Univerzální TM, Diagonální jazyk, Obecné gramatiky

Marta Vomlelová

MS 303

Rekurzivní jazyky

Definition 10.1 (TM zastaví)

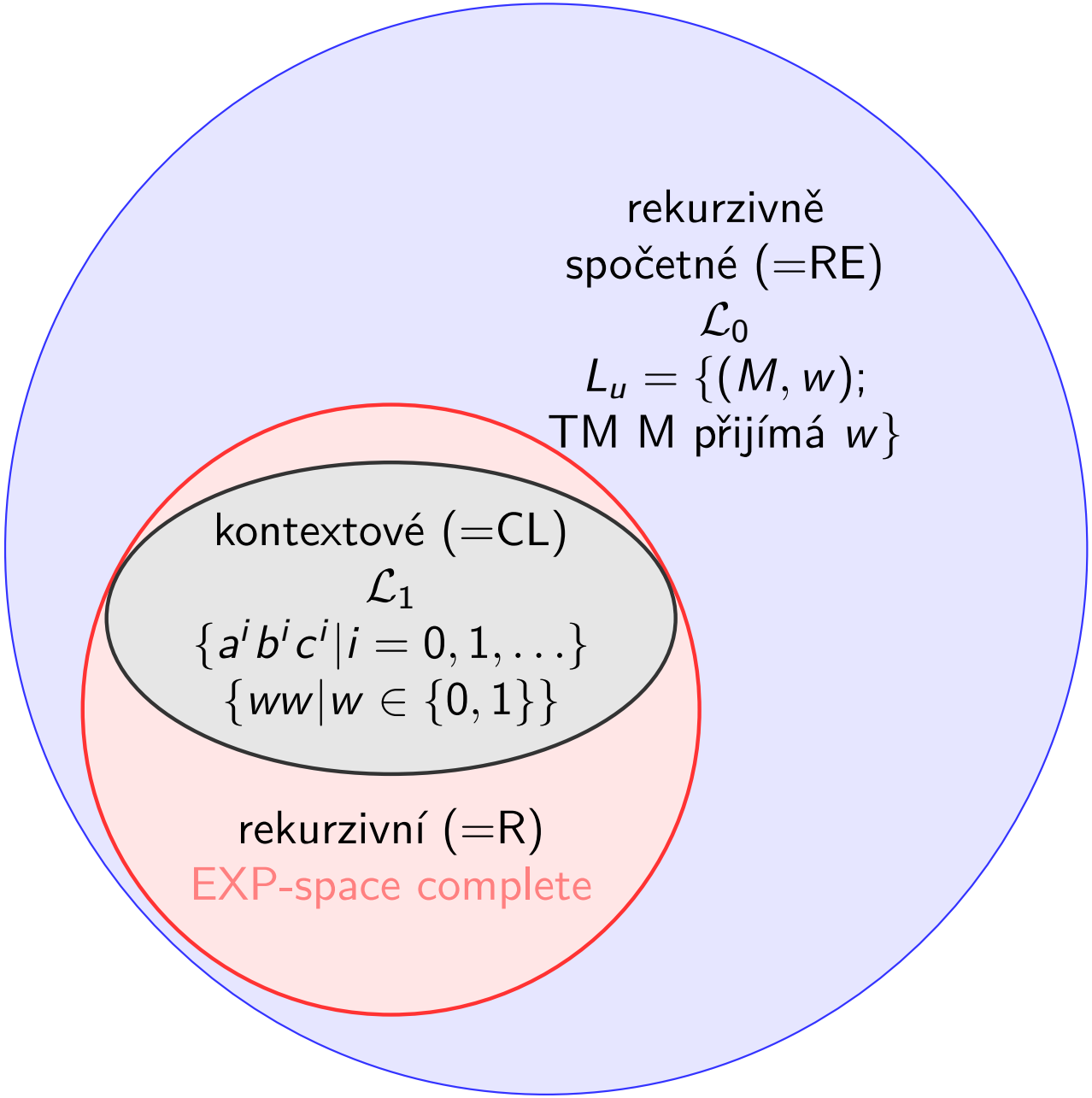
TM **zastaví** pokud vstoupí do stavu q, s čteným symbolem X, a není instrukce pro tuto konfiguraci, t.j., $\delta(q, X)$ není definováno.

- Z definice, v přijímajícím stavu $q \in F$ TM zastaví,
- dokud nezastaví, nevíme, jestli přijme nebo nepřijme slovo.

Definition 10.2 (Rekurzivní jazyky)

Říkáme, že TM M **rozhoduje jazyk** L, pokud $L = L(M)$ a pro každé $w \in \Sigma^*$ stroj nad w zastaví.
Jazyky rozhodnutelné TM nazýváme **rekurzivní jazyky**.

Hierarchie jazyků (kontextové a výš)



$L_d = \{w;$ TM s kódem w nepřijímá vstup $w\}$

Jazyk který není rekurzivně spočetný

Směřujeme k důkazu nerozhodnutelnosti jazyka dvojic (M, w) takových, že:

- M je binárně kódovaný Turingův stroj s abecedou $\{0, 1\}$,
- $w = M \in \{0, 1\}^*$ a
- M nepřijímá vstup w.

Postup:

- Kódování TM binárním kódem pro libovolný počet stavů TM.
- Kód TM vezmeme TM jako binární řetězec.
- Pokud kód nedává smysl, reprezentuje TM bez transakcí. Tedy každý kód reprezentuje nějaký TM.
- **Diagonální jazyk** L_d;
  $L_d = \{w; \text{TM reprezentovaný jako } w \text{ takový, že } \textbf{nepřijímá } w\}$.
- Neexistuje TM přijímající jazyk L_d. Spuštění takového stroje na vlastním kódu by vedlo k paradoxu.

Jazyk L_d není rekurzivně spočetný. Proto $\overline{L_d}$ není rekurzivní. Lze dokázat, že $\overline{L_d}$ je rekurzivně spočetný.

Kódování Turingova sroje

- Pro kódování TM $M = (Q, \{0,1\}, \Gamma, \delta, q_1, B, \{q_2\})$ očíslujeme stavy, symboly a směry L, R.
- Předpokládejme:
  - Počáteční stav je vždy q_1.
  - Stav q_2 je vždy jediný koncový stav (nepotřebujeme víc, TM zastaví).
  - První symbol je vždy 0, druhý 1, třetí B, prázdný symbol. Ostatní symboly pásky očíslujeme libovolně.
  - Směr L je 1, směr R je 2.
- Jeden krok $\delta(q_i, X_j) = (q_k, X_l, D_m)$ kódujeme: $0^i 1 0^j 1 0^k 1 0^l 1 0^m$. Všechna $i, j, k, l, m \geq 1$ takže se dvě jedničky za sebou nevyskytují.
- Celý TM se skládá z kódů všech přechodů v nějakém pořadí oddělených dvojicemi jedniček 11: $C_1 11 C_2 11 \ldots C_{n-1} 11 C_n$.

Uspořádání řetězců $\{0,1\}^*$ (technický detail)

- Řetězce bereme uspořádané podle délky, stejně dlouhé uspořádáme lexikograficky.
- První je ϵ, druhý 0, třetí 1, čtvrtý 00 atd.
- i–tý řetězec označujeme w_i.

Příklad kódování TM

Turingův stroj

$M = (\{q_1, q_2, q_3\}, \{0, 1\}, \{0, 1, B\}, \delta, q_1, B, \{q_2\})$

δ	0	1	B
$\rightarrow q_1$		$(q_3, 0, R)$	
$*q_2$			
q_3	$(q_1, 1, R)$	$(q_2, 0, R)$	$(q_3, 1, L)$.

- Kód pro transakce:

C_1	C_2	C_3	C_4
0100100010100	0001010100100	00010010010100	000100010010010

- Kód celého TM:
  0100100010100110001010100100110001001001010011000100010010010.

Definition 10.3 (Diagonální jazyk)

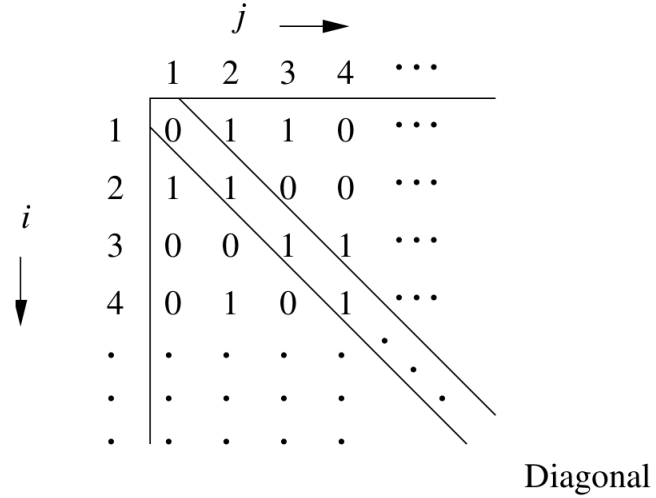
Diagonální jazyk L_d je definovaný
$L_d = \{w \in \{0,1\}^*; \text{TM reprezentovaný jako } w \text{ který nepřijímá slovo } w\}$.

$L_d = \{w; \text{na diagonále je } 0\}$.

Theorem 10.1

L_d není rekurzivně spočetný jazyk, tj. neexistuje TM přijímající L_d.

i - kód TM
j - vstup w
0/1 - přijímá/nepřijímá



Proof.

- Předpokládejme L_d je RE, $L_d = L(M_d)$ pro nějaký TM M_d.
- Jeho jazyk je $\{0,1\}$, tedy je v seznamu na obrázku: 'Přijímá TM M_i vstupní slovo w_j?'
- Alespoň jeden řetězec ho kóduje, řekněme $code(M_d) = w_d$.
- Je $w_d \in L_d$
  - Pokud 'ano', na diagonále má být 0, tj. $w_d \notin L(M_d) = L_d$, spor.
  - Pokud 'ne', na diagonále má být 1, $w_d \in L(M_d) = L_d$, spor.

Proto takový M_d neexistuje. Tedy L_d není rekurzivně spočetný.

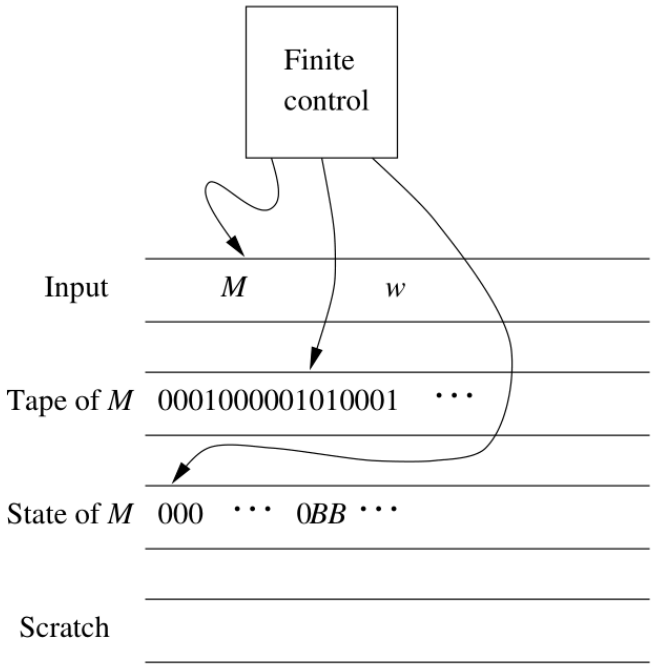
Univerzální Turingův stroj

Definition 10.4 (Univerzální jazyk)

Definujeme **univerzální jazyk** L_u jakožto množinu binárních řetězců které kódují pár (M, w), kde M je TM a $w \in L(M)$, tj. M přijímá slovo w.
TM přijímající L_u se nazývá **Univerzální Turingův stroj**.

Theorem 10.2 (Existence Univerzálního Turingova stroje)

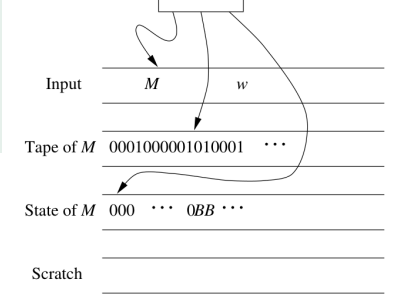
Existuje Turingův stroj U, pro který $L_u = L(U)$.



Popíšeme U jako vícepáskový Turingův stroj.

- Přechody M jsou napsány na první pásce spolu s řetězcem w.
- Na druhé pásce simulujeme výpočet M, používající formát jako kód M, tj. symboly 0^i oddělené jedničkou 1.
- Třetí páska obsahuje stav M reprezentovaný i nulami.

Operace univerzálního Turingova stroje



Operace U jsou následující:

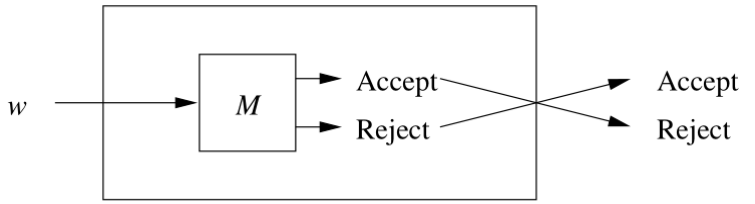
- Otestuj, zda je kód M legitimní;
  pokud ne, U zastav bez přijetí.
- Inicializuj druhou pásku kódovaným slovem w: 10 pro 0 ve w, 100 pro 1; blank jsou nechané prázdné a nahrazeny 1000 pouze 'v případě potřeby'.
- Napiš 0, počáteční stav M, na třetí pásku. Posuň hlavu druhé pásky na první simulované políčko.
- Simuluj jednotlivé přechody M
  - Najdi na první pásce správnou transakci $0^i10^j10^k10^l10^m$, 0^i na pásce 3, 0^j na pásce 2.
  - Změň obsah pásky 3 na 0^k.
  - Nahraď 0^j na 2. pásce řetězcem 0^l. Použij čtvrtou 'scratch tape' pro správné mezery.
  - Posuň hlavu 2. pásky na pozici vedle 1 vlevo nebo vpravo, podle pohybu m.
- Pokud jsme nenašli instrukci pro M, zastavíme.
- Pokud M přejde do přijímajícího stavu, pak U také přijme. □

Tedy máme TM přijímající $L_u = \{(M, w) \mid w \in L(M)\}$.

Postova věta: $(L \in RE \ \& \ \overline{L} \in RE) \Leftrightarrow L$ je rekurzivní

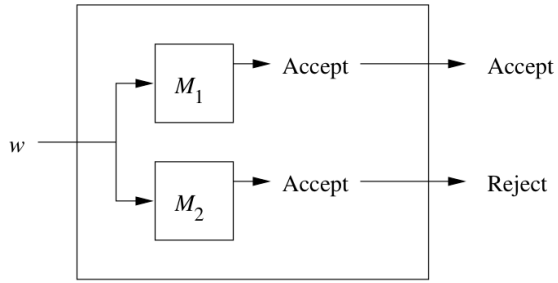
Lemma

Je–li L rekurzivní jazyk, je rekurzivní i $\overline{L}$.



Theorem 10.3 (Postova věta)

Jazyk L je rekurzivní, právě když L i $\overline{L}$ (doplněk) jsou rekurzivně spočetné.



Proof:

$\Rightarrow$ předchozí lemma.

$\Leftarrow$
- Máme TM $L = L(M_1)$ a $\overline{L} = L(M_2)$.
- pro dané slovo w naráz simulujeme M_1 i M_2 (dvě pásky, stav se dvěma komponentami).
- Pokud jeden z M_i přijme, M zastaví a odpoví.
- Jazyky jsou komplementární, jeden z M_i vždy zastaví, L je rekurzivní. □

Nerozhodnutelnost univerzálního jazyka

Theorem 10.4 (Nerozhodnutelnost univerzálního jazyka)

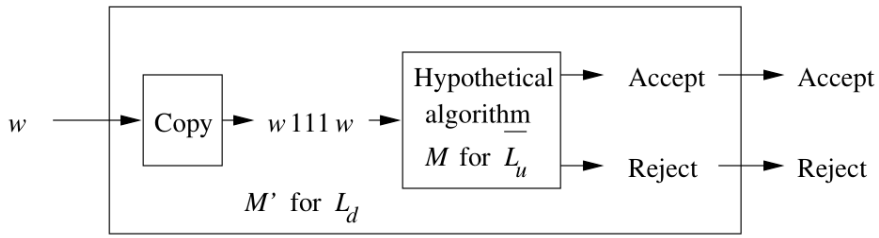
L_u je rekurzivně spočetný, ale není rekurzivní.

Proof.

- Máme TM přijímající L_u, tj. je RE.
- Předpokládejme, že je L_u rekurzivní.
- Pak $\overline{L_u}$ by byl také rekurzivní.
- Pro TM přijímající $\overline{L_u}$ můžeme zkonstruovat TM přijímající L_d (vpravo).
- Protože víme, že L_d není RE, $\overline{L_u}$ není RE a L_u není rekurzivní.

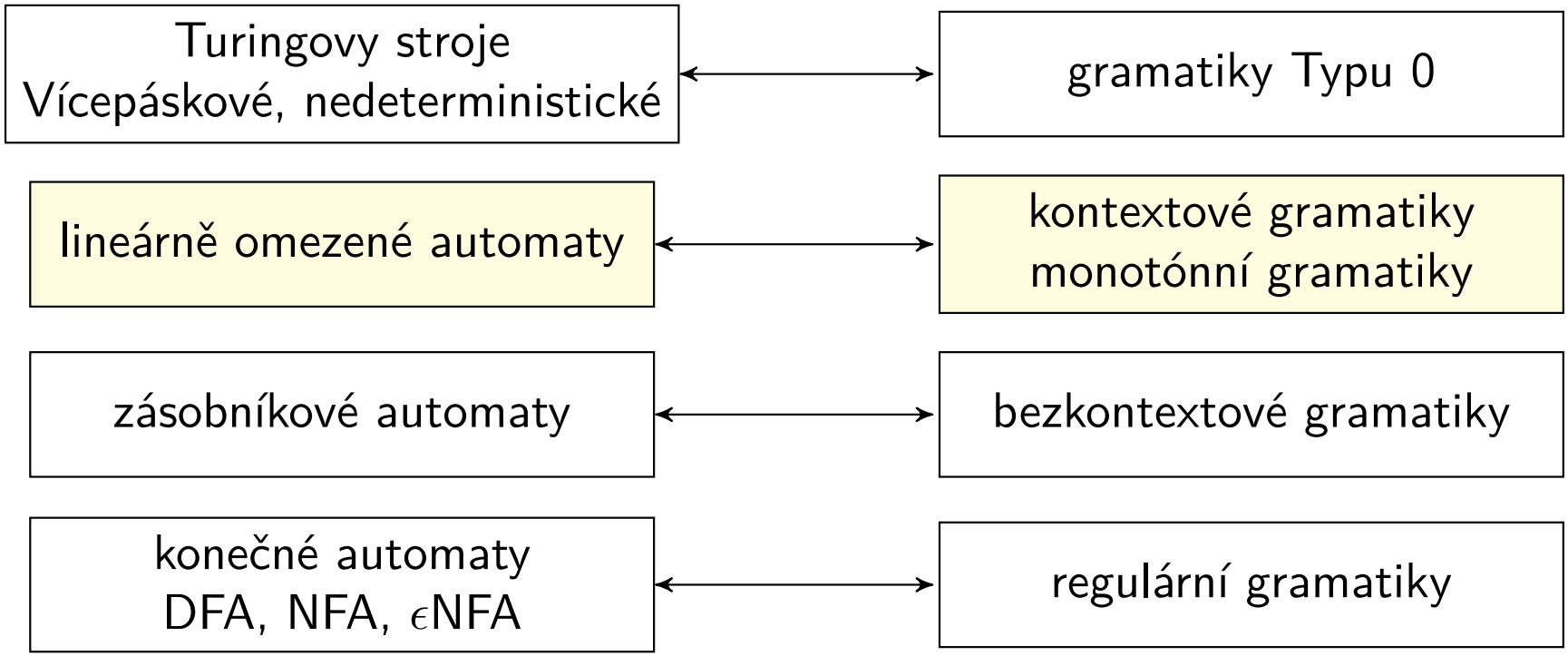
□

Modifikace TM pro $\overline{L_u}$ na TM pro L_d:



- Řetězec w přepiš na $w111w$ (2–páskový, převeď na 1–páskový).
- Simuluj M na novém vstupu. Přijmi iff M přijme.
- Stroj M' přijímá $\overline{L_u}(w, w)$, tj. případy kdy M_i nepřijímá w_i, tj. jazyk L_d.

Kontextové jazyky



- **gramatiky typu 1** (kontextové jazyky $\mathcal{L}_1$)
  - pouze pravidla ve tvaru $\alpha A \beta \to \alpha \omega \beta$
    $A \in V, \alpha, \beta \in (V \cup T)^*, \omega \in (V \cup T)^+$
  - jedinou výjimkou je pravidlo $S \to \epsilon$, potom se ale S nevyskytuje na pravé straně žádného pravidla

Chomského hierarchie

- **gramatiky typu 0** (rekurzivně spočetné jazyky $\mathcal{L}_0$)

  pravidla v obecné formě $\alpha \to \beta$, $\alpha, \beta \in (V \cup T)^*, \alpha$ obsahuje neterminál

gramatiky typu 1 (kontextové jazyky $\mathcal{L}_1$)

- pouze pravidla ve tvaru $\alpha A \beta \to \alpha \omega \beta$

  $A \in V, \alpha, \beta \in (V \cup T)^*, \omega \in (V \cup T)^+$
  - jedinou výjimkou je pravidlo $S \to \epsilon$, potom se ale S nevyskytuje na pravé straně žádného pravidla

- **gramatiky typu 2** (bezkontextové jazyky $\mathcal{L}_2$)

  pouze pravidla ve tvaru $A \to \omega, A \in V, \omega \in (V \cup T)^*$
- **gramatiky typu 3** (regulární/pravé lineární jazyky $\mathcal{L}_3$)

  pouze pravidla ve tvaru $A \to \omega B, A \to \omega, A, B \in V, \omega \in T^*$

Monotónní a kontextové gramatiky

Example 10.1 (kontextový jazyk)

$L = \{a^n b^n c^n | n \geq 1\}$ je kontextový jazyk, není bezkontextový.

$S \to aSBC | abC$

$CB \to BC$ není kontextové, nutno rozepsat!

Monotónní gramatika:

$bB \to bb$

$bC \to bc$

$cC \to cc$

- Je snažší napsat monotónní gramatiku

  a upravit ji na kontextovou.
- Vygeneruji stejný počet a, B, C.
- Přeuspořádám - proto potřebuji B, C neterminály.
- Dovolím přepsat jen v abecedním pořadí
  - Začínám s malými a s jedním b za nimi.
  - B se může přepsat, když je na řadě - vlevo od něj b.

  $\Rightarrow$ Především nedovolím přepsat B, pokud je před ním c.

Separované gramatiky ×

Definition 10.5 (Separovaná gramatika)

Gramatika je **separovaná**, pokud obsahuje pouze pravidla tvaru $\alpha \to \beta$, kde:

- buď $\alpha, \beta \in V^+$ (neprázdné posloupnosti neterminálů)
- nebo $\alpha \in V$ a $\beta \in T \cup \{\epsilon\}$.

Lemma

Ke každé gramatice G lze sestrojit ekvivalentní separovanou gramatiku G'.

Proof:

- Nechť $G = (V, T, P, S)$
- pro každý terminál $a \in T$ zavedeme nový neterminál A'.
- v pravidlech z P
  - nahradíme terminály odpovídajícími neterminály
  - přidáme pravidla $A' \to a$
- Výsledná gramatika je separovaná a zřejmě $L(G) = L(G')$. □

Od monotonie ke kontextovosti

Definition 10.6 (monotónní (nevypouštějící) gramatika)

Gramatika je **monotónní (nevypouštějící)**, jestliže pro každé pravidlo $(\alpha \to \beta) \in P$ platí $|\alpha| \leq |\beta|$. Monotónní gramatiky slovo v průběhu generování nezkracují.

Lemma

Ke každé monotónní gramatice lze nalézt ekvivalentní gramatiku kontextovou.

Proof:

- nejprve převedeme gramatiku na separovanou
  - tím se monotonie neporuší (a pravidla $A' \to a$ jsou kontextová)
- zbývající pravidla $A_1 \dots A_m \to B_1 \dots B_n$, $m \leq n$ převedeme na pravidla s novými neterminály C

$$
\begin{array}{llll}
A_1 A_2 \dots A_m & \to C_1 A_2 \dots A_m & C_1 C_2 \dots C_m & \to B_1 C_2 \dots C_m \\
C_1 A_2 \dots A_m & \to C_1 C_2 \dots A_m & B_1 C_2 \dots C_m & \to B_1 B_2 \dots C_m \\
\vdots & \vdots & \vdots & \vdots \\
C_1 \dots C_{m-1} A_m & \to C_1 \dots C_{m-1} C_m & B_1 \dots B_{m-1} C_m & \to B_1 \dots B_{m-1} B_m \dots B_n
\end{array}
$$

Příklad kontextového jazyka

Example 10.2

Jazyk $L = \{a^i b^j c^k | 1 \leq i \leq j \leq k\}$ je kontextový jazyk, není bezkontextový.

Proof: (na jedničku povinné)

$S \to aSBC \mid aBC$	generování symbolů a
$B \to BBC$	množení symbolů B
$C \to CC$	množení symbolů C
$CB \to BC$	uspořádání symbolů B a C
$aB \to ab$	začátek přepisu B na b
$bB \to bb$	pokračování přepisu B na b
$bC \to bc$	začátek přepisu C na c
$cC \to cc$	pokračování přepisu C na c

$CB \to BC$ není kontextové pravidlo, nahradíme ho

$CB \to XB$, $XB \to XY$, $XY \to BY$, $BY \to BC$

□

Důležitý je přepis, který chybí: $cB \not\to cb$.

Lineárně omezené automaty

- Ještě potřebujeme ekvivalent pro kontextové gramatiky
- kontextovou gramatiku dostaneme z libovolné monotónní gramatiky

Definition 10.7 (lineárně omezený automat (LBA))

Lineárně omezený automat LBA je **nedeterministický** TM, kde na pásce je označen levý a pravý konec $\underline{l}, \underline{r}$. Tyto symboly nelze při výpočtu přepsat a nesmí se jít nalevo od $\underline{l}$ a napravo od $\underline{r}$.
Slovo w **je přijímáno lineárně omezeným automatem**, pokud existuje přijímající výpočet $q_0\underline{l}w\underline{r} \vdash^* \underline{l}\alpha p\beta\underline{r}$, $p \in F$.

- Prostor výpočtu je definován vstupním slovem a automat při jeho přijímání nesmí překročit jeho délku
- u monotónních (kontextových) derivací to není problém – žádné slovo v derivaci není delší než vstupní slovo.

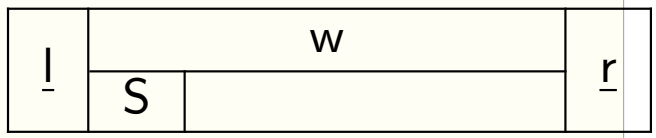
Od kontextových jazyků k LBA×

Theorem 10.5

Každý kontextový jazyk lze přijímat pomocí LBA.

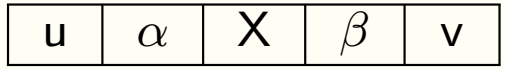
Proof: z kontextové gramatiky k LBA

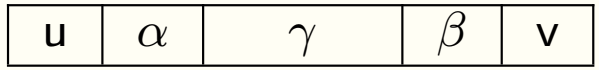
- prázdné slovo vyřešíme zvlášť
- derivaci gramatiky budeme simulovat pomocí LBA
- použijeme pásku se dvěma stopami
- slovo w dáme nahoru, na začátek dolní stopy S



Aplikace pravidla

$\alpha X \beta \to \alpha\gamma\beta$





- přepisujeme slovo ve druhé stopě podle pravidel G
  - nedeterministicky vybereme část k přepsání
  - provedeme přepsání dle pravidla (pravá část se odsune)
- pokud jsou ve druhé stopě samé terminály, porovnáme ji s první stopou
  - slovo přijmeme nebo zamítneme.

□

Od LBA ke kontextovým jazykům×

Theorem 10.6

LBA přijímají pouze kontextové jazyky.

‘

Proof: z LBA ke kontextovým gramatikám

- potřebujeme převést LBA na monotónní gramatiku
  - tj. gramatika nesmí generovat nic navíc
- výpočet ukryjeme do ’dvoustopých’ neterminálů
- generuj slovo ve tvaru $(a_0, [q_0, \underline{l}, a_0]), (a_1, a_1), \ldots, (a_n, [a_n, \underline{r}])$

w		
$q_0, \underline{l}, a_0$		$a_n, \underline{r}$

- simuluj práci LBA ve ’druhé’ stopě (stejně jako u TM)
  - pro $\delta(p, x) = (q, x', R)$: $Px \to x'Q$
  - pro $\delta(p, x) = (q, x', L)$: $yPx \to Qyx'$
- pokud je stav koncový, smaž ’druhou’ stopu.
- Speciálně je třeba ošetřit přijímání prázdného slova
  - pokud LBA přijímá ϵ, přidáme speciální startovací pravidlo.

Mějme lineáně omezený automat pro jazyk $L = \{a^{2^n} | n \geq 1\}$, LBA $M = (\{q_0, q_1, q_2, q_F\}, \{a\}, \{a, \underline{l}, \underline{r}\}, \delta, q_0, B, \{q_F\})$ s δ v tabulce:

δ	komentář
$\delta(q_0, \underline{l}) = (q_0, \underline{l}, R)$	přeskočím prázdnou zarážku
$\delta(q_0, a) = (q_1, a, R)$	zvětší čítač ($2k+1$ symbolů)
$\delta(q_2, a) = (q_1, a, R)$	zvětší čítač ($2k+1$ symbolů)
$\delta(q_1, a) = (q_2, a, R)$	nuluje čítač ($2k$ symbolů)
$\delta(q_2, \underline{r}) = (q_F, \underline{r}, L)$	konec výpočtu, přijímám.

Example 10.3 (Gramatika z lineárně omezeného automatu×)

Monotónní gramatika $G = (V, \{a\}, S, P_1 \cup P_2 \cup P_3)$

$$V = \left\{ \begin{array}{c} S, L, C, R, \begin{bmatrix} a \\ a \end{bmatrix}, \begin{bmatrix} a \\ q_0 \underline{l} a \end{bmatrix}, \begin{bmatrix} a \\ \underline{l} q_0 a \end{bmatrix}, \\ \begin{bmatrix} a \\ \underline{l} a \end{bmatrix}, \begin{bmatrix} a \\ q_2 a \end{bmatrix}, \begin{bmatrix} a \\ q_1 a \end{bmatrix}, \begin{bmatrix} a \\ a \underline{r} \end{bmatrix}, \begin{bmatrix} a \\ M_{\leftarrow} \end{bmatrix}, \\ \begin{bmatrix} a \\ q_1 a \underline{r} \end{bmatrix}, \begin{bmatrix} a \\ a q_2 \underline{r} \end{bmatrix}, \begin{bmatrix} a \\ \underline{l} q_0 a \underline{r} \end{bmatrix} \end{array} \right\} \quad P_1 = \left\{ \begin{array}{lcl} S & \to & LR | LCR \\ C & \to & CC | \begin{bmatrix} a \\ a \end{bmatrix} \\ L & \to & \begin{bmatrix} a \\ q_0 \underline{l} a \end{bmatrix} \\ R & \to & \begin{bmatrix} a \\ a \underline{r} \end{bmatrix} \end{array} \right\}$$

Pravidla pro přechodovou funkci

$$P_2 = \left\{ \begin{array}{ccc} \begin{bmatrix} a \\ q_0 \underline{l} a \end{bmatrix} & \to & \begin{bmatrix} a \\ \underline{l} q_0 a \end{bmatrix} \\ \begin{bmatrix} a \\ \underline{l} q_0 a \end{bmatrix} \begin{bmatrix} a \\ a \end{bmatrix} & \to & \begin{bmatrix} a \\ \underline{l} a \end{bmatrix} \begin{bmatrix} a \\ q_1 a \end{bmatrix} \\ \begin{bmatrix} a \\ q_2 a \end{bmatrix} \begin{bmatrix} a \\ a \end{bmatrix} & \to & \begin{bmatrix} a \\ a \end{bmatrix} \begin{bmatrix} a \\ q_1 a \end{bmatrix} \\ \begin{bmatrix} a \\ q_1 a \end{bmatrix} \begin{bmatrix} a \\ a \end{bmatrix} & \to & \begin{bmatrix} a \\ a \end{bmatrix} \begin{bmatrix} a \\ q_2 a \end{bmatrix} \\ \begin{bmatrix} a \\ q_2 a \end{bmatrix} \begin{bmatrix} a \\ a \underline{r} \end{bmatrix} & \to & \begin{bmatrix} a \\ a \end{bmatrix} \begin{bmatrix} a \\ q_1 a \underline{r} \end{bmatrix} \\ \begin{bmatrix} a \\ q_1 a \underline{r} \end{bmatrix} & \to & \begin{bmatrix} a \\ a q_2 \underline{r} \end{bmatrix} \\ \begin{bmatrix} a \\ a q_2 \underline{r} \end{bmatrix} & \to & \begin{bmatrix} a \\ M_{\leftarrow} \end{bmatrix} \end{array} \right\}$$

Pravidla mazání

$$P_3 = \left\{ \begin{array}{ccc} \begin{bmatrix} a \\ a \end{bmatrix} \begin{bmatrix} a \\ M_{\leftarrow} \end{bmatrix} & \to & \begin{bmatrix} a \\ M_{\leftarrow} \end{bmatrix} a \\ \begin{bmatrix} a \\ \underline{l} a \end{bmatrix} \begin{bmatrix} a \\ M_{\leftarrow} \end{bmatrix} & \to & aa \end{array} \right\}$$

Obecněji

- Při více písmenech bychom měli varianty (téměř) všech neterminálů pro každé písmeno.
- V pravidlech P_1 jen totéž písmeno nahoře i dole.
- Pokud skončíme uvnitř slova, musíme mazat do obou stran až k zarážkám.

Theorem 10.7 (Rekurzivně spočetné jsou $\mathcal{L}_0$)

Každý rekurzivně spočetný jazyk je typu 0.

Proof: Od Turingova stroje ke gramatice

pro Turingův stroj T najdeme gramatiku G, $L(T) = L(G)$

- $G = (\{S, C, D, E\} \cup \{\underline{X}\}_{x \in \Sigma \cup \Gamma} \cup \{Q_i\}_{q_i \in Q}, \Sigma, P, S)$, P je:
- gramatika nejdříve vygeneruje pásku stroje a kopii slova

  $wB^n\underline{W}^R Q_0 B^m$, kde B^i představují volný prostor pro výpočet
- potom simuluje výpočet (stavy jsou součástí slova)
- v koncovém stavu smažeme pásku, necháme pouze kopii slova

1)	$S \to DQ_0E$	
	$D \to xD\underline{X} \mid E$	generuje slovo a jeho revizní kopii pro výpočet
	$E \to BE \mid \epsilon$	generuje volný prostor pro výpočet
2)	$\underline{X}P \to Q\underline{X'}$	pro $\delta(p, x) = (q, x', R)$
	$\underline{X}P\underline{Y} \to \underline{X'}YQ$	pro $\delta(p, x) = (q, x', L)$
3)	$P \to C$	pro $p \in F$
	$C\underline{A} \to C, \underline{A}C \to C$	mazání pásky
	$C \to \epsilon$	konec výpočtu

Příklad

Turingův stroj pro jazyk $L = \{a^{2n} | n \geq 0\}$, TM
$M = (\{q_0, q_1, q_2, q_F\}, \{a\}, \{a\}, \delta, q_0, B, \{q_F\})$ s δ v tabulce:

δ	komentář
$\delta(q_0, a) = (q_1, a, R)$	první symbol
$\delta(q_1, a) = (q_0, a, R)$	nuluje čítač ($2k$ symbolů)
$\delta(q_0, B) = (q_F, B, L)$	přijme a zastaví

Example 10.4

Gramatika $G = (\{S, C, D, E, Q_0, Q_1, Q_F, \underline{a}\}, \{a\}, S, P_1 \cup P_2 \cup P_3)$,

Inicializace

$$P_1 = \left\{ \begin{array}{lcl} S & \to & DQ_0E \\ D & \to & aD\underline{a} | E \\ E & \to & BE | \epsilon \end{array} \right\}$$

Přechodová funkce

$$P_2 = \left\{ \begin{array}{lcl} \underline{a}Q_0 & \to & Q_1\underline{a} \\ \underline{a}Q_1 & \to & Q_0\underline{a} \\ BQ_0\underline{a} & \to & B\underline{a}Q_F \end{array} \right\}$$

Mazání

$$P_3 = \left\{ \begin{array}{lcl} Q_F & \to & C \\ C\underline{a} & \to & C \\ \underline{a}C & \to & C \\ BC & \to & C \\ C & \to & \epsilon \end{array} \right\}.$$

Konkrétně pro $aa \in L(M)$ vygeneruji $aaB\underline{aa}Q_0$, mezivýsledek $aaB\underline{a}Q_F\underline{a}$ a výsledek aa.

Od Turingova stroje ke gramatice

Ještě $L(T) = L(G)$?

- $w \in L(T)$
  - existuje konečný výpočet stroje T (konečný prostor)
  - gramatika vygeneruje dostatečně velký prostor pro výpočet
  - simuluje výpočet a smaže dvojníky.
- $w \in L(G)$
  - pravidla v derivaci nemusí být v pořadí, jakém chceme
  - derivaci můžeme přeuspořádat tak, že pořadí je 1),2),3).
  - podtržené symboly smazány, tj. vygenerován koncový stav.

Example 10.5

Turingův stroj $M = (\{q_0, q_1, q_F\}, \{a\}, \{a, B\}, \delta, q_0, B, \{q_F\})$

$\delta(q_0, B) = (q_F, B, R)$

$\delta(q_0, a) = (q_1, a, R)$

$\delta(q_1, a) = (q_0, a, R)$

Gramatika po zjednodušení

$G = (\{S, C, D, E, \underline{a}, Q_0, Q_1\}, \{a\}, P, S)$

$S \to DQ_0$

$D \to aD\underline{a} | B$

$BQ_0 \to C$

$\underline{a}Q_0 \to Q_1\underline{a}$

$\underline{a}Q_1 \to Q_0\underline{a}$

$C\underline{a} \to C$

$C \to \epsilon$

Od gramatik k Turingově stroji

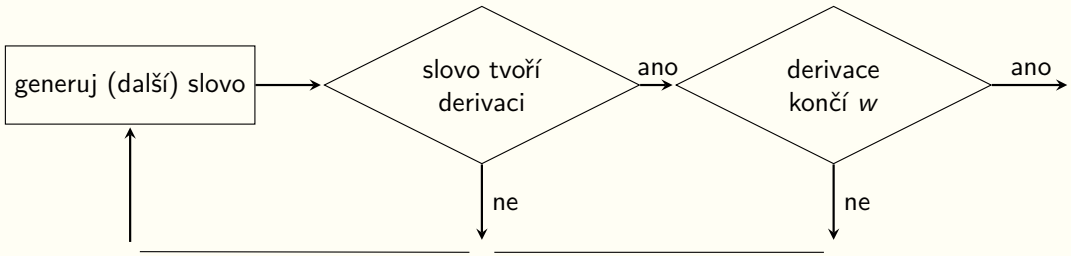
Theorem 10.8

Každý jazyk typu 0 je rekurzivně spočetný.

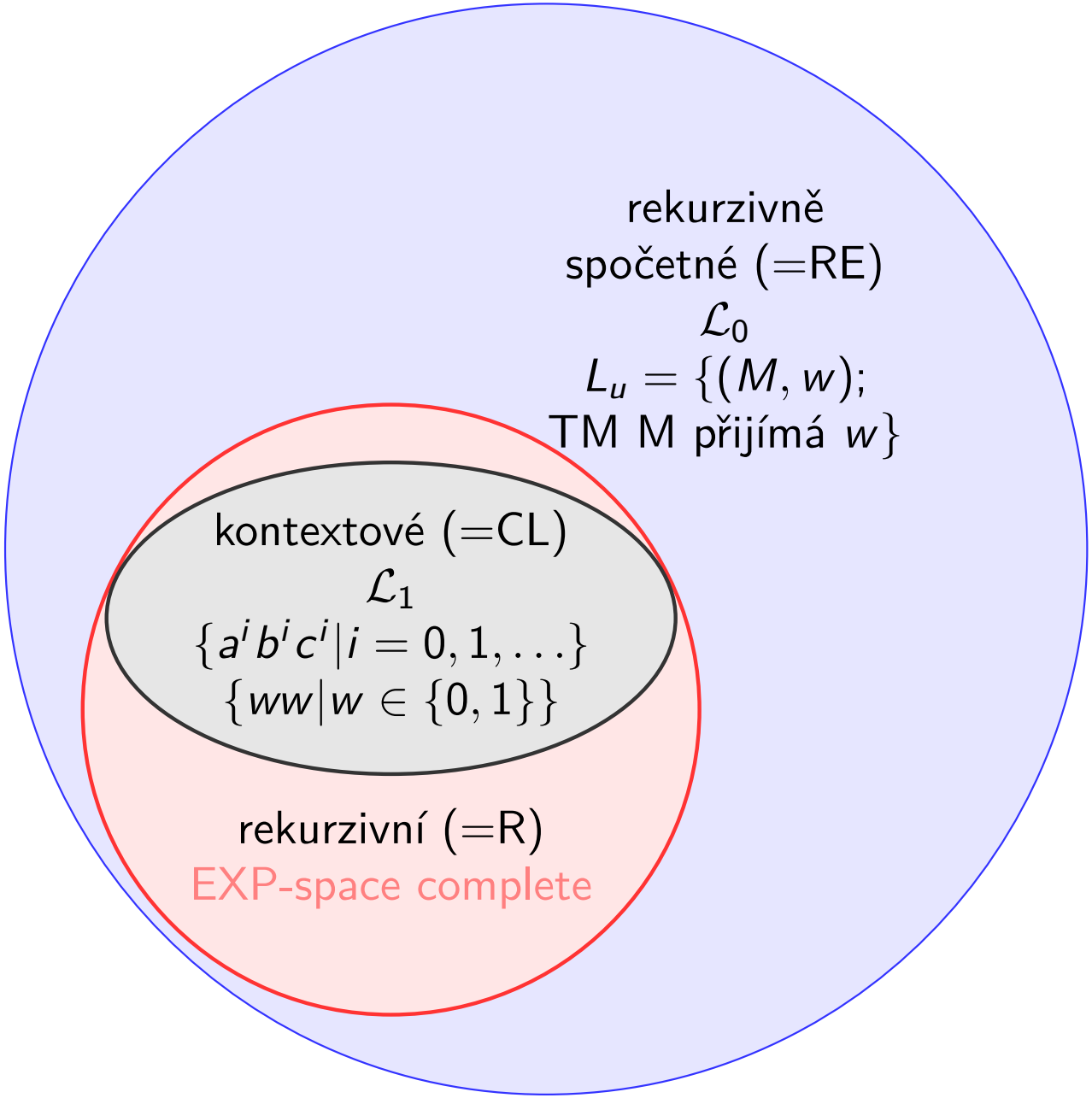
Proof:

idea: TM postupně generuje všechny derivace

- derivaci $S \Rightarrow \omega_1 \Rightarrow \ldots \Rightarrow \omega_n = w$ kódujeme jako slovo $\#S\#\omega_1\# \ldots \#w\#$
- umíme udělat TM, který přijímá slova $\#\alpha\#\beta\#$, kde $\alpha \Rightarrow \beta$
- umíme udělat TM, který přijímá slova $\#\omega_1\# \ldots \#\omega_k\#$, kde $\omega_1 \Rightarrow^* \omega_k$
- umíme udělat TM postupně generující všechna slova.



Hierarchie jazyků (kontextové a výš)



$L_d = \{w;$ TM s kódem w nepřijímá vstup $w\}$