Examen Automaten en Berekenbaarheid – Bespreking

Prof. Dr. Bart Bogaerts
Mathieu Reymond

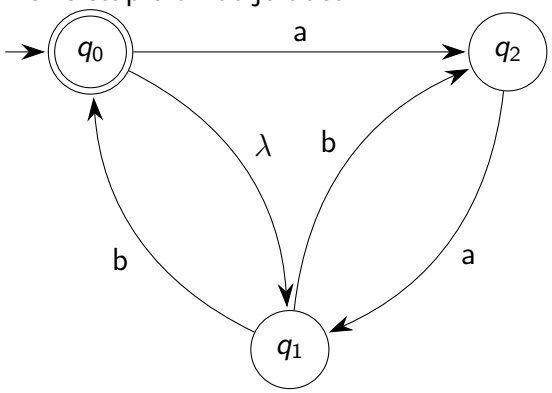
1/7/2020

Wegens corona-maatregelen was dit examen lichtjes abnormaal: het was een mix van oefeningen (vraag 1–3) en theorie (vraag 4–5). In dit document bespreek ik kort de voornaamste pitfalls, veel voorkomende fouten en hoe correcte antwoorden bekomen konden worden.

Eerst enkele algemene statistieken. Van de 62 deelnemers waren er 21 geslaagd, 33% dus. De hoogst behaalde score was 17.

Vraag 1

Zet onderstaande NFA (non-deterministic finite automaton) om naar een **minimale** DFA (deterministic finite automaton), door gebruik te maken van de geziene procedures. Teken het resultaat.
Leg in elke stap uit wat je doet.



Bespreking

Deze vraag was optioneel en was een "weggevertje". Het enige wat hier moest gebeuren was twee geziene procedures toepassen op een nieuwe automaat. Dit is uitgebreid ingeoefend in de WPOs.

De reden om deze optioneel te maken was: een derglijke oefening vereist best wat schrijfwerk en wie de cursus goed begrepen heeft, kan zijn tijd nuttiger spenderen in het maken van de andere vragen (die meer inzicht vereisen).

De scores op deze vraag lagen erg hoog. Heel wat mensen haalden hier de maximumscore.

Vraag 2

Zijn de volgende talen over het vocabularium $\{a, b\}$ regulier? Zijn ze context-vrij? Zijn ze context-sensitief? Zijn ze (semi-)beslisbaar? Toon al jouw antwoorden aan (voor positieve antwoorden bijvoorbeeld door een grammatica of automaat van de juiste soort op te stellen of door geziene eigenschappen te gebruiken; voor negatieve antwoorden bijvoorbeeld door een pompstelling, een reductie van het halting probleem, of andere geziene eigenschappen te gebruiken). Wanneer je Turing machines construeert mag je een hoogniveau beschrijving van het gedrag geven zonder de transitiefunctie in detail uit te schrijven. Je mag dan gebruik maken van bouwblokken die simpele operaties doen (bijvoorbeeld een TM die optelling of vermenigvuldiging doet).

1. $\{a^p \mid p = n!$ voor een positief natuurlijk getal $n\}$ (herinner dat $n! = 1 \cdot 2 \cdot ... \cdot n$)
2. $\{a^i b^j a^k \mid i + j \neq k\}$
3. $\{w \in \{a, b\}^* \mid$ elke opeenvolging van a's in w bevat minstens 3 a's$\}$

Bespreking

Deze vraag vroeg wat meer inzicht en creativiteit en leek moeilijker bevonden dan de eerste vraag. Ze werd op 10 gekwoteerd: 1 punt voor correct gebruik "chomsky hierarchie" (daarmee bedoel ik bijvoorbeeld dat eens aangetoond is dat een taal context-vrij is, men automatisch ook kan concluderen dat ze ook context-sensitief, en... is; analoog, na aantonen dat een taal **niet** context-vrij is, kan men snel besluiten dat ze dus ook niet regulier is) en 3 punten voor elke deelopgave.

1. Voor de eerste vraag kon je een Turing-machine opstellen met drie tapes. Op de ene tape wordt een counter (n) bijgehouden; op de tweede tape wordt $n!$ bijgehouden en op de derde tape staat de input. De machine werkt als volgt: start met $n = 1$; bereken $n! = 1$; check of dit gelijk is aan de input; indien ja, stop in een finale state. Indien neen: increment n met 1, herbereken $n!$ (door te vermenigvuldigen met n), en check of dit gelijk is aan de input. Indien ja, stop in een finale state. Indien neen: repeat wat we net gedaan hebben.

   - Met deze machine kan je concluceren dat de taal recursief enumereerbaar is.
   - Door additioneel deze machine wat vroeger te laten stoppen (zodra $n!$ groter wordt dan de input kan je afbreken), kan je garanderen dat je machine altijd stopt. Dus de taal is zeker recursief.
   - Als je ze snel genoeg afbreekt (zie vorig punt), neemt je machine bovendien (op elke tape) hoogstens zoveel plaats in als de input. Dus dit is een LBA. (Vraag jezelf eens af: hoe zit dit met meerdere tapes?). Dus de taal is context-sensitief.

   Een hoogniveau beschrijving van de werking van de 3-tape turing machine volstond voor dit onderdeel. Daarnaast moest je uiteraard ook nog beargumenteren dat de taal context-sensitief is. Om aan te tonen dat ze niet context-vrij is, kon je de pompstelling (rechtsreeks) toepassen.

2. Voor deze vraag kon je gemakkelijk aantonen dat ze context-vrij is. De meeste mensen hebben dit gedaan ahv een push-down automaat. Maar, dit doen vereist rekening houden

met wat subtiliteiten (heb je rekening gehouden met het feit dat i, j en k misschien nul kunnen zijn?), en bij sommige oplossingen zat er een off-by-one error in de laatste stap (beslissen of er nu geaccepteerd moet worden of niet. Een gemakkelijkere oplossing is gebruik maken van een context-vrije grammatica. Voor de taal $L_1 = \{a^i b^j a^k \mid i+j > k\}$ kon je volgende grammatica opstellen

$$\begin{aligned} S_1 &\to aS_1a|T_1 \\ T_1 &\to bT_1a|A \\ A &\to aA|a \end{aligned}$$

waarbij de eerste twee regels gelijkheid van aantal $i+j$ en k behouden en de laatste regel zorgt dat er strikt meer as op het eind staan. Analoog kon je voor de taal $L_1 = \{a^i b^j a^k \mid i+j < k\}$ de volgende grammatica opstellen

$$\begin{aligned} S_2 &\to aS_1a|XT_2 \\ T_2 &\to bT_2a|\lambda \\ X &\to aX|Xb|a|b \end{aligned}$$

waarbij de eerste twee regels gelijkheid van aantal $i+j$ en k behouden en de laatste regel ervoor zorgt dat er vooraan nog minstens 1 extra a of b toegevoegd wordt (mogelijks meer). Deze kan je dan gemakkelijk tot een grammatica voor de volledig taal hervormen ($S \to S_1|S_2$). De oplossing met grammaticas is (naar mijn mening) eenvoudiger dan die met automaten. Het is belangrijk om bij een dergelijke vraag eerst even goed na te denken welke route de gemakkelijkste is.

Om aan te tonen dat deze taal **niet regulier** is, kon je de pompstelling gebruiken. Dit is echter niet evident te doen op de taal $L = \{a^i b^j a^k \mid i+j \neq k\}$ zelf. In de plaats kon je dit doen op $\bar{L}$. Immers: als $\bar{L}$ niet regulier is, dan is L ook niet regulier. Een subtiliteit hier is dat

$$\bar{L} \neq \{a^i b^j a^k \mid i+j = k\}$$

Je kan dit bijvoorbeeld inzien doordat $aa \in L$ ($i=2, j=0, k=0$) en ook $aa \in \{a^i b^j a^k \mid i+j=k\}$ (met $i=1, j=0, k=1$). Ook is het zo dat $abab \notin L$ en $abab \notin \{a^i b^j a^k \mid i+j=k\}$. Maar zelfs zonder expliciete beschrijving van $\bar{L}$ was het toepassen van de pompstelling niet moeilijk. Je kon bijvoorbeeld de string $a^m b^m a^{2m} \in \bar{L}$ beschouwen. Als deze opgepompt wordt, is het niet moeilijk in te zien dat men buiten $\bar{L}$ (in L dus) terecht komt.

3. De laatste vraag was een weggevertje. Daar kon men gemakkelijk een reguliere expressie (of een DFA/NFA) voor opstellen. Bijv $(aaa(a^*) + b)^*$.

Vraag 3

Is het volgend probleem beslisbaar? Indien ja, geef een Turing machine die dit probleem beslist. Je mag hiervoor gebruik maken van gezien uitbreidingen zoals meerdere tapes. Indien neen, toon aan door middel van een reductie van een probleem waarvan we gezien hebben dat het onbeslisbaar is.

Input: Een (encodering van) een reguliere expressie r. Een (encodering van) een standaard Turing machine M.

Probleem: Accepteert M alle strings uit $L(r)$?

Bespreking

Reducties en gelijkaardige vragen worden vaak als moeilijk bevonden. Het antwoord kan meestal kort zijn, maar het vinden vereist wat inzicht en creativiteit. De gemakkelijkste oplossing voor deze vraag was extreem kort en gaat als volgt:

> Het **element-van-de-taal** probleem "gegeven w en $enc(M)$, is $w \in L(M)$" is onbeslisbaar. Immers, als dit probleem beslisbaar zou zijn, dan zou elke recursief enumereerbare taal recursief zijn (zie je in waarom?).
>
> **Als** het probleem uit de opgave beslisbaar zou zijn, dan kunnen we het element-van-de-taal probleem op te lossen: elke string is zelf een reguliere expressie. (indien niet duidelijk, werk dit nog even uit?)

De meeste mensen hebben echter een reductie van halting of uniform halting gemaakt. Dit kan ook: de enige pitfall die daarin zat is dat er een verschil is tussen "stoppen" en "accepteren". Maar om deze moeilijkheid te omzeilen kan je een machine M aanpassen naar een machine M' die zich hetzelfde als M gedraagt, maar altijd als ze halt in een finale state terecht komt.

Vraag 4

Deze vraag bestaat uit 24 meerkeuzevragen. Er wordt giscorrectie toegepast. Elk juist antwoord levert je een kwart van een punt op. Voor elk fout antwoord verlies je een kwart van een punt.

Beschouw de volgende klasses talen:

- `Reg`: de klasse van reguliere talen
- `CF`: de klasse van contextvrije talen
- `dCF`: de klasse van deterministische contextvrije talen
- `Rec`: de klasse van recursieve talen
- `RE`: de klasse van recursief enumereerbare talen
- `CS`: de klasse van contextsensitieve talen

En beschouw over elke klasse $\mathcal{C}$ de volgende beweringen:

- **Comp**: de klasse talen is gesloten onder het nemen van complement. Dit wil zeggen, als $L \in \mathcal{C}$, dan ook $\overline{L} \in \mathcal{C}$.
- **Int**: de klasse talen is gesloten onder het nemen van doorsnedes. Dit wil zeggen, als $L_1 \in \mathcal{C}$ en $L_2 \in \mathcal{C}$, dan ook $L_1 \cap L_2 \in \mathcal{C}$.
- **RegInt**: de klasse talen is gesloten onder het nemen van doorsnede met een reguliere taal. Dit wil zeggen, als $L_1 \in \mathcal{C}$ en $L_2 \in \texttt{Reg}$, dan ook $L_1 \cap L_2 \in \mathcal{C}$.
- **Subset**: de klasse talen is gesloten onder het nemen van deelverzamelingen. Dit wil zeggen, als $L_1 \in \mathcal{C}$ en $L_2 \subseteq L_1$, dan ook $L_2 \in \mathcal{C}$.

Vul voor alle **lege** vakjes in de volgende tabel in of de bewering waar is voor de overeenkomstige klasse talen of niet. Bijvoorbeeld, voor het vakje gemarkeerd als "YYYYYYY" dat **niet** ingevuld dient te worden zou het antwoord "ja" zijn. Immers de klasse van reguliere talen is gesloten onder het nemen van complement; het complement van een reguliere taal is opnieuw regulier.

Het vakje gemarkeerd "ZZZZZZZ" hebben we niet in de cursus gezien, maar is ook waar: de klasse van context-sensitieve talen is gesloten onder complement.

De vakjes gemarkeerd "XXXXXXX" moet je ook **niet** invullen. Sommige daarvan zijn waar, andere niet.

	Reg	CF	dCF	Rec	RE	CS
Comp	YYYYYYY	VALS	WAAR	WAAR	VALS	ZZZZZZZ
Int	WAAR	VALS	VALS	WAAR	XXXXXXX	WAAR
RegInt	XXXXXXX	WAAR	WAAR	XXXXXXX	XXXXXXX	WAAR
Subset	VALS	XXXXXXX	XXXXXXX	VALS	XXXXXXX	XXXXXXX

Duid bovendien voor elk van de volgende beweringen aan of ze waar zijn of niet:

Bewering	Waar/fout
Voor elke ambigue context-vrije grammatica bestaat er een niet-ambigue context-vrije grammatica die dezelfde taal definiëert	FOUT
Voor elke reguliere taal L_r bestaat er een minimale NFA (een NFA M met $L(M) = L_r$ waarvoor er geen NFA met minder toestanden L ook accepteert)	WAAR
Elke links-lineaire grammatica definiëert een context-vrije taal	WAAR
Er bestaan geen reguliere talen over een leeg alfabet	FOUT
Het is voor geen enkele Turing machine M mogelijk om te beslissen of M eindigt met de lege string als input of niet	FOUT
Voor elke context-vrije taal bestaat er een context-vrije grammatica die zowel in Chomsky normaalvorm als in Greibach normaalvorm is	FOUT
Voor elke standaard Turing machine M en elke string w bestaat er een machine H die altijd stopt zodat H in een finale toestand stopt als en slechts als M stopt op w.	WAAR
De pompstelling voor context-vrije talen geldt ook voor reguliere talen.	WAAR
Als L een recursieve taal is, en M een standaard Turing machine die L accepteert, dan stopt M altijd.	FOUT
Context-vrije talen zijn altijd oneindig	FOUT

Bespreking

Ongeveer de helft van de vakjes in de eerste tabel waren letterlijk gezien in de cursus en dus "weggevertjes". De andere helft vereiste wat inzicht om tot de juiste conclusie te komen.

- Alle groen ingevulde vakjes waren letterlijk gezien.

- Om het blauwe vakjes in te zien kon je bijvoorbeeld geziene constructies toepassen op een andere soort automaten. Hiervoor moest je de constructies/bewijzen natuurlijk goed begrijpen.

  Bijvoorbeeld voor het eerste blauwe vakje kon je de gezien constructie voor complement van een reguliere taal (finale states niet-finaal maken en niet-finale state finaal maken) toepassen op een deterministische PDA.

  Bijvoorbeeld voor Rec–**Int** kon je de doorsnede-constructie voor reguliere talen (simuleer beide automaten tegelijk) veralgemenen naar Turing Machines.

  Voor de twee blauwe vakjes op de lijn van **RegInt** kon je hetzelfde doen: de constructie voor reguliere intersectie vereist het simuleren van een automaat voor een reguliere taal tegelijkertijd met de bestaande automaat. Het is niet moeilijk in te zien dat dit gedaan kan worden "in parallel" met een deterministsiche PDA of met een TM.

- Voor de oranje vakjes kon je het verband tussen unie en doorsnede gebruiken $\overline{X \cup Y} = \bar{X} \cap \bar{Y}$. Aangezien dCF niet gesloten is onder unies (voorbeeld gezien in de les) maar wel onder complement (blauw vakje) kan het neit gesloten zijn onder intersectie. Gelijkaardig: CS is gesloten onder unie (unies nemen van de grammaticas), en (omwille van vakje ZZZZZZZ) ook onder complement, dus is het ook gesloten onder doorsnede.

- Voor de paarse vakjes: geen enkele van al de klasses talen is gesloten onder deelverzamelingen. Immers de volledige taal Σ^* is een taal van elke klasse. Moest een klasse gesloten zijn onder het nemen van deelverzamelingen, zou **elke taal** in die klasse zitten.

In het tweede deel zaten een aantal subtiliteiten in de verwoording.

1. Voor elke ambigue context-vrije grammatica bestaat er een niet-ambigue context-vrije grammatica die dezelfde taal definiëert **Dit is fout. We hebben gezien dat er inherent ambigue context-vrije talen bestaan**.

2. Voor elke reguliere taal L_r bestaat er een minimale NFA (een NFA M met $L(M) = L_r$ waarvoor er geen NFA met minder toestanden L ook accepteert **Dit is uiteraard waar. Elke NFA heeft een aantal states. Een van de NFAs moet "het kleinst" zijn. We hebben geen constructie gezien om die te vinden. Maar hij bestaat uiteraard.**

3. Elke links-lineaire grammatica definiëert een context-vrije taal **Dit is waar: elke dergelijke grammatica definieert een reguliere taal en elke reguliere taal is ook context-vrij.**

4. Er bestaan geen reguliere talen over een leeg alfabet. **Dit is fout. Er bestaat 1 reguliere taal over een leeg alfabet, namelijk de lege taal.**

5. Het is voor geen enkele Turing machine M mogelijk om te beslissen of M eindigt met de lege string als input of niet **Deze bewering is fout. Het is voor heel wat machines mogelijk om te beslissen of ze zullen stoppen of niet. Bijvoorbeeld voor een machine die geen transities heeft, kan je heel gemakkelijk beslissen dat ze stopt.**

6. Voor elke context-vrije taal bestaat er een context-vrije grammatica die zowel in Chomsky normaalvorm als in Greibach normaalvorm is **Dit is fout. Greibach en Chomsky normaalvorm zijn erg verschillend en het is niet gemakkelijk om grammaticas te**

construeren die in beide normaalvormen tegelijk zijn. De enige producties die dan teogelaten zouden zijn, zijn van de vorm $S \to a$

7. Voor elke standaard Turing machine M en elke string w bestaat er een machine H die altijd stopt zodat H in een finale toestand stopt als en slechts als M stopt op w. **Dit is waar. De subtiliteit hier is de volgorde van de kwantificatie. Voor elke M en elke w bestaat er zo'n machine H (zelfs een machine H zonder transities!). Het enige dat die machine H moet doen is elke input rejecten of accepten (afhankelijk van of M stopt op w of niet). De machine H bestaat dus, maar we kunnen niet in het algemeen beslissen welke machine dit is.**

8. De pompstelling voor context-vrije talen geldt ook voor reguliere talen. **Dit is uiteraard waar aangezien elke reguliere taal ook context-vrij is.**

9. Als L een recursieve taal is, en M een standaard Turing machine die L accepteert, dan stopt M altijd. **Dit is niet waar: als L een recursieve taal is dan bestaat er een machine die L accepteert en altijd stopt, maar dat betekent dit niet per se dat alle machines die die taal accepteren altijd stoppen. In het bijzonder: het is niet moeilijk om een machine die L beslist om te zetten naar een machine die L accepteert maar die niet altijd stopt.** Deze vraag werd zeer vaak fout beantwoord.

10. Context-vrije talen zijn altijd oneindig **Dit is uiteraard fout. Bijvoorbeeld $\{a\}$ is context-vrij, en behoorlijk eindig.**

Deze vraag was een zeer goede indicator voor score op het volledige examen. Er zijn slechts drie mensen die slagen op deze vraag en niet slagen op het examen en ook slechts drie mensen die niet slagen voor deze vraag maar in totaal wel geslaagd zijn.

Vraag 5

Beschouw de volgende twee beweringen. Zijn ze juist of niet? Bewijs of geef een tegenvoorbeeld (bewijs de correcte bewering(en) formeel en geef een tegenvoorbeeld voor de foutieve bewering(en)).

Bewering 1: Zij L een context-vrije taal (al dan niet eindig). Dan bestaat er een natuurlijk getal k zodanig dat voor elke string $w \in L$ met $|w| \geqslant k$ er een $j \in \mathbb{N}$ bestaat met $0 < j < k$ zodat voor elk natuurlijk getal i, L ook een string bevat met lengte $|w| + j \cdot i$.

Bewering 2: Zij L een context-vrije taal (al dan niet eindig). Dan bestaan er een natuurlijk getal k en een natuurlijk getal j met $0 < j < k$ zodanig dat voor elke string $w \in L$ met $|w| \geqslant k$ en elk natuurlijk getal i, L ook een string bevat met lengte $|w| + j \cdot i$.

Bespreking

Beide beweringen waren waar. Enkele studenten zijn de mist ingegaan door te focussen op eindige talen. Hun argument was dikwijls van de vorm. Neem $L = \{a\}$. Neem de string $w = a$. Het is duidelijk dat als $j > 0$ en $i > 0$ er geen string in L zit met lengte $|w| + j \cdot i$. Dit argument is echter niet juist. Immers, voor eindige talen kunnen we k groter nemen dan de langste string in de taal. In dat geval zeggen de beweringen niets.

Bewering 1 was het gemakkelijkst aan te tonen. De pompstelling geeft ons een getal m zodanig dat we elke string w kunnen opsplitsen in $w = uvxyz$ zodat elke uv^ixy^iz opnieuw in L zit. Je kan dan $k = m + 1$ en $j = |vy|$ kiezen. De stelling is dan voldaan want de pompstelling garandeert ook dat $|vy| \leqslant m$.

Bewering 2 was moeilijker en telde voor minder punten mee in het uiteindelijke resultaat. De moeilijkheid hier is dat j plots onafhankelijk van de string gekozen moest worden. Dit wordt niet gegarandeerd door de pompstelling. Maar... eens bewering 1 was opgelost, kon je inzien dat bewering 2 ook waar is als volgt: neem $k' = k! + 1$ en $j' = k!$ waarbij k het getal gevonden uit bewering 1 is. Uit bewering 1 weten we dat we elke string langer dan k (en dus zeker elke string langer dan k') kunnen opblazen met elk veelvoud van *een getal* kleiner dan k maar we weten niet wat dat getal precies is. Omdat we j' gelijk genomen hebben aan $k!$ weten we zeker dat j' een veelvoud zal zijn van dat getal.