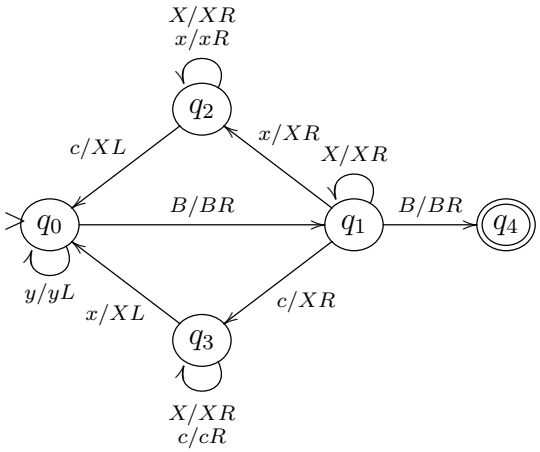
Berekenbaarheid 2016
Uitwerkingen Tentamen
26 januari 2017

1. Geef een standaard Turing-machine M_1 die de volgende taal herkent door eindtoestand:

$$L_1 := \{w \in \{a, b, c\}^* \mid |w|_a + |w|_b = |w|_c\}$$

Hierin is $|w|_a$ een notatie voor het aantal a's in het woord w. Er geldt bijv. $cbccbaccabcbca \in L_1$, want dit woord bevat 3 a's, 4 b's en 7 c's, en $3+4=7$.

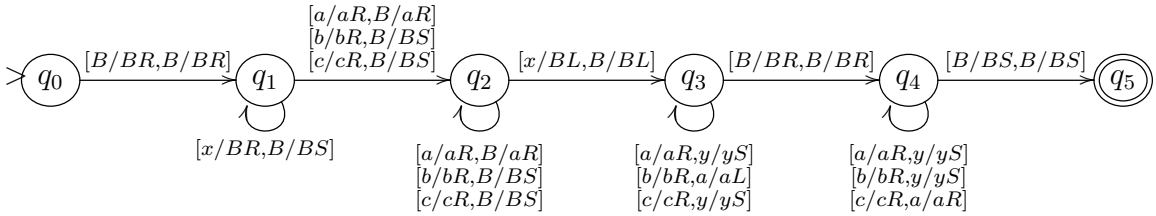


$x \in \{a, b\}$, $y \in \{a, b, c, X\}$

2. Geef een non-deterministische 2-tape Turing-machine M_2 die de volgende taal herkent door eindtoestand:

$$L_2 := \{u_1 v u_2 \mid u_1, u_2, v \in \{a, b, c\}^*,\ v \neq \lambda \text{ en } |v|_a = |v|_b = |v|_c\}$$

Zorg ervoor dat een woord $w \in L_2$ wordt herkend in hoogstens $3|w| + 4$ stappen, waarin $|w|$ de lengte van w is.

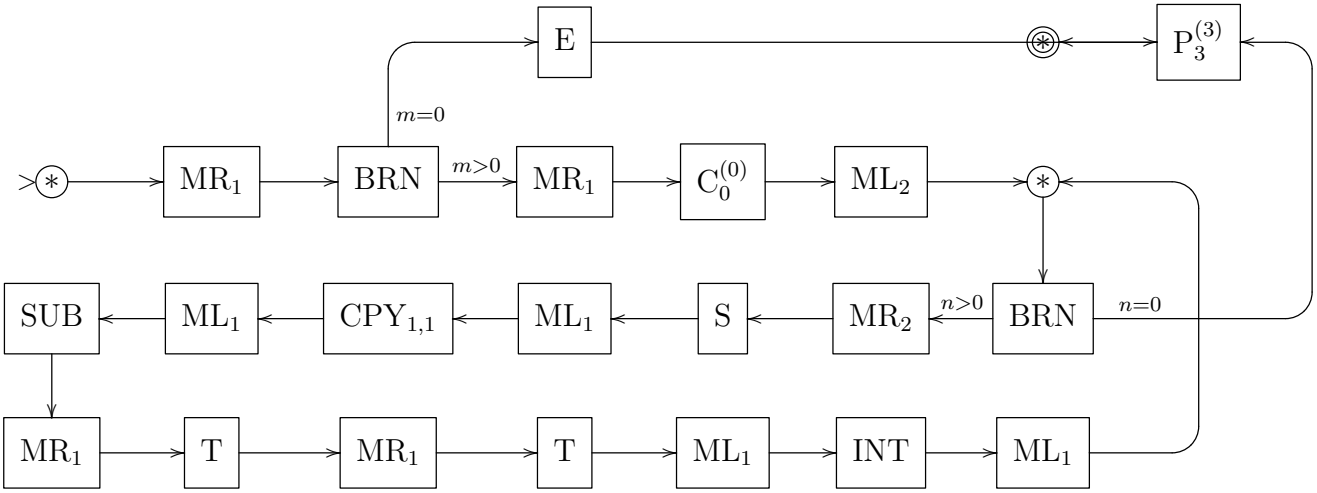


$x \in \{a, b, c, B\}$, $y \in \{a, B\}$

3. Geef een numerieke Turing-machine M_3 die de functie f_3 uitrekent die gegeven is door:

$$\begin{aligned} f_3(n,m) &= \lceil n/m \rceil && \text{als } m \neq 0 \\ f_3(n,m)\uparrow && \text{als } m = 0 \end{aligned}$$

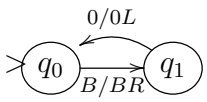
Hierin betekent $\lceil\ \rceil$ afronden naar boven. Er geldt dus bijv. dat $f_3(1,4) = \lceil \frac{1}{4} \rceil = 1$. Je mag gebruik maken van de macro's op pagina 7 van dit tentamen.



4. Geef de code $R(M_4)$ van een deterministische Turing-machine M_4 waarvoor geldt dat $U(R(M_4)R(M_4))\uparrow$, waarbij U een universele Turing-machine is. Verklaar je antwoord. Zie pagina 8 voor een relevant citaat uit het boek van Sudkamp.

Een universele Turing-machine heeft de eigenschap dat $U(R(M)w) = M(w)$ voor alle codes $R(M)$ en inputs w. Er geldt hier dus $U(R(M_4)R(M_4)) = M_4(R(M_4))$, en dus moet M_4 niet termineren (of abnormaal termineren) met als input een eigen code.

Dit geldt bijvoorbeeld als we voor M_4 nemen:



want een code van een Turing-machine begint altijd met een nul, en met zo'n input termineert deze machine omdat hij steeds heen en weer zal gaan tussen de blank aan het begin van de tape en deze eerste nul.

Een code voor deze machine is bijvoorbeeld:

$$R(M_4) = \mathbf{000}10111011011101\mathbf{00}1101010101\mathbf{000}$$

5. Is het volgende probleem P_5 onbeslisbaar?

Input: Een code $R(M)$ van een Turing-machine M.

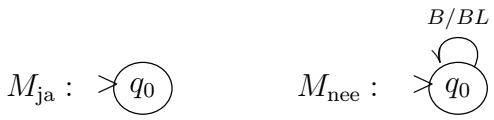
Vraag: Bevat de taal $L(M)$ een code $R(M')$ van een Turing-machine M' waarvoor geldt dat $L(M')$ recursief opsombaar is?

Zo ja, laat zien dat dit zo is. Zo nee, verklaar waarom dit niet zo is.

Ja, het probleem P_5 is onbeslisbaar.

Dit probleem valt onder de stelling van Rice, want de eigenschap is een eigenschap van de taal $L(M)$. In feite is alles wat gevraagd wordt of deze taal een correcte code van een Turing machine bevat, want de taal van een Turing machine is per definitie recursief opsombaar, dus de toevoeging 'waarvoor geldt dat $L(M')$ recursief opsombaar is' voegt niets toe.

Om de stelling van Rice te mogen gebruiken moeten we alleen nog laten zien dat het probleem niet triviaal is. Dit volgt uit het bestaan van machines M_{ja} en M_{nee}, waarbij $L(M_{\text{ja}}) = \{0,1\}^*$ en $L(M_{\text{nee}}) = \emptyset$. Neem hiervoor bijv. de machines:



6. Laat zien dat het volgende probleem P_6 onbeslisbaar is:

Input: Een code $R(M)$ van een Turing-machine M.

Vraag: Bevat de taal $L(M)$ de code $R(M)$?

Dit probleem valt niet onder de stelling van Rice, want twee machines die dezelfde taal herkennen kunnen twee verschillende codes hebben, waarbij de ene wel in die taal zit en de andere niet. Dus in dat geval zijn de talen hetzelfde, maar het antwoord van het probleem verschillend. En dus vraagt het probleem niet naar een eigenschap van die taal.

We laten zien dat P_6 onbeslisbaar is door het blank-tape-probleem B te reduceren naar het probleem P_6.

Hiertoe moeten we een willekeurige code $R(M)$ omzetten in een code $R(M')$ zodat dan geldt:

$$R(M)\downarrow \iff R(M') \in L(M')$$

Een constructie die dit doet is door M' te laten zijn:

- Wis de tape.
- M

Want in dat geval stopt M' voor *iedere* input (dus ook met zijn eigen code) als M stopt met de lege tape, en stopt M' voor geen enkele input (dus ook niet met zijn eigen code) als M niet stopt met de lege tape.

Als P_6 beslisbaar zou zijn, konden we dus het blank-tape-probleem B beslissen door bij een machine M waarvan we willen weten of deze stopt op de blanco tape de M' te construeren en die aan P_6 te geven. Maar we weten dat B onbeslisbaar is, en dus is P_6 ook onbeslisbaar.

7. Leg uit hoe de verzameling van Turing-berekenbare functies en de verzameling van μ-recursieve functies gedefinieerd zijn (waarbij je Turing-machines bekend mag veronderstellen), en leg uit wat de relatie tussen deze twee verzamelingen is.

   De Turing-berekenbare functies zijn de numerieke functies die te berekenen zijn met een Turing machine.

   De μ-recursieve functies zijn precies de numerieke functies die je kunt maken als:

   - Een basisfunctie: de successorfunctie s, de nulfunctie $c_0^{(0)}$, en de projectiefuncties $p_k^{(n)}$ voor iedere $1 \leq k \leq n$.
   - De functie $f \circ (g_1, \ldots, g_n)$ gedefinieerd met compositie uit eerder gemaakte μ-recursieve functies f, g_1, $\ldots$, g_n.
   - De functie $\mathbf{primrec}(g, h)$ gedefinieerd met primitieve recursie uit eerder gemaakte μ-recursieve functies g en h.
   - De functie f gedefinieerd met onbegrensde minimalisatie (μ-recursie) uit een eerder gemaakte μ-recursieve functie g, dus als:

   $$f(x_1, \ldots, x_n) = \mu y.\, g(x_1, \ldots, x_n, y)$$

   Deze twee verzamelingen numerieke functies zijn identiek.

8. Geef numerieke functies f_8, g_8 en h_8 zodat:

   $$f_8 \circ (g_8, h_8) = \mathsf{add}$$
   $$g_8 \circ (h_8, f_8) = \mathsf{add}$$
   $$h_8 \circ (f_8, g_8) = \mathsf{add}$$

   Zie pagina 8 voor de definitie van de functie add. Verklaar je antwoord.

   $$f_8 = \mathsf{add}$$
   $$g_8 = p_2^{(2)}$$
   $$h_8 = p_1^{(2)}$$

$$
\begin{aligned}
[f_8 \circ (g_8, h_8)](x, y) &= f_8(g_8(x, y), h_8(x, y)) \\
&= \mathsf{add}(p_2^{(2)}(x, y), p_1^{(2)}(x, y)) \\
&= \mathsf{add}(y, x) \\
&= y + x \\
&= x + y \\
&= \mathsf{add}(x, y)
\end{aligned}
$$

$$
\begin{aligned}
[g_8 \circ (h_8, f_8)](x, y) &= g_8(h_8(x, y), f_8(x, y)) \\
&= p_2^{(2)}(h_8(x, y), f_8(x, y)) \\
&= f_8(x, y) \\
&= \mathsf{add}(x, y)
\end{aligned}
$$

$$
\begin{aligned}
[h_8 \circ (f_8, g_8)](x, y) &= h_8(f_8(x, y), g_8(x, y)) \\
&= p_1^{(2)}(f_8(x, y), g_8(x, y)) \\
&= f_8(x, y) \\
&= \mathsf{add}(x, y)
\end{aligned}
$$

9. De 'toren van machten' functie f_9 is gedefinieerd door de recursievergelijkingen:

$$
\begin{aligned}
f_9(x, 0) &= 1 \\
f_9(x, y + 1) &= x^{f_9(x,y)}
\end{aligned}
$$

Er geldt bijv. $f_9(2, 4) = 65536$. Geef functies g_9 en h_9 zodat:

$$f_9 = \mathsf{primrec}(g_9, h_9)$$

en schrijf deze functies als compositie van functies op pagina 8.

De recursievergelijkingen voor deze definitie met primitieve recursie zijn:

$$
\begin{aligned}
f_9(x, 0) &= g_9(x) \\
f_9(x, y + 1) &= h_9(x, y, f_9(x, y))
\end{aligned}
$$

Dus we willen dat:

$$
\begin{aligned}
g_9(x) &= 1 \\
h_9(x, y, f_9(x, y)) &= x^{f_9(x,y)}
\end{aligned}
$$

Dit bereiken we door te definiëren:

$$
\begin{aligned}
g_9(x) &= 1 \\
h_9(x, y, w) &= x^w
\end{aligned}
$$

En als compositie geschreven is dat:

$$g_9 = c_1^{(1)}$$
$$h_9 = \mathsf{exp} \circ (p_1^{(3)}, p_3^{(3)})$$

Met andere woorden, de definitie van f_9 in termen van functies uit de lijst op pagina 8 is:

$$f_9 = \mathbf{primrec}\big(c_1^{(1)}, \mathsf{exp} \circ (p_1^{(3)}, p_3^{(3)})\big)$$

10. Laat zien dat er primitief recursieve functies f_{10} en g_{10} bestaan zodat:

$$f_{10}(2^{x+1}3^{y+1}) = x$$
$$g_{10}(2^{x+1}3^{y+1}) = y$$

Je mag zelf weten wat deze functies doen met getallen die niet van de vorm $2^{x+1}3^{y+1}$ zijn. Er moet bijv. gelden dat $f_{10}(3888) = 3$ en $g_{10}(3888) = 4$, want $2^{3+1}3^{4+1} = 16 \cdot 243 = 3888$. Je mag gebruiken dat de functies of pagina 8 primitief recursief zijn, en dat begrensde operatoren toegepast op expressies met alleen primitief recursieve functies weer primitief recursieve functies definiëren.

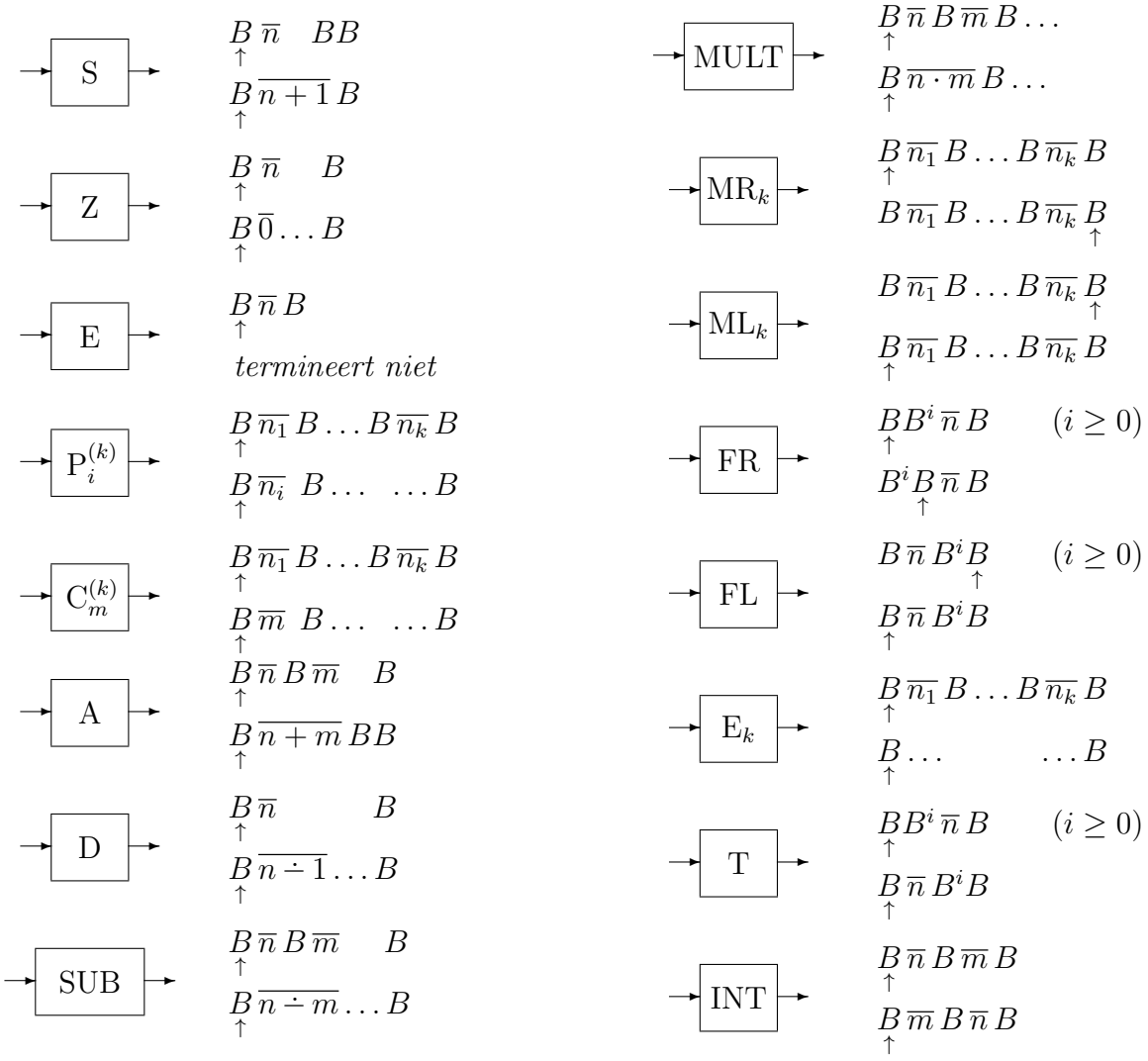
Deze functies zijn te definiëren met de volgende vergelijkingen:

$$f_{10}(w) = \sum_{x=0}^{w} \sum_{y=0}^{w} \big(x \cdot \mathsf{eq}(w, 2^{x+1}3^{y+1})\big)$$
$$g_{10}(w) = \sum_{x=0}^{w} \sum_{y=0}^{w} \big(y \cdot \mathsf{eq}(w, 2^{x+1}3^{y+1})\big)$$

Dit gebruikt dat uit $w = 2^{x+1}3^{y+1}$ volgt dat $x \leq w$ en $y \leq w$, en dat volgt uit het feit dat $x < 2^x$, dus zeker $x \leq 2^{x+1}$.

Omdat in de rechterkant alleen functies worden gebruikt waarvan we weten dat ze primitief recursief zijn en begrensde operatoren, volgt dat f_{10} en g_{10} ook primitief recursief zijn.

Macro's voor Turing-machines voor numerieke berekeningen



Codering van transities

Symbol	Encoding
0	1
1	11
B	111
q_0	1
q_1	11
$\vdots$	$\vdots$
q_n	1^{n+1}
L	1
R	11

Let $en(x)$ denote the encoding of a symbol x. A transition $\delta(q_i, x) = [q_j, y, d]$ is encoded by the string

$$en(q_i)0en(x)0en(q_j)0en(y)0en(d).$$

Primitief recursieve functies

$$\begin{aligned} \mathsf{id}(x) &= x \\ z(x) &= 0 \\ s(x) &= x+1 \\ p_i^{(k)}(x_1, \ldots, x_k) &= x_i \\ c_n^{(k)}(x_1, \ldots, x_k) &= n \end{aligned}$$

$$\begin{aligned} \mathsf{pred}(y) &= y \dot{-} 1 \\ \mathsf{add}(x,y) &= x + y \\ \mathsf{mult}(x,y) &= x \cdot y \\ \mathsf{sub}(x,y) &= x \dot{-} y \\ \mathsf{exp}(x,y) &= x^y \\ \mathsf{fact}(x) &= x! \\ \mathsf{sg}(x) &= \text{als } x \neq 0 \text{ dan 1 anders 0} \\ \mathsf{cosg}(x) &= \text{als } x \neq 0 \text{ dan 0 anders 1} \\ \mathsf{lt}(x,y) &= \text{als } x < y \text{ dan 1 anders 0} \\ \mathsf{gt}(x,y) &= \text{als } x > y \text{ dan 1 anders 0} \\ \mathsf{le}(x,y) &= \text{als } x \leq y \text{ dan 1 anders 0} \\ \mathsf{ge}(x,y) &= \text{als } x \geq y \text{ dan 1 anders 0} \end{aligned}$$

$$\begin{aligned} \mathsf{eq}(x,y) &= \text{als } x = y \text{ dan 1 anders 0} \\ \mathsf{ne}(x,y) &= \text{als } x \neq y \text{ dan 1 anders 0} \\ \mathsf{max}(x,y) &= \text{het maximum van } x \text{ en } y \\ \mathsf{min}(x,y) &= \text{het minimum van } x \text{ en } y \\ \mathsf{quo}(x,y) &= \text{als } y \neq 0 \text{ dan } \lfloor x/y \rfloor \text{ anders 0} \\ \mathsf{rem}(x,y) &= \text{als } y \neq 0 \text{ dan } x \bmod y \text{ anders } x \\ \mathsf{divides}(x,y) &= \text{als } y \neq 0 \text{ en } y \mid x \text{ dan 1 anders 0} \\ \mathsf{even}(x) &= \text{als } x \text{ even is dan 1 anders 0} \\ \mathsf{prime}(x) &= \text{als } x \text{ priem is dan 1 anders 0} \\ \mathsf{pn}(x) &= \text{het } x\text{-de priemgetal} \\ & \quad (\text{dus } \mathsf{pn}(0) = 2,\ \mathsf{pn}(1) = 3, \text{ etc.}) \end{aligned}$$