AUTOMATY A GRAMATIKY

12

Varianty Turingových strojů
Lineární omezení TS
Rekurzivní jazyky
Univerzální Turingův stroj
Problém zastavení

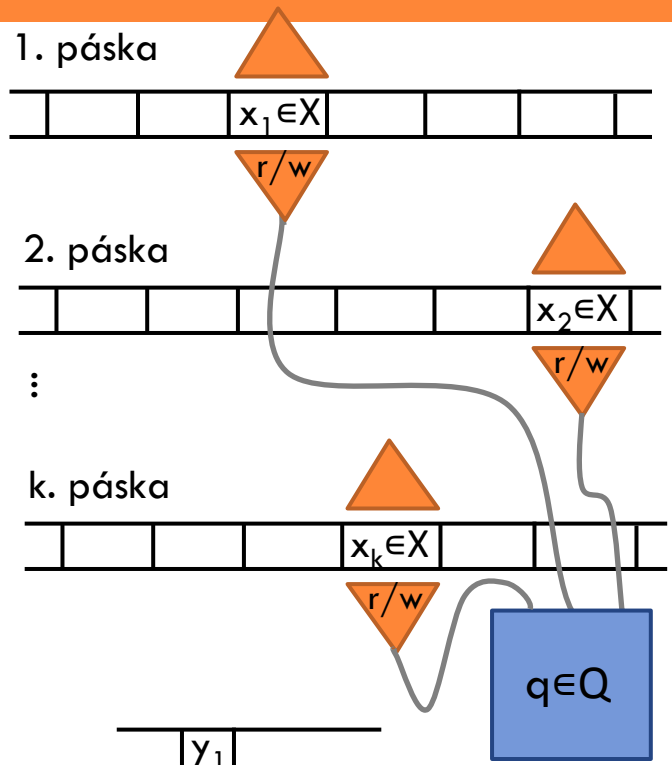
Pavel Surynek

Univerzita Karlova v Praze

Matematicko-fyzikální fakulta
Katedra teoretické informatiky a matematické logiky

Vícepáskový Turingův stroj (1)

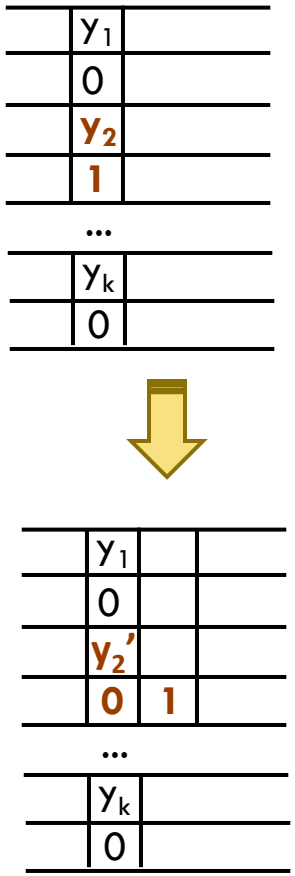
- **k-páskový** TS T = (Q, X, δ, q_0, b, F), kde
  - δ: $(Q\text{-}F)\times X^k \rightarrow Q\times X^k\times\{-1, 0, +1\}^k$
    - případně nedeterministický - δ: $(Q\text{-}F)\times X^k \rightarrow 2^{Q\times X^k\times\{-1, 0, +1\}^k}$
- existuje 1-páskový TS T', že L(T')=L(T)
  - případně nedeterministický 1-páskový T'
- T' použije **2k-stopou pásku**
  - k-stop na obsah původních pásek, zbylé stopy na označení čtených pozic
  - i-tá buňka pásky T' bude obsahovat 2k-tici
    - $(y_1,p_1,y_2,p_2,\ldots,y_k,p_k)$, kde
      - y_j je obsah i-té buňky j-té pásky stroje T
      - $p_j = 1$ když stroj T čte i-tou buňku j-té pásky, $p_j = 0$ jinak



y_1	
0	
y_2	
1	
...	
y_k	
0	

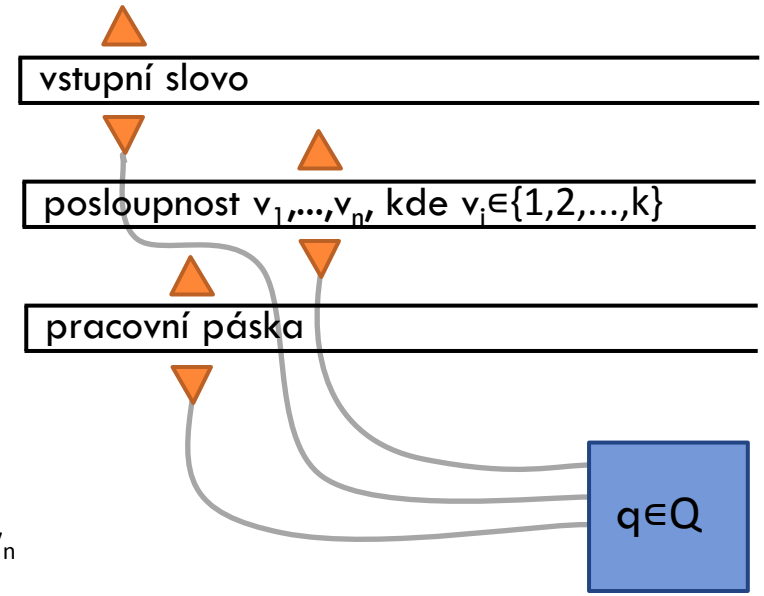
Vícepáskový Turingův stroj (2)

- **simulace kroku**
  - na začátku T’ čte nejlevější buňku, že existuje páska původního T, kde T čte odpovídající buňku
  - T’ postupuje **vpravo**
    - narazí-li na buňku obsahující $(y_1,p_1,y_2,p_2,\ldots,y_k,p_k)$ s $p_j=1$, uloží y_j do stavu
    - po přečtení všech čtených pozic určí výsledek přechodové funkce δ a uloží jej do stavu
  - T’ postupuje **vlevo**
    - narazí-li na buňku obsahující $(y_1,p_1,y_2,p_2,\ldots,y_k,p_k)$ s $p_j=1$, aktualizuje y_j podle přechodové funkce a $p_j=1$ na sousedních buňkách
    - po aktualizaci všech čtených pozic simulace dalšího kroku
  - je-li simulovaný stav přijímajícím stavem T, T’ přijímá



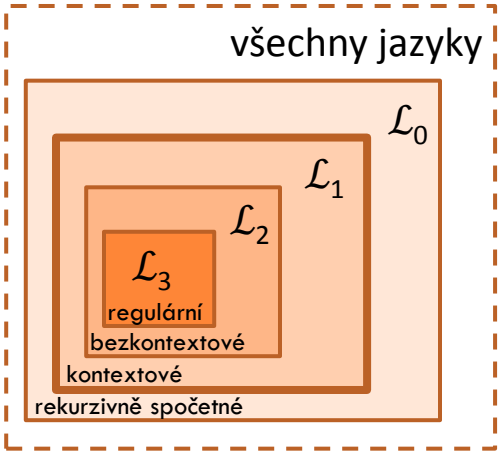
Nedeterministický TS

- **nedeterministický** 1-páskový TS $T = (Q, X, \delta, q_0, b, F)$, kde
  - $\delta: (Q-F)\times X \rightarrow 2^{Q\times X\times\{-1, 0, +1\}}$
- existuje 3-páskový **deterministický** TS T', že L(T') = L(T)
  - $(\exists k \in \mathbb{N})\ |\delta(q,x)| \leq k$ pro každé $q \in Q$ a $x \in X$
    - volba prvků ve výsledku přechodové funkce lze zakódovat jako posloupnost $v_1,\ldots,v_n$, kde $v_i \in \{1,2,\ldots,k\}$
  - 1. páska je pouze vstupní
    - obsahuje vstupní slovo, T' na ni nic nepíše
  - 2. páska obsahuje volbu výsledků přechodové funkce
    - T' systematicky na tuto pásku generuje všechny konečné posloupnosti $v_1,\ldots,v_n$, s $v_i \in \{1,2,\ldots,k\}$
  - 3. páska je pracovní
    - pracovní prostor pro simulovaný TS T



- **simulace výpočtu**
  - T' vygeneruje na 2. pásku **další** konečnou posloupnost $v_1,\ldots,v_n$
  - na 3. pásku zkopíruje vstupní slovo z 1. pásky
  - v s-tém kroku pro $s \leq n$, čte T' v_s na druhé pásce, x na 3. pásce a nachází se ve stavu q
    - když $|\delta(q,x)| < v_s$ simulace pokračuje generováním další posloupnosti na 2. pásce
    - jinak T' na 3. pásce provede krok podle v_s-tého prvku $\delta(q,x)$
  - T' přijme, pokud simulovaný T přijme (při některém simulovaném výpočtu)

Lineární omezení



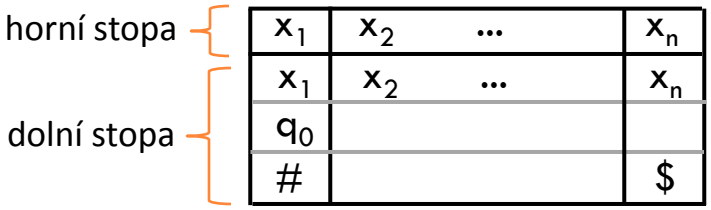
- nedeterministický Turingův stroj
  $T = (Q, X \cup \{\#,\$\}, \delta, q_0, b, F)$ je **lineárně omezený**, jestliže
  - na pásce je označen levý a pravý konec symbolem # resp. $ a výpočet nikdy nepřekročí # vlevo resp. $ vpravo
    - symboly # a $ nelze přepsat
  - počáteční konfigurace se vstupním sloven $w \in X^*$ je $(\lambda, q_0, \#w\$)$
    - lineární omezení lze zavést i pro k-pásek
- lineárně omezené Turingovy stroje přijímají právě **kontextové jazyky**
  - souvislost s nezkracující gramatikou
    - délka odvozeného slova nikdy nepřekročí délku nakonec odvozeného terminálního slova

Pozn.:
deterministické lineárně omezené TS definují třídu tzv. deterministických kontextových jazyků

LBA problém: platí deterministické kontextové = kontextové**?**

Lineární omezení ⇒ nezkracování

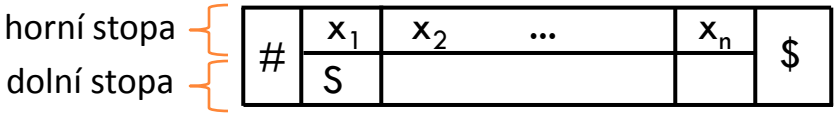
- mějme **lineárně omezený** TS $T = (Q, X \cup \{\#,\$\}, \delta, q_0, b, F)$
  - nezkracující gramatikou chceme vygenerovat slovo $w=x_1x_2...x_n$ s $x_i \in X$, že $w \in L(T)$
  - použijeme **více stop** v neterminálech
    - (nedeterministicky) vygenerujeme $(x_1,[q_0,\#,x_1])$, (x_2,x_2), ... $(x_n,[\$,x_n])$
      - horní stopa obsahuje w
      - dolní stopa obsahuje prostor pro výpočet
        - dolní stopa obsahuje stav, symbol, případně indikátor konce



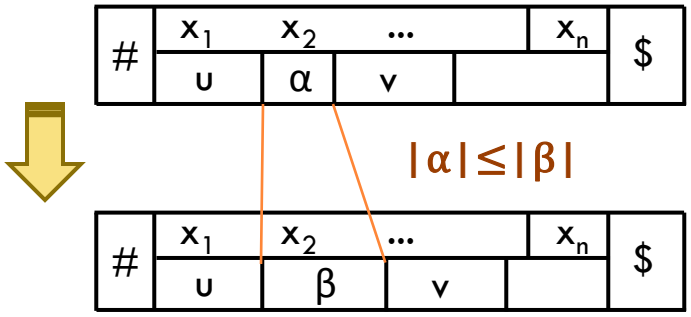
    - nezkracujícími pravidly simulujeme výpočet v dolní stopě
      - stejně jako u převodu TS ⇒ gramatika (pouze přizpůsobit více stopám)
      - při dosažení přijímajícího stavu vymažeme dolní stopu
  - přijímání prázdného slova ošetříme zvlášť
    - pokud $\lambda \in L(T)$, přidáme pravidlo $S \rightarrow \lambda$

Nezkracování ⇒ lineární omezení

- mějme **nezkracující gramatiku** G = (V_N, V_T, S, P)
  - odvozování slova w=$x_1x_2...x_n$ s x_i∈X pomocí G budeme simulovat lineárně omezeným TS T
    - T bude mít dvoustopou pásku
      - horní stopa bude obsahovat vstupní slovo
      - v dolní stopě bude probíhat odvozování slova



    - nedeterministicky vybereme pravidlo α → β
      - důležitost nedeterminismu lineárně omezených TS
    - určí výskyt α v dolní stopě (při více výskytech vybere nedeterministicky)
      - přepíše výskyt α na β
        - vytvoření prostoru
        - přepis
    - jsou-li na druhé stopě samé terminály
      - porovná druhou a první stopu
        - při shodě přijme



Rekurzivní jazyky

- Turingův stroj $T = (Q, X, \delta, q_0, b, F)$ **rozhoduje** jazyk L, jestliže $L(T) = L$ a T skončí pro každé slovo $w \in X^*$
  - nepřijetí slova u Turigova stroje nastane, když
    - (a) skončí v nepřijímacím stavu
    - (b) neskončí - zacyklí se
      - problematická situace
        - když stále ještě neskončil, nelze říci, zda nakonec nepřijme později
  - jazyk L, který je rozhodován nějakým TS, se nazývá **rekurzivní**
- **Postova věta:**
  - jazyk L je **rekurzivní** ⇔ L a -L jsou **rekurzivně spočetné**
    - ⇒
      - pro L máme rozhodovací TS T, odtud L je rekurzivně spočetný, prohozením přijímajících a nepřijímajících stavů v T máme rozhodovací TS pro -L, tedy -L je rekurzivně spočetný
    - ⇐
      - pro L máme TS T_1, že $L(T_1) = L$, pro -L máme TS T_2, že $L(T_2) = -L$
      - zkonstruujeme TS, který paralelně simuluje T_1 a T_2 nad daným slovem
        - přijme-li T_1, přijmeme
        - přijme-li T_2, odmítneme
        - platí, že T_1 nebo T_2 přijme a toto nastane po konečně mnoha krocích

Univerzální Turingův stroj (1)

- uvažujme 1-páskové deterministické Turingovy stroje pracující s abecedou {0, 1, (b)}
  - očíslujeme prvky komponent TS
    - **Q - stavy** q_1 (počáteční), q_2 (přijímající), q_3, ... (další stavy)
    - **X - symboly** x_1 (symbol 0), x_2 (symbol 1), x_3 (symbol b), x_4, ... (další symboly)
    - **{-1,0,+1} - směry** d_1 (směr -1), d_2 (směr 0), d_3 (směr +1)
- kódování **přechodové funkce**
  - položka přechodové funkce $\delta(q_i,x_j)=(q_k,x_l,d_m)$ bude kódována slovem
    - $0^i10^j10^k10^l10^m$
    - i,j,k,l,m>0, kód neobsahuje podslovo 11
      - 11 může být využito jako oddělovač
- kódování **Turingova stroje T - kód(T)**
  - konkatenace kódů jednotlivých položek přechodové funkce
    - $kód_1.11.kód_2.11.kód_3.11...11.kód_n$, kde
      - $kód_i$ je kód i-té položky přechodové funkce

Univerzální Turingův stroj (2)

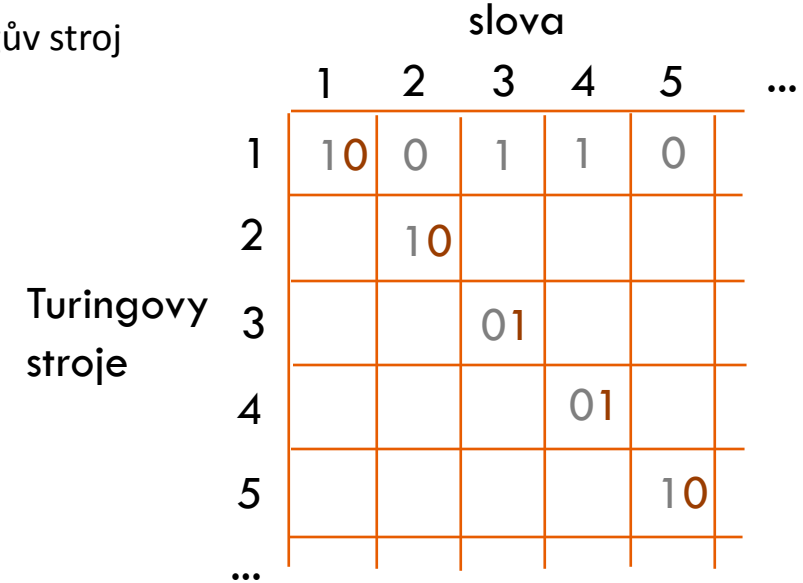
- **univerzální Turingův stroj** U pro abecedu {0, 1, (b)} dostane na vstupu kód Turingova stroje T a vstupní slovo w∈X* ve tvaru
  - kód(T).111.w
- U **simuluje** práci T nad slovem w
  - **předpoklady**
    - w zakóduje
      - 1 → 001, 0 → 000
    - označí **písmeno čtené T**
      - například pomocí 011 (čtený symbol 1) resp. 010 (čtený symbol 0)
        - na začátku bude označené první písmeno w
    - na konec vstupního slova zapíše kód počátečního stavu (oddělený 111)
      - reprezentuje **aktuální stav**
        - na začátku je zapsáno 111.0
    - další prostor na pásce je využit při simulaci
  - v cyklu se **opakuje**
    - zkontroluje, zda je aktuální stav přijímající v T
      - pokud ano, U přijme vstup
    - v kód(T) vyhledá aplikovatelný přechod pro aktuální stav a čtené písmeno
      - nalezne-li přechod, aplikuje jej - přepíše se aktuální stav a čtené písmeno
      - nenajde-li přechod, simulace končí, U nepřijme vstup

Pozn.:
univerzální TS U má abecedu {0, 1, b}, tj. stejnou abecedu jako simulovaný TS T

Pozn.:
v technické praxi U odpovídá např.:
interpret Pascalu napsaný v Pascalu nebo
interpret Javy napsaný v Javě
....

Diagonalizace s TS

- pro TS T lze kód(T) interpretovat jako pořadové číslo T
  - interpretace $b(b_{n-1}...b_1b_0) = \sum b_i 2^i + 2^n$, kde $b_i \in \{0,1\}$ pro i=0,1,...,n-1
  - T je **b(kód(T))-tý Turingův stroj**
    - jestliže číslu i neodpovídá kód, který je smysluplný, předpokládáme, že i-tý TS nic nepřijímá
      - T_i nechť je i-tý Turingův stroj



  - definujeme jazyk $L_d = \{ w \mid w \in \{0,1\}^* \wedge w \text{ je i-tý binární řetězec} \Rightarrow w \notin L(T_i) \}$
    - pro L_d neexistuje TS, který jej přijímá, tedy **L_d není rekurzivně spočetný**

Univerzální jazyk

- definujeme jazyk L_u = { kód(T).111.w | TS T přijímá w }
  - L_u se nazývá **univerzální jazyk**
  - L_u = L(U) pro U univerzální Turingův stroj
    - **L_u je rekurzivně spočetný jazyk** (díky univerzálnímu TS)
- **L_u není rekurzivní** (nelze rozhodovat)
  - pro spor předpokládejme, že L_u rekurzivní je
  - podle Postovy věty L_u i $-L_u$ jsou rekurzivně spočetné
  - když $-L_u$ je rekurzivně spočetný, je i L_d rekurzivně spočetný, což je spor
    - mějme TS T, že L(T) = $-L_u$, zkonstruujeme TS D, že L(D) = L_d
      - vstup x
      - D ověří, zda x je kód nějakého TS
        - pokud ne, D přijme
      - zapíše na pásku x.111.x a spustí na tomto vstupu T
        - když T přijme, D přijme

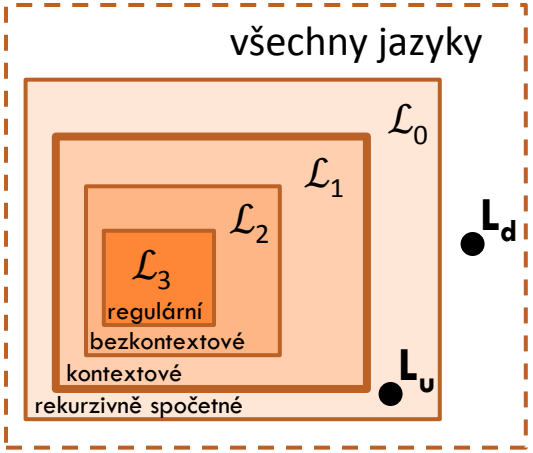
Pozn.:
alternativně bez Postovy věty:
když rozhodovací T' pro L_u
(i) přijme, pak D nepřijme
(ii) nepřijme, pak D přijme

- přijímání u TS se někdy definuje pomocí **zastavení** (nepokračování přechodové funkce)
  - přijímání zastavením dává stejné jazyky jako přijímání přijímajícím stavem
  - rozhodování L_u pak znamená rozhodovat o zastavení daného TS pro dané slovo
    - L_u je znám také jako **problém zastavení Turingova stroje (Halting problem)**

Důležité **důsledky**

- rekurzivně spočetné jazyky **nejsou uzavřené na doplňky**
  - zatímco kontextové jsou uzavřené na doplňky
    - L_u je rekurzivně spočetný, ale není kontextový
    - L_d ukazuje, že nikoli všechny jazyky je možné přijímat TS či generovat gramatikou

Pozn.:
rozhodnutelnost bezkontextových jazyků ukazuje algoritmus CYK



- $\mathcal{L}_0 \supsetneq$ **rekurzivní jazyky** $\supsetneq \mathcal{L}_1$
  - neformálně
    - kontextové jazyky jsou rozhodnutelné (rekurzivní)
      - pro daný vstup w vygenerujeme všechna možná odvoditelná slova v, že $|v| \leq |w|$, je-li mezi nimi w přijmeme, jinak odmítneme
        - na pásce ukládáme dosud vygenerovaná slova a kontrolujeme opakování
      - máme horní odhad prostorové složitosti
      - rozhodnutelný problém s větší prostorovou složitostí než je nelezený horní odhad není kontextový