Automaten en Berekenbaarheid

Tweede gekwoteerde oefenzitting december 2016: een oplossing

1 Algebra van talen

Hieronder zijn A en B talen over hetzelfde alfabet Σ. Stel dat A niet-beslisbaar is, en dat B 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) $A \cap B$ (b) $A \cup B$

Antwoord In beide gevallen hangt het af van de talen:

(a) Neem $A = A_{TM}$ (als voorbeeld van een niet-beslisbare taal) en $B = \Sigma^+$, dan is $A \cap B$ niet-beslisbaar. ($A \cap B$ is gelijk aan $A \setminus \{\epsilon\}$ - dat kan gelijk zijn aan A, maar niet noodzakelijk.)

Neem $A = A_{TM}$ over alfabet Σ_1 en $B = \Sigma_2^*$ waarbij $\Sigma_1 \cap \Sigma_2 = \emptyset$. Beide zijn talen over $\Sigma_1 \cup \Sigma_2$, en voldoen aan de voorwaarden. Dan is $A \cap B$ leeg en beslisbaar.

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

Neem $A = A_{TM}$ over alfabet Σ_1 en $B = \Sigma_2^*$ waarbij $\Sigma_1 \cap \Sigma_2 = \emptyset$. Beide zijn talen over $\Sigma_1 \cup \Sigma_2$, en voldoen aan de voorwaarden. Dan is $A \cup B$ niet beslisbaar (werk zelf uit waarom dat zo is).

2 Accepteren, stoppen, reduceren

Beschrijf in detail een Turing-reductie $\leq_T$ van A_{TM} naar H_{TM}, en omgekeerd: als één van de twee richtingen niet mogelijk zijn, geef een argument - of misschien zijn geen van beide richtingen mogelijk ?

Antwoord Beide richtingen kunnen:

- $A_{TM} \leq_T H_{TM}$: het orakel voor H_{TM} noemen we H; de orakelmachine O^H werkt als volgt voor input $< M, w >$

  laat H lopen op de input $< M, w >$

  1. indien H accepteert, simuleer M op w; die simulatie stopt zeker
     - indien M w accepteert, dan accepteert $O^H < M, w >$
     - indien M w verwerpt, dan verwerpt $O^H < M, w >$
  2. indien H verwerpt, dan verwerpt $O^H < M, w >$

  O^H beslist A_{TM} relatief t.o.v. H_{TM}

- $H_{TM} \leq_T A_{TM}$: het orakel voor A_{TM} noemen we A; de orakelmachine O^A werkt als volgt voor input $< M, w >$

  laat A lopen op $< M, w >$

  1. indien A accepteert, dan accepteert $O^A < M, w >$
  2. indien A verwerpt, dan laat A lopen op $< M^c, w >$ (zie later voor wat M^c is)
     - indien $A < M^c, w >$ accepteert, dan accepteert $O^A < M, w >$
     - anders verwerpt $O^A < M, w >$

  O^A beslist H_{TM} relatief t.o.v. A_{TM}

M^c is de machine M met de accepterende toestand en de verwerpende toestand omgewisseld. Denk niet dat $L_{M^c} = \overline{L_M}$!

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) $\{< M, s > \mid \text{M } \textit{stopt na hoogstens } |s|^{17} \textit{ stappen bij invoer } s\}$;

Antwoord

(a) We hebben hier een taal-invariante, niet-triviale eigenschap van Turingmachines, dus passen we Rice toe: de taal is niet beslisbaar. $Pos_P = \{< M > \mid \exists s_1, s_2 \in L_M : |s_1| \neq |s_2|\}$; $Neg_P = \{< M > \mid \forall s_1, s_2 \in L_M : |s_1| = |s_2|\}$

- niet-triviaal: $\{a, aa\} \in Pos_P$ en $\{a, b\} \in Neg_P$ (beide over alfabet $\{a, b\}$
- taal-invariant: als $M_1 \in Pos_P$ en $L_{M_1} = L_{M_2}$, dan $M_2 \in Pos_P$ (die is niet exact de definitie van taal-invariant, maar wel equivalent)

De taal is herkenbaar met de volgende herkenner: die krijgt als invoer de encodering van een Turingmachine $< M >$ en doet het volgende

- maak van $< M >$ een enumerator $< E >$ voor M_L (zie de cursus)
- pas nu $< E >$ aan: telkens E een outputstring van M_L schrijft, vergelijk die in lengte met alle vorige geschreven strings; vind je er ééntje met verschillende lengte [1], dan accepteer je $< M >$, anders doe je voort

Voor $< M >$ in de taal eindigt dit zeker. Als $< M >$ niet in de taal zit (dus in Neg_P) dan kan dit blijven lopen, omdat de enumerator telkens weer dezelfde strings kan (en mag) outputten. 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. Bovendien is $|s|^{17}$ een Turingberekenbare functie, dus die UTM kan beginnen met die bovengrens te berekenen.

Merk op dat de taal $\{< M > \mid \forall s \in \Sigma^*, \ M \textit{ stopt na hoogstens } |s|^{17} \textit{ stappen bij invoer } s\}$ niet-beslisbaar is, maar wel co-herkenbaar.

[1] het is genoeg om enkel met de vorige te vergelijken

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 α-conversies 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:
We doen dat voor $\Sigma = \{0, 1\}$: dat is voldoende, omdat we een berekenbare omzetting kunnen doen van en naar andere alfabetten.
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 (M_w hoeft niet meer dan $|w|$ toestanden te hebben dan M [hoe ?], maar soms kan het met minder [hoe ?])
- tel het aantal toestanden X van M_w, en bereken (m.b.v. O_S) de bijhorende waarde $N_X = S(X)$ van de bezige-beverfunctie
- simuleer nu M op w voor ten hoogste N_X stappen; er zijn drie mogelijkheden

  1. M accepteert w voor de grens N_X bereikt is: M_O accepteert dan $< M, w >$
  2. M verwerpt w voor de grens N_X bereikt is: M_O verwerpt dan $< M, w >$
  3. de grens N_X wordt bereikt; d.w.z. dat M_w in een lus gaat, bijgevolg accepteert M niet w; M_O verwerpt dan $< M, w >$

  Vermits M_O altijd eindigt, is M_O een beslisser.

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 [2] taal L is de oneindige unie van talen die elk hoogstens één van de elementen van L bevatten: elk van die singleton of lege talen is beslisbaar (zelfs regulier). L kan gelijk welke taal zijn, ook eindig en zelfs leeg (door $B_i = \emptyset$ te nemen), maar hoeft dus niet beslisbaar te zijn.

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 een pijl de overeenkomstige lambda aan, zoals in: $\lambda x. x$
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. pijlen gebruiken we indices om aan te geven welke variabelen horen bij welke lambda.

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

[2] eindige of oneindige