HERTENTAMEN FUNDAMENTELE INFORMATICA 3
Dinsdag 5 juli 2016, 10.00 - 13.00 uur

Dit tentamen bestaat uit zeven opgaven, waarbij steeds tussen [en] staat hoeveel punten er ongeveer mee te verdienen zijn. In totaal zijn er 100 punten te verdienen. Geef de gevraagde Turingmachines 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. [30 pt] Bij deze opgave moet je twee Turingmachines construeren die werken met natuurlijke getallen. Ga hierbij uit van de unaire representatie van de natuurlijke getallen.

   (a) Construeer een (gewone) Turingmachine T_1 die twee natuurlijke getallen x en y als invoer heeft, en die het product xy berekent. Om precies te zijn: T_1 begint in de configuratie $q_{01}\underline{\Delta}x\Delta y$ (met q_{01} de starttoestand van T_1) en eindigt in de configuratie $h_a\underline{\Delta}x\Delta y\Delta xy$.

   Leg duidelijk uit hoe T_1 werkt.

   Als je bij dit onderdeel gebruik wilt maken van componenten, zul je die ook moeten uitwerken (tekenen dus).

   (b) Construeer nu een (gewone) Turingmachine T_2 die twee **positieve** natuurlijke getallen n en x als invoer heeft, en die de macht x^n berekent. Om precies te zijn: T_2 begint in de configuratie $q_{02}\underline{\Delta}n\Delta x$ (met q_{02} de starttoestand van T_2) en eindigt in de configuratie $h_a\underline{\Delta}n\Delta x\Delta x^n$.

   Leg ook duidelijk uit hoe T_2 werkt.

   Je mag voor T_2 gebruik maken van de componenten *NB*, *PB*, *Insert*(σ), *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*(σ) 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).

   Hint: maak gebruik van Turingmachine T_1 uit onderdeel (a). Als je T_1 hiervoor iets moet aanpassen, beschrijf de benodigde aanpassing dan duidelijk.

---

2. [14 pt] In Stelling 7.26 van het boek wordt bewezen dat voor iedere 2-tapes TM T_1 een gewone 1-tapes TM T_2 bestaat, die dezelfde taal accepteert en (indien van toepassing) dezelfde functie berekent.

   De TM T_2 die in het bewijs van de stelling geconstrueerd wordt, simuleert T_1. Daartoe moet uiteraard de inhoud van de twee tapes van T_1 gecombineerd worden op de ene tape van T_2.

   Stel dat de invoer van T_1 (en dus ook van T_2) de string $x = a_1a_2 \dots a_n$ is, voor zekere $n \geq 1$. Dan wordt de tape-inhoud van T_2 eerst omgezet van $\Delta a_1 a_2 \dots a_n$ in $\$\Delta'\Delta' a_1 \Delta a_2 \Delta \dots \Delta a_n \#$

   Leg duidelijk uit waarom deze omzetting plaatsvindt. Besteed daarbij aandacht aan

   - de extra Δ's tussen de letters,
   - de accenten op enkele symbolen
   - en de $ en # (wat is de functie van deze symbolen, en wanneer komen we ze tegen).

---

3. [15 pt]

   (a) Wanneer noemen we een *unrestricted* grammatica $G = (V, \Sigma, S, P)$ context-gevoelig?

   Laat de taal L gedefinieerd zijn door $L = \{a^n b^{2^n} \mid n \geq 0\}$. Ofwel:

   $$L = \{b,\ abb,\ aabbbb,\ aaabbbbbbbb,\ \dots\}$$

   (b) Geef een context-gevoelige grammatica G, zó dat $L(G) = L$. Leg uit wat de functie is van de diverse variabelen en producties in G.

   Als het niet lukt om een context-gevoelige grammatica voor L te bedenken, kun je het grootste deel van punten verdienen met een unrestricted grammatica voor de taal, met de bijbehorende uitleg.

   (c) Geef een afleiding in G van het woord *aabbbb*.

---

4. [11 pt] Geef voor elk van de volgende drie verzamelingen L_i aan of L_i aftelbaar of overaftelbaar is.

   - L_1 = de verzameling van alle functies van $\mathbb{N}$ naar $\{0, 1\}$. Licht je antwoord kort (maximaal enkele regels) toe.
   - L_2 = de verzameling van alle functies van $\mathbb{N}$ naar $\mathbb{N}$. Licht je antwoord kort (maximaal enkele regels) toe.
   - L_3 = de verzameling van alle niet-dalende functies f van $\mathbb{N}$ naar $\mathbb{N}$ (voor elke $x \in \mathbb{N}$ geldt dus: $f(x + 1) \geq f(x)$). Licht je antwoord kort (maximaal enkele regels) toe.

---

5. [8 pt] Laat L_1 en L_2 twee recursief opsombare talen zijn.

   Toon aan dat ook $L_1 \cap L_2$ recursief opsombaar is, door (duidelijk) de werking van een Turingmachine T te beschrijven die $L_1 \cap L_2$ accepteert.

---

6. [10 pt] Beschouw de volgende twee beslissingsproblemen:

   > *Subset*: Gegeven twee Turingmachines T_1 en T_2, is $L(T_1) \subseteq L(T_2)$?
   >
   > *Equivalent*: Gegeven twee Turingmachines T_3 en T_4, is $L(T_3) = L(T_4)$?

   Gegeven is dat *Subset* niet beslisbaar is.

   Toon aan dat ook *Equivalent* niet beslisbaar is, met behulp van een reductie tussen *Subset* en *Equivalent*.

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

   Als je geen geschikte reductie tussen Equivalent *en* Subset *weet, kun je een deel van de punten daarvan verdienen door uit te leggen hoe je voor algemene beslissingsproblemen P_1 en P_2 aantoont dat $P_1 \leq P_2$.*

---

7. [12 pt] Bij geen enkel onderdeel van deze opgave is het nodig om projecties en dergelijke te gebruiken bij de beschrijving van de gevraagde functies. Ook hoef je bij het beschrijven van de operatie van primitieve recursie de benodigde functies g en h niet voor algemene parameters $x, y, z, \ldots$ te beschrijven.

   (a) Toon aan dat de functie *Pow* gedefinieerd door $Pow(x, y) = x^y$ primitief recursief is. Je mag ervan uitgaan dat operaties als optellen, aftrekken en vermenigvuldigen primitief recursief zijn.

   De functie $PrNo(i)$ levert het i-de priemgetal op, ofwel: $PrNo(0) = 2, PrNo(1) = 3, PrNo(2) = 5, \ldots$.
   De functie $Exponent(i, x)$ levert de exponent van $PrNo(i)$ in de priemfactorisatie van x op, bijvoorbeeld $Exponent(1, 18) = 2$, want de exponent van priemgetal 3 in $x = 18$ is 2. Als $x = 0$, definiëren we $Exponent(i, x) = 0$.

   De functie *HighestPrime* wordt gedefinieerd door

$$HighestPrime(k) = \begin{cases} 0 & \text{als } k \leq 1 \\ \max\{i \mid Exponent(i, k) > 0\} & \text{als } k \geq 2 \end{cases}$$

   Ofwel, $HighestPrime(k)$ is het nummer van de grootste priemfactor van k.

   (b) Toon aan dat de functie *HighestPrime* primitief recursief is.
   Rechtvaardig de keuzes die je hierbij maakt.
   Je mag gebruiken dat de functies *PrNo* en *Exponent* primitief recursief zijn, dat vergelijkingen tussen twee getallen primitief recursief zijn, en dat operaties als volledige gevalsonderscheiding, begrensde minimalisatie en begrensde maximalisatie de primitieve recursiviteit behouden.

`einde tentamen`