Logische Complexiteit
Uitwerkingen Deeltoets

Jeroen Goudsmit

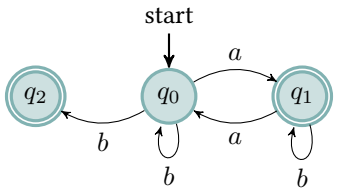
dinsdag 8 maart 2011

Hieronder uitwerkingen bij de deeltoets logische complexiteit 2010-2011. Dit document is bedoelt je te laten zien hoe je iedere vraag *had kunnen* beantwoorden, niet hoe je per se had moeten antwoorden. Als je fouten ziet, meldt dat dan a.u.b. bij Jeroen Goudsmit.

Opgave 1 — Construeer een automaat

Construeer een NFA met niet meer dan drie staten die de volgende taal herkent:

$$\{\ w \in \{a,b\}^* \mid \text{ als het aantal } a\text{'s in } w \text{ even is, eindigt } w \text{ op een } b\ \}.$$



Deze NFA houdt in q_0 en q_1 bij of er een even danwel oneven aantal a's voorbij gekomen is. Er mag een willekeurig aantal b's tussendoor verwerkt worden. Na een oneven aantal a's is de automaat altijd klaar om te accepteren, maar als je een even aantal a's gezien hebt moet je eest nog een a danwel b lezen alvorens je accepteren kan.

Opgave 2 — Beschrijf een Context-Vrije Taal

Omschrijf de taal van de volgende grammatica:

$$\begin{array}{lcl} \mathsf{S} & \to & 1\ \mathsf{S}\ 0 \mid \mathsf{A} \\ \mathsf{A} & \to & 0\ \mathsf{A}\ 1 \mid 1\ 0 \end{array}$$

Lemma 1. *De bovenstaande grammatica G herkent de taal*

$$\mathcal{L} := \{\ 1^i 0^j 101^j 0^i \mid i,j \in \mathbb{N}\ \}$$

Informeel bewijs. Een woord wordt van binnen naar buiten opgebouwd. Eerst kan i keer de regel $\mathsf{S} \to 1\ \mathsf{S}\ 0$ toegepast worden (resultaat $1^i\mathsf{S}0^i$). Dan de regel $\mathsf{S} \to \mathsf{A}$. Daarna j keer de regel $\mathsf{A} \to 0\ \mathsf{A}\ 1$ (resultaat $1^i0^j\mathsf{A}1^j0^i$). Ten slotte de enige terminerende regel $\mathsf{A} \to 1\ 0$ (resultaat $1^i0^j101^j0^i$). □

Formeel bewijs. We bewijzen voor iedere $w \in \Sigma^*$, waar $\Sigma = \{0,1\}$, dat $w \in \mathcal{L}(G)$ dan en slechts dan als $w \in \mathcal{L}$.

We bewijzen eerst dat voor alle $v \in \Sigma^*$ dat $\mathsf{A} \overset{*}{\Rightarrow} v$ geldt dan en slecht dan als $v = 0^j 101^j$ voor een zekere $j \in \mathbb{N}$. Van links naar rechts bewijzen we dit met inductie langs het aantal stappen n in de afleiding va $\mathsf{A} \overset{*}{\Rightarrow} v$. In het geval dat $n = 1$ dan is het duidelijk, want dan geldt $v = 0^0 100^0$. Voor $n = m + 1$ merken we op dat nu geldt

$$\mathsf{A} \Rightarrow 0\ \mathsf{A}\ 1 \Rightarrow 0\ u_1\ 1 \Rightarrow \ldots \Rightarrow 0\ u_m\ 1 = v$$

voor een zeker rijtje $u_1 \Rightarrow \ldots \Rightarrow u_m$ met $u_1, \ldots, u_m \in \left(\Sigma \cup \{\mathsf{A}\}\right)^*$. Nu weten we met inductie dat $u_m \Rightarrow 0^j 101^j$ voor een zekere $j \in \mathbb{N}$. Dus volgt ook dat $v = 00^j 101^j 1 = 0^{j+1} 101^{j+1}$ van de propere vorm is.

Om nu te bewijzen dat als $v = 0^j 101^j$ voor $j \in \mathbb{N}$, dan $\mathsf{A} \overset{*}{\Rightarrow} v$, gaan we met inductie over j. In het geval $j = 0$ dan staat er $v = 10$, en $\mathsf{A} \Rightarrow 01$ is duidelijk waar. Als $j = j' + 1$ dan zien we $v = 00^{j'} 101^{j'} 1$. Inductie garandeert ons dat $\mathsf{A} \overset{*}{\Rightarrow} 0^{j'} 101^{j'}$. Hiermee verkrijgen we het onderstaande, wat de claim bewijst.

$$\mathsf{A} \Rightarrow 0\ \mathsf{A}\ 1 \overset{*}{\Rightarrow} 0\ 0^{j'} 101^{j'}\ 1 = 0^{j'+1} 101^{j'+1} = v$$

Met dit resultaat in de hand maken we nu 't bewijs af. We gaan eerst van links naar rechts. Als $w \in \mathcal{L}$ dan is w van de vorm $1^i 0^j 101^j 0^i$ voor $i, j \in \mathbb{N}$. Met inductie over i kunnen we aantonen dat $\mathsf{S} \overset{*}{\Rightarrow} w$. Immers, als $i = 0$ dan volgt het direct uit $\mathsf{S} \Rightarrow \mathsf{A} \overset{*}{\Rightarrow} 0^j 101^j$ wat al bekend is. Als $i = i' + 1$ dan maken we net zo'n afleiding als hierboven.

Nu bewijzen we de implicatie van rechts naar links. Stel dat $\mathsf{S} \overset{*}{\Rightarrow} w$ in n stappen. Als $n = 2$ dan w gelijk zijn aan 10. Voor kleinere n wordt er niks afgeleid. En als $n = m + 3$ dan zien we

$$\mathsf{S} \Rightarrow 1\ \mathsf{S}\ 0 \overset{*}{\Rightarrow} 1\ v_1\ 0 \Rightarrow \ldots \Rightarrow 1\ v_m\ 0 = w \tag{1}$$

$$\mathsf{S} \Rightarrow \mathsf{A} \overset{*}{\Rightarrow} w \tag{2}$$

Stel we hebben te maken met afleiding (1). Dan geldt $v_1 \Rightarrow \ldots \Rightarrow v_m$, wat een afleiding is in m stappen. Hiermee volgt dat $v_m \in \mathcal{L}$, dus $v_m = 1^i 0^j 101^j 0^i$ voor zekere $i, j \in \mathbb{N}$. Als gevolg weten we dat $w = 1^{i+1} 0^j 101^j 0^{i+1} \in \mathcal{L}$.

Als we nu te maken hebben met afleiding (2), dan geldt $\mathsf{A} \overset{*}{\Rightarrow} w$. Wegens het voorgaande weten we nu dat $w = 0^j 101^j$ voor een zekere $j \in \mathbb{N}$. Dus is w zeker een element van $\mathcal{L}$. Dit bewijst de implicatie van links naar rechts. □

Opgave 3 — Pomplemma

Laat $\Sigma = \{a, b, c\}$.

Het pomplemma voor reguliere talen stelt: "als A een reguliere taal is, dan ..."

(i) Maak de bovenstaande zin af.

- is er een pomplengte p zodanig dat voor ieder woord $w \in A$ met $|w| \geq p$ geldt dat er $x, y, z \in \Sigma^*$ zijn met $w = xyz$, $|xy| \leq p$, $|y| \geq 1$ en $xy^i z \in A$ voor alle $i \in \mathbb{N}$.

(ii) Laat zien dat de taal $B := \{ab^n c^n \mid n \geq 0\}$ niet regulier is.

- Stel B is wel regulier. Dan is er een pomplengte p. Kies het woord $w = ab^p c^p$ en merk op $w \in B$ en $|w| = 2p + 1 \geq p$. Zij $x, y, z \in \Sigma^*$ zodanig dat $w = xyz$, $|xy| \leq p$, $|y| \geq 1$ en voor alle $i \in \mathbb{N}$ er geldt $xy^i z \in B$. Stel dat x leeg is. Dan begint y met a, en bevat z geen a. Dit betekent dat $xz = z$ zeker niet in de taal B zit, een tegenspraak. Dus begint x met a. We zien nu dat $x = ab^k$, $y = b^l$ en $z = b^m c^p$ met $k + l + m = p$ en $|y| = l \geq 1$. We weten dat $xy^0 z \in B$, dus moet gelden $ab^k b^m c^p \in B$. Dit kan enkel en alleen wanneer $n = k + m$. Maar omdat $n = k + m$ en $n = k + l + m$ volgt $l = 0$. Dit geeft ook een tegenspraak, namelijk met $l \geq 1$. Dus kan B helemaal niet regulier zijn.

(iii) Laat zien dat de taal C, gegeven als

$$C := \{a^i b^j c^k \mid \text{ als } i = 1 \text{ dan } j = k\},$$

niet regulier is. Hint: gebruik je antwoord op onderdeel (ii).

▶ Het is duidelijk dat $C \cap \mathcal{L}(ab^*c^*) = B$, waar $\mathcal{L}(ab^*c^*)$ staat voor de taal gegenereerd door de reguliere expressie ab^*c^*. Stel C is regulier, dan is B dat ook omdat reguliere talen gesloten zijn onder doorsnede. Maar B is niet regulier, zoals hierboven aangetoond. Dus is C dat ook niet.

(iv) Laat zien dat er een getal p bestaat zodat elk woord in C van lengte minimaal p de eigenschappen heeft die je genoemd hebt onder (i).

▶ Neem $p = 3$. Bekijk een willekeurig woord $w = a^i b^j c^k \in C$ met $|w| = i + j + k \geq p$. Als $i = 0$ dan nemen we x leeg en y gelijk aan het eerste teken van w en z de rest. Merk hier op dat er altijd zo'n teken is om in y te stoppen, omdat $j + k \geq p - i = 3$. Hoe vaak je y ook herhalen zal, 't resulterende woord blijft altijd in de taal. Als $i = 1$, dan nemen we $x = \varepsilon$, $y = a$ en $z = b^j c^k$. Ook hier werkt het herhalen prima. Als $i \geq 2$ dan nemen we $x = a^2$, y het eerste teken wat na de eerste twee tekens komt en z de rest. Merk op dat $|xy| = 2 + 1 \leq 3 = p$. Het steeds herhalen van y zal ervoor zorgen dat ofwel het aantal a's ofwel het aantal b's opgeblazen wordt, beiden geen probleem omdat er niet-een a's zijn.

(v) Leg uit waarom (iii) en (iv) niet met elkaar in tegenspraak zijn.

▶ Het pomplemma garandeert enkel het bestaan van een pomplengte voor reguliere talen. Ze zegt niks over hoe niet-reguliere talen er uit zullen zien. Die mogen dus ook best een pomplengte hebben.

Opgave 4 — Constructies op talen

Gegeven een taal $\mathcal{L} \subseteq \Sigma^*$ definieren we:

$$\mathcal{L}^\diamond := \{xy \mid x \in \mathcal{L}, y \in \Sigma^* \text{ en } |x| = |y|\}.$$

Bewijs de volgende claim.

Lemma 2. *Als $\mathcal{L}$ regulier is dan is $\mathcal{L}^\diamond$ context-vrij.*

Bewijs. De taak $\mathcal{L}$ is regulier, dus is er een DFA $M = \langle Q, \Sigma, \delta, q_0, F \rangle$. We geven nu een PDA $\underline{M}$ zodanig dat $\mathcal{L}(\underline{M}) = \mathcal{L}^\diamond$. Het idee is dat de PDA eerst een woord uit $\mathcal{L}$ inleest en onthoud hoe lang dit woord is. Vervolgens worden net zoveel tekens weer gelezen, en accepteert de machine.

Hiertoe definiëren we $\underline{Q} = Q \cup \{q_s, q_*, q_f\}$ en $\Gamma = \{*, \$\}$. De transitiefunctie $\underline{\delta}$ stellen we als volgt in, en verder is ze leeg. Deze functie $\underline{\delta}$ zorgt ervoor dat vanuit q_s je naar een kopie van de automaat M gaat die bijhoud hoeveel er gelezen is. Vanaf een accepterende toestand in M ga je naar q_*, waar een willekeurig woord gegeten kan worden. Uiteindelijk, als de stack leeg is, ga je naar q_f om te accepteren. In symbolen komt dat neer op:

$$\begin{array}{rcll}
\underline{\delta}(q_s, \varepsilon, \varepsilon) & = & \{\langle q_0, \$ \rangle\} & \\
\underline{\delta}(q, x, \varepsilon) & = & \{\langle r, * \rangle\} & \text{waar } \delta(q, x) = r \text{ voor } q, r \in Q \text{ en } x \in \Sigma \\
\underline{\delta}(q, \varepsilon, \varepsilon) & = & \{\langle q_*, \varepsilon \rangle\} & \text{waar } q \in F \\
\underline{\delta}(q_*, x, *) & = & \langle q_*, \varepsilon \rangle & \text{voor } x \in \Sigma \\
\underline{\delta}(q_*, \varepsilon, \$) & = & \langle q_f, \varepsilon \rangle &
\end{array}$$

Dit geeft ons alle benodigde informatie om te zeggen $\underline{M} := \langle \underline{Q}, \Sigma, \Gamma, \underline{\delta}, q_s, \{q_f\} \rangle$. Laten we nu bewijzen dat $\mathcal{L}(\underline{M}) = \mathcal{L}^\diamond$.

Stel $w \in \mathcal{L}^\diamond$, dan zijn er $u \in \mathcal{L} = \mathcal{L}(M)$ en $v \in \Sigma^*$ van gelijke lengte. Dus is er een accepterende berekening $q_0 = r_0, r_1, \ldots, r_n \in F$ van $u = u_1 \ldots u_n$ op M. Dit toont aan dat

$$\begin{aligned} q_\mathrm{s}, u_1 \ldots u_n, v_1 \ldots v_n, \varepsilon &\vdash q_0, u_1 \ldots u_n, v_1 \ldots v_n, \$ \\ &\vdash \delta(q_0, u_1), u_1 \ldots u_n, v_1 \ldots v_n, *\$ \vdash \ldots \vdash q, v_1 \ldots v_n, *^n\$ \\ &\vdash q_*, v_1 \ldots v_n, *^n\$ \vdash \ldots \vdash q_*, \varepsilon, \$ \\ &\vdash q_\mathrm{f}, \varepsilon, \varepsilon, \end{aligned}$$

waarmee w geaccepteerd wordt, dus $w \in \mathcal{L}(\underline{M})$

Andersom, stel dat $w \in \mathcal{L}(\underline{M})$. Dit betekent dat er een accepterende berekening is van $w = w_1 \ldots w_n$. Gebruik makende van $q_\mathrm{s}, w, \varepsilon \overset{*}{\vdash} q_\mathrm{f}, \varepsilon, w'$ voor $w' \in \Gamma^*$ kunnen we aantonen dat $w = uv$ voor $u \in \mathcal{L}$ en u en v van gelijke lengte. We zien dat er een m is zodanig dat in $m + 2$ stappen er geldt

$$q_\mathrm{s}, w, \varepsilon \overset{*}{\vdash} q_*, w_{m+1} \ldots w_n, *^n\$,$$

met $u := w_1 \ldots, w_m \in \mathcal{L}$. Als $m \geq n$ dan volgt er direct een tegenspraak, want dan kan de automaat nooit meer ontspannen uit q_*, wat geen accepterende toestand is. Vanaf hier zijn de enige n volgende mogelijke stappen zodanig dat

$$q_*, w_{m+1} \ldots w_n, *^n\$ \overset{*}{\vdash} q_*, w_{n-m-m+1} \ldots w_n, \$.$$

Ook hier zie je dat $2m \leq n$ moet gelden. Sterker nog, $2m \geq n$, want anders staan er nog $*$'s op de stack, en zit de automaat ook vast in q_*. Dus weten we $2m = n$, waarmee er geen letter meer te lezen over is. Dus is w gelijk aan uv voor $v = w_{m+1} \ldots w_{2m}$, zodat u en v dezelfde lengte hebben. Dit toont dan weer aan dat $w = uv \in \mathcal{L}^\diamond$ □

Opgave 5 — 2-PDA's

Een 2-PDA is een PDA met twee stacks. Voor een 2-PDA heeft de transitiefunctie dus de vorm

$$\delta \colon Q \times \Sigma_\varepsilon \times \Gamma_\varepsilon \times \Gamma_\varepsilon \to \mathcal{P}\big(Q \times \Gamma_\varepsilon \times \Gamma_\varepsilon\big),$$

waarbij $(r, u, v) \in \delta(q, a, x, y)$ betekent:

> "als in toestand q letter a gelezen wordt en x is de top van stack 1 en y de top van stack 2, dan mag je het volgende doen: pop x en push u op stack 1, pop y en push v op stack 2, en ga naar toestand r."

Leg uit dat elke TM-herkenbare taal de taal van een of andere 2-PDA is. Een informeel argument volstaat.

▶ Als een taal TM-herkenbaar is, dan is er een Turing machie die 'm herkent. Zo'n Turing machine kunnen we altijd simuleren op een 2-PDA. Dat gaat vrij eenvoudig. De oneindige tape van een Turing machine vouw je in gedachte naar beneden om het punt waar de TM staat. Zo krijg je dus twee stacks, een met de tape links (boven-naar-onder is rechts-naar-links) en een met de tape rechts (boven-naar-onder is links-naar-rechts). Een stap naar links op de Turing machine simuleer je met een pop in de linker stack en een push in de rechter stack, en een stap naar rechts gaat andersom, dus een push links en pop rechts.