HERTENTAMEN FUNDAMENTELE INFORMATICA 3
Maandag 28 mei 2018, 14.00 - 17.00 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. Als je daarbij gebruik wilt maken van componenten, zul je die ook moeten uitwerken (tekenen dus).
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. [17 pt]

   (a) Teken een gewone (deterministische, 1-tape) Turingmachine T, zó dat

   $$L(T) = \{x \in \{a, b, c\}^* \mid n_a(x) = n_b(x) = n_c(x)\}$$

   Leg duidelijk uit hoe T werkt.

   (b) Leg uit wat er gebeurt met T bij invoer $x_1 = abcac$ en $x_2 = bbcacc$. Gaat T naar h_r of crasht T? Zo ja, geef dan de laatste configuratie van T voordat dat gebeurt.

---

2. [20 pt]

   (a) Teken een niet-deterministische Turingmachine T_1 die een string $x \in \{a, b\}^*$ als invoer heeft (en dus begint in configuratie $q_0\Delta x$), en die eindigt in configuratie $h_a\Delta y$, waarbij y een willekeurig anagram ('een permutatie van de letters') van x is.
   Leg duidelijk uit hoe T_1 werkt.

   (b) Beschrijf in woorden (kort) hoe je T_1 kunt gebruiken om een (eveneens niet-deterministische) Turingmachine T_2 te construeren die de taal

   $$L = \{x \in \{a, b\}^* \mid n_a(x) = n_b(x)\}$$

   accepteert. T_2 hoef je dus niet per se te tekenen.
   Beredeneer dat T_2 inderdaad precies L accepteert.

---

3. [10 pt]

   (a) Wanneer noemen we een taal L (volgens de definitie) recursief opsombaar?
   Wanneer noemen we een taal L (volgens de definitie) recursief?

   (b) Toon aan dat als een taal L en zijn complement L' allebei recursief opsombaar zijn, dat L dan recursief is.

---

4. [15 pt] Bij een willekeurig tentamen van een willekeurig vak werd gevraagd om een *unrestricted grammar* G voor de taal

$$L_0 = \{1^{n^2} \mid n \geq 0\} = \{\Lambda, 1, 1111, 111111111, \ldots\}$$

(alle kwadraten in unaire notatie). Het idee was om (1) eerst strings van de vorm LA^nB^nR te genereren, en (2) vervolgens in die strings elke A over de B's heen naar rechts te laten lopen, waarbij de A bij elke B een 1 achterlaat. Op die manier krijg je precies $n \times n = n^2$ 1'en.

Een willekeurige student gaf als antwoord een grammatica G_1, met startsymbool S en de volgende producties:

$$\begin{array}{ccc} S \to LABR & & AB \to AABB \\ AB \to 1BA & A1 \to 1A & AR \to R \\ B1 \to 1B & L1 \to 1L & LB \to L \\ & LR \to \Lambda & \end{array}$$

Grammatica G_1 is bijna correct, maar net niet helemaal. Toon dit laatste aan. Dat wil zeggen: doe een van de volgende twee dingen (of allebei, als beide dingen mogelijk zijn):

- geef een string $x \in \{1\}^*$, die kan worden afgeleid in G_1, maar die niet in L_0 zit. Geef in dit geval ook een afleiding van x in G_1. In de afleiding mag je meerdere stappen achter elkaar samenvatten met $\Rightarrow^*$, als het om gelijksoortige producties gaat;
- geef een string $x \in L_0$, die niet kan worden afgeleid in G_1.

5. [11 pt] In het bewijs van Stelling 8.14 in het boek wordt beschreven, hoe je bij een willekeurige Turingmachine $T = (Q, \Sigma, \Gamma, q_0, \delta)$ 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 Turingmachine na te bouwen. De tweede soort is voor het simuleren van berekeningen van de Turingmachine. De derde soort is voor het reconstrueren van de invoer van de Turingmachine, aan het eind van de berekening.

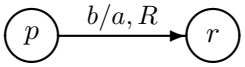
   (a) Wanneer we met G een berekening van T voor de invoer $a_1 \dots a_n \in \Sigma^*$ willen simuleren, wordt met de producties van de eerste soort de volgende string gegenereerd:

   $$q_0(\Delta\Delta)(a_1a_1)\dots(a_na_n)(\Delta\Delta)\dots(\Delta\Delta)$$

   Leg duidelijk uit waarom de letters a_i dubbel voorkomen in de string; wat is de rol van elk van de twee voorkomens bij het simuleren van de berekening?

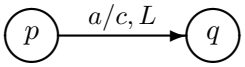
   (b) We gaan nu kijken naar de producties van de tweede soort. Je mag deze producties hieronder algemeen beschrijven, met behulp van variabele(n) σ_i; zeg dan wel wat de waarde(s) van de σ_i('s) kan/kunnen zijn. Als alternatief mag je ook aannemen dat $\Sigma = \{a, b\}$ en dat $\Gamma = \{a, b, c\}$.

      i. Stel dat T de volgende transitie bevat:

      Geef de producties van de tweede soort in G die met deze transitie corresponderen.

      ii. Stel dat T de volgende transitie bevat:

      Geef de producties van de tweede soort in G die met deze transitie corresponderen.

---

6. [13 pt]

   (a) Wanneer noemen we een verzameling A *aftelbaar oneindig*?
   Wanneer noemen we een verzameling A *aftelbaar*?

   (b) Geef voor elk van de volgende verzamelingen aan of ze aftelbaar of overaftelbaar zijn (je hoeft je antwoorden niet toe te lichten):

      i. de verzameling van alle reguliere talen over een alfabet Σ,
      ii. de verzameling van alle recursief opsombare talen over een alfabet Σ,
      iii. de verzameling van alle talen over een alfabet Σ,
      iv. de verzameling van alle eindige deelverzamelingen van $\mathbb{N}$.

---

7. [14 pt] Beschouw de volgende twee beslissingsproblemen:

   Accepts:
   Gegeven een Turingmachine T_1 en een string x, accepteert T_1 de string x ?

   ReachesState:
   Gegeven een Turingmachine T_2 en een non-halting state q van T_2, bereikt T_2 ooit toestand q als T_2 start met een lege tape (voor invoer Λ dus) ?

   Gegeven is dat *Accepts* niet beslisbaar is.

   Toon aan dat ook *ReachesState* niet beslisbaar is, met behulp van een reductie met *Accepts*. Laat uiteraard ook zien dat aan alle eisen van een reductie voldaan is, en vergeet niet om de conclusie te trekken.

   Als je geen geschikte reductie tussen ReachesState *en* Accepts *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 $P_1 \leq P_2$.*

`einde tentamen`