Automaten en Berekenbaarheid: opgave 2

Prof. B. Demoen (Bart.Demoen@cs.kuleuven.be)
W. Van Onsem (Willem.VanOnsem@cs.kuleuven.be)

11 december 2014

Richtlijnen Je schrijft enkel op de gekleurde bladen die je gekregen hebt: op je werkplek liggen geen andere bladen dan die gekleurde bladen en de bladen van de cursustekst. De bladen van je cursustekst waarop oplossingen van oefeningen of aanwijzingen tot oplossingen van oefeningen staan, steek je ook weg. Je jas, je boekentas, je gsm... leg je vooraan in het auditorium, zoals gebruikelijk bij een examen. Bij het afgeven zorg je dat je naam op je bladen staat, en dat je alles aan elkaar niet. Er zijn 4 vragen.

Vraag 1 (Berekenbaarheid)**.** Zijn volgende talen herkenbaar, co-herkenbaar en/of beslisbaar? Indien wel, beschrijf een beslisser/herkenner. Indien niet, toon dit aan (bijvoorbeeld aan de hand van een reductie). Σ slaat altijd op het invoeralfabet van een machine. L_M slaat op de taal die beslist wordt door M. De machines zijn allemaal Turing machines.

1. $\{\langle M\rangle \mid \exists s_1, s_2 \in L_M : |s_1| \neq |s_2|\}$;
2. $\big\{\langle M, s\rangle \mid M \text{ stopt na hoogstens } 2^{|s|} \text{ stappen bij invoer } s\big\}$;
3. $\big\{\langle M\rangle \mid M \text{ stopt voor iedere invoer } s \in \Sigma^\star \text{ na hoogstens } 2^{|s|}\big\}$; en
4. $\{\langle M, y\rangle \mid M \text{ beweegt de lees/schrijf-kop ooit naar links bij invoer } y\}$.

Antwoord.

1. $\{\langle M\rangle \mid \exists s_1, s_2 \in L_M : |s_1| \neq |s_2|\}$
   (a) **niet beslisbaar**: dit volgt rechtstreeks uit de *Stelling van Rice*. De taal is niet *triviaal*: we kunnen een Turing machine construeren die de taal $\{a, aa\}$ beslist en een Turing machine die de taal $\{a\}$ beslist. De ene machine zit in de taal de andere niet. De taal is ook *taal-invariant*: er zijn geen aspecten van de Turing machine opgenomen in de voorwaarde.
   (b) **herkenbaar**: We beschrijven een herkenner. Deze herkenner werkt in fases beginnend met fase 0. In fase i wordt er voor elke string $s \in \Sigma^i$ een Turing machine opgestart (gesimuleerd) en worden alle tot dusver begonnen Turing machines een stap verder gezet. Wanneer er twee Turing machines accepteren en de strings hebben een verschillende lengte, accepteren we ook.
   (c) **niet co-herkenbaar**: immers is de taal herkenbaar en niet beslisbaar.
2. $\big\{\langle M, s\rangle \mid M \text{ stopt na hoogstens } 2^{|s|} \text{ stappen bij invoer } s\big\}$
   (a) **beslisbaar**: We beschrijven een beslisser. Laat M lopen op s als na $2^{|s|}$ stappen nog is afgelopen, verwerpen we $\langle M, s\rangle$; indien de simulatie stopt voor of op $2^{|s|}$ stappen, accepteren we.
   (b) **herkenbaar**: volgt uit het feit dat de taal beslisbaar is.
   (c) **co-herkenbaar**: volgt uit het feit dat de taal beslisbaar is.

3. $\{\langle M\rangle \mid M \text{ stopt voor iedere invoer } s \in \Sigma^\star \text{ na hoogstens } 2^{|s|}\}$

   (a) **niet-beslisbaar**: We tonen eerst aan dat $\textsc{AllHalt}_{\text{TM}}$ niet herkenbaar is.

   Bewijs. Indien $\textsc{AllHalt}_{\text{TM}}$ herkenbaar zou zijn, zou All_{TM} herkenbaar zijn. Stel een herkenner H voor $\textsc{AllHalt}_{\text{TM}}$. Bij invoer $\langle M\rangle$ passen we M aan tot M' zodat indien M zou verwerpen, deze in een oneindige lus zou terecht komen. Dit betekent dus dat elke string s die M niet zou accepteren (het zij actief verwerpen of in een oneindige lus terecht komt), bij M' sowieso in een oneindige lus terecht komen. M' wordt herkent door H wanneer M' nooit in een oneindige lus terecht komt, en M dus alle strings accepteert. All_{TM} is echter noch herkenbaar noch co-herkenbaar, bijgevolg is $\textsc{AllHalt}_{\text{TM}}$ ook niet herkenbaar. □

   Indien L beslisbaar was met een beslisser B, dan konden we een herkenner bouwen voor $\textsc{AllHalt}_{\text{TM}}$.

   Bij invoer $\langle M\rangle$ geven we $\langle M\rangle$ eerst mee aan de beslisser voor onze taal B. Wanneer B accepteert is het evident dat M op alle strings zou stoppen: het stopt immers binnen de $2^{|s|}$ voor elke string s. Indien B verwerpt wil dit natuurlijk niet zeggen dat M niet uiteindelijk zou stoppen op alle invoer: misschien heeft de machine meer tijd nodig om sommige strings te beslissen. In dat geval passen we M aan naar M'. Elke string moet worden vooraf gegaan door een speciaal karakter $\supset$: wanneer dit niet aanwezig is op de tape, verwerpen we meteen. In het andere geval verwijderen we dit teken en laten we M lopen op de rest van de string. Door alle invoer dus te laten voorafgaan met één karakter, geven we de machine M' dubbel zoveel tijd om de string te beslissen. We draaien opnieuw B op M' als B accepteert dan accepteren we, indien niet, passen we de machine opnieuw aan tot B ooit accepteert.

   Door herhaaldelijk de machine aan te passen, kopen we extra tijd voor M tot dat het uiteindelijk voldoende is om de machine M alle strings te laten beslissen. We accepteren enkel wanneer dit het geval is.

   (b) **niet herkenbaar**: Volgt uit het feit dat we een co-herkenner kunnen beschrijven en de taal niet beslisbaar is.

   (c) **co-herkenbaar**: We beschrijven een co-herkenner. De co-herkenner laat zoals bij 1b in verschillende fases Turing machines lopen. Het verschil is echter dat we Turing machines laten stoppen na $2^{|s|}$ stappen, ook indien de machine zelf nog niet zo stoppen. Wanneer zo'n machine echter meer tijd nodig heeft, accepteren we.

4. $\{\langle M, y\rangle \mid M \text{ beweegt de lees/schrijf-kop ooit naar links bij invoer } y\}$

   (a) **beslisbaar**: we beschrijven een beslisser. We simuleren de Turing machine M met een soort debugger (een programma die M een stap laat zetten, maar nadien het geheugen, etc. kan inspecteren). In eerste instantie draaien we de Turing machine tot deze voorbij de invoer leest. Indien de Turing machine probeert naar links te bewegen, dan verwerpen we natuurlijk onmiddellijk. Verder kunnen we ook lusdetectie uitvoeren: wanneer de Turing machine blijft stilstaan houden we de combinaties van toestanden en karakters die onder de tape staan bij. Wanneer we eenzelfde koppel een tweede maal tegenkomen, weten we dat we in een oneindige lus zitten en accepteren we: immers leidt deze lus nooit tot het naar links bewegen van de kop.

   Wanneer we het invoergedeelte voorbij gaan is er één variant: rechts van de lees/schrijfkop staan altijd lege symbolen: immers mogen we niet naar links gaan. We weten verder ook in welke toestand we zitten bij het verlaten van het invoergedeelte. Met deze toestand en de invariant kunnen we bepalen welke toestanden we allemaal zullen aandoen: wanneer we naar rechts gaan, nemen we bij de volgende toestand uiteraard de overgang met het lege karakters als karakter onder de tape. Wanneer de tape blijft stilstaan, rekenen we door met het karakter dat we net

hebben weggeschreven. Wanneer we zo een boog zullen bereiken die naar links gaat, verwerpen we; anders accepteren we. Merk op dat we dit gedeelte niet simuleren: we voeren een analyse uit op de toestandsgraaf.

(b) **herkenbaar**: volgt uit het feit dat de taal beslisbaar is.

(c) **co-herkenbaar**: volgt uit het feit dat de taal beslisbaar is.

◊

Vraag 2 (Inwendige operaties)**.** Stel twee verschillende niet-beslisbare talen L_1 en L_2, geef voor elke uitspraak hieronder aan:

1. indien de uitspraak waar is voor elke L_1 en L_2, waarom (bewijs);
2. indien de uitspraak afhangt van de keuze voor L_1 en L_2: twee voorbeelden (voor het geval waar de uitspraak klopt en voor het geval waar de uitspraak niet klopt); en
3. indien de uitspraak niet klopt, een tegenvoorbeeld of reden.

Uitspraken:

1. $L_1 \setminus L_2$ is herkenbaar;
2. $L_1 \cap L_2$ is beslisbaar; en
3. $L_1 \cup L_2$ is herkenbaar.

Antwoord.

1. Dit hangt af van de talen:
   (a) Stel $L_1 = A_{\text{TM}}$ en $L_2 = H_{\text{TM}}$: het is zo dat $A_{\text{TM}} \subsetneq H_{\text{TM}}$ immers moet voor elk koppel $\langle M, s\rangle$ M stoppen bij invoer s om s te accepteren. Bijgevolg is $L_1 \setminus L_2$ leeg, dus beslisbaar en herkenbaar.
   (b) Stel $L_1 = \overline{A_{\text{TM}}}$ en $L_2 = A_{\text{TM}}$, dan is $L = L_1 \setminus L_2 = \overline{A_{\text{TM}}}$, deze taal is niet herkenbaar, maar co-herkenbaar.
2. Dit hangt af van de talen:
   (a) Stel $L_2 = \overline{L_1}$. In dat geval is $L_1 \cap L_2 = \emptyset$ en dit is beslisbaar.
   (b) Stel $L_1 = A_{\text{TM}}$ en $L_2 = H_{\text{TM}}$ dan is $L_1 \cap L_2 = A_{\text{TM}}$ en dit is niet beslisbaar.
3. Dit hangt af van de talen:
   (a) Stel $L_2 = \overline{L_1}$. In dat geval is $L_1 \cup L_2 = \Sigma^\star$ en dit is beslisbaar en dus herkenbaar.
   (b) Stel $L_1 = \overline{A_{\text{TM}}}$ en $L_2 = \overline{H_{\text{TM}}}$, dan is $L_1 \cup L_2 = \overline{H_{\text{TM}}}$ en dit is niet herkenbaar.

◊

Vraag 3 (synchrone PDA)**.** Een sPDA is een PDA met 2 stapels, maar waarbij stapels synchroon moeten bewegen: beide stapels "pushen" en "poppen" op hetzelfde ogenblik. Het is met andere woorden niet toegelaten dat één stapel groeit terwijl de andere wordt afgebouwd. Is een sPDA strikt[1] sterker dan een gewone PDA? Bewijs je antwoord.

Antwoord. Een sPDA is niet sterker dan een PDA. We bewijzen dat beide automaten even krachtig zijn door constructies tussen een sPDA en PDA te geven in beide richtingen.

1. Een PDA naar een sPDA.

[1] Met "strikt" bedoelen we dat een sPDA minstens één taal meer kan beslissen dan een gewone PDA.

Constructie 1. Gegeven een PDA $\langle Q, \Sigma, \Gamma, \delta, q_s, F\rangle$, we beschouwen een PDA $\langle Q, \Sigma, \Gamma, \delta', q_s, F\rangle$ die dezelfde taal beslist met volgende overgangsregels:

$$\forall q \in Q, \sigma \in \Sigma, \gamma, \gamma_2 \in \Gamma_\epsilon : \delta'(q, \sigma, \gamma, \gamma_2) = \{\langle q', \gamma', \gamma'\rangle \mid \langle q', \gamma'\rangle \in \delta(q, \sigma, \gamma)\} \tag{1}$$

We hebben hier gekozen om dezelfde inhoud te "pushen" op de tweede stapel (maar men zou bijvoorbeeld ook altijd hetzelfde karakter kunnen pushen). Er wordt verder nooit gekeken naar het karakter op de tweede stapel: het is eender wat daar staat, al weten we dat dit altijd hetzelfde is als de eerste stapel.

2. Een sPDA naar een PDA.

   Constructie 2. Gegeven een sPDA $\langle Q, \Sigma, \Gamma, \delta, q_s, F\rangle$ we beschouwen een PDA $\langle Q, \Sigma, \Gamma \times \Gamma, \delta', q_s, F\rangle$

$\Diamond$

Vraag 4 (Bezige bever)**.** Stel dat de bezige bever functie $S(n)$ toch berekenbaar is. Beschrijf een Turing machine op hoog niveau die A_{TM} kan beslissen.

Antwoord. De bezige bever beschrijft het maximale aantal stappen dat een Turing machine met alfabet $\{0, 1\}$ kan zetten, zonder in een oneindige lus te geraken. We kunnen dus een (gemodificeerde) Turing machine laten lopen en het aantal tot dusver gezette stappen bijhouden. Wanneer dit aantal over dat van de bezige bever gaat,

Er zijn echter enkele problemen: de Turing machine kan een willekeurig alfabet hebben, en de invoer staat aanvankelijk op de tape, de tape is dus niet leeg.

Het eerste probleem kunnen we oplossen door het alfabet Σ om te zetten naar $\Sigma' = \{0, 1\}$. We kunnen dit doen door een *encodering* af te spreken, ASCII bijvoorbeeld schrijft a als 0011 1101. Het impliceert verder dat we de Turing machine wat aanpassen: waar we vroeger een karakter inlazen met één operatie, zullen we nu verschillende bits moeten inlezen, en aan de hand van toestanden bijhouden wat we al gezien hebben, ook zullen we dus de lees/schrijf-kop meer naar links/rechts moeten bewegen. Dit introduceert extra toestanden.

Het tweede probleem kunnen we oplossen door de Turing machine verder aan te passen: we laten de Turing machine zelf eerst de invoer-string op de tape schrijven en terugkeren naar het begin. Daarna pas laten we "originele" machine lopen.

Beide oplossingen introduceren extra toestanden, maar dat is geen probleem met betrekking tot de beslisbaarheid. We kunnen vervolgens het aantal toestanden van de resulterende Turing machine tellen, de bezige bever $S(n)$ uitrekenen, en de Turing-machine voor dat aantal stappen simuleren. $\Diamond$