TENTAMEN FUNDAMENTELE INFORMATICA 3
Vrijdag 3 juni 2016, 14.00 - 17.00 uur

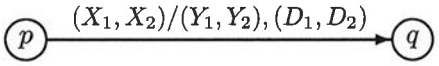
Dit tentamen bestaat uit vijf opgaven.
Geef de gevraagde Turingmachines door middel van hun transitiediagram.
Wanneer er bij een vraag om uitleg of toelichting gevraagd wordt, is het belangrijk om die ook te geven.
Als je het antwoord op een onderdeel niet weet, en je hebt dat antwoord nodig bij een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

---

1. Als je bij deze opgave gebruik wilt maken van componenten, zul je die ook moeten uitwerken (tekenen dus).

   (a) Construeer een (gewone) Turingmachine T_1 die twee strings $w_1, w_2 \in \{a, b\}^*$ als invoer heeft, en die accepteert, dan en slechts dan als $n_a(w_1) = n_a(w_2)$ (ofwel: w_1 en w_2 bevatten evenveel a's).
   Leg duidelijk uit hoe T_1 werkt.

   (b) Construeer nu een 2-tapes Turingmachine T_2 die twee strings $w_1, w_2 \in \{a, b\}^*$ als invoer heeft (allebei op de eerste tape, zoals gebruikelijk, eerst w_1, dan w_2) en die *in lineaire tijd* bepaalt of w_2 een anagram ('een permutatie van de letters') van w_1 is. Zo ja, dan accepteert T_2. Zo nee, dan accepteert T_2 niet.
   Wellicht ten overvloede: een 2-tapes Turingmachine heeft transities van de vorm:

   Je mag ook variabelen $X, Y, \ldots$ gebruiken om meerdere transities samen te vatten, als je maar duidelijk maakt wat $X, Y, \ldots$ dan kunnen zijn.
   Hint: om te bepalen of w_2 een anagram van w_1 is, zou je kunnen kijken naar het aantal voorkomens van letters in w_1 en w_2.

---

2. (a) Geef een *unrestricted* grammatica G voor de taal

   $$\{xyx^r \mid x \in \{a, b\}^* \text{ en } y \text{ is een anagram van } x\}$$

   Voor 'anagram', zie opgave 1. Merk op dat x^r betekent 'x-reverse' en niet 'x-tot-de-r-de'.
   Leg uit wat de functie is van de diverse variabelen en producties in G.

   (b) Geef een afleiding in G voor het woord *aababababaa*, dus $x = aab$ en $y = aba$.
   Bij toepassing van meerdere 'gelijksoortige' producties in de afleiding, mag je de stappen samenvatten met $\Rightarrow^*$.

---

3. In het bewijs van Stelling 8.13 in het boek wordt beschreven hoe je bij een willekeurige unrestricted grammatica G een niet-deterministische Turingmachine (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 $V = \{S, A, B, C\}$, $\Sigma = \{a, b, c\}$ en de volgende producties:
   $S \to ABCS \mid \Lambda$
   $BA \to AB \qquad CB \to BC \qquad CA \to AC$
   $A \to a \qquad B \to b \qquad C \to c$

   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 na het simuleren van de afleiding eindigen in configuratie $h_a\Delta y$.

   Leg ook duidelijk uit hoe T_2 werkt.

4. De stelling van Rice luidt als volgt:

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

   P_R:
   Gegeven een Turingmachine T_2, heeft T_2 eigenschap R ?

   niet beslisbaar.

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

   (b) In het bewijs van de stelling van Rice wordt, zonder beperking der algemeenheid, aangenomen dat een Turingmachine T_0 die de lege taal $\emptyset$ accepteert, eigenschap R niet heeft.
   Vervolgens wordt het beslissingsprobleem

   Accepts-Λ:
   Gegeven een Turingmachine T_1, is $\Lambda \in L(T_1)$?

   naar P_R gereduceerd. Daartoe wordt voor een willekeurige instantie T_1 van *Accepts-*Λ een instantie T_2 van P_R geconstrueerd, zó dat T_1 een ja-instantie van *Accepts-*Λ is, dan en slechts dan als T_2 een ja-instantie van P_R is.
   Leg duidelijk uit hoe, voor een willekeurige instantie T_1 van *Accepts-*Λ, de Turingmachine T_2 eruit ziet / hoe T_2 werkt.

   (c) Toon aan dat voor de geconstrueerde Turingmachine T_2 inderdaad geldt, dat T_1 een ja-instantie van *Accepts-*Λ is, dan en slechts dan als T_2 een ja-instantie van P_R is.

---

5. (a) Laat $n \geq 0$ en laat $g : \mathbb{N}^n \to \mathbb{N}$ en $h : \mathbb{N}^{n+2} \to \mathbb{N}$ twee functies zijn.
   Wat betekent het als we zeggen dat een functie f is ontstaan uit g en h door de operatie van primitieve recursie? Hoeveel argumenten heeft de functie f in dit geval?

   (b) **Bij dit onderdeel mag je gebruiken dat volledige gevalsonderscheiding de primitieve recursiviteit behoudt. Ook hoef je bij je bewijs niet zover te gaan dat je ook projecties gebruikt.**
   Laat P een $(n+1)$-*place* predikaat zijn, en laat de begrensde maximalisatie van P gedefinieerd zijn door.

$$m^P(X,k) = \begin{cases} \max\{y \leq k \mid P(X,y) \text{ is true}\} & \text{als deze verzameling niet leeg is} \\ 0 & \text{anders} \end{cases}$$

   Toon rechtstreeks met primitieve recursie aan dat de begrensde maximalisatie van een primitief recursief predikaat zelf ook primitief recursief is.
   Geef bij toepassing van de operatie primitieve recursie de gebruikte functies g en h ook voor algemene parameters $X_1, x_2, x_3, \ldots$.

`einde tentamen`