

Opdracht 1

Topics on Parsing and Formal Languages - fall 2010

Rick van der Zwet
<hvdzwet@liacs.nl>

8 december 2010

Samenvatting

Dit schrijven zal uitwerkingen van opgaven behandelen uit het boek [JS2009] gebruikt bij het college. In deze opdracht zullen zeven opgaven (3,20,22,47,54,68,69) van hoofdstuk 3 behandeld worden.

1 Opgave 3.3

Als $L \subseteq \Sigma^*$ is regulier dan is de taal

$$2L := a_1, a_1, a_2, a_2, \dots, a_k, a_k : \text{elke } a_i \in \Sigma \text{ en } a_1 a_2 \dots a_k \in L \quad (1)$$

regulier. Zie dat er een ‘verdubbeling’ optreed van symbolen, dit gedrag is de modelleren in een *DFA*. Omdat L regulier is bestaat er een *DFA* $M = (Q, \Sigma, \delta, q_0, F)$ die L beschrijft. Construeer nu een nieuwe *DFA* P die de nieuwe taal $2L$ gaat beschrijven, neem hiervoor het alfabet (Σ) , en de begin-toestand q_0 over van L . De toestanden (Q) worden verdubbeld. voor alle $q \in Q$: voeg q' toe aan Q . Neem ook de transities over van M , maar maak aanpassingen zodanig dat de nieuwe toestanden ook gelezen worden. Dus $\delta(q, x)$ wordt $\delta(q, q'), \delta(q', x)$. De acceptatie toestand F is gelijk aan de oude acceptatie toestand.

De nieuwe *DFA* P beschrijft $2L$ en dus is $2L$ regulier. \square

2 Opgave 3.20

Laat $\Sigma = \{0, 1\}$ zijn. Een voorbeeld van de taal $L \subseteq \Sigma^*$ voor welke geldt dat, de Myhill-Nerode[JS2009][pg. 77–81] gelijkheid relatie R_L de eigenschap heeft dat elk woord in Σ^* zijn eigen equivalentie klasse is. Dit wilt zeggen dat xR_Ly dan en slechts dan voor alle $z \in \Sigma^*$, is er een $xz \in L$ dan en slechts als $yz \in L$. Omdat hier gezocht wordt naar een taal waarbij de woorden allemaal equivalentie klassen op-zich zijn, zal alle elementen met elkaar vergeleken kunnen worden door middel van R_L en hier mogen geen positieve 'gevallen' uitkomen.

De (triviale) taal $L = 01, 10$ is een voorbeeld hiervan. Beiden woorden hebben hun eigen klasse. Als $x = 0$ en $y = 1$, bestaat er een $z = 1$, zodanig dat $xz \in L, yz \notin L$.

De taal $M = \{0^* : |M|_0 \text{ is een priemgetal}\}$ is een ander voorbeeld. Als we kijken naar aR_Lb kan er altijd een z gevormd worden zodanig dat $az \in M, bz \notin M$. Bijvoorbeeld $a = 0^3, b = 0^5, z = 0^4$. Deze eigenschap wordt afgedwongen door het 'feit' dat er geen relatie bestaat tussen de lengtes tussen opeenvolgende priemgetallen.¹

3 Opgave 3.22*

Om aan te tonen dat een $2DFA$ exponentieel meer expressief is dan een DFA voor bepaalde talen kijken we naar de volgende taal; Laat n een integer ≥ 1 en $F_n \subseteq \{0, 1, 2, 3, 4\}^*$ als volgende gedefinieerd:

$$F_n = \{3 0^{i_1} 1 0^{i_2} \dots 1 0^{i_n} 2^k 0^{i_k} 4 : 1 \leq k, j, i_j \leq n\} \quad (2)$$

Als eerste kan de F_n door een $2DFA$ van $O(n)$ toestanden geaccepteerd worden, welke aan te tonen is door eerst te kijken naar de eigenschappen van F_n hierbij is de zien dat de 0-reeks van de lengte i_k zowel voor als na de 2^k moet voorkomen. Waarbij 2^k aangeeft, waar deze 0-reeks gevonden kan worden bij de plek k . De $2DFA$ moet dus twee dingen doen. Het aantal keer nul op twee

¹Ik heb dit voor gemak aangenomen, de praktijk is echter dat is nog een groot wiskundig vraagstuk is.

plekken vergelijken. Waarbij 1 plek vast staat en de tweede plek variabele wordt gedefinieerd. De aanpak kan als volgt gezien worden:

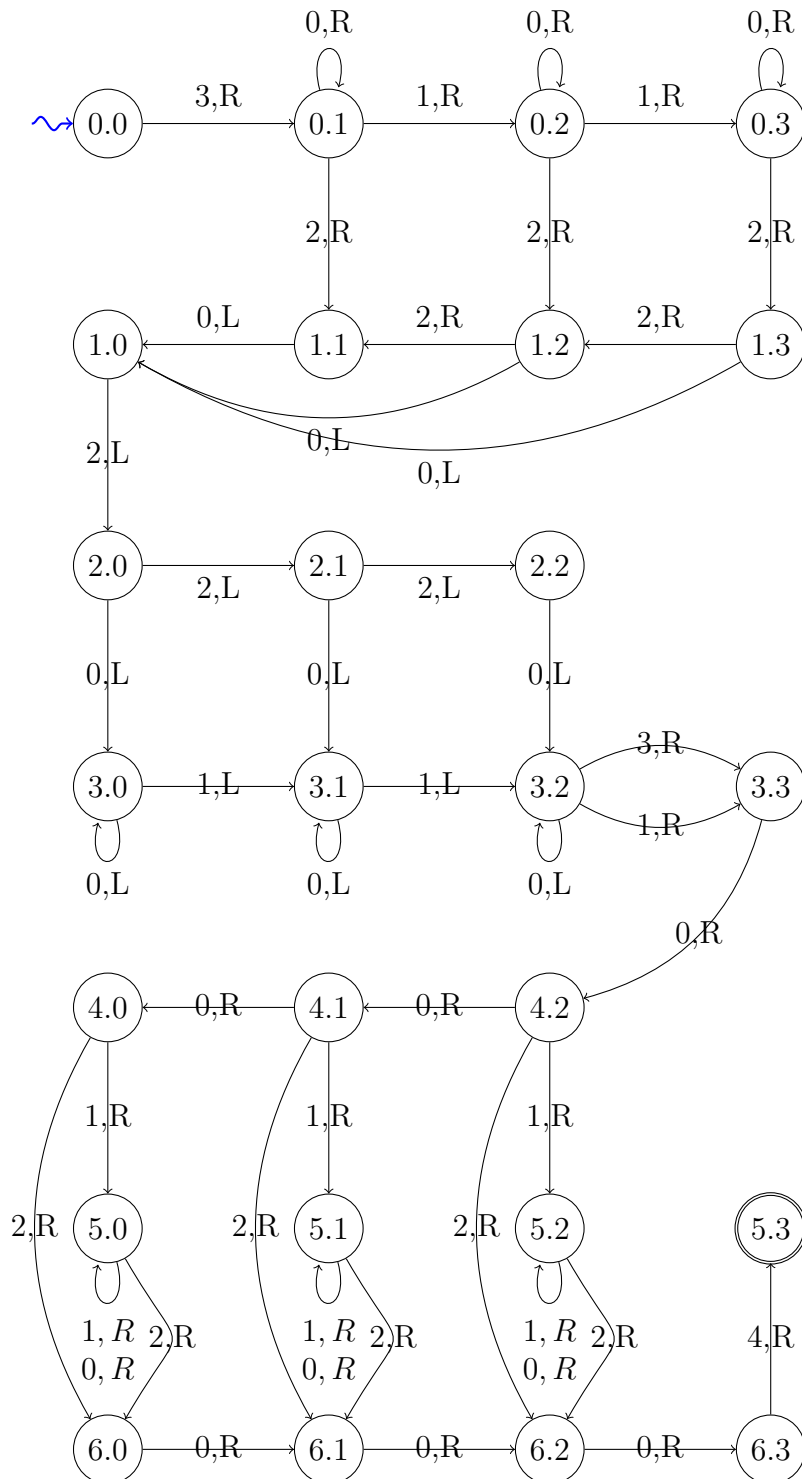
1. controleren of $|1| \leq 2$ de 0-reeks dmv 'knopen-strook' (knopen 0.X en 1.X).
2. $|2|$ tellen onthouden dmv 'knopen-strook' (knopen 2.X).
3. naar gewenste locatie lopen (knopen 3.X).
4. $|0|$ tellen onthouden dmv 'knopen-strook' (knopen 4.X).
5. naar begin 2de 0-reeks lopen (knopen 5.X).
6. 0-reeks vergelijken (knopen 6.X).

Voor deze aanpak zijn 3 'knopen-stroken' nodig, 3 'transitie-lagen' en 1 controle strook. In totaal dus in ongeveer $O(7n)$ knopen. Figuur 1 laat de *2DFA* zien.

Om te laten zien dat een *DFA* minstens n^n toestanden nodig heeft moet je gebruik maken van het feit dat een *DFA* niet terug kan lopen. Het zal dus een 'geheugen' moeten maken om het maximale 0 woord voor de 'splitsing' (2^k) te kunnen onthouden om zo te kijken of het overeen komt het 0 woord na de 'splitsing'. Omdat je voor elke F_i waarbij $1 \leq i \leq n$ minimaal n toestanden nodig hebt om te kunnen tellen en hiervan weer n unieke sets bestaan, heb je dus minimaal n^n toestanden nodig.

Dit vertaald zich in een Myhill-Nerode relatie om het aantal toestanden te kunnen bepalen. F_p en F_q waarbij $p < q, 1 \leq p, q \leq n$ is onderscheidbaar, door voor $z = 2*0*4$ te nemen, zodanig dat het woord door F_q geaccepteerd wordt, maar niet door F_p , door $|z|_2 \geq p$ te kiezen.

Voor elke $F_i : 1 \leq i \leq n$ geldt dat deze 'intern' ook i^2 unieke toestanden geeft. We splitsen de woorden bij 2, zodat we over een 'prefix' en 'suffix' kunnen praten. Een 'prefix' past maar bij 'e' en 'suffix'. De 'suffixes' die gegenereerd worden van de vorm $2*0*4$, kan i^2 vormen aannemen (bijvoorbeeld $i = 2204, 2004, 22004, 2204$). Deze unieke suffixes zorgen ervoor dat de Myhill-Nerode toepassing een substring z kan vinden welke een van de unieke suffixes bevat, waardoor het 'prefix' niet samenkomt met een 'suffix'.



Figuur 1: Opdracht 2.22a uitgewerkt voor $1 \leq n \leq 3$. Niet geldige stappen zijn niet genoemd en moet voor compleetheid naar een 'putje' knoop gestuurd worden. Uitbreiden kan door het expanden van 'triviale' knopen.

4 Opgave 3.47

Om te bewijzen dat ‘de klasse van talen geaccepteerd door *DFA*’ \Rightarrow ‘de klasse van talen gespecificeerd door de reguliere expressies’[JS2009][Theorem 1.4.2, (b) \Rightarrow (a)] aan de hand van het volgende voorbeeld: Laat $M = (Q, \Sigma, \delta, q_1, F)$ een *DFA* zijn, waarbij $Q = \{q_1, q_2, \dots, q_n\}$. Definieer nu $R_{i,j,k}$ als de taal van alle woorden die van toestand i naar toestand j gaan zonder door toestanden te gaan die hoger dan k genummerd zijn.

$R_{i,j,k}$ accepteert als $q_i \in q_1, q_j \in F$. Om $R_{i,j,k+1}$ te verkrijgen, nemen we eerst de toestanden van $R_{i,j,k}$, deze verenigen we dan met alle toestanden die we vanuit $R_{i,j,k}$ kunnen bereiken met een transitie $\delta(a, k+1) : i \leq a \leq j$, die in een nieuwe toestand uitkomen, de lussen hebben geen meerwaarde voor ons doel. De begin toestand is te markeren als $R_{i,j,0} = q_1 : i = j = 1$.

5 Opgave 3.54

Laat $\Sigma = \{1, 2, 3, \dots, n\}$ zijn en definieer:

$$L_n = \{w \in \Sigma^* : |w|_i = 1 \text{ voor alle } i \in \Sigma\} \quad (3)$$

Dit zijn dus de woorden waarbij alle elementen van Σ precies één keer voor komen. Bijvoorbeeld $L_3 = \{123, 132, 213, 231, 312, 321\}$.

a) Om te laten zijn dat er minimaal een reguliere expressie van lengte 2^{n-1} nodig is om L_n te specificeren, passen we de *Myhill-Nerode* stelling toe. Neem $x, y \in \Sigma^*$. Als geldt $|x| = |y|$ en de elementen zijn gelijk, maar enkel hun volgorde verschilt (bijvoorbeeld 123, 132) dan zitten ze in dezelfde klasse. Voor $xRiy$ als de lengte verschilt, dan is er een z te vinden zodanig dat y wel geaccepteerd wordt en x niet (bijvoorbeeld voor 34, 345, is dit 12), vanwege $|x| + a \neq |y| + a = n$ desals $|x| \neq |y|$. Als de ‘inhoud’ verschilt maar de lengte gelijk is dan $z = \Sigma - \{y\}$, omdat $\Sigma \setminus \{x\} \neq \Sigma \setminus \{y\}$.

b) Echter voor $\overline{L_n}$ is wel een reguliere expressie te vinden en wel in de grootte $O(n^2)$. $\bigcup_{a=0..n} \bigcup_{i \in \Sigma_a} \{x^*ix^*ix^*\}$ waarbij $x \in (\Sigma_a \setminus \{i\})^*$.

6 Opgave 3.68

Voor een woord $w \in \Sigma^*$, is een $\text{palc}(w)$ de korte palindroom x zodanig dat w een prefix is van x . en $\text{palc}(L) = \bigcup_{w \in L} \{\text{palc}(w)\}$.

Om te laten zien dat $\text{palc}(w) = wt^{-1}w^R$, waarbij wt^{-1} het woord is w waar het suffix t eraf gehaald is en t de langste palindroom suffix van w is, moet er gebruik gemaakt worden van een tegenstelling. Als w nog een palindroom aan het einde zal bevatten uu , dan ziet x er als volgt uit $wuuuuw$, dit is niet de korte palindroom (ww) , welke $uuuu$ er nog makkelijk uit weggehaald had kunnen worden. Een willekeurige suffix van w kan ook geen palindroom vormen die nog ‘weggesneden’ kan worden, omdat hij anders al in den beginne een palindroom had moeten zijn.

Als L is regulier, dan is $\text{palc}(L)$ ook regulier, welke er een unieke ‘vertaling’ van een woord w naar zijn $\text{palc}(w)$. De nieuwe woorden zullen in het slechtste geval evenveel blijven, maar de taal kan ook kleiner worden.

De andere kant op geldt dit niet, als L regulier is, dan is het onbeslist of $\text{palc}^{-1}(L)$ ook regulier is. Het kan namelijk zeer goed dat een (ingewikkelde) functie die woorden genereerde welke een palindroom zijn en die aan een vast prefix (p) toegevoegd. De palc functie zal deze allen naar één woord omzetten, welke dan regulier is.

7 Opgave 3.69

Laat $x_1, x_2, \dots, x_k \in \Sigma^*$. Om te laten zien dat $\Sigma^* - x_1^*x_2^* \cdots x_k^*$ eindig is dan en slechts als $|\Sigma| = 1$ en $\text{gcd}(|x_1|, |x_2|, \dots, |x_k|) = 1$ moet er een paar dingen bewezen worden a) aantonen dat het niet geldt voor de tegenvoorbeelden a) $\text{gcd} > 1$ b) $|\Sigma| > 1$ en tevens moet aangetoond worden waarom de eindigheid voor dit specifieke geval geldt.

Merk op dat met een gcd van 1 alle getallen in de verzameling $\{1, \text{priemgetallen}\}$ zitten. Bij een alfabet van 1 letter hoeft er enkel maar gesproken worden over lengtes. Standaard zitten alle lengtes in de taal. Aangezien de ‘deel-lengtes’ (x_1, x_2, \dots, x_k) nul of meer keer in de verzameling in de verzameling mogen zitten. Zal je uiteindelijk overblijven met een eindige set lengtes. In

het geval dat er lengte 1 gevonden wordt, zal zelfs de lege set het antwoord zijn.

a) Als $|\Sigma| > 1$, bijvoorbeeld $\{1, 2\}$ dan kan er een oneindige verzameling gemaakt worden, door $x_1, x_2, \dots, x_k \in 1^*$. De 2^* zal dan een oneindige verzameling vormen.

b) $\gcd > 1$, bijvoorbeeld 2 dan kan er een oneindige verzameling gemaakt worden, doordat niet alle woorden gemaakt kunnen worden. Enkel worden van 'even' lengte in dit geval. Waardoor de 'oneven' woorden een eindige string gaan vormen. Bij grotere waarden (r) kunnen ook 'groepen' (de $|r| * -1$) bijvoorbeeld niet meer bereikt worden, welke dan tot een oneindige verzameling kunnen groeien.

Referenties

[JS2009] Jeffrey Shallit, *A second course in formal languages and automata theory*, Cambridge University Press, 2009.