Logische Complexiteit
Uitwerkingen Eindtoets

dinsdag 10 april 2012
13:30 – 16:30

Hieronder uitwerkingen van de eindtoets op dinsdag 12 april 2012. 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 — Verschil

Beschouw de taal V_{DFA} als hieronder gedefinieerd. Bewijs dat deze taal beslisbaar is.

$$\mathrm{V}_{\text{DFA}} := \left\{ \langle M, N \rangle \;\middle|\; \begin{array}{l} M \text{ en } N \text{ zijn DFA's over hetzelfde alfabet en} \\ \text{het verschil } \mathcal{L}(M) - \mathcal{L}(N) \text{ is eindig groot} \end{array} \right\}$$

Lemma 1. *De taal* V_{DFA} *is beslisbaar.*

Bewijs. We construeren een beslisser V als volgt:

$V :=$ "Op invoer $\langle M, N \rangle$ met M en N DFA's:
1. Construeer de DFA $\overline{N}$ voor het complement van de taal van N.
2. Construeer de DFA A voor de doorsnede tussen M en $\overline{N}$.
3. Test of A een oneindige taal heeft.
4. Als (3) succesvol, verwerp. Anders, accepteer.

Merk op dat de taal der codes van DFA's met oneindige taal beslisbaar is, als behandeld op het werkcollege. De machine V accepteert een paar $\langle M, N \rangle$ enkel en alleen indien $\mathcal{L}(M) - \mathcal{L}(N)$ eindig is. Alle stappen zijn eindig, en daarmee is V een beslisser. □

Opgave 2 — Onherkenbaar Verschil

De hieronder gedefinieerde taal V_{TM} lijkt wat op die in Opgave 1, maar nu met TM's in plaats van DFA's. Dit geeft een heel andere taal. Bewijs dat de taal V_{TM} noch herkenbaar, noch beslisbaar is.

$$\mathrm{V}_{\text{TM}} := \left\{ \langle M, N \rangle \;\middle|\; \begin{array}{l} M \text{ en } N \text{ zijn TM's over hetzelfde alfabet en} \\ \text{het verschil } \mathcal{L}(M) - \mathcal{L}(N) \text{ is eindig groot} \end{array} \right\}$$

Ik vond het mooi om eerst het volgende lemma te bewijzen. Je kan natuurlijk direct een reductie uit het (onherkenbare) complement van A_{TM} naar V_{TM} geven. Hier koos ik ervoor de reducties uit elkaar te pluizen, zodat de stappen beter zichtbaar zijn.

Lemma 2. *De taal $A := \{ \langle M \rangle \mid$ de taal van M is eindig $\}$ is onherkenbaar.*

Bewijs. We geven een reductie van het complement van A_{TM} naar A. Gegeven een TM M en een woord w construeren we de volgende machine $S_{M,w}$:

$S_{M,w}$:= "Op invoer v:
1. Simuleer M op w.
2. Als M accepteert, accepteer, anders verwerp."

Stel $\langle M, w \rangle \in \mathrm{A}_{\text{TM}}$, dan accepteert $S_{M,w}$ ieder woord, dus heeft ze een oneindige taal, en daarmee $\langle S_{M,w} \rangle \notin A$. Andersom, als $\langle S_{M,w} \rangle \notin \mathrm{A}_{\text{TM}}$, dan accepteert $S_{M,w}$ niks. Daarmee is haar taal leeg, dus $\langle S_{M,w} \rangle \notin A$. Dit geeft de gewenste reductie. Omdat het complement van A_{TM} onherkenbaar is, is A het ook. □

Lemma 3. *De taal V_{TM} is onherkenbaar. Hiermee is ze ook onbeslisbaar.*

Bewijs. De tweede uitspraak volgt uit de eerste. Immers, als er een beslisser was die V_{TM} herkende, was V_{TM} herkenbaar.

We reduceren nu A naar V_{TM}, waar A is als in het vorige lemma. Gegeven $\langle M \rangle$ maken we $\langle M, E \rangle$, met E de machine die niks accepteert. Zie dat $\mathcal{L}(M)$ gelijk is aan $\mathcal{L}(M) - \mathcal{L}(E) = \mathcal{L}(M) - \emptyset$, waaruit het gewenste direct volgt. □

Opgave 3 — Dubbel oneindige tapes

Herinner je dat in onze definitie van een Turingmachine de tape enkel oneindig is naar rechts toe. Een *dubbel oneindige TM* (DTM) heeft een tape die beide kanten op oneindig is. In de startconfiguratie is de gehele tape leeg, behalve het stuk waar de invoer staat, en de leeskop van de DTM staat aan de linkerkant van de invoer.

(a) Bewijs dat een taal $\mathcal{L}$ herkend wordt door een DTM precies als $\mathcal{L}$ herkend wordt door een TM.

Lemma 4. *Zij $\mathcal{L}$ een taal. Nu wordt $\mathcal{L}$ herkend door een DTM precies als $\mathcal{L}$ herkend wordt door een TM.*

Bewijs. Stel er is een DTM M die $\mathcal{L}$ herkend. Nu maken we een twee-tape TM N die hetzelfde doet. Stel je voor dat de tape van M doorgebogen wordt om het beginpunt, en het bovenste deel wordt de eerste tape van N, en het onderste deel de tweede tape. De machine N bestaat uit de volgende onderdelen. Als eerste markeert ze het begin van beide tapes met een $*$. Daarna gaat ze naar het begin van de "onderste" of "rechter" tape, en staat op de andere tape op de markering.

Iedere toestand q van M geeft twee toestanden, $\langle q, \text{links} \rangle$ en $\langle q, \text{rechts} \rangle$ in de machine N. De machine heeft transities tussen $\langle q, \text{rechts} \rangle$ en $\langle r, \text{rechts} \rangle$ als M ze tussen q en r heeft, en verplaatst en leest hier van de "onderste" of "rechter" tape. Hetzelfde voor $\langle q, \text{links} \rangle$ en $\langle r, \text{links} \rangle$, maar dan over de andere tape in de andere richting. Als de machine in $\langle q, \text{rechts} \rangle$ staat en een $*$ leest op de onderste tape, dan gaat ze naar $\langle q, \text{links} \rangle$, en zet een stap naar rechts (gesimuleerd links dus). Een analoog idee voor de "bovenste" of "linker" tape. De begintoestand wordt $\langle q_0, \text{rechts} \rangle$.

De andere kant op is eenvoudiger. Gegeven de TM M maken we een DTM die eerst het begin van de tape markeert, en er direct rechts van gaat staan. Als de machine op de markering komt, dan zet ze een stap naar rechts. Dit simuleert precies het gedrag van M indien ze naar links zou willen aan het begin van de tape. □

Ruimtecomplexiteit hebben we gedefinieerd voor TM's die werken op een enkelzijdig oneindige tape. Voor een DTM definiëren we de ruimtecomplexiteit als:

$$n \in \mathbb{N} \quad \mapsto \quad \max \left\{ \begin{array}{l} \text{maximale afstand van leeskop t.o.v.} \\ \text{startconfiguratie bij berekenen met invoer } w \end{array} \;\middle|\; w \text{ een woord van lengte } n \right\}$$

(b) Beargumenteer dat het hebben van een dubbel of enkel oneindige tape geen verschil maakt wat betreft het hebben van polynomiale ruimtecomplexiteit.

Als M een DTM is met ruimtecomplexiteit f, dan heeft de twee-tape Turing machine uit het bovenstaande bewijs dezelfde complexiteit, dus is er een gewone TM met die complexiteit in het kwadraat. Was f een polynoom, dan is f^2 dit nog steeds. De andere kant op is er slechts een lineair tijdsverschil, dus dit veranderd niks aan de "polynoomheid" van de complexiteit.

Opgave 4 — Maximale Vervulling

Bewijs dat de onderstaande taal MaxSat NP-volledig is. Herinner je dat CNV staat voor conjunctieve normaalvorm.

$$\text{MaxSat} := \left\{ \langle \phi, n \rangle \;\middle|\; \begin{array}{l} \phi \text{ is een formule in CNV waarvoor een bedeling} \\ \text{bestaat die minstens } n \text{ clausules waarmaakt} \end{array} \right\}$$

Lemma 5. *De taal* MaxSat *is* NP*-volledig.*

Bewijs. Dat MaxSat in NP zit is niet erg lastig aan te tonen. We geven een controleur, de getuige hier wordt zo'n verlangde bedeling.

$M :=$ "Op invoer $\langle \langle \phi, n \rangle, c \rangle$ met ϕ een formule in CNVen n een getal:
1. Test of c de code is van een bedeling, zeg $\langle v \rangle = c$.
2. Ontbindt ϕ in clausules $\phi_1 \wedge \ldots \wedge \phi_m$.
3. Tel het aantal i's tussen 1 en m zodat $v(\phi_i)$ waar is.
4. Als $i \geq n$, accepteer. Anders, verwerp.

Merk op dat M een propere polynomiale tijd controleur is. Als $\langle \phi, n \rangle \in$ MaxSat dan is er een bedeling v zodanig dat v minstens n clausules waarmaakt, nu accepteert M de invoer $\langle \langle \phi, n \rangle, \langle v \rangle \rangle$. Andersom, als M invoer $\langle \langle \phi, n \rangle, c \rangle$ accepteert dan moet er een bedeling zijn die minstens n clausules van ϕ waarmaakt. Alle stappen zijn polynomiaal in de lengte van ϕ, dus M werkt in polynomiale tijd.

Voor NP-volledigheid moeten we nog aantonen dat iedere taal reduceert tot MaxSat. Hiertoe tonen we aan dat Sat reduceert tot MaxSat. Immers, als ϕ een formule in CNVis met n clausules, dan zit $\langle \phi \rangle$ in Sat precies als er een waarmakende bedeling is. Nu betekent dit laatste zoveel als dat er een bedeling is die alle n veel clausules in ϕ waarmaakt, equivalent aan $\langle \phi, n \rangle \in$ Sat. □

Opgave 5 — Accepteren

Bewijs dat de taal $\text{A}_{\text{NFA}} := \{ \langle M, w \rangle \mid M \text{ is een NFA die } w \text{ accepteert} \}$ in PSPACE zit.

Lemma 6. *De taal* A_{NFA} *zit in* PSPACE.

Het onderstaande is een van de manieren om het probleem op te lossen. De crux zat in het vinden van een manier om een NFA te simuleren die eindigt. De berekening van een NFA als gedefinieerd in het boek kan namelijk willekeurig lang zijn, dus is er geen stop-criterium voor een machine die simpelweg steeds nondeterministisch kiezen zou. Hieronder wat uitwerking voor wanneer je eerst de ε-pijlen weghaalt.

Bewijs. We geven een nondeterministische TM M die A_{NFA} beslist in polynomiale ruimte. Wegens de Stelling van Savitch is er dus ook een deterministische TM die A_{NFA} beslist in polynomiale ruimte, precies wat we diende te bewijzen.

Als eerste merken we op dat het verwijderen van ε-pijlen in een NFA kan in polynomiale tijd ten opzichte van het aantal toestanden. Houd een tabel bij van toestanden bij toestanden. Als je van q naar r kan met een ε pijl, en van r kan je naar s, dan kan je van q naar s.

$M :=$ "Op invoer $\langle N, w \rangle$ met N een NFA en w een woord over het invoeralfabet van N:
1. Verwijder de ε-pijlen in N.
2. Markeer de accept-toestand van N.
3. Voor $i = 0$ tot de lengte van w:
3a. Kies non-deterministisch een toestand r zodat de gemarkeerde toestand en r verbonden zijn door een w_i-pijl in N.
3b. Haal de markering weg en markeer r.
4. Accepteer als de accept-toestand gemarkeerd is, en verwerp anders.

Het idee is dat M de NFA N simuleert op invoer w. Als de NFA N het woord w accepteert, dan is er een een pad van begintoestand tot eindtoestand. Dus M zal dan $\langle N, w \rangle$ accepteren. Andersom, als M de invoer $\langle N, w \rangle$ accepteert dan is er een berekening van M die een pad uitkiest wat overeenkomt met een accepterende berekening van N. Hiermee heeft N dus een accepterende berekening voor w, waarmee N de invoer w accepteert. In ieder geval zijn er maar zoveel herhalingen tussen (2) en (4) als er letters in w zijn, en alle stappen kosten polynomiaal veel ruimte. Dus de machine is een (nondeterministische) beslisser in polynomiaal veel ruimte. □

Opgave 6 — Anagram

Een *anagram* van een zin w is een zin die te verkrijgen is door de letters uit w in volgorde te verplaatsen, met eventueel een ander aantal spaties. Bewijs dat de onderstaande taal Fijn onbeslisbaar is.

$$\text{Fijn} := \{ \langle M \rangle \mid M \text{ is een TM die precies anagrammen van “fijn tentamen” accepteert} \}$$

Lemma 7. *De taal* Fijn *is onbeslisbaar.*

Bewijs. We passen de stelling van Rice toe. Het is duidelijk dat er sprake is van een eigenschap van talen van Turingmachines. Om niet-trivialiteit aan te tonen, merk op dat de code van de TM die niks accepteert niet in de taal Fijn zit. Bekijk de volgende TM:

$M :=$ “Op invoer w:
1. Verwijder alle spaties uit w;
2. Sorteer alle letters in w;
3. Accepteer indien w gelijk is aan “aeefijmnnntt”, verwerp anders.

Het is duidelijk dat M enkel en alleen anagrammen van “fijn tentamen” accepteren zal. Dus $\langle M \rangle$ zit *wel* in Fijn. Hiermee is Fijn niet-triviaal, en dus wegens de stelling van Rice ook onbeslisbaar. □