HERTENTAMEN FUNDAMENTELE INFORMATICA 3
Dinsdag 6 augustus 2013, 14.00 - 17.00 uur

Dit tentamen bestaat uit 5 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 Turing machine T_1 die als invoer een string $x \in \{0,1\}^*$ heeft, de string opvat als een binaire representatie van een getal n, en als uitvoer een binaire representatie van $n-1$ oplevert.

   Om precies te zijn: laat de string y een binaire representatie van $n-1$ zijn, en laat q_0 de begintoestand van T_1 zijn, dan moet T_1 een berekening van $q_0\Delta x$ naar $h_a\Delta y$ uitvoeren.

   Zowel de invoer als de uitvoer van T_1 mag voorloopnullen bevatten. Als $n = 0$, mag T_1 h_a niet bereiken. Als $x = \Lambda$, komt dat overeen met $n = 0$.

   Indien T_1 gebruik maakt van componenten, zul je ook die componenten moeten geven. Leg ook duidelijk uit hoe T_1 werkt.

   (b) Construeer een Turing machine T_2 die als invoer een string $x \in \{0,1\}^*$ heeft, de string opvat als een binaire representatie van een getal n, en als uitvoer de unaire representatie van n oplevert.

   Om precies te zijn: als q_0 de begintoestand van T_2 is, dan moet T_2 een berekening van $q_0\Delta x$ naar $h_a\Delta 1^n$ uitvoeren.

   De invoer van T_2 mag voorloopnullen bevatten. Als $x = \Lambda$, komt dat overeen met $n = 0$.

   Bij dit onderdeel mag je gebruik maken van de componenten *Insert*(σ) en *Delete* (en niet van andere componenten, tenzij je ze zelf uitwerkt). *Insert*(σ) verandert de tape-inhoud van een Turing machine van $y\underline{z}$ in $y\underline{\sigma}z$ (waarbij z geen Δ bevat) en *Delete* doet precies het omgekeerde. Leg ook duidelijk uit hoe T_2 werkt.

   Hint: maak gebruik van de Turing machine T_1 uit onderdeel (a). Desgewenst mag je die iets aanpassen, als je maar duidelijk maakt waar je hem aanpast.

---

2. Laat $G_1 = (V, \Sigma, S, P)$ de unrestricted grammatica zijn met $V = \{S, A, B, C, L\}$, $\Sigma = \{a, b, c\}$ en de volgende producties:
   $S \to SABBC \mid SAB \mid LAB$
   $BA \to AB \qquad CA \to AC \qquad CB \to BC$
   $LA \to a \qquad aA \to aa \qquad aB \to ab \qquad bB \to bb \qquad bC \to bc \qquad cC \to cc$

   (a) Wat zijn de eerste vier elementen van $L(G_1)$ in canonieke volgorde? Geef deze elementen ook in de canonieke volgorde.

   (b) Wat is $L(G_1)$? Geef een precies antwoord in woorden of formules. Motiveer je antwoord, door uit te leggen wat de functie is van de diverse variabelen en producties in G_1.

   (c) Geef een context-gevoelige grammatica G_2, zó dat $L(G_2) = L(G_1)$.

---

3. In het boek wordt beschreven hoe je bij een willekeurige unrestricted grammatica G een niet-deterministische Turing machine (NTM) T kunt construeren, zó dat $L(T) = L(G)$. De NTM moet dus precies die strings accepteren, die de grammatica genereert. De werking van T bestaat uit drie fases. De tweede fase bestaat eruit dat T op zijn tape een willekeurige afleiding in G simuleert.

   (a) Wat gebeurt er in de andere twee fases van T? Beargumenteer ook dat de drie fases er inderdaad voor zorgen dat $L(T) = L(G)$.

   (b) Laat $G = (V, \Sigma, S, P)$ de unrestricted grammatica zijn met $V = \{S, A, B, R\}$, $\Sigma = \{a, b\}$ en de volgende producties:
   $S \to ABS \mid ABR \qquad BA \to AB$
   $BR \to b \qquad Bb \to bb \qquad Ab \to ab \qquad Aa \to aa$
   Geef een NTM T_2 die op zijn tape een willekeurige afleiding in deze grammatica G simuleert (vanuit S) en het resultaat van de afleiding op de tape achterlaat. T_2 kan dus dienen als component voor de tweede fase van T.

   Om precies te zijn: laat de string y het resultaat van de afleiding zijn, en laat q_0 de begintoestand van T_2 zijn, dan moet T_2 beginnen in configuratie $q_0\Delta$ en eindigen in configuratie $h_a\Delta y$.

   Leg ook duidelijk uit hoe T_2 werkt.

---

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

   De stelling van Rice luidt als volgt:

   > Als R een niet-triviale taaleigenschap van Turing machines is, dan is het beslissingsprobleem
   >
   > P_R:
   > Gegeven een Turing machine T, heeft T eigenschap R ?
   >
   > niet beslisbaar.

   In het boek wordt de stelling van Rice bewezen met behulp van een reductie tussen *Accepts*-Λ en P_R, waarbij *Accepts*-Λ als volgt gedefinieerd is:

   > *Accepts*-Λ: Gegeven een Turing machine T, is $\Lambda \in L(T)$?

   Gegeven is dat *Accepts*-Λ niet beslisbaar is.

   (b) Moeten we, om de stelling van Rice op deze manier te bewijzen, aantonen dat *Accepts*-$\Lambda \leq P_R$ of dat $P_R \leq$ *Accepts*-Λ ? Motiveer je antwoord.

   (c) In het bewijs in het boek wordt onder meer verwezen naar een Turing machine T_0 die helemaal niets accepteert. Neem aan dat T_0 eigenschap R niet heeft, en laat T_R een Turing machine zijn die eigenschap R wel heeft. Dan wordt er bij een instantie T_1 van het ene probleem als volgt een instantie T_2 van het andere probleem gemaakt:

   - T_2 krijgt een invoer x,
   - loopt voorbij x,
   - voert dan T_1 uit.
   - Als T_1 h_a bereikt,
     - veegt T_2 de tape rechts van x schoon,
     - gaat T_2 terug naar positie 0 op de tape, links van x,
     - en voert dan T_R uit.

   Van welk probleem (*Accepts*-Λ of P_R) moet T_1 een instantie zijn om de bij onderdeel (b) gewenste reductie te krijgen, en van welk probleem wordt T_2 dan een instantie?

   Toon aan dat met deze constructie inderdaad aan alle eisen van de gewenste reductie is voldaan.

   Als je deze laatste vraag niet kunt beantwoorden, 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 $P_1 \leq P_2$.

5. 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 *Mod* wordt als volgt gedefinieerd:

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

   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 $Exponent(i, x) = 0$.

   (b) Stel dat $x \geq 1$ en dat $y = Exponent(i, x) + 1$.
   Wat weten we dan van de waarde van $Mod(x, PrNo(i)^{y-1})$ en van de waarde van $Mod(x, PrNo(i)^y)$?

   (c) Gegeven is dat de functies *PrNo* en *Mod* primitief recursief zijn.
   Toon aan dat de functie *Exponent* primitief recursief is. Je mag gebruiken dat primitieve recursiviteit behouden blijft bij toepassing van de operaties begrensde kwantificatie en begrensde minimalisatie. Maak duidelijk welke stappen je allemaal maakt in dit bewijs.

`einde tentamen`