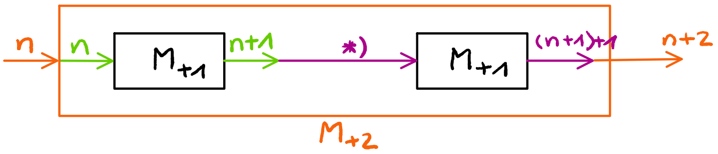
§8. Techniken zur Konstruktion von Turingmaschinen

Donnerstag, 18. Januar 2024 10:21

Ansatz: "kleine TM" benutzen, um komplexere TM zu konstruieren!

Bsp: TM, die succ' realisiert (Nachfolger in Dyadischen Darstellung)

Idee: Wir bezeichnen diese mit M_{+1} und verwenden die mehrfach

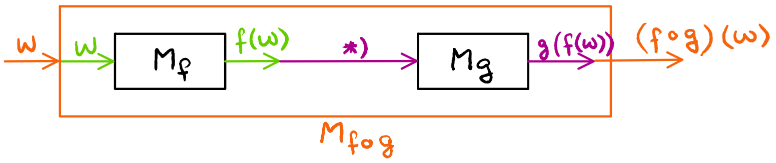


*) die normierte Finalkonfiguration der 1. TM ist normierte Startkonfiguration der 2. TM

a) Komposition von Turingmaschinen

Gegeben seien zwei TMen M_f und M_g, die die zwei (Wort-)Funktionen f und g berechnen

Idee:



formal: Gegeben seien $M_f = M_1$ und $M_g = M_2$ mit $M_i = (Q_i, \Sigma, T_i, \delta_i, q_{0i}, F_i$

Wir konstruieren die neue TM M wie folgt:

$$M = (Q, \Sigma, T, \delta, q_0, F)$$

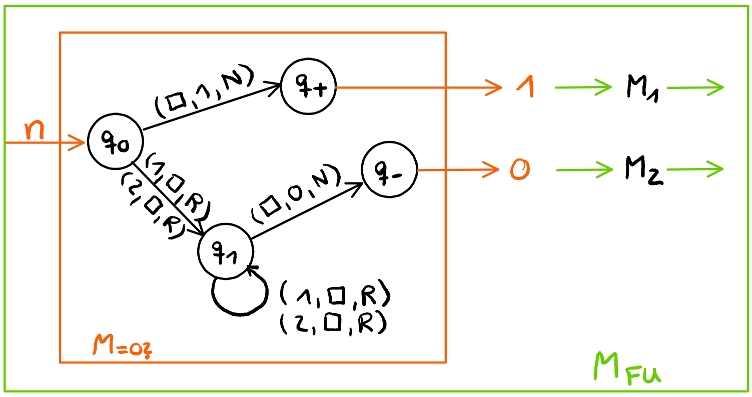
mit $Q =_{df} Q_1 \cup Q_2$, $T = T_1 \cup T_2$, $q_0 = q_{0_1}$, $F = F_2$

und $\delta =_{df} \delta_1 \cup \delta_2 \cup \{(q_F, x, q_{0_2}, x, N) \mid q_F \in F_1, x \in T_1\}$

→ M berechnet $f \circ g$

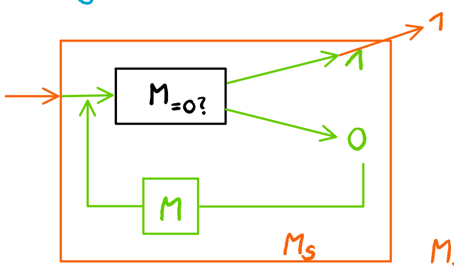
b) Turingmaschinen, die Fallunterscheidungen realisieren

Bsp: TM, die testet, ob ein Parameter $\boxed{= 0?}$ ist:



$M_{FU} = M_{vor}$; if $n = 0$ then M_1 else M_2

c) Turingmaschinen, die Schleifen realisieren



M_S = while $n \neq 0$ do M

d) Turingmaschinen, die Unterprogramme realisiert

Bsp: Es sei $M = (Q, \Sigma, \Gamma, \delta, q_0, F)$ eine TM mit einem Zustand q_{move}

In diesem Zustand wird ein "Unterprogramm" aufgerufen, das den gesamten aktuellen Bandinhalt von M in eine Zelle nach rechts verschiebt – und dann zurückgibt

→ Dieses Unterprogramm wird von folgender Maschine realisiert

$M_{move} = (Q_{move}, \Sigma, \Gamma, \delta_{move}, p_0, \{p_2\})$, $Q_{move} = \{p_0, p_1, p_2\} \cup \{p_x \mid x \in \Gamma \setminus \{\square\}\}$

δ_{move}	x	$\square$
p_0	$(p_x, \square, R)$	$(p_2, \square, N)$
p_y	(p_x, y, R)	(p_1, y, L)
p_1	(p_1, x, L)	$(p_2, \square, R)$

// $(x \in \Gamma \setminus \{\square\})$
$(y \in \Gamma \setminus \{\square\})$

p_2 übergibt an den "aktuellen" Zustand von M

THESE: Alles was Programme können, können auch Turingmaschinen!

Das bedeutet: Funktionen, die von Programmen berechnet werden, sind Turingberechenbar.

Sprachen, die von Programmen entschieden werden, sind Turingberechenbar.

Varianten von Turingmaschinen

a) Turingmaschinen mit Halbband (Einschränkung)

TM mit einem Arbeitsband der Form:

¢	I	N	P	U	T	$\square$	$\square$	$\square$	$\square$	...

Frage: Wie wirkt sich die Beschränkung auf ein Halbband auf das Berechnungsvermögen dieser Maschine aus?

Antwort: Nicht: alles was Standard-TMen können, können auch TM mit Halbband!
→ dank M_{move}

b) Turingmaschinen mit endlichen Arbeitsband

¢	I	N	P	U	T	\$

→ Solche Maschinen heißen linear beschränkte Automaten (LBA)

→ klar ist: LBA können weniger als TMen

→ aber: Es gibt einen LBA (und damit eine TM, die ausschließl. den Eingabebereich benutzt) der $\{a^n b^n c^n \mid n \in \mathbb{N}\}$ entscheidet