvuldiging Mul primitief recursief is, is $gn'(k+1)$ derhalve te schrijven als primitieve recursieve functie van (zijn eerste nul argumenten), de vorige waarde k van zijn laatste argument (waarbij we k overigens negeren) en zijn vorige functiewaarde $gn'(k)$.

De functie gn' wordt derhalve verkregen uit twee primitieve recursieve functies g en h met primitieve recursie, en is daarom zelf ook primitief recursief.

16.40
(b)(i)

Er geldt:

$gn^{m+1}(x_0, x_1, x_2, \ldots, x_m) = 2^{x_0} 3^{x_1} 5^{x_2} \ldots PrNo(m-1)^{x_{m-1}} PrNo(m)^{x_m} =$
$gn^m(x_0, x_1, x_2, \ldots, x_{m-1}) \circ PrNo(m)^{x_m}$

16.42
(b)(ii)

Voor eenvoudigere oplossing van 6(b)(ii), zie blz. ⑧

We gaan gn^{m+1} construeren met primitieve recursie, en bekijken daarom de functiewaarde als zijn laatste argument 0 is, en als dat $k+1$ is voor $k \geq 0$

Volgens (b)(i) is

$gn^{m+1}(x_0, x_1, x_2, \ldots, x_{m-1}, 0) = gn^m(x_0, x_1, x_2, \ldots, x_{m-1}) \circ PrNo(m)^0 =$
$gn^m(x_0, x_1, x_2, \ldots, x_{m-1}). \quad \forall x_0, x_1, x_2, \ldots, x_{m-1} \in \mathbb{N}$

Volgens aanname is dit een primitieve recursieve functie g van de eerste m argumenten.

Verder is

$gn^{m+1}(x_0, x_1, x_2, \ldots, x_{m-1}, k+1) = 2^{x_0} 3^{x_1} 5^{x_2} \ldots PrNo(m-1)^{x_{m-1}} PrNo(m)^{k+1}$
$= 2^{x_0} 3^{x_1} 5^{x_2} \ldots PrNo(m-1)^{x_{m-1}} \cdot PrNo(m)^k \circ PrNo(m) =$
$gn^{m+1}(x_0, x_1, x_2, \ldots, x_{m-1}, k) \circ PrNo(m) =$
$Mul(gn^{m+1}(x_0, x_1, x_2, \ldots, x_{m-1}, k), PrNo(m)). \quad \forall x_0, x_1, x_2, \ldots, x_{m-1}, k \in \mathbb{N}$

Nu is volgens aanname de vermenigvuldiging Mul primitief recursief.
Verder is voor vaste m de waarde $PrNo(m)$ constant, dat wil zeggen niet afhankelijk van de argumenten $x_0, x_1, x_2, \ldots, x_{m-1}, k$

We kunnen PrNo(m) dus schrijven als $C^m_{PrNo(m)}(x_0, x_1, x_2, \ldots, x_{m-1})$
Dit betekent dat $gn^{m+1}(x_0, x_1, x_2, \ldots, x_{m-1}, k+1)$ te schrijven is als primitieve recursieve functie h van zijn eerste m argumenten, de waarde k (die we negeren) en de vorige functiewaarde.

Kortom, gn^{m+1} wordt met primitieve recursie verkregen uit twee primitieve recursieve functies g en h, en is dus ook zelf primitief recursief.

16.5g.

3(b),ii) Alternatieve oplossing (met dank aan KK)

$S \rightarrow BA \mid ABBBAA$
voor $i=0$ | voor $i \geq 1$

$BA \rightarrow BBAA$	verhoog k
$AB \rightarrow AABBR$	verhoog i, met boodschapper R om ook k te verhogen
$BRB \rightarrow BBR$	loop naar rechts met R
$BRA \rightarrow BBAA$	verhoog ook k
$A \rightarrow a$ $B \rightarrow b$	rond af

Nog een andere oplossing (met dank aan diverse studenten)

$S \rightarrow aB_1SB_2a \mid SB_2a \mid Ma$

aB_1SB_2a: verhoog i en k; SB_2a: verhoog k; Ma: verhoog k $\Rightarrow k > i$

M is eigenlijk een B_2, de meest linkse

$B_1a \rightarrow aB_1$ $B_1b \rightarrow bB_1$ $B_1M \rightarrow bM$	loop met B_1 naar M en word daar een b, ofwel sorteer a^i en b^i
$aB_2 \rightarrow B_2a$ $bB_2 \rightarrow B_2b$ $MB_2 \rightarrow Mb$	loop met B_2 naar M en word daar een b, ofwel sorteer b^k en a^k
$M \rightarrow b$	klaar

Uitwerking tentamen Fundamentele Informatica 3, vrijdag 13 januari 2017

15:02

6(b)(ii) Alternatieve oplossing

Bij (a) hebben we aangetoond dat de functie $gn^1(x_0) = 2^{x_0}$ primitief recursief is. Op analoge wijze tonen we aan dat de functie $f(x) = a^x$ voor een vaste constante a (met $a \geq 1$) primitief recursief is:

$$f(0) = a^0 = 1 = C_1^0() =: g()$$
$$f(k+1) = a^{k+1} = a * a^k = a * f(k) = Mul(a, f(k)) \quad \forall k \geq 0$$

Omdat volgens aanname de vermenigvuldiging primitief recursief is, en de constante functie a ook primitief recursief is, is $f(k+1)$ dus te schrijven als een primitieve recursieve functie van de vorige waarde k van zijn enige = laatste argument (waarbij we k overigens negeren) en zijn vorige functiewaarde $f(k)$.

De functie f wordt nu verkregen uit twee primitieve recursieve functies g en h met primitieve recursie, en is daarom zelf ook primitief recursief.

Nu nemen we voor a de waarde $PrNo(m)$. Voor een gegeven m is $PrNo(m)$ een constant getal, zodat de functie $f(x) = (PrNo(m))^x$ primitief recursief is.

Bij (b)(i) zagen we dat

$$gn^{(m+1)}(x_0, x_1, x_2, \ldots, x_m) = gn^{(m)}(x_0, x_1, x_2, \ldots, x_{m-1}) * PrNo(m)^{x_m}$$

Als $gn^{(m)}$ primitief recursief is, is dit het product van twee primitieve recursieve functies van (projecties van) de argumenten $x_0, x_1, x_2, \ldots, x_m$ van $gn^{(m+1)}$. Omdat de vermenigvuldiging Mul volgens aanname primitief recursief is, is $gn^{(m+1)}$ dan ook primitief recursief

15.17.