3 Talen en Berekenbaarheid

3.1 Inleiding

In hoofdstuk 3, Talen en Berekenbaarheid, zal er dieper worden ingegaan op Turingmachines en de werking ervan. Na het instuderen van dit hoofdstuk is het best om onderstaande vragen op te lossen. Het zijn niet zo veel vragen, maar ze bevatten steeds een redelijk groot deel van de leerstof (vaak ook verschillende stukken gecombineerd).

Achteraan dit document zal ik ook verwijzen naar een implementatie van ene Turingmachine in Haskell. Dit kan een dieper inzicht geven op hoe zo een machine werkt.

3.2 Vragen

Vraag 1. Bewijs dat A_{TM} niet beslisbaar is en steun daarbij niet op de stelling van *Rice*. Zou het helpen als het toegelaten was op de stelling van *Rice* te steunen? Is A_{TM} herkenbaar? Co-herkenbaar?

A_{TM} is niet beslisbaar

Bewijs. Controleren of A_{TM} beslisbaar is, is beter bekend als het acceptatieprobleem voor Turingmachines. De geassocieerde taal ziet er als volgt uit.

$$A_{TM} = \{< M, s > \mid M \textit{ is een Turingmachine en } s \in L_M\}$$

Stel er bestaat een beslisser B voor A_{TM}. Dat betekent dat bij input $< M, s >$ B accepteert als M bij input s stopt in zijn q_a en verwerpt als M bij input s stopt in zijn q_r of blijft lopen.

$$B(< M, s >) \textit{ is accept als } M \textit{ } s \textit{ accepteert en anders reject}$$

Construeer nu de contradictie machine C met de eigenschap om telkens het tegenovergestelde te accepteren (of te rejecten) van B.

$$C(< M >) = \textit{opposite}(B(< M, M >)) \textit{ voor elke Turingmachine } M$$

Daarbij is $\textit{opposite}(\textit{accept}) = \textit{reject}$ en $\textit{opposite}(\textit{reject}) = \textit{accept}$. Neem nu voor M hierboven C zelf, dan krijgen we de volgende bewering.

$$C(< C >) = opposite(B(< C, C >))$$

Als $C(< C >) = accept$, dan is $B(< C, C >) = accept$, dan is $opposite(B(< C, C >)) = reject$, dan is $C(< C >) = reject$, dan is $B(< C, C >) = reject$, dan is $opposite(B(< C, C >)) = accept$, dan is $C(< C >) = accept$ ...
Dus C kan niet bestaan. Indien B bestaat kan C wel bestaan, dus B bestaat ook niet! A_{TM} is dus niet beslisbaar. □

A_{TM} is herkenbaar

Bewijs. De herkenner A voor A_{TM} laat, met input $< M, s >$, M lopen op s. Indien deze M s accepteerd, dan accepteerd A zijn input. Indien deze de input $reject$, of gewoon niet stopt, dan zal A deze ook niet accepteren. A_{TM} is dus herkenbaar (ook hier zien we dat A_{TM} niet beslisbaar is omdat deze kan blijven lopen). □

A_{TM} is niet co-herkenbaar

Bewijs. A_{TM} kan echter niet co-herkenbaar zijn. We bewijzen dit met contradictie. Indien A_{TM} co-herkenbaar is, is deze dus herkenbaar en co-herkenbaar (zie vorig bewijs). Wanneer een taal deze beide eigenschappen bezit, is deze beslisbaar. Dit is een contradictie met het eerste bewijs. □

De stelling van Rice

Definitie 1 (Stelling van Rice)**.** *Voor elke niet-triviale, taal-invariante eigenschap P van Turingmachines geldt dat Pos_P (en ook Neg_P) niet beslisbaar is.*

Met deze stelling zouden we het hele bewijs kunnen inkorten. Door een niet-triviale, taal-invariante eigenschap P te vinden van alle Turingmachines die A_{TM} herkennen, hebben we meteen aangetoond dat de taal in kwestie, A_{TM} niet beslisbaar is.[1]

[1] Voor meer informatie over het bewijs, zie het laatste hoofdstuk.

Vraag 2. Bespreek de twee noties ($A \leq_m B$ en $A \leq_T B$) van reduceerbaarheid, hun verband en op welke manier die noties kunnen gebruikt worden om aan te tonen dat een taal (on)beslisbaar/herkenbaar is.

Veel-één reductie ($\leq_m$)

Om over te gaan naar de definitie van de reductie van talen, kunnen we best eerst de definifie van Turing-berekenbaar erbij halen (indien we dit niet doen, kunnen we zeker zijn van deze bijvraag).

Definitie 2 (Turing-berekenbare functie)**.** *Een functie f heet Turing berekenbaar indien er een Turingmachine bestaat die bij input s uiteindelijk stopt met $f(s)$ op de band.*

Definitie 3 (Reductie van talen)**.** *We zeggen dat taal L_1 (over Σ_1) naar taal L_2 (over Σ_2) kan gereduceerd worden indien er een afbeelding f met signatuur $\Sigma_1^* \longrightarrow \Sigma_2^*$ bestaat zodanig dat $f(L_1) \subseteq L_2$ en $f(\overline{L_1}) \subseteq \overline{L_2}$, en zodanig dat f Turing-berekenbaar is. We noteren dat door $L_1 \leq_m L_2$.*

Tot hiertoe is het al duidelijk wat $L_1 \leq_m L_2$ wil zeggen. Het is nu nog belangrijk om het verband met herkenbaarheid en beslisbaarheid aan te tonen.

Definitie 4. *Als $L_1 \leq_m L_2$ en L_2 is beslisbaar, dan is L_1 beslisbaar.*

Bewijs. Het is belangrijk te weten dat de functie f die elementen uit L_1 omzet naar element uit L_2 Turing-berekenbaar is. Concreet wil dit zeggen dat we de mogelijkheid hebben om een turingmachine op te stellen met als in put $s_1 \in L_1$ en als output $f(s_1) \in L_2$.
Neem nu dat L_2 beslisbaar is, met zijn beslisser B. We construeren nu een machine C die elementen uit L_1 omzet (via f) naar elementen uit L_2, waarna we de beslisser B laten beslissen. Hupsa, de combinatie van C en B is de beslisser van L_1 en ook deze taal is dus ook beslisbaar. □

Definitie 5. *Als $L_1 \leq_m L_2$ en L_2 is herkenbaar, dan is L_1 herkenbaar.*

Bewijs. Dit bewijs werkt hetzelfde als het voorgaande, om na te gaan dat wanneer L_2 beslisbaar is, dat dan ook L_1 beslisbaar is. Hier moeten we enkel de beslisser B vervangen door een herkenner H. □

Definitie 6. *Als $L_1 \leq_m L_2$ en L_1 is niet-herkenbaar, dan is L_2 niet-herkenbaar.*

Bewijs. Stel L_1 is niet-herkenbaar en L_2 wel. We hebben zonet bewezen dat als L_2 herkenbaar is, ook L_1 herkenbaar moet zijn. Contradictie. □

Definitie 7. *Als $L_1 \leq_m L_2$ en L_1 is niet-beslisbaar, dan is L_2 niet-beslisbaar.*

Bewijs. Stel L_1 is niet-beslisbaar en L_2 wel. We hebben zonet bewezen dat als L_2 beslisbaar is, ook L_1 beslisbaar moet zijn. Contradictie. □

Orakels en hiërarchie van beslisbaarheid ($\leq_T$)

De tweede notatie heeft betrekking tot orakelmachines in plaats van Turingmachines. Een orakelmachine heeft een andere structuur en werking die deze in staat stelt om, onder andere, A_{TM} te beslissen. Een orakelmachine heeft bezit eigenlijk een grote map van booleans, met elke boolean behorend tot een string. Elke mogelijke string is gekoppeld aan deze 0 of 1 waarde. Een orakel dat een taal beslist zet alle strings die tot die taal behoren op 1, alle andere op 0. Door een inputstring s te encoderen naar de locatie van de corresponderende booleaanse waarde, kan nagegaan worden of de string tot de taal behoort of niet. Het kan dus nooit in een oneindige lus terecht komen!

Het nadeel is echter dat dit enkel een theoretische voorstelling is, die enkel conceptueel gebruikt kan worden. Het is onmogelijk om een bitmap met booleans te implementeren voor elke bestaande string[2].

[2] Want dat zijn er te veel.

Definitie 8 (Turingreduceerbaar)**.** *Een taal A is Turingreduceerbaar naar taal B, indien A beslisbaar is relatief t.o.v. B, t.t.z. er bestaat een orakelachine O^B die A beslist. De notatie is $A \leq_T B$.*

Dit is inderdaad zeer gelijkend op het eerste deel van deze vraag. In plaats van een beslisser voor B te hebben, die A ook beslist, gebruiken we nu een orakel. Dit orakel is dan (zoals eerder vermeld) een theoretisch hulpmiddel dat we kunnen gebruiken om onze kennis toe te passen op meerdere talen. Deze kunnen we echter in realiteit niet implementeren zoals we net beschreven hebben.

Definitie 9. *Indien $A \leq_T B$ en B is beslisbaar, dan is A beslisbaar.*

Bewijs. De definitie zegt ons dat $A \leq_T B$ enkel geldt indien we een orakel O^B hebben dat B én A beslist. Dit is dus volledig afleidbaar van de definitie.
Of anders: stel dat B beslisbaar is en A niet. Dan hebben we een orakel O^B dat (theoretisch) B beslist, maar niet A (want deze is niet beslisbaar). Dit is meteen een contradictie met de definitie. □

Definitie 10. *Indien $A \leq_m B$ dan is ook $A \leq_T B$.*
M.a.w. $\leq_m$ is fijner dan $\leq_T$.

Dit is vanzelfsprekend indien we beseffen dat het orakel (nogmaals) een theoretische uitbreiding is op de Turingmachine. We

gebruiken de Turingmachines om talen te herkennen of the beslissen. Het is mogelijk zo een machine te implementeren in een taal naar keuze. Er is echter een grens op het aantal talen dat we kunnen beslissen, aangezien een aantal in een oneindige lus kunnen komen tijdens het beslissingsproces. Dit is in de praktijk een probleem. Een theoretische oplossing daarvoor is het orakel. We kunnen deze machine wel gebruiken om theoretisch verder te redeneren. Dit will zeggen dat het orakel alle talen beslist dit een Turingmachine kan besliseen, plus de talen die een turingmachine niet kan beslissen (oneindig lus). Hierdoor is $A \leq_m B$ fijner dan $A \leq_T B$.

Vraag 3. Definieer de enumeratormachine. Bewijs dat elke herkenbare taal kan geënumereerd worden en dat elke taal die door een enumerator wordt geënumereerd ook herkenbaar is. Kan elke beslisbare taal geënumereerd worden? Bespreek in deze context de uitspraak "de verzameling van Turing machines is een herkenbare taal".

De enumeratormachine is de machine zoals origineel voorgesteld door Alan Turing in 1936. Deze machine bezit deels de Turingmachine zoals we deze al kennen met enkele kleine aanpassingen. Zo heeft de machine ook een outputband, outputmarker en een enumeratortoestand q_e. De δ van de enumerator heeft als signatuur

$$Q \times \Gamma \to Q \times \Gamma \times \Gamma_e \times \{L, R, S\}$$

Met Q de huidige toestang en Γ het te lezen symbool. Als output een nieuwe toestand Q en Γ, samen met $\{L, R, S\}$. Het verschil met de Turingmachine die we eerder hebben gezien is Γ_e. Dit symbool zal bij overgang geschreven wordt op de outputband. Na het schrijven van Γ_e verplaatst de outputmarker zich 1 plaats naar rechts. De output kan dan een string zijn, verschillende strings opgesplitst door een scheidingsteken of zelfs een oneindige string... Eender hoe, het heeft zin om te spreken over de verzameling (eindige) outputstrings door de enumerator geproduceerd of geënumereerd. Die verzameling is de taal door de enumerator bepaald of geënumereerd. De enumerator mag daarbij dezelfde string meer dan eens op de outputband zetten.

Definitie 11. *De taal door een enumerator bepaald is herkenbaar en elke herkenbare taal wordt door een enumerator geënumereerd. Beide stellingen zullen in aparte bewijzen worden bewezen.*

Bewijs. **Elke taal door een enumerator bepaald is herkenbaar.** Neem de Turingmachine T voor de taal L, bepaald door een gegeven enumerator E. Laat T nu E als een subroutine gebruiken.
T neemt een string s aan en geeft deze door naar de onderliggende E[3]. Telkens E in q_e komt zal T de laatst geproduceerde outputstring lezen op de outputband van E. Indien deze gelijk is aan de inputstring s, dan accepteert T s. Indien dit niet zo is, gaat de enumerator doorrrekenen[4]. □

Bewijs. **Elke herkenbare taal wordt geënumereerd door een enumerator.** Neem de Turingmachine T die de willekeurige taal L herkent. Het is voldoende een enumerator E te construeren die L enumereerd[5]. Om E op te stellen, maken we gebruik van enkele hulpmachines.

1. Een machine die voor een gegeven getal n de eerste n strings uit Σ^* genereerd[6]. Deze T_{gen} zal de strings op de band van E plaatsen.

[3] Door deze op de band te plaatsen en E te starten.
[4] De enumerator kan dus oneindig blijven doorrekenen waardoor de taal door een enumerator bepaald herkenbaar en niet beslisbaar is.
[5] Indien dit lukt is dit mogelijk voor elke L, aangezien L willekeurig is.
[6] Namelijk $s_1, s_2, \ldots, s_n$.

2. T_n zal op elk van de gegeven n strings, n stappen van T uitvoeren. Het zal dus het aantal stappen van T limiteren zodat deze niet in een oneindige lus kan gaan voor de strings die niet worden herkend. Waarom doen we dit? We willen eigenlijk alle strings overlopen[7] en daar de aanvaardbare strings ($s \in L$) uit filteren. Deze behoren tot de herkenbare taal L. Hierdoor moeten we ook de strings nakijken die niet tot L behoren. Deze kunnen er echter voor zorgen dat de T in een oneindige lus gaat. Daarom limiteren we het aantal stappen zodat ook E niet oneindig moet blijven wachten. Indien T de string aanvaard, plaatst E de string op de outputband. Indien T deze niet aanvaard of niet geëindigd is in $\leq n$ stappen, dan zal E de string niet schrijven.

3. T_{driver} zal een opeenvolging van getallen n genereren om de voorgaande machines op te stellen en T_{gen} en T_n oproepen.

Resultaat: Alle $s \in L$ zullen op de outputband worden gezet, maar toch zal de uitvoering van de oneindige lus gestopt worden zodat de enumerator zeker eindigt. Hiermee is het halting probleem voorkomen. Well done. □

Ten slotte moeten we vermelden dat elke taal die beslisbaar is kan geënuereerd worden, aangezien elke beslisbare taal ook herkenbaar is. Uit het laatste bewijs zou ook duidelijk moeten zijn

[7] Die mogelijk zijn op het alfabet Σ.

dat de verzameling van Turinmachines, A_{TM}, een herkenbare taal is. We voeren het bewijs gewoon opnieuw uit, maar met $L = A_{TM}$.

Vraag 4. Geef het bewijs van de stelling: E_{TM} is niet beslisbaar. Doe dit zonder de stelling van Rice te gebruiken. Bespreek de uitspraken E_{TM} is herkenbaar en E_{TM} is coherkenbaar. Zijn er ook alternatieve bewijzen? Hoe zit het met E_{CFG}?

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper

vestibulum turpis. Pellentesque cursus luctus mauris.

Vraag 5. Leg de stelling van Rice uit, en geef het bewijs. Geef minstens één eigenschap van Turingmachines die niet voldoet aan de voorwaarde voor de stelling van Rice en toon aan dat die eigenschap geen aanleiding geeft tot een niet-beslisbare taal. Karakteriseer volledig alle eigenschappen van Turingmachines die aan de stelling van Rice voldoen m.b.v. $IsIn_{TM,S}$.

De stelling van Rice

Vooraleer we verder gaan met de stelling van Rice is het verstandig om sommige termen te verklaren. Niet-triviale en taal-invariante eigenschappen zijn een belangrijk onderdeel van de stelling. In onderstaande verklaringen duidt Pos_P op de verzameling van machines die in het bezit zijn van een eigenschap P en Neg_P de verzameling van machines zonder eigenschap P.

Definitie 12 (Niet-triviale eigenschap)**.** *Een eigenschap P van Turingmachines heet niet-triviaal indien $Pos_p \neq \emptyset$ en ook $Neg_p \neq \emptyset$. Er bestaan dus Turingmachines die deze eigenschap P bezitten, maar ook machines die deze niet bezitten.*

Definitie 13 (Taal-invariante eigenschap)**.** *De eigenschap P heet taal-invariant indien alle machines die dezelfde taal bepalen hebben ofwel allemaal P, ofwel heeft geen enkele ervan*

P.

$$L_{M_1} = L_{M_2} \Rightarrow P(M_1) = P(M_2)$$

Met deze twee definities in het achterhoofd, kunnen we overgaan naar de formele definitie van de stelling van Rice, met het bewijs als gevolg.

Definitie 14 (Stelling van Rice)**.** *Voor elke niet-triviale, taal-invariante eigenschap P van Turingmachines geldt dat Pos_P (en ook Neg_P) niet beslisbaar is.*

Bewijs. Neem de Turingmachine $M_\emptyset$ de lege taal beslist. Laten we er nu van uit gaan dat deze machine een bepaalde eigenschap P heeft. De stelling zegt ons dat de eigenschap P niet-triviaal is. Uit de definitie kunnen we dan afleiden dat $Pos_P \neq \emptyset$ (en ook $Neg_P \neq \emptyset$). Aangezien deze verzameling niet leeg is, moet er een Turingmachine X bestaan met deze eigenschap P. Laat ons zeggen dat deze Turingmachine de taal L_X beslist.

We gaan nu proberen een contradictie te bekomen door aan te nemen dat de stelling niet waar is. We nemen dus aan dat Pos_P (en dus ook Neg_P) beslisbaar is. We gaan nu een beslisser B proberen op te stellen voor Pos_P die deze beslist[8]. Om B te maken, gaan we eerst een hulpmachine $H_{M,s}$ opstellen.

Deze hulpmachine $H_{M,s}$ heeft een Turingmachine M en een string s in zich. Deze staan vast voor de machine en kunnen

[8] Later zullen we deze B gebruiken om een beslisser A te maken voor de taal met Turingmachines A_{TM}

dus niet veranderen[9]. De input van deze machine is een string $x \in L_X$. Wanneer $H_{M,s}$ gestart wordt, zal deze eerst M laten lopen over s. Indien M s reject, zal $H_{M,x}$ altijd rejecten. Indien M s accept, dan zal $H_{M,s}$ overgaan naar fase 2. Hier zal de hulpmachine X over x laten lopen. Indien X x ook accept, dan zal de hulpmachine accepten. Indien X x reject, dan zal ook de hulpmachine rejecten.
Er zijn nu twee mogelijkheden voor $H_{M,s}$. Indien M s accept, dan gaat $H_{M,s}$ altijd overgaan tot het testen van x in X. In dit geval beslist $H_{M,s}$ de volledige taal L_X. De andere optie is dat M s reject. In dat geval gaat de hulpmachine altijd rejecten en dus enkel de lege taal accepteren.
Laat nu de beslisser B los op $H_{M,s}$. Dit wil zeggen dat de beslisser accept of reject voor de gegeven M en s.
Stel dat we nu een beslisser A maken voor A_{TM}. In dat geval moeten we dus elke M en s in A_{TM} testen. We kunnen dus zeggen dat A accept indien B $H_{M,s}$ accept, anders reject. We bekomen dus de volgende conclusie.

$$
\begin{array}{c}
A \text{ accepts } < M, s > \\
\Updownarrow \\
B \text{ accepts } H_{M,s} \\
\Updownarrow \\
H_{M,s} \text{ heeft eigenschap } P \\
\Updownarrow \\
H_{M,s} \text{ accepts } L_X \\
\Updownarrow \\
M \text{ accepts } s
\end{array}
$$

[9] Ze zijn als het ware gehardcoded.

Conclusie: A is een beslisser voor A_{TM}, maar dit is onmogelijk aangezien A_{TM} niet beslisbaar is[10]. Hieruit kunnen we concluderen dat alle bovenstaand equivalenties onwaar zijn en dus is ook Pos_P niet beslisbaar. **Contradictie**. □

[10] Zie vraag 1 van dit hoofdstuk voor het bewijs.

Vraag 6. Wat is een orakelmachine? Bespreek de uitspraak "de verzameling orakelmachines (over een gegeven orakel) is strikt krachtiger dan de verzameling van Turing machines". Leg hierbij ook uit wat men bedoeld met "krachtiger". Kan een verzameling orakelmachines (voor bepaald gegeven orakel) alle talen beslissen? Kan een orakelmachine ook A_{TM} of H_{TM} beslissen?

Orakelmachine

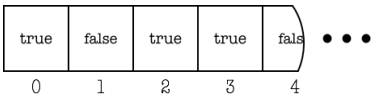
Zoals we weten is een Turingmachine een handig hulpmiddel om te bepalen of strings tot een taal horen of niet. We kunnen de machines gebruiken om talen te herkennen, of specifieker, te beslissen. Niet alle talen kunnen echter beslist worden door zo een Turingmachine. Zo is het bijvoorbeeld onmogelijk om een Turingmachine op te stellen die de taal A_{TM} beslist.

Dit zorgt er voor dat we op zoek moeten gaan naar een betere machine die naast het beslissen van de al besliste talen, ook andere talen kan beslissen. De nieuwe machine moet dus krachtiger zijn. Het resultaat is een orakelmachine.

Een orakelmachine is een uitbreiding op de Turingmachine, die een orakel bevat. Je kan een orakel bekijken als een black box waar de Turingmachine vragen een kan stellen. In theorie is een orakel eigenlijk een soort bitmap, of anders gezegd een rij van booleans. Stel we ordenen alle strings volgens de lexicografische orde met kortere strings eerst. Elke string op index i komt nu

overeen met een booleaanse waarde in de bitmap, die ook gealloceerd is op locatie i. Indien de string op een bepaalde locatie tot een gegeven taal L behoord, dan zal de overeenkomstige booleaanse waarde op $true$ staan. Indien dit niet het geval is, blijft de waarde op $false$.

De werking van een orakelmachine is nu heel eenvoudig. Het krijgt als input een string s. De machine vraagt nu aan het orakel of de string tot een taal behoort. Het orakel is in staat om de string om te vormen naar de index in de rij volgens de lexicografische volgorde en raadpleegt de overeenkomstige booleaanse waarde in de bitmap. Is die waarde $true$, dan accepteert het orakel de string s. Indien de waarde $false$ is, dan wordt de string s geweigert. Een orakelmachine waarvan de bitmap een configuratie heeft voor een bepaalde taal L te beslissen, noemen we O^L.



Figuur 1: Simpele voorstelling van een orakel.

Een orakel kan voor vele problemen gebruikt worden. Het halting probleem is hier echter maar één enkel voorbeeld van. Vaak wordt een orakel gebruikt om op een abstracte manier een antwoord te krijgen op een bepaalde vraag. Deze vraag kan zelfs onoplosbaar zijn.

Krachtiger dan een Turingmachine

Zoals zonet vermeld, is een orakelmachine krachtiger dan een Turingmachine. Dit wil zeggen dat een orakelmachine sowiso meer talen kan beslissen dan een Turingmachine. Het kan namelijk alle talen beslissen die een Turingmachine kent vermeerderd met heel wat talen die anders in een oneindige lus zullen komen. A_{TM} is daar een voorbeeld van. Een orakelmachine is dus strikt krachtiger dan een Turingmachine.

Verzameling orakelmachines

Normaal wordt een orakelmachine gegeven en kunnen we deze vraag concreet beantwoorden. Dit is nu dus onmogelijk. De redenering gaat echter van de volgende vorm zijn.

Een orakel O^A is gegeven dat een taal A beslist. Volgens de definitie van Turingreduceerbaar (zie onderaan) kunnen we concluderen dat elke B, waarvoor geldt dat $B \leq_T A$, kan beslist worden door de orakelmachine O^B. Het is dus niet mogelijk om alle talen te beslissen, enkel die die Turingreduceerbaar zijn.

> **Definitie 15** (Turingreduceerbaar)**.** *Een taal A is Turingreduceerbaar naar taal B, indien A beslisbaar is relatief t.o.v. B, t.t.z. er bestaat een orakelachine O^B die A beslist. De notatie is $A \leq_T B$.*

Het is misschien wel mogelijk om een orakel te ontwerpen dat dit wel kan. In plaats van een bitmap voor alle strings, maken we een bitmap voor alle talen. Een element bevat hier geen

booleaanse waarde, maar een ander orakel voor de overeenkomstige taal. Theoretisch is dit volgens mij mogelijk, maar gaat misschien iets te ver voor op het examen.

Beslisbaarheid A_{TM} en H_{TM}

Dit is een bijvraag en hier gaan we dus niet dieper op in. De bewijzen van de onbeslisbaarheid worden hier dus niet gegeven. Deze staan ook al in andere vragen. Volgens mij kan hier heel beknopt op geantwoord worden. A_{TM} en H_{TM} zijn niet beslisbaar met standaard Turingmachines. Met een orakelmachine kunnen we uiteraard een corresponderende bitmap maken voor de talen en zijn ze dus wel beslisbaar.

Vraag 7. Geef informeel de definitie van een lineair begrensde automaat (LBA). Argumenteer dat het aanvaardingsprobleem voor LBA's (A_{LBA}) beslisbaar is. Geef de stappen in een bewijs dat E_{LBA} (de verzameling van LBA's die de lege taal bepalen) niet beslisbaar is.

Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Donec odio elit, dictum in, hendrerit sit amet, egestas sed, leo. Praesent feugiat sapien aliquet odio. Integer vitae justo. Aliquam vestibulum fringilla lorem. Sed neque lectus, consectetuer at, consectetuer sed, eleifend ac, lectus. Nulla facilisi. Pellentesque eget lectus. Proin eu metus. Sed porttitor. In hac habitasse platea dictumst. Suspendisse eu lectus. Ut mi mi, lacinia sit amet, placerat et, mollis vitae, dui. Sed ante tellus, tristique ut, iaculis eu, malesuada ac, dui. Mauris nibh leo, facilisis non, adipiscing quis, ultrices a, dui.