# Oefenzittingen Automaten en Berekenbaarheid

Begeleider:

Johan Wittocx
Celestijnenlaan 200A, bureau 02.50
johan.wittocx@cs.kuleuven.be

Tien oefenzittingen:

1. Reguliere expressies en NFA's (p4–19)
2. NFA's en DFA's (p20–29)
3. Minimale DFA's en pompend lemma voor reguliere talen (p30–54)
4. Context-vrije talen, CFG's en PDA's (p55–72)
5. Pompend lemma voor context-vrije talen en ambiguiteit (p73–81)
6. Turing machines en beslisbaarheid (p82–95)
7. Onbeslisbaarheid (p96–116)
8. Veel-één reductie en recursieve functies (p117–127)
9. Herschrijfsystemen (p128–160)
10. Herhalingsoefeningen

(Gedeeltelijke) oplossingen van alle oefeningen zullen on-line beschikbaar zijn op
`http://people.cs.kuleuven.be/~bart.demoen/AB/index2010.html`

## Oefenzitting 1

1. Voor een string $w = w_1 w_2 \dots w_n$ duiden we de omgekeerde string $w_n \dots w_2 w_1$ aan met $\widehat{w}$ en voor een taal $L$ schrijven we $\widehat{L} = \{\widehat{w} \mid w \in L\}$. Als $L$ regulier is, is $\widehat{L}$ dan ook regulier?
2. Geef voor elk van de volgende talen over het alfabet $\{0, 1\}$ een reguliere expressie die die taal bepaalt. Geef ook een NFA die die taal aanvaardt.
   (a) $\{w \mid$ op elke oneven positie van w staat een $1\}$
   (b) $\{w \mid w$ bevat minstens twee 0'en en hoogstens één $1\}$
   (c) Het complement van $\{11, 111\}$
   (d) $\{w \mid w$ bevat evenveel maal de substring 01 als de substring $10\}$
3. Toon aan dat er voor elke $n \geq 1$ NFA's bestaan die de volgende talen aanvaarden.
   (a) $\{a^k \mid k$ is een veelvoud van $n\}$ over het alfabet $\{a\}$.
   (b) $\{x \mid x$ is de binaire voorstelling van een natuurlijk getal dat een veelvoud is van $n\}$

(14/10/2010) # Automaten en berekenbaarheid: oefeningenzitting 1

people.cs.kuleuven.be/~johan.wittocx/ab.html

## Oef 1

Zij E een reguliere expressie

Definieer dan de reguliere expressie $\overleftarrow{E}$ door

$\overleftarrow{E} = a$ als $E = a$

$\overleftarrow{E} = \varepsilon$ als $E = \varepsilon$

$\overleftarrow{E} = \phi$ als $E = \phi$

$\overleftarrow{E} = \overleftarrow{E_1} | \overleftarrow{E_2}$ als $E = E_1 | E_2$

$\overleftarrow{E} = \overleftarrow{E_2}\overleftarrow{E_1}$ als $E = E_1 E_2$

$\overleftarrow{E} = (\overleftarrow{E_1})^*$ als $E = E_1^*$

moet je zeker geven op het examen!

Als $L$ regulier is, dan bestaat er een E zodat: $L = L_E$

De taal $L_{\overleftarrow{E}}$ is ook regulier

We tonen aan dat $\hat{L} = L_{\overleftarrow{E}}$, wat bewijst dat $\hat{L}$ regulier is.

Basis: $E = a$ : $L_{\overleftarrow{E}} = L_a = \{a\} = \hat{L}$.

$E = \varepsilon$

$E = \phi$

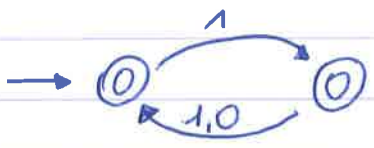
inductie $E = E_1 | E_2$ $L_{\overleftarrow{E}} = L_{\overleftarrow{E_1}|\overleftarrow{E_2}} = L_{\overleftarrow{E_1}} \cup L_{\overleftarrow{E_2}} = \hat{L}_{E_1} \cup \hat{L}_{E_2} = \hat{L}$

$E = E_1 E_2$

$E = E_1^*$

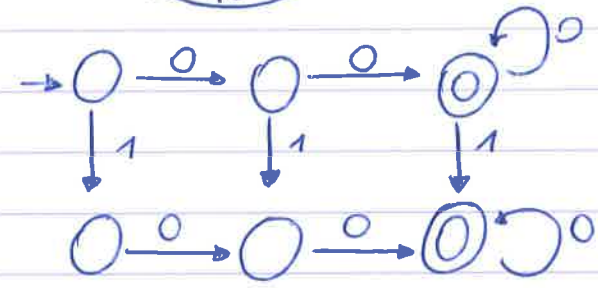
## Oef 2

a)




$(1(01)^*)^*(\varepsilon|1)$

b)




$1000^* | 0100^* | 00^*100^* | 000^*$

# Oplossingen voor oefenzitting 1

1. Ja. Als een regulier expressie die $L$ bepaalt van achter naar voren wordt gelezen bekom je een reguliere expressie die $\hat{L}$ bepaalt. De correctheid van dit argument kan eenvoudig door inductie bewezen worden.

2. (Alleen de reguliere expressies. De cursus geeft een algoritme om die om te zetten naar NFA's.)

   (a) $(1(0|1))^*(\epsilon|1)$

   (b) $(1000^* \mid 0100^* \mid 0^*0010^* \mid 000^*)$

   (c) $((0|1)^*0(0|1)^* \mid 1 \mid 11111^* \mid \epsilon)$

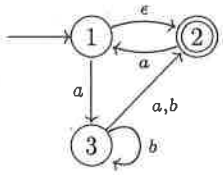
   (d) $(0(0|1)^*0 \mid 1(0|1)^*1 \mid 1 \mid 0 \mid \epsilon)$

3. (a) Voor elke $n \geq 1$ is $(a^n)^*$ een reguliere expressie die deze taal bepaalt.

   (b) Voor elke $n \geq 1$ aanvaardt de NFA $(Q, \{0,1\}, \delta, q_0, \{q_0\})$ deze taal, met $Q = \{q_0, \ldots, q_{n-1}\}$ en $\delta(q_i, a) = \{q_j\}$ waarbij $j = 2i + a \bmod n$. (Bewijs door middel van inductie dat deze oplossing correct is!)

# Oefenzitting 2

1. Zet om naar een DFA.




2. Een *all-paths*-NFA $(Q, \Sigma, \delta, q_0, F)$ verschilt van een NFA doordat een string $w$ enkel aanvaard wordt als *elk* pad dat je met die string kan volgen in de grafe-voorstelling van $(Q, \Sigma, \delta, q_0, F)$ eindigt in een eindtoestand, en er minstens één pad is dat je kan volgen. Formeel zeggen we dat $w$ aanvaard wordt als

   - voor elke mogelijke schrijfwijze $w = y_1 y_2 \dots y_m$, $y_i \in \Sigma_\varepsilon$ en elke sequentie $r_0, r_1, \dots, r_m$ van toestanden zodanig dat $r_0 = q_0$ en $r_{i+1} \in \delta(r_i, y_{i+1})$, geldt dat $r_m \in F$;
   - er minstens één schrijfwijze $w = y_1 y_2 \dots y_m$, $y_i \in \Sigma_\varepsilon$ bestaat zodanig dat er een sequentie $r_0, \dots, r_m$ van toestanden bestaat met $r_0 = q_0$ en $r_{i+1} \in \delta(r_i, y_{i+1})$.

   Toon aan dat een taal $L$ regulier is als en slechts als ze herkend wordt door een all-paths-NFA.

3. Neem 2 strings $s$ $(s_1 s_2 ... s_n)$ en $t$ $(t_1 t_2 ... t_m)$ over hetzelfde alfabet (met minstens 2 tekens). Met die 2 strings kan je een verzameling $s \,||\, t$ van nieuwe strings maken door $s$ en $t$ te mengen, zoals je met 2 pakken speelkaarten zou doen: in een string uit $s \,||\, t$ zitten de $s_i$ in de juiste volgorde en de $t_j$ ook, maar niet noodzakelijk bij elkaar. Bijvoorbeeld:

$$ab \,||\, c = \{abc, acb, cab\}$$

   We kunnen ook twee talen $L_1$ en $L_2$ *mengen* als volgt:

$$L_1 \,||\, L_2 = \{w \mid \text{er bestaat een } s \in L_1 \text{ en een } t \in L_2 \text{ zodanig dat } w \in s \,||\, t\}$$

   Gegeven twee reguliere talen $L_1$ en $L_2$, bewijs dat $L_1 \,||\, L_2$ ook regulier is.

4. Noem een string $x$ een *prefix* van een string $y$ als er een string $z$ bestaat zodanig dat $y = xz$. Als bovendien $x \neq y$, dan noemen we $x$ een *echte prefix* van $y$. Toon aan dat de klasse van reguliere talen gesloten is onder de volgende operaties.

   (a) NOPREFIX$(L) = \{w \in L \mid \text{geen enkele echte prefix van } w \text{ zit in } L\}$

   (b) NOEXTEND$(L) = \{w \in L \mid w \text{ is geen echte prefix van een string in } L\}$

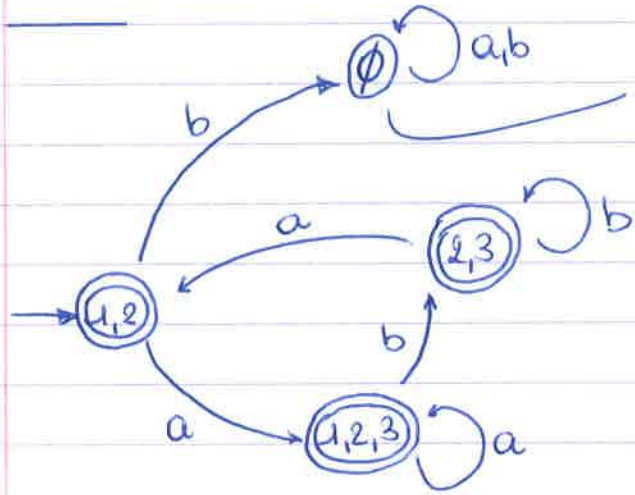
5. Zij $L$ een reguliere taal over een alfabet $\Sigma$. Bewijs dat

$$L_{\frac{1}{2}} = \{x \in \Sigma^* \mid \text{er bestaat een } y \in \Sigma^* \text{ zodanig dat } |x| = |y| \text{ en } xy \in L\}$$

   ook regulier is.

21/10/2010 Oefeningenzitting 2

Oef 1




deze toestand mag weg als je een minimale DFA wil

Oef 2

Als L regulier is, dan bestaat er een DFA die L herkent
Die DFA is ook een all-paths NFA die L herkent
Een all-paths NFA kan omgezet worden naar een DFA die dezelfde taal herkent.
Verander de definitie op p27 van $F_d$ door $\{S \mid S \neq \phi \text{ en } S \subseteq F_q\}$

Oef 3

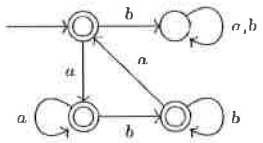
Zij $(Q_1, \Sigma, \delta_1, q_{s_1}, F_1)$ en $(Q_2, \Sigma, \delta_2, q_{s_2}, F_2)$ twee DFA's die $L_1$ resp $L_2$ herkennen
Dan herkent de NFA $(Q_1 \times Q_2, \Sigma, \delta, (q_{s_1}, q_{s_2}), F_1 \times F_2)$ met
$\delta((q_1, q_2), a) = \{(\delta_1(q_1, a), q_2), (q_1, \delta_2(q_2, a))\}$ de taal $L_1 \| L_2$

Bewijs Toon door inductie aan dat

$(q_1, q_2) \in \delta^*(w)$ asa $\exists w_1, w_2 : w \in w_1 \| w_2$
$\delta_1^*(w_1) = q_1$
$\delta_2^*(w_2) = q_2$

# Oplossingen voor oefenzitting 2

1. Door het algoritme op blz. 27 uit de cursus te volgen krijg je (na het verwijderen van onbereikbare toestanden) de volgende DFA.




2. Merk op dat een DFA een all-paths-NFA is. Dus bestaat er voor elke reguliere taal een all-paths-NFA die die taal herkent. De omgekeerde inclusie kan je bewijzen door de constructie op blz. 27 uit de cursus een klein beetje te wijzigen en zo een all-paths-NFA om te zetten naar een DFA die dezelfde taal herkent. Het enige wat moet veranderen is de defintie van $F_d$. Een toestand van de DFA die geconstrueerd wordt is nu een eindtoestand als en slechts als hij niet leeg is en enkel eindtoestanden van de NFA bevat. Formeel: $F_d = \{S \mid \emptyset \neq S \subseteq F_n\}$.

3. Zij $(Q_1, \Sigma, \delta_1, s_1, F_1)$ en $(Q_2, \Sigma, \delta_2, s_2, F_2)$ DFA's die respectievelijk $L_1$ en $L_2$ herkennen. Dan herkent de NFA $(Q_1 \times Q_2, \Sigma, \delta_{||}, (s_1, s_2), F_1 \times F_2)$ de taal $L_1 \,||\, L_2$. Hierbij is $\delta_{||}$ gedefinieerd door

$$\delta_{||}((q_1, q_2), a) = \{(q_1', q_2), (q_1, q_2')\}$$

met $q_1' = \delta_1(q_1, a)$ en $q_2' = \delta_2(q_2, a)$.

Noem de automaat die we geconstrueerd hebben $M$ en noem de oorspronkelijke automaten $M_1$ en $M_2$. Om te bewijzen dat onze oplossing correct is moet we aantonen dat $w \in L_M \Rightarrow w \in L_1 \,||\, L_2$ en dat $w \in L_M \Leftarrow w \in L_1 \,||\, L_2$. We tonen richting $\Rightarrow$ aan en laten richting $\Leftarrow$ als oefening. Merk eerst op dat het voldoende is om de volgende eigenschap aan te tonen:

> Als we in $M$ met een string $w$ vanuit de starttoestand in toestand $(q_1, q_2)$ kunnen geraken, dan is $w$ een mengeling van strings $w_1$ en $w_2$ zodanig dat we in $M_1$ met $w_1$ vanuit de starttoestand $s_1$ in toestant $q_1$ geraken en in $M_2$ met $w_2$ vanuit starttoestand $s_2$ in $q_2$ geraken.

Inderdaad, als deze eigenschap geldt, dan leiden we af dat voor elke $w \in L_M$ dat $w$ een mengeling is van twee strings $w_1$ en $w_2$ die aanvaard worden door respectievelijk $M_1$ en $M_2$. We tonen de eigenschap aan door inductie.

**basis:** Als $w = \epsilon$, dan geraken we met $w$ in toestand $(s_1, s_2)$. We nemen $w_1 = \epsilon$ en $w_2 = \epsilon$. Dan geldt $w \in \epsilon \,||\, \epsilon$ en met $w_1$ geraken we in $s_1$ en met $w_2$ in $s_2$.

**inductie** Stel dat $w = w'a$ en dat we met $w$ in $(q_1, q_2)$ geraken. Dat wil zeggen dat we met $w'$ in een toestand $(q_1', q_2')$ geraken zodanig dat $(q_1, q_2) \in \delta_{||}((q_1', q_2'), a)$. Uit de definitie van $\delta_{||}$ volgt dat ofwel $q_1' = q_1$ ofwel $q_2' = q_2$. We veronderstellen dat $q_1 = q_1'$ (het andere geval is analoog). Uit de definitie van $\delta_{||}$ halen we dan dat $q_2 = \delta_2(q_2', a)$. Door de inductiehypothese weten we dat $w' \in w_1' \,||\, w_2'$ met $w_1'$ zodanig dat we in $M_1$ vanuit de starttoestand naar $q_1' = q_1$ geraken en met $w_2'$ in $M_2$ vanuit de starttoestand naar $q_2'$. Er volgt dat $w \in w_1' \,||\, w_2'a$ en met $w_1'$ geraken we in $q_1$ en met $w_2'a$ geraken we in $q_2$.

4. Zij $M = (Q, \Sigma, \delta, q_0, F)$ een DFA die $L$ herkent.

   (a) Zij $M' = (Q, \Sigma, \delta', q_0, F)$ de DFA waarbij

$$\delta'(q, a) = \begin{cases} \{\delta(q, a)\} & \text{als } q \notin F \\ \emptyset & \text{als } q \in F \end{cases}$$

Merk op dat $L(M') \subseteq L$. We tonen nu aan dat $M'$ de taal NOPREFIX$(L)$ herkent. Eerst tonen we aan dat $L(M') \subseteq$ NOPREFIX$(L)$. Als $L(M')$ leeg is, is dit voldaan. Als $L(M')$ niet leeg is, neem dan een string $w = z_1 \ldots z_n$ die door $M'$ herkend wordt. Stel dat we $w$ kunnen schrijven als $xy$, met $x \in L$. Omdat $w$ door $M'$ aanvaard wordt, bestaat er een sequentie van toestanden $r_0, \ldots, r_n$ zodanig dat $r_0 = q_0$, $r_n \in F$ en $r_{i+1} = \delta'(r_i, z_{i+1})$. Door de constructie

van $M'$ volgt nu dat $r_j \notin F$ voor $0 \leq j < n$. Stel $x = z_1 \dots z_m$. Omdat $x \in L$, moet $r_m \in F$. Bijgevolg zal $m = n$, $y = \varepsilon$ en dus $w \in \text{NOPREFIX}(L)$.

Nu moeten we nog aantonen dat $L(M') \supseteq \text{NOPREFIX}(L)$. Neem $w = z_1 \dots z_n \in \text{NOPREFIX}(L)$. Dan geldt $w \in L$ en dus bestaat er een sequentie $r_1, \dots, r_n$ van toestanden zodanig dat $r_0 = q_0$, $r_n \in F$ en $r_{i+1} = \delta(r_i, z_{i+1})$. Voor een $r_i$ met $i \neq n$ geldt dat $r_i \notin F$. Als dat zo was. zou immers $z_0 \dots z_i$ een echte prefix van $w$ zijn. Hieruit volgt dat $w \in L(M')$.

(b) Zij $M'' = (Q, \Sigma, \delta, q_0, F'')$ de DFA waarbij

$$F'' = \{q \in F \mid \text{Er bestaat geen pad van lengte} > 0 \text{ van } q \text{ naar een } q' \in F\}$$

We tonen aan dat $M''$ de taal NOEXTEND($L$) herkent. Eerst tonen we aan dat $L(M'') \subseteq \text{NOEXTEND}(L)$. Als $L(M'')$ leeg is, is dit voldaan. Als $L(M'')$ niet leeg is, neem dan een string $w$ die door $M'$ herkend wordt. Veronderstel dat voor een string $y$ geldt dat $wy \in L$. Schrijf $w = z_1 \dots z_m$ en $y = z_{m+1} \dots z_n$. Dan bestaat er een sequentie $r_0, \dots, r_m, \dots, r_n$ van toestanden. zodanig dat $r_0 = q_0$, $\delta(r_i, z_{i+1}) = r_{i+1}$, $r_m \in F''$ en $r_n \in F$. Als $m \neq n$, dan bestaat er een pad met lengte $> 0$ van $r_m$ naar $r_n$, wat vanwege de constructie van $F''$ niet mogelijk is. Bijgevolg moet $y = \varepsilon$.

Nu moet je nog aantonen dat $L(M'') \supseteq \text{NOEXTEND}(L)$.

5. Zij $(Q, \Sigma, \delta, q_0, F)$ een DFA die $L$ herkent. Dan wordt $L_{\frac{1}{2}}$ herkend door de NFA $(Q^3 \cup \{s\}, \Sigma, \delta', s, F')$, waarbij $s$ een nieuwe toestand is, $F' = \{(m, m, q) \mid m \in Q, q \in F\}$, en $\delta'$ gegeven wordt door

$$\delta'(s, a) = \begin{cases} \{(q_0, m, m) \mid m \in Q\} & \text{als } a = \varepsilon \\ \emptyset & \text{in alle andere gevallen} \end{cases}$$

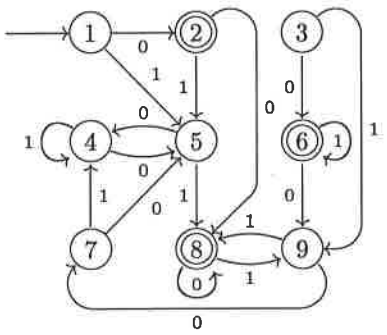
en

$$\delta'((q_1, m, q_2), a) = \{(\delta(q_1, a), m, \delta(q_2, b) \mid b \in \Sigma\}.$$

Intuitief herkent deze NFA een string $x$ door eerst te gokken waar je met $x$ zou geraken in de oorspronkelijke DFA. Die gok, zeg $m$, wordt opgeslaan in de tweede toestand van elk tripel van toestanden in de NFA. Vervolgens wordt de oorspronkelijke DFA op $x$ uitgevoerd (wat wordt bijgehouden in de eerste toestand), en wordt de oorspronkelijke DFA op een willekeurige string $y$ uitgevoerd vanuit toestand $m$. Als de NFA uiteindelijk in een toestand $(m, m, q)$ uitkomt, was de gok in het begin juist: $x$ eindigt inderdaad in toestand $m$ op de oorspronkelijke DFA. Met de string $y$, die even lang is als $x$, kom je vanuit $m$ in $q$ terecht. Als bovendien $q \in F$, dan volgt dat $xy \in L$, en dus $x \in L_{\frac{1}{2}}$.

# Oefenzitting 3

1. Minimaliseer de volgende DFA




2. Zij $\Sigma_1 = \{0,1\}, \Sigma_2 = \{a\}$ en $\Sigma_3 = \{a,b,c\}$. Toon aan dat de volgende talen niet regulier zijn.
   (a) $\{w \in \Sigma_1^* \mid |w|$ is even en de 1'en in $w$ komen enkel voor in de laatste helft van $w\}$
   (b) $\{a^{(2^n)} \in \Sigma_2^* \mid n \geq 0\}$
   (c) $\{a^n \in \Sigma_2^* \mid n$ is een priemgetal$\}$
   (d) $\{0^m 1^n \in \Sigma_1^* \mid m \neq n\}$
   (e) $\{w \in \Sigma_1^* \mid w$ is geen palindroom$\}$
   (f) $\{a^i b^j c^k \in \Sigma_3^* \mid i,j,k \geq 0$ en als $i = 1$, dan moet $j = k\}$

3. Definieer de operatie `verwijder_middelste` op talen door

$$\texttt{verwijder\_middelste}(L) = \{xz \mid \exists y : xyz \in L \text{ and } |x| = |y| = |z|\}.$$

   Toon aan dat als $L$ regulier is, `verwijder_middelste`$(L)$ niet noodzakelijk regulier is.

28/10/2016

# Oefeningenzitting 3

## Oef 2

Als $L$ regulier is, dan bestaat er een $p \in \mathbb{N}$ zodat voor elke $w \in L$ met $|w| > p$ er een opdeling van $w$ in 3 stukken $x, y, z$ bestaat $(\forall w x : w = xyz)$, zodat:

① $xy^i z \in L \quad \forall i \in \mathbb{N}$

② $|y| > 1$

③ $|xy| \leq p$

$L = \{ ww \mid w \in \{0,1\}^k \}$

Kies een willekeurige $p \in \mathbb{N}$.

Neem $w = 0^p 1 0^p 1$. Dan: $w \in L$ en $|w| = 2p + 2 > p$

Kies strings $xyz \in \{0,1\}^*$ zodat $w = xyz$. Veronderstel dat $xyz$ aan ①, ② en ③ voldoet.

Omdat $|y| > 1$, bevat $y$ minstens 1 cijfer.

Omdat $|xy| \leq p$ en de eerste $p$ cijfers in $w$ zijn 0'en

→ $y$ bevat enkel 0'en.

Omdat $xyz \in L$, moet $|xyz|$ even zijn. Dus moet $|y|$ even zijn

Dus bevat $y$ minstens 2 0'en.

Er volgt dat de eerste helft van $xyyz$ enkel 0'en bevat.

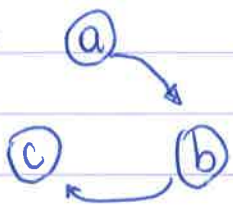
De tweede helft bevat echter 2 1'en, dus $xyyz \notin L$

We besluiten dat $L$ niet regulier is. ∎

Een bewijs met behulp van het pompend lemma heeft telkens deze structuur.

## Oef 1

Buurmatrix:




| | a | b | c |
|---|---|---|---|
| a | 0 | 1 | 0 |
| b | 0 | 0 | 1 |
| c | 0 | 0 | 0 |

Oef 2 (vervolg)

---

a) $L = \{ \omega_1 \omega_2 \mid |\omega_1| |\omega_2| \wedge \omega_1 \in \{0\}^* \wedge \omega_2 \in \{0,1\}^* \}$

Kies een willekeurige $p \in \mathbb{N}$. Neem $\omega = 0^{p+1} 1^{p+1}$

Dan $\omega \in L$ en $|\omega| = 2p > p$

kies strings $x, y, z \in \{0,1\}^*$ zodat $\omega = xyz$

Veronderstel dat $xyz$ aan ①, ② en ③ voldaat.

Omdat $|y| > 1$, bevat $y$ minstens 1 cijfer

$|xy| \leq p$ dus bevat $y$ enkel 0'en. Omdat $xz \in L$ moet $|xz|$ even zijn. Dus moet $|y|$ even zijn. Dus bevat $y$ minstens 2 cijfers.

Er volgt dat de eerste hlft van $xz$ 1'en bevat

Dus $xz \notin L$

We besluiten dat $L$ niet regulier is. ∎

b) $L = \{ a^{2^n} \mid n \in \mathbb{N} \}$

Kies een willekeurige $p \in \mathbb{N}$. Neem $\omega = a^{2^{p+1}}$. Dan $\omega \in L$ en $|\omega| = a^{2^p} > p$

kies strings $x, y, z \in \{0,1\}^*$ zodat $\omega = xyz$

Veronderstel dat $xyz$ aan ①, ② en ③ voldaat.

Omdat $|y| > 1$ bevat $y$ minstens 1 a. Dus $|xz| < |\omega| = 2^{p+1}$

Omdat $|xy| \leq p$ geldt $|y| \leq p$. $|xz| \geq 2^{p+1} - p > 2^{p+1} - 2^p = 2^p$

→ $xy$ ligt dus tussen 2 strings van $L$.

$2^p < |xy| < 2^{p+1}$

# Oplossingen voor oefenzitting 3

1. De toestanden 3 en 6 kunnen weggelaten worden. Verder kunnen 2 en 8 samengevoegd worden, evenals 5 en 9 en ook de twee toestanden 4 en 7. De toestand 1 blijft apart.

2. (De taal waarvoor aangetoond moet worden dat ze niet regulier is telkens aangeduid met $L$)

   (a) Veronderstel $L$ regulier is met pomplengte $d$. Elke indeling $xyz$ van de string $0^d1^d$ die aan de drie voorwaarden van het pompend lemma voldoet is zodanig dat $y$ enkel nullen bevat en niet leeg is. Het is duidelijk dat $xz$ dan niet meer in de taal zit.

   (b) Veronderstel dat $L$ regulier is met pomplengte $d$. Verdeel de string $a^{(2^{d+1})}$ in $xyz$ zodanig dat $|xy| \leq d$ en $|y| > 0$. Hieruit volgt dat $|xz| < 2^{d+1}$ en $|y| \leq d$. We gaan nu aantonen dat ook $|xz| > 2^d$. Dat laat zien dat $|xy|$ niet tot $L$ kan behoren, aangezien het 'tussen twee strings van $L$ ligt'. En dus hebben we een contradictie via het pompend lemma. Het bewijs dat $|xz| > 2^d$ volgt eenvoudig uit de observatie dat $2^d > d$. $|xz| \geq 2^{d+1} - d > 2^{d+1} - 2^d = 2^d$.

   (c) Veronderstel dat $L$ regulier is met pomplengte $d$ en zij $p$ het eerste priemgetal groter dan $d$. Verdeel $a^n$ in $xyz$ volgens de voorwaarden van het pompend lemma. Dan kan $|xz|$ niet nul zijn, want anders behoorde $a^{2n}$ tot $L$. Er geldt nu dat $|xy^{|xz|}z| = |x| + (|xz| \cdot |y|) + |z| = |xz| \cdot (1 + |y|)$. Dit is duidelijk geen priemgetal. Dus behoort $xy^{|xz|}z$ niet tot $L$. Contradictie.

   (d) Stel dat $L$ regulier is. Dan moet $L^c \cap (0^*1^*)$ ook regulier zijn. Contradictie.

   (e) Stel dat $L$ regulier is. Dan is $L^c \cap (0^*10^*)$ ook regulier. Van deze taal kan je echter eenvoudig aantonen met het pompend lemma dat ze niet regulier is.

   (f) Stel dat $L$ regulier is. Dan is $L \cap (ab^*c^*)$ ook regulier. Contradictie.

3. Neem $L$ de taal over $\Sigma = \{0, 1, \#\}$ die gegenereerd wordt door de reguliere expressie $0^*\#1^*$. Dan is

$$\texttt{verwijder\_middelste}(L) \cap 0^*1^* = \{0^n1^n \mid n \in \mathbb{N}\}.$$

   Die taal is niet regulier, dus is $\texttt{verwijder\_middelste}(L)$ evenmin regulier.

$|uxy| \leq p$
$|uy| > 0$
$\Rightarrow |uxz|$ kan niet nul zijn want anders faalt

# Oefenzitting 4

1. Toon aan dat de klasse van context vrije talen gesloten is onder concatenatie en ster.
2. Bewijs dat de doorsnede van een reguliere taal met een context-vrije taal context-vrij is.
3. Geef voor elk van de volgende talen een CFG die die taal genereert.
   (a) $\{w \in \{0,1\}^* \mid$ de lengte van $w$ is oneven en het middelste symbool is een $0\}$
   (b) $\{w \in \{0,1\}^* \mid w$ bevat strikt meer 1'en dan 0'en$\}$
   (c) $\{w \in \{0,1\}^* \mid w$ bevat tweemaal zoveel 1'en als 0'en$\}$
   (d) $\{w_1 \# w_2 \# \ldots \# w_n \mid n \geq 1$, elke $w_i \in \{0,1\}^*$ en er bestaat een $i$ en een $j$ waarvoor $w_i = \widehat{w_j}\}$
4. Geef voor de eerste twee talen uit (3) een PDA (volgens de definitie op blz. 64 — hoogstens één symbool tegelijkertijd op de stack duwen) die die taal herkent.
5. Een CFG $(V, \Sigma, R, S)$ is *rechts-lineair* als elke regel in $R$ van de vorm $A \to wB$ of $A \to w$ is, waarbij $A$ en $B$ niet-eindsymbolen zijn, en $w$ een (mogelijks lege) string over $\Sigma$ is. Welke verzameling van talen wordt herkend door rechts-lineaire grammatica's?
6. Toon aan dat elke context-vrije taal over het alfabet $\{1\}$, regulier is.
7. Toon aan dat $\{x \# y \mid x, y \in \{0,1\}^*$ en $x \neq y\}$ een context-vrije taal is over $\{0, 1, \#\}$.

4/11/2010

# Oefeningenzitting 4

## Oef 3

b) 00111 — wel
11011 — wel
11000111 — wel
00011010110 — wel
00110001 — niet

## Oef 6

Zie stelling van Parith (p72)

## Oef 1

Concatenatie

$S \rightarrow S_1 S_2$

Kleene ster

$S \rightarrow SS \mid \varepsilon$

## Oef 3

a) $S \rightarrow ASA \mid 0$
$A \rightarrow 1 \mid 0$

b) $S \rightarrow A1A$
$A \rightarrow AA \mid 1A0 \mid 0A1 \mid 1 \mid \varepsilon$

c) $S \rightarrow 0S1S1S \mid 10S1S \mid 11S0S \mid \varepsilon$
of $S \rightarrow SS \mid 0S1S1 \mid 1S0S1 \mid 1S1S0 \mid \varepsilon$

d) $S \rightarrow A\#S \mid S\#A \mid B$
$A \rightarrow 0A \mid 1A \mid \#A \mid \varepsilon$
$B \rightarrow 0B0 \mid 1B1 \mid \#A\# \mid \# \mid \varepsilon$

# Oplossingen voor oefenzitting 4

1. Eenvoudig op te lossen met behulp van CFGs. Net zoals op p75 van de cursus aangetoond wordt dat de klasse van context-vrije talen gesloten is onder de unie.

2. Zij $R$ een reguliere taal en $C$ een context-vrije taal over een alfabet $\Sigma$. Zij $(Q, \Sigma, \delta, q_0, F)$ een DFA die $R$ herkent en $(Q', \Sigma, \Gamma', \delta', q_0', F')$ een PDA die $C$ herkent. Dan wordt $C \cap R$ herkent door de PDA $(Q \times Q', \Sigma, \Gamma', \delta'', (q_0, q_0'), F \times F')$, waarbij $\delta''((q, q'), a, b) = (\delta(q, a), \delta'(q', a, b))$ als $a \neq \epsilon$ en $\delta''((q, q'), \epsilon, b) = (q, \delta'(q', \epsilon, b))$.

3. Ga na dat de volgende grammatica's correcte oplossingen vormen. Dat is niet altijd triviaal!

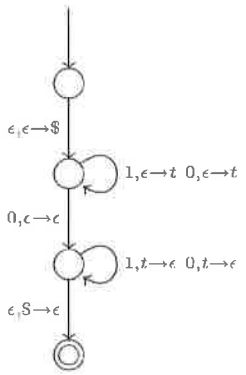
   (a) $S \to ASA \mid 0$
   $A \to 0 \mid 1$

   (b) $S \to A1A$
   $A \to AA \mid 0A1 \mid 1A0 \mid 1 \mid \epsilon$

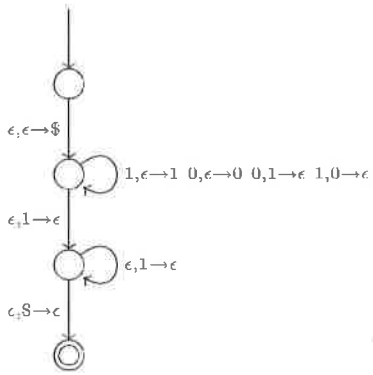
   (c) $S \to 11S0S \mid 10S1S \mid 0S1S1S \mid \epsilon$

   (d) $S \to S\#A \mid A\#S \mid B$
   $A \to 0A \mid 1A \mid A\#A \mid \epsilon$
   $B \to 0B0 \mid 1B1 \mid \#A\# \mid \epsilon \mid \#$

4. (a)




   (b)



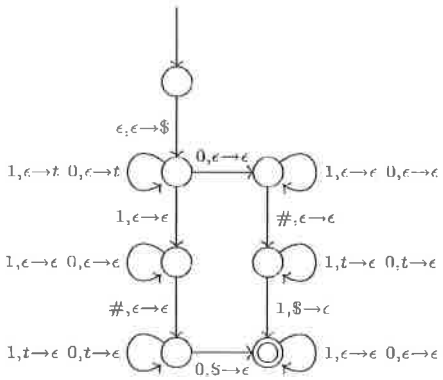

5. Antwoord: de reguliere talen. Een eenvoudige manier om dit aan te tonen bestaat uit een NFA om te zetten in een rechts lineaire grammatica en vice versa. Hierbij worden niet-terminalen van de grammatica omgezet in toestanden van de NFA, en regels $A \to a_1 \ldots a_n B$ worden paden $A \xrightarrow{a_1} A_1 \xrightarrow{a_2} \cdots \xrightarrow{a_n} A_n \xrightarrow{\epsilon} B$ in de DFA, waarbij $A_1, \ldots, A_n$ hulptoestanden zijn.

6. Voor deze oefening heb je de stelling van Parikh nodig (zie blz. 72). Merk op dat voor elke taal $L$ over $\{1\}$ er precies één taal $L'$ over $\{1\}$ is zodanig dat $Ord(L') = L$, namelijk $L' = L$. Aangezien Parikh zegt dat er voor elke context-vrije taal, er een reguliere taal bestaat met dezelfde orde, geldt nu dat elke context-vrije taal over $\{1\}$ ook regulier is.

7. Construeer een PDA die deze taal herkent. Een mogelijkheid om dat te doen is als volgt. Construeer een PDA voor $\{x\#y \mid |x| \neq |y|\}$ en een voor

   $\{x\#y \mid$ het $n$'de symbool van $x$ verschilt van het $n$'de symbool van $y\}$

   en neem dan de unie. De eerste van deze twee subPDA's is eenvoudig te construeren. De andere ziet er als volgt uit



# Oefenzitting 5

1. Welke van de volgende talen zijn context-vrij, welke niet? Bewijs je antwoord.

   (a) $\{w \in \{0,1\}^* \mid w \text{ is een palindroom}\}$

   (b) $\{0^n\#0^{2n}\#0^{3n} \mid n \geq 0\}$

   (c) $\{w \in \{a,b,c\}^* \mid w \text{ bevat een gelijk aantal } a\text{'s, } b\text{'s en } c\text{'s}\}$

   (d) $\{xy \mid x,y \in \{0,1\}^* \text{ en } |x| = |y|, \text{ maar } x \neq y\}$ p 75

   (e) $\{w\#x \mid w,x \in \{0,1\}^* \text{ en } w \text{ is een substring van } x\}$

   (f) $\{a^i b^j c^k \mid 0 \leq i \leq j \leq k\}$

2. Een $k$-PDA is een PDA met $k$ stacks. Een 0-PDA is dus een NFA en een 1-PDA een gewone PDA.

   (a) Toon aan dat 2-PDA's strikt sterker zijn dan 1-PDA's. (Met andere woorden, toon aan dat de klasse van talen die met een 1-PDA te herkennen zijn een strikte deelverzameling is van de klasse van talen die met een 2-PDA te herkennen zijn.)

   (b) Toon aan dat 3-PDA's even sterk zijn als 2-PDA's.

3. Zij

$$\Sigma = \{\textbf{if, condition, then, else, a:=1}\}$$
$$V = \{\langle\text{STMT}\rangle, \langle\text{IF-THEN}\rangle, \langle\text{IF-THEN-ELSE}\rangle, \langle\text{ASSIGN}\rangle\}$$

   en zij $G = (V, \Sigma, R, \langle\text{STMT}\rangle)$ de grammatica met regels

$$\begin{aligned}
\langle\text{STMT}\rangle &\to \langle\text{ASSIGN}\rangle \mid \langle\text{IF-THEN}\rangle \mid \langle\text{IF-THEN-ELSE}\rangle \\
\langle\text{IF-THEN}\rangle &\to \textbf{if condition then } \langle\text{STMT}\rangle \\
\langle\text{IF-THEN-ELSE}\rangle &\to \textbf{if condition then } \langle\text{STMT}\rangle \textbf{ else } \langle\text{STMT}\rangle \\
\langle\text{ASSIGN}\rangle &\to \textbf{a:=1}
\end{aligned}$$

   Toon aan dat $G$ ambigu is en construeer een niet-ambigue grammatica die dezelfde taal herkent.

   ~ dangling else

(18/11/2010) # Oefeningenzitting 5

## Oef 1

Als $L$ contextvrij is, dan $\exists p \in \mathbb{N}$ zodat $\forall w \in L$ met $|w| > p$ geldt dat $w$ geschreven kan worden als $uvxyz$ met:
- $uv^i xy^i z \in L \quad \forall i \in \mathbb{N}$
- $|vy| > 0$
- $|vxy| \leq p$

f) $L = \{ a^i b^j c^k \mid 0 \leq i \leq j \leq k \}$

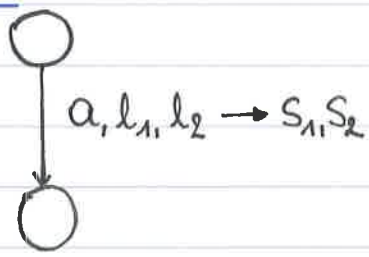
Stel: $L$ is contextvrij en heeft pomplengte $P$.
Neem $w = a^P b^P c^P$. Dan: $w \in L$ en $|w| \geq P$.
Dus kan $w$ geschreven worden als $uvxyz$ volgens de voorwaarden van het lemma.
- $uv^2xy^2z \in L$, dus bevatten $v$ en $y$ telkens maar 1 soort symbool.
- $uv^2xy^2z \in L$, dus kan $v$ geen $a$'s bevatten. Daaruit volgt dat $y$ ook geen $a$'s bevat.
- $uxz \in L$, dus bevat $y$ geen $c$'s en bevat $v$ ook geen $c$'s.
- $|vy| > 0$, dus bevatten $v$ of $y$ minstens één $c$.

En volgt nu dat $uv^2xy^2z$ meer $b$'s dan $c$'s bevat en dus niet tot $L$ behoort
= CONTRADICTIE.

a) $L = \{ w \in \{0,1\}^* \mid w \text{ is een palindroom} \}$

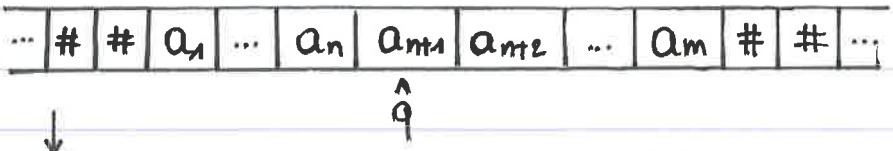
Stel: $L$ is contextvrij en heeft pomplengte $P$
Neem $w = 0^P 1^P 0^P$. Dan $w \in L$ en $|w| \geq p$.
Dus kan $w$ geschreven worden als $uvxyz$ volgens de voorwaarden van het lemma
- $uv^2xy^2z \in L$, dus kan $v$ geen $0$ bevatten. Daaruit volgt dat $y$ ook geen $0$'en bevatten (en omgekeerd), anders staan er achteraan nooit evenveel $0$'en als vooraan.
- $|vy| > 0$, dus $v$ of $y$ moet minstens een $1$ bevatten

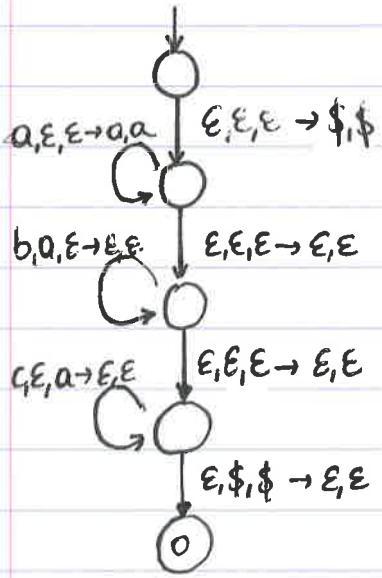
# Oef 2

a) 


$\{a^m b^m c^m \mid m \in \mathbb{N}\}$ = niet contextvrije taal wel door 2-PDA herkend.

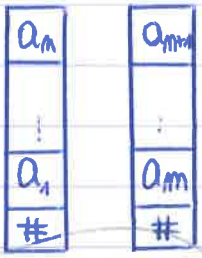
⇒ 2-PDA sterker dan 1-PDA.




TM: $q, a_{n+1} \rightarrow q', b, R$

2-PDA: q → q' met ε, ε, a_{n+1} → b, ε



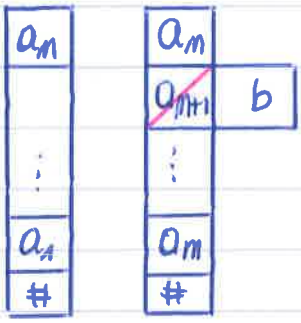

TM: $q, a_{n+1} \rightarrow q', b, L$

2-PDA: q → ○ → q' met ε, ε, a_{n+1} → ε, b ; ε, am, ε → ε, am

b) TM ≥ 3-PDA ≥ 2-PDA ≥ TM


naar rechts:


naar links:



# Oplossingen voor oefenzitting 5

1. ($L$ is steeds de taal uit de opgave)

   (a) $L$ is context-vrij. Schrijf een CFG om dat te bewijzen.

   (b) Veronderstel dat $L$ context-vrij is. Zij $p$ de pomplengte voor $L$. We gaan aantonen dat de string $w = 0^p\#0^{2p}\#0^{3p}$ niet gepompt kan worden. Verdeel $w$ in $uvxyz$ zodanig dat $|vy|$ niet leeg is. Als we willen dat $uv^2xy^2z$ in $L$ zit, dan mag noch $v$, noch $y$ een # bevatten. Dus is er een van de segmenten $0^p$, $0^{2p}$ of $0^{3p}$ disjunct met $v$ en met $y$. Maar dan bevat $uv^2xy^2z$ de nullen niet in de juiste verhoudingen. Contradictie.

   (c) $L \cap a^*b^*c^* = \{a^nb^nc^n \mid n \in \mathbb{N}\}$ en dat is niet context-vrij (zie cursus p74). Dus kan $L$ ook niet context-vrij zijn (zie oefening 2 van oefenzitting 4).

   (d) $L = L_1 \cap L_2$ waarbij $L_1 = \{w \in \{0,1\}^* \mid w \text{ is even}\}$ en $L_2 = \overline{\{ww \mid w \in \{0,1\}^*\}}$. Aangezien $L_1$ regulier is (eenvoudige oefening) en $L_2$ context-vrij (cursus p75), is $L$ context-vrij.

   (e) Pas het pompend lemma toe om aan te tonen dat deze taal niet context-vrij is. Je kan bijvoorbeeld de string $0^p1^p\#0^p1^p$ gebruiken, waarbij $p$ de pomplengte is.

   (f) Pas het pompend lemma toe om aan te tonen dat deze taal niet context-vrij is. Je kan bijvoorbeeld de string $a^pb^pc^p$ gebruiken, waarbij $p$ de pomplengte is.

2. (a) Het is eenvoudig om een 2-PDA te construeren die $\{a^nb^nc^n \mid n \in \mathbb{N}\}$ herkent.

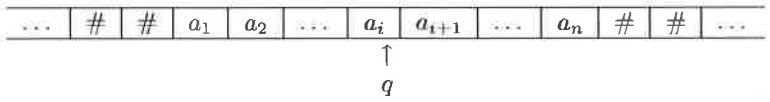
   (b) Dit kan je aantonen door de volgende drie uitspraken te bewijzen:

      i. Een TM is sterker dan een 3-PDA.

      ii. Een 3-PDA is sterker dan een 2-PDA.

      iii. Een 2-PDA is sterker dan een TM.

   Voor de eerste van deze moet je aantonen dat je de drie stacks kan simuleren op een TM. De tweede uitspraak is triviaal: negeer gewoon een van de drie stacks, dan heb je een 2-PDA. De derde uitspraak bewijzen is wat lastiger. De idee is een TM te simuleren op een 2-PDA. Dat doe je door een toestand




   van de TM voor te stellen op de 2-PDA door

   - Stack 1 bevat $a_{i-1}, a_{i-2} \ldots, a_1, \#$ (in die volgorde. $a_{i-1}$ is de top.)
   - Stack 2 bevat $a_i, a_{i+1}, \ldots, a_n, \#$.
   - We zitten in een toestand $q$.

   Er rest dan nog de invoer eerst helemaal in te lezen op stack 1 en daarna om te keren op stack 2. De toestand die je dan bekomt komt overeen met de starttoestand van de TM. Vervolgens moet de overgangsfunctie van de TM gecodeerd worden op de 2-PDA. Dat is een kwestie van symbolen van de ene stack op de andere over te hevelen.

3. De ambiguiteit van deze taal kan je inzien door twee verschillende parse-trees op te stellen voor de zin **if condition then if condition then a:=1 else a:=1**. Een oplossing kan bijvoorbeeld zijn elke else met de dichtsbijzijnde 'vrije' if te matchen. Dat wordt bereikt met de volgende grammatica.

$$\begin{aligned}
\langle \text{STMT} \rangle &\rightarrow \langle \text{MATCHED} \rangle \mid \langle \text{UNMATCHED} \rangle \\
\langle \text{MATCHED} \rangle &\rightarrow \textbf{if condition then } \langle \text{MATCHED} \rangle \textbf{ else } \langle \text{MATCHED} \rangle \\
\langle \text{MATCHED} \rangle &\rightarrow \langle \text{ASSIGN} \rangle \\
\langle \text{UNMATCHED} \rangle &\rightarrow \textbf{if condition then } \langle \text{MATCHED} \rangle \textbf{ else } \langle \text{UNMATCHED} \rangle \\
\langle \text{UNMATCHED} \rangle &\rightarrow \textbf{if condition then } \langle \text{STMT} \rangle \\
\langle \text{ASSIGN} \rangle &\rightarrow \text{a:=1}
\end{aligned}$$

# Oefenzitting 6

1. Een *'blijf staan Turing machine'* verschilt van een gewone Turing machine doordat de head naar rechts kan bewegen of kan blijven staan, maar niet naar links kan bewegen. Toon aan dat deze variant van Turing machines niet equivalent is met de originele variant. Welke talen worden herkend door 'blijf staan Turing machines'?

2. Zij $INF_{\text{DFA}} = \{\langle A \rangle \mid A \text{ is een DFA en } L_A \text{ is een oneindige taal}\}$. Toon aan dat $INF_{\text{DFA}}$ beslisbaar is.

3. Is $VIJF_{\text{DFA}} = \{\langle A \rangle \mid A \text{ is een DFA en } L_A \text{ bestaat uit precies 5 strings}\}$ beslisbaar? Bewijs je antwoord.

4. Toon aan dat de vraag of een context-vrije grammatica minstens één string uit $1^*$ kan genereren, beslisbaar is.

5. Zij $A$ en $B$ twee disjuncte talen die co-Turing-herkenbaar zijn. Toon aan dat er een beslisbare taal $C$ bestaat zodanig dat $A \subseteq C$ en $B \subseteq \overline{C}$.

6. Een *Turing machine met RESET* verschilt van een gewone Turing machine doordat de head enkel naar rechts kan bewegen of in één stap helemaal aan het begin van de tape gezet kan worden. De transitiefunctie van zo een machine is dus van de vorm $\delta : Q \times \Gamma \to Q \times \Gamma \times \{\text{R}, \text{RESET}\}$. Toon aan dat deze variant van Turing machines equivalent is met de originele variant.

25/11/2010

# Oefeningenzitting 6

## Oef 1

Herkenbaar: $\exists TM : L = L_{TM}$

Co-herkenbaar: $\exists TM$ met $\bar{L} = L_{TM}$

Beslisbaar: Herkenbaar + co-herkenbaar

$= \exists$ TM met $L = L_{TM}$ en TM stopt voor elke invoer

vb $A_{DFA}$ $E_{DFA}$ $EQ_{DFA}$ $All_{DFA}$ $A_{CFG}$ $E_{CFG}$

Niet-beslisbaar: vb $EQ_{CFG}$ $All_{CFG}$

$q,a \longrightarrow q',a',R$

$q_1,a_1 \longrightarrow q_2,a_2,S$

$q_2,a_2 \longrightarrow q_3,a_3,S$

$\vdots$

$q_{n-1},a_{n-1} \longrightarrow q_n,a_n,S$

a) $q_n,a_n \longrightarrow q',a',R$

b) $q_n,a_n \longrightarrow q',a',R$ en $q' \in \{q_{reject}, q_{accept}\}$

$q_n,a_n \longrightarrow q',a',S$

c) $q_n,a_n \longrightarrow q_i,a_i,S$ en $i \leq n$. ~ ONEINDIGE LUS

⇒ omzetten naar een TM die ENKEL naar rechts kan

$q,a \longrightarrow q',a',R$ $a'$ mag volledig willekeurig zijn.

$\vdots$

a) $q_1,a_1 \longrightarrow q',\cancel{a',R}$

b) $q_1,a_1 \longrightarrow q',\cancel{a',R}$

c) $q_1,a_1 \longrightarrow q_{reject},\cancel{a',R}$

⇒ Herkent precies dezelfde taal

⇒ Hier maken we een DFA van

⇒ zwakker dan een TM

① string w in A: $M_2$ aanvaard en als eerste
want $w \in A$ en $w \in B$
② string w in B: $M_1$ aanvaard en als eerste
...

⇒ BESLISBAAR!

# Oplossingen voor oefenzitting 6

1. De talen die door 'blijf staan TMs' herkend kunnen worden zijn allemaal regulier. Formeel kan je dit bewijzen door een gegeven blijf staan TM eerst te veranderen in een die enkel naar rechts kan. Die overloopt dus gewoon de invoer en beslist dan (of heeft al beslist) in een of meerdere stappen of hij zal accepteren of niet. Je kan zo een TM gemakkelijk omzetten in een NFA.

2. Het volgende algoritme beslist $INF_{\text{DFA}}$.

   (a) Zij $k$ het aantal toestanden van $A$.

   (b) Construeer een DFA die de doorsnede van $L_A$ met $\underbrace{\Sigma\Sigma\ldots\Sigma}_{k \text{ maal}}\Sigma^*$ herkent. Noem deze DFA $B$.

   (c) Beslis of $B$ de lege taal genereert. Zo ja, reject. Zo nee, accept.

3. $VIJF_{\text{DFA}}$ is beslisbaar. Een algoritme om dat na te gaan kijkt eerst na of $A$ een eindige taal genereert. Vervolgens telt het het aantal strings die door $A$ aanvaard worden en lengte $\leq k$ hebben, met $k$ het aantal toestanden van $A$.

4. Het algoritme construeert eerst de CFG die de doorsnede van de gegeven CFG met $1^*$ herkent. Vervolgens wordt nagekeken of die nieuwe CFG de lege taal genereert.

5. Zij $TM_{\overline{A}}$ en $TM_{\overline{B}}$ Turing Machines die respectievelijk $\overline{A}$ en $\overline{B}$ herkennen. Het algoritme dat $C$ beslist gaat als volgt:

   (a) Run $TM_{\overline{A}}$ en $TM_{\overline{B}}$ in parallel op de invoer.

   (b) Als $TM_{\overline{A}}$ eerst accepteert, reject. Als $TM_{\overline{B}}$ eerst accepteert, accept.

6. Een TM met reset simuleren op een gewone TM is geen probleem. Het omgekeerde, een TM $T_1$ simuleren op een TM $T_2$ met reset, kan bijvoorbeeld als volgt.
   We gaan er steeds voor zorgen dat als er een normaalgezien een symbool $a$ onmiddelijk links van de head staat in $T_1$, daar in $T_2$ het symbool $\dot{a}$ staat. Daar kunnen we initieel voor zorgen door de invoerstring 1 plaats op te schuiven en vooraan $\dot{\#}$ toe te voegen. Probeer in te zien dat je daarvoor genoeg hebt aan naar rechts te gaan en 1 keer te resetten. Daarna passen we het programma van $T_1$ aan.

   - Beweegt $T_1$ de head naar naar rechts en schrijft $a$, dan doet $T_2$
     (a) Reset
     (b) Beweeg rechts tot aan het symbool met de $\cdot$.
     (c) Veeg de $\cdot$ uit en ga naar rechts.
     (d) Schrijf $\dot{a}$ en ga naar rechts.
   - Beweegt $T_1$ de head naar links en schrijft $a$, dan doet $T_2$
     (a) Schrijf $a$ en reset.
     (b) Als je nu boven $\dot{\#}$ staat, schuif dan de huidige inhoud van de band een plaats naar rechts, schrijf $\dot{\#}$ op de meest linkse positie, $\#$ of de tweede positie en ga op de tweede positie staan. Stop. Anders, ga naar (c).
     (c) Schrijf een $\cdot$ boven het symbool en ga naar rechts.
     (d) Zie je nu een symbool met een $\cdot$, ga naar (f). Anders, ga naar (e).
     (e) Reset en ga rechts tot aan het eerste symbool met een $\cdot$. Veeg de $\cdot$ uit en zet er een boven het volgende symbool. Ga naar rechts. Ga naar (d).
     (f) Ga naar rechts tot aan het tweede symbool met een $\cdot$ en veeg de $\cdot$ uit. Stop.

# Oefenzitting 7

1. Een *nutteloze toestand van een TM* $M$ is een toestand waarin $M$ voor geen enkele invoer kan komen. Zij $U = \{\langle M, q\rangle \mid M \text{ is een TM en } q \text{ een nutteloze toestand van } M\}$. Toon aan dat $U$ onbeslisbaar is.

2. Toon aan dat $R_{\text{DFA}} = \{\langle A\rangle \mid A \text{ is een DFA die } \widehat{w} \text{ aanvaardt asa hij } w \text{ aanvaardt}\}$ beslisbaar is. Is $R_{\text{TM}} = \{\langle M\rangle \mid M \text{ is een TM die } w \text{ aanvaardt asa ze } \widehat{w} \text{ aanvaardt}\}$ ook beslisbaar? Bewijs je antwoord.

3. Toon aan dat de taal van alle Turing machines $M$ waarvoor er een invoer bestaat zodanig dat $M$ bij die invoer een niet-blanco symbool zal overschrijven met een blanco symbool, onbeslisbaar is.

4. Toon aan dat het 'overeenkomstprobleem van Post' beslisbaar is over het alfabet $\Sigma = \{a\}$. (Dat wil zeggen: toon aan dat het voor een verzameling domino's van de vorm $\left[\frac{a^m}{a^n}\right]$ met $n, m \in \mathbb{N}$, beslisbaar is of er een overeenkomst bestaat.)

5. Toon aan dat er een onbeslisbare deelverzameling van $1^*$ bestaat.

6. Toon aan dat de vraag of een context-vrije grammatica $G$ alle strings uit $1^*$ kan genereren beslisbaar is.

7. Toon aan dat een taal $L$ herkenbaar is als en slechts als er een beslisbare taal $L'$ bestaat zodanig dat $L = \{x \mid \text{Er bestaat een } y \text{ zodanig dat } \langle x, y\rangle \in L'\}$.

8. Bewijs dat er voor elke oneindige herkenbare taal $L$ een oneindige beslisbare taal $L' \subseteq L$ bestaat.

2/12/2010

# Oefening 7

## Oef 1

Stel dat B een beslisser is voor U, dan beslist het volgende algoritme $E_{TM}$:

Bij invoer $\langle M \rangle$:

Run B op $\langle M, q \text{ accept} \rangle$ en accepteer als B accepteert.

Run B op $\langle M, q \text{ accept} \rangle$ en verwerp als B verwerp.

Aangezien $E_{TM}$ niet beslisbaar is, volgt nu dat B niet kan bestaan.

$f: \langle M \rangle \mapsto \langle M, q \text{ accept} \rangle$ is een reductie van $E_{TM}$ naar U.

## Oef 2

$R_{DFA} = \{\langle A \rangle \mid A \text{ is een DFA en } \forall \text{ string } w \text{ geldt dat } w \in L_A \text{ a.s.a. } \hat{w} \in L_A\}$

↳ moet voor elke string gelden!

Beslisser B zodanig dan $L_B = \hat{L_A}$

① Gegeven een DFA A, kan je een DFA B maken zodanig dat $L_B = \hat{L_A}$ (oefz. 1)

② $EQ_{DFA}$ is beslisbaar (cursus p. 101, stelling 10.3)

③ $\langle A \rangle \in R_{DFA} \Leftrightarrow L_A = \hat{L_A}$ ($\Leftarrow$ Stel $L_A = \hat{L_A}$, dan geldt voor elke string $w$ dat als $w \in L_A$, dat $\hat{w} \in \hat{L_A}$ en dus $\hat{w} \in L_A$
Per definitie van $R_{DFA}$ zit $\langle A \rangle$ dan in $R_{DFA}$.

$\Rightarrow$ Stel dat $L_A \neq \hat{L_A}$. Dan bestaat er een string $w_1$ die in $L_A$ zit en niet in $\hat{L_A}$ of er bestaat een string $w_2$ in $\hat{L_A}$ die niet in $L_A$ zit.
In het eerste geval volgt dat $\hat{w_1} \notin L_A$, want anders hadden we $\hat{w_1} = w_1 \in \hat{L_A}$
In het tweede geval volgt dat $\hat{w_2} \in L_A$, maar $w_2 \notin L_A$.
In beide gevallen hebben we dus een string $w$ zodat $w \in L_A$ en $\hat{w} \notin L_A$. Dus geldt $\langle A \rangle \notin R_{DFA}$).

Een beslisser voor $R_{DFA}$ werkt als volgt: Bij invoer $\langle A \rangle$:

Construeer B zodat $L_B = \hat{L_A}$ (①) en accepteer a.s.a. $\langle A, B \rangle \in EQ_{DFA}$ (②)

# Oplossingen voor oefenzitting 7

1. Stel dat $U$ beslisbaar is met een TM $B$. Bekijk dan de volgende machine die als input $\langle M \rangle$ krijgt voor een TM $M$.

   - Bij input $\langle M \rangle$, run $B$ op $\langle M, q_a \rangle$, waarbij $q_a$ de accept-toestand van $M$ is.
   - Als $B$ aanvaardt, aanvaard. Zo niet, verwerp.

   Deze machine beslist $E_{\text{TM}}$. Contradictie.

2. $R_{\text{DFA}}$ is beslisbaar met het volgende algoritme. Bij input $\langle A \rangle$

   - Construeer de DFA $\widehat{A}$ die het omgekeerde van $L(A)$ herkent.
   - Run een beslisser voor $EQ_{\text{DFA}}$ op $\langle A, \widehat{A} \rangle$.

   $R_{\text{TM}}$ is niet beslisbaar. Dit kan je inzien met behulp van de stelling van Rice.

3. Stel dat je een TM $B$ hebt die dit probleem beslist. Dan kan je daarmee bijvoorbeeld $E_{\text{TM}}$ beslissen door het volgende algoritme. Bij invoer $\langle M \rangle$:

   - Bouw $M$ om tot een TM $M'$ die dezelfde taal herkent, maar nooit een blanco over een niet blanco kan schrijven. (Probeer in te zien dat dat mogelijk is.)
   - Verander $M'$ in $M''$ zodanig dat $M''$, als $M'$ normaalgezien zou accepteren, eerst nog even een blanco over een niet-blanco schrijft.
   - Run $B$ op $M''$.

4. Kijk na of er in de gegeven verzameling een dominosteen $\left[\frac{a^m}{a^n}\right]$ bestaat met $m \leq n$ en een met $m \geq n$. Zo ja, aanvaard. Zo nee, verwerp. Probeer in te zien dat dit algoritme correct is!

5. Het aantal deelverzamelingen van $1^*$ is niet aftelbaar. Dus heeft $1^*$ een heleboel onbeslisbare deelverzamelingen.

6. De volgende eigenschap helpt je om dit gemakkelijk te bewijzen: als alle strings $1, 1^2, \ldots, 1^{p!+p}$, met $p$ de pomplengte voor $L(G)$ in $L(G)$ zitten, dan bevat $L(G)$ heel $1^*$. Het bewijs van de eigenschap steunt op het pompend lemma voor context-vrije talen:
   Stel dat $\{1, 1^2, \ldots, 1^{p!+p}\} \subset L(G)$. Om aan te tonen dat $1^k$ voor elke $k > p! + p$ ook in $L(G)$ zit schrijven we $k$ als $k = p + m + q$, met $m = k - p \mod p!$ en $q = k - p - m$. Merk op dat $q$ deelbaar is door $p!$. We weten dat $p \leq p + m \leq p + p!$, dus behoort $1^{p+m}$ tot $L(G)$. Bijgevolg kan die string volgens het pumping lemma opgedeeld worden in $uvxyz$ zodanig dat $uv^ixy^iz$ voor elke $i$ ook tot $L(G)$ behoort en $1 \leq |vy| \leq p$. Als we $i = 1 + q/|vy|$ nemen (wat kan, want $|vy|$ is een deler van $p!$ en dus van $q$), dan is $|uv^ixy^iz| = k$, en mogen we besluiten dat $1^k \in L(G)$.

7. Stel dat $L$ herkenbaar is met herkenner $M$. Definieer $L' := \{\langle w, n \rangle \mid M \text{ accepteert } w \text{ in precies } n \text{ stappen}\}$. Dan is $L'$ beslisbaar: een beslisser voor $L'$ kan gewoon $M$ $n$ stappen laten lopen op $w$ en zien of $M$ dan accepteert. Het is ook duidelijk dat $L'$ de gevraagde eigenschap heeft.

   Voor de omgekeerde richting van het bewijs, veronderstel dat $L'$ een beslisbare taal is met strings van de vorm $\langle x, y \rangle$. Zij $M'$ een beslisser voor $L'$. Het volgende algoritme is dan een herkenner voor $L$:

**Input**: een string $x$
**for** $n \in \mathbb{N}$ **do**
  **for** *elke string* $y$ *met* $|y| < n$ **do**
    Laat $M'$ $n$ stappen lopen op invoer $\langle x, y \rangle$;
    Als $M'$ accepteert, accepteer.
  **end**
**end**

8. Zij $M$ een herkenner voor $L$ en definieer

$$L' := \{w \in L \mid \text{voor elke } w' \in L \text{ met } |w| < |w'| \text{ geldt dat } M \text{ strikt meer stappen nodig heeft om } w' \text{ te herkennen dan om } w \text{ te herkennen}\}$$

We moeten aantonen dat $L'$ beslisbaar en oneindig is. Een beslisser voor $L'$ werkt als volgt. Voor een invoer $w$ worden alle strings $w_1, w_2, w_3, \ldots$ uit $\Sigma^*$ die langer zijn dan $w$ overlopen. De beslisser laat eerst $M$ één stap lopen op zowel $w$ als $w_1$, vervolgens twee stappen op $w$, $w_1$ en $w_2$, vervolgens drie stappen op $w$, $w_1$, $w_2$ en $w_3$, etc. Aangezien $L$ oneindig is, zal minstens een van de $w_i$ tot $L$ behoren. Dus zal $M$ op een bepaald ogenblik een van de strings waarop hij loopt, aanvaarden. Er zijn dan een aantal verschillende mogelijkheden:

- $M$ verwerpt $w$. Dan mag de beslisser $w$ ook verwerpen.
- $M$ aanvaardt na $n$ stappen een van de $w_i$ en is na $n$ stappen niet op $w$ gestopt. Dan kan de beslisser stoppen en $w$ verwerpen. Er is dan immers een string in $L$ die langer is dan $w$, en sneller aanvaardt wordt door $M$.
- $M$ aanvaardt na $n$ stappen $w$, en elk van de $w_i$'s heeft minstens $n$ stappen nodig om aanvaardt te worden door $M$. Dan zou het kunnen dat $w$ tot $L'$ behoort. Om daar helemaal zeker van te zijn, laat de beslisser $M$ gedurende $n-1$ stappen lopen op alle strings die strikt langer zijn dan $w$ en maximaal $m$ symbolen bevatten, waarbij $m$ het maximum is van $n$ en $|w|+1$. Als $M$ gedurende dat proces een van de strings aanvaardt, dan mag de beslisser $w$ verwerpen. In het andere geval wordt $w$ aanvaardt. Inderdaad, stel dat er een string $w'$ is die strikt langer is dan $w$ en door $M$ aanvaardt wordt binnen de $n-1$ stappen. Die string is dan strikt langer dan $m$. Zij $w''$ de string die bestaat uit de eerste $m$ symbolen van $w'$. Dan wordt $w''$ binnen de $n-1$ stappen aanvaardt door $M$, want de enige symbolen die $M$ kan zien wanneer hij loopt op $w'$ zijn de symbolen van $w''$. Uit deze tegenspraak volgt dat $w$ tot $L'$ behoort.

Nu moeten we nog aantonen dat $L'$ oneindig is. Stel dat $L'$ eindig is en dat $w$ een langste string is in $L'$. Dan is er een $n$ zodanig dat $M$ $w$ accepteert in $n$ stappen. Neem een string $w_1 \in L$ die langer is dan $w$. Zo een string bestaat, want $L$ is oneindig. Zij $n_1$ het aantal stappen dat $M$ nodig heeft om $w_1$ te accepteren. Aangezien $w$ tot $L'$ behoort is $n_1 > n$. Omdat $w_1 \notin L'$, bestaat er een $w_2 \in L$, zodat $|w_1| < |w_2|$ en waarvoor $M$ $n_2$ stappen nodig heeft om te accepteren en $n_2 < n_1$. Op die manier kunnen we een rij strings $|w_1| < |w_2| < |w_3| < \ldots$ opbouwen met bijhorende $n_1 > n_2 > n_3 > \ldots$. Zodoende vinden we uiteindelijk een $w_i \in L$ met bijhorende $n_i < n$. Maar dat betekent dat $w \notin L'$. Uit deze contradictie vinden we dat $L'$ oneindig of leeg moet zijn. Een gelijkaardig argument laat zien dat $L'$ niet leeg kan zijn.

# Oefenzitting 8

1. Zijn de volgende uitspraken waar of niet? Bewijs je antwoord.
   (a) $EQ_{\text{CFG}}$ is co-herkenbaar.
   (b) $A_{\text{TM}}$ kan gereduceerd worden naar $E_{\text{TM}}$.
   (c) Als een taal $L_1$ Turing gereduceerd kan worden naar een reguliere taal $L_2$, dan is $L_1$ ook regulier.
   (d) Een taal is herkenbaar als en slechts als ze gereduceerd kan worden naar $A_{\text{TM}}$.
2. Schrijf de volgende primitief recursieve functies met behulp van compositie, primitieve recursie en de basisfuncties.
   (a) $exp : \mathbb{N} \times \mathbb{N} \to \mathbb{N} : (x, y) \mapsto x^y$
   (b) $pred : \mathbb{N} \to \mathbb{N} : x \mapsto x - 1$ als $x \neq 0$ en $0 \mapsto 0$.
   (c) $dif : \mathbb{N} \times \mathbb{N} \to \mathbb{N} : (x, y) \mapsto |x - y|$

(9/12/2010) Oefeningenzitting 8

$L_1$ kan gereduceerd worden naar $L_2$ als $\exists f: \Sigma^* \to \Sigma^*$ zodat
- $f$ is berekenbaar
- $f(L_1) \subseteq L_2$
- $f(\overline{L_1}) \subseteq \overline{L_2}$

($L_1 \leq_m L_2$)

$L_1$ is Turing-reduceerbaar naar $L_2$ als $\exists O^{L_2}$ die $L_1$ beslist. ($L_1 \leq_T L_2$)

Eigenschap: $L_1 \leq_m L_2$ en $L_2$ is beslisbaar, dan is $L_1$ beslisbaar.
$L_1 \leq_m L_2$ en $L_2$ is herkenbaar, dan is $L_1$ herkenbaar
$L_1 \leq_T L_2$ en $L_2$ is beslisbaar, dan is $L_1$ beslisbaar.
(niet voor herkenbaar)

PRIMITIEF RECURSIEVE FUNCTIES

- $nul: \mathbb{N} \to \mathbb{N}: x \mapsto 0$
- $succ: \mathbb{N} \to \mathbb{N}: x \mapsto x+1$
- $p_i^m: \mathbb{N}^m \to \mathbb{N}: (x_1, \dots, x_m) \mapsto x_i$ met $m \geq 1$ en $i \leq m$

- $een: \mathbb{N} \to \mathbb{N}: x \mapsto 1$ ($een = succ \circ nul = C_n[succ, nul]$)
- $f: \mathbb{N}^m \to \mathbb{N}$
  $g_1: \mathbb{N}^m \to \mathbb{N}$
  $\vdots$
  $g_m: \mathbb{N}^m \to \mathbb{N}$
  $C_n[f, g_1, \dots, g_m]: \mathbb{N}^m \to \mathbb{N}: (x_1, \dots, x_m) \mapsto f(g_1(x_1, \dots, x_m), \dots, g_m(x_1, \dots, x_m))$.

- $plus: \mathbb{N}^2 \to \mathbb{N}: (x,y) \mapsto x+y$

$$plus(x,y) = \begin{cases} x & \text{als } y = 0 \\ plus(x,y-1)+1 & \text{als } y > 0 \to \text{recursief.} \end{cases}$$

$\hookrightarrow succ(plus(x,y-1))$

BASIS: $p_1^1$ en SOM $= P_n[P_1^1, C_n[succ, P_3^3]]$

RECURSIEVE FUNCTIE

$r(\overset{a}{x}, \overset{b}{y-1}, \overset{c}{plus(x,y-1)})$
$= succ(plus(x,y-1))$
$= succ(c)$
$= succ(p_3^3(a,b,c))$
$\rightsquigarrow r = C_n[succ, P_3^3]$

$b: \mathbb{N}^m \to \mathbb{N}$
$r: \mathbb{N}^{m+2} \to \mathbb{N}$ altijd 2 of groter
$P_n[b,r]: \mathbb{N}^{m+1} \to \mathbb{N}: (x_1, \dots, x_m, y)$

$$\mapsto \begin{cases} b(x_1, \dots, x_m) & \text{als } y = 0 \\ r(x_1, \dots, x_m, y-1, P_n[b,r](x_1, \dots, x_m, y-1)) & \text{als } y > 0 \end{cases}$$

Oef 2

a) $exp(x,y) = \begin{cases} 1 & \text{als } y = 0 \\ x^{y-1} \cdot x & \text{als } y > 0 \end{cases}$

BASIS: $b(x) = 1$ dus $b = zer = C_n[succ, nul]$

RECURSIE: $r(\underset{a}{x}, \underset{b}{y-1}, \underset{b}{exp(x, y-1)}) = x^{y-1} \cdot x$

$= prod(exp(x, y-1), x) = prod(c, a)$

$= prod(P_3^3(a,b,c), P_1^3(a,b,c))$

dus: $r = C_n[prod, P_3^3, P_1^3]$

$exp = P_n[b, r]$

b) $pred = C_n[P_n[mul, P_2^3], mul, P_1^1]$

c) $min = P_n[P_1^1, C_n[pred, P_3^3]]$

$dif = C_n[som, C_n[min, P_1^2, P_2^2], C_n[min, P_2^2, P_1^2]]$

# Oplossingen voor oefenzitting 8

1. (a) Waar. Stel dat je twee CFG's $G_1$ en $G_2$ hebt die een verschillende taal herkennen. Omdat $A_{\text{CFG}}$ beslisbaar is kan je gewoon voor alle strings (bijvoorbeeld van klein naar groot) uitproberen of ze tot $L(G_1)$ en $L(G_2)$ behoren. Na een tijdje kom je dan gegarandeerd een string tegen die slechts in één van de talen $L(G_1)$ en $L(G_2)$ zit.

   (b) Niet waar. Immers, stel dat $A_{\text{TM}} \leq_m E_{\text{TM}}$. Dan volgt uit stelling 16.7 dat $\overline{A_{\text{TM}}} \leq_m \overline{E_{\text{TM}}}$. Aangezien $\overline{E_{\text{TM}}}$ herkenbaar is, moet volgens stelling 16.4 $\overline{A_{\text{TM}}}$ ook herkenbaar zijn. Maar $\overline{A_{\text{TM}}}$ kan niet herkenbaar zijn, aangezien $A_{\text{TM}}$ niet beslisbaar is, maar wel herkenbaar.

   (c) Niet waar. Elke beslisbare taal kan gereduceerd worden tot een reguliere taal.

   (d) Waar. De implicatie van rechts naar links volgt onmiddellijk uit stelling 16.4. De implicatie van links naar rechts kan je als volgt bewijzen. Stel dat $L$ een herkenbare taal is en zij $M$ een TM die $L$ herkent. Dan definieert de functie $f : w \mapsto \langle M, w \rangle$ een reductie van $L$ naar $A_{\text{TM}}$.

2. (a) $Pr[Een, Cn[Product, p_1^3, p_3^3]$ met $Een$ de constante functie 1 en $Product$ de functie die het product van twee getallen berekent.

   (b) $Cn[Pr[nul, p_2^3], nul, p_1^1]$

   (c) Defineer eerst de functie $min : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ met $min(x, y) = x - y$ als $x > y$ en $min(x, y) = 0$ als $x \leq y$. Dat kan bijvoorbeeld met $min = Pr[p_1^1, Cn[pred, p_3^3]]$. Vervolgens kan je $dif$ definiëren door $dif = Cn[Som, Cn[min, p_1^2, p_2^2], Cn[min, p_2^2, p_1^2]]$.

# Extra oefeningen over $\Lambda$-calculus

(Hieronder wordt de verzameling {FALSE, TRUE} aangeduid met *Bool*)

1. Definieer in zuivere $\lambda$-calculus:

   (a) $\text{IsZero} : \mathbb{N} \to \mathit{Bool} : n \mapsto \begin{cases} \text{TRUE} & \text{als } n = 0 \\ \text{FALSE} & \text{anders} \end{cases}$

   (b) $\text{L} : \mathbb{N} \to \mathit{List}(\mathbb{N}) : n \mapsto [n, n-1, n-2, \ldots, 0]$

   (c) $\text{Pred} : \mathbb{N} \to \mathbb{N} : n \mapsto \begin{cases} n-1 & \text{als } n > 0 \\ 0 & \text{als } n = 0 \end{cases}$

   (d) $\text{Min} : \mathbb{N}^2 \to \mathbb{N} : (x, y) \mapsto \begin{cases} x - y & \text{als } x \geq y \\ 0 & \text{anders} \end{cases}$

   (e) $\text{Eq} : \mathbb{N} \to \mathit{Bool} : (x, y) \mapsto \begin{cases} \text{TRUE} & \text{als } x = y \\ \text{FALSE} & \text{als } x \neq y \end{cases}$

2. Definieer in $\lambda$-calculus een functie SomCijfers die een getal afbeeldt op de som van zijn cijfers. Je mag gebruik maken van '=', '<', '+', '−', '·' en IF.

(16/12/2010)

# Oefeningenzitting 9

<term> = ~~<constante>~~
| (<term>)
| <term> <term>
| <variabele>
| λ<variabele>. <term>

ZUIVERE λ-CALCULUS

vb 0: λSx. x
1: λSx. S(x)
2: λSx. S(S(x))

α: λx.t ⟶ λy. t[x/y]
β: (λx.t) ⇝ t[x/q]
η: λx. Fx ⟶ F als x niet voorkomt in F

HB p.160

Oef 2

a) (λx. x 17)(+3)
β→ (+3 17) ⟶ 20

b) (λx. x 17)(λy. 12)
β→ (λy. 12 17)
η→ 12

Oef 4

+2(λx. (λy. λx. + (+xy) x) x) (λy. +xy)

Oef 5

a) $A_{+C_0C_n} \to [(\lambda xypq.\ x\,p\,(ypq))\ (\lambda fx.x)]\ (\lambda fx.\ f^n(x))$
β→ $(\lambda ypq.\ (\lambda fx.x)\ p\ (ypq))\ (\lambda fx.\ f^n(x))$
β→ $\lambda pq.\ (\lambda fx.x)\ p\ ((\lambda fx.\ f^n(x))\ p\ q)$
β→ $\lambda pq.\ (\lambda fx.x)\ p\ (\lambda x.\ p^n(x))\ q)$
η→ $\lambda pq.\ (\lambda fx.x)\ p\ (p^n(q))$
β→ $\lambda pq.\ (\lambda x.x)\ (p^n(q))$ β→ $\lambda pq.\ p^n(q)$
α→ $\lambda fx.\ f^n(x) \to C_n$

# Oplossingen voor oefenzitting 9

(Hieronder wordt de verzameling $\{\text{FALSE}, \text{TRUE}\}$ aangeduid met $Bool$)

1. Definieer in zuivere $\lambda$-calculus:

   (a) $\text{IsZero} : \mathbb{N} \to Bool : n \mapsto \begin{cases} \text{TRUE} & \text{als } n = 0 \\ \text{FALSE} & \text{anders} \end{cases}$

   Oplossing:

   $$\text{IsZero} := \lambda npq.n(\lambda x.q)p$$

   De intuitie voor deze oplossing is als volgt. Als je naar de definitie van de Church-numeral $c_n$ kijkt zie je dat die twee argumenten neemt en het eerste argument $n$ maal toepast op het tweede. In de oplossing is $(\lambda x.q)$ – de constante functie $q$ – het eerste argument dat de Church-numeral krijgt. Als $n > 0$, dan reduceert $c_n(\lambda x.q)p$ dus tot $q$. Bijgevolg reduceert IsZero $c_n$ tot $\lambda pq.q :=$ FALSE als $n > 0$. Anderzijds betekent $c_0(\lambda x.q)p$: pas 0 keer de functie $(\lambda x.q)$ toe op $p$. Dus reduceert $c_0(\lambda x.q)p$ tot $p$ en IsZero $c_0$ tot $\lambda pq.p :=$ TRUE.

   (b) $\text{L} : \mathbb{N} \to List(\mathbb{N}) : n \mapsto [n, n-1, n-2, \ldots, 0]$

   Oplossing:

   $$\text{L} := \lambda n.\ n\ F\ (\text{CONS}\ c_0\ \text{NIL})$$

   Hierbij is $c_0$ de Church-numeral 0 en $F$ is gedefinieerd door

   $$F := \lambda x.\ \text{CONS}\ (A_+\ c_1\ (\text{HEAD}\ x))\ x$$

   en NIL is de lege lijst, bijvoorbeeld

   $$\text{NIL} := \lambda f.\text{TRUE}$$

   (Het is niet noodzakelijk voor de oefening om NIL op deze manier te definiëren. Maar met deze definitie is het gemakkelijk om een 'IsNil' functie te schrijven.)

   (c) $\text{Pred} : \mathbb{N} \to \mathbb{N} : n \mapsto \begin{cases} n-1 & \text{als } n > 0 \\ 0 & \text{als } n = 0 \end{cases}$

   Oplossing:

   $$\text{Pred} := \lambda n.\ \text{IF}\ (\text{IsZero}\ n)\ c_0\ (\text{HEAD}\ (\text{TAIL}\ (\text{L}\ n)))$$

   (d) $\text{Min} : \mathbb{N}^2 \to \mathbb{N} : (x, y) \mapsto \begin{cases} x - y & \text{als } x \geq y \\ 0 & \text{anders} \end{cases}$

   Oplossing:

   $$\text{Min} := \lambda\ x\ y\ .\ y\ \text{Pred}\ x$$

   (e) $\text{Eq} : \mathbb{N}^2 \to Bool : (x, y) \mapsto \begin{cases} \text{TRUE} & \text{als } x = y \\ \text{FALSE} & \text{als } x \neq y \end{cases}$

   Oplossing:

   $$\text{Eq} := \lambda x\ y\ .\ \text{AND}\ (\text{IsZero}\ (\text{Min}\ x\ y))\ (\text{IsZero}\ (\text{Min}\ y\ x))$$

2. Definieer in $\lambda$-calculus een functie SomCijfers die een getal afbeeldt op de som van zijn cijfers.

   Oplossing: Schrijf eerst

   $$\text{SomCijfers} := \lambda n.\ \text{IF}\ (=\ n\ 0)\ 0\ (+(\text{mod}\ n\ 10)\ (\text{SomCijfers}(\text{div}\ n\ 10)))$$

   waarbij div de gehele deling voorstelt. Om SomCijfers op een niet-recursieve manier te definiëren kan je de vastepunts-combinator $Y$ gebruiken:

   $$\text{SomCijfers} := Y(\lambda f.\ \lambda n.\ \text{IF}\ (=\ n\ 0)\ 0\ (+(\text{mod}\ n\ 10)\ (f(\text{div}\ n\ 10))))$$

   Het uitschrijven van definities voor mod en div laat ik als oefening ...

# Oefenzitting 10

1. Waar of niet waar?

   (a) Elke eindige taal is regulier.

   (b) Elke deelverzameling van een niet-reguliere taal is niet-regulier.

   (c) De klasse van beslisbare talen is gesloten onder het nemen van het complement.

2. Zij $L$ een reguliere taal over een alfabet $\Sigma$ met minstens 2 tekens. De *afstand* $d(s,t)$ tussen twee strings $s$ en $t$ van gelijke lengte is het aantal posities waarop $s$ en $t$ verschillen. Met andere woorden, als $s = s_1 s_2 \ldots s_n$ en $t = t_1 t_2 \ldots t_n$, dan is $d(s,t) = \#\{i \mid s_i \neq t_i\}$. Definieer nu de taal

   $$fout_1(L) = \{s \in \Sigma^* \mid \text{er bestaat een } t \in L \text{ zodat } |s| == |t| \text{ en } d(s,t) \leq 1\}$$

   Informeel is dat de taal met strings met hoogstens één foutje ten opzichte van een string in L. Bewijs dat $fout_1(L)$ regulier is.

3. Geef voor elk van de volgende talen aan of ze regulier, context-vrij, beslisbaar, Turing-herkenbaar en/of co-Turing-herkenbaar zijn.

   (a) $\{x = y + z \mid x, y \text{ en } z \text{ zijn natuurlijke getallen in binaire notatie en } x \text{ is de som van } y \text{ en } z.\}$ over het alfabet $\{0, 1, +, =\}$.

   (b) De taal van alle strings $w$ over het alfabet $\left\{\left[\begin{smallmatrix}0\\0\end{smallmatrix}\right], \left[\begin{smallmatrix}0\\1\end{smallmatrix}\right], \left[\begin{smallmatrix}1\\0\end{smallmatrix}\right], \left[\begin{smallmatrix}1\\1\end{smallmatrix}\right]\right\}$ zodanig dat het binair getal gevormd door de onderste rij van $w$ gelijk is aan drie maal het binair getal gevormd door de bovenste rij van $w$.

   (c) $\{\langle M \rangle \mid M \text{ is een TM zodanig dat } L(M) \text{ precies 1234 strings bevat}\}$

   (d) $\{a^i b^j c^k \mid i, j, k \geq 0 \text{ en } i \neq j \text{ of } j \neq k\}$

(23/12/2010) # Oefeningenzitting 10

## Oef 2

Vb $\Sigma_1 = \{a, b, c, \ldots x\}$ $\Sigma_2 = \{0, 1, 2, \ldots, 9\}$

$L_1$ is een taal over $\Sigma_1$ en $L_2$ een taal over $\Sigma_2$.

Als $a_1 a_2 \ldots a_n \in \Sigma_1^*$ en $b_1 b_2 \ldots b_m \in \Sigma_2^*$ dan definiëren we:

$$exp(a_1 a_2 \ldots a_m, b_1 b_2 \ldots b_m) = \underbrace{a_1 a_1 \ldots a_1}_{b_1} \underbrace{a_2 \ldots a_2}_{b_2} \ldots \underbrace{a_n \ldots a_n}_{b_m}$$

Vb $exp(abc, 213) = aabccc$

Notee $exp(L_1, L_2) = \{exp(\omega_1, \omega_2) \mid \omega_1 \in L_1, \omega_2 \in L_2 \text{ en } |\omega_1| = |\omega_2|\}$

Als $L_1$ en $L_2$ regulier zijn, is $exp(L_1, L_2)$ het ook ?

## Oef 1

a) WAAR : disjunctie voor alle strings van de taal

b) NIET WAAR : reguliere talen $\subseteq$ niet-reguliere talen

c) WAAR: beslisbare taal $\Rightarrow \exists$ TM

$\Rightarrow$ accept & reject omwisselen

$\Rightarrow \exists$ TM die het complement aanvaardt

## Oef 2

Zij $(Q, \Sigma, \delta, q_s, F)$ een DFA die $L$ herkent. → makkelijker dan RegExp

Construeer de NFA $(Q \times \{0,1\}, \Sigma, \delta', (q_s, 0), F \times \{0,1\})$

Deze automaat herkent $fout_1(L)$. Bijgevolg is $fout_1(L)$ regulier.

(*)

Oplossing vb (# fouten)

→ 2 machines: $(Q_1, \Sigma_1, \delta_1, q_{s1}, F_1)$

$(Q_2, \Sigma_2, \delta_2, q_{s2}, F_2)$

$\Rightarrow (Q_1 \times Q_2, \Sigma_1, \delta' \quad (q_{s1}, q_{s2}), F_1 \times F_2)$

geen NFA/geen DFA!

# Oplossingen voor oefenzitting 10

1. (a) Waar. Stel dat $L = \{s_1, \ldots, s_n\}$, dan wordt $L$ gegenereerd door de reguliere expresie $s_1|s_2|\ldots|s_n$.

   (b) Niet waar. De lege taal is regulier en een deelverzameling van elke taal.

   (c) Waar. Stel dat $L$ beslisbaar is. Dan is $L$ herkenbaar en co-herkenbaar. Dus is $\overline{L}$ co-herkenbaar en herkenbaar. Bijgevolg is $\overline{L}$ beslisbaar.

2. Stel dat $(Q, \Sigma, \delta, q_s, F)$ een DFA is die $L$ herkent. We gaan deze omvormen tot een NFA die $fout_1(L)$ herkent. De toestanden van die NFA zijn gegeven door $Q \times \{0, 1\}$. We hebben dus twee kopies van de toestanden van de oorspronkelijke DFA. We gaan er nu voor zorgen dat wanneer we nog geen fouten gemaakt hebben voor een gegeven invoer, we in een toestand $(q, 0)$ zitten. Wanneer we een fout maken gaan we over naar een toestand $(q, 1)$. Vanaf we in zo een toestand zitten mogen we geen fouten meer maken. Formeel wordt de NFA gegeven door $(Q \times \{0,1\}, \Sigma, \delta', (q_s \times 0), F \times \{0,1\})$, waarbij $\delta'$ gegeven is door

$$\delta'(q \times 1, a) = \{\delta(q, a) \times 1\}$$
$$\delta'(q \times 0, a) = \{\delta(q, a) \times 0\} \cup \{\delta(q, b) \times 1 \mid b \neq a\}$$

3. De volgende tabel geeft een overzicht van de oplossing

| | Regulier | Context-Vrij | Beslisbaar | Herkenbaar | Co-herkenbaar |
|---|---|---|---|---|---|
| (a) | | | ✓ | ✓ | ✓ |
| (b) | ✓ | ✓ | ✓ | ✓ | ✓ |
| (c) | | | | | |
| (d) | | ✓ | ✓ | ✓ | ✓ |

   (a) Deze taal is duidelijk beslisbaar. Je kan immers gemakkelijk een computerprogramma schrijven dat deze taal beslist. Dat ze niet context-vrij is kan aangetoond worden met het pompend lemma voor context-vrije talen. Een niet te pompen string is bijvoorbeeld $1^{2p} = 1^p + 1^p 0^p$, met $p$ de pomplengte.

   (b) Het omgekeerde van deze taal wordt herkent door de DFA $(\{q_0, q_1, q_2\}, \Sigma, \delta, q_0, \{q_0\})$ waarbij de (niet-totale) functie $\delta$ gegeven is door

| | $\left[\begin{smallmatrix}0\\0\end{smallmatrix}\right]$ | $\left[\begin{smallmatrix}0\\1\end{smallmatrix}\right]$ | $\left[\begin{smallmatrix}1\\0\end{smallmatrix}\right]$ | $\left[\begin{smallmatrix}1\\1\end{smallmatrix}\right]$ |
|---|---|---|---|---|
| $q_0$ | $q_0$ | | | $q_1$ |
| $q_1$ | | $q_0$ | $q_2$ | |
| $q_2$ | $q_1$ | | | $q_2$ |

   Bijgevolg is de taal zelf ook regulier.

   (c) Deze taal is niet beslisbaar vanwege de stelling van Rice. We noteren de taal met $L$.
   Stel dat $L$ herkenbaar is met een TM $T$. Dan herkent het volgende algoritme $E_{\text{TM}}$. Bij invoer $M$, construeer een TM $M'$ die precies 1234 strings meer herkent dan $M$. Run vervolgens $T$ op $M'$.
   Om aan te tonen dat $L$ niet co-herkenbaar is werken we met een reductie vanuit de niet-herkenbare taal $\overline{A_{\text{TM}}}$. De functie $f$ die we daarvoor moeten construeren geeft bij invoer $\langle M, w\rangle$ een turingmachine $M'$ die bij invoer $w'$ eerst $M$ laat lopen op $w$ en wanneer $M$ accepteert, $M_{1234}$ laat lopen op $w'$. Hierbij is $M_{1234}$ een TM die precies 1234 strings aanvaardt. Als $\langle M, w\rangle \in \overline{A_{\text{TM}}}$, dan herkent $M'$ geen enkele string, en geldt er dus $\langle M'\rangle \in \overline{L}$. Als $\langle M, w\rangle \notin \overline{A_{\text{TM}}}$, dan herkent $M'$ precies 1234 strings en geldt er $\langle M'\rangle \notin \overline{L}$. De functie $f$ definieert dus inderdaad een reductie van $\overline{A_{\text{TM}}}$ naar $\overline{L}$.

   (d) Stel dat deze taal $L$ regulier is. Dan is $L^c \cap a^*b^*c^* = \{a^n b^n c^n \mid n \in \mathbb{N}\}$ ook regulier, wat duidelijk niet het geval is[1]. Ze is context-vrij, want wordt herkend door de grammatica
   $S \to AM \mid NC$
   $A \to \epsilon \mid Aa$
   $B \to \epsilon \mid Bb$

[1] Je zou ook het pompend lemma kunnen gebruiken met bijvoorbeeld de string $a^p b^{p+p!} c^{p+p!}$ als string die niet te pompen is.

$C \to \epsilon \mid Cc$
$M \to bMc \mid bB \mid cC$
$N \to aNb \mid aA \mid bB$