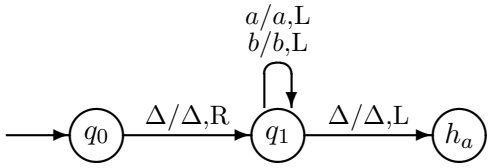
HERTENTAMEN COMPUTABILITY
Vrijdag 16 juni 2023, 09.00 - 12.00 uur

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. Wanneer er bij een vraag om uitleg, motivatie of toelichting gevraagd wordt, is het belangrijk om die ook te geven.

---

1. [6 pt] Beschouw onderstaande Turing machine T met invoeralfabet $\{a, b\}$:



   Wat is, voor deze concrete Turingmachine T, de taal $L(T)$? Motiveer je antwoord.

---

2. [33 pt] Laat $L = \{a^i b^{i+1} a^{i+2} \mid i \geq 0\}$.

   (a) Geef de eerste drie elementen in de canonieke volgorde van L.

   (b) Teken een gewone (deterministische, 1-tape) Turingmachine T_1, zó dat $L(T_1) = L$. T_1 krijgt dus een willekeurige string $x \in \{a, b\}^*$ als invoer, en accepteert deze dan en slechts dan als $x \in L$.

   Als je voor T_1 gebruik wilt maken van componenten, zul je die componenten ook moeten uitwerken (tekenen dus).

   Leg ook duideljk uit hoe T_1 werkt.

   (c) Een 1-tape Turingmachine voor taal L vereist voor een invoer x al gauw een aantal stappen dat kwadratisch is in de lengte van x. Teken een **2-tapes** Turingmachine T_2 die in een **lineair** aantal stappen bepaalt of een invoer $x \in \{a, b\}^*$ wel of niet in L zit. In het eerste geval moet T_2 (uiteraard) accepteren, in het tweede geval (impliciet of expliciet) verwerpen of crashen.

   Als je voor T_2 gebruik wilt maken van componenten, moeten dat wel 2-tapes componenten zijn, en zul je ze ook moeten uitwerken (tekenen dus).

   Leg ook duidelijk uit hoe T_2 werkt.

---

3. [16 pt] Stel dat G_1 een unrestricted grammar is, met startvariabele T, zó dat $L(G_1) = L_1$ voor zekere taal $L_1 \subseteq \{a, b\}^*$.

   Laat nu G_2 een unrestricted grammar zijn die alle variabelen en producties van G_1 bevat, en daarbovenop drie extra variabelen S, E en F (met S als startvariabele van G_2) en de volgende extra producties:

   $$S \to ET \qquad Ea \to aE \qquad Eb \to bE \qquad E \to F \qquad Fa \to F \qquad Fb \to F \qquad F \to \Lambda$$

   (a) Beredeneer dat G_2 elke string $x \in L_1$ waaruit een substring y is weggehaald kan genereren.
   Om precies te zijn: als $x \in L_1$ en $x = x_1 y x_2$ voor zekere $x_1, y, x_2 \in \{a, b\}^*$, dan is $x_1 x_2 \in L(G_2)$.
   Verwijs in je redenering naar specifieke producties in G_2 die je gebruikt in de afleiding.

   (b) Het is niet per se zo dat G_2 *alleen* de hierboven beschreven strings $x_1 x_2$ kan genereren. Het is namelijk mogelijk dat in G_2, door het weghalen van de substring y, producties uit G_1 kunnen worden toegepast die in G_1 zelf niet konden worden toegepast. Illustreer dit door

   - een eenvoudige grammatica G_1 te geven waarbij dit het geval is,
   - een string $z \in \{a, b\}^*$ te geven die in G_2 gegenereerd kan worden, terwijl z niet van de vorm $x_1 x_2$ is voor een string $x_1 y x_2 \in L(G_1)$,
   - een afleiding voor z in G_2 te geven.

---

4. [12 pt] In het bewijs van Stelling 8.13 in het boek wordt beschreven hoe je bij een willekeurige *unrestricted grammar* 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 voor een invoer x bestaat uit drie fases:

   1. MovePastInput
   2. Simuleer op de tape een willekeurige afleiding in G
   3. Vergelijk invoer x met de bij stap 2 afgeleide string y. Accepteer, dan en slechts dan als x gelijk is aan y.

   Laat $G = (V, \Sigma, S, P)$ de unrestricted grammar zijn met $V = \{S, B, C\}$, $\Sigma = \{a, b, c\}$ en de volgende producties:

   $$S \to aBCS \mid \Lambda$$
   $$Ba \to aB \qquad Ca \to aC \qquad CB \to BC \qquad B \to b \qquad C \to c$$

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

---

5. [20 pt] Laat P_1 een beslissingsprobleem zijn. Om P_1 op te lossen met een Turingmachine, moeten de instanties van P_1 gecodeerd worden als strings over een alfabet Σ_1, met behulp van een coderingsfunctie e_1.

   (a) Wanneer noemen we coderingsfunctie e_1 een *redelijke* coderingsfunctie? Geef de drie eisen waaraan e_1 dan moet voldoen.
   Als je het antwoord op dit onderdeel niet weet, dan kun je het 'kopen' bij de docent. Wellicht kun je dan wel onderdeel (b) maken.

   We noemen een taal L recursief als er een Turingmachine bestaat die de karakteristieke functie χ_L van L berekent.

   Laat P_1 nu inderdaad een beslissingsprobleem zijn, met een redelijke coderingsfunctie e_1, die strings over een alfabet Σ_1 produceert.

   We noemen P_1 beslisbaar als $Y(P_1)$, de verzameling van gecodeerde ja-instanties van P_1, een recursieve taal is.

   Laat P_2 een ander beslissingsprobleem zijn, met een redelijke coderingsfunctie e_2, die strings over een alfabet Σ_2 produceert.

   We zeggen dat P_1 reduceerbaar is naar P_2 (genoteerd als $P_1 \leq P_2$), wanneer

   - er een algoritme bestaat,
   - dat elke instantie I_1 van P_1 omzet in een instantie I_2 van P_2,
   - zó dat I_1 een ja-instantie van P_1 is, dan en slechts dan als I_2 een ja-instantie van P_2 is.

   (b) Toon aan met behulp van bovenstaande definities, dat als P_2 beslisbaar is en $P_1 \leq P_2$, dat dan ook P_1 beslisbaar is.
   Als het je niet lukt om een formeel bewijs te geven met behulp van bovenstaande definities, kun je een deel van de punten verdienen door intuïtief te beredeneren dat als P_2 beslisbaar is en $P_1 \leq P_2$, dat dan ook P_1 beslisbaar is.

---

6. [13 pt] Beschouw de volgende twee beslissingsproblemen:

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

   EmptyTapeAfter-Λ:
   Gegeven een Turingmachine T_2, halt T_2 voor invoer Λ met (op het moment van halten) een lege tape?

   Wellicht ten overvloede: als T_2 wel halt voor invoer Λ, maar de tape is niet leeg op het moment van halten, dan is T_2 een nee-instantie van *EmptyTapeAfter*-Λ.

   Toon aan dat *Accepts*-$\Lambda \leq$ *EmptyTapeAfter*-Λ. Doe dit door een reductie tussen de twee beslissingsproblemen te beschrijven. Laat uiteraard ook zien dat aan alle eisen van een reductie is voldaan.

`einde tentamen`