Kapitola 11

Modely strojů, programů a výpočtů

Je nesporné, že technologický pokrok ve výrobě počítačů a jejich programového vybavení během několika posledních desetiletí výrazně rozšířil jejich aplikační možnosti. Tato skutečnost však nic nemění na tom, že algoritmizovatelnost (nebo jinými slovy řešitelnost algoritmickými postupy) má své hranice. Ukážeme to na výpočetním modelu nazývaném Turingův stroj, který byl zaveden ještě před tím, než se začaly používat první samočinné počítače. Nejprve se ovšem budeme věnovat velmi jednoduchým modelům, které se přes svoji jednoduchost (nebo právě díky ní) používají i při rutinní programátorské práci.

11.1 Jazyky a automaty

Abeceda, jazyky nad abecedou a operace s jazyky

Informace uchovávané a zpracovávané v počítačích jsou vyjádřeny prostřednictvím posloupností symbolů, programy pro počítače se rovněž zapisují jako posloupnosti symbolů strukturované podle pravidel použitého programovacího jazyka. Je tedy žádoucí, aby matematické modely, které mají mít schopnost chování počítačů nebo strukturu jejich programů vyjádřit, dokázaly pracovat se symboly. Začneme tedy připomenutím základních pojmů, na kterých staví matematická teorie jazyků.

Abecedou nazýváme libovolnou konečnou množinu A, jejíž prvky nazýváme **symboly**. **Řetěz** (též **slovo**) nad abecedou A je konečná posloupnost symbolů této abecedy včetně prázdné posloupnosti, tedy uspořádaná n-tice symbolů pro $n \geq 0$. Na rozdíl od notace používané pro prvky kartézských součinů, řetězy vyjadřujeme prostým zápisem příslušných symbolů vedle sebe zleva doprava.

Prázdný řetěz nad abecedou A označujeme pomocí ε (ε nesmí být prvkem abecedy A!). Množinu všech řetězů nad abecedou A označujeme jako A^*. **Délkou** řetězu $w = a_1a_2 \dots a_n \in A^*$ nazýváme počet jeho symbolů n a zapisujeme $|w| = n$. Jsou-li $x = a_1a_2 \dots a_m$ a $y = b_1b_2 \dots b_n$ dva řetězy nad stejnou abecedou, pak z nich operací **zřetězení** (též **spojení**, angl. **concatenation**) [1] získáme řetěz

$$x \circ y \;=\; a_1a_2 \dots a_mb_1b_2 \dots b_n$$

Zřetězení $x \circ y$ (většinou píšeme jen xy) tedy vznikne prostým připojením řetězu y za řetěz x. Při zřetězení se délky sčítají, tedy $|xy| = |x| + |y|$. Prostřednictvím zřetězení se mohou řetězy dostávat do různých vzájemných vztahů. Tak např. říkáme, že

- řetěz x je **podřetězem** řetězu y, pokud platí $y = uxv$ pro nějaká $u, v \in A^*$
- řetěz x je **prefixem (předponou)** řetězu y, pokud platí $y = xv$ pro nějaké $v \in A^*$
- řetěz x je **sufixem (příponou)** řetězu y, pokud platí $y = ux$ pro nějaké $u \in A^*$.

[1] Zde je možné zdůvodnit oprávněnost notace A^* pro množinu všech řetězů nad abecedou A – je to totiž uzávěr (ve smyslu zavedeném na konci odstavce 1.2) množiny A vzhledem k operaci zřetězení.

Pro přirozené $k(\geq 0)$ a libovolný řetěz w vyjadřujeme řetěz vzniklý jeho k-násobným zřetězením jako w^k, formálně

$$w^0 = \varepsilon, \qquad w^k = w^{k-1} \circ w \text{ pro } k > 0.$$

Pro $w = a_1 a_2 \dots a_n \in A^*$ nazýváme řetěz $a_n a_{n-1} \dots a_1$ **reverzí** řetězu w a značíme w^R.

Libovolnou množinu řetězů nad abecedou A, tedy podmnožinu množiny A^*, nazýváme **jazykem** nad abecedou A. Konečné jazyky je možné definovat prostým výčtem, u nekonečných jazyků je účelné zabývat se možnostmi jejich konečné reprezentace. Pro tyto účely přicházejí do úvahy tři rozdílné postupy:

a) použijeme nějakou standardní notaci, ve které se definice jazyka vyjádří formou strukturovaného výrazu, tedy vlastně řetězem nad jistou definiční abecedou

b) popíšeme (abstraktní) zařízení, které je schopno rozhodnout, zda zadaný řetěz patří do daného jazyka či nikoliv

c) popíšeme (abstraktní) zařízení, které je schopno systematicky generovat všechna slova daného jazyka

K prvnímu z naznačených přístupů můžeme např. dospět tak, že zavedeme nějaké velmi jednoduché (konečné) základní jazyky a určitý počet operací nad jazyky. Konkrétní jazyk se pak pokusíme vytvořit ze základních jazyků pomocí vhodné kombinace těchto operací. Pro jazyky jakožto množiny řetězů je přirozené uvažovat např. množinové operace sjednocení a průniku, těmi však nelze překročit hranici konečnosti. Podobný nedostatek má i operace **zřetězení jazyků** L_1 a L_2 nad abecedou A definovaná vztahem

$$L \;=\; L_1 \circ L_2 \;=\; \{w \in A^* :\; w = x \circ y \text{ pro nějaká } x \in L_1, y \in L_2\}$$

Podobně jako u spojení slov, i u zřetězení jazyků běžně vynecháváme symbol této operace, takže píšeme $L_1 L_2$.

Díky tomu, že A^* obsahuje nekonečně mnoho řetězů pro každou (neprázdnou) abecedu A, je pro libovolný konečný jazyk L jeho **doplněk** $\overline{L} = A^* - -L$ nekonečný. Další operací schopnou vytvořit nekonečný jazyk je **iterace** (též tzv. **Kleeneho operace**) L^* definovaná vztahem

$$L^* \;=\; \{w \in A^* :\; w = w_1 \circ w_2 \circ \dots \circ w_k \text{ pro libovolné } k \;\geq 0 \text{ a } w_1, w_2, \dots, w_k \in L\}.$$

Iterace L^* je tedy tvořena takovými řetězy, které lze získat spojením libovolného (i nulového!) počtu řetězů nad L. Pro každý jazyk L platí $\varepsilon \in L^*$, speciálně pak je $\emptyset^* = \{\varepsilon\}$. Pokud vytváříme spojení nenulového počtu řetězů z L, dostaneme jeho tzv. **pozitivní iteraci** L^+:

$$L^+ \;=\; \{w \in A^* :\; w = w_1 \circ w_2 \circ \dots \circ w_k \text{ pro libovolné } k > 0 \text{ a } w_1, w_2, \dots, w_k \in L\}.$$

Platí $L^* = L^+ \cup \{\varepsilon\}$, ale obecně nelze psát $L^+ = L^* - -\{\varepsilon\}$ (kdy tento vztah platí?).

Operace sjednocení, zřetězení a iterace tvoří základ formalismu tzv. **regulárních výrazů**, které představují příklad prvního z výše uvedených postupů při definici jazyků. Pomocí nich je možné popsat libovolný ze třídy tzv. **regulárních jazyků**, pro které je druhý z uvedených postupů (tzn. rozpoznávání řetězů jazyka) vyjádřen pomocí konečných automatů, jimž se věnuje následující odstavec.

Konečné automaty

Konečný automat se koncipuje jako velmi jednoduché zařízení schopné definovaným způsobem reagovat na libovolný řetěz, který je mu předložen na vstup. Toto zařízení bude tedy mít nějakou „řídicí jednotku“ a vstupní pásku, na níž mu budeme zadávat vstup. Jediné, čeho bude konečný automat schopen, je posun čtecí hlavy po vstupní pásce (pouze jedním směrem!)

a reakce na vstup vyjádřená změnou jeho vnitřního stavu v závislosti na stavu aktuálním a na právě čteném symbolu na vstupu.

Abychom pro zpracování různých vstupů zajistili stejné výchozí podmínky, bude se automat na počátku své činnosti vždy nacházet ve specifikovaném tzv. **počátečním stavu**. Protože má automat rozpoznávat řetězy patřící do nějakého jazyka, bude nás po přečtení celého vstupu automatem zajímat jediné: zda automat řetěz **přijal (akceptoval)**. Tento výsledek bude vyjádřen tím, že se automat zastaví v některém ze svých **koncových stavů**.

Definice 11.1: Deterministický konečný automat (zkráceně FSA z angl. **finite-state automaton**) je pětice $M = \langle Q, A, \delta, s, F\rangle$, kde

- Q je konečná **množina stavů** automatu
- A je (konečná) **abeceda**
- $\delta : \ Q \times A \mapsto Q$ je **přechodové zobrazení** automatu
- $s \in Q$ je **počáteční stav**
- $F \subseteq Q$ je množina **koncových stavů**

Nejdůležitější složku konečného automatu – jeho přechodové zobrazení – interpretujeme tak, že pokud automat ve stavu q čte ze vstupní pásky symbol $a \in A$, pak $p = \delta(q, a) \in Q$ jednoznačně určuje stav, do něhož se M dostane. Abychom mohli předpokládanou činnost automatu (tj. „výpočet“ jím prováděný) formálně popsat, zavedeme tzv. **konfiguraci**, která zachycuje jak vnitřní stav automatu, tak i dosud nezpracovanou část vstupního řetězu.

Definice 11.2: Konfigurací konečného automatu $M = \langle Q, A, \delta, s, F\rangle$ nazýváme libovolnou dvojici $(q, w) \in Q \times A^*$. **Přechodem** mezi konfiguracemi automatu M nazýváme relaci $\vdash_M \subseteq (Q \times A^*) \times (Q \times A^*)$ definovanou vztahem

$$(q, aw) \vdash_M (p, w) \ \Leftrightarrow_{df} \ \text{pro nějaké } q \in Q, a \in A, w \in A^* \text{ platí } \delta(q, a) = p.$$

Přechod mezi konfiguracemi je tedy vyjádřením činnosti, kterou konečný automat vykoná v jednom výpočetním kroku. Pro každou konfiguraci s neprázdným řetězem ve druhé složce existuje jednoznačně určená konfigurace, která je s ní v relaci přechodu, takže přechod konfigurací je zobrazením množiny $Q \times A^+$ do množiny $Q \times A^*$.

K relaci přechodu můžeme získat (viz odst. 1.2) reflexivně-tranzitivní uzávěr $\vdash_M^*$, který vyjadřuje vztah mezi konfiguracemi realizovaný libovolným počtem přechodů (včetně žádného). Říkáme, že řetěz $w \in A^*$ je **přijat** automatem M, pokud existuje stav $q \in F$ takový, že $(s, w) \vdash_M^* (q, \varepsilon)$. Konečně jazyk $L(M)$ **přijímaný automatem** M je tvořen všemi řetězy, které tento automat přijímá.

Tak např. uvažujme konečný automat $M = \langle Q, A, \delta, s, F\rangle$, kde $Q = \{q_0, q_1, q_2, q_3\}$, $A = \{0, 1\}$, $s = q_0$, $F = \{q_0\}$ a přechodové zobrazení δ je definováno tabulkou na obr. 11.1a). Ukážeme posloupnost přechodů, jejímž prostřednictvím tento automat přijme řetěz 01101001 (pro zvýraznění podtrháváme aktuálně čtený symbol v řetězu):

$$\begin{array}{llllllllll} (q_0, \underline{0}1101001) & \vdash_M & (q_1, \underline{1}101001) & \vdash_M & (q_3, \underline{1}01001) & \vdash_M & (q_1, \underline{0}1001) & \vdash_M \\ & \vdash_M & (q_0, \underline{1}001) & \vdash_M & (q_2, \underline{0}01) & \vdash_M & (q_3, \underline{0}1) & \vdash_M \\ & \vdash_M & (q_2, \underline{1}) & \vdash_M & (q_0, \varepsilon) & & & \end{array}$$

Naproti tomu např. pro řetěz 01011 dostaneme posloupnost přechodů

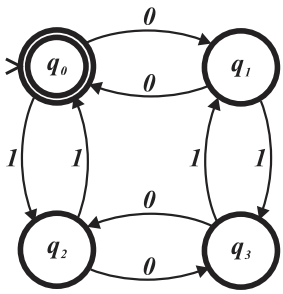
$$\begin{array}{llllllllll} (q_0, \underline{0}1011) & \vdash_M & (q_1, \underline{1}011) & \vdash_M & (q_3, \underline{0}11) & \vdash_M & (q_2, \underline{1}1) & \vdash_M & (q_0, \underline{1}) & \vdash_M \\ & \vdash_M & (q_2, \varepsilon)\ , & & & & & & & \end{array}$$

což znamená, že automat tento řetěz nepřijal. Je současně vidět, že pro libovolnou počáteční konfiguraci (s, w) je zapotřebí přesně $|w|$ přechodů k tomu, aby se automat dostal do nějaké konfigurace (q, ε) rozhodující o přijetí či nepřijetí řetězu w.

Přehledné vyjádření struktury konečného automatu představuje jeho **přechodový** (též **stavový**) **diagramu**. Příklad přechodového diagramu automatu zadaného v předchozím příkladu

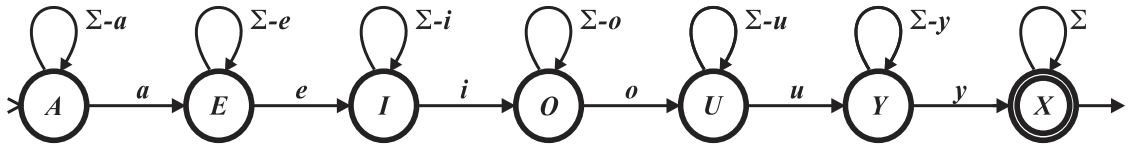
q	a	$\delta(q,a)$	q	a	$\delta(q,a)$
q_0	0	q_1	q_2	0	q_3
q_0	1	q_2	q_2	1	q_0
q_1	0	q_0	q_3	0	q_2
q_1	1	q_3	q_0	1	q_1

a)



b)

Obrázek 11.1: Konečný automat a) přechodové zobrazení, b) přechodový diagram



Obrázek 11.2: Zkrácený zápis přechodového diagramum

ukazuje obr. 11.1b). Stavy automatu jsou vyjádřeny uzly diagramu, přičemž se od ostatních stavů vhodně odliší počáteční stav a stavy koncové. Každá hrana odpovídá jedné kombinaci argumentů přechodového zobrazení: pro $\delta(q,a) = p$ je vedena ze stavu q do stavu p a ohodnocena symbolem a. Tak např. v diagramu na obrázku vyjadřuje hodnotu $\delta(q_2, 0) = q_3$ hrana vedoucí ze stavu q_2 do stavu q_3 ohodnocená symbolem 0.

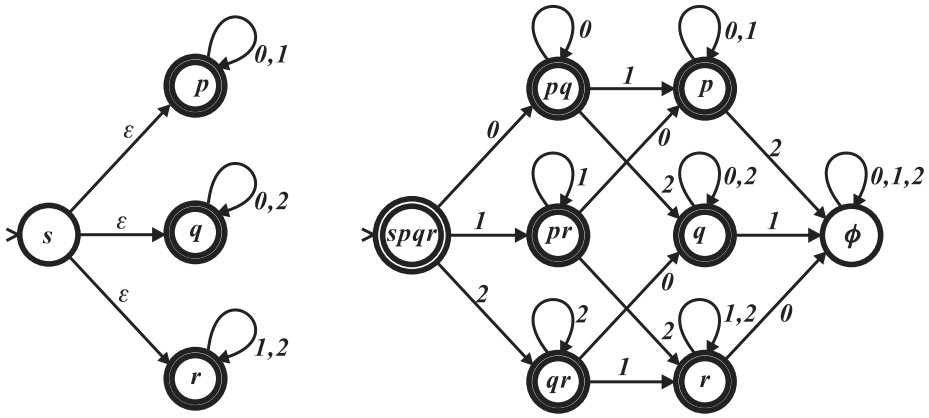
Při zápisu přechodových diagramů se někdy používají zjednodušující konvence. Tak např. rovnoběžné hrany se nahrazují hranou jedinou, která je ohodnocena symboly původních hran oddělenými čárkou. Další typickou zkratkou je např. vyjádření toho, že přechod po hraně se realizuje pro všechny symboly různé od symbolu x – to se vyjadřuje ohodnocením $A - x$, popř. $\neq x$ nebo prostě $\overline{x}$. Použití těchto konvencí ukazuje přechodový diagram na obr. 11.2, jenž reprezentuje automat přijímající řetězy, které obsahují podposloupnost symbolů *aeiouy* (v tomto pořadí).

Koncept (deterministického konečného) automatu je velmi jednoduchý, a tak lze uvažovat o jeho možných obohaceních. Jednou z možností je zavedení výstupu – při čtení každého symbolu automat zapíše jeden symbol na svoji výstupní pásku. Takové zařízení se nazývá (deterministický konečný) **překladový automat** [2] a formálně se zavede doplněním původní definice konečného automatu o výstupní abecedu Θ a výstupní zobrazení

$$\lambda : \; Q \times A \; \mapsto \; \Theta.$$

Dalším zajímavým obohacením konečných automatů je zavedení konceptu **nedeterminismu**. Intuitivní význam tohoto konceptu spočívá v tom, že automat má při čtení každého vstupního symbolu zadáno několik (včetně žádné) rovnocenných možností postupu, jejichž výběr není ničím ovlivněn. Činnost nedeterministického automatu si pak můžeme představit tak, že automat sleduje pouze jednu výpočetní posloupnost přechodů, ve které si v každém místě volby dokáže vybrat mezi několika alternativami tu „správnou“ (pokud existuje). Jiná inter-

[2] Pojem Mealyho automatu, který se používá v kontextu analýzy a návrhu logických systémů, odpovídá našemu zavedení překladového automatu s tím, že se nevyčleňují žádné koncové stavy. Mooreův automat používaný v témže kontextu se od Mealyho liší pouze tím, že jeho výstupní zobrazení je závislé pouze na vnitřních stavech, tedy $\lambda : \; Q \; \mapsto \; \Theta$. Vlastnostem těchto typů automatů se podrobněji věnují reference citované na začátku této kapitoly.



Obrázek 11.3: Přechod od nedeterministického k ekvivalentnímu deterministickému automatu

pretace spočívá v představě, že automat je schopen rozvíjet všechny možné alternativy výpočtu paralelně – namísto prosté posloupnosti přechodů tak rozvíjí jakýsi kořenový strom přechodů.

Definice 11.3: Nedeterministický konečný automat (zkráceně NFSA) je pětice $M = \langle Q, A, \Delta, s, F\rangle$, kde

- Q je konečná **množina stavů** automatu
- A je (konečná) **abeceda**
- $\Delta : \ (Q \times (A \cup \{\varepsilon\})) \mapsto 2^Q$ je **přechodové zobrazení** automatu
- $s \in Q$ je **počáteční stav**
- $F \subseteq Q$ je množina **koncových stavů**

Rozšíření definičního oboru druhého argumentu přechodového zobrazení o prázdný řetěz dává automatu možnost provádět tzv. ε-přechody, při nichž se nečte žádný vstup. Obor hodnot přechodového zobrazení je dán systémem všech podmnožin množiny stavů, hodnotou $\Delta(q, a)$ je tedy obecně libovolná (např. i prázdná) podmnožina množiny stavů Q. Definice konfigurace NFSA se shoduje s definicí pro FSA, rozdíl nastává až při zavedení relace přechodu mezi konfiguracemi.

Definice 11.4: Přechodem mezi konfiguracemi NFSA M nazýváme relaci $\vdash_M \subseteq \ (Q \times A^*) \times (Q \times A^*)$ definovanou vztahem

$$(q, w) \vdash_M (p, w') \ \Leftrightarrow_{df} \text{ pro nějaké } q \in Q, u \in A \cup \{\varepsilon\}, w \in A^* \text{ platí } w = uw' \text{ a } p \in \Delta(q, u).$$

NFSA M **přijímá řetěz** $w \in A^*$, pokud existuje stav $q \in F$ takový, že $(s, w) \vdash^*_M (q, \varepsilon)$. Konečně **jazyk** $L(M)$ **přijímaný** NFSA M je tvořen všemi řetězy, které automat M přijímá.

Jako příklad NFSA uvádíme na obr. 11.3a) přechodový diagram automatu, který přijímá ty řetězy nad abecedou $\{0, 1, 2\}$, ve kterých se alespoň jeden ze symbolů této abecedy nevyskytuje.

Pro libovolný deterministický konečný automat $M = \langle Q, A, \delta, s, F\rangle$ snadno sestrojíme NFSA $M' = \langle Q, A, \Delta', s, F\rangle$ tak, aby platilo $L(M) = L(M')$ (takové automaty M a M' se nazývají **ekvivalentní**). Stačí určit přechodové zobrazení Δ' takto:

$$\begin{aligned} \Delta'(q, a) &= \{\delta(q, a)\} \ \text{ pro všechna } q \in Q, \ a \in A \\ \Delta'(q, \varepsilon) &= \emptyset \ \text{ pro všechna } q \in Q \end{aligned}$$

Může se tedy na první pohled zdát, že nedeterminismus rozšiřuje výpočetní možnosti konečných automatů, ale ve skutečnosti tomu tak není, což dokazuje následující tvrzení.

Věta 11.5: Ke každému nedeterministickému konečnému automatu existuje ekvivalentní deterministický konečný automat.

Důkaz: Nechť $M = \langle Q, A, \Delta, s, F\rangle$ je zadaný NFSA, ukážeme, jak sestrojit s ním ekvivalentní FSA $M' = \langle Q', A, \delta', s', F'\rangle$. Náš postup bude jen formalizací představy simulace paralelního provádění alternativních přechodů NFSA při zpracování vstupního řetězu – v každém okamžiku tyto paralelně uvažované přechody dospějí do určité podmnožiny stavů. K přijetí řetězu přitom stačí, aby alespoň jeden z množiny stavů (paralelně) dosažených po jeho přečtení byl koncový.

Je ovšem třeba vzít v úvahu, že NFSA může provádět také ε-přechody, takže ke každému stavu q, do něhož se může NFSA dostat, musíme přidat i stavy, dosažitelné z q prostřednictvím (libovolného počtu) ε-přechodů – tuto množinu stavů budeme označovat jako $E(q)$. Formálně tedy

$$E(q) = \{p \in Q :\ (q, \varepsilon) \vdash_M^* (p, \varepsilon)\}$$

Nyní již můžeme určit všechny složky vytvářeného FSA:

$$Q' = 2^Q \qquad s' = \{E(s)\} \qquad F' = \{P \subseteq Q :\ P \cap F \neq \emptyset\}$$

$$\delta'(P, a) = \cup_{q\in P} \cup_{p\in\Delta(q,a)} E(p) \ \text{ pro všechna } P \subseteq Q, a\ \in A$$

Zhuštěný zápis určující hodnoty přechodového zobrazení δ' můžeme přečíst tak, že se pro všechny stavy q z množiny P zjistí, do jakých stavů p je možno v NFSA M přejít ze stavu q pro vstup a, ale pro tyto stavy p se musí ještě uvažovat všechny stavy $E(p)$ dosažitelné pomocí ε-přechodů (bez čtení vstupu).

Nyní ještě zbývá dokázat, že sestrojený automat je ekvivalentní původnímu. K tomu postačí ověřit, že pro libovolná $p, q \in Q$ a $w \in A$ platí následující ekvivalence pro relace přechodu mezi konfiguracemi automatů M a M':

$$(q, w) \vdash_M^* (p, \varepsilon) \quad \Leftrightarrow \quad \exists P \subseteq Q :\ (p \in P\ \&\ (E(q), w) \vdash_{M'}^* (P, \varepsilon))$$

Důkaz platnosti tohoto vztahu se provede indukcí podle délky řetězu $|w| = n$ a vzhledem k jeho povýtce technickému charakteru jej v tomto textu nebudeme reprodukovat. △

Postup uvedený v předchozím důkazu můžeme použít k získání FSA ekvivalentního NFSA na obr. 11.3a). Vzhledem k omezenému počtu ε-přechodů je snadné určit množiny $E(x)$:

$$E(s) = \{p, q, r, s\}, \quad E(p) = \{p\}, \quad E(q) = \{q\}, \quad E(r) = \{r\}$$

Nyní budeme počítat hodnoty přechodového zobrazení δ', věnujeme se ovšem pouze stavům, které jsou dostupné z počátečního stavu $E(s)$:

$$\begin{aligned}
\delta'(E(s), 0) &= \delta'(\{p, q, r, s\}, 0) = E(p) \cup E(q) = \{p, q\} \\
\delta'(E(s), 1) &= \delta'(\{p, q, r, s\}, 1) = E(p) \cup E(r) = \{p, r\} \\
\delta'(E(s), 2) &= \delta'(\{p, q, r, s\}, 2) = E(q) \cup E(r) = \{q, r\}
\end{aligned}$$

V dalším výpočtu se již ε-přechody neuplatní, takže budeme zápis $E(\ldots)$ vynechávat.

$$\begin{array}{lll}
\delta'(\{p, q\}, 0) = \{p, q\} & \delta'(\{p, q\}, 1) = \{p\} & \delta'(\{p, q\}, 2) = \{q\} \\
\delta'(\{p, r\}, 0) = \{p\} & \delta'(\{p, r\}, 1) = \{p, r\} & \delta'(\{p, r\}, 2) = \{r\} \\
\delta'(\{q, r\}, 0) = \{q\} & \delta'(\{q, r\}, 1) = \{r\} & \delta'(\{q, r\}, 2) = \{q, r\} \\
\delta'(\{p\}, 0) = \{p\} & \delta'(\{p\}, 1) = \{p\} & \delta'(\{p\}, 2) = \emptyset \\
\delta'(\{q\}, 0) = \{q\} & \delta'(\{q\}, 1) = \emptyset & \delta'(\{q\}, 2) = \{q\} \\
\delta'(\{r\}, 0) = \emptyset & \delta'(\{r\}, 1) = \{r\} & \delta'(\{r\}, 2) = \{r\} \\
\delta'(\emptyset, 0) = \emptyset & \delta'(\emptyset, 1) = \emptyset & \delta'(\emptyset, 2) = \emptyset
\end{array}$$

Přechodový diagram automatu získaného uvedeným postupem uvádíme na obr. 11.3b) – při označení stavů používáme zkrácenou notaci, takže např. $pqrs$ znamená stav $\{p, q, r, s\}$, apod. Srovnáme-li složitost a názornost diagramů obou ekvivalentních automatů z obr. 11.3, pak jistě dáme přednost původní nedeterministické verzi. Ještě výraznější rozdíl by se projevil, pokud bychom např. chtěli zobecnit daný automat pro obdobně definovaný jazyk nad abecedou $\{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$ – viz cvičení 11.1-3.

Můžeme tedy konstatovat, že nedeterminismus konečných automatů nerozšiřuje třídu jazyků, které jsou tyto automaty schopné rozpoznat - jsou to stále jen dříve zmíněnné regulární jazyky. Nedeterminismus je ovšem prostředkem, který zjednodušuje návrh automatu přijímajícího určitý jazyk, neboť stačí navrhnout jeho nedeterministickou verzi, a převod na deterministický ekvivalent je pak už rutinní záležitostí. Třída regulárních jazyků je sice velmi omezená, přesto jsme dosud neuvedli příklad jazyka, který není regulární. Namísto příkladu uvedeme nejprve obecné tvrzení, které umožní neregulární jazyky poznat.

Věta 11.6: (**Pumping lemma**) Nechť L je regulární jazyk. Potom existuje přirozené číslo $n \geq 1$ takové, že každý řetěz $w \in L$ délky $|w| \geq n$ lze psát ve tvaru $w = xyz$, přičemž $y \neq \varepsilon, |xy| \leq n$ a $xy^i z \in L$ pro každé $i \geq 0$.

Důkaz: Tvrzení lze snadno dokázat prostřednictvím grafové analýzy přechodového diagramu automatu, který přijímá jazyk L – bližší návod viz cvičení 11.1-5. △

Použitím uvedeného tvzení můžeme snadno dokázat, že např. jazyk

$$L_1 = \{0^n 1^n : \ n \geq 0\}$$

není regulární. Intuitivní zdůvodnění je, že konečný automat nemá možnost „pamatovat si“ svými stavy libovolně veliké n, aby mohl srovnat počet nul a počet jedniček. Ze stejného důvodu není regulární ani obecnější jazyk

$$L_2 = \{w \in \{0, 1\}^* : \ w \text{ obsahuje stejný počet nul jako jedniček}\}.$$

Předpokládejme, že pro kterýkoliv z těchto jazyků existuje n s požadovanými vlastnostmi. Pak pro každý rozklad řetězu $0^n 1^n$ (patřícího do L_1 i L_2) na tři části xyz takové, že $|xy| \leq n$, je nutně $y = 0^k$ pro nějaké kladné $k \leq n$ a řetěz $xz = 0^{n-k} 1^n$ nepatří ani do jednoho z uvedených jazyků.

Gramatiky

Poslední ze zmiňovaných přístupů konečného popisu jazyků představují systémy pro generování řetězů jazyka založené na gramatikách.

Definice 11.7: Gramatikou nazýváme čtveřici $G = \langle A, N, P, S \rangle$, kde

- A je tzv. **terminální abeceda**
- N je tzv. **neterminální abeceda**
- $P \subseteq ((A \cup N)^* N (A \cup N)^*) \times (A \cup N)^*$ je tzv. **množina přepisovacích pravidel**
- $S \in N$ je tzv. **počáteční symbol**

Jednotlivá přepisovací pravidla píšeme ve tvaru $\alpha \to \beta$, kde $\alpha \in (A \cup N)^* N (A \cup N)^*, \beta \in (A \cup N)^*$ (podmínka pro tvar levé strany pravidla jen vyjadřuje požadavek, že musí obsahovat alespoň jeden neterminál). Podobně jako u konečných automatů, je také u gramatik zapotřebí dát jednotlivé složky do vzájemné souvislosti ve vztahu k předpokládanému záměru. U automatu jsme formalizovali jeho činnost pomocí pojmu přechod mezi konfiguracemi, u gramatik má odpovídající roli odvození (derivace).

Definice 11.8: Nechť $G = \langle A, N, P, S \rangle$ je gramatika, $w, v \in (A \cup N)^*$ nějaké řetězy.

- Říkáme, že z řetězu w lze **přímo odvodit** (**přímo derivovat**) v gramatice G řetěz v (zapisujeme $w \Rightarrow_G v$), pokud mají řetězy w, v tvar $w = x\alpha y, v = x\beta y$, kde $x, y \in (A \cup N)^*$ a $\alpha \to \beta \in P$ je pravidlo gramatiky G.
- Říkáme, že z řetězu w lze **odvodit** (**derivovat**) v gramatice G řetěz v (zapisujeme $w \Rightarrow_G^* v$), pokud existuje posloupnost řetězů $x = x_0,\ x_1, \ldots,\ x_n = y \in (A \cup N)^*$ taková, že $x_i \Rightarrow_G x_{i+1}$ pro $i = 0, 1, \ldots, n-1$. Posloupnost $x_0, x_1, \ldots, x_n$ pak nazýváme **odvozením** (**derivací**) řetězu v z řetězu w (v gramatice G) a n nazýváme **délkou** tohoto odvození.
- **Jazyk** $L(G)$ **generovaný gramatikou** G je množina všech terminálních řetězů derivovatelných z počátečního symbolu gramatiky S, tedy

$$L(G) = \{w \in A^* :\ S \Rightarrow_G^* w\}$$

Koncept gramatiky v právě uvedené obecnosti je velmi silný a třída jazyků generovaných nějakou gramatikou značně překračuje rámec regulárních jazyků. Omezení generativní schopnosti gramatik se dosáhne stanovením vhodných restrikcí na tvar přepisovacích pravidel. Za standardní se považuje následující tzv. **Chomského hierarchie gramatik**:

typ 0 (neomezená) – na přepisovací pravidla se neklade žádné omezení, tzn. mohou mít tvar uvedený v definici 11.7

typ 1 (kontextová) – pravidla musí být tvaru $xAy \to x\beta y$, kde $x, y \in (A \cup N)^*,\ A \in N,\ \beta \in (A \cup N)^+$

typ 2 (bezkontextová) – pravidla musí být tvaru $A \to \beta$, kde $A \in N,\ \beta \in (A \cup N)^*$

typ 3 (regulární) – pravidla musí být tvaru $A \to a$ nebo $A \to aB$, kde $a \in A,\ A, B \in N$

Hierarchie zavedená pro gramatiky se přirozeným způsobem přenáší i na jazyky – jazyk označujeme typem gramatiky generující tento jazyk. **Regulární jazyk** je tedy generovatelný regulární gramatikou, **bezkontextový jazyk** gramatikou bezkontextovou, apod. Výše uvedená omezení pro gramatiky typu 3 a typu 1 vylučují odvodit z počátečního symbolu prázdný řetěz. Aby mohly regulární a kontextové jazyky obsahovat také prázdný řetěz, připouští se v jejich gramatikách pravidlo $S \to \varepsilon$ za podmínky, že počáteční symbol S se nevyskytne na pravé straně žádného přepisovacího pravidla.

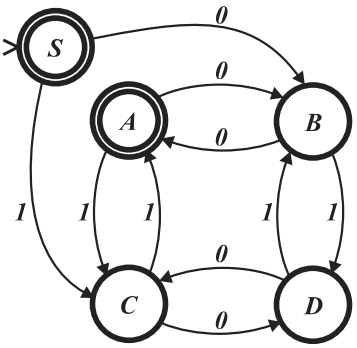
Až dosud jsme označení *regulární jazyk* používali pro jazyk přijímaný konečným automatem, předchozí odstavec však toto označení vztahuje ke generovatelnosti regulární gramatikou. Ukážeme, že to není v žádném rozporu.

Věta 11.9: Nechť L je jazyk přijímaný konečným deterministickým automatem M. Potom existuje regulární gramatika generující jazyk L.

Důkaz: Nechť $M = \langle Q, A, \delta, s, F\rangle$, předpokládáme $s \notin F$ a požadovanou gramatiku $G = \langle A, N, P, S\rangle$ sestrojíme následujícím způsobem. Položíme $N = Q$ a $S = s$, takže zbývá vytvořit jen množinu přepisovacích pravidel. Pro každou kombinaci argumentů přechodového zobrazení $(q, a) \in Q \times A$ přidáme podle hodnoty $p = \delta(q, a)$ do množiny P pravidlo $q \to ap$. Je-li navíc $p \in F$, přidáme do P ještě pravidlo $q \to a$.

Další část důkazu, tj. ekvivalence této gramatiky výchozímu automatu, je pouze technickým ověřením, že každou posloupnost konfigurací vedoucí ke přijetí slova $w \in L(M)$ lze simulovat odpovídající derivací slova w v gramatice G.

Ještě je třeba ošetřit v úvodu důkazu vyloučený případ $s \in F$, který způsobuje, že přijímaný jazyk obsahuje prázdný řetěz. Do gramatiky pak nestačí jen přidat přepisovací pravidlo $S \to \varepsilon$, ale výchozí automat musíme napřed ekvivalentně upravit tak, aby do jeho počátečního stavu nevedl žádný přechod. Obecný postup vyplyne z následujícího příkladu. △



Obrázek 11.4: Úprava FSA přijímajícího prázdný řetěz

Způsobem uvedeným v předchozím důkazu můžeme např. pro konečný automat vyjádřený na obr. 11.1 získat ekvivalentní regulární gramatiku $G = \langle\{0,1\}, \{S, A, B, C, D\}, P, S\rangle$. Nejprve jsme vyřešili přijetí prázdného řetězu zavedením nového počátečního stavu (viz obr. 11.4), z něhož vedou stejné hrany jako z původního, ale žádná hrana v něm nekončí. Pro přehlednost jsme navíc přejmenovali ostatní stavy, aby se lépe formulovala přepisovací pravidla:

$S \to 0B$	$A \to 0B$	$B \to 0A$	$C \to 0D$	$D \to 0C$
$S \to 1C$	$A \to 1C$	$B \to 0$	$C \to 1A$	$D \to 1B$
$S \to \varepsilon$		$B \to 1D$	$C \to 1$	

Věta 11.10: Nechť L je jazyk generovaný regulární gramatikou $G = \langle A, N, P, S\rangle$. Potom existuje konečný automat přijímající jazyk L.

Důkaz: Nechť $G = \langle A, N, P, S\rangle$, požadovaný automat $M = \langle Q, A, \Delta, s, F\rangle$ sestrojíme jako nedeterministický následujícím způsobem:

$$\begin{aligned} Q &= N \cup \{f\}, \text{ kde } f \notin N \text{ je nový symbol} \\ s &= S \\ F &= \{f\} \text{ nebo } F = \{f, s\} \text{ (je} - \text{li } (S \to \varepsilon) \in P) \end{aligned}$$

Přechodový diagram (tj. zobrazení Δ) automatu M získáme tak, že za každé pravidlo $A \to aB$ vytvoříme hranu ze stavu A do stavu B ohodnocenou symbolem a. Podobně za každé pravidlo $A \to a$ vytvoříme hranu ze stavu A do stavu f ohodnocenou symbolem a. Zbývající část důkazu, tj. ověření ekvivalence $L(G) = L(M)$, má už jen mechanický charakter a ponecháme ji na čtenáři. △

Na příkladu regulárních jazyků jsme ukázali vzájemný vztah různých forem vyjádření téhož jazyka. Podobné možnosti existují i pro bezkontextové jazyky, které jsou z aplikačního hlediska velmi důležité (především při tvorbě překladačů programovacích jazyků). Důkladné zpracování této problematiky je však mimo rámec tohoto textu (viz např. [26]).

Cvičení

11.1-1. Dokažte, že FSA popsaný přechodovým diagramem na obr. 11.1 přijímá jazyk obsahující pouze řetězy se sudým počtem výskytů symbolu 0 i symbolu 1.
(*Návod*: Postupujte indukcí podle délky řetězu s tím, že tvrzení rozšíříte o specifikaci řetězů w_i, pro které nastane $(s, w_i) \vdash_M^* (q_i, \varepsilon)$ pro $i = 0, 1, 2, 3$.)

11.1-2. Navrhněte přechodový diagram FSA, který přijímá

(a) binární řetězy s lichou paritou (tj. lichým počtem jedniček)
(b) binární řetězy, v nichž se nevyskytují čtyři jedničky za sebou
(c) binární řetězy, v nichž jsou každé dva výskyty jedniček odděleny alespoň třemi nulami

11.1-3. Zobecněte NFSA z obr. 11.3a) tak, aby přijímal podobně definované řetězy nad abecedou $\{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$. Bude se tedy jednat o automat přijímající zápisy desítkových čísel, v nichž se alespoň jedna desítková číslice nevyskytuje. Zamyslete se nad složitostí přímého návrhu ekvivalentního FSA.

11.1-4. Použitím grafových pojmů na přechodový diagram konečného automatu formulujte podmínku pro to, aby

(a) automat přijímal prázdný jazyk
(b) automat přijímal nějaký řetěz délky n
(c) automat přijímal nekonečný jazyk

(*Návod*: Vyjádřete nejprve v grafových pojmech, čemu na přechodovém diagramu odpovídá posloupnost přechodů mezi konfiguracemi FSA vedoucí k přijetí nějakého řetězu.)

11.1-5. Doplňte vhodným způsobem nějakou datovou strukturu používanou pro reprezentaci orientovaných grafů tak, aby se mohla použít k vyjádření přechodového diagramu NFSA. Na základě této reprezentace navrhněte algoritmus simulující činnost NFSA bez ε-přechodů při procesu zpracování zadaného vstupního řetězu. Určete časovou a paměťovou složitost navrženého algoritmu.
(*Návod*: Inspirujte se algoritmem procházení grafu do šířky.)

11.1-6. Dokažte tvrzení věty 11.6.
(*Návod*: Použijte přechodový diagram deterministického automatu přijímajícího daný jazyk a vyjděte z grafové interpretace toho, že tento automat přijal nějaký řetěz délky alespoň rovné počtu stavů automatu.)

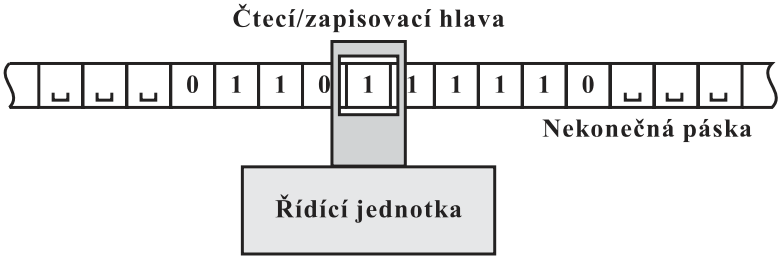
11.1-7. Prostřednictvím konstrukce vhodných (nedeterministických) konečných automatů dokažte, že třída regulárních jazyků je uzavřená vůči operacím sjednocení, zřetězení, iterace, doplňku a průniku.

11.2 Turingovy stroje

Studium výpočetních modelů jsme začali velmi jednoduchým strojem – konečným automatem. Uvedli jsme též několik možných zobecnění tohoto pojmu, která však výpočetní schopnosti konečných automatů nezměnila. Pro dosažení skutečného kvalitativního skoku je třeba provést zásadnější úpravu, např. poskytnout stroji neomezenou paměť.

V tomto odstavci se budeme věnovat hledání hranice zvyšování výpočetních schopností strojů. Tato otázka těsně souvisí s tím, co je možné považovat za algoritmicky řešitelné a jak vůbec rozumět samotnému pojmu algoritmus. Odpověď na podobné otázky přináší studium **Turingových strojů**. Jsou to abstraktní zařízení (matematické modely), která nesou jméno anglického matematika Alana M. Turinga, jenž definoval jejich strukturu v článku publikovaném v roce 1936. Vlastnosti Turingových strojů nám vyhovují, neboť jejich návrh sleduje splnění následujících tří kritérií:

- musí mít charakter automatu, tedy jednoduchého abstraktního stroje, co do činnosti podobného konečnému automatu
- musí být co nejjednodušší, aby je bylo možno snadno a dobře popsat, formálně definovat a studovat
- musí být dostatečně obecné s ohledem na výpočty, které jsou schopny provádět.



Obrázek 11.5: Turingův stroj

11.2.1 Struktura a chování Turingova stroje

Základními částmi Turingova stroje jsou řídicí jednotka (s konečným počtem vnitřních stavů) a páska (viz obr. 11.5). Páska je vybavena čtecí/zapisovací hlavou, jejímž prostřednictvím řídicí jednotka zjišťuje obsah právě čteného políčka pásky, a tento obsah může rovněž změnit. Řídicí jednotka pracuje v diskrétních krocích – v závislosti na svém vnitřním stavu a na symbolu čteném z pásky provede v každém kroku dvě operace[3]:

- přejde do dalšího stavu
- přepíše čtený symbol, nebo posune čtecí/zapisovací hlavu o jedno místo vpravo nebo vlevo.

Páska Turingova stroje je jednostranně nekonečná – má levý konec, ale směrem vpravo není počet použitelných políček omezen. Aby nemohla nastat situace, kdy se má čtecí/zapisovací hlava posunout nalevo od konce pásky, budeme předpokládat, že první (nejlevější) políčko pásky obsahuje vždy speciální symbol $\triangleright$, jenž se nikde jinde nebude používat. Navíc, jakmile Turingův stroj tento symbol přečte, čtecí/zapisovací hlava se vždy posune doprava. Pro posun hlavy budeme používat symboly $\leftarrow$ a $\rightarrow$, o nichž předpokládáme, že nepatří do žádné z nadále používaných abeced. Nyní již můžeme uvést formální definici Turingova stroje.

Definice 11.11: Turingův stroj je uspořádaná pětice $M = \langle Q, A, \delta, s, F\rangle$, kde

- Q je konečná **množina stavů**
- A je **abeceda** obsahující mj. prázdný symbol (mezeru) $\sqcup$ a symbol konce pásky $\triangleright$, ale neobsahující symboly $\leftarrow$ a $\rightarrow$
- $s \in Q$ je **počáteční stav**
- $F \subseteq Q$ je množina **koncových stavů**
- $\delta : (Q - F) \times A \mapsto Q \times (A \cup \{\leftarrow, \rightarrow\})$ je **přechodové zobrazení** takové, že
  (a) pro všechna $q \in Q - F$ platí, že je-li $\delta(q, \triangleright) = (p, b)$, pak $b = \rightarrow$
  (b) pro všechna $q \in Q - F$ a $a \in A$ platí, že je-li $\delta(q, a) = (p, b)$, pak $b \neq \triangleright$

Podmínky pro přechodové zobrazení vyjadřují požadavek pohybu hlavy vpravo při dosažení levého konce pásky, resp. zákaz použití symbolu konce pásky pro jiné účely. Levá složka $(Q-F)$ definičního oboru přechodového zobrazení δ vyjadřuje skutečnost, že pro koncové stavy není zobrazení definováno – Turingův stroj se „zastaví“. Interpretace hodnoty přechodového zobrazení $\delta(q, a) = (p, b)$ je taková, že čte-li stroj ve stavu $q \in Q - F$ na pásce symbol $a \in A$, pak přejde do stavu $p \in Q$ a čtený symbol přepíše na b (pro $b \in A$), případně posune hlavu o jedno místo vlevo nebo vpravo (pro $b = \leftarrow$ nebo $b = \rightarrow$). Hodnoty přechodového zobrazení Turingova stroje bychom mohli (podobně jako u konečného automatu) vyjadřovat pomocí přechodového diagramu, přidržíme se však tabulkového zápisu, neboť grafickou formu použijeme při hierarchickém vytváření Turingových strojů formou zobecněného skládání (viz dále).

Uvažujme příklad Turingova stroje $M = \langle\{q_0, q_1, f\}, \{a, \sqcup, \triangleright\}, \delta, q_0, \{f\}\rangle$, jehož přechodové zobrazení je dáno tabulkou na obr. 11.6a). Činnost tohoto stroje lze popsat tak, že po zahájení

[3] Jiná (ekvivalentní) definice Turingova stroje předpokládá, že čtený symbol se vždy přepíše a pak se hlava posune vlevo, vpravo, nebo zůstane na místě.

q	x	$\delta(q,x)$
q_0	a	$(q_1, \sqcup)$
q_0	$\sqcup$	$(a, \sqcup)$
q_0	$\triangleright$	$(q_0, \rightarrow)$
q_1	a	(q_0, a)
q_1	$\sqcup$	$(q_0, \rightarrow)$
q_1	$\triangleright$	$(q_1, \rightarrow)$

a)

q	x	$\delta(q,x)$
q_0	a	$(q_0, \leftarrow)$
q_0	$\sqcup$	$(b, \sqcup)$
q_0	$\triangleright$	$(q_0, \rightarrow)$

b)

Obrázek 11.6: Přechodové zobrazení Turingova stroje

ve stavu q_0 postupně vymaže (tj. přepíše mezerami) všechny symboly a nacházející se od prvního čteného místa (včetně) až k nejbližší mezeře vlevo nebo k levému konci pásky. Při nalezení mezery nebo symbolu konce pásky se stroj zastaví.

Jiné chování vykazuje Turingův stroj popsaný přechodovým zobrazením na obr. 11.6b). Nejprve postupuje vpravo a hledá první mezeru. Po jejím nalezení se vrací doleva a hledá opět mezeru. Narazí-li přitom na konec pásky, zastaví se, jinak na první nalezené mezeře opět začne hledat mezeru vpravo. Je vidět, že v tomto případě může nastat situace, kdy Turingův stroj svoji činnost nikdy neskončí – to je podstatný rozdíl proti tomu, jak se chovají konečné automaty.

Formální popis činnosti Turingova stroje provedeme opět pomocí pojmu konfigurace a relace přechodu mezi konfiguracemi. Úkolem konfigurace je vyjádřit současně stav řídicí jednotky Turingova stroje, obsah jeho pásky a polohu čtecí hlavy. Provede se to tak, že pásku popisují dva řetězy symbolů – první se týká úseku od levého okraje až ke čtené pozici, druhý odpovídá obsahu pásky napravo od čteného místa.

Definice 11.12: Konfigurací Turingova stroje $M = \langle Q, A, \delta, s, F\rangle$ nazýváme uspořádanou trojici

$$(q, \alpha, \beta) \ \in \ Q \times \triangleright A^* \times (A^*(A - \{\sqcup\}) \cup \{\epsilon\}).$$

Obsah pásky vyjádřený dvojicí (α, β) pro $\alpha = wa,\ w \in \triangleright A^*,\ a \in A$ lze přehledněji zapsat jako jediný řetěz, v němž podtrhneme právě čtený symbol, tedy jako $w\underline{a}\beta$. Konfiguraci $(q, wa, \beta),\ w \in \triangleright A^*,\ a \in A$ tedy budeme zapisovat jako $(q, w\underline{a}\beta)$. Je-li stav q konfigurace koncový, nazýváme celou konfiguraci $(q, w\underline{a}\beta)$ **koncovou**. Podobně nazýváme $(s, \triangleright\underline{\sqcup}w)$ **počáteční konfigurací** Turingova stroje M pro vstup w.

Definice 11.13: Pro Turingův stroj $M = \langle Q, A, \delta, s, F\rangle$ zavedeme relaci **přechodu mezi konfiguracemi** $\vdash_M$ následujícími pravidly. Nechť $(q, w\underline{a}u)$ pro nějaké $a \in A$ je konfigurace stroje M. Potom rozlišujeme tři možné případy podle druhé složky hodnoty přechodové funkce:

přepsání symbolu – je-li $\delta(q,a) = (p,b)$ pro $b \in A$, potom $(q, w\underline{a}u) \ \vdash_M \ (p, w\underline{b}u)$

posun vlevo – je-li $\delta(q,a) = (p, \leftarrow)$ a $w = w'a'$ a navíc

1. $a \neq \sqcup$ nebo $u \neq \epsilon$, potom platí $(q, w\underline{a}u) = (q, w'a'\underline{a}u) \ \vdash_M \ (p, w\underline{a'}au)$
2. $a_1 = \sqcup$ a $u_1 = \epsilon$, potom platí $(q, w\underline{\sqcup}) = (q, w'a'\underline{\sqcup}) \ \vdash_M \ (p, w'\underline{a'})$

posun vpravo – je-li $\delta(q,a) = (p, \rightarrow)$ a navíc

1. $u = a'u', a' \in A$, potom platí $(q, w\underline{a}u) = (q, w\underline{a}a'u') \ \vdash_M \ (p, wa\underline{a'}u')$
2. $u = \epsilon$, potom platí $(q, w\underline{a}) \ \vdash_M \ (p, wa\underline{\sqcup})$

Relace přechodu mezi konfiguracemi pouze formalizuje již dříve popsanou činnost Turingova stroje. V případě 1 se jen přepisuje čtený symbol bez pohybu hlavy. V případě 2 se hlava posouvá vlevo a alternativy (a) a (b) rozlišují, zda je část pásky napravo od nové polohy hlavy neprázdná. V případě 3 se hlava posouvá doprava a smysl alternativ je obdobný jako v případě 2.

Z relace přechodu mezi konfiguracemi $\vdash_M$ lze odvodit mocninu $\vdash_M^n$ a reflexivně-tranzitivní uzávěr $\vdash_M^*$. Posloupnost konfigurací $C_0, C_1, \ldots, C_n (n \geq 0)$ takových, že platí

$$C_0 \ \vdash_M \ C_1 \ \vdash_M \ C_2 \ \vdash_M \ \ldots \ \vdash_M \ C_n,$$

nazýváme **výpočtem délky** n Turingova stroje M.

Definice 11.14: Nechť $M = \langle Q, A, \delta, s, F \rangle$ je Turingův stroj a nechť $F = \{y, n\}$ sestává ze dvou koncových stavů (y/n jako yes/no). Libovolná koncová konfigurace, jejíž stav je roven y se nazývá **přijímající konfigurací**, zatímco koncová konfigurace se stavem rovným n se nazývá **odmítající konfigurací**. Předpokládejme, že v počáteční konfiguraci bude mít páska Turingova stroje obsah $\triangleright \sqcup w$, kde w je slovo nad nějakou pevně zvolenou tzv. **vstupní abecedou** $A_0 \subseteq (A - \{\sqcup, \triangleright\})$. Říkáme, že

- M **přijímá** vstup $w \in A_0^*$, pokud platí $(s, \triangleright\underline{\sqcup}w) \vdash_M^* (y, v)$ pro nějaké $v \in A^*$
- M **odmítá** vstup $w \in A_0^*$, pokud platí $(s, \triangleright\underline{\sqcup}w) \vdash_M^* (n, v)$ pro nějaké $v \in A^*$
- M **rozhoduje jazyk** $L \subseteq A_0$ nad vstupní abecedou A_0, pokud pro libovolný vstupní řetěz $w \in A_0$ platí: je-li $w \in L$, pak M přijímá vstup w a je-li $w \notin A_0$, pak M odmítá vstup w.
- jazyk L je **rekurzivní**, pokud existuje nějaký Turingův stroj, který jej rozhoduje.

Sledování jednotlivých konfigurací a přechodů mezi nimi představuje ten nejdetailnější pohled na činnost Turingových strojů. Přes jednoduchost základních kroků lze jen obtížně z tabulky přechodů odvodit intuitivní smysl chování určitého stroje. Stejně těžko si lze představit, že bychom mohli přímo „programovat“ nějaký Turingův stroj tak, aby prováděl nějakou náročnější operaci, např. aby rozhodoval zvolený jazyk. Tyto obtíže se ovšem dají do značné míry zmenšit použitím vhodné techniky vytváření Turingových strojů, jak ukážeme dále. Existují ovšem daleko zásadnější problémy, se kterými se nelze úspěšně vypořádat. Tak např. pro zadaný Turingův stroj a zadaný jazyk není možné určit, zda stroj jazyk rozhoduje. Jak ukážeme později v textu, není navíc ani možné zjistit, zda se stroj pro určité konkrétní vstupní slovo zastaví či nikoliv.

Cvičení

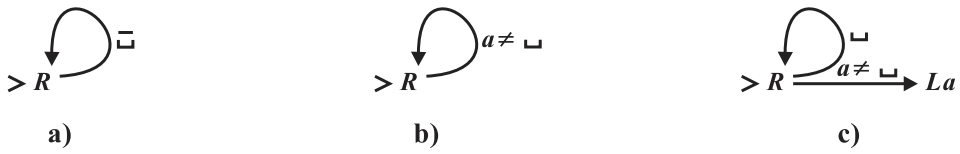
11.2-1. Mějme Turingův stroj $M = \langle Q, A, \delta, s, \{h\} \rangle$, kde $Q = \{s, r, h\}$, $A = \{0, 1, \sqcup, \triangleright\}$ a přechodové zobrazení δ je dáno následující tabulkou.

q / a	0	1	$\sqcup$	$\triangleright$
s	$(r, 1)$	$(r, 0)$	$(h, \sqcup)$	$(s, \rightarrow)$
r	$(s, \rightarrow)$	$(s, \rightarrow)$	$(s, \rightarrow)$	$(r, \rightarrow)$

(a) Simulujte výpočet Turingova stroje M z počáteční konfigurace $(s, \triangleright\underline{0}01110)$.
(b) Neformálně popište, co dělá M po odstartování ve stavu s nad libovolným políčkem.

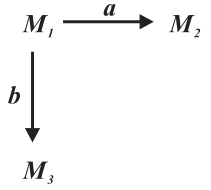
11.2.2 Zjednodušený popis Turingových strojů

Abychom mohli snáze konstruovat Turingovy stroje schopné provádět komplikovanější operace v rámci jejich výpočetních možností, zvolíme postup, který pomáhá zvládat složitost v mnoha oblastech: vytvoříme stroje schopné provádět nějaké jednoduché, užitečné a srozumitelně definované operace. Tyto stroje pak dovolíme skládat podle určitých pravidel do strojů složitějších, a tak budeme dostávat stroje stále mocnější. Za **základní stroje** můžeme jistě



Obrázek 11.8: Notace pro zápis Turingových strojů

považovat takové, které jsou schopné zapsat na pásku určitý symbol, a stroje zajišťující pohyb čtecí/zapisovací hlavy jedním směrem.



Obrázek 11.7: Skládání Turingových strojů

Předpokládejme tedy, že abeceda A je pevně stanovena, takže pro každé $a \in A - \{\triangleright\}$ definujeme Turingův stroj $M_a = \langle \{s, h\}, A, \delta, s, \{h\} \rangle$ takový, že platí $\delta(s, b) = (h, a)$ pro každé $b \in A - \{\triangleright\}$ (a přirozeně také $\delta(s, \triangleright) = (h, \rightarrow)$). Stroj M_a tedy provede jedinou věc: přepíše symbol ve čteném políčku na a a zastaví se. Protože zápis určitého znaku je velmi častou operací, budeme namísto označení M_a (pokud nehrozí nedorozumění) psát stručněji pouze a.

Podobně jednoduché budou Turingovy stroje $M_{\leftarrow}$ a $M_{\rightarrow}$, které posunou čtecí/zapisovací hlavu stroje o jedno místo v požadovaném směru a pak se zastaví. Pro jednoduchost budeme Turingovy stroje $M_{\leftarrow}$, resp. $M_{\rightarrow}$ označovat zkráceně jako L (left) a R (right).

Pro **skládání Turingových strojů** zavedeme vyjádření podobné přechodovým diagramům konečných automatů (viz obr. 11.7), které budeme nazývat **kompozičním diagramem** nebo prostě jen **diagramem**. Uzly kompozičního diagramu vyjadřují jednotlivé dílčí Turingovy stroje. Propojení uzlu M_1 s uzlem M_2 pomocí hrany ohodnocené symbolem $a \in A$ znamená, že pokud po skončení činnosti stroje M_1 je na čteném políčku symbol a, bude výpočet dále pokračovat podle Turingova stroje M_2. Pokud bude na čteném políčku symbol b, výpočet pokračuje podle Turingova stroje M_3. Jakýkoliv jiný symbol na čteném políčku po zastavení stroje M_1 způsobí celkové zastavení.

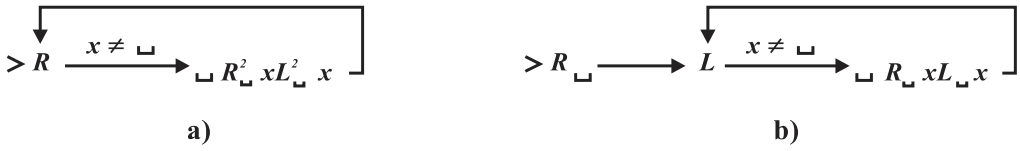
Není těžké ověřit, že úplný formální popis Turingových strojů vzniklých podobným skládáním lze snadno získat z popisu použitých komponent a jejich propojení v kompozičním diagramu (viz cvičení 11.2-3).

Při vytváření kompozičních diagramů budeme používat některé další zkratky. Namísto rovnoběžných hran mezi uzly můžeme symboly odpovídající těmto hranám uvést oddělené čárkou u hrany jediné. Jestliže pokračování činnosti vyjádřené hranou následuje pro libovolný čtený symbol, pak můžeme ohodnocení hrany zcela vynechat, příp. vynecháme i danou hranu a označení odpovídajících Turingových strojů jen zapíšeme za sebou (zleva doprava). Tak např. $R \longrightarrow R$ vyjadřující dva posuny čtecí/zapisovací hlavy doprava budeme psát jako RR nebo dokonce R^2.

výjimkou jediného, je možné tento požadavek vyjádřit zjednodušeně ve tvarech uvedených na obr. 11.8 a),b). Hodnotu symbolu, pro který se daný přechod uskutečnil, můžeme využít dále v diagramu způsobem, který ukazuje příklad na obr. 11.8 c). Turingův stroj na tomto obrázku postupuje od aktuálního políčka doprava tak dlouho, až narazí na první nemezerový znak. Potom se hlava posune nad sousední místo vlevo a zapíše do něj dotyčný nemezerový znak.

Další jednoduché Turingovy stroje jsou vyjádřeny diagramy na obr. 11.9 a) až d):

Obrázek 11.9: Příklady jednoduchých Turingových strojů



Obrázek 11.10: Příklady Turingových strojů

- $R_{\sqcup}$ nalezne první mezeru vpravo od aktuálně čteného místa
- $L_{\sqcup}$ nalezne první mezeru vlevo od aktuálně čteného místa
- $R_{\overline{\sqcup}}$ nalezne první nemezerový znak vpravo od aktuálně čteného místa
- $L_{\overline{\sqcup}}$ nalezne první nemezerový znak vlevo od aktuálně čteného místa

Pomocí popsaných komponent složíme nyní Turingovy stroje realizující poněkud komplikovanější operace. Na obr. 11.10a) je znázorněn „kopírující“ Turingův stroj. Předpokládáme, že tento stroj začne pracovat v počáteční konfiguraci $(s, \triangleright\underline{\sqcup}w\sqcup)$, kde w je libovolný řetěz neobsahující žádnou mezeru. Stroj se zastaví v koncové konfiguraci $(h, \triangleright \sqcup w\underline{\sqcup}w\sqcup)$, řetěz w tedy bude na pásce zkopírován.

Stroj se nejprve posune vpravo a pamatuje[4] si čtený nemezerový symbol v proměnné x. Tento symbol přepíše mezerou a pomocí $R^2_{\sqcup}$ vyhledá druhou mezeru vpravo, kterou přepíše hodnotou proměnné x. Pak pomocí $L^2_{\sqcup}$ vyhledá druhou mezeru vlevo (tj. políčko, odkud se četl posledně zapisovaný symbol) a zapíše místo ní původní symbol. Činnost pokračuje opět od začátku, tedy posunem vpravo a čtením nemezerového symbolu. Pokud se ovšem četla mezera (mezi dvěma kopiemi řetězu w), stroj se zastaví.

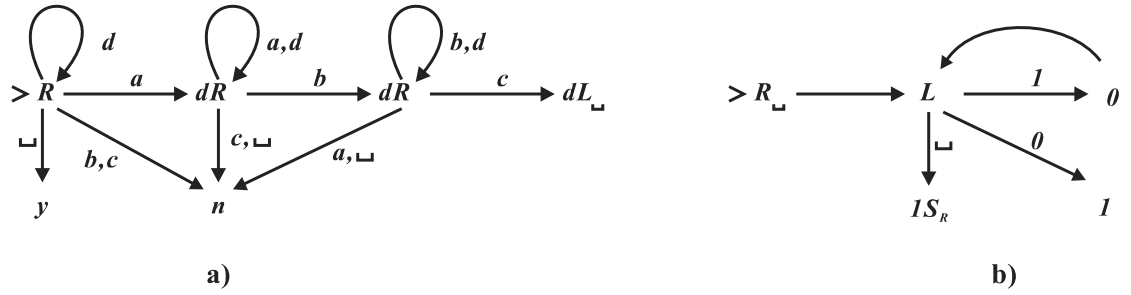
Podobným způsobem pracuje i reverzující stroj na obr. 11.10b), který vstupní obsah pásky $\triangleright\underline{\sqcup}w\sqcup$ transformuje na $\triangleright\underline{\sqcup}ww^R\sqcup$. Nejprve se přesune na první mezeru vpravo, tedy na mezeru za řetězem w. Nyní se posune vlevo, čtený nemezerový symbol si pamatuje v proměnné x a přepíše políčko mezerou. Pak vyhledá první mezeru vpravo a přepíše ji hodnotou proměnné x. Vrací se vlevo a první mezeru (v místě, kde četl x) přepíše zpět na původní hodnotu. Pokračuje opět od posunu vlevo a čtení nemezerového symbolu, pokud se již četla mezera (vlevo od původního retězu w), stroj se zastaví.

Příklad Turingova stroje rozhodujícího nějaký jazyk ukazujeme na obr. 11.11a). Uvedený stroj rozhoduje jazyk $L = \{a^n b^n c^n \ : \ n \geq 0\}$ a postupuje přitom podle následující strategie: Nejprve nalezne první symbol a zleva (přitom přeskočí případné pomocné symboly d, jejichž význam vyplyne z následujícího). Symbol a nahradí symbolem d, přeskočí všechny případné symboly a a d a hledá první symbol b zleva. Ten také nahradí symbolem d a přeskokem případných symbolů b a d hledá první symbol c zleva, který nahradí rovněž symbolem d. Pak se jeho čtecí/zapisovací hlava vrátí před levý okraj zpracovávaného slova a celý postup se opakuje.

Každým postupem zleva doprava tedy stroj nahradí po jednom symbolu a, b, c symbolem d, a takto pokračuje, dokud se nevyčerpají všechny symboly a, b, c současně (pak je slovo přijato), nebo ještě nějaký symbol zbývá (pak je slovo odmítnuto).

Definice 11.15: Nechť $M = \langle Q, A, \delta, s, \{h\}\rangle$ je Turingův stroj, $A_0 \subseteq (A - \{\triangleright, \sqcup\})$ zvolená vstupní abeceda a $w \in A_0^*$ slovo nad touto abecedou. Nechť se M zastaví pro vstup w a platí

[4] Pamatování čtených symbolů v proměnných je operace, kterou podle definice není Turingův stroj vybaven. Lze však ukázat, že tuto operaci jde vyjádřit standardními prostředky (viz cvič. 11.2-4)



Obrázek 11.11: Turingův stroj a) rozhodující jazyk $\{a^n b^n c^n\}$, b) počítající funkci následníka

$(s, \triangleright\underline{\sqcup}w) \vdash_M^* (h, \triangleright\underline{\sqcup}z)$ pro nějaké $z \in A_0^*$. Pak z nazýváme **výstupem** M na vstup w a značíme $M(w)$ [5]. Mějme nyní libovolné zobrazení f z A_0^* do A_0^*. Říkáme, že M **počítá zobrazení** f, pokud pro libovolné $w \in A_0^*$ platí $M(w) = f(w)$. Zobrazení f nazýváme **rekurzivní**, pokud existuje nějaký Turingův stroj M, který počítá f.

Turingův stroj může ovšem počítat i číselné funkce s definičním oborem i oborem hodnot na množině přirozených čísel **N**. K tomu zcela postačí předpokládat, že vstup i výstup je na pásce vhodně zakódován, např. ve dvojkové soustavě s jednotkovým řádem nejvíce vpravo. Pro funkce více proměnných lze navíc použít nějakou konvenci, která dovolí od sebe na vstupu oddělit hodnoty jednotlivých argumentů – k tomu může sloužit např. znak ';' (středník).

Příklad 11.16: Ukážeme, jakým způsobem je možné zkonstruovat Turingův stroj počítající funkci následník, tedy $s(n) = n+1$. Vycházíme z toho, jak probíhá přičtení jedničky ve dvojkové soustavě. Nejprve je třeba čtecí/zapisovací hlavu posunout vpravo za nejnižší řád vstupního čísla. Pak se hledá první nulová číslice zprava, přičemž čtené jedničky se přepisují na nuly. První nalezená nula se naopak nahradí jedničkou, pokud se žádná nula nenalezla, je třeba na výstupu přidat jedničku v nejvyšším řádu. Popsaný postup je realizován Turingovým strojem na obr. 11.11b).

Striktně vzato není ještě hledaný Turingův stroj hotov – je totiž třeba zařídit správnou koncovou polohu čtecí/zapisovací hlavy. Navíc, v případě přidání jedničky do nejvyššího řádu, kterou se přepíše úvodní mezera, musíme ještě výsledek posunout o jedno místo vpravo. Tyto doplňky však již ponecháme jako cvičení. △

Definice 11.17: Nechť $M = \langle Q, A, \delta, s, F\rangle$ je Turingův stroj, $A_0 \subseteq (A - \{\sqcup, \triangleright\})$ zvolená vstupní abeceda a $L \subseteq A_0^*$ jazyk nad touto abecedou. Říkáme, že M **polorozhoduje** L, pokud pro libovolný vstup $w \in A_0^*$ platí: $w \in L$ právě když se M zastaví pro vstup w. Jazyk L se nazývá **rekurzivně vyčíslitelný**, právě když existuje Turingův stroj M, který polorozhoduje L.

Oproti Turingovým strojům, které rozhodují příslušnost slova do jazyka podle stavu dosaženého v koncové konfiguraci, polorozhodující stroje dávají najevo negativní výsledek tak, že svůj výpočet nikdy neukončí. Je zřejmé, že z každého Turingova stroje, který rozhoduje nějaký jazyk, můžeme snadnou úpravou získat stroj, který polorozhoduje tentýž jazyk – stačí změnit koncový stav n tak, aby se v něm stroj zacyklil. Dokázali jsme tak základní vztah mezi rekurzivními a rekurzivně vyčíslitelnými jazyky obsažený v první části následujícího tvrzení.

Věta 11.18: Nechť L je rekurzivní jazyk. Potom platí:

[5] Povšimněme si, že výstup $z = M(w)$ je definován pouze tehdy, když se M zastaví pro vstup w, a to v koncové konfiguraci $(h, \triangleright\underline{\sqcup}z)$ pro $z \in A_0^*$.

- L je rekurzivně vyčíslitelný
- jeho doplněk $\overline{L}$ je rekurzivní.

Důkaz: Zbývá dokázat pouze druhou část tvrzení. Turingův stroj rozhodující doplněk jazyka L získáme ze stroje rozhodujícího L prostou záměnou rolí koncových stavů y a n. △

Nabízí se otázka, zda třída rekurzivně vyčíslitelných jazyků není vlastně shodná s třídou jazyků rekurzivních. Jak ukážeme v odstavci 11.3, tyto třídy však shodné nejsou.

Cvičení

11.2-2. Navrhněte Turingův stroj, který rozhoduje jazyk generovaný bezkontextovou gramatikou $G = \langle\{0,1\},\{S\},P,S\rangle$ s přepisovacími pravidly

$$P = \{S \to 01,\ S \to 0S1,\ S \to SS\}$$

Návrh proveďte nejprve přímo, tj. stanovením všech složek Turingova stroje včetně přechodového zobrazení, a potom stroj vyjádřete vhodným kompozičním diagramem.
(*Návod*: Využijte toho, že se jedná o jazyk správně strukturovaných závorek – 0 představuje otvírací a 1 zavírací závorku. Lze postupovat např. tak, že se nalezne první 1 zleva, a k ní pak postupem vlevo odpovídající 0, přičemž se vždy oba takto zpracované znaky na pásce nahrazují nějakým neutrálním znakem.)

11.2-3. Nechť jsou Turingovy stroje z obr. 11.7 zadány jako

$$M_1 = \langle Q_1, A, \delta_1, s_1, H_1\rangle,\quad M_2 = \langle Q_2, A, \delta_2, s_2, H_2\rangle,\quad M_3 = \langle Q_3, A, \delta_3, s_3, H_3\rangle.$$

Pomocí složek těchto strojů definujte Turingův stroj $M = \langle Q, A, \delta, s, H\rangle$, který odpovídá kompozičnímu diagramu uvedenému na tomto obrázku.
(*Návod*: Lze předpokládat, že množiny stavů jsou disjunktní, položíme tedy $Q = Q_1 \cup Q_2 \cup Q_3$, $s = s_1$, $H = H_2 \cup H_3$. Nyní již zbývá definovat jen přechodové zobrazení δ tak, aby odpovídalo předpokládanému chování Turingova stroje daného kompozičním diagramem.)

11.2-4. Ukažte, jak lze standardními prostředky (tj. bez uchování hodnoty čteného symbolu v proměnné x) vyjádřit Turingův stroj na obr. 11.8 c). Předpokládejte, že abeceda tohoto stroje je $A = \{a, b, \sqcup, \triangleright\}$.
(*Návod*: Rozepište do paralelních přechodových větví všechny možnosti, pro něž se ukončí postup vpravo.)

11.2-5. Určete, zda Turingovy stroje vyjádřené kompozicemi LR a RL jsou co do své činnosti zcela ekvivalentní.

11.2-6. Navrhněte Turingův stroj s abecedou $A = \{a, b, \sqcup, \triangleright\}$, který na pásce nalezne první výskyt dvojice aa, a pak se zastaví. Určete předpokládanou počáteční konfiguraci a popište činnost vašeho stroje v dalších mezních případech.

11.2-7. Navrhněte Turingův stroj $S_{\leftarrow}$, který posune řetěz na pásce o jedno místo vlevo. Na začátku jeho činnosti předpokládáme konfiguraci $(q, \triangleright\alpha\underline{\sqcup}aw\sqcup)$, $\alpha, w \in A^*$, $a \in A$, na konci pak $(h, \triangleright\alpha\underline{a}w\sqcup)$. S pomocí tohoto stroje sestavte diagram stroje, který z daného slova vypustí všechny výskyty symbolu b.

11.2-8. Navrhněte Turingův stroj, který počítá funkci reverzace slova $f(w) = w^R$ nad abecedou $\{a, b\}$.
(*Návod*: Inspirujte se Turingovým strojem na obr. 11.10b).)

11.2-9. Sestavte co nejjednodušší Turingovy stroje, které rozhodují jazyky nad abecedou $\{a, b\}$:
(a) $\emptyset$, (b) $\{\epsilon\}$, (c) $\{a\}$, (d) $\{a\}^*$

11.2.3 Některá zobecnění Turingových strojů

Nelze vyloučit, že některé složky v definici Turingových strojů jsou zadány příliš omezujícím způsobem. Mohli bychom tedy očekávat, že se výpočetní možnosti Turingových strojů rozšíří, pokud vhodně upravíme jejich definici. Uvedeme nyní několik možností takových zdánlivě zobecňujících úprav, na nichž ukážeme, že odpovídající zobecněný typ Turingova stroje může být simulován standardním modelem.

Vícepáskové Turingovy stroje

Je snadno možné si představit Turingův stroj vybavený ne jednou, ale několika páskami. Každá páska má svoji čtecí/zapisovací hlavu a stroj pracuje tak, že podle kombinace symbolů čtených v daném stavu všemi hlavami přejde do dalšího stavu a posune některé hlavy vlevo, jiné vpravo a zbývající ponechá na místě s přepsáním čteného symbolu. Strukturu takového stroje vyjádříme následující definicí.

Definice 11.19: Nechť $k \geq 1$ je přirozené číslo. **k-páskovým Turingovým strojem** nazýváme pětici $M = \langle Q, A, \delta, s, F\rangle$, kde Q, A, s a F jsou jako v definici standardního Turingova stroje a

$$\delta \ : \ (Q - F) \times A^k \ \mapsto \ Q \times (A \cup \{\leftarrow, \rightarrow\})^k$$

je přechodové zobrazení. Pro libovolný stav $q \in Q-F$ a libovolnou k-tici symbolů $(a_1, a_2, \ldots, a_k)$ tedy $\delta(q, (a_1, a_2, \ldots, a_k)) = (p, (b_1, b_2, \ldots, b_k)$ znamená, že p je nový stav a každé b_j vyjadřuje akci čtecí/zapisovací hlavy stroje M na j-té pásce. Opět předpokládáme, že nelze přepsat ani překročit levý okraj žádné pásky, tedy je-li $a_j = \triangleright$ pro nějaké $j \leq k$, potom je $b_j = \rightarrow$.

Protože výpočet k-páskového Turingova stroje probíhá s využitím všech k pásek, musí obsah všech těchto pásek brát v úvahu definice jeho konfigurace. Ta bude zahrnovat aktuální stav a k-tici obsahů jeho pásek s rozlišením polohy všech čtecích/zapisovacích hlav. Každá konfigurace je tedy prvkem množiny

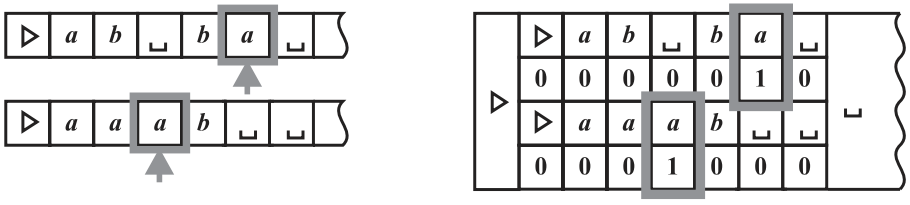
$$K \ \times \ (\triangleright A^* \times (A^*(A - \{\sqcup\}) \cup \{\epsilon\}))^k$$

Pro přehlednější vyjádření konfigurace použijeme opět zápisu s podtrženým symbolem označujícím polohu čtecí/zapisovací hlavy, tedy $(q, (w_1\underline{a_1}u_1, \ldots, w_k\underline{a_k}u_k))$. Je-li $\delta(q, (a_1, a_2, \ldots, a_k)) = (p, (b_1, b_2, \ldots, b_k)$, pak M přejde v jediném kroku na konfiguraci $(p, (w'_1\underline{a'_1}u'_1, \ldots, w'_k\underline{a'_k}u'_k))$, kde pro každé $i = 1, \ldots, k$ vznikne $w'_i\underline{a'_i}u'_i$ na i-té pásce z $w_i\underline{a_i}u_i$ úpravou odpovídající akci b_i přesně v duchu definice 11.13.

Definice funkce počítané k-páskovým Turingovým strojem M podobně jako definice jazyka rozhodovaného či polorozhodovaného M je koncipována obdobně jako u standardního modelu. Přijmeme pouze úmluvu, že vstup je vždy zaznamenán na první pásce stejným způsobem, jako tomu bylo u standardního modelu, ostatní pásky jsou prázdné. Všechny hlavy jsou na počátku umístěny nad první mezerou zleva. Po skončení výpočtu je výsledek uložen opět na první pásce, obsah ostatních se ignoruje.

Návrh k-páskového stroje provádějícího požadovaný algoritmus může být jednodušší než pro stroj disponující pouze jedinou páskou. Jak jsme již naznačili výše, tento fakt neznamená větší operační možnosti, ale jen možnost lepší časové složitosti práce k-páskového stroje, což ukazuje následující tvrzení.

Věta 11.20: Nechť $M = \langle Q, A, \delta, s, F\rangle$, je k-páskový Turingův stroj pro $k \geq 1$. Potom existuje standardní Turingův stroj $M' = \langle Q', A', \delta', s', F\rangle$, kde $A \subseteq A'$ a pro který platí: Pro libovolné slovo $x \in A^*$ se M zastaví pro vstup x s výstupem y na první pásce, právě když se M' pro vstup x zastaví ve stejném koncovém stavu a se stejným výstupem y na pásce. Navíc, jestliže se M pro vstup x zastaví po n krocích, pak se M' pro vstup x zastaví po $O(n.(|x| + n))$.



Obrázek 11.12: Simulování k-páskového Turingova stroje

Místo důkazu předchozí věty ukážeme jen základní myšlenky vedoucí ke konstrukci simulujícího Turingova stroje M'. Původní páskovou abecedu stroje M rozšíříme pro M' tak, aby se mohla libovolná část jeho (jediné!) pásky chápat jako $2k$ rovnoběžných stop. Korespondenci mezi původními k páskami a jedinou „vícestopou“ páskou stroje M' ukazuje obr. 11.12 pro případ dvou pásek. Stopa s lichým pořadovým číslem $2i-1$ je „datová“ – její obsah koresponduje s obsahem i-té pásky. Stopa se sudým pořadovým číslem $2i$ je „řídicí“ a polohou své jediné jedničky určuje umístění čtecí/zapisovací hlavy na i-té pásce.

Simulace výpočtu stroje M pro vstup $w \in A^*$ na stroji M' proběhne ve třech etapách:

1. Vstup w na pásce se nejprve převede do vícestopé reprezentace odpovídající uložení w na první pásce stroje M a vynulování ostatních $k-1$ pásek.
2. Simulují se jednotlivé kroky k-páskového stroje M až do jeho (případného) zastavení. Každému kroku stroje M odpovídá následující posloupnost operací stroje M', o níž předpokládáme, že vždy začíná s čtecí/zapisovací hlavou umístěnou na první mezeře vpravo za tou částí pásky, která je rozdělena na stopy.

   (a) Prochází se doleva a sbírá se (pomocí změn vnitřního stavu) informace o symbolech čtených hlavami jednotlivých pásek. Při tomto postupu se neprovádí žádný zápis na pásku a procházení skončí nalezením mezery před levým okrajem pásky.

   (b) Prochází se doprava a provádějí se akce odpovídající přechodu stroje M. To znamená, že se v každé dvojici datová/řídicí stopa provede buď přepsání čteného symbolu v datové stopě, nebo přesun jedničky ve stopě řídicí.
3. Když skončí předchozí etapa, M' nejprve převede „vícestopý“ obsah své pásky do standardního formátu podle obsahu první stopy, umístí čtecí/zapisovací hlavu tam, kde by ji byl měl M, a zastaví se ve stejném stavu, kde by byl skončil M.

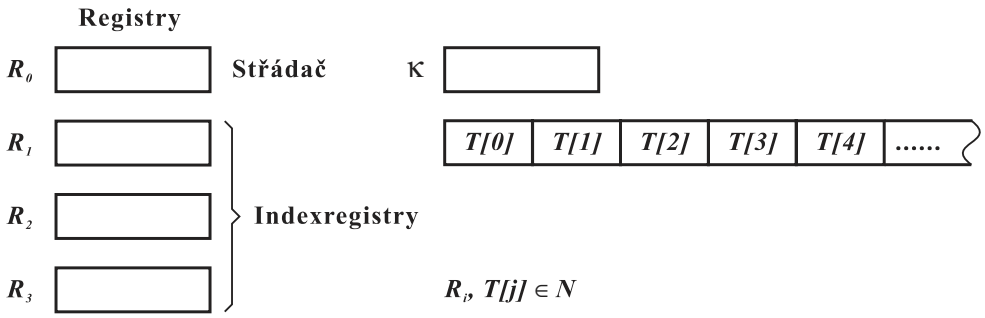
Stroj s oboustranně nekonečnou páskou

Předpokládejme nyní, že náš stroj má pásku, která je nekonečná v obou směrech. Všechna políčka na pásce jsou prázdná s výjimkou těch, která obsahují vstupní slovo, čtecí/zapisovací hlavu předpokládáme umístěnou vlevo od vstupu.

Takto koncipovanou variantu Turingova stroje lze snadno simulovat dvoupáskovým Turingovým strojem: jedna páska představuje obsah původní pásky počínaje prvním vstupním symbolem, druhá páska reprezentuje (v opačném pořadí) obsah levé poloviny původní pásky. Dvoupáskový stroj už dokážeme simulovat standardním modelem, v tomto případě to bude dokonce s lineární časovou složitostí. Při simulaci stroje s $k \geq 1$ oboustranně nekonečnými páskami bychom postupovali podobně.

Stroj s více hlavami

Mějme nyní Turingův stroj s jedinou páskou, nad níž se (nezávisle na sobě) pohybuje několik čtecích/zapisovacích hlav. Předpokládejme, že jsou adekvátně vyřešeny situace, kdy by různé hlavy nacházející se na stejném políčku chtěly čtený symbol přepsat různě (např. může platit, že hlava s nejnižším číslem má přednost). Hlavy ovšem nemají přímou možnost detekovat, zda se na jejich políčku pásky nachází současně jiná hlava.



Obrázek 11.13: Turingův stroj s přímým přístupem

Takový stroj bychom mohli simulovat podobně, jako jsme to udělali u k-páskového Turingova stroje. Pro stroj s k hlavami bychom rozdělili pásku na $k+1$ stop: jedna by byla datová a dalších k stop by zachycovalo polohu jednotlivých hlav. Simulace jednoho kroku stroje s více hlavami by pak proběhla ve dvou průchodech nad páskou – při prvním průchodu by se zjistily symboly čtené jednotlivými hlavami, při druhém by se provedly náležité přepisy políček pásky a/nebo pohyb hlav. Tato simulace by tedy proběhla opět v kvadratickém čase v závislosti na počtu kroků původního stroje.

Stroj s dvojrozměrnou páskou

Další potenciální zobecnění přináší možnost, aby Turingův stroj pracoval s dvojrozměrnou mřížkou namísto jednorozměrné pásky, případně s mřížkou vyšší dimenze. Takový požadavek může vzniknout při řešení různých typů skládaček apod. Je ovšem zapotřebí nejprve přesně definovat chování takového stroje – namísto pohybu hlavy vlevo/vpravo zde bude možnost pohybu vlevo/vpravo/nahoru/dolů, vždy jen o jedno místo. Dále se musí zajistit, aby se stroj nedostal mimo mřížku, což lze udělat obdobně jako u standardního Turingova stroje.

I v tomto případě lze činnost takového stroje simulovat standardním modelem, časová složitost simulace je opět vyjádřena nějakým polynomem v proměnných t (počet kroků zobecněného stroje) a $|x|$ (délka vstupu). Popsaná zobecnění lze ovšem také libovolně kombinovat, takže si můžeme např. představit stroj s oboustranně nekonečnou dvojrozměrnou mřížkou a k hlavami, atd.

Turingův stroj s náhodným přístupem

Předchozí zobecňující pokusy se zdají potvrzovat, že standardní model Turingova stroje byl definován jako dostatečně mocný výpočetní prostředek. Všechna dosavadní rozšíření konceptu Turingova stroje ovšem zachovávala jeho základní nevýhodu oproti současným počítačům, a sice sekvenční charakter jeho paměti. Aby stroj zjistil obsah nějakého políčka pásky, musí projít všemi políčky mezi aktuálně čteným a požadovaným políčkem. Běžné počítače však mají paměť s náhodným přístupem, což dovoluje pracovat s obsahem buňky poté, co byla zadána její adresa.

Abychom vybavili Turingův stroj schopností přístupu k libovolnému políčku v jediném kroku, doplníme jeho architekturu o několik registrů schopných pamatovat si adresy políček pásky (viz obr. 11.13). Současně stanovíme i jeho operační možnosti jako určité minimum toho, na co jsme zvyklí u reálných počítačů.

Turingův stroj s náhodným přístupem má pevný počet registrů (jedním z nich je čítač instrukcí – PC) a jednostranně nekonečnou pásku. Každé políčko pásky a každý registr je schopen uložit libovolné přirozené číslo. Stroj pracuje s registry a páskou podle pevného programu, který je obdobou přechodového zobrazení u standardního modelu. Program je tvořen posloupností instrukcí, jejichž přípustné typy a zápis ukazuje následující tabulka. Symbol $j (0 \leq j < k)$ v instrukci určuje číslo registru, $T[i]$ označuje i-té políčko na pásce, R_j označuje j-tý registr,

$s \leq n$ je pořadové číslo nějaké isntrukce v programu, c je libovolné přirozené číslo. Všechny instrukce zvyšují hodnotu čítače PC o jedničku, pokud není explicite stanoveno jinak.

kód	**popis operace**	**kód**	**popis operace**
read j	$R_0 := T[j]$	sub j	$R_0 := max\{R_0 - R_j, 0\}$
write j	$T[j] := R_0$	sub $=c$	$R_0 := max\{R_0 - c, 0\}$
store j	$R_j := R_0$	half	$R_0 := R_0$ **div** 2
load j	$R_0 := R_j$	jump s	$PC := s$
load $=c$	$R_0 := c$	jpos s	**if** $R_0 > 0$ **then** $PC := s$
add j	$R_0 := R_0 + R_j$	jzero s	**if** $R_0 = 0$ **then** $PC := s$
add $=c$	$R_0 := R_0 + c$	halt	$PC := 0$

Tabulka instrukcí Turingova stroje s náhodným přístupem

Intuitivně lze charakterizovat architekturu a instrukční soubor tohoto Turingova stroje tak, že všechny jeho registry R_j mohou být využívány jako indexregistry (při adresaci pásky) nebo operandové registry, R_0 funguje navíc jako střadač pro aritmetické operace a jako datový registr pro operace s páskou.

Definice 11.21: Turingův stroj s náhodným přístupem je určen dvojicí $M = \langle k, P\rangle$, kde $k > 0$ je počet **registrů** a $P = (p_1, p_2, \ldots, p_n)$ je **program**, tedy konečná posloupnost instrukcí, v němž každá instrukce p_i je tvaru podle tabulky uvedené výše. Navíc platí, že poslední instrukce p_n je vždy *halt* (nemusí to však být jediné *halt* v programu). **Konfigurací** Turingova stroje $\langle k, P\rangle$ s náhodným přístupem nazýváme $(k+2)$-tici $(i; R_0, R_1, \ldots, R_{k-1}; T)$, kde

- $i \in \mathbf{N}, 0 \leq i \leq n$ je aktuální hodnota čítače instrukcí (konfigurace se nazývá **koncovou konfigurací**, pokud je $i = 0$)
- $R_j \in \mathbf{N}$ pro každé $0 \leq j < k$ je aktuální hodnota j-tého registru
- T je obsah pásky určený konečnou množinou uspořádaných dvojic přirozených čísel, tedy $T \subset (\mathbf{N} - \{0\}) \times (\mathbf{N} - \{0\})$ takových, že pro libovolné $t \geq 1$ obsahuje T nejvýše jednu dvojici (t, m) (t-té políčko pásky obsahuje hodnotu m), pro políčka neuvedená v T se předpokládá obsah rovný nule.

Budeme předpokládat, že Turingův stroj s náhodným přístupem zahajuje svoji činnost v **počáteční konfiguraci** tvaru $(1; 0, 0, \ldots, 0; \emptyset)$, tedy čítač instrukcí ukazuje na první instrukci programu a všechny registry i páska jsou vynulovány.

Definice 11.22: Nechť $M = \langle k, P\rangle$ je Turingův stroj s náhodným přístupem. Říkáme, že konfigurace $C = (i; R_0, R_1, \ldots, R_{k-1}; T)$ **přejde na stroji** M na konfiguraci $C' = (i;, R'_0, R'_1, \ldots, R'_{k-1}; T')$ a značíme $C \vdash_M C'$, pokud hodnoty i', R'_j a T odpovídají (v duchu sémantiky instrukcí uvedené dříve) provedení instrukce p_i za výchozího stavu určeného hodnotou i čítače instrukcí, hodnotami R_j uloženými v registrech a obsahem pásky T.

Pro ilustraci uvedeme pro několik typů instrukcí jejich sémantiku rozepsanou do účinku na jednotlivé složky konfigurace. Pro ostatní typy instrukcí by byl rozpis obdobný.

instrukce	**hodnota konfigurace po přechodu**
read r	$i' = i + 1$
	if $(R_r, m) \in T$ **then** $R'_0 = m$ **else** $R'_0 = 0$
	$R'_j = R_j$ pro $j = 1, \ldots, k-1$
	$T' = T$
write r	$i' = i + 1$
	$R'_j = R_j$ pro $j = 0, 1, \ldots, k-1$

	if $R_0 \neq 0$ **then** $T' = (T - \{(R_r, *)\} \cup \{(R_r, R_0)\}$ **else** $T' = T - \{(R_r, *)\}$
add $=c$	$i' = i + 1$ $R'_0 = R_0 + c$ $R'_j = R_j$ pro $j = 1, \ldots, k-1$ $T' = T$
jpos s	(předpokládáme $1 \leq s \leq n$) **if** $R_0 > 0$ **then** $i' = s$ **else** $i' = i + 1$ $R'_j = R_j$ pro $j = 0, 1, \ldots, k-1$ $T' = T$

Pro relaci přechodu mezi konfiguracemi $\vdash_M$ na Turingově stroji s náhodným přístupem M se obvyklým způsobem definuje její tranzitivní uzávěr $\vdash_M^+$, resp. reflexivně-tranzitivní uzávěr $\vdash_M^*$.

Příklad 11.23: Pro ilustraci sestavíme program pro Turingův stroj s náhodným přístupem, který počítá největšího společného dělitele dvou (kladných) přirozených čísel. Vzhledem k omezené množině instrukcí nemůžeme použít klasického Eukleidova algoritmu s výpočtem zbytku po dělení, musíme tedy pracovat pouze s odčítáním. Algoritmus vyjádříme nejprve formou následujícího pseudokódu:

```
GCD(a, b)
1  while a ≠ b do                Opakuj dokud jsou různá
2      if  a > b
3          then a := a − b       zmenšíme a
4          else b := b − a       nebo zmenšíme b
5  return a
```

Předpokládejme nyní, že hodnoty a, b jsou při zahájení činnosti programu uloženy v registrech R_1, R_2 a při ukončení programu bude výsledek uložen v registru R_0. Přepis pseudokódu do instrukcí Turingova stroje s náhodným přístupem by pak mohl vypadat (s trochou optimalizace) takto:

1.	load 1	$R_0 := a$
2.	sub 2	R_0 obsahuje rozdíl $a - b$
3.	jpos 9	skok na zmenšování a na hodnotu $a - b$
4.	load 2	$R_0; = b$
5.	sub 1	R_0 obsahuje rozdíl $b - a$
6.	jzero 11	teprve teď víme, že pro $a = b$, skok na konec
7.	store 2	$b := b - a$
8.	jump 1	návrat na začátek cyklu
9.	store 1	$a := a - b$ neboť R_0 obsahuje $a - b$
10.	jump 1	návrat na začátek cyklu
11.	load 1	nastavení R_0 na výsledek
12.	halt	zastavení

Příklad výpočtu Turingova stroje se třemi registry podle tohoto programu ukazuje následující posloupnost přechodů, v níž je pro zvýraznění počáteční a koncová konfigurace podtržena (protože stroj vůbec nepoužívá pásku, je složka T ve všech konfiguracích prázdná).

$$\underline{(1;0,24,16,0;\emptyset)} \vdash (2;24,24,16;\emptyset) \vdash (3;8,24,16;\emptyset) \vdash (9;8,24,16;\emptyset) \vdash (10;8,8,16;\emptyset) \vdash$$
$$\vdash (1;8,8,16;\emptyset) \vdash (2;8,8,16;\emptyset) \vdash (3;0,8,16;\emptyset) \vdash (4;0,8,16;\emptyset) \vdash$$
$$\vdash (5;16,8,16;\emptyset) \vdash (6;8,8,16;\emptyset) \vdash (7;8,8,16;\emptyset) \vdash (8;8,8,8;\emptyset) \vdash$$
$$\vdash (1;8,8,8;\emptyset) \vdash (2;8,8,8;\emptyset) \vdash (3;0,8,8;\emptyset) \vdash (4;0,8,8;\emptyset) \vdash$$
$$\vdash (5;8,8,8;\emptyset) \vdash (6;0,8,8;\emptyset) \vdash (11;0,8,8;\emptyset) \vdash \underline{(12;8,8,8;\emptyset)}$$

Abychom mohli srovnat výpočetní schopnosti Turingových strojů s náhodným přístupem a standardních Turingových strojů, stanovíme pravidla pro zadávání vstupu a vracení výstup-

ních hodnot podobným způsobem, jaký jsme použili u standardních strojů. Políčka pásky (i registry) mohou sice obsahovat libovolné přirozené číslo, budeme však předpokládat, že těmito čísly kódujeme symboly nějaké abecedy.

Nechť je tedy pevně stanovena abeceda A, v níž bude zadáván vstup, přitom $\sqcup \in A$ a $\triangleright \notin A$ (symbol $\triangleright$ již není zapotřebí, neboť stroji s náhodným přístupem nehrozí nebezpečí, že by překročil levý okraj pásky). Nechť $\mathbf{K}$ je pevně zvolená bijekce (kódování) mezi A a $\{0, 1, \ldots, |A| - 1\}$, přičemž $\mathbf{K}(\sqcup) = 0$.

Definice 11.24: Počáteční konfigurace Turingova stroje $M = \langle k, P\rangle$ s náhodným přístupem pro vstup $w = a_1 a_2 \ldots a_n \in (A - \{\sqcup\})^*$ má tvar $(i, R_0, R_1, \ldots, R_k, T)$, kde $i = 1, R_j = 0$ pro $j = 1, 2, \ldots, k$ a $T = \{(1, \mathbf{K}(a_1)), (2, \mathbf{K}(a_2)), \ldots, (n, \mathbf{K}(a_n))\}$.

- Říkáme, že M **přijímá** řetěz $w \in (A - \{\sqcup\})^*$, pokud výpočet započatý v počáteční konfiguraci pro vstup w vede na koncovou konfiguraci s hodnotou $R_0 = 1$.
- Říkáme, že M **odmítá** w, pokud výpočet započatý v počáteční konfiguraci pro vstup w vede na koncovou konfiguraci s hodnotou $R_0 = 0$.
- Nechť $A_0 \subseteq A$ je nějaká abeceda a $L \subseteq A_0^*$ jazyk nad touto abecedou. Říkáme, že M **rozhoduje** jazyk L, pokud platí: je-li $w \in L$, pak M přijímá w, je-li $w \notin L$, pak M odmítá w.
- Říkáme, že M **polorozhoduje** L, pokud platí: $w \in L$ právě když se M na vstup w dostane do nějaké koncové konfigurace.
- Nechť $f : A_0^* \mapsto A_0^*$ je nějaké zobrazení. Říkáme, že M **počítá zobrazení** f, pokud se pro všechna $w \in A_0^*$ dostane stroj M do koncové konfigurace s obsahem pásky $T = \{(1, \mathbf{K}(b_1)), (2, \mathbf{K}(b_2)), \ldots, (n, \mathbf{K}(a_n))\}$, kde $f(w) = b_1 b_2 \ldots b_n\}$.

Pokusíme se nyní porovnat výpočetní sílu Turingova stroje s náhodným přístupem a standardního modelu. Je možno očekávat, že stroj s náhodným přístupem je výpočetně alespoň tak silný, jako standardní model, jinak řečeno pro libovolný Turingův stroj $M = \langle Q, A, \delta, s, F\rangle$ lze sestrojit Turingův stroj s náhodným přístupem M', který simuluje činnost stroje M. Stručně naznačíme způsob vytvoření takového stroje. M' bude obsahovat registr, označme jej např. k, který sleduje polohu čtecí/zapisovací hlavy stroje M. Na počátku odkazuje k na začátek vstupního řetězu na pásce. Každý stav $q \in Q$ stroje M bude M' simulovat určitou posloupností svých instrukcí, jejíž struktura bude zřejmá z následujícího příkladu.

Mějme např. $A = \{\sqcup, a, b\}, \mathbf{K}(a) = 1, \mathbf{K}(b) = 2$ a předpokládejme, že pro stav q stroje M platí

$$\delta(q, \triangleright) = (s, \rightarrow)\delta(q, \sqcup) = (p, \rightarrow),\ \ \delta(q, a) = (p, \leftarrow),\ \ \delta(q, b) = (r, \sqcup),$$

Posloupnost instrukcí simulující stav q by pak v pseudokódu mohla vypadat takto:

```
q:  if k = 0 then k := k + 1; goto s        levý konec pásky
    else case T[k] of
        0:  k := k + 1; goto p              T[k] = ⊔
        1:  k := k − 1; goto p              T[k] = a
        2:  T[k] := ⊔; goto r               T[k] = b
    end
```

První řádka pseudokódu vyjadřuje reakci na dosažení levého konce pásky, alternativy v příkazu **case** popisují činnost pro jednotlivé případy hodnot na pásce.

Ukazuje se však, že každý Turingův stroj s náhodným přístupem je simulovatelný standardním modelem, což vyjadřuje následující tvrzení.

Věta 11.25: Libovolný jazyk rozhodovaný, popř. polorozhodovaný Turingovým strojem s náhodným přístupem může být rozhodovaný, popř. polorozhodovaný standardním Turingovým strojem. Také libovolná funkce počítaná Turingovým strojem s náhodným přístupem může být počítaná standardním Turingovým strojem. Navíc, pokud se původní stroj zastaví pro nějaký

vstup, pak počet kroků provedený simulujícím standardním Turingovým strojem je omezen polynomiální funkcí počtu kroků původního stroje.

Důkaz: Nastíníme jen hlavní myšlenky důkazu spočívajícího v konstrukci simulujícího Turingova stroje M'. Pro Turingův stroj s náhodným přístupem $M = \langle k, P\rangle$ vytvoříme M' jako $(k+3)$-páskový stroj.

První páska slouží k zadání vstupu a popř. i uložení výsledku, pokud M počítá nějakou funkci. Druhá páska slouží k pamatování dvojic (t, m) reprezentujících obsah pásky T stroje M. K tomu je zapotřebí se dohodnout, že každá dvojice zde bude zobrazena jako dvě dvojková čísla oddělená čárkou a uzavřená v závorkách, např. řetěz (101,1100) by zobrazoval dvojici (5,12), která reprezentuje skutečnost, že $T[5] = 12$. Jednotlivé obrazy dvojic mohou být na této pásce odděleny libovolným počtem mezer, proto bude zaveden zvláštní koncový znak (např. #), za kterým už se na druhé pásce žádná dvojice nebude nacházet. Dalších k pásek stroje M' zobrazuje dvojkově obsah jednotlivých registrů stroje M a konečně poslední páska slouží při simulaci provádění instrukcí s přímým operandem $= c$ (viz dále). Hodnota čítače instrukcí stroje M je vyjádřena stavem stroje M' způsobem, který dále popíšeme.

Simulace má tři fáze. Během první fáze dostane stroj M' na první pásce vstup $x = a_1 a_2 \dots a_r \in \Sigma^*$ a převede jej na řetěz $1, \mathbf{K}(a_1))(2, \mathbf{K}(a_2)) \dots (r, \mathbf{K}(a_r))$ na druhé pásce. Stroj M' tak může začít simulaci od počáteční konfigurace stroje M.

Během druhé fáze stroj M' simuluje každý krok stroje M provedením několika vlastních kroků. Hodnotu čítače instrukcí přitom vyjadřuje stav stroje M', takže je možné říci, že stavy použité v této fázi se rozpadají do n disjunktních tříd $Q_1 \cup Q_2 \cup \dots \cup Q_n$, kde n je počet instrukcí programu P stroje M. Podmnožina stavů Q_i je vytvořena tak, aby posloupnost přechodů mezi nimi simulovala provedení instrukce p_i. Způsob vytvoření takových podmnožin stavů pro jednotlivé typy instrukcí ukážeme na několika příkladech.

Předpokládejme, že instrukce p_i má např. tvar *write 1*, tedy požaduje se zápis hodnoty registru R_0 (označme ji y) do políčka pásky, jehož číslo je uloženo v registru R_1 (označme je x). Při simulaci se tedy nejprve uloží dvojice (x, y) (získaná kopií obsahu pásek/registrů R_1 a R_0 a doplněním závorek a čárky) na konec druhé pásky. Pak se na druhé pásce vlevo od této dvojice hledá dvojice tvaru (x, y') pomocí opakovaného srovnávání první složky zaznamenaných dvojic s obsahem pásky/registru R_1. Při nalezení se tato dvojice z druhé pásky vymaže (přepíše mezerami) a pokračuje se přechodem na první stav podmnožiny Q_{i+1}. Simuluje-li se instrukce *read 1* je třeba zajistit jen prohledání druhé pásky, a v případě úspěchu pak okopírování druhé složky nalezené dvojice do pásky/registru R_1.

Je-li instrukce p_i tvaru např. *load 2* nebo *store 2*, pak se jedná o prosté kopírování obsahu jedné pásky/registru na druhou. Pro instrukci tvaru *add 3* je třeba provést operaci dvojkového přičtení čísla uloženého na pásce/registru R_3 k hodnotě na pásce/registru R_0, což je jistě operace, kterou je možné na vícepáskovém Turingově stroji snadno realizovat. U instrukce s přímým operandem, např. ve tvaru *add =22*, se nejprve vhodnou posloupností přechodů zaznamená dvojkové vyjádření operandu 22 na poslední (pomocnou) pásku, a pak se tato hodnota přičte k obsahu pásky/registru R_0.

Provedení řídicí instrukce tvaru např. *jzero 5* se simuluje tak, že se nejprve projde obsah pásky/registru R_0 a pokud obsahuje alespoň jednu číslici 1, přejde se na první stav podmnožiny Q_{i+1}, v opačném případě se přejde na první stav podmnožiny Q_5. Pro všechny zbývající typy instrukcí by se způsob simulace navrhnul obdobně. Zvláštní případ představuje instrukce *halt* – při ní nastane přechod do třetí fáze simulace, kdy se musí ukončení výpočtu stroje M' přizpůsobit charakteru činnosti stroje M. To znamená, že se buď na první pásku zaznamená výsledná výstupní hodnota, případně se přejde do přijímajícího či odmítajícího stavu y nebo n. Důkaz části tvrzení týkající se časové složitosti nebudeme uvádět. △

Nedeterministické Turingovy stroje

Vyzkoušeli jsme řadu možností, jak posílit výpočetní schopnosti Turingových strojů, ale zatím žádná z nich nepřinesla očekávaný efekt. Uvidíme, zda se situace změní, pokud připustíme, aby se Turingův stroj choval nedeterministicky.

Nedeterministické chování u Turingova stroje si lze snadno představit: pro některé (nebo všechny) kombinace stavu a symbolu na pásce může existovat více možností, jak se stroj zachová. Formálně zavedeme **nedeterministický Turingův stroj** jako pětici $M = \langle Q, A, \Delta, s, F\rangle$, kde Q, A, s a F mají stejný význam jako u standardního modelu a Δ je podmnožinou kartézského součinu

$$((Q - F) \times A) \ \times \ (Q \times (A \cup \{\leftarrow, \rightarrow\})),$$

tedy relací namísto zobrazení množiny $((Q - F) \times A)$ do $Q \times (A \cup \{\leftarrow, \rightarrow\})$. Konfigurace nedeterministického Turingova stroje a přechod mezi konfiguracemi $\vdash_M$ budou již definovány obvyklým způsobem s jediným rozdílem: přechod již není jednoznačný, ale má (obecně) více možností, na kterou konfiguraci bude převedena konfigurace stávající. To ovšem znamená, že nedeterministický stroj může k jednomu vstupu dodat několik různých výstupů nebo může skončit výpočet v různých stavech. Tato vlastnost se jeví jako nejméně rušivá při definici jazyka polorozhodovaného nedeterminstickým Turingovým strojem.

Definice 11.26: Nechť $M = \langle Q, A, \Delta, s, F\rangle$ je nedeterministický Turingův stroj. Říkáme, že M **přijímá vstup** $w \in (A - \{\triangleright, \sqcup\})^*$, pokud platí

$$(s, \triangleright\underline{\sqcup}w) \vdash_M^* (h, u\underline{a}v) \text{ pro nějaká } h \in F, a \in A, u, v \in A^*.$$

Nedeterministický stroj tedy přijme nějaký vstup, přestože některé výpočty pro tento vstup mohou být nekončící – stačí, aby existoval alespoň jeden končící výpočet. Říkáme, že M **polorozhoduje jazyk** $L \subseteq (A - \{\triangleright, \sqcup\})^*$, pokud pro libovolný řetěz $w \in (A - \{\triangleright, \sqcup\})^*$ platí: $w \in L$, právě když M přijímá w.

Definice jazyka rozhodovaného nebo funkce počítané nedeterministickým Turingový strojem vyžaduje mírně doplněný postup.

Definice 11.27: Nechť $M = \langle Q, A, \Delta, s, \{y, n\}\rangle$ je nedeterministický Turingův stroj. Říkáme, že M **rozhoduje jazyk** $L \subseteq (A - \{\triangleright, \sqcup\})^*$, pokud platí následující dvě podmínky pro libovolný řetěz $w \in (A - \{\triangleright, \sqcup\})^*$:

a) Existuje přirozené číslo k (závislé na M a w) takové, že každý výpočet začínající konfigurací $(s, \triangleright\underline{\sqcup}w)$ skončí po nejvýše k krocích.
b) $w \in L$, právě když existuje výpočet začínající konfigurací $(s, \triangleright\underline{\sqcup}w)$ a končící ve stavu y. nějaká $u, v \in A^*, a \in A$.

Konečně říkáme, že M **počítá zobrazení** $f : \ (A - \{\triangleright, \sqcup\})^* \mapsto (A - \{\triangleright, \sqcup\})^*$, pokud platí následující dvě podmínky pro libovolný řetěz $w \in (A - \{\triangleright, \sqcup\})^*$:

a) Existuje přirozené číslo k (závislé na M a w) takové, že každý výpočet začínající konfigurací $(s, \triangleright\underline{\sqcup}w)$ skončí po nejvýše k krocích.
b) $v = f(w)$, právě když každý výpočet začínající konfigurací $(s, \triangleright\underline{\sqcup}w)$ vede na koncovou konfiguraci $(h, \triangleright\underline{\sqcup}v)$

Podmínka a) v obou uvedených případech zaručuje, že každý výpočet nedeterministického stroje M pro libovolný vstup je konečný (maximální limitní počet kroků ovšem může záviset na vstupu). Podmínka b) v případě rozhodování jazyka znamená, že alespoň jeden z možných výpočtů pro vstup w musí být přijímající, zatímco pro stroj počítající nějaké zobrazení je požadavek výrazně silnější: všechny možné (a nutně končící!) výpočty pro daný vstup musí dávat stejný výsledek.

Věta 11.28: Nechť M je nedeterministický Turingův stroj, který polorozhoduje nebo rozhoduje nějaký jazyk, popř. počítá nějaké zobrazení. Potom existuje standardní Turingův stroj M', který polorozhoduje nebo rozhoduje stejný jazyk, popř. počítá totéž zobrazení.

Důkaz: Důkaz tohoto tvrzení se opírá o simulaci činnosti nedeterministického stroje systematickou probírkou všech možných výpočtů pro daný vstup. Deterministický stroj prochází stromem možných výpočtů do šířky a díky stanoveným podmínkám dokáže simulovat polorozhodování či rozhodování jazyka a rovněž výpočet zobrazení. Podrobnější nástin důkazu lze nalézt např. v [24]. △

11.3 Nerozhodnutelné problémy

Naše neúspěšné předchozí pokusy o zobecnění Turingových strojů ukázaly, že se v jejich případě zřejmě jedná o nejobecnější výpočetní model v rámci uvažovaného kontextu. K tomuto závěru přispívá i skutečnost, že další navržené modely mající za cíl co nejobecnější vyjádření mechanicky proveditelných výpočtů (např. Markovovy algoritmy a Kleeneho teorie rekurzivních funkcí) se ukázaly být výpočetně ekvivalentní s Turingovými stroji. Tato zjištění vedla k formulaci tzv. **Church-Turingovy teze**:

Intuitivní pojem „algoritmus“ je totéž jako Turingův stroj, který na každý vstup reaguje končícím výpočtem.

Tuto tezi nelze dokázat způsobem obvyklým v matematice, neboť srovnává pojem vymezený pouze intuitivně s pojmem zavedeným přesnou definicí. Je ovšem možné ji chápat jako přesné vymezení pojmu „algoritmus“ a v duchu této interpretace pak zjišťovat hranice možností algoritmů. V následujícím textu potvrdíme existenci takové hranice formulací klasického „problému zastavení“, který patří mezi algoritmicky neřešitelné úlohy.

Univerzální Turingův stroj

Turingův stroj, jak jej dosud známe, je velmi podobný pevně naprogramovanému počítači – program pro jeho činnost je jednou provždy zadán jeho přechodovou funkcí. Zdálo by se, že tedy Turingovým strojům chybí univerzálnost běžných číslicových počítačů daná schopností uložit do své paměti libovolný program, a ten pak provádět. Ukážeme, že tento nedostatek je jen zdánlivý, neboť lze sestrojit tzv. **univerzální Turingův stroj**, který je schopen simulovat činnost libovolného jiného Turingova stroje.

Každý Turingův stroj je možné popsat konečnými prostředky: má konečnou abecedu, konečnou množinu vnitřních stavů a rovněž jeho přechodovou funkci lze vyjádřit konečnou formou. Můžeme si tedy představit, že za použití vhodné notace můžeme stroj zcela popsat nějakým řetězem symbolů vhodně zvolené abecedy - takovému řetězu budeme říkat **popis Turingova stroje**. Na základě popisu zadaného při spuštění na vstupu univerzálního Turingova stroje pak může tento stroj simulovat činnost popsaného stroje. Uvedeme základní myšlenky obecné metody tvorby popisu libovolného Turingova stroje a způsobu, jak s ním univerzální stroj bude pracovat.

Nechť $M = \langle Q, A, \delta, s, F\rangle$ je nějaký Turingův stroj, označme jako m, n nejmenší přirozená čísla taková, že platí

$$2^m \geq |Q| \quad \text{a} \quad 2^n \geq |A| + 2.$$

Nyní můžeme pro účely popisu zvolit velmi jednoduché kódování stavů stroje M i symbolů jeho abecedy:

- každý vnitřní stav q_i bude kódován řetězem $qi_1 \dots i_m$, kde $i_1 \dots i_m$ je m-místné binární vyjádření pořadového čísla stavu i
- každý symbol $a_j \in A \cup \{\leftarrow, \rightarrow\}$ abecedy A obohacené o symboly pro pohyb hlavy vlevo a vpravo kódován řetězem $aj_1 \dots j_n$, kde $j_1 \dots j_n$ je n-místné binární vyjádření pořadového čísla symbolu j.

stav	symbol	δ
s	x	$(q, \sqcup)$
s	$\sqcup$	$(h, \sqcup)$
s	$\triangleright$	$(s, \rightarrow)$
q	x	(s, x)
q	$\sqcup$	$(s, \rightarrow)$
q	$\triangleright$	$(q, \rightarrow)$

a)

stav/symbol	kód
s	$q00$
q	$q01$
h	$q10$
$\sqcup$	$a000$
$\triangleright$	$a001$
$\leftarrow$	$a010$
$\rightarrow$	$a011$
x	$a100$

b)

Obrázek 11.14: Popis Turingova stroje

Vzhledem ke speciálnímu významu symbolů $\sqcup, \triangleright, \leftarrow, \rightarrow$ budeme předpokládat, že jim jsou přiřazena první čtyři místa v pořadí, jejich kódy tedy po řadě budou $a0\ldots00, a0\ldots01, a0\ldots10$, $a0\ldots11$.

Podstatnou – a v zásadě i postačující – informaci o Turingově stroji poskytuje jeho přechodové zobrazení δ. Toto zobrazení vyjádříme řetězem, který bude mít formu posloupnosti uspořádaných čtveřic (q, a, p, b), kde q, p jsou kódy stavů a a, b jsou kódy symbolů abecedy. Každá čtveřice vyjadřuje jeden řádek v tabulce pro přechodové zobrazení, tedy $\delta(q, a) = (p, b)$. Pro snazší prohledávání této posloupnosti budeme předpokládat, že čtveřice jsou lexikograficky uspořádány, takže první bude odpovídat hodnotě $\delta(s, \sqcup)$. Vytvoříme-li popis Turingova stroje uvedeným způsobem, bude výsledkem řetěz nad abecedou obsahující pouze symboly $a, q, 0, 1, (,)$ a ',' (čárku).

Jako příklad uvažujme Turingův stroj $M = \langle Q, A, \delta, s, \{h\}\rangle$, kde je $Q = \{s, q, h\}, A = \{\sqcup, \triangleright, x\}$ a přechodová funkce je zadána tabulkou na obr. 11.14a). Ke kódování stavů zřejmě postačí dvoumístná číselná identifikace, pro kódy symbolů použijeme identifikaci třímístnou, jak ukazuje tabulka na obr. 11.14b). Posloupnost symbolů $w = \triangleright \sqcup x \sqcup xx \sqcup$ bychom v popsaném kódování vyjádřili řetězem $\kappa(w) = a001a000a100a000a100a100a000$. Celou tabulku přechodového zobrazení, tedy popis zadaného Turingova stroje M vyjadřuje řetěz

$$\begin{aligned}\kappa(M) \;=\; & (q00, a100, q01, a000), (q00, a000, q10, a000), (q00, a001, q00, a011),\\ & (q01, a100, q00, a100), (q01, a000, q00, a011), (q01, a001, q01, a011).\end{aligned}$$

Nyní již můžeme začít s popisem univerzálního Turingova stroje, označme jej jako U. Stroj U dostane popis $\kappa(M)$ nějakého Turingova stroje M a popis $\kappa(w)$ jeho vstupu w a bude se chovat tak, jak by pracoval stroj M při vstupu w. Musí tedy platit, že U se zastaví pro vstup $\kappa(M)\kappa(w)$, právě když se M zastaví pro vstup w, navíc výpočet na stroji U dá i stejný výsledek, jako dal stroj M, tedy ve funkčním zápisu

$$U(\kappa(M)\kappa(w)) = \kappa(M(w)).$$

Stroj U navrhneme jako třípáskový: na první pásce bude mít zakódovaný obsah pásky stroje M, na druhé pásce zakódovaný popis stroje M samotného a na třetí pásce pak kód stavu stroje M v daném okamžiku simulovaného výpočtu. Stroj U začíná výpočet se vstupem $\kappa(M)\kappa(w)$ na první pásce. Nejprve přesune $\kappa(M)$ na druhou pásku a na začátek první pásky doplní zakódované symboly $\triangleright\sqcup$, k nimž přisune zprava řetěz $\kappa(w)$. Na třetí pásku pak zaznamená kód počátečního stavu s stroje M ve tvaru $q0\ldots0$ (délky binárních identifikací stavů a symbolů m, n se určí snadno prohlídkou $\kappa(M)$).

Nyní začne U simulovat jednotlivé kroky výpočtu stroje M. Při zahájení simulace každého kroku jsou čtecí/zapisovací hlavy druhé a třetí pásky umístěny na začátku a hlava první pásky se nachází na začátku kódu toho symbolu, který by četl stroj M v daném kroku.

Simulace probíhá tak, že U čte druhou pásku a hledá na ní čtveřici, jejíž první složka se shoduje s obsahem třetí pásky a druhá složka se shoduje s právě čteným symbolem na první pásce. Pokud tuto čtveřici nalezne, změní kód stavu na třetí pásce podle třetí složky této čtveřice a provede nad první páskou akci odpovídající kódu ve čtvrté složce čtveřice. To znamená, že pro kód $a0\ldots010$ přesune hlavu na první symbol a vlevo, pro kód $a0\ldots011$ přesune hlavu na první symbol a vpravo, pro ostatní kódy provede přepis obsahu první pásky. Pokud by se při pohybu vpravo četl na první pásce symbol $\sqcup$, stroj U jej musí převést na $a0\ldots0$, což je kód odpovídající mezeře. Pokud se v nějakém kroku nenajde hledaná čtveřice, znamená to dosažení koncového stavu a stroj U se zastaví.

Problém zastavení

Programátoři vědí, jak je těžké sestavit správně fungující program a jak často se např. povede vytvořit nekončící (nekonečný) cyklus. Jistě by velmi uvítali nástroj – nazvěme jej univerzální testovací program $TEST$ – který by jim jejich práci trochu usnadnil následujícím způsobem. Program $TEST$ by dostal na vstupu zadán libovolný program P a vstup tohoto programu X a dokázal by určit, zda se program P pro vstup X zastaví či nikoliv. Formálně zapsáno

$$TEST(P,X) \quad = \quad \begin{cases} y & \text{program } P \text{ se pro vstup } X \text{ zastaví} \\ n & \text{program } P \text{ se pro vstup } X \text{ nezastaví} \end{cases}$$

Program $TEST$ můžeme nyní využít k následujícímu experimentu: budeme jím testovat, jak programy reagují, pokud dostanou svůj vlastní kód jako vstupní data. Výsledek ale trochu obrátíme, jak ukazuje následující pseudokód definující program $POKUS$:

$POKUS(X)$
L: **if** $TEST(X,X)$ **then goto** L **else halt**

Volání programu $POKUS(X)$ tedy způsobí nekonečné zacyklení, pokud se program X pro vstup X zastaví, v opačném případě se výpočet skončí. S programem $POKUS$ provedeme další experiment: jak se zachová, pokud dostane na vstupu svůj vlastní kód, tedy jak proběhne výpočet $POKUS(POKUS)$? Tento výpočet skončí právě tehdy, jestliže $TEST(POKUS, POKUS)$ dopadne negativně (vrátí n), což znamená že program $POKUS$ na vstup $POKUS$ nikdy neskončí. To je ale zjevný spor, který nás vrací na začátek našich úvah – žádný program $TEST$ popsaných vlastností nemůže (bohužel) existovat.

Provedeme nyní několik obdobných kroků, v nichž ovšem namísto nespecifikovaných „programů“ budeme používat Turingovy stroje. Výsledkem bude zjištění rekurzivně vyčíslitelného jazyka, který není rekurzivní. Zvolme jazyk H (nad abecedou používanou při kódování popisu Turingových strojů) následujícím předpisem

$$H = \{\kappa(Mw) : \text{ Turingův stroj } M \text{ se zastaví pro vstupní řetěz } w\}.$$

Všimněme si, že H je rekurzivně vyčíslitelný jazyk, neboť jej polorozhoduje právě univerzální Turingův počítač z předchozího odstavce – U se zastaví pro vstupní řetěz $\kappa(Mw)$, právě když tento řetěz patří do H. Kdybychom nyní dokázali, že H je rekurzivní jazyk, pak by byl rekurzivní **každý rekurzivně vyčíslitelný jazyk**, což vyplyne z následující úvahy.

Předpokládejme na okamžik, že pro jazyk H existuje Turingův stroj M_0, který jej rozhoduje. Potom pro libovolný Turingův stroj M, který polorozhoduje jazyk $L(M)$ dokážeme sestrojit stroj M' rozhodující jazyk $L(M)$ takto: Stroj M' nejprve transformuje obsah své vstupní pásky z $\triangleright\underline{\sqcup}w$ na obsah $\triangleright\underline{\sqcup}\kappa(M)\kappa(w)$, a na tomto vstupu již bude pracovat stejně, jako by postupoval Turingův stroj M_0. Podle našeho předpokladu umí M_0 *rozhodnout*, zda se stroj M pro řetěz w zastaví, tedy zda M přijme řetěz w.

Nyní ovšem ukážeme, že jazyk H **není rekurzivní**. Náš postup bude jen formalizací výše provedených konstrukcí s programy $TEST$ a $POKUS$. Kdyby H byl rekurzivní, pak by také jazyk

$$H_1 = \{\kappa(M) : \text{ Turingův stroj } M \text{ se zastaví pro vstup } \kappa(M)\}$$

byl nutně rekurzivní. Kdyby totiž existoval Turingův stroj M_0 rozhodující jazyk H, pak by stroji M_1 rozhodujícímu jazyk H_1 stačilo transformovat svůj vstup $\triangleright\underline{\sqcup}\kappa(M)$ na $\triangleright\underline{\sqcup}\kappa(M)\kappa(M)$ a pak předat zpracování stroji M_0. Stačí tedy dokázat, že H_1 není rekurzivní.

Pokud by byl jazyk H_1 rekurzivní, pak by byl rekurzivní i jeho doplněk

$$\begin{aligned}\overline{H_1} \;=\; & \{w : w \text{ není zakódovaným popisem žádnéhoTuringova stroje}\} \cup \\ & \{w : w \text{ je zakódováním}\kappa(M) \text{ stroje } M, \text{ který se nezastaví na vstup } \kappa(M)\}\end{aligned}$$

Jazyk $\overline{H_1}$ ovšem není nejenom rekurzivní, ale ani rekurzivně vyčíslitelný. Předpokládejme, že existuje Turingův stroj M_1, který polorozhoduje jazyk $\overline{H_1}$. Je potom $\kappa(M_1)$ v jazyce $\overline{H_1}$?

Podle definice $\overline{H_1}$ je $\kappa(M_1) \in \overline{H_1}$, právě když M_1 nepřijme vstupní řetěz $\kappa(M_1)$. Přitom ovšem od M_1 očekáváme, že polorozhoduje jazyk $\overline{H_1}$, takže má být $\kappa(M_1) \in \overline{H_1}$, právě když M_1 přijme $\kappa(M_1)$. Docházíme tedy ke zjištění, že M_1 přijímá řetěz $\kappa(M_1)$, právě když M_1 nepřijímá řetěz $\kappa(M_1)$. To je zřejmý spor a náš výchozí předpoklad o existenci Turingova stroje M_0 je chybný.

Předchozími úvahami jsme potvrdili **nerozhodnutelnost problému zastavení pro Turingovy stroje**. Bez důkazu uvedeme nyní ještě několik dalších nerozhodnutelných problémů, které se týkají rovněž Turingových strojů.

- Je dán Turingův stroj M. Zastaví se M pro prázdný vstup?
- Je dán Turingův stroj M. Existuje nějaký řetěz w, po jehož zadání na vstupu se M zastaví?
- Je dán Turingův stroj M. Zastaví se M pro každý vstupní řetěz?
- Jsou dány Turingovy stroje M_1 a M_2. Zastaví se M_1 a M_2 pro stejné vstupní řetězy?