TENTAMEN FUNDAMENTELE INFORMATICA 3
Vrijdag 13 januari 2017, 14.00 - 17.00 uur

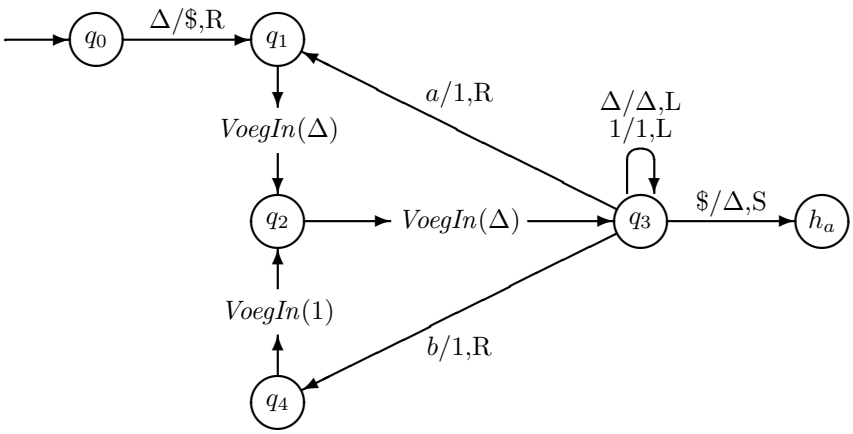
Dit tentamen bestaat uit zes opgaven, waarbij steeds tussen [en] staat hoeveel punten er ongeveer mee te verdienen zijn. In totaal zijn er 100 punten te verdienen.
Geef de gevraagde Turingmachines door middel van hun transitiediagram.
Wanneer er bij een vraag om uitleg of toelichting gevraagd wordt, 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. [16 pt] In het boek wordt soms bij de opbouw van Turingmachines gebruik gemaakt van een component *Insert*. In deze opgave gebruiken we een variant *VoegIn*, die in principe hetzelfde doet, maar de leeskop een positie verder naar rechts achterlaat: *VoegIn*(σ) verandert de tape-inhoud van $y\underline{z}$ in $y\sigma\underline{z}$ (waarbij z geen Δ bevat).

   (a) Teken de component *VoegIn*, dat wil zeggen: teken een Turingmachine die het hierboven beschreven effect heeft. Je mag ervan uitgaan dat de string z op de tape alleen letters a en b bevat. Het symbool σ dat op de tape wordt gezet, kan elk mogelijk tapesymbool zijn. Desgewenst mag je nieuwe tapesymbolen (hulpsymbolen) introduceren.

   Beschouw nu onderstaande Turingmachine T_0.



   (b) Geef de complete berekening van Turingmachine T_0 voor invoer ab.

   Je mag een subberekening uitgevoerd door de component *VoegIn* beschrijven als één stap. Het begin van de berekening ziet er dus als volgt uit:

$$q_0\Delta ab \vdash \$q_1ab \vdash \$\Delta q_2ab$$

   Wanneer T_0 het lusje doorloopt op toestand q_3, vat die stappen dan samen met behulp van $\vdash^*$.

   De berekening zal niet meer dan vijftien stappen vergen.

   (c) Beredeneer wat het effect van Turingmachine T_0 is voor een willekeurige string $x \in \{a, b\}^*$. Beschrijf de eindconfiguratie als T_0 begint in $q_0\Delta x$.

---

2. [24 pt]

   (a) Laat T een gewone (deterministische) Turingmachine (TM) zijn, met invoeralfabet $\{a, b\}$ en $L(T) = L$. Construeer een niet-deterministische Turingmachine (NTM) T' die de taal L' van alle 'verlengingen' van woorden in L accepteert:

   $$L' = \{x \in \{a, b\}^* \mid x = yz \text{ voor zekere } y \in L \text{ en } z \in \{a, b\}^*\}$$

   Je mag voor T' gebruik maken van component T. Andere componenten mag je alleen gebruiken als je ze zelf ook tekent.
   Leg ook uit waarom T' inderdaad L' accepteert.

In Stelling 7.31 van het boek wordt bewezen dat 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. T_2 gaat naar h_a zodra hij een pad van T_1 tegenkomt dat naar h_a leidt. Tijdens het simuleren van alle mogelijke paden/bitstrings van T_1 bevat

- de eerste tape van T_2 steeds de invoer x
- de tweede tape van T_2 de bitstring die gesimuleerd wordt
- de derde tape de inhoud van de tape van T_1 tijdens het gevolgde pad, voorafgegaan door een speciaal \$ symbool; voordat we een pad/bitstring gaan simuleren, kopiëren we daarom de inhoud van de eerste tape naar deze tape,
- de vierde tape de bitstrings waarbij T_1 tijdens het simuleren naar h_r ging, van de tape afliep of anderszins crashte.

   (b) Wat is de functie van het speciale symbool \$ op de derde tape van T_2? En *wanneer* komen we dat symbool tegen?

   (c) Wat is de functie van de bitstrings op de vierde tape van T_2? En *wanneer* bekijken we ze, en waar kijken we dan specifiek naar?

---

3. [20 pt of 16 pt]

   (a) Wanneer noemen we een *unrestricted* grammatica $G = (V, \Sigma, S, P)$ context-gevoelig?

   (b) Kies een van de volgende twee onderdelen (b1) of (b2), die feitelijk alleen verschillen in de taal L:

   (b1) [18 pt] Laat de taal L gedefinieerd zijn door

   $$L = \{a^i b^{i+k} a^k \mid i, k \geq 0 \text{ met } i < k\}$$

      i. Geef de eerste vijf elementen van L in de canonieke volgorde.

ii. Geef een context-gevoelige grammatica G, zó dat $L(G) = L$. Leg uit wat de functie is van de diverse variabelen en producties in G. Als het niet lukt om een context-gevoelige grammatica voor L te bedenken, kun je het grootste deel van de punten verdienen met een unrestricted grammatica voor de taal, met de bijbehorende uitleg.

iii. Geef een afleiding in G van het woord *abbbaa*. Wanneer je in de afleiding een aantal 'gelijksoortige' producties achter elkaar toepast, vat die stappen dan samen met behulp van $\Rightarrow^*$.

(b2) [14 pt] Laat de taal L gedefinieerd zijn door

$$L = \{a^i b^{i+k} c^k \mid i, k \geq 0 \text{ met } i < k\}$$

i. Geef de eerste vijf elementen van L in de canonieke volgorde.

ii. Geef een context-gevoelige grammatica G, zó dat $L(G) = L$. Leg uit wat de functie is van de diverse variabelen en producties in G. Als het niet lukt om een context-gevoelige grammatica voor L te bedenken, kun je het grootste deel van de punten verdienen met een unrestricted grammatica voor de taal, met de bijbehorende uitleg.

iii. Geef een afleiding in G van het woord *abbbcc*. Wanneer je in de afleiding een aantal 'gelijksoortige' producties achter elkaar toepast, vat die stappen dan samen met behulp van $\Rightarrow^*$.

---

4. [6 pt] Stel dat L een recursief opsombare taal is over een alfabet Σ, en dat T een Turingmachine is met $L(T) = L$. Dan is L niet per se recursief. Als L niet recursief is, moeten er strings $x \in \Sigma^*$ kunnen zijn waarvoor T oneindig loopt. Stelling 8.3 uit het boek zegt namelijk dat als een taal wordt geaccepteerd door een Turingmachine die stopt voor iedere invoer, dat die taal dan recursief is.

   Toon aan dat als L inderdaad niet recursief is, dat er dan **oneindig veel** strings x zijn waarvoor T oneindig loopt.

---

5. [17 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 nu de eigenschap dat een Turingmachine de lege string accepteert in een even aantal stappen. Toon **met een concreet tegenvoorbeeld** aan dat deze eigenschap geen niet-triviale taaleigenschap is,

   (c) Vanwege het voorgaande is de stelling van Rice niet van toepassing op het beslissingsprobleem

   > P_1: gegeven een Turingmachine T, accepteert T de lege string in een even aantal stappen?

Toon aan dat P_1 desondanks niet beslisbaar is, door

- een ander beslissingsprobleem P_2 te beschrijven waarop de stelling van Rice wel van toepassing is (als je P_2 beschreven hebt, is het niet nodig te beredeneren dat de stelling van Rice inderdaad op P_2 van toepassing is),
- vervolgens een reductie te beschrijven tussen P_1 en P_2; 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 P_1 en een beslissingsprobleem P_2 weet, kun je een deel van de punten daarvan verdienen door uit te leggen hoe je voor algemene beslissingsproblemen P en P' aantoont dat $P \leq P'$.

---

6. [17 pt] 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.

   Het Gödelnummer van een rij natuurlijke getallen $(x_0, x_1, x_2, \ldots, x_{m-1})$ is gedefinieerd als

   $$gn(x_0, x_1, x_2, \ldots, x_{m-1}) = 2^{x_0} 3^{x_1} 5^{x_2} \ldots PrNo(m-1)^{x_{m-1}}$$

   In feite is de Gödelnummer functie een verzameling van allemaal verschillende Gödelnummer functies: een functie gn^1 met één argument x_0, een functie gn^2 met twee argumenten x_0, x_1, een functie gn^3 met drie argumenten x_0, x_1, x_2, enzovoort.

   We gaan nu aantonen dat al deze verschillende functies primitief recursief zijn. Je mag daarbij gebruiken dat de vermenigvuldiging van twee getallen primitief recursief is.

   (a) Toon aan dat de functie gn^1 met één argument, gedefinieerd door $gn^1(x_0) = 2^{x_0}$ primitief recursief is.

   (b) We gaan nu voor willekeurige $m \geq 1$ kijken naar twee opeenvolgende Gödelnummer functies gn^m (met m argumenten) en gn^{m+1} (met $m+1$ argumenten).

      i. Druk voor een willekeurige rij natuurlijke getallen $(x_0, x_1, x_2, \ldots, x_m)$ de functiewaarde $gn^{m+1}(x_0, x_1, x_2, \ldots, x_m)$ uit in $gn^m(x_0, x_1, x_2, \ldots, x_{m-1})$ en x_m.

      ii. Toon aan dat als de functie gn^m primitief recursief is, dat dan ook de functie gn^{m+1} primitief recursief is.

`einde tentamen`