TENTAMEN COMPUTABILITY
Donderdag 27 maart 2025, 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. [22 pt] Laat

$$L = \{x \in \{a, b, c\}^* \mid n_a(x) < n_b(x) \text{ en } n_a(x) < n_c(x)\}$$

Dus bijvoorbeeld $cb, cabbc, aabbbcccc \in L$, maar $cbac \notin L$ omdat $n_a(cbac) = n_b(cbac)$.

Teken een gewone (deterministische, 1-tape) Turingmachine T, zó dat $L(T) = L$.

Je mag voor T gebruik maken van de componenten NB en PB, zoals die in het boek beschreven zijn. Andere componenten mag je alleen gebruiken als je ze zelf uitwerkt (tekent dus). Wellicht ten overvloede:

- NB verplaatst de leeskop naar de eerste Δ rechts van de huidige positie,
- PB verplaatst de leeskop (zo mogelijk) naar de eerste Δ links van de huidige positie.

Leg ook duidelijk uit hoe T werkt.

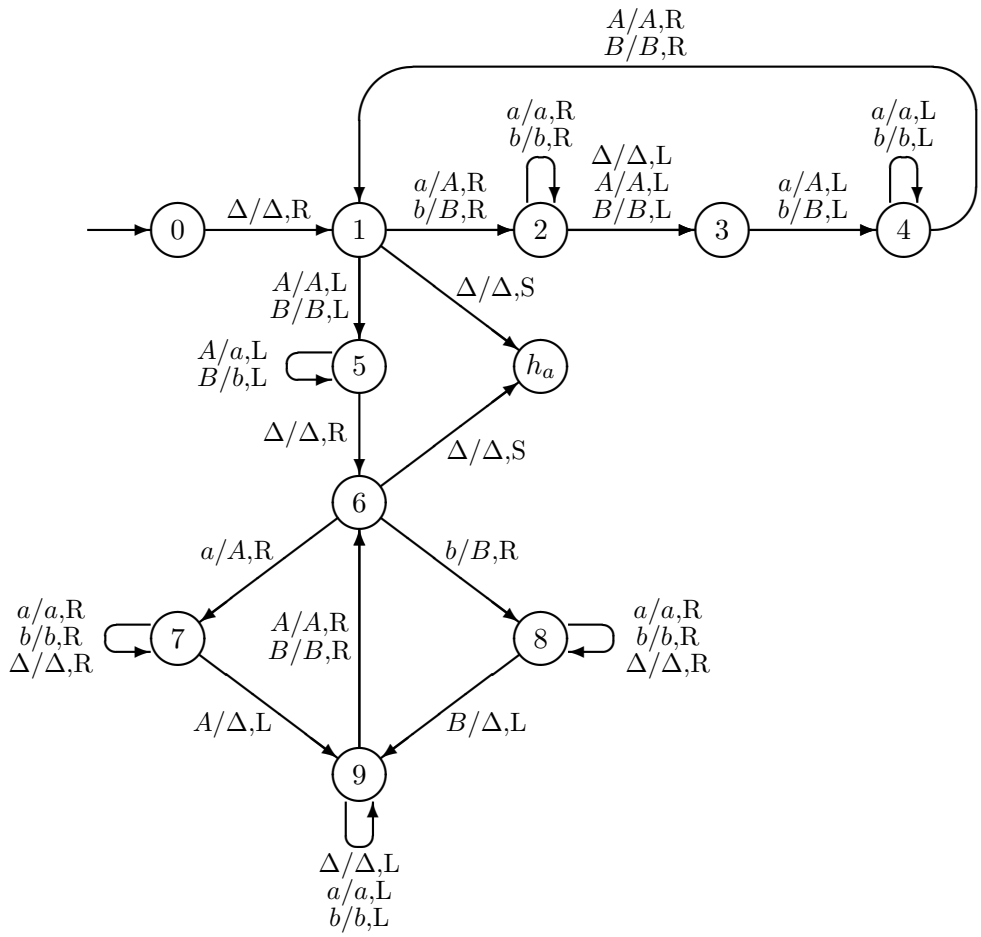
---

2. [16 pt] Laat $L \subseteq \Sigma^*$ een taal zijn. De karakteristieke functie $\chi_L : \Sigma^* \to \{0, 1\}$ wordt gedefinieerd door

$$\chi_L(x) = \begin{cases} 1 & \text{als } x \in L \\ 0 & \text{als } x \notin L \end{cases}$$

Z.O.Z.

Beschouw nu de volgende Turingmachine T_1:



Er geldt dat

$$L(T_1) = XX = \{ss \mid s \in \{a,b\}^*\}$$

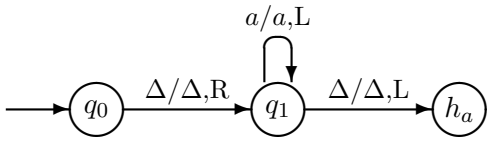
T_1 werkt door (1) van buiten naar binnen kleine letters te markeren als hoofdletters en zo het midden van de invoer te bepalen, (2) de linker helft van de invoer weer terug te veranderen in kleine letters, en (3) herhaaldelijk de eerstvolgende letter van de linker helft te vergelijken met de eerstvolgende letter van de rechter helft, waarbij de bekeken letters van de linker helft weer hoofdletters worden, terwijl de bekeken letters van de rechter helft worden vervangen door Δ.

Pas T_1 aan, zó dat de resulterende Turingmachine T_2 de karakteristieke functie van XX berekent. Als je hierbij transities van T_1 aanpast, beschrijf of teken die aanpassing dan precies. Als je toestanden en transities toevoegt aan T_1, teken die dan. Desgewenst mag je ook heel T_2 tekenen.

Je mag voor de aanpassingen gebruik maken van de componenten *NB* en *PB*, zoals die in het boek beschreven zijn (zie opgave 1). Andere componenten mag je alleen gebruiken als je ze zelf uitwerkt (tekent dus).

3. [10 pt] De invoer van een universele Turingmachine bestaat o.a. uit een codering van een Turingmachine T. Dit is een string over $\{0, 1\}$.

   Beschouw onderstaande Turingmachine T:



   Geef een codering $e(T)$ van T, waarbij je de functie e uit het boek gebruikt. Geef duidelijk aan, welk onderdeel van de Turingmachine je waarmee codeert. Vermeld ook expliciet de gebruikte codering voor de individuele toestanden, tapesymbolen en richtingen.

---

4. [20 pt] Laat

$$L = \{a, abaa, abaabaaa, abaabaaabaaaa, \ldots\}$$

   Ofwel, een string $x \in L$ bestaat uit één of meerdere reeksen a's gescheiden door steeds een b, waarbij de eerste reeks maar één a kent, en elke volgende reeks a's één a langer is. Je kunt ook zeggen dat een string $x \in L$ van de vorm $a^1ba^2ba^3b \ldots a^k$ is, met $k \geq 1$.

   Bij de huiswerkopgave vroegen we om een Turingmachine voor deze taal te construeren. Bij dit tentamen vragen we: geef een *unrestricted grammar* G, zó dat $L(G) = L$.

   Leg ook uit wat de functie is van de diverse variabelen en producties in je grammatica G.

---

5. [16 pt] We noemen een taal $L \subseteq \Sigma^*$ *recursief* als er een Turingmachine bestaat die de karakteristieke functie χ_L berekent (voor de definitie van de karakteristieke functie, zie opgave 2).

   (a) Laat L_1 en L_2 twee recursieve talen zijn. Toon aan dat ook $L_1 \cap L_2$ recursief is, door (duidelijk) de werking van een Turingmachine T te beschrijven die de karakteristieke functie van $L_1 \cap L_2$ berekent. Een beschrijving op het niveau van functionaliteit is voldoende. Je hoeft niet alle toestanden en transities te tekenen.

   De klasse van recursieve talen is dus gesloten onder doorsnede. De klasse van recursieve talen is ook gesloten onder complement, dus als een taal L recursief is, is zijn complement L' dat ook.

   Laat nu P een beslissingsprobleem zijn. Om P op te lossen met een Turingmachine, moeten de instanties van P gecodeerd worden als strings over een alfabet Σ, met behulp van een coderingsfunctie e. We noemen e een *redelijke* coderingsfunctie, als

   - het beslisbaar is of een string $x \in \Sigma^*$ een gecodeerde instantie is (d.w.z.: of $x = e(I)$ voor zekere instantie I van P), en
   - e injectief is, en

- als voor $x \in \Sigma^*$ geldt dat $x = e(I)$ voor zekere instantie I van P, dan is x algoritmisch te decoderen naar I.

Laat P nu inderdaad een beslissingsprobleem zijn, met een redelijke coderingsfunctie e, die strings over een alfabet Σ produceert.

(b) We noemen P beslisbaar als $Y(P)$, de verzameling van gecodeerde ja-instanties van P, een recursieve taal is.
Toon aan, met behulp van bovenstaande definities en resultaten, dat als $Y(P)$ een recursieve taal is, dat dan ook $N(P)$, de verzameling van gecodeerde nee-instanties van P, een recursieve taal is.

---

6. [16 pt] Beschouw de volgende drie beslissingsproblemen:

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

AcceptsSomething:
Gegeven een Turingmachine T_2, met invoeralfabet Σ, is er een string $x \in \Sigma^*$ die door T_2 geaccepteerd wordt?

AcceptsSomethingEvenSteps:
Gegeven een Turingmachine T_3, met invoeralfabet Σ, is er een string $x \in \Sigma^*$ die door T_3 in een even aantal stappen geaccepteerd wordt?

Een stap in de definitie van *AcceptsSomethingEvenSteps* is het één maal volgen van één transitie.

Er is gegeven dat *Accepts-*Λ niet beslisbaar is. Toon aan dat ook *AcceptsSomethingEvenSteps* niet beslisbaar is, met behulp van een reductie met *Accepts-*Λ. Laat uiteraard ook zien dat aan alle eisen van een reductie is voldaan, en vergeet niet om de conclusie te trekken.

Als het je niet lukt om aan te tonen dat AcceptsSomethingEvenSteps *niet beslisbaar is, mag je ook aantonen dat* AcceptsSomething *niet beslisbaar is, met behulp van een reductie met* Accepts-Λ. *Je verliest dan 3 van de 16 punten.*

`einde tentamen`