Automaten en Berekenbaarheid

Tweede gekwoteerde oefenzitting december 2015: een oplossing

1 Algebra van talen

Hieronder zijn L_1, L_2 en L_3 talen over hetzelfde alfabet Σ. Stel dat L_1 en L_2 verschillende niet-beslisbare talen zijn, en dat L_3 een oneindige beslisbare taal is die verschilt van Σ^*. Zijn volgende talen dan beslisbaar, niet-beslisbaar of hangt dit af van de talen in kwestie ? Bewijs of geef (tegen)voorbeelden.
(a) $L_1 \cap L_3$ (b) $L_1 \cup L_3$ (c) $L_1 \cup L_2$

Antwoord

(a) Neem $L_1 = A_{TM}$ (als voorbeeld van een niet-beslisbare taal) en $L_3 = \Sigma^+$, and is $L_1 \cap L_3$ niet-beslisbaar.

Neem $L_1 = A_{TM}$ over alfabet Σ_1 en $L_3 = \Sigma_3^*$ waarbij $\Sigma_1 \cap \Sigma_3 = \emptyset$. Beide zijn talen over $\Sigma_1 \cup \Sigma_3$, en voldoen aan de voorwaarden. Dan is $L_1 \cap L_3$ leeg en beslisbaar.

(b) Neem bijvoorbeeld $L_1 = A_{TM}$ en $L_3 = \Sigma^+$, dan is de unie ervan ofwel Σ^* ofwel L_3 en dus beslisbaar.

Neem $L_1 = A_{TM}$ over alfabet Σ_1 en $L_3 = \Sigma_3^*$ waarbij $\Sigma_1 \cap \Sigma_3 = \emptyset$. Beide zijn talen over $\Sigma_1 \cup \Sigma_3$, en voldoen aan de voorwaarden. Dan is $L_1 \cup L_3$ niet beslisbaar (werk zelf uit waarom dat zo is).

(c) Neem $L_2 = \overline{L_1}$ met L_1 een willekeurige niet beslisbare taal: de unie is Σ^* en beslisbaar.

Neem $L_2 = L_1 \ \{w\}$ waarbij $w \in L_1$ - er bestaat zeker zulk een w, want als L_1 een willekeurige niet beslisbare taal is, dan is die niet leeg. De unie is nu gelijk aan L_1, dus niet beslisbaar.

2 Accepteren, stoppen, reduceren

Gegeven is een deelverzameling S van de Turing machines en de eigenschap dat het halting probleem voor S beslisbaar is, namelijk $H_S = \{< M, w > | M \in S, M \textit{ stopt op } w\}$ is beslisbaar.
Bewijs m.b.v. een veel-één reductie ($\leq_m$) dat dan ook het acceptatieprobleem voor S beslisbaar is, namelijk dat ook $A_S = \{< M, w > | M \in S, M \textit{ accepteert } w\}$ beslisbaar is.
Detailleer de reductie voldoende, laat zien dat het een geldige reductie is, en dat die ook echt doet wat nodig is.

Antwoord
We construeren een reductie $A_S \leq_m H_S$. Kies eerst een string s die niet in H_S zit (waarom kan dat ?). De afbeelding $f : \Sigma^* \to \Sigma^*$ wordt door het volgende algoritme bepaald:

- als w niet van de vorm $< M, x >$ is met $M \in S$, dan is $f(w) = s$
- als w van de vorm $< M, x >$ is met $M \in S$, dan laat f eerst de beslisser voor H_S lopen op $< M, x >$; bij
  - *accept:* laat M lopen op x (dat stopt - waarom ?); als M x accepteert, definieer dan $f(w) = w$, anders $f(w) = s$
  - *reject:* $f(w) = s$

We zien dat $f(A_S) \subseteq H_S$ en $f(\overline{A_S}) \subseteq \overline{H_S}$, en f is Turing-berekenbaar. Merk op: we hebben de beslisser voor H_S gebruikt tijdens de *uitvoering* van f

3 Beslissen of (co-)herkennen

Van elke taal hieronder moet je aangeven of die taal (I) beslisbaar, (II) herkenbaar en niet beslisbaar, of (III) co-herkenbaar en niet beslisbaar is. Bij die drie gevallen geef je ook (I) een beschrijving van een beslisser, (II) een bewijs van niet-beslisbaarheid en een beschrijving van een herkenner, of (III) een bewijs van niet-beslisbaarheid en een beschrijving van een co-herkenner
Hieronder is M steeds een Turingmachine, Σ is het invoeralfabet van die Turingmachine, en L_M is de taal die bepaald wordt door M.

(a) $\{< M > \mid \exists s_1, s_2 \in L_M : |s_1| \neq |s_2|\}$;

(b) $\{< M, s > \mid \text{M } \textit{stopt na hoogstens } 2^{|s|} \textit{ stappen bij invoer } s\}$;

(c) $\{< M, y > \mid M \textit{ beweegt de lees-/schrijfkop minstens } 1 \textit{ keer naar links bij invoer } y\}$.

Antwoord

(a) We hebben hier een taal-invariante, niet-triviale eigenschap van Turingmachines, dus passen we Rice toe: de taal is niet beslisbaar.

De taal is herkenbaar met de volgende herkenner: die krijgt als invoer een Turingmachine M en doet het volgende

- maak van M een enumerator E voor M_L (zie de cursus)
- pas nu E aan: telkens een outputstring van M_L geschreven wordt, vergelijk die in lengte met alle vorige geschreven string; vind je er ééntje met verschillende lengte, dan accepteer je M, anders doe je voort

Voor M in de taal eindigt dit zeker. De taal is dus niet co-herkenbaar.

(b) Dit is een beslisbare taal: een UTM met teller kan M op s simuleren tot ten hoogste zoveel stappen zijn uitgevoerd als mag op s en op gepaste wijze accepteren of rejecten.

Merk op dat de taal $\{< M > \mid \forall s \in \Sigma^*,\ M \textit{ stopt na hoogstens } 2^{|s|} \textit{ stappen bij invoer } s\}$ niet-beslisbaar is.

(c) De taal is beslisbaar. Een beslisser kan als volgt werken:

- simuleer M op y en als de leeskop naar links beweegt, accept
- als de leeskop *te lang* blijft staan, dan zit je in een lus, dus reject; *te lang* heeft te maken met het aantal combinaties van een toestand en een symbool onder de leeskop; de beslisser rekent dat getal uit en telt tijdens de simulatie; een lus betekent nu reject
- als de leeskop rechts voorbij de invoer is gekomen, dan kunnen er alleen nog maar # onder de leeskop komen bij het naar rechts gaan; dat kan gebeuren in gelijk welke toestand, maar dat is ook een eindig aantal; opnieuw is onthouden in welke toestand naar rechts gegaan wordt en nadien een # gezien wordt voldoende om die oneindige lus te ontdekken: reject; samen met de *stilstaan*-lus is alles nu afgedekt

  Merk op: als een machine nooit naar links beweegt bepaalt een reguliere taal; maar beslissen of een machine een reguliere taal bepaalt, is niet beslisbaar ...

4 Conversies

Met de Church-numerals c_n gedefiniëerd als in de cursus $c_n = \lambda fx.\ f^n(x)$, en de machtsverheffing als $A_{\text{exp}} = \lambda xy.y\ x$, toon aan dat $A_{\text{exp}}\ c_2\ c_2 = c_4$. Geef bij elke conversie aan welke soort je toepast, en op welke redex.
$A_{\text{exp}}\ c_2\ c_2 = \lambda xy.\ y\ x(\lambda fx.\ f\ (f\ x))(\lambda gy.\ g\ (g\ y))$

$\xrightarrow{\beta}$ (2 argumenten voor de A_{exp}) $\lambda \underline{f}x.\ f\ (f\ x))\underline{(\lambda gy.\ g\ (g\ y))}$

$\xrightarrow{\beta} \lambda x.\ (\lambda gy.\ g\ (g\ y))\ ((\lambda \underline{g}y.\ g\ (g\ y))\ \underline{x})$

$\xrightarrow{\beta} \lambda x.\ (\lambda \underline{g}y.\ g\ (g\ y))\ \underline{(\lambda z.\ x\ (x\ z))}$

$\xrightarrow{\beta} \lambda x.\ (\lambda y.\ (\lambda z.\ x\ (x\ z))\ ((\lambda \underline{z}.\ x\ (x\ z))\ \underline{y}))$

$\xrightarrow{\beta} \lambda x.\ (\lambda y.\ (\lambda \underline{z}.\ x\ (x\ z))\ \underline{(x\ (x\ y))}$

$\xrightarrow{\beta} \lambda x.\ \lambda y.\ (x\ (x\ (x\ (x\ y))))$

$\xrightarrow{\alpha} \lambda fx.\ f^4(x)$
Tussendoor zijn een paar αconversies weggelaten.

5 Functies en orakels

We zagen dat een orakel O_L voor de taal L, L kan beslissen. Een beetje analoog: een functieorakel O_f voor een functie f kan voor elke input i de waarde van $f(i)$ op de tape zetten.
Stel we hebben een functieorakel O_S voor de *bezige bever*-functie S. Beschrijf in voldoende detail een Turingmachine M die O_S als subroutine mag gebruiken om A_{TM} te beslissen.

Antwoord:
M_O werkt als volgt op invoer $< M, w >$:

- transformeer M in een machine M_w die gegeven de lege string, eerst w op de band zet en dan M uitvoert op die invoer
- tel het aantal toestanden X van M_w, en bereken (m.b.v. O_S) de bijhorende waarde NX van de bezige-beverfunctie
- simuleer nu M op w voor ten hoogste NX stappen; er zijn drie mogelijkheden
  1. M accepteert w voor de grens NX bereikt is: M_O accepteert dan $< M, w >$
  2. M verwerpt w voor de grens NX bereikt is: M_O verwerpt dan $< M, w >$
  3. de grens NX wordt bereikt; d.w.z. dat M_w in een lus gaat, bijgevolg accepteert M niet w; M_O verwerpt dan $< M, w >$

6 Oneindige unie van talen

Laat $B_i, i \in \mathbb{N}$ een rij beslisbare talen zijn. Vermits de unie van twee beslisbare talen beslisbaar is, kan men gemakkelijk bewijzen dat voor elke eindige $n \in \mathbb{N}$, $\cup_{i=1}^{n} B_i$ beslisbaar is.
Bewijs of geef een tegenvoorbeeld van volgende uitspraak

$\cup_{i>0} B_i$ is beslisbaar

Antwoord:
Elke oneindige taal L is de oneindige unie van talen die elk slechts zijn één van de elementen van L bevatten: elk van die singleton talen is beslisbaar (zelfs regulier). Maar L kan gelijk wat zijn, dus hoeft niet beslisbaar te zijn. L kan gelijk welke taal zijn, ook eindig of zelfs leeg (door $B_i = \emptyset$ te nemen).

7 Vrij of ongebonden

In lambda-calculus zijn de voorkomens van variabelen vrij of gebonden. Omcirkel bij onderstaande expressie de gebonden variabelen en wijs met pijl de overeenkomstige lambda aan, zoals in: 

Ongebonden variabelen duid je aan met een hoedje ($\hat{x}$). Doe het eerst in het klad en schrijf het dan netjes over: kladwerk hieronder wordt niet bekeken.

Antwoord:
I.p.v. met pijlen, indices om aan te geven welke variabelen horen bij welke lambda.

$$+\ \hat{y}\ (\lambda\ x\ .\ +\ \hat{y}\ (\lambda\ y_1\ .\ (\lambda\ x_1\ .\ +\ (+\ y_1\ y_1)\ \ x_1)\ \ y_1))\,(\lambda\ y_2\ .\ +\ \ y_2\ \hat{x})$$