EXTRA TENTAMEN FUNDAMENTELE INFORMATICA 3
Maandag 14 augustus 2017, 14.00 - 17.00 uur

Dit tentamen bestaat uit vijf 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. [24 pt] Laat
$$L = \{a^i b^j a^k \mid 0 \leq i \leq j < k\}$$
   (a) Geef de eerste zes elementen van L in de canonieke volgorde.
   (b) Construeer een (gewone, 1-tape) Turing machine T die als invoer een string $x \in \{a, b\}^*$ heeft, en x accepteert dan en slechts dan als $x \in L$. Leg ook duidelijk uit hoe T werkt.
   (c) Wat doet T voor de invoeren $x = bab$ en $x = abba$? Dat wil zeggen: wanneer (en in welke toestand) ontdekt T voor elk van deze twee invoeren dat x niet in L zit en wat doet T dan?

---

2. [19 pt]
   (a) Laat $T_1 = (Q, \Sigma, \Gamma, q_0, \delta)$ een niet-deterministische Turingmachine (NTM) zijn, en laat $x \in \Sigma^*$.
      i. Wat is de initiële configuratie van T_1 die correspondeert met invoer x?
      ii. Leg uit in woorden wat het betekent als x door T_1 geaccepteerd wordt.

   In Stelling 7.31 van het boek wordt bewezen dat er voor iedere NTM T_1 een gewone TM T_2 bestaat met $L(T_2) = L(T_1)$.

   In het bewijs wordt voor het gemak aangenomen dat er in de NTM T_1 voor iedere combinatie van niet-halting toestand en tapesymbool precies twee mogelijke transities zijn. De paden die T_1 voor een bepaalde invoer x kan doorlopen corresponderen dan mooi met bitstrings. De TM T_2 gaat nu met behulp van vier tapes alle mogelijke paden van T_1 simuleren.[1]

   (b) Tijdens het simuleren van alle mogelijke paden/bitstrings van T_1 bevat de eerste tape van T_2 steeds de invoer x. Waarom is dit? Ofwel: waarom is het belangrijk om de invoer x ongeschonden op een van de tapes te hebben staan?
   (c) Wat staat er op elk van de andere drie tapes van T_2, tijdens het simuleren van alle mogelijke paden van T_1 ?
   (d) In welke situatie(s) gaat T_2 bij het simuleren van alle mogelijke paden van T_1 expliciet naar een halting state (zodat het simuleren stopt)? En naar welke halting state gaat T_2 in dat geval, cq in die gevallen?

---

[1] Inderdaad, met vier tapes is T_2 geen gewone Turingmachine. T_2 is echter wel deterministisch, en kan daarom op zijn beurt gesimuleerd worden door een echt gewone Turingmachine.

3. [16 pt] Bij een willekeurig tentamen van een willekeurig vak werd gevraagd om een context-gevoelige grammatica G voor de taal

$$L_0 = \{a^{2n}da^nda^{3n} \mid n \geq 1\}$$

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

$$\begin{array}{c} S \to aadAdaaa \\ A \to XXaAZZZ \mid a \\ dX \to ad \qquad Zd \to da \end{array}$$

      i. 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 $x \in \{a, d\}^*$, 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.

      ii. Pas G_1 aan, zó dat de grammatica wel precies L_0 genereert. Probeer hierbij zo weinig mogelijk producties toe te voegen, te verwijderen of te veranderen.

   (b) Een andere willekeurige student gaf als antwoord een grammatica G_2, met startsymbool S en de volgende producties:

$$\begin{array}{ccc} S \to DTD & & \\ T \to LBTR \mid LBR & & \\ BL \to LB & DL \to LD & RD \to DR \\ LD \to aad & La \to aaa & \\ DR \to daaa & aR \to aaaa & \\ dB \to da & aB \to aa & \end{array}$$

      i. Grammatica G_2 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 $x \in \{a, d\}^*$, die kan worden afgeleid in G_2, maar die niet in L_0 zit. Geef in dit geval ook een afleiding van x in G_2. 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_2.

      ii. Pas G_2 aan, zó dat de grammatica wel precies L_0 genereert. Probeer hierbij zo weinig mogelijk producties toe te voegen, te verwijderen of te veranderen.

4. [21 pt] 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, heeft T eigenschap R ?

   niet beslisbaar.

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

   (b) Beschouw het volgende beslissingsprobleem:

   AcceptsEven: Gegeven een Turingmachine T, met invoeralfabet Σ, accepteert T precies alle strings $x \in \Sigma^*$ van even lengte (en geen andere strings).

   Toon met behulp van de stelling van Rice aan dat *AcceptsEven* niet beslisbaar is: Dat wil zeggen: laat duidelijk zien dat aan alle voorwaarden van de stelling van Rice is voldaan. Gebruik voor het aantonen van niet-trivialiteit concrete Turingmachines.

   (c) Beschouw nu het volgende beslissingsprobleem:

   AcceptsOdd: Gegeven een Turingmachine T, met invoeralfabet Σ, accepteert T precies alle strings $x \in \Sigma^*$ van oneven lengte (en geen andere strings).

   Toon aan dat ook *AcceptsOdd* niet beslisbaar is, met behulp van een reductie met *AcceptsEven*. 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 AcceptsOdd *en* AcceptsEven *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 $P_1 \leq P_2$.*

5. [20 pt]

   (a) De functie *Div* wordt als volgt gedefinieerd:

$$Div(x,y) = \begin{cases} 0 & \text{als } y = 0 \\ \lfloor x/y \rfloor \ (x/y \text{ naar beneden afgerond}) & \text{als } y \geq 1 \end{cases}$$

   Om aan te tonen dat de functie *Div* primitief recursief is, definieert het boek een functie Q door $Q(y,x) = Div(x,y)$.

   Toon aan dat de functie Q primitief recursief is. Je mag ervan uitgaan

   - dat berekeningen als optellen, aftrekken en vermenigvuldigen, en eenvoudige vergelijkingen tussen getallen primitief recursief zijn,
   - en dat logische operaties als $\wedge$, $\vee$ en $\neg$, en (volledige) gevalsonderscheiding de primitieve recursiviteit behouden.

   Geef, wanneer je de operatie primitieve recursie gebruikt, ook een beschrijving van de daarbij voorkomende functies g en h voor algemene parameters $x_1, x_2, \ldots$ Je hoeft hierbij niet zover te gaan dat je projecties en dergelijke gebruikt.

   Laat $T = (Q, \Sigma, \Gamma, q_0, \delta)$ een Turingmachine zijn.

   (b) Leg precies uit wat het *tape number* behorend bij een configuratie xqy van T is. Hierin is $x = x_0x_1 \ldots x_{i-1} \in (\Gamma \cup \{\Delta\})^*$, q een toestand en $y = x_ix_{i+1} \ldots x_j \in (\Gamma \cup \{\Delta\})^*$.

   N.B.: het gaat dus alleen om het tape number, en niet om het *configuration number*. En de strings x en y in de configuratie hebben natuurlijk niets met de getallen x en y in het vorige onderdeel te maken.

   (c) Om aan te tonen dat elke berekenbare functie μ-recursief is, wordt onder andere aangetoond dat iedere stap in een berekening met een Turingmachine T overeenkomt met de toepassing van een primitieve recursieve functie $Move_T$ op een configuration number m van T. Hiertoe wordt ook een nieuw tape number berekend: $NewTapeNumber(m)$.

   Druk $NewTapeNumber(m)$ uit in $TapeNumber(m)$, $Posn(m)$, $Symbol(m)$, $NewSymbol(m)$ en de functie $PrNo$. Je mag ervan uitgaan dat m een geldig configuration number is.

   Wellicht ten overvloede: de functie $PrNo(i)$ levert het i-de priemgetal op, ofwel: $PrNo(0) = 2, PrNo(1) = 3, PrNo(2) = 5, \ldots$.

`einde tentamen`