TENTAMEN FUNDAMENTELE INFORMATICA 3
Vrijdag 5 juni 2015, 14.00 - 17.00 uur

Dit tentamen bestaat uit vijf opgaven.
Geef de gevraagde Turing machines door middel van hun transitiediagram.
Wanneer er bij een vraag om uitleg of toelichting gevraagd wordt, is het belangrijk om die ook te geven.
Als je het antwoord op een onderdeel niet weet, en je hebt dat antwoord nodig bij een later onderdeel, dan kun je het antwoord 'kopen' bij de docent.

---

1. (a) Construeer een gewone (deterministische, 1-tapes) Turing machine T die als het ware de taal

$$XX = \{ss \mid s \in \{a, b\}^*\}$$

opsomt (in een of andere volgorde). Dat wil zeggen: T begint met een lege invoer en genereert een tape-inhoud van de volgende vorm:

$$x_1 \# x_2 \# x_3 \# x_4 \# \ldots$$

waarbij

- de x_i's elementen van XX zijn,
- er voor elke $x \in XX$ precies één i is, zodat $x = x_i$,
- en de x_i's, als ze eenmaal 'definitief' zijn, niet meer overschreven worden.

Je mag voor T gebruik maken van de componenten NB, PB, $Insert(\sigma)$, $Delete$ en $Copy$ zoals die in het boek beschreven zijn. Andere componenten mag je alleen gebruiken als je ze zelf uitwerkt. Wellicht ten overvloede:

- NB verplaatst de leeskop naar de eerste Δ rechts van de huidige positie,
- PB verplaatst de leeskop (zo mogelijk) naar de eerste Δ links van de huidige positie,
- $Insert(\sigma)$ verandert de tape-inhoud van $y\underline{z}$ in $y\underline{\sigma}z$ (waarbij z geen Δ bevat),
- $Delete$ verandert de tape-inhoud van $y\underline{\sigma}z$ in $y\underline{z}$ (waarbij z geen Δ bevat),
- $Copy$ verandert de tape-inhoud van $\underline{\Delta}x$ in $\underline{\Delta}x\Delta x$ (waarbij x geen Δ bevat).

Leg ook duidelijk uit hoe T werkt.

N.B.: T hoeft niet te voldoen aan de definitie van 'a TM enumerating a language' uit het boek. Zie onderdeel (b).

(b) In welk(e) opzicht(en) wijkt je TM T uit onderdeel (a) af van de definitie van 'a TM enumerating a language' uit het boek?

---

2. Met een *unrestricted grammar* kun je niet alleen talen genereren, maar ook functies berekenen. We zeggen dat een unrestricted grammar $G = (V, \Sigma, S, P)$ een partiële functie $f : \Sigma^* \to \Sigma^*$ berekent, als er variabelen A, B, C, D in V zijn, zó dat voor elke $x, y \in \Sigma^*$

$$f(x) = y \qquad \text{dan en slechts dan als} \qquad AxB \Rightarrow_G^* CyD$$

Ofwel: de grammatica begint met de invoer x tussen de variabelen A en B, en produceert de string $f(x) = y$ tussen de variabelen C en D.

Laat nu $f : \{a, b\}^* \to \{a, b\}^*$ gedefinieerd zijn door $f(x) = xx$. Bijvoorbeeld: $f(aab) = aabaab$. Geef een unrestricted grammar G die de functie f berekent. Leg uit wat de functie is van de diverse variabelen en producties in G.

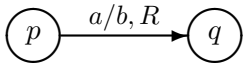
Als je geen unrestricted grammar voor de functie f weet, kun je een deel van de punten verdienen door een unrestricted grammar te geven die de taal $XX = \{ss \mid s \in \{a, b\}^*\}$ genereert.

---

3. In het bewijs van Stelling 8.14 in het boek wordt beschreven, hoe je bij een willekeurige Turing machine $T = (Q, \Sigma, \Gamma, q_0, \delta)$ een unrestricted grammar $G = (V, \Sigma, S, P)$ kunt construeren, zó dat $L(G) = L(T)$.

De grammatica G bevat drie soorten producties. De eerste soort is bedoeld om initiële configuraties van de Turing machine op te bouwen. De tweede soort is voor het simuleren van berekeningen van de Turing machine. De derde soort is voor het reconstrueren van de invoer van de Turing machine, als die invoer geaccepteerd wordt.

Wanneer de Turing machine T een transitie van de vorm



heeft, dan levert dat in G producties van de tweede soort op van de vorm $p(\sigma_1 a) \to (\sigma_1 b)q$ voor bepaalde σ_1.

(a) Leg uit waarom er tussen de haken in de productie steeds twee symbolen staan: bijvoorbeeld σ_1 en a. Wat is de functie van elk van de symbolen in zo'n paar met symbolen? Wat zijn de mogelijke waardes voor σ_1 in deze producties?

(b) Geef nu ook de producties van de eerste soort in G, de producties om initiële configuraties van T op te bouwen. Leg uit wat de functie is van de verschillende producties.

Je mag de producties algemeen beschrijven in termen van elementen σ_i van Σ en/of Γ. Als alternatief mag je ook aannemen dat $\Sigma = \{a, b\}$ en dat $\Gamma = \{a, b, c\}$.

(c) Geef nu ook de producties van de derde soort in G, de producties om de invoer van de Turing machine te reconstrueren als die invoer geaccepteerd wordt. Leg uit wat de functie is van de verschillende producties.

Je mag de producties algemeen beschrijven in termen van elementen σ_i van Σ en/of Γ. Als alternatief mag je ook aannemen dat $\Sigma = \{a, b\}$ en dat $\Gamma = \{a, b, c\}$.

---

4. De stelling van Rice luidt als volgt:

   Als R een niet-triviale taaleigenschap (*language property*) van Turing machines is, dan is het beslissingsprobleem

   P_R:
   Gegeven een Turing machine T, heeft T eigenschap R ?

   niet beslisbaar.

   (a) Wanneer noemen we een eigenschap R van Turing machines een *niet-triviale taaleigenschap*?

   (b) Voor welk van de volgende drie beslissingsproblemen is de stelling van Rice (direct) toepasbaar?

   - AcceptsSubsetXX:
     Gegeven een Turing machine T, accepteert T alleen maar woorden uit de taal $XX = \{ss \mid s \in \{a, b\}^*\}$?
     In andere woorden: accepteert T geen woorden buiten XX ?
   - AcceptsIntersectionWithXX:
     Gegeven twee Turing machines T_1 en T_2, is $L(T_1) = L(T_2) \cap XX$?
   - AcceptsUnionIsXX:
     Gegeven twee Turing machines T_1 en T_2, is $L(T_1) \cup L(T_2) = XX$?

   Motiveer je antwoorden. Dat wil zeggen: laat bij elk van de problemen zien waarom er wel of niet aan de voorwaarden van de stelling van Rice is voldaan. *Hint: bij minstens één van de problemen is de stelling van Rice niet (direct) toepasbaar.*

   (c) Kies een beslissingsprobleem hierboven waarop de stelling van Rice niet direct toepasbaar is, en toon aan dat dat probleem toch niet beslisbaar is, met behulp van een reductie met een ander beslissingsprobleem.

   Laat hierbij uiteraard zien dat aan alle eisen van een reductie voldaan is, en vergeet niet om de conclusie te trekken.

Z.O.Z.

5. (a) Laat de functies $f_1, \ldots, f_6$ gegeven zijn door:
   - $f_1 = C_0^0$,
   - $f_2 = p_2^2$,
   - $f_3 = s$ (de opvolger functie),
   - f_4 wordt verkregen uit f_3 en f_2 door compositie,
   - f_5 wordt verkregen uit f_3 en f_4 door compositie,
   - f_6 wordt verkregen uit f_1 en f_5 door primitieve recursie.

   Geef eenvoudige, gesloten formules voor de functies f_4, f_5 en f_6 (waarbij je ook duidelijk kunt zien hoeveel argumenten ze hebben). Onder een eenvoudige, gesloten formule verstaan we bijvoorbeeld een formule van de vorm $f(x_1, x_2) = x_1 * x_2$.

   (b) De functie *Mod* wordt als volgt gedefinieerd:

$$Mod(x, y) = \begin{cases} x & \text{als } y = 0 \\ \text{de rest bij deling van } x \text{ door } y & \text{als } y \geq 1 \end{cases}$$

   Om aan te tonen dat de functie *Mod* primitief recursief is, definieert het boek een functie R door $R(y, x) = Mod(x, y)$.

   Toon aan dat de functie R primitief recursief is. Je mag ervan uitgaan dat berekeningen als optellen en aftrekken, eenvoudige vergelijkingen, logische operaties en (volledige) gevalsonderscheiding primitieve recursiviteit behouden.

   Geef, wanneer je de operatie primitieve recursie gebruikt, ook een beschrijving van de daarbij voorkomende functies g en h voor algemene parameters $x_1, x_2, \ldots$ Je hoeft hierbij niet zover te gaan dat je projecties gebruikt.

`einde tentamen`