Turingovy stroje

Churchova teze

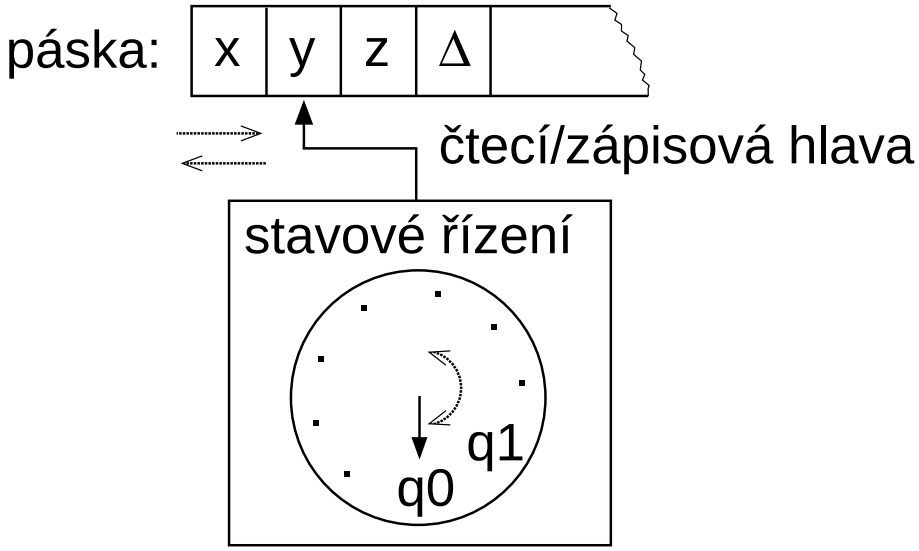
❖ ***Churchova (Church-Turingova) teze**: Turingovy stroje (a jim ekvivalentní systémy) definují svou výpočetní silou to, co intuitivně považujeme za efektivně vyčíslitelné.*

❖ Churchova teze není teorém, nemůžeme formálně dokazovat, že něco odpovídá našim intuitivním představám, nicméně je podpořena řadou argumentů:

- Turingovy stroje jsou velmi robustní – uvidíme, že jejich různé úpravy nemění jejich výpočetní sílu (determinismus x nedeterminismus, počet pásek, ...).
- Byla navržena řada zcela odlišných výpočetních modelů (λ-kalkulus, parciálně rekurzívní funkce, Minského stroje, ...), jejichž síla odpovídá Turingovým strojům.
- Není znám žádný výpočetní proces, který bychom označili za efektivně vyčíslitelný a který by nebylo možné realizovat na Turingově stroji. [a]

[a] Existují formalizované výpočetní procesy, realizovatelné např. na TS s orákulem (nápovědou) rozhodujícím atomicky nějaký Turingovsky nerozhodnutelný problém (např. problém zastavení), které ale nepovažujeme za efektivní výpočetní procesy.

Turingův stroj



Definice 7.1 Turingův stroj (TS) je šestice tvaru $M = (Q, \Sigma, \Gamma, \delta, q_0, q_F)$, kde:

- Q je konečná množina vnitřních (řídicích) stavů,
- Σ je konečná množina symbolů nazývaná vstupní abeceda, $\Delta \notin \Sigma$,
- Γ je konečná množina symbolů, $\Sigma \subset \Gamma$, $\Delta \in \Gamma$, nazývaná pásková abeceda,
- parciální funkce $\delta : (Q \setminus \{q_F\}) \times \Gamma \to Q \times (\Gamma \cup \{L, R\})$, kde $L, R \notin \Gamma$, je přechodová funkce,
- q_0 je počáteční stav, $q_0 \in Q$ a
- q_F je koncový stav, $q_F \in Q$.

Konfigurace Turingova stroje

❖ Symbol Δ značí tzv. blank (prázdný symbol), který se vyskytuje na místech pásky, která nebyla ještě použita (může ale být na pásku zapsán i později).

❖ Konfigurace pásky je dvojice sestávající z nekonečného řetězce reprezentujícího obsah pásky a pozice hlavy na tomto řetězci – přesněji jde o prvek množiny $\{\gamma\Delta^\omega \mid \gamma \in \Gamma^*\} \times \mathbb{N}$. [a]

❖ Konfiguraci pásky zapisujeme jako $\Delta xyz\underline{z}\Delta x\Delta\Delta\ldots$ (podtržení značí pozici hlavy).

❖ Konfigurace stroje je pak dána stavem řízení a konfigurací pásky – formálně se jedná o prvek množiny $Q \times \{\gamma\Delta^\omega \mid \gamma \in \Gamma^*\} \times \mathbb{N}$.

❖ I když je páska nekonečná, dokážeme ji jednoznačně specifikovat konečným řetězcem popisující obsah neprázdných políček (těch je vždy konečně mnoho). Proto můžeme zjednodušit zápis konfigurace a specifikovat pouze neprázdné políčka pásky $\gamma \in \Gamma^*$.

---

[a] Pro libovolnou abecedu Σ je Σ^ω množina všech *nekonečných* řetězců nad Σ, tj. nekonečných posloupností symbolů ze Σ. Pro připomenutí: Σ^* je množina všech *konečných* řetězců nad Σ.

Přechodová relace TS

❖ Pro libovolný řetězec $\gamma \in \Gamma^\omega$ a číslo $n \in \mathbb{N}$ označme γ_n n-tý symbol daného řetězce a označme $s_b^n(\gamma)$ řetězec, který vznikne z γ záměnou γ_n za b.

❖ Krok výpočtu TS M definujeme jako *nejmenší* binární relaci $\underset{M}{\vdash}$ takovou, že $\forall q_1, q_2 \in Q\ \forall \gamma \in \Gamma^\omega\ \forall n \in \mathbb{N}\ \forall b \in \Gamma$:

- $(q_1, \gamma, n) \underset{M}{\vdash} (q_2, \gamma, n+1)$ pro $\delta(q_1, \gamma_n) = (q_2, R)$ – operace posuvu doprava při γ_n pod hlavou,
- $(q_1, \gamma, n) \underset{M}{\vdash} (q_2, \gamma, n-1)$ pro $\delta(q_1, \gamma_n) = (q_2, L)$ a $n > 0$ – operace posuvu doleva při γ_n pod hlavou a
- $(q_1, \gamma, n) \underset{M}{\vdash} (q_2, s_b^n(\gamma), n)$ pro $\delta(q_1, \gamma_n) = (q_2, b)$ – operace zápisu b při γ_n pod hlavou.

Výpočet TS

❖ Výpočet TS M začínající z konfigurace K_0 je posloupnost konfigurací K_0, K_1, K_2, ...,

- ve které $K_i \underset{M}{\vdash} K_{i+1}$ pro všechna $i \geq 0$ taková, že K_{i+1} je v dané posloupnosti, a
- která je buď
  - nekonečná, a nebo
  - konečná s koncovou konfigurací (q, γ, n), přičemž rozlišujeme následující typy zastavení TS:
    1. normální – přechodem do koncového stavu, tj. $q = q_F$, a
    2. abnormální, kdy $q \neq q_F$ a:
       (a) pro (q, γ_n) není δ definována, nebo
       (b) hlava je na nejlevější pozici pásky a dojde k posunu doleva, tj. $n = 0$ a $\delta(q, \gamma_n) = (q', L)$ pro nějaké $q' \in Q$.

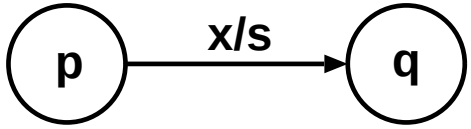
Poznámka – alternativní definice TS

❖ Používají se i některé alternativní definice TS, u kterých se dá snadno ukázat vzájemná převoditelnost:

- namísto jediného q_F je povolena množina koncových stavů,
- namísto q_F je zavedena dvojice q_{accept} a q_{reject},
- na prvním políčku pásky je „napevno“ zapsán symbol konce pásky, z něhož není možný posun doleva,
- při zavedení obou předchozích bodů je δ obvykle definovaná jako totální funkce,
- přepis a posuv hlavy jsou spojeny do jedné operace
- apod.

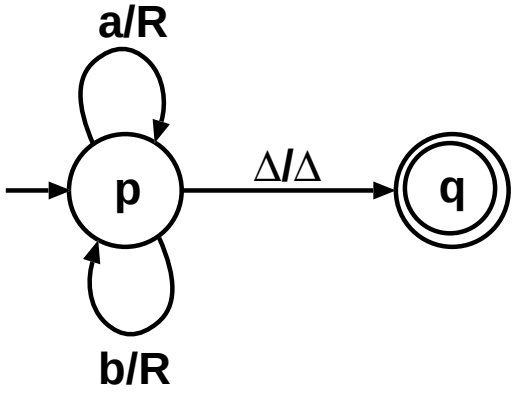
Grafická reprezentace TS

- Grafická reprezentace přechodu (x – co se čte, s – zápis/L/R):



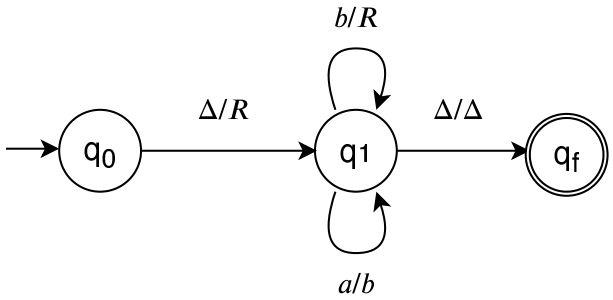
- Grafická reprezentace počátečního a koncového stavu:

Příklad 7.1 TS, který posouvá hlavu doprava na první Δ počínaje aktuální pozicí (např. $\Delta\underline{a}ab\Delta... \longrightarrow \Delta aab\underline{\Delta}...$):

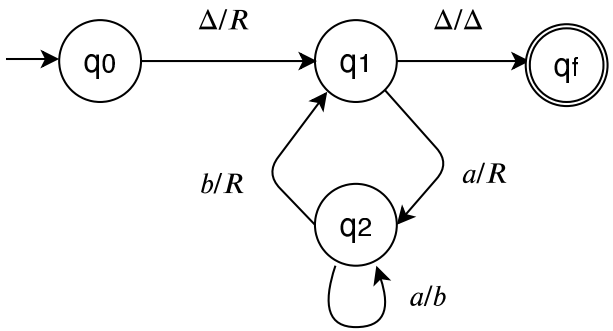


Příklady TS

Příklad 7.2 TS, který modifikuje pásku ve tvaru $\underline{\Delta}a^n\Delta$ pro $n > 0$ na $\Delta b^n\underline{\Delta}$

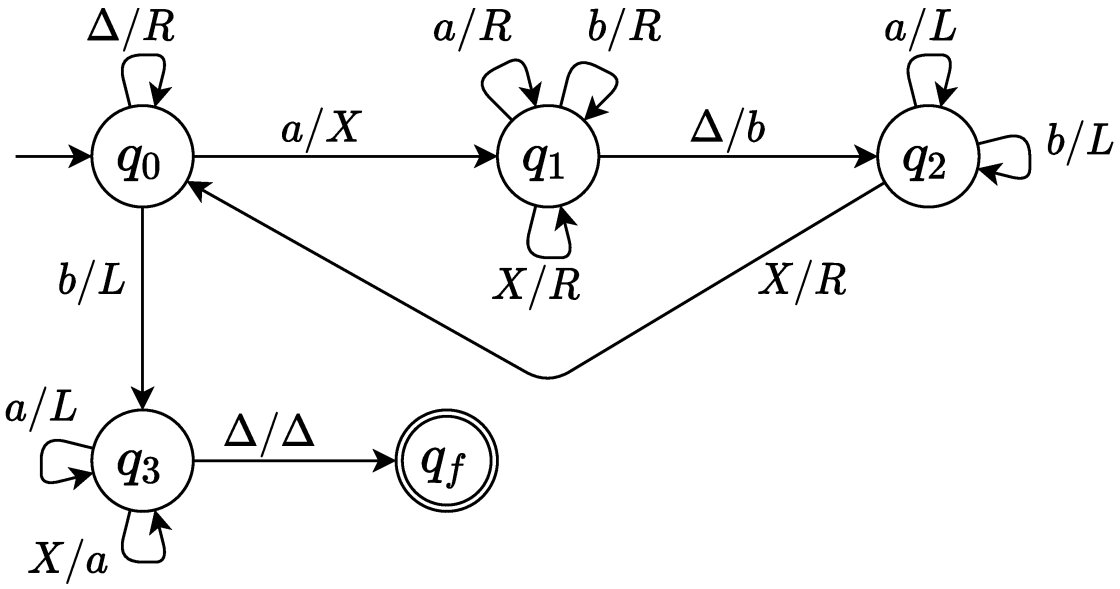


Příklad 7.3 TS, který modifikuje pásku ve tvaru $\underline{\Delta}a^{2n}\Delta$ pro $n > 0$ na $\Delta(ab)^n\underline{\Delta}$



Příklady TS

Příklad 7.4 TS, který modifikuje pásku ve tvaru $\underline{\Delta}a^n\Delta$ pro $n > 0$ na $\underline{\Delta}a^n b^n \Delta$



Poznámka 7.1

- Uvědomme si, že na TS lze také nahlížet jako na výpočetní mechanismy implementující funkce $\Sigma^* \to \Gamma^*$ tím, že transformují počáteční neblankový prefix své pásky na jiný neblankový prefix při přechodu do koncového stavu.
- Vzhledem k tomu, že TS nemusí každý svůj vstup přijmout, jsou funkce jimi implementované obecně parciální.
- Blíže se budeme vyčíslováním funkcí TS zabývat dále.

Turingovy stroje jako akceptory jazyků

Jazyk přijímaný TS

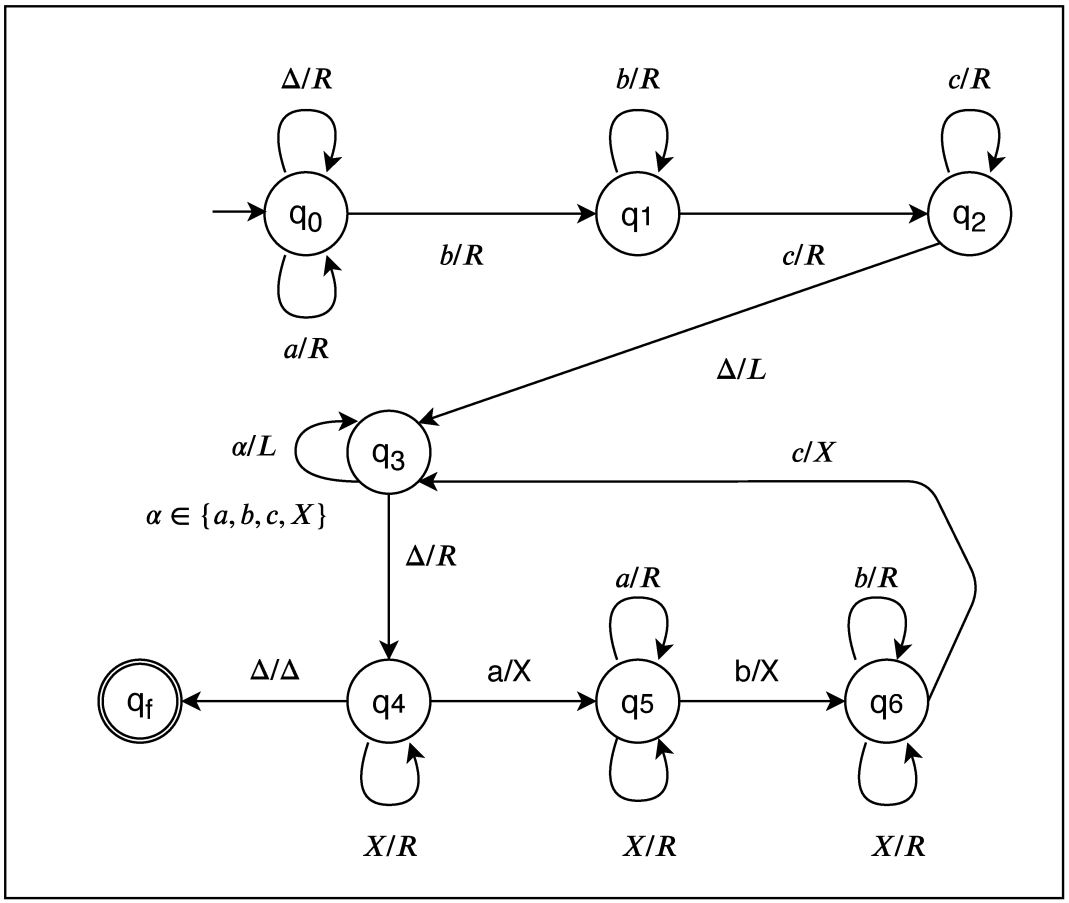
Definice 7.2

1. Řetězec $w \in \Sigma^*$ je přijat TS $M = (Q, \Sigma, \Gamma, \delta, q_0, q_F)$, jestliže M při aktivaci z počáteční konfigurace pásky $\underline{\Delta}w\Delta...$ a počátečního stavu q_0 zastaví přechodem do koncového stavu q_F, tj. $(q_0, \Delta w\Delta^\omega, 0) \overset{*}{\underset{M}{\vdash}} (q_F, \gamma, n)$ pro nějaké $\gamma \in \Gamma^*$ a $n \in \mathbb{N}$.
2. Množinu $L(M) = \{w \mid w$ je přijat TS $M\} \subseteq \Sigma^*$ nazýváme jazyk přijímaný TS M.

❖ Alternativně můžeme přijetí řetězce TS definovat tak, že TS začíná s konfigurací pásky $\underline{\Delta}w\Delta...$ a zastaví s konfigurací pásky $\underline{\Delta}Y\Delta...$, $Y \in \Gamma \setminus \Sigma$, ($Y$ *značí Yes*).

TS pro nebezkontextový jazyk

Příklad 7.5 TS, který přijímá jazyk $L = \{a^n b^n c^n \mid n > 0\}$



Vícepáskové Turingovy stroje

Vícepáskové TS

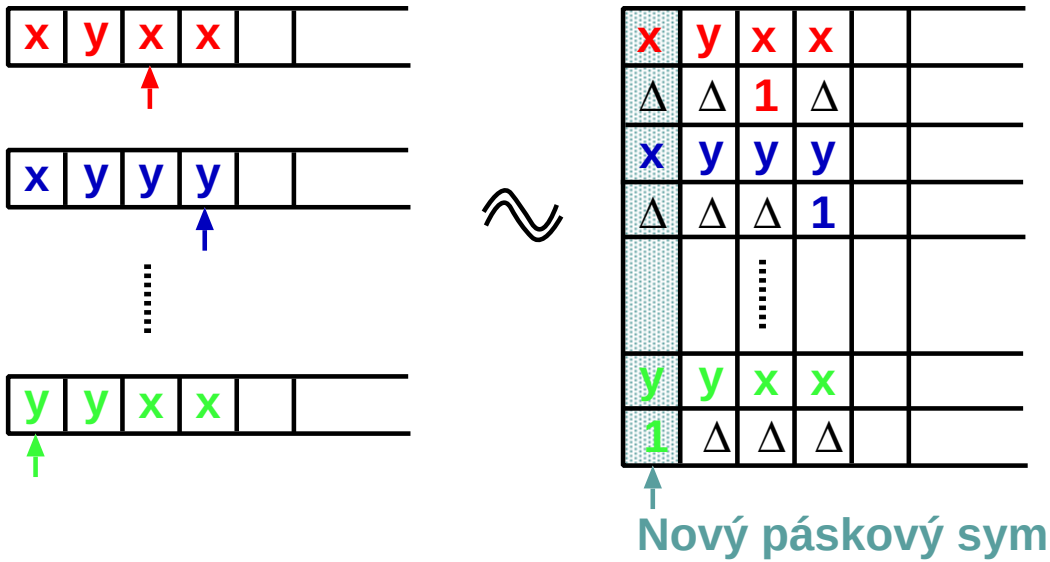
❖ Uvažujme TS, který má k pásek s páskovými abecedami $\Gamma_1, \Gamma_2, ..., \Gamma_k$ a k *odpovídajících hlav* s přechodovou funkcí tvaru

$$\delta : (Q \setminus \{q_F\}) \times \Gamma_1 \times \Gamma_2 \times ... \times \Gamma_k \longrightarrow Q \times \Gamma_1' \times \Gamma_2' \times ... \times \Gamma_k'$$

kde $\Gamma_i' = \Gamma_i \cup \{L, R\}$.

Věta 7.1 Pro každý k-páskový TS M existuje jednopáskový TS M' takový, že $L(M) = L(M')$.

Důkaz. (idea)



Důkaz pokračuje dále.

Pokračování důkazu. Důkaz provedeme tak, že ukážeme algoritmus převodu na ekvivalentní jednopáskový TS:

- Předpokládáme, že přijímaný řetězec je u k-páskového stroje na počátku zapsán na první pásce, všechny ostatní pásky jsou prázdné a všechny hlavy jsou na nejlevější pozici.
- Původních k pásek simulujeme rozšířením páskové abecedy o $2k$-tice, v nichž vždy i-tá složka pro liché i reprezentuje obsah $(\frac{i+1}{2})$-ní pásky a na pozici $i+1$ je Δ nebo 1 podle toho, zda se na ní nachází příslušná hlava či nikoliv.
- Počet načítaných kombinací symbolů v původním automatu je konečný a tudíž si výše uvedené rozšíření můžeme skutečně dovolit.
- Při simulaci k-páskového TS pak nejprve převedeme původní obsah první pásky na ekvivalentní obsah zakódovaný v $2k$-ticích a pak každý krok simulujeme několika kroky.
- Při rozhodování o dalším kroku stroj M' projde celou pásku a zapamatuje si ve svém stavu uspořádanou k-tici aktuálně čtených symbolů. Stavy jsou tedy $(k+1)$-tice, kde první složka je stav původního TS, a dovolují tedy stroji M' korektně simulovat původní stroj.
- Po přečtení pásky a aktualizace stavu M' přemístí hlavu na speciální pozici nalevo od užitečného obsahu pásky.

Důkaz pokračuje dále.

Pokračování důkazu.

- Po rozhodnutí o dalším kroku se posunujeme postupně doprava na pozice, která se mají modifikovat, provedeme příslušnou změnu a vrátíme se zpět doleva.
- Za nový aktuální stav považujeme $(k+1)$-tici danou novým stavem simulovaného TS a novou k-tici reprezentující modifikace na simulovaných páskách.
- Uvědomme si, že jeden krok simulace vícepáskového stroje vyžaduje 2 průchody páskou – více viz přednáška o složitosti.
- Navíc je nutné korektně simulovat „přepadnutí“ hlavy na kterékoliv pásce a převod dosud nevyužitých míst pásky s Δ na odpovídající $2k$-tici blank symbolů.
- Při řádné formalizaci popsaného algoritmu pak není těžké ukázat, že výsledný TS skutečně simuluje původní TS.

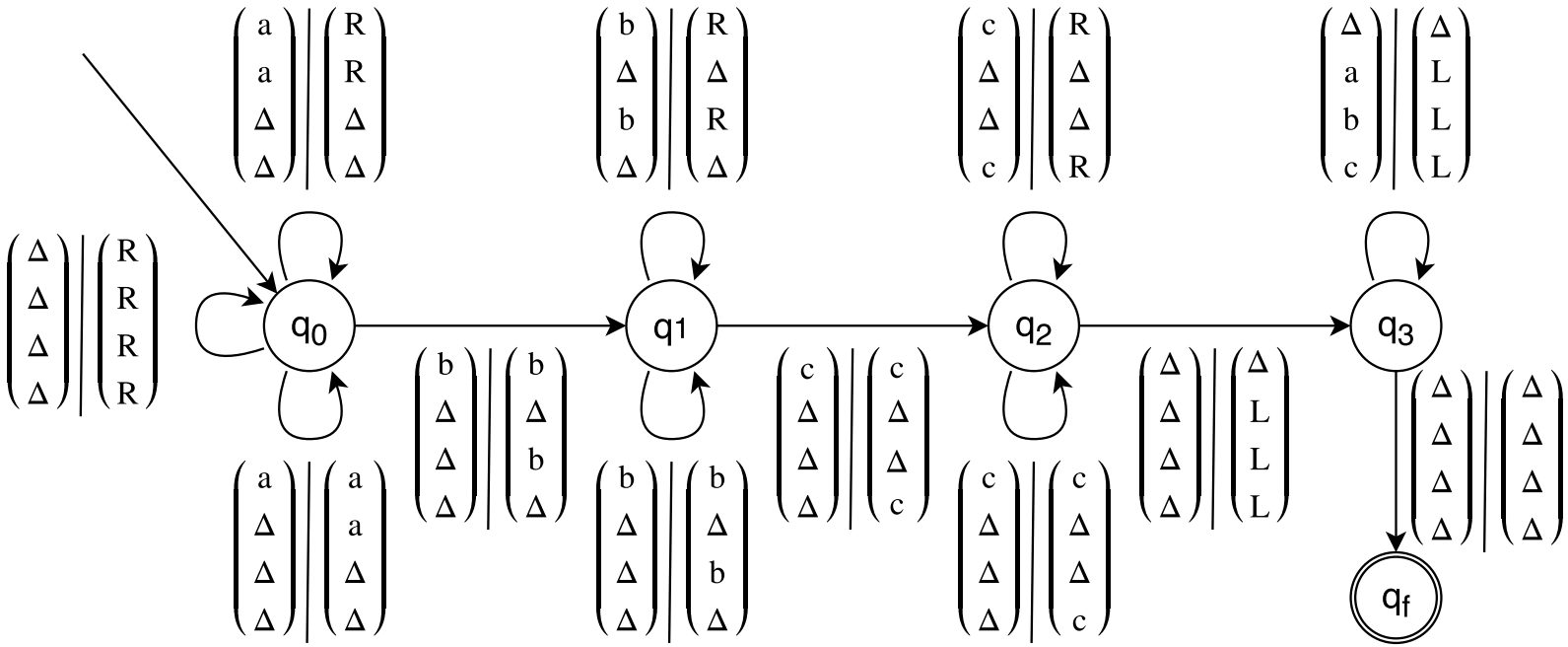
□

❖ Závěr:

Zvětšení paměťových možností TS nerozšiřuje jejich schopnosti přijímat jazyky!

Příklad vícepáskového TS

Příklad 7.6 4-páskový TS, který přijímá jazyk $L = \{a^n b^n c^n \mid n > 0\}$



Nedeterministické Turingovy stroje

Nedeterministické TS

Definice 7.3 Nedeterministický TS je šestice $M = (Q, \Sigma, \Gamma, \delta, q_0, q_F)$, kde význam jednotlivých složek je shodný s deterministickým TS až na δ, jež má tvar:

$$\delta : (Q \setminus \{q_F\}) \times \Gamma \longrightarrow 2^{Q \times (\Gamma \cup \{L,R\})}$$

Definice 7.4 Jazyk $L(M)$ NTS $M = (Q, \Sigma, \Gamma, \delta, q_0, q_F)$ je množina řetězců $w \in \Sigma^*$ takových, že M při aktivaci z q_0 při počátečním obsahu pásky $\underline{\Delta} w \Delta ...$ **může** zastavit přechodem do q_F.

Věta 7.2 Pro každý NTS M existuje DTS M' takový, že $L(M) = L(M')$.

Důkaz. (idea)

- NTS M budeme simulovat třípáskovým DTS. Význam jednotlivých pásek tohoto stroje je následující:
  - Páska 1 obsahuje vstupní řetězec.
  - Páska 2 je pracovní páska. Obsahuje kopii pásky 1 ohraničenou vhodnými speciálními značkami. Po neúspěšném pokusu o přijetí je její obsah smazán a obnoven z první pásky.
  - Páska 3 obsahuje kódovanou volbu posloupností přechodů; při neúspěchu bude její obsah nahrazen jinou posloupností.

Důkaz pokračuje dále.

Pokračování důkazu.

- Zvolená posloupnost přechodů je kódována posloupností čísel přiřazených přechodům simulovaného stoje.
- Jednotlivé posloupnosti přechodů na pásce 3 generujme pomocí BFS: nejprve všechny výpočty délky 1, potom všechny výpočty délky 2 atd.
- Vlastní simulace probíhá takto:
  1. Okopíruj obsah pásky 1 na pásku 2.
  2. Generuj příští posloupnost přechodů na pásce 3.
  3. Simuluj provedení posloupnosti z pásky 3 na obsahu pásky 2.
  4. Vede-li zkoumaná posloupnost do q_F simulovaného stroje, zastav – vstupní řetězec je přijat. V opačném případě smaž pásku 2 a vrať se k bodu 1.
- Není obtížné nahlédnout, že jazyk takto vytvořeného stroje odpovídá jazyku původního NTS.

□

❖ Závěr:

Zavedením nedeterminismu do TS se nezvyšují jejich schopnosti přijímat jazyky!

Univerzální Turingovy stroje

Kódování TS

❖ Kódovací systém pro TS zahrnuje (1) kódování stavů (tak, aby byly odlišeny všechny stavy včetně q_0 a q_F), (2) symbolů z Γ a (3) přechodové funkce δ.

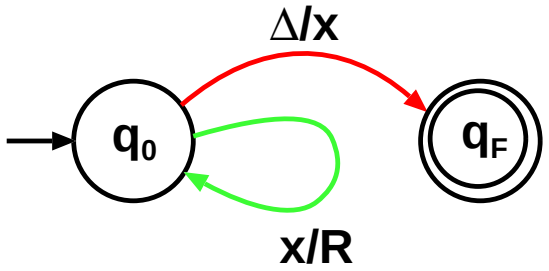
❖ Kódování stavů: Množinu stavů Q uspořádáme do posloupnosti q_0, q_F, q, p, ..., t. Stav q_j zakódujeme jako 0^j, přičemž indexujeme (např.) od nuly.

❖ Kódování symbolů a příkazů L/R: Předpokládejme, že $\Gamma = \Sigma \cup \{\Delta\}$. Uspořádáme Σ do posloupnosti a_1, a_2, ..., a_n a zvolíme tyto kódy: $\Delta \mapsto \varepsilon$, $L \mapsto 0$, $R \mapsto 00$, $a_i \mapsto 0^{i+2}$.

❖ Přechod $\delta(p, x) = (q, y)$, kde $y \in \Gamma \cup \{L, R\}$, reprezentujeme čtveřicí (p, x, q, y) a kódujeme zřetězením kódů p, x, q, y při použití 1 jako oddělovače, tj. jako $\langle p\rangle 1\langle x\rangle 1\langle q\rangle 1\langle y\rangle$, kde $\langle\_\rangle$ značí kód _.

❖ Celý TS kódujeme jako posloupnost kódů přechodů oddělených a ohraničených 1.

Příklad 7.7



kód: $1110100011000110 01$

Univerzální TS

❖ Zavádí koncept „programovatelného“ stroje, který umožňuje ve vstupním řetězci specifikovat konkrétní TS (tj. program) i data, nad nimiž má tento stroj pracovat.

❖ TS, který má být simulován, budeme kódovat, jak bylo uvedeno na předchozí straně, vstupní řetězec budeme kódovat jako posloupnost příslušných kódů symbolů oddělených a ohraničených 1. Kód stroje a vstupního řetězce oddělíme např. #.

Příklad 7.8 TS z předchozí strany mající na vstupu xxx:

$$1110100011000110 01\#10001000 10001$$

❖ Univerzální TS, který zpracuje toto zadání můžeme navrhnout jako třípáskový stroj, který

- má na 1. pásce zadání (a později výstup),
- 2. pásku používá k simulaci pracovní pásky původního stroje a
- na 3. pásce má zaznamenán řídicí stav simulovaného stroje a aktuální pozici hlavy (pozice hlavy i je kódována jako 0^i).

❖ Univerzální stroj pracuje takto:

1. Stroj zkontroluje, zda vstup odpovídá nějakému $M\#w$ a pokud ne, abnormálně zastaví.
2. Přepíše w na 2. pásku, na 3. pásku umístí kód q_0 a za něj poznačí, že hlava se nachází na levém okraji pásky.
3. Na 2. pásce vyhledá aktuální symbol pod hlavou simulovaného stroje a na 1. pásce vyhledá přechod proveditelný ze stavu zapsaného na začátku 3. pásky pro tento vstupní symbol. Pokud žádný přechod možný není, stroj abnormálně zastaví.
4. Stroj provede na 2. a 3. pásce změny odpovídající simulovanému přechodu (přepis aktuálního symbolu, změna pozice hlavy, změna řídicího stavu).
5. Pokud nebyl dosažen stav q_F simulovaného stroje, přejdeme na bod 3. Jinak stroj vymaže 1. pásku, umístí na ní obsah 2. pásky a zastaví přechodem do svého koncového stavu.

❖ Víme, že výše uvedený stroj můžeme převést na jednopáskový univerzální TS, který budeme v dalším značit jako T_U.

Jazyky rekurzívně vyčíslitelné a jazyky rekurzívní

Rekurzívní vyčíslitelnost a rekurzívnost

❖ Turingův stroj se nazývá úplný (*total*), právě když se pro každý vstup zastaví.

❖ *Poznámka*: Nedeterministický Turingův stroj je úplný, právě když pro každý vstup je každá výpočetní větev konečná (tj. pro každý vstup vždy zastaví).

Definice 7.5 Jazyk $L \subseteq \Sigma^*$ se nazývá

- rekurzívně vyčíslitelný, jestliže $L = L(M)$ pro nějaký TS M,
- rekurzívní, jestliže $L = L(M)$ pro nějaký *úplný* TS M.

❖ Je-li M úplný Turingův stroj, pak říkáme, že M rozhoduje jazyk $L(M)$.

❖ Ke každému rekurzívnímu jazyku existuje TS, který ho rozhoduje, tj. zastaví pro každé vstupní slovo – tento TS lze samozřejmě upravit tak, aby pro každý řetězec z daného jazyka zastavil s páskou $\Delta Y \Delta\Delta...$ a jinak zastavil s páskou $\Delta N \Delta\Delta....$

❖ TS přijímající rekurzívně vyčíslitelný jazyk L zastaví pro každé $w \in L$, ovšem pro $w \notin L$ může zastavit, ale také může donekonečna cyklit.

Rozhodovací problémy

❖ Rozhodovací problém (*decision problem*) P může být chápán jako funkce f_P s oborem hodnot $\{true, false\}$.

❖ Rozhodovací problém je obvykle specifikován:

- definičním oborem A_P reprezentujícím množinu možných instancí problému (vstupů) a
- podmnožinou $B_P \subseteq A_P$, $B_P = \{p \mid f_P(p) = true\}$ instancí, pro které je hodnota f_P rovna $true$.

❖ V teorii formálních jazyků používáme ke kódování jednotlivých instancí problémů řetězce nad vhodnou abecedou Σ. Pak je rozhodovací problém P přirozeně specifikován jazykem $L_p = \{w \in \Sigma^* \mid w = code(p), p \in B_P\}$, kde $code : A_P \to \Sigma^*$ je injektivní funkce, která přiřazuje instancím problému příslušný řetězec (nezávisle na f_P).

Příklad 7.9 Příklady rozhodovacích problémů:

- P_1 – orientovaný graf je silně souvislý.
- P_2 – dvě bezkontextové gramatiky jsou ekvivalentní,
- P_3 – n je prvočíslo.

❖ *Poznámka*: Dále budeme o rozhodovacích problémech hovořit jednoduše jako o problémech.

Rozhodování problémů TS

Definice 7.6 Nechť P je problém specifikovaný jazykem L_P nad Σ. Problém P nazveme:

- rozhodnutelný, pokud L_P je rekurzívní jazyk, tj. existuje TS, který L_P rozhoduje (přijme každý řetězec $w \in L_P$, a zamítne každý řetězec $w \in \Sigma^* \setminus L_P$),
- nerozhodnutelný, když není rozhodnutelný, a
- částečně rozhodnutelný, jestliže L_P je rekurzívně vyčíslitelný jazyk.

❖ ***Poznámka:*** Z definice 7.6 plyne, že každý rozhodnutelný problém je současně částečně rozhodnutelný, ale některé nerozhodnutelné problémy nejsou ani částečně rozhodnutelné.

TS a jazyky typu 0

Jazyky přijímané TS jsou typu 0

❖ Pro zápis konfigurace TS v řídícím stavu q a s konfigurací pásky $\Delta xyz\Delta...$ zavedeme konvenci $[\Delta xqyz\Delta...]$.

Věta 7.3 Každý rekurzívně vyčíslitelný jazyk je jazykem typu 0.

Důkaz. * Nechť $L = L(M)$ pro nějaký TS $M = (Q, \Sigma, \Gamma, \delta, q_0, q_F)$. Sestrojíme gramatiku $G = (N, \Sigma, P, S)$ typu 0 takovou, že $L(G) = L(M)$. Gramatika G dovoluje vytvářet derivace odpovídající reverzi posloupnosti konfigurací TS M při přijetí $w \in L(M)$:

1. $N = \{S\} \cup Q \cup (\Gamma \setminus \Sigma) \cup \{[,]\}$ (množiny jsou po dvou disjunktní).
2. P je nejmenší množina obsahující následující pravidla:
   (a) $S \to [q_f \Delta Y \Delta]$,
   (b) $\Delta] \to \Delta\Delta]$ – *doplnění* Δ,
   (c) $qy \to px$, jestliže $\delta(p, x) = (q, y)$,
   (d) $xq \to px$, jestliže $\delta(p, x) = (q, R)$,
   (e) $qyx \to ypx$ pro každé $y \in \Gamma$, jestliže $\delta(p, x) = (q, L)$,
   (f) $[q_0\Delta \to \varepsilon, \quad \Delta\Delta] \to \Delta], \quad \Delta] \to \varepsilon$ – *zajištění* $[q_0\Delta w\Delta...\Delta] \overset{+}{\underset{G}{\Rightarrow}} w$.

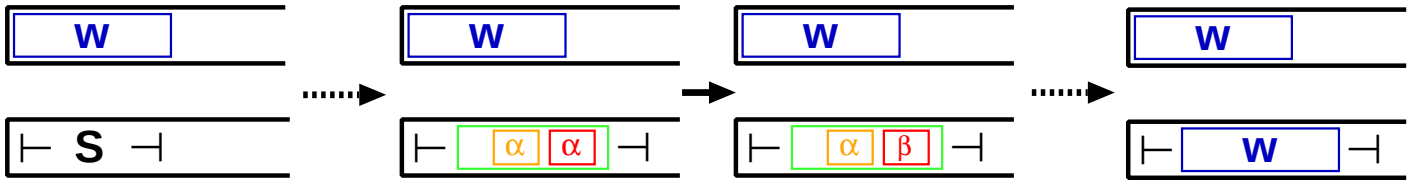
Snadno se nyní nahlédne, že $w \in L(M)$ právě tehdy, když existuje derivace $S \Rightarrow_G [q_F\Delta Y\Delta] \Rightarrow_G ... \Rightarrow_G [q_0\Delta w\Delta...] \Rightarrow_G ... \Rightarrow_G w$, a že $L(G) = L(M)$. □ *

Jazyky typu 0 jsou přijímány TS

Věta 7.4 Každý jazyk typu 0 je přijímán nějakým TS (tj. je rekurzívně vyčíslitelný).

Důkaz. Nechť $L = L(G)$ pro $G = (N, \Sigma, P, S)$ je jazykem typu 0. Sestrojíme nedeterministický dvoupáskový TS M takový, že $L(G) = L(M)$:

- 1. páska obsahuje přijímaný vstupní řetězec w.
- Na 2. pásce se M pokouší pomocí simulace použití přepisovacích pravidel $(\alpha \to \beta) \in P$ vytvořit derivaci w:



1. Stroj nejprve umístí na 2. pásku symbol S.
2. Stroj opakovaně simuluje na 2. pásce provádění pravidel $(\alpha \to \beta) \in P$. Nedeterministicky zvolí pravidlo a také výskyt α na pásce. Při přepisu α na β, $|\alpha| \neq |\beta|$, může využít posuv části užitečného obsahu pásky vlevo či vpravo.
3. Stroj srovná finální obsah 2. pásky s 1. páskou. Shodují-li se, zastaví přechodem do q_F. Jinak posouvá hlavu doleva až do abnormálního zastavení.

Snadno se nyní nahlédne, že skutečně $L(G) = L(M)$. Navíc lze M podobně jako u vícepáskových DTS převést na jednopáskový NTS a ten dále na jednopáskový DTS. □

Jazyky typu 0 = jazyky přijímané TS

Věta 7.5 ***Třída jazyků přijímaných TS (neboli jazyků rekurzívně vyčíslitelných) je shodná se třídou jazyků typu 0.***

Důkaz. Důsledek dvou předchozích vět. □

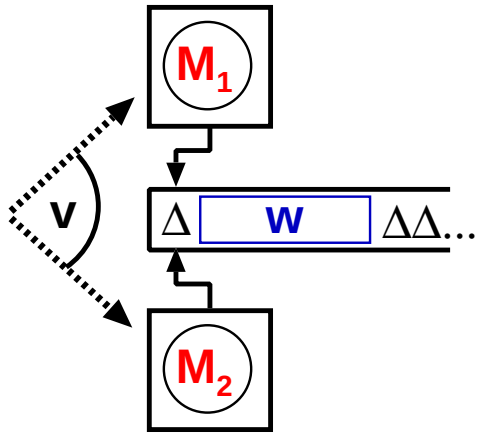
Vlastnosti jazyků rekurzívních a rekurzívně vyčíslitelných

Uzavřenost vůči $\cup$, $\cap$, . a $*$

Věta 7.6 Třídy rekurzívních a rekurzívně vyčíslitelných jazyků jsou uzavřeny vůči operacím $\cup$, $\cap$, . a $*$.

Důkaz. Nechť L_1, L_2 jsou jazyky přijímané TS M_1, M_2. Zřejmě můžeme předpokládat, že množiny stavů TS M_1, M_2 jsou disjunktní.

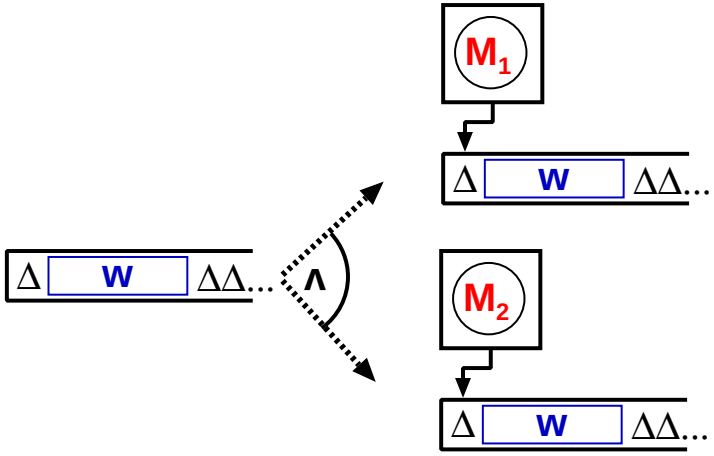
- NTS $M_{L_1 \cup L_2}$, $L(M_{L_1 \cup L_2}) = L_1 \cup L_2$, sestrojíme tak, že sjednotíme po složkách stroje M_1 a M_2, zavedeme nový počáteční stav, z něj nedeterministické přechody přes Δ/Δ do obou původních počátečních stavů a sloučíme původní koncové stavy do jediného nového koncového stavu.



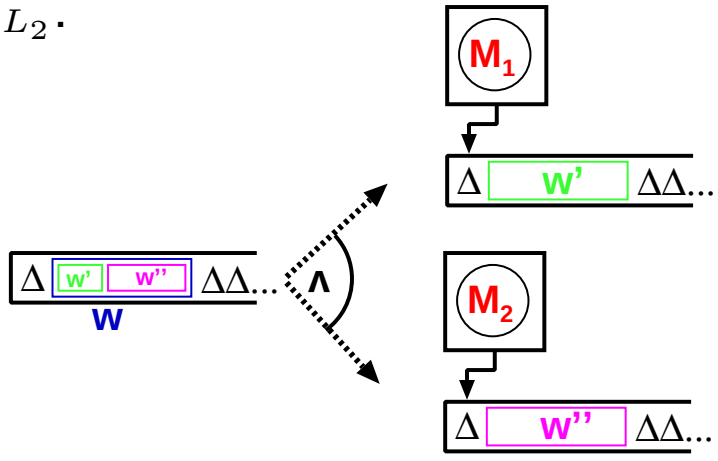
Důkaz pokračuje dále.

Pokračování důkazu.

- Třípáskový TS $M_{L_1 \cap L_2}$, $L(M_{L_1 \cap L_2}) = L_1 \cap L_2$, okopíruje vstup z první pásky na druhou, na ní simuluje stroj M_1, pokud ten přijme, okopíruje vstup z první pásky na třetí, na ní simuluje stroj M_2, pokud i ten přijme, přijme i stroj $M_{L_1 \cap L_2}$.



- Třípáskový NTS $M_{L_1 . L_2}$, $L(M_{L_1 . L_2}) = L_1 . L_2$, okopíruje nedeterministicky zvolený prefix vstupu z první pásky na druhou, na ní simuluje stroj M_1, pokud ten přijme, okopíruje zbytek vstupu z první pásky na třetí, na ní simuluje stroj M_2, pokud i ten přijme, přijme i stroj $M_{L_1 . L_2}$.



Důkaz pokračuje dále.

Pokračování důkazu.

- Dvoupáskový NTS $M_{L_1^*}$, $L(M_{L_1^*}) = L_1^*$, je zobecněním předchozího stroje: po částech kopíruje vstup z první pásky na druhou a na ní simuluje opakovaně stroj M_1. Obsah druhé pásky má ohraničený speciálními značkami a po každé simulaci stroje M_1 ho smaže. Umožňuje samozřejmě posuv pravé značky dále doprava při nedostatku místa.

Jsou-li stroje M_1 a M_2 úplné, je možné vybudovat stroje podle výše uvedených pravidel také jako úplné (u $M_{L_1 \cup L_2}$, $M_{L_1 \cap L_2}$, $M_{L_1.L_2}$ je to okamžité, u $M_{L_1^*}$ nepřipustíme načítání prázdného podřetězce vstupu z 1. na 2. pásku – pouze umožníme jednorázově přijmout prázdný vstup). To dokazuje uzavřenost vůči uvedeným operacím také u rekurzívních jazyků.

□

(Ne)uzavřenost vůči komplementu

Věta 7.7 Třída rekurzívních jazyků je uzavřena vůči komplementu.

Důkaz. TS M přijímající rekurzívní jazyk L vždy zastaví. Snadno upravíme M na M', který při nepřijetí řetězce vždy přejde do unikátního stavu q_{reject}. TS $\overline{M}$, $L(\overline{M}) = \overline{L}$, snadno dostaneme z M' záměnou q_F a q_{reject}. □

❖ Třída rekurzívně vyčíslitelných jazyků není uzavřena vůči komplementu!

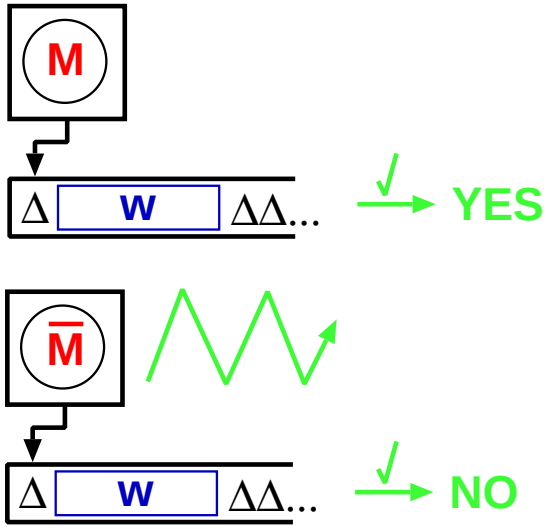
- Výše uvedené konstrukce nelze užít – cyklení zůstane zachováno.
- Důkaz neuzavřenosti bude uveden v dalších přednáškách.

Věta 7.8 Jsou-li L i $\overline{L}$ rekurzívně vyčíslitelné, pak jsou oba rekurzívní.

Důkaz.

Mějme M, $L(M) = L$, a $\overline{M}$, $L(\overline{M}) = \overline{L}$. Úplný TS přijímající L sestrojíme takto:

- Použijeme dvě pásky. Na jedné budeme simulovat M, na druhé $\overline{M}$. Simulace se bude provádět proloženě krok po kroku: krok M, krok $\overline{M}$, krok M, ...
- Přijmeme, právě když by přijal M, zamítneme abnormálním zastavením, právě když by přijal $\overline{M}$. Jedna z těchto situací určitě nastane v konečném počtu kroků.



Existence úplného TS pro $\overline{L}$ plyne z uzavřenosti rekurzívních jazyků vůči komplementu.

□

❖ Důsledkem výše uvedených vět je mj. to, že pro L a $\overline{L}$ musí vždy nastat jedna z následujících situací:

- L i $\overline{L}$ jsou rekurzívní,
- L ani $\overline{L}$ nejsou rekurzívně vyčíslitelné,
- jeden z těchto jazyků je rekurzívně vyčíslitelný, ale ne rekurzívní, druhý není rekurzívně vyčíslitelný.

Lineárně omezené automaty

Lineárně omezené automaty

❖ Lineárně omezený automat (LOA) je nedeterministický TS, který nikdy neopustí tu část pásky, na níž je zapsán jeho vstup.

❖ Formálně můžeme LOA definovat jako NTS, který má v Γ speciální symbol, kterým unikátně označujeme pravý konec vstupu na pásce, přičemž tento symbol není možné přepsat, ani z něj provést posun doprava.

❖ Deterministický LOA můžeme přirozeně definovat jako (deterministický) TS, který nikdy neopustí část pásky se zapsaným vstupem.

❖ Není známo, zda deterministický LOA je či není striktně slabší než LOA.

LOA a kontextové jazyky

Věta 7.9 Třída jazyků, kterou lze generovat kontextovými gramatikami, odpovídá třídě jazyků, které lze přijímat LOA.

Důkaz.

- Uvážíme definici kontextových gramatik jako gramatik s pravidly v podobě $\alpha \to \beta$, kde $|\alpha| \leq |\beta|$, nebo $S \to \varepsilon$.
- *LOA $\longrightarrow$ G1:
  - Použijeme podobnou konstrukci jako u TS $\longrightarrow$ G0.
  - Na počátku vygenerujeme příslušný pracovní prostor, který se pak již nebude měnit: odpadá nekontextové pravidlo $\Delta\Delta] \to \Delta]$.
  - Užití nekontextových pravidel $[q_0\Delta \to \varepsilon$ a $\Delta] \to \varepsilon$ obejdeme (1) zavedením zvláštních koncových nonterminálů integrujících původní informaci a příznak, že se jedná o první/poslední symbol a (2) integrací symbolu reprezentujícího řídící stav a pozici hlavy s následujícím páskovým symbolem.*
- G1 $\longrightarrow$ LOA:
  - Použijeme podobnou konstrukci jako u G0 $\longrightarrow$ TS s tím, že nepovolíme, aby rozsah druhé pásky někdy překročil rozsah první pásky.

□

Kontextové a rekurzívní jazyky

Věta 7.10 Každý kontextový jazyk je rekurzívní.

Důkaz. (Idea)

- Počet konfigurací, které se mohou objevit při přijímání w příslušným LOA M je vzhledem k nemožnosti zvětšovat pracovní prostor pásky konečný: lze shora ohraničit funkcí c^n pro vhodnou konstantu c – exponenciála plyne z nutnosti uvažovat výskyt všechny možných symbolů na všech místech pásky.
- Pro zápis libovolného čísla z intervalu $0, ..., c^n - 1$ nikdy nebude třeba více než n symbolů, užijeme-li c-ární soustavu.
- Můžeme zkonstruovat úplný LOA ekvivalentní s M, který bude mít každý symbol na pásce strukturovaný jako dvojici:
  - S využitím 1. složek těchto dvojic simulujeme M.
  - V 2. složkách počítáme počet kroků; dojde-li k přetečení, odmítneme vstup.

□

Věta 7.11 *Ne* každý rekurzívní jazyk je kontextový.

Důkaz. (Idea) Lze užít techniku diagonalizace prezentovanou dále . □

Vlastnosti kontextových jazyků

Věta 7.12 Třída kontextových jazyků je uzavřena vůči operacím $\cup$, $\cap$, ., $*$ a komplementu.

Důkaz.

- Uzavřenost vůči $\cup$, $\cap$, . a $*$ lze ukázat stejně jako u rekurzívně spočetných jazyků.
- Důkaz uzavřenosti vůči komplementu je značně komplikovaný (všimněme si, že LOA je *nedeterministický* a nelze tudíž užít konstrukce použité u rekurzívních jazyků). Jedná se o přímí důsledek Immerman-Szelepcényiho věty (více viz složitost, 20 let otevřený problém, vyřešen 1987, Gödelova cena 1995, zásadní československá stopa v teoretické informatice).

□

❖ Poznamenejme, že již víme, že u kontextových jazyků

- lze rozhodovat členství věty do jazyka (rekurzívnost) a
- nelze rozhodovat inkluzi jazyků (neplatí ani pro bezkontextové jazyky).

❖ Dále lze ukázat, že pro kontextové jazyky nelze rozhodovat prázdnost jazyka (užije se redukce z Postova problému přiřazení – viz další přednášky).

Vztah vyčíslitelných funkcí a Turingových strojů

Základy teorie vyčíslitelných funkcí

Budeme se snažit identifikovat takové funkce, které jsou „spočitatelné“, tj. vyčíslitelné v obecném smyslu (bez ohledu na konkrétní výpočetní systém). Abychom snížili extrémní velikost třídy těchto funkcí, která je dána také varietou definičních oborů a oborů hodnot, omezíme se, uvažujíce možnost kódování, na funkce tvaru:

$$\boxed{f : \mathbb{N}^m \to \mathbb{N}^n}$$

kde $\mathbb{N} = \{0, 1, 2, \ldots\}, \quad m, n \in \mathbb{N}$

❖ Konvence: n-tici $(x_1, x_2, \ldots, x_n) \in \mathbb{N}^n$ budeme označovat jako $\overline{x}$

❖ Klasifikace parciálních funkcí:

- *Totální funkce* – definovaná pro každý $\overline{x} \in \mathbb{N}^m$
- *Striktně parciální funkce* – $\exists\, \overline{x} \in \mathbb{N}^m : f(\overline{x}) = \perp$

Počáteční funkce

Hierarchie vyčíslitelných funkcí je založena na dostatečně elementárních tzv. *počátečních funkcích*, které tvoří „stavební kameny“ vyšších funkcí.

❖ Jsou to tyto funkce:

1. *Nulová funkce* (zero function): $\xi() = 0$
   zobrazuje „prázdnou n-tici“ $\mapsto 0$

2. *Funkce následníka* (successor function): $\sigma : \mathbb{N} \to \mathbb{N}$
   $\sigma(x) = x + 1$

3. *Projekce* (projection): $\pi_k^n : \mathbb{N}^n \to \mathbb{N}$
   Vybírá z n-tice k-tou složku, např.: $\pi_2^3(7, 6, 4) = 6$ a $\pi_1^2(5, 17) = 5$
   Speciální případ: $\pi_0^n : \mathbb{N}^n \to \mathbb{N}^0$, tj. např. $\pi_0^3(1, 2, 3) = ()$

Vytváření složitějších funkcí

Nyní definujme tři způsoby vytváření nových, složitějších funkcí:

1. *Kombinace*:
   Kombinací dvou funkcí $f : \mathbb{N}^k \to \mathbb{N}^m$ a $g : \mathbb{N}^k \to \mathbb{N}^n$ získáme funkci, pro kterou:

$$\boxed{\begin{array}{l} f \times g : \mathbb{N}^k \to \mathbb{N}^{m+n} \\ f \times g(\overline{x}) = (f(\overline{x}), g(\overline{x})), \overline{x} \in \mathbb{N}^k \end{array}}$$

   Např.: $\pi_1^3 \times \pi_3^3(4, 12, 8) = (4, 8)$

2. *Kompozice*:
   Kompozice dvou funkcí $f : \mathbb{N}^k \to \mathbb{N}^m$ a $g : \mathbb{N}^m \to \mathbb{N}^n$ je funkce, pro kterou:

$$\boxed{\begin{array}{l} g \circ f : \mathbb{N}^k \to \mathbb{N}^n \\ g \circ f(\overline{x}) = g(f(\overline{x})), \overline{x} \in \mathbb{N}^k \end{array}}$$

   Např.:
   $\sigma \circ \xi() = 1$
   $\sigma \circ \sigma \circ \xi() = 2$

3. *Primitivní rekurze*:

Příklad 7.10 Předpokládejme, že chceme definovat funkci násobení $mult : \mathbb{N}^2 \to \mathbb{N}$.

$$mult(x, y) = \underbrace{x + \cdots + x}_{y}$$

Zřejmě:

(a) Pro $y = 0$ platí $x * 0 = 0$

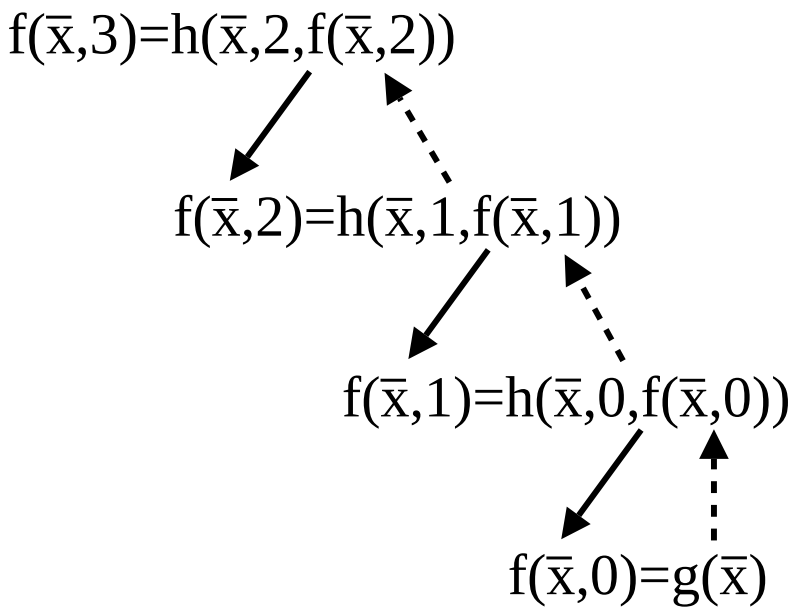
(b) Pro $y > 0$ je výsledek $x + mult(x, y - 1)$

Takže funkci $mult$ můžeme definovat následujícím předpisem:
$mult(x, 0) = 0$
$mult(x, y + 1) = x + mult(x, y)$

Primitivní rekurze je technika, která umožňuje vytvořit funkci $f : \mathbb{N}^{k+1} \to \mathbb{N}^m$ na základě jiných dvou funkcí $g : \mathbb{N}^k \to \mathbb{N}^m$ a $h : \mathbb{N}^{k+m+1} \to \mathbb{N}^m$ rovnicemi:

$$\boxed{\begin{array}{l} f(\overline{x}, 0) = g(\overline{x}) \\ f(\overline{x}, y+1) = h(\overline{x}, y, f(\overline{x}, y)),\ \overline{x} \in \mathbb{N}^k \end{array}}$$

Ilustrace schématu vyčíslení (pro $y = 3$):



Primitivně rekurzivní funkce

Definice 7.7 *Třída primitivně rekurzivních funkcí* je třída totálních funkcí, které mohou být vytvořeny z počátečních funkcí konečnou aplikací:

- kombinace
- kompozice
- primitivní rekurze

❖ Charakteristika (bez důkazu): Primitivně rekurzivní funkce je taková, kterou lze zapsat jako počítačový program obsahující pouze konečné for cykly a žádné while cykly nebo skoky.

Funkce mimo primitivně rekurzivní funkce

Existují funkce, které jsou vyčíslitelné a nejsou primitivně rekurzivními funkcemi. Jsou to všechny striktně parciální funkce (jako div), ale i totální funkce.

Věta 7.13 Existuje totální funkce z $\mathbb{N}$ do $\mathbb{N}$, která není primitivně rekurzivní.

Důkaz.
Definice funkcí, které jsou primitivně rekurzivní budeme chápat jako řetězce a můžeme je uspořádat v lexikografickém pořadí s označením $f_1, f_2, \ldots, f_n, \ldots$
Definujeme nyní funkci $f : \mathbb{N} \to \mathbb{N}$ tak, že $f(n) = f_n(n) + 1$ pro $\forall n \in \mathbb{N} \setminus \{0\}$. f je jasně totální a vyčíslitelná. f však není primitivně rekurzivní (kdyby byla, pak $f \equiv f_m$ pro nějaké $m \in \mathbb{N}$. Pak ale $f(m) = f_m(m)$ a ne $f_m(m) + 1$, jak vyžaduje definice funkce f).

□

Příkladem totální funkce, která není primitivně rekurzivní byla prezentována W. Ackermannem (1928) a nazývá se *Ackermannova funkce*. Je dána rovnicemi:

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

Parciálně rekurzivní funkce

K rozšíření třídy vyčíslitelných funkcí za totální vyčíslitelné funkce zavedeme techniku známou pod názvem *minimalizace*. Tato technika umožňuje vytvořit funkci $f : \mathbb{N}^n \to \mathbb{N}$ z jiné funkce $g : \mathbb{N}^{n+1} \to \mathbb{N}$ předpisem, v němž $f(\overline{x})$ je nejmenší y takové, že:

1. $g(\overline{x}, y) = 0$
2. $g(\overline{x}, z)$ je definována pro $\forall z < y$, $z \in \mathbb{N}$

❖ Funkce definovaná minimalizací je skutečně vyčíslitelná. Výpočet hodnoty $f(\overline{x})$ zahrnuje výpočet $g(\overline{x}, 0), g(\overline{x}, 1), \ldots$ tak dlouho, pokud nedostaneme:

- $g(\overline{x}, y) = 0$ $\qquad (f(\overline{x}) = y)$
- $g(\overline{x}, z)$ je nedefinována $\qquad (f(\overline{x})$ je nedefinována)

Definice 7.8 *Třída parciálně rekurzivních funkcí* je třída parciálních funkcí, které mohou být vytvořeny z počátečních funkcí aplikací:

- kombinace
- kompozice
- primitivní rekurze
- minimalizace

Turingovsky vyčíslitelné funkce

Definice 7.9 Turingův stroj $M = (Q, \Sigma, \Gamma, \delta, q_0, q_F)$ *vyčísluje (počítá)* parciální funkci $f : \Sigma^{*m} \to \Sigma_1^{*n}$, $\Sigma_1 \subseteq \Gamma, \Delta \notin \Sigma_1$, jestliže pro každé $(w_1, w_2, \ldots, w_m) \in \Sigma^{*m}$ a odpovídající počáteční konfiguraci $\underline{\Delta} w_1 \Delta w_2 \Delta \ldots \Delta w_m \Delta^\omega$ stroj M:

1. v případě, že $f(w_1, \ldots, w_m)$ je definována, pak M zastaví a páska obsahuje $\underline{\Delta} v_1 \Delta v_2 \Delta \ldots \Delta v_n \Delta\Delta\Delta$, kde $(v_1, v_2, \ldots, v_n) = f(w_1, \ldots, w_m)$
2. v případě, že $f(w_1, \ldots, w_m)$ není definována, M cyklí nebo zastaví abnormálně.

Parciální funkce, kterou může počítat nějaký Turingův stroj se nazývá funkcí *Turingovsky vyčíslitelnou*.

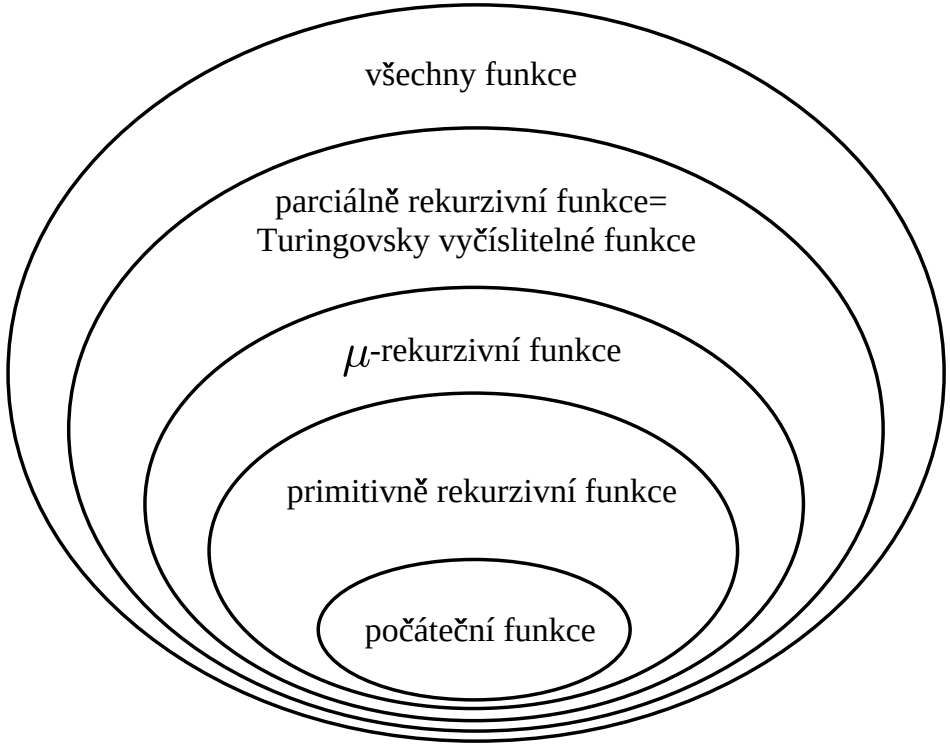
Věta 7.14 Každá parciálně rekurzivní funkce je Turingovsky vyčíslitelná.

Idea důkazu: počáteční funkce, kombinaci, kompozici, projekci, primitivní rekurzi i minimalizaci lze implementovat pomocí Turingova stroje.

Věta 7.15 Každý výpočetní proces prováděný Turingovým strojem je procesem vyčíslení nějaké parciálně rekurzivní funkce.

Idea důkazu: Definujme primitivně rekursivní funkci provádějící k kroků daného TS. Využijeme techniku minimalizaze k výpočtu počtu kroků nutných k zastavení TS (může vrátit nedefinováno).

Hierarchie funkcí



❖ Třída totálních vyčíslitelných funkcí se nazývá *μ-rekurzivní funkce*.

❖ Příklad funkce, která není Turingovsky vyčíslitelná:

Nechť L je libovolný jazyk.

Funkce $f(w) = \begin{cases} |w| & \text{jestliže } w \in L \\ 0 & \text{jestliže } w \notin L \end{cases}$