HERTENTAMEN FUNDAMENTELE INFORMATICA 3
Maandag 20 augustus 2012, 10.00 - 13.00 uur

Dit tentamen bestaat uit 6 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 stapelautomaten en 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. [21 pt] Laat G_1 een context-vrije grammatica zijn, met startsymbool A, terminaal alfabet $\Sigma = \{a, b, c\}$ en producties

$$A \to bB \mid \Lambda \qquad B \to Ac \mid aBB$$

   (a) Construeer volgens de bottom-up methode een stapelautomaat M_1 zó dat $L(M_1) = L(G_1)$.

   Laat G_2 een context-vrije grammatica zijn, met startsymbool S, terminaal alfabet $\Sigma = \{a, b, c, \$\}$ en producties

$$S \to S_1\$ \qquad S_1 \to S_1a \mid bAa \mid bc \mid a \qquad A \to bB \mid \Lambda \qquad B \to Ac \mid aBB$$

   (b) Elimineer de linksrecursie in G_2, zonder de gegenereerde taal te veranderen. Het resultaat noemen we G_3.

   (c) Pas in G_3 factorisatie toe op die producties bij eenzelfde variabele waarvoor de rechterzijden met hetzelfde symbool beginnen. De resulterende grammatica noemen we G_4.

   (d) Geef voor elk van de producties van de grammatica G_4 uit het vorige onderdeel de bijbehorende lookahead, d.w.z. geef aan bij welke terminalen (als volgende invoersymbool) de productie hoort.

   (e) Geef een deterministische stapelautomaat die als top-down parser voor $L(G_4)$ kan fungeren. *Om de hoeveelheid werk te beperken, hoef je alleen de producties vanuit de variabelen A en B in de stapelautomaat te verwerken.*

---

2. [14 pt] Laat $L_1 = \{a^i b^j c^k \mid i, j, k \geq 0 \text{ en } 2i > j\}$.

   (a) Geef de eerste vijf elementen van L_1 in de canonieke volgorde.

   (b) Geef een deterministische stapelautomaat M_1 zó dat $L(M_1) = L_1$.

   (c) Geef een voorbeeld van een context-vrije taal L_2 waarvoor geldt dat $L_1 \cap L_2$ niet context-vrij is. Leg ook intuïtief uit waarom $L_1 \cap L_2$ niet context-vrij is.

---

3. [14 pt] Construeer een Turing machine T die als invoer een niet-negatief unair getal x krijgt, en deze accepteert dan en slechts dan als x een macht van 3 is.

   Als dit niet lukt, kun je een deel van de punten verdienen met een Turing machine die x accepteert dan en slechts dan als x een macht van 2 is.

   Indien T gebruik maakt van componenten, zul je ook die componenten moeten geven. Leg duidelijk uit hoe T werkt. Leg in het bijzonder ook uit wat T doet als zijn invoer x geen macht van 3 (respectievelijk: macht van 2) is.

---

4. [11 pt] Je hebt Turing machines (TMs) en niet-deterministische Turing machines (NTMs). Ook heb je stapelautomaten (PDAs) en deterministische stapelautomaten (DPDAs).

   (a) Hoe wordt 'acceptatie van een string x' gedefinieerd voor een NTM T? Het is voldoende om een antwoord 'in woorden' te geven; je hoeft dus geen formules te gebruiken (maar het mag natuurlijk wel).

   (b) Zijn er talen die wel door een NTM geaccepteerd kunnen worden, maar niet door een TM? Zo ja, geef een voorbeeld van zo'n taal, en leg intuïtief uit waarom die taal niet door een TM geaccepteerd kan worden (een formeel bewijs is niet nodig).

   (c) Zijn er talen die wel door een PDA geaccepteerd kunnen worden, maar niet door een DPDA? Zo ja, geef een voorbeeld van zo'n taal, en leg intuïtief uit waarom die taal niet door een DPDA geaccepteerd kan worden (een formeel bewijs is niet nodig).

---

5. [19 pt] In het bewijs van Stelling 8.13 in het boek wordt beschreven, hoe je bij een willekeurige unrestricted grammatica $G = (V, \Sigma, S, P)$ een niet-deterministische Turing machine (NTM) $T = (Q, \Sigma, \Gamma, q_0, \delta)$ kunt construeren, zó dat $L(T) = L(G)$. Dat wil zeggen: T accepteert precies die invoerstrings die door G gegenereerd worden.

   De werking van T bestaat uit drie fasen, en een van deze fasen is het simuleren van een afleiding in G.

   (a) Beschrijf (in woorden) hoe het simuleren van een afleiding in G gebeurt door T. Besteed hierbij tevens aandacht aan de volgende vragen: Hoe begint de simulatie? Wanneer stopt de simulatie? Waar is er sprake van niet-determinisme?

   (b) Wat zijn de andere twee fasen in de werking van T, en in welke volgorde vinden de drie fasen plaats?

   (c) Toon aan dat de hierboven beschreven NTM T inderdaad precies $L(G)$ accepteert.

---

6. [`21 pt`] Post's Correspondence Problem (*PCP*) luidt als volgt:

   Gegeven een verzameling paren

   $$\{(\alpha_1, \beta_1), (\alpha_2, \beta_2), \ldots, (\alpha_n, \beta_n)\},$$

   waarbij de α_i's en de β_i's niet-lege strings over een alfabet Σ zijn. Bestaat er een rijtje indices $i_1, i_2, \ldots, i_k$ (met $k \geq 1$) zó dat

   $$\alpha_{i_1}\alpha_{i_2}\ldots\alpha_{i_k} = \beta_{i_1}\beta_{i_2}\ldots\beta_{i_k}$$

   Het Modified Post's Correspondence Problem (*MPCP*) luidt vrijwel hetzelfde. Alleen wordt aan het eind gevraagd:

   Bestaat er een rijtje indices $i_2, \ldots, i_k$ (met $k \geq 1$) zó dat

   $$\alpha_1\alpha_{i_2}\ldots\alpha_{i_k} = \beta_1\beta_{i_2}\ldots\beta_{i_k}$$

   De verzameling paren (α_i, β_i) wordt een correspondentiesysteem genoemd, en kan grafisch worden weergegeven als een verzameling 'dominostenen'.

   (a) Beschouw het volgende correspondentiesysteem I:

0110	110	0	1
0	00	011	11

      i. Is I een ja-instantie van *PCP*? Zo ja, geef een match voor het systeem. Zo nee, beargumenteer waarom niet.
      ii. Is I een ja-instantie van *MPCP*? Zo ja, geef een match voor het systeem. Zo nee, beargumenteer waarom niet.

   (b) Gegeven is dat *MPCP* niet beslisbaar is. Toon aan dat ook *PCP* niet beslisbaar is, met behulp van een reductie tussen *MPCP* en *PCP*.

   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 PCP *en* MPCP *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`