

TIN: 3. demonstrační cvičení

$\mathcal{L}_2$ jazyky a rozhodnutelnost

Tomáš Kocourek

Brno University of Technology, Faculty of Information Technology
Božetěchova 1/2, 612 00 Brno – Královo Pole
xkocou10@fit.vutbr.cz



October 15, 2025

Bezkontextové jazyky

Konstrukce RZA nad abecedou $\Sigma = \{a, b, c\}$

Sestrojte **RZA** přijímající jazyk $co - \{a^n b^n c^n \mid n \geq 1\}$

Konstrukce RZA nad abecedou $\Sigma = \{a, b, c\}$

Sestrojte **RZA** přijímající jazyk $co - \{a^n b^n c^n \mid n \geq 1\}$

ZA

- $\delta : Q \times (\Sigma \cup \{\epsilon\}) \times \Gamma \rightarrow 2^{Q \times \Gamma^*}$

Konstrukce RZA nad abecedou $\Sigma = \{a, b, c\}$

Sestrojte **RZA** přijímající jazyk $co - \{a^n b^n c^n \mid n \geq 1\}$

ZA

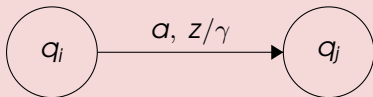
- $\delta : Q \times (\Sigma \cup \{\epsilon\}) \times \Gamma \rightarrow 2^{Q \times \Gamma^*}$
- Odstraňujeme **právě jeden** symbol z vrcholu zásobníku

Konstrukce RZA nad abecedou $\Sigma = \{a, b, c\}$

Sestrojte **RZA** přijímající jazyk $co - \{a^n b^n c^n \mid n \geq 1\}$

ZA

- $\delta : Q \times (\Sigma \cup \{\epsilon\}) \times \Gamma \rightarrow 2^{Q \times \Gamma^*}$
- Odstraňujeme **právě jeden** symbol z vrcholu zásobníku

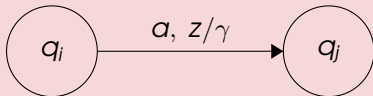


Konstrukce RZA nad abecedou $\Sigma = \{a, b, c\}$

Sestrojte **RZA** přijímající jazyk $co - \{a^n b^n c^n \mid n \geq 1\}$

ZA

- $\delta : Q \times (\Sigma \cup \{\epsilon\}) \times \Gamma \rightarrow 2^{Q \times \Gamma^*}$
- Odstraňujeme **právě jeden** symbol z vrcholu zásobníku



RZA

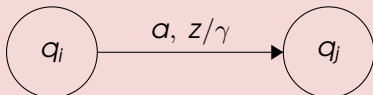
- $\delta : Q \times (\Sigma \cup \{\epsilon\}) \times \Gamma^* \rightarrow 2^{Q \times \Gamma^*}$

Konstrukce RZA nad abecedou $\Sigma = \{a, b, c\}$

Sestrojte **RZA** přijímající jazyk $co - \{a^n b^n c^n \mid n \geq 1\}$

ZA

- $\delta : Q \times (\Sigma \cup \{\epsilon\}) \times \Gamma \rightarrow 2^{Q \times \Gamma^*}$
- Odstraňujeme **právě jeden** symbol z vrcholu zásobníku



RZA

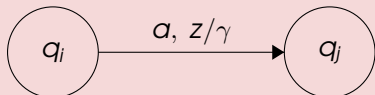
- $\delta : Q \times (\Sigma \cup \{\epsilon\}) \times \Gamma^* \rightarrow 2^{Q \times \Gamma^*}$
- Odstraňujeme **libovolný počet** symbolů z vrcholu zásobníku

Konstrukce RZA nad abecedou $\Sigma = \{a, b, c\}$

Sestrojte **RZA** přijímající jazyk $co - \{a^n b^n c^n \mid n \geq 1\}$

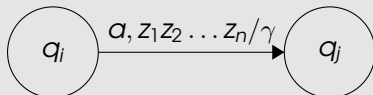
ZA

- $\delta : Q \times (\Sigma \cup \{\epsilon\}) \times \Gamma \rightarrow 2^{Q \times \Gamma^*}$
- Odstraňujeme **právě jeden** symbol z vrcholu zásobníku



RZA

- $\delta : Q \times (\Sigma \cup \{\epsilon\}) \times \Gamma^* \rightarrow 2^{Q \times \Gamma^*}$
- Odstraňujeme **libovolný počet** symbolů z vrcholu zásobníku



Konstrukce RZA nad abecedou $\Sigma = \{a, b, c\}$

Sestrojte **RZA** přijímající jazyk $co - \{a^n b^n c^n \mid n \geq 1\}$

- Pracujeme s abecedou $\Sigma = \{a, b, c\}$
- $co - \{a^n b^n c^n \mid n \geq 1\} =$

Konstrukce RZA nad abecedou $\Sigma = \{a, b, c\}$

Sestrojte **RZA** přijímající jazyk $co - \{a^n b^n c^n \mid n \geq 1\}$

- Pracujeme s abecedou $\Sigma = \{a, b, c\}$
- $co - \{a^n b^n c^n \mid n \geq 1\} =$
 - ① $\{xyz \in \{a, b, c\}^* \mid y \in \{ac, ba, ca, cb\}\} \cup$

Konstrukce RZA nad abecedou $\Sigma = \{a, b, c\}$

Sestrojte **RZA** přijímající jazyk $co - \{a^n b^n c^n \mid n \geq 1\}$

- Pracujeme s abecedou $\Sigma = \{a, b, c\}$
- $co - \{a^n b^n c^n \mid n \geq 1\} =$
 - ① $\{xyz \in \{a, b, c\}^* \mid y \in \{ac, ba, ca, cb\}\} \cup$
 - ② $\{w \in \{a, b, c\}^* \mid \#_a(w) \neq \#_b(w)\} \cup$

Konstrukce RZA nad abecedou $\Sigma = \{a, b, c\}$

Sestrojte **RZA** přijímající jazyk $co - \{a^n b^n c^n \mid n \geq 1\}$

- Pracujeme s abecedou $\Sigma = \{a, b, c\}$
- $co - \{a^n b^n c^n \mid n \geq 1\} =$
 - ① $\{xyz \in \{a, b, c\}^* \mid y \in \{ac, ba, ca, cb\}\} \cup$
 - ② $\{w \in \{a, b, c\}^* \mid \#_a(w) \neq \#_b(w)\} \cup$
 - ③ $\{w \in \{a, b, c\}^* \mid \#_a(w) \neq \#_c(w)\} \cup$

Konstrukce RZA nad abecedou $\Sigma = \{a, b, c\}$

Sestrojte **RZA** přijímající jazyk $co - \{a^n b^n c^n \mid n \geq 1\}$

- Pracujeme s abecedou $\Sigma = \{a, b, c\}$
- $co - \{a^n b^n c^n \mid n \geq 1\} =$
 - 1 $\{xyz \in \{a, b, c\}^* \mid y \in \{ac, ba, ca, cb\}\} \cup$
 - 2 $\{w \in \{a, b, c\}^* \mid \#_a(w) \neq \#_b(w)\} \cup$
 - 3 $\{w \in \{a, b, c\}^* \mid \#_a(w) \neq \#_c(w)\} \cup$
 - 4 $\{w \in \{a, b, c\}^* \mid \#_b(w) \neq \#_c(w)\} \cup$

Konstrukce RZA nad abecedou $\Sigma = \{a, b, c\}$

Sestrojte **RZA** přijímající jazyk $co - \{a^n b^n c^n \mid n \geq 1\}$

- Pracujeme s abecedou $\Sigma = \{a, b, c\}$
- $co - \{a^n b^n c^n \mid n \geq 1\} =$
 - 1 $\{xyz \in \{a, b, c\}^* \mid y \in \{ac, ba, ca, cb\}\} \cup$
 - 2 $\{w \in \{a, b, c\}^* \mid \#_a(w) \neq \#_b(w)\} \cup$
 - 3 $\{w \in \{a, b, c\}^* \mid \#_a(w) \neq \#_c(w)\} \cup$
 - 4 $\{w \in \{a, b, c\}^* \mid \#_b(w) \neq \#_c(w)\} \cup$
 - 5 $\{\epsilon\}$

Konstrukce RZA nad abecedou $\Sigma = \{a, b, c\}$

Sestrojte **RZA** přijímající jazyk $co - \{a^n b^n c^n \mid n \geq 1\}$

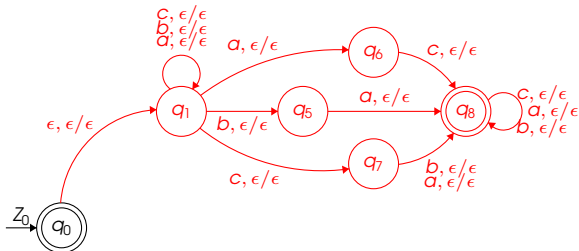
Konstrukce RZA nad abecedou $\Sigma = \{a, b, c\}$

Sestrojte **RZA** přijímající jazyk $co - \{a^n b^n c^n \mid n \geq 1\}$



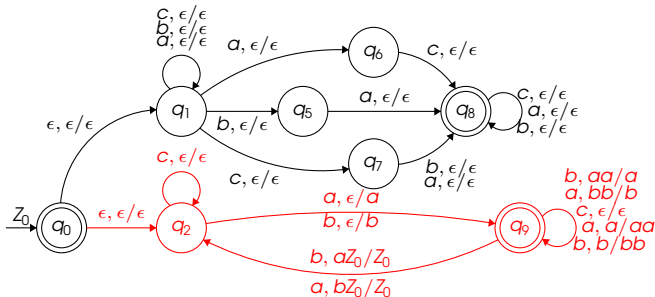
Konstrukce RZA nad abecedou $\Sigma = \{a, b, c\}$

Sestrojte **RZA** přijímající jazyk $co - \{a^n b^n c^n \mid n \geq 1\}$



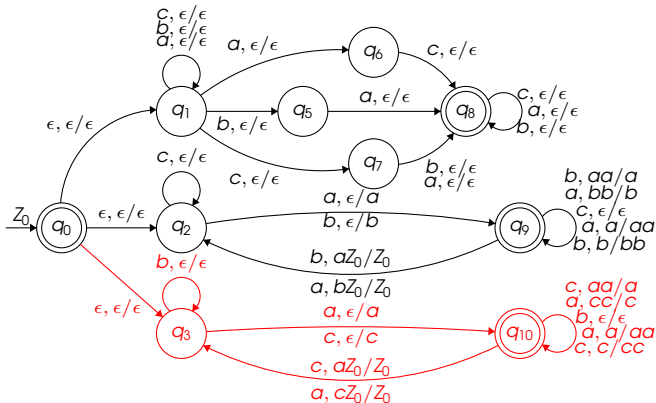
Konstrukce RZA nad abecedou $\Sigma = \{a, b, c\}$

Sestrojte **RZA** přijímající jazyk $co - \{a^n b^n c^n \mid n \geq 1\}$

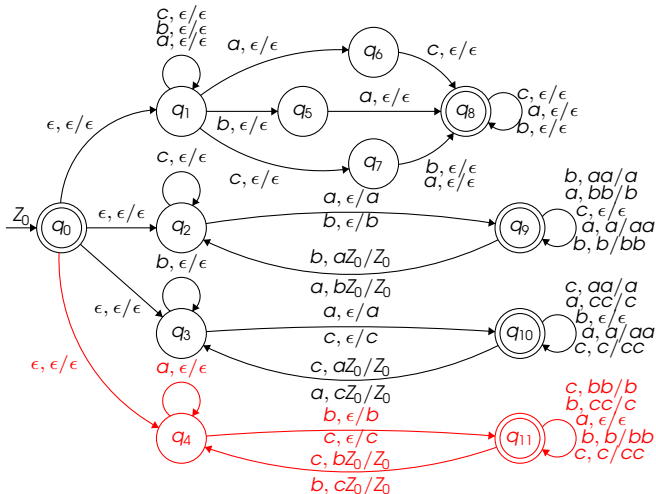


Konstrukce RZA nad abecedou $\Sigma = \{a, b, c\}$

Sestrojte **RZA** přijímající jazyk $co - \{a^n b^n c^n \mid n \geq 1\}$

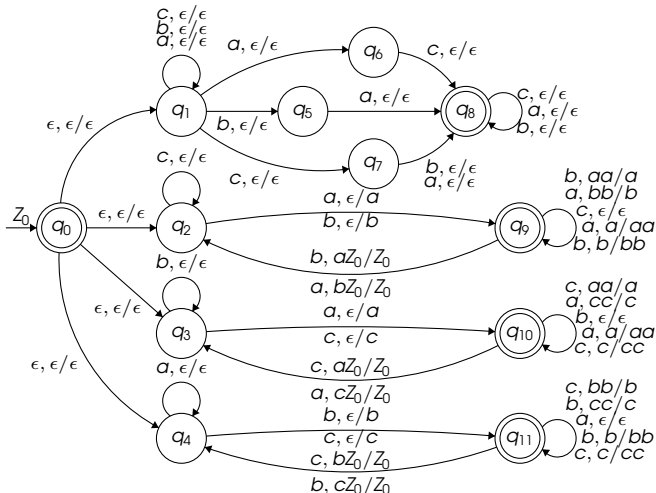


Sestrojte **RZA** přijímající jazyk $co - \{a^n b^n c^n \mid n \geq 1\}$



Konstrukce RZA nad abecedou $\Sigma = \{a, b, c\}$

Sestrojte **RZA** přijímající jazyk $co - \{a^n b^n c^n \mid n \geq 1\}$



Konstrukce RZA nad abecedou $\Sigma = \{a, b, c\}$

Sestrojte **RZA** přijímající jazyk $co - \{a^n b^n c^n \mid n \geq 1\}$

- Jazyk $L = co - \{a^n b^n c^n \mid n \geq 1\}$ lze přijímat RZA

Konstrukce RZA nad abecedou $\Sigma = \{a, b, c\}$

Sestrojte **RZA** přijímající jazyk $co - \{a^n b^n c^n \mid n \geq 1\}$

- Jazyk $L = co - \{a^n b^n c^n \mid n \geq 1\}$ lze přijímat RZA
- Důsledky:

Konstrukce RZA nad abecedou $\Sigma = \{a, b, c\}$

Sestrojte **RZA** přijímající jazyk $co - \{a^n b^n c^n \mid n \geq 1\}$

- Jazyk $L = co - \{a^n b^n c^n \mid n \geq 1\}$ lze přijímat RZA
- Důsledky:
 - ① L lze přijímat i obyčejným ZA

Konstrukce RZA nad abecedou $\Sigma = \{a, b, c\}$

Sestrojte **RZA** přijímající jazyk $co - \{a^n b^n c^n \mid n \geq 1\}$

- Jazyk $L = co - \{a^n b^n c^n \mid n \geq 1\}$ lze přijímat RZA
- Důsledky:
 - 1 L lze přijímat i obyčejným ZA
 - 2 $L \in \mathcal{L}_2$

Konstrukce RZA nad abecedou $\Sigma = \{a, b, c\}$

Sestrojte **RZA** přijímající jazyk $co - \{a^n b^n c^n \mid n \geq 1\}$

- Jazyk $L = co - \{a^n b^n c^n \mid n \geq 1\}$ lze přijímat RZA
- Důsledky:
 - 1 L lze přijímat i obyčejným ZA
 - 2 $L \in \mathcal{L}_2$
 - 3 třída $\mathcal{L}_2$ není uzavřena na doplněk
 - to plyne z faktu, že $co - L \notin \mathcal{L}_2$

Konstrukce RZA nad abecedou $\Sigma = \{a, b, c\}$

Sestrojte **RZA** přijímající jazyk $co - \{a^n b^n c^n \mid n \geq 1\}$

- Jazyk $L = co - \{a^n b^n c^n \mid n \geq 1\}$ lze přijímat RZA
- Důsledky:
 - 1 L lze přijímat i obyčejným ZA
 - 2 $L \in \mathcal{L}_2$
 - 3 třída $\mathcal{L}_2$ není uzavřena na doplněk
 - to plyne z faktu, že $co - L \notin \mathcal{L}_2$
 - 4 $\mathcal{L}_{DZA} \subset \mathcal{L}_2$
 - to plyne z výše uvedeného a z uzavřenosti $\mathcal{L}_{DZA}$ na doplněk

Konstrukce RZA nad abecedou $\Sigma = \{a, b, c\}$

Sestrojte **RZA** přijímající jazyk $co - \{a^n b^n c^n \mid n \geq 1\}$

- Jazyk $L = co - \{a^n b^n c^n \mid n \geq 1\}$ lze přijímat RZA
- Důsledky:
 - 1 L lze přijímat i obyčejným ZA
 - 2 $L \in \mathcal{L}_2$
 - 3 třída $\mathcal{L}_2$ není uzavřena na doplněk
 - to plyne z faktu, že $co - L \notin \mathcal{L}_2$
 - 4 $\mathcal{L}_{DZA} \subset \mathcal{L}_2$
 - to plyne z výše uvedeného a z uzavřenosti $\mathcal{L}_{DZA}$ na doplněk
 - 5 třída $\mathcal{L}_{DZA}$ není uzavřena na sjednocení

Algoritmy nad zásobníkovými automaty

Formálně zapište algoritmus pro sestrojení průniku ZA a KA

Algoritmy nad zásobníkovými automaty

Formálně запиšte algoritmus pro sestrojení průniku ZA a KA

Průnik KA

- třída $\mathcal{L}_3$ je uzavřena na průnik
- $L_1 \cap L_2 = \overline{\overline{L_1} \cup \overline{L_2}}$

Algoritmy nad zásobníkovými automaty

Formálně запиšte algoritmus pro sestrojení průniku **ZA** a **KA**

Průnik KA

- třída $\mathcal{L}_3$ je uzavřena na průnik
- $L_1 \cap L_2 = \overline{\overline{L_1} \cup \overline{L_2}}$

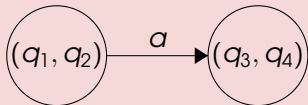


Algoritmy nad zásobníkovými automaty

Formálně запиšte algoritmus pro sestrojení průniku **ZA** a **KA**

Průnik KA

- třída $\mathcal{L}_3$ je uzavřena na průnik
- $L_1 \cap L_2 = \overline{\overline{L_1} \cup \overline{L_2}}$

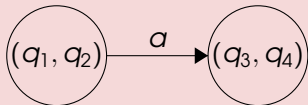


Algoritmy nad zásobníkovými automaty

Formálně zapište algoritmus pro sestrojení průniku **ZA** a **KA**

Průnik KA

- třída $\mathcal{L}_3$ je uzavřena na průnik
- $L_1 \cap L_2 = \overline{\overline{L_1} \cup \overline{L_2}}$



Průnik ZA

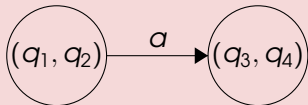
- třída $\mathcal{L}_2$ není uzavřena na průnik

Algoritmy nad zásobníkovými automaty

Formálně zapište algoritmus pro sestrojení průniku **ZA** a **KA**

Průnik KA

- třída $\mathcal{L}_3$ je uzavřena na průnik
- $L_1 \cap L_2 = \overline{\overline{L_1} \cup \overline{L_2}}$



Průnik ZA

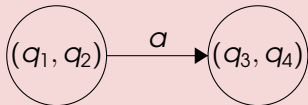
- třída $\mathcal{L}_2$ není uzavřena na průnik
- $L_1 = \{a^n b^n c^m \mid m, n \geq 0\}$

Algoritmy nad zásobníkovými automaty

Formálně zapište algoritmus pro sestrojení průniku **ZA** a **KA**

Průnik KA

- třída $\mathcal{L}_3$ je uzavřena na průnik
- $L_1 \cap L_2 = \overline{\overline{L_1} \cup \overline{L_2}}$



Průnik ZA

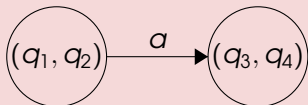
- třída $\mathcal{L}_2$ není uzavřena na průnik
- $L_1 = \{a^n b^n c^m \mid m, n \geq 0\}$
- $L_2 = \{a^m b^n c^n \mid m, n \geq 0\}$

Algoritmy nad zásobníkovými automaty

Formálně zapište algoritmus pro sestrojení průniku **ZA** a **KA**

Průnik KA

- třída $\mathcal{L}_3$ je uzavřena na průnik
- $L_1 \cap L_2 = \overline{\overline{L_1} \cup \overline{L_2}}$



Průnik ZA

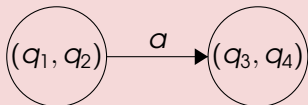
- třída $\mathcal{L}_2$ není uzavřena na průnik
- $L_1 = \{a^n b^n c^m \mid m, n \geq 0\}$
- $L_2 = \{a^m b^n c^n \mid m, n \geq 0\}$
- $L_3 = L_1 \cap L_2$

Algoritmy nad zásobníkovými automaty

Formálně запиšte algoritmus pro sestrojení průniku **ZA** a **KA**

Průnik KA

- třída $\mathcal{L}_3$ je uzavřena na průnik
- $L_1 \cap L_2 = \overline{\overline{L_1} \cup \overline{L_2}}$



Průnik ZA

- třída $\mathcal{L}_2$ není uzavřena na průnik
- $L_1 = \{a^n b^n c^m \mid m, n \geq 0\}$
- $L_2 = \{a^m b^n c^n \mid m, n \geq 0\}$
- $L_3 = L_1 \cap L_2$
- $L_3 = \{a^n b^n c^n \mid n \geq 0\} \notin \mathcal{L}_2$

Algoritmy nad zásobníkovými automaty

Formálně zapište algoritmus pro sestrojení průniku ZA a KA

- Vstup:

Algoritmy nad zásobníkovými automaty

Formálně запиšte algoritmus pro sestrojení průniku ZA a KA

- Vstup:

$$\textcircled{1} \text{ ZA } A_1 = (Q_1, \Sigma_1, \Gamma_1, \delta_1, q_0^1, Z_0^1, F_1)$$

Algoritmy nad zásobníkovými automaty

Formálně запиšte algoritmus pro sestrojení průniku ZA a KA

• Vstup:

- 1 ZA $A_1 = (Q_1, \Sigma_1, \Gamma_1, \delta_1, q_0^1, Z_0^1, F_1)$
- 2 KA $A_2 = (Q_2, \Sigma_2, \delta_2, q_0^2, F_2)$

Algoritmy nad zásobníkovými automaty

Formálně запиšte algoritmus pro sestrojení průniku ZA a KA

• Vstup:

$$\textcircled{1} \text{ ZA } A_1 = (Q_1, \Sigma_1, \Gamma_1, \delta_1, q_0^1, Z_0^1, F_1)$$

$$\textcircled{2} \text{ KA } A_2 = (Q_2, \Sigma_2, \delta_2, q_0^2, F_2)$$

• Výstup:

Algoritmy nad zásobníkovými automaty

Formálně запиšte algoritmus pro sestrojení průniku ZA a KA

• Vstup:

$$\textcircled{1} \text{ ZA } A_1 = (Q_1, \Sigma_1, \Gamma_1, \delta_1, q_0^1, Z_0^1, F_1)$$

$$\textcircled{2} \text{ KA } A_2 = (Q_2, \Sigma_2, \delta_2, q_0^2, F_2)$$

• Výstup:

- ZA $A = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$, kde $L(A) = L(A_1) \cap L(A_2)$

Algoritmy nad zásobníkovými automaty

Formálně запиšte algoritmus pro sestrojení průniku ZA a KA

• Vstup:

$$\textcircled{1} \text{ ZA } A_1 = (Q_1, \Sigma_1, \Gamma_1, \delta_1, q_0^1, Z_0^1, F_1)$$

$$\textcircled{2} \text{ KA } A_2 = (Q_2, \Sigma_2, \delta_2, q_0^2, F_2)$$

• Výstup:

- ZA $A = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$, kde $L(A) = L(A_1) \cap L(A_2)$

• Algoritmus:

Algoritmy nad zásobníkovými automaty

Formálně zapište algoritmus pro sestrojení průniku ZA a KA

• Vstup:

$$\textcircled{1} \text{ ZA } A_1 = (Q_1, \Sigma_1, \Gamma_1, \delta_1, q_0^1, Z_0^1, F_1)$$

$$\textcircled{2} \text{ KA } A_2 = (Q_2, \Sigma_2, \delta_2, q_0^2, F_2)$$

• Výstup:

$$\bullet \text{ ZA } A = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F), \text{ kde } L(A) = L(A_1) \cap L(A_2)$$

• Algoritmus:

$$\textcircled{1} Q = Q_1 \times Q_2$$

Algoritmy nad zásobníkovými automaty

Formálně запиšte algoritmus pro sestrojení průniku ZA a KA

• Vstup:

$$\textcircled{1} \text{ ZA } A_1 = (Q_1, \Sigma_1, \Gamma_1, \delta_1, q_0^1, Z_0^1, F_1)$$

$$\textcircled{2} \text{ KA } A_2 = (Q_2, \Sigma_2, \delta_2, q_0^2, F_2)$$

• Výstup:

- ZA $A = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$, kde $L(A) = L(A_1) \cap L(A_2)$

• Algoritmus:

$$\textcircled{1} Q = Q_1 \times Q_2$$

$$\textcircled{2} \Sigma = \Sigma_1 \cup \Sigma_2$$

- nebo $\Sigma = \Sigma_1 \cap \Sigma_2$ a zvlášť ošetřit případ, kdy $\Sigma_1 \cap \Sigma_2 = \emptyset$

Algoritmy nad zásobníkovými automaty

Formálně запиšte algoritmus pro sestrojení průniku ZA a KA

• Vstup:

$$\textcircled{1} \text{ ZA } A_1 = (Q_1, \Sigma_1, \Gamma_1, \delta_1, q_0^1, Z_0^1, F_1)$$

$$\textcircled{2} \text{ KA } A_2 = (Q_2, \Sigma_2, \delta_2, q_0^2, F_2)$$

• Výstup:

- ZA $A = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$, kde $L(A) = L(A_1) \cap L(A_2)$

• Algoritmus:

$$\textcircled{1} Q = Q_1 \times Q_2$$

$$\textcircled{2} \Sigma = \Sigma_1 \cup \Sigma_2$$

- nebo $\Sigma = \Sigma_1 \cap \Sigma_2$ a zvlášť ošetřit případ, kdy $\Sigma_1 \cap \Sigma_2 = \emptyset$

$$\textcircled{3} \Gamma = \Gamma_1$$

Algoritmy nad zásobníkovými automaty

Formálně запиšte algoritmus pro sestrojení průniku **ZA** a **KA**

• Vstup:

$$① \text{ ZA } A_1 = (Q_1, \Sigma_1, \Gamma_1, \delta_1, q_0^1, Z_0^1, F_1)$$

$$② \text{ KA } A_2 = (Q_2, \Sigma_2, \delta_2, q_0^2, F_2)$$

• Výstup:

$$\bullet \text{ ZA } A = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F), \text{ kde } L(A) = L(A_1) \cap L(A_2)$$

• Algoritmus:

$$① Q = Q_1 \times Q_2$$

$$② \Sigma = \Sigma_1 \cup \Sigma_2$$

• nebo $\Sigma = \Sigma_1 \cap \Sigma_2$ a zvlášť ošetřit případ, kdy $\Sigma_1 \cap \Sigma_2 = \emptyset$

$$③ \Gamma = \Gamma_1$$

$$④ q_0 = (q_0^1, q_0^2)$$

Algoritmy nad zásobníkovými automaty

Formálně zapíšte algoritmus pro sestrojení průniku ZA a KA

• Vstup:

- 1 $ZA\ A_1 = (Q_1, \Sigma_1, \Gamma_1, \delta_1, q_0^1, Z_0^1, F_1)$

- 2 $KA\ A_2 = (Q_2, \Sigma_2, \delta_2, q_0^2, F_2)$

• Výstup:

- $ZA\ A = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$, kde $L(A) = L(A_1) \cap L(A_2)$

• Algoritmus:

- 1 $Q = Q_1 \times Q_2$

- 2 $\Sigma = \Sigma_1 \cup \Sigma_2$

- nebo $\Sigma = \Sigma_1 \cap \Sigma_2$ a zvlášť ošetřit případ, kdy $\Sigma_1 \cap \Sigma_2 = \emptyset$

- 3 $\Gamma = \Gamma_1$

- 4 $q_0 = (q_0^1, q_0^2)$

- 5 $Z_0 = Z_0^1$

Algoritmy nad zásobníkovými automaty

Formálně запиšte algoritmus pro sestrojení průniku ZA a KA

• Vstup:

$$\textcircled{1} \text{ ZA } A_1 = (Q_1, \Sigma_1, \Gamma_1, \delta_1, q_0^1, Z_0^1, F_1)$$

$$\textcircled{2} \text{ KA } A_2 = (Q_2, \Sigma_2, \delta_2, q_0^2, F_2)$$

• Výstup:

$$\bullet \text{ ZA } A = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F), \text{ kde } L(A) = L(A_1) \cap L(A_2)$$

• Algoritmus:

$$\textcircled{1} Q = Q_1 \times Q_2$$

$$\textcircled{2} \Sigma = \Sigma_1 \cup \Sigma_2$$

• nebo $\Sigma = \Sigma_1 \cap \Sigma_2$ a zvlášť ošetřit případ, kdy $\Sigma_1 \cap \Sigma_2 = \emptyset$

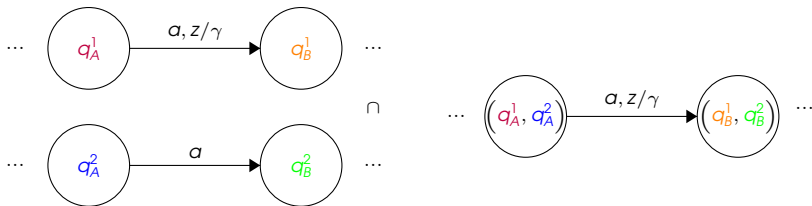
$$\textcircled{3} \Gamma = \Gamma_1$$

$$\textcircled{4} q_0 = (q_0^1, q_0^2)$$

$$\textcircled{5} Z_0 = Z_0^1$$

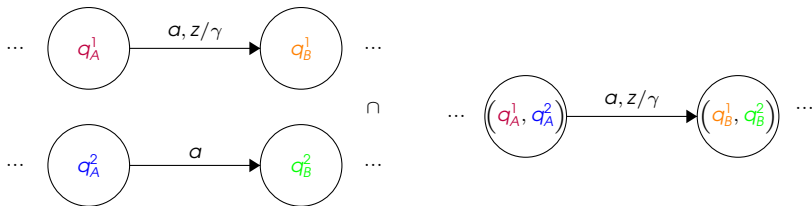
$$\textcircled{6} F = F_1 \times F_2$$

Algoritmy nad zásobníkovými automaty

Formálně запиšte algoritmus pro sestrojení průniku **ZA** a **KA**

Algoritmy nad zásobníkovými automaty

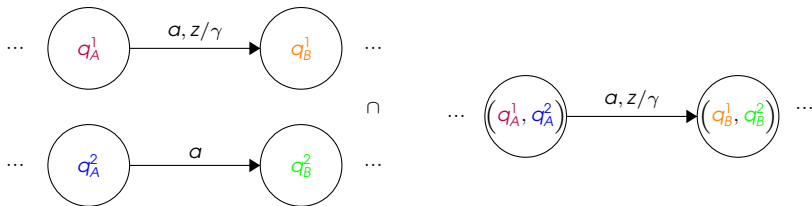
Formálně zapište algoritmus pro sestrojení průniku **ZA** a **KA**



$$7 \quad \forall q_A^1, q_B^1 \in Q_1 : \forall q_A^2, q_B^2 \in Q_2 : \forall a \in \Sigma : \forall z \in \Gamma : \forall \gamma \in \Gamma^* :$$

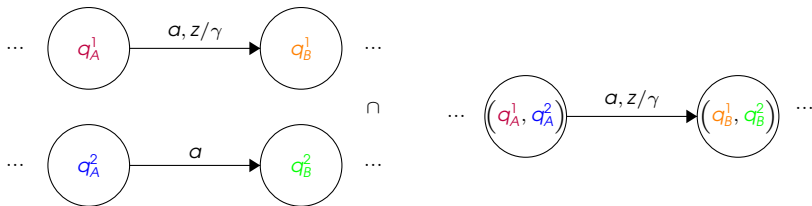
Algoritmy nad zásobníkovými automaty

Formálně zapíšte algoritmus pro sestrojení průniku **ZA** a **KA**



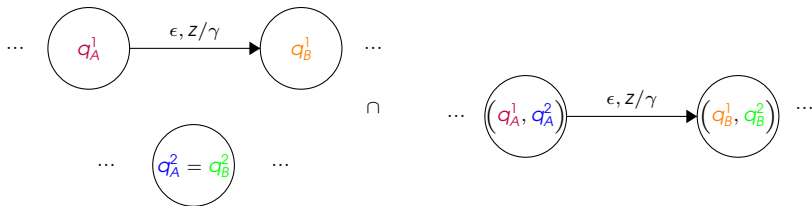
$$7 \quad \forall q_A^1, q_B^1 \in Q_1 : \forall q_A^2, q_B^2 \in Q_2 : \forall a \in \Sigma : \forall z \in \Gamma : \forall \gamma \in \Gamma^* : \\ ((q_B^1, q_B^2), \gamma) \in \delta((q_A^1, q_A^2), a, z) \Leftrightarrow$$

Algoritmy nad zásobníkovými automaty

Formálně запиšte algoritmus pro sestrojení průniku **ZA** a **KA**

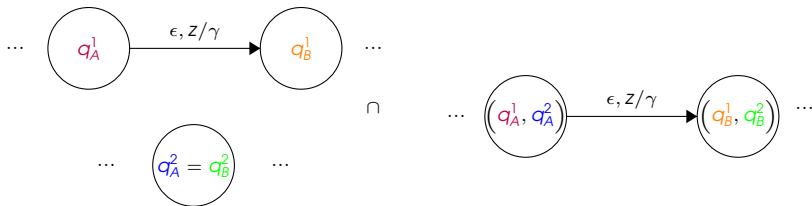
$$\begin{aligned}
 7 \quad & \forall q_A^1, q_B^1 \in Q_1 : \forall q_A^2, q_B^2 \in Q_2 : \forall a \in \Sigma : \forall z \in \Gamma : \forall \gamma \in \Gamma^* : \\
 & ((q_B^1, q_B^2), \gamma) \in \delta((q_A^1, q_A^2), a, z) \Leftrightarrow \\
 & ((q_B^1, \gamma) \in \delta_1(q_A^1, a, z) \wedge q_B^2 \in \delta_2(q_A^2, a))
 \end{aligned}$$

Algoritmy nad zásobníkovými automaty

Formálně zapište algoritmus pro sestrojení průniku **ZA** a **KA**

Algoritmy nad zásobníkovými automaty

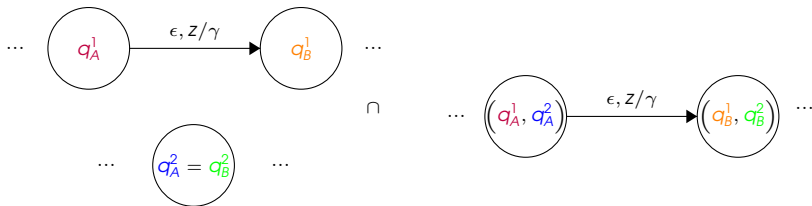
Formálně zapíšte algoritmus pro sestrojení průniku **ZA** a **KA**



$$8 \quad \forall q_A^1, q_B^1 \in Q_1 : \forall q_A^2, q_B^2 \in Q_2 : \forall Z \in \Gamma : \forall \gamma \in \Gamma^* :$$

Algoritmy nad zásobníkovými automaty

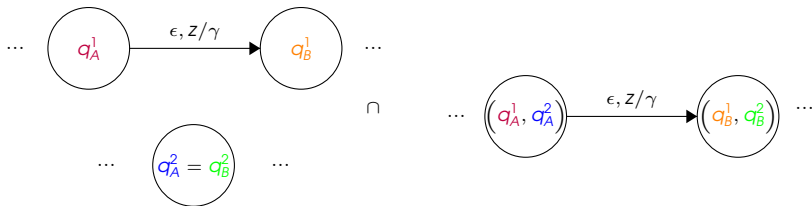
Formálně запиšte algoritmus pro sestrojení průniku **ZA** a **KA**



$$8 \quad \forall q_A^1, q_B^1 \in Q_1 : \forall q_A^2, q_B^2 \in Q_2 : \forall Z \in \Gamma : \forall \gamma \in \Gamma^* : \\ ((q_B^1, q_B^2), \gamma) \in \delta((q_A^1, q_A^2), \epsilon, Z) \Leftrightarrow$$

Algoritmy nad zásobníkovými automaty

Formálně запиšte algoritmus pro sestrojení průniku **ZA** a **KA**



$$\begin{aligned}
 & \textcircled{8} \quad \forall q_A^1, q_B^1 \in Q_1 : \forall q_A^2, q_B^2 \in Q_2 : \forall Z \in \Gamma : \forall \gamma \in \Gamma^* : \\
 & ((q_B^1, q_B^2), \gamma) \in \delta((q_A^1, q_A^2), \epsilon, Z) \Leftrightarrow \\
 & ((q_B^1, \gamma) \in \delta_1(q_A^1, \epsilon, Z) \wedge q_A^2 = q_B^2)
 \end{aligned}$$

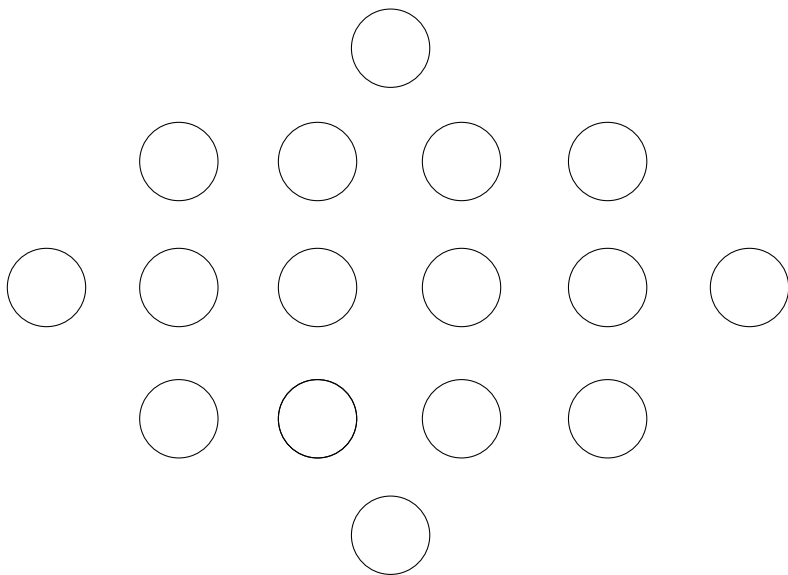
Algoritmy nad zásobníkovými automaty

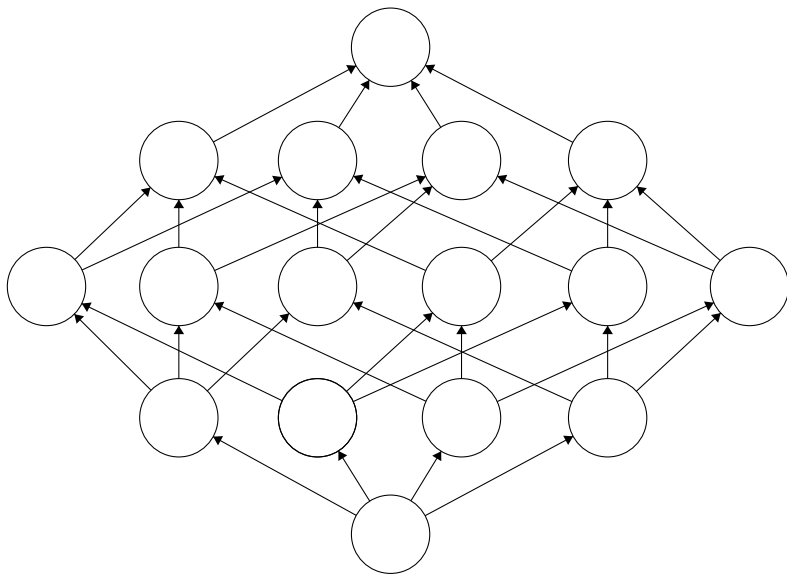
Formálně zapište algoritmus pro sestrojení průniku ZA a KA

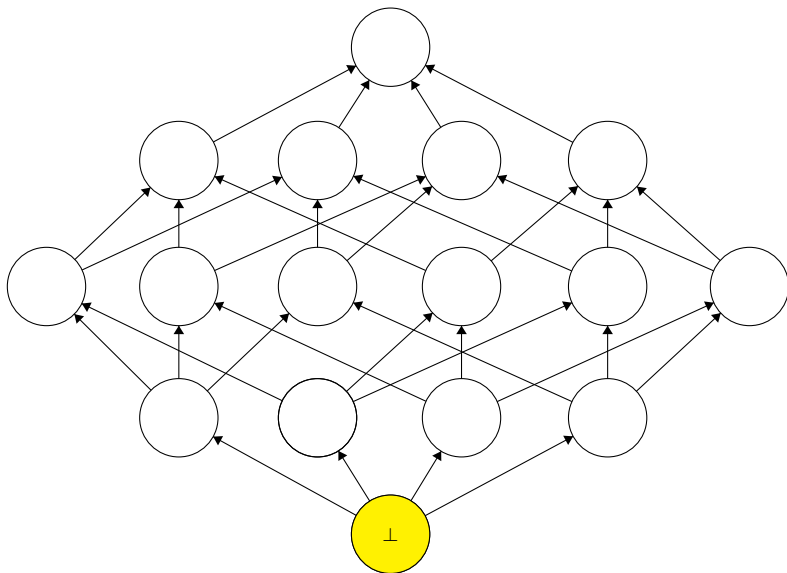
Pozn.: nelze napsat

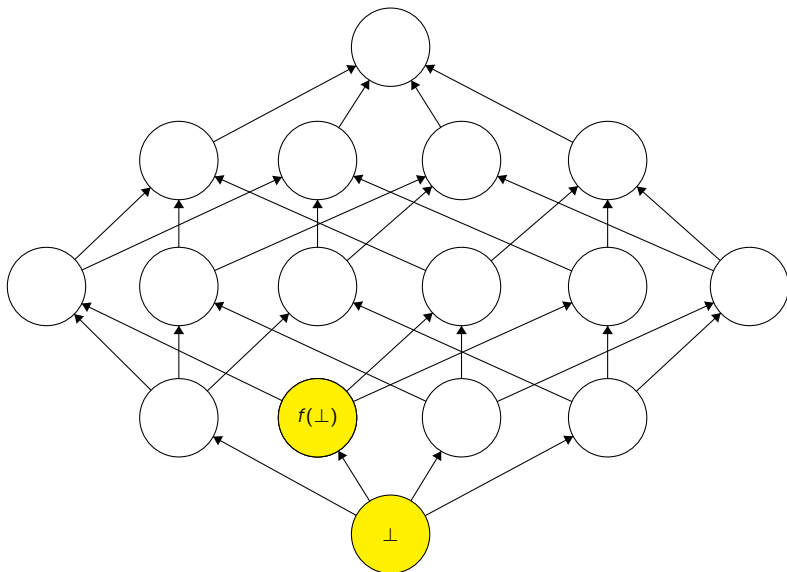
- $\forall q_A^1, q_B^1 \in Q_1 : \forall q \in Q_2 : \forall z \in \Gamma : \forall \gamma \in \Gamma^* :$
 $((q_B^1, q), \gamma) \in \delta((q_A^1, q), \epsilon, z) \Leftrightarrow (q_B^1, \gamma) \in \delta_1(q_A^1, \epsilon, z)$

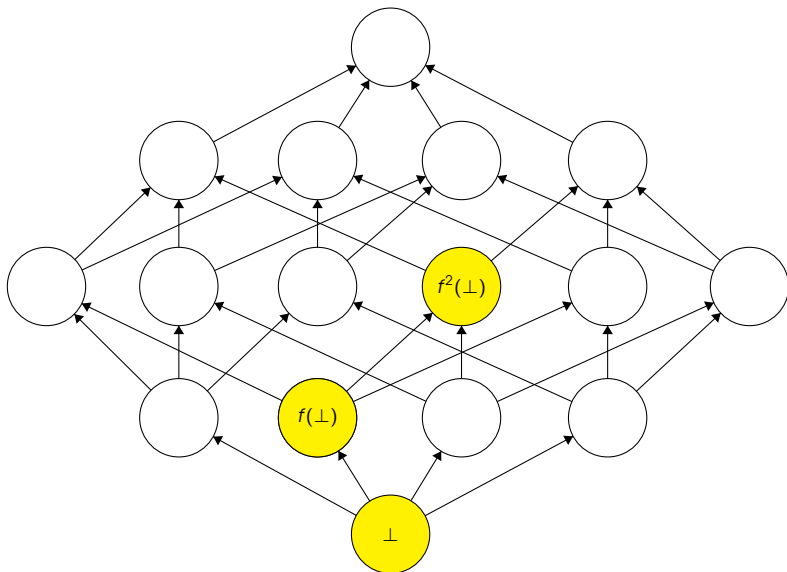
Algoritmy zpřesňující odhad výsledku

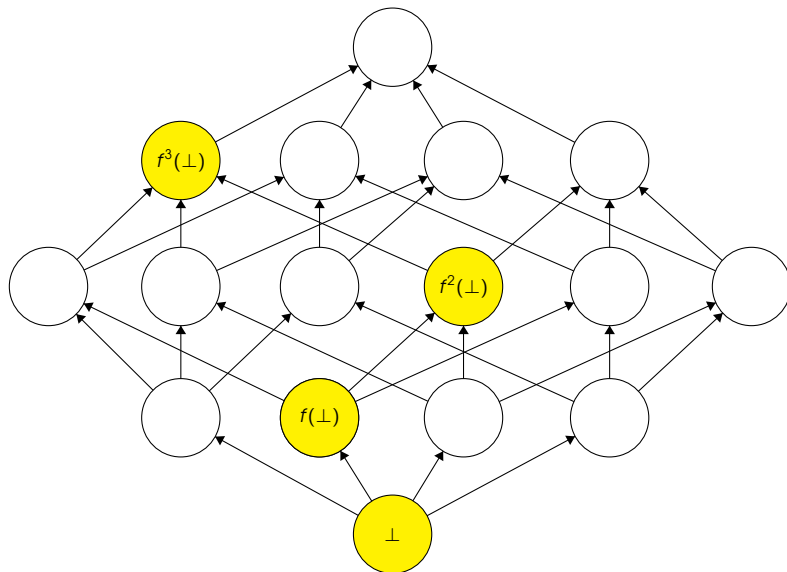


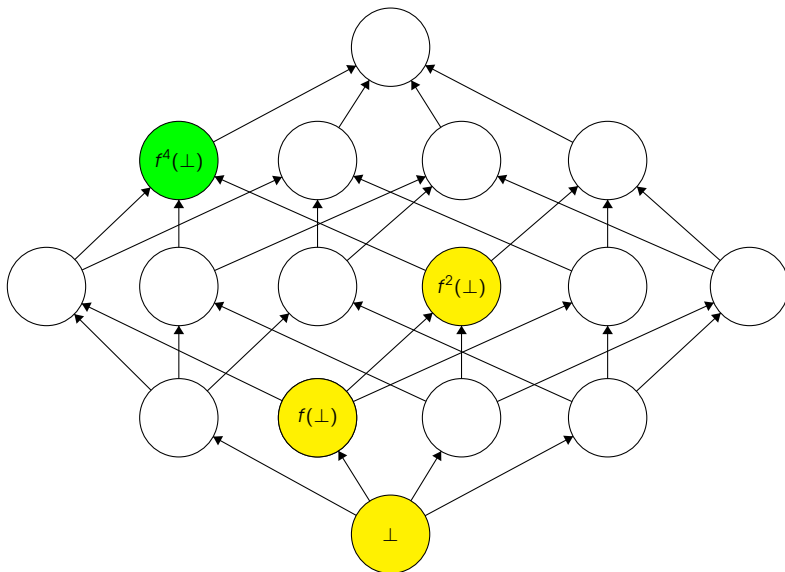












Poset

- dvojice $(D, \sqsubseteq)$
- D je nosná množina
- $\sqsubseteq \subseteq D^2$ je uspořádání na D

Poset

- dvojice $(D, \sqsubseteq)$
- D je nosná množina
- $\sqsubseteq \subseteq D^2$ je uspořádání na D

Pevný bod funkce

- $f : X \rightarrow X$
- $a \in X$ je pevný bod f , pokud $f(a) = a$

Poset

- dvojice $(D, \sqsubseteq)$
- D je nosná množina
- $\sqsubseteq \subseteq D^2$ je uspořádání na D

Pevný bod funkce

- $f : X \rightarrow X$
- $a \in X$ je pevný bod f , pokud $f(a) = a$

Úplný svaz

- šestice $(D, \sqsubseteq, \sqcup, \sqcap, \top, \perp)$
- supremum $\sqcup$, infimum $\sqcap$
- největší prvek $\top$
- nejmenší prvek $\perp$
- existuje $\sqcup, \sqcap$ pro každou podmnožinu nosiče D

Poset

- dvojice $(D, \sqsubseteq)$
- D je nosná množina
- $\sqsubseteq \subseteq D^2$ je uspořádání na D

Pevný bod funkce

- $f : X \rightarrow X$
- $a \in X$ je pevný bod f , pokud $f(a) = a$

Úplný svaz

- šestice $(D, \sqsubseteq, \sqcup, \sqcap, \top, \perp)$
- supremum $\sqcup$, infimum $\sqcap$
- největší prvek $\top$
- nejmenší prvek $\perp$
- existuje $\sqcup, \sqcap$ pro každou podmnožinu nosiče D

Monotónní funkce

- $(D, \sqsubseteq)$ je poset
- $f : D \rightarrow D$ je zobrazení
- mon. rostoucí: $\forall a, b \in D : a \sqsubseteq b \Rightarrow f(a) \sqsubseteq f(b)$
- mon. klasající: $\forall a, b \in D : a \sqsubseteq b \Rightarrow f(b) \sqsubseteq f(a)$

Knaster-Tarskiho teorém

Pevné body monotónního zobrazení $f : D \rightarrow D$ nad úplným svazem $(D, \sqsubseteq, \sqcup, \sqcap, \top, \perp)$ tvoří opět úplný svaz.

(To garantuje existenci nejmenšího/největšího pevného bodu f)

Knaster-Tarskiho teorém

Pevné body monotónního zobrazení $f : D \rightarrow D$ nad úplným svazem $(D, \sqsubseteq, \sqcup, \sqcap, \top, \perp)$ tvoří opět úplný svaz.
(To garantuje existenci nejmenšího/největšího pevného bodu f)

Nejmenší pevný bod

Knaster-Tarskiho teorém

Pevné body monotónního zobrazení $f : D \rightarrow D$ nad úplným svazem $(D, \sqsubseteq, \sqcup, \sqcap, \top, \perp)$ tvoří opět úplný svaz.

(To garantuje existenci nejmenšího/největšího pevného bodu f)

Nejmenší pevný bod

- iniciační hodnota $\perp$

Knaster-Tarskiho teorém

Pevné body monotónního zobrazení $f : D \rightarrow D$ nad úplným svazem $(D, \sqsubseteq, \sqcup, \sqcap, \top, \perp)$ tvoří opět úplný svaz.

(To garantuje existenci nejmenšího/největšího pevného bodu f)

Nejmenší pevný bod

- iniciační hodnota $\perp$
- f je mon. rostoucí

Knaster-Tarskiho teorém

Pevné body monotónního zobrazení $f : D \rightarrow D$ nad úplným svazem $(D, \sqsubseteq, \sqcup, \sqcap, \top, \perp)$ tvoří opět úplný svaz.

(To garantuje existenci nejmenšího/největšího pevného bodu f)

Nejmenší pevný bod

- iniciační hodnota $\perp$
- f je mon. rostoucí
- $\perp \sqsubseteq f(\perp) \Rightarrow f(\perp) \sqsubseteq f(f(\perp))$

Knaster-Tarskiho teorém

Pevné body monotónního zobrazení $f : D \rightarrow D$ nad úplným svazem $(D, \sqsubseteq, \sqcup, \sqcap, \top, \perp)$ tvoří opět úplný svaz.

(To garantuje existenci nejmenšího/největšího pevného bodu f)

Nejmenší pevný bod

- iniciační hodnota $\perp$
- f je mon. rostoucí
- $\perp \sqsubseteq f(\perp) \Rightarrow f(\perp) \sqsubseteq f(f(\perp))$
- postupně počítáme $f(\perp), f(f(\perp)), f(f(f(\perp))), \dots$

Knaster-Tarskiho teorém

Pevné body monotónního zobrazení $f : D \rightarrow D$ nad úplným svazem $(D, \sqsubseteq, \sqcup, \sqcap, \top, \perp)$ tvoří opět úplný svaz.

(To garantuje existenci nejmenšího/největšího pevného bodu f)

Nejmenší pevný bod

- iniciační hodnota $\perp$
- f je mon. rostoucí
- $\perp \sqsubseteq f(\perp) \Rightarrow f(\perp) \sqsubseteq f(f(\perp))$
- postupně počítáme $f(\perp), f(f(\perp)), f(f(f(\perp))), \dots$
- $\exists k \in \mathbb{N} : f^k(\perp) = f^{k+1}(\perp)$

Knaster-Tarskiho teorém

Pevné body monotónního zobrazení $f : D \rightarrow D$ nad úplným svazem $(D, \sqsubseteq, \sqcup, \sqcap, \top, \perp)$ tvoří opět úplný svaz.

(To garantuje existenci nejmenšího/největšího pevného bodu f)

Nejmenší pevný bod

- iniciační hodnota $\perp$
- f je mon. rostoucí
- $\perp \sqsubseteq f(\perp) \Rightarrow f(\perp) \sqsubseteq f(f(\perp))$
- postupně počítáme $f(\perp), f(f(\perp)), f(f(f(\perp))), \dots$
- $\exists k \in \mathbb{N} : f^k(\perp) = f^{k+1}(\perp)$
- $f^k(\perp)$ je nejmenší pevný bod zobrazení f

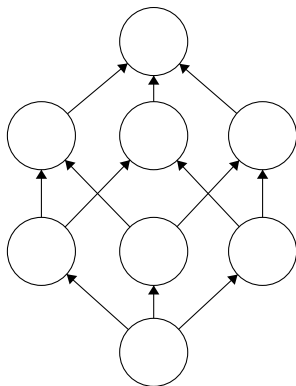
Knaster-Tarskiho teorém

Pevné body monotónního zobrazení $f : D \rightarrow D$ nad úplným svazem $(D, \sqsubseteq, \sqcup, \sqcap, \top, \perp)$ tvoří opět úplný svaz.

(To garantuje existenci nejmenšího/největšího pevného bodu f)

Nejmenší pevný bod

- iniciační hodnota $\perp$
- f je mon. rostoucí
- $\perp \sqsubseteq f(\perp) \Rightarrow f(\perp) \sqsubseteq f(f(\perp))$
- postupně počítáme $f(\perp), f(f(\perp)), f(f(f(\perp))), \dots$
- $\exists k \in \mathbb{N} : f^k(\perp) = f^{k+1}(\perp)$
- $f^k(\perp)$ je nejmenší pevný bod zobrazení f



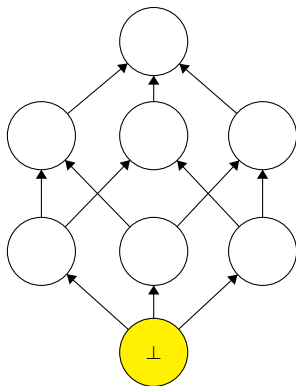
Knaster-Tarskiho teorém

Pevné body monotónního zobrazení $f : D \rightarrow D$ nad úplným svazem $(D, \sqsubseteq, \sqcup, \sqcap, \top, \perp)$ tvoří opět úplný svaz.

(To garantuje existenci nejmenšího/největšího pevného bodu f)

Nejmenší pevný bod

- iniciační hodnota $\perp$
- f je mon. rostoucí
- $\perp \sqsubseteq f(\perp) \Rightarrow f(\perp) \sqsubseteq f(f(\perp))$
- postupně počítáme $f(\perp), f(f(\perp)), f(f(f(\perp))), \dots$
- $\exists k \in \mathbb{N} : f^k(\perp) = f^{k+1}(\perp)$
- $f^k(\perp)$ je nejmenší pevný bod zobrazení f



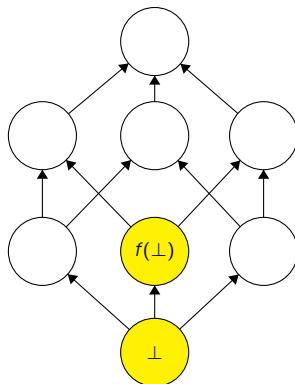
Knaster-Tarskiho teorém

Pevné body monotónního zobrazení $f : D \rightarrow D$ nad úplným svazem $(D, \sqsubseteq, \sqcup, \sqcap, \top, \perp)$ tvoří opět úplný svaz.

(To garantuje existenci nejmenšího/největšího pevného bodu f)

Nejmenší pevný bod

- iniciační hodnota $\perp$
- f je mon. rostoucí
- $\perp \sqsubseteq f(\perp) \Rightarrow f(\perp) \sqsubseteq f(f(\perp))$
- postupně počítáme $f(\perp), f(f(\perp)), f(f(f(\perp))), \dots$
- $\exists k \in \mathbb{N} : f^k(\perp) = f^{k+1}(\perp)$
- $f^k(\perp)$ je nejmenší pevný bod zobrazení f



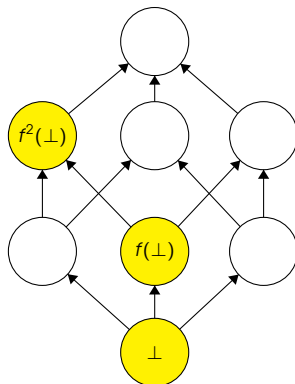
Knaster-Tarskiho teorém

Pevné body monotónního zobrazení $f : D \rightarrow D$ nad úplným svazem $(D, \sqsubseteq, \sqcup, \sqcap, \top, \perp)$ tvoří opět úplný svaz.

(To garantuje existenci nejmenšího/největšího pevného bodu f)

Nejmenší pevný bod

- iniciační hodnota $\perp$
- f je mon. rostoucí
- $\perp \sqsubseteq f(\perp) \Rightarrow f(\perp) \sqsubseteq f(f(\perp))$
- postupně počítáme $f(\perp), f(f(\perp)), f(f(f(\perp))), \dots$
- $\exists k \in \mathbb{N} : f^k(\perp) = f^{k+1}(\perp)$
- $f^k(\perp)$ je nejmenší pevný bod zobrazení f



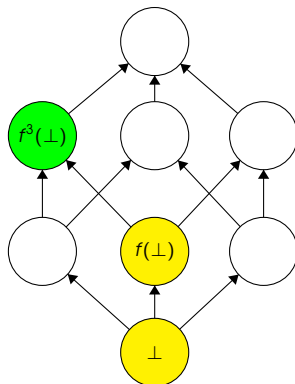
Knaster-Tarskiho teorém

Pevné body monotónního zobrazení $f : D \rightarrow D$ nad úplným svazem $(D, \sqsubseteq, \sqcup, \sqcap, \top, \perp)$ tvoří opět úplný svaz.

(To garantuje existenci nejmenšího/největšího pevného bodu f)

Nejmenší pevný bod

- iniciační hodnota $\perp$
- f je mon. rostoucí
- $\perp \sqsubseteq f(\perp) \Rightarrow f(\perp) \sqsubseteq f(f(\perp))$
- postupně počítáme $f(\perp), f(f(\perp)), f(f(f(\perp))), \dots$
- $\exists k \in \mathbb{N} : f^k(\perp) = f^{k+1}(\perp)$
- $f^k(\perp)$ je nejmenší pevný bod zobrazení f



Konstrukce algoritmu zpřesňujícího odhad výsledku

Pro **BKG** $G = (N, \Sigma, P, S)$ **sestrojte algoritmus počítající množinu**
$$N_\epsilon = \{A \in N \mid A \Rightarrow_G^+ \epsilon\}$$

Konstrukce algoritmu zpřesňujícího odhad výsledku

Pro **BKG** $G = (N, \Sigma, P, S)$ **sestrojte algoritmus počítající množinu**
 $N_\epsilon = \{A \in N \mid A \Rightarrow_G^+ \epsilon\}$

Analýza

- doména $D = 2^N$

Konstrukce algoritmu zpřesňujícího odhad výsledku

Pro **BKG** $G = (N, \Sigma, P, S)$ **sestrojte algoritmus počítající množinu**
 $N_\epsilon = \{A \in N \mid A \Rightarrow_G^+ \epsilon\}$

Analýza

- doména $D = 2^N$
- uspořádání $\sqsubseteq = \subseteq$

Konstrukce algoritmu zpřesňujícího odhad výsledku

Pro **BKG** $G = (N, \Sigma, P, S)$ **sestrojte algoritmus počítající množinu**
 $N_\epsilon = \{A \in N \mid A \Rightarrow_G^+ \epsilon\}$

Analýza

- doména $D = 2^N$
- uspořádání $\sqsubseteq = \subseteq$
- výchozí hodnota $\perp = \emptyset$

Konstrukce algoritmu zpřesňujícího odhad výsledku

Pro BKG $G = (N, \Sigma, P, S)$ **sestrojte algoritmus počítající množinu**
 $N_\epsilon = \{A \in N \mid A \Rightarrow_G^+ \epsilon\}$

Analýza

- doména $D = 2^N$
- uspořádání $\sqsubseteq = \subseteq$
- výchozí hodnota $\perp = \emptyset$
- úplný svaz $(2^N, \subseteq, \cup, \cap, N, \emptyset)$

Konstrukce algoritmu zpřesňujícího odhad výsledku

Pro **BKG** $G = (N, \Sigma, P, S)$ **sestrojte algoritmus počítající množinu**
 $N_\epsilon = \{A \in N \mid A \Rightarrow_G^+ \epsilon\}$

Analýza

- doména $D = 2^N$
- uspořádání $\sqsubseteq = \subseteq$
- výchozí hodnota $\perp = \emptyset$
- úplný svaz $(2^N, \subseteq, \cup, \cap, N, \emptyset)$
- zpřesňující funkce
 $f : 2^N \rightarrow 2^N$ mon. rostoucí

Konstrukce algoritmu zpřesňujícího odhad výsledku

Pro **BKG** $G = (N, \Sigma, P, S)$ sestrojte algoritmus počítající množinu

$$N_{\epsilon} = \{A \in N \mid A \Rightarrow_G^+ \epsilon\}$$

Analýza

- doména $D = 2^N$
- uspořádání $\sqsubseteq = \subseteq$
- výchozí hodnota $\perp = \emptyset$
- úplný svaz $(2^N, \subseteq, \cup, \cap, N, \emptyset)$
- zpřesňující funkce
 $f : 2^N \rightarrow 2^N$ mon. rostoucí
- $f(X) = \{A \in N \mid \exists (A, \alpha) \in P : \alpha \in X^*\}$

Konstrukce algoritmu zpřesňujícího odhad výsledku

Pro **BKG** $G = (N, \Sigma, P, S)$ sestojte algoritmus počítající množinu
$$N_\epsilon = \{A \in N \mid A \Rightarrow_G^+ \epsilon\}$$

Analýza

- doména $D = 2^N$
- uspořádání $\sqsubseteq = \subseteq$
- výchozí hodnota $\perp = \emptyset$
- úplný svaz $(2^N, \subseteq, \cup, \cap, N, \emptyset)$
- zpřesňující funkce
 $f : 2^N \rightarrow 2^N$ mon. rostoucí
- $f(X) = \{A \in N \mid \exists (A, \alpha) \in P : \alpha \in X^*\}$
- hledáme nejmenší pevný bod $f^k(\emptyset)$ funkce f , $k \in \mathbb{N}$

Konstrukce algoritmu zpřesňujícího odhad výsledku

Pro **BKG** $G = (N, \Sigma, P, S)$ **sestrojte algoritmus počítající množinu**
 $N_\epsilon = \{A \in N \mid A \Rightarrow_G^+ \epsilon\}$

Analýza

- doména $D = 2^N$
- uspořádání $\sqsubseteq = \subseteq$
- výchozí hodnota $\perp = \emptyset$
- úplný svaz $(2^N, \subseteq, \cup, \cap, N, \emptyset)$
- zpřesňující funkce
 $f : 2^N \rightarrow 2^N$ mon. rostoucí
- $f(X) = \{A \in N \mid \exists (A, \alpha) \in P : \alpha \in X^*\}$
- hledáme nejmenší pevný bod $f^k(\emptyset)$ funkce f , $k \in \mathbb{N}$

Algoritmus

Konstrukce algoritmu zpřesňujícího odhad výsledku

Pro **BKG** $G = (N, \Sigma, P, S)$ **sestrojte algoritmus počítající množinu**
 $N_\epsilon = \{A \in N \mid A \Rightarrow_G^+ \epsilon\}$

Analýza

- doména $D = 2^N$
- uspořádání $\sqsubseteq = \subseteq$
- výchozí hodnota $\perp = \emptyset$
- úplný svaz $(2^N, \subseteq, \cup, \cap, N, \emptyset)$
- zpřesňující funkce
 $f : 2^N \rightarrow 2^N$ mon. rostoucí
- $f(X) = \{A \in N \mid \exists (A, \alpha) \in P : \alpha \in X^*\}$
- hledáme nejmenší pevný bod $f^k(\emptyset)$ funkce f , $k \in \mathbb{N}$

Algoritmus

Vstup: $G = (N, \Sigma, P, S)$

Konstrukce algoritmu zpřesňujícího odhad výsledku

Pro **BKG** $G = (N, \Sigma, P, S)$ **sestrojte algoritmus počítající množinu**
 $N_\epsilon = \{A \in N \mid A \Rightarrow_G^+ \epsilon\}$

Analýza

- doména $D = 2^N$
- uspořádání $\sqsubseteq = \subseteq$
- výchozí hodnota $\perp = \emptyset$
- úplný svaz $(2^N, \subseteq, \cup, \cap, N, \emptyset)$
- zpřesňující funkce
 $f : 2^N \rightarrow 2^N$ mon. rostoucí
- $f(X) = \{A \in N \mid \exists (A, \alpha) \in P : \alpha \in X^*\}$
- hledáme nejmenší pevný bod $f^k(\emptyset)$ funkce f , $k \in \mathbb{N}$

Algoritmus

Vstup: $G = (N, \Sigma, P, S)$

Výstup: $N_\epsilon = \{A \in N \mid A \Rightarrow_G^+ \epsilon\}$

Konstrukce algoritmu zpřesňujícího odhad výsledku

Pro **BKG** $G = (N, \Sigma, P, S)$ **sestrojte algoritmus počítající množinu**
 $N_\epsilon = \{A \in N \mid A \Rightarrow_G^+ \epsilon\}$

Analýza

- doména $D = 2^N$
- uspořádání $\sqsubseteq = \subseteq$
- výchozí hodnota $\perp = \emptyset$
- úplný svaz $(2^N, \subseteq, \cup, \cap, N, \emptyset)$
- zpřesňující funkce
 $f : 2^N \rightarrow 2^N$ mon. rostoucí
- $f(X) = \{A \in N \mid \exists (A, \alpha) \in P : \alpha \in X^*\}$
- hledáme nejmenší pevný bod $f^k(\emptyset)$ funkce f , $k \in \mathbb{N}$

Algoritmus

Vstup: $G = (N, \Sigma, P, S)$

Výstup: $N_\epsilon = \{A \in N \mid A \Rightarrow_G^+ \epsilon\}$

Postup:

Konstrukce algoritmu zpřesňujícího odhad výsledku

Pro BKG $G = (N, \Sigma, P, S)$ **sestrojte algoritmus počítající množinu**
 $N_\epsilon = \{A \in N \mid A \Rightarrow_G^+ \epsilon\}$

Analýza

- doména $D = 2^N$
- uspořádání $\sqsubseteq = \subseteq$
- výchozí hodnota $\perp = \emptyset$
- úplný svaz $(2^N, \subseteq, \cup, \cap, N, \emptyset)$
- zpřesňující funkce
 $f : 2^N \rightarrow 2^N$ mon. rostoucí
- $f(X) = \{A \in N \mid \exists (A, \alpha) \in P : \alpha \in X^*\}$
- hledáme nejmenší pevný bod $f^k(\emptyset)$ funkce f , $k \in \mathbb{N}$

Algoritmus

Vstup: $G = (N, \Sigma, P, S)$

Výstup: $N_\epsilon = \{A \in N \mid A \Rightarrow_G^+ \epsilon\}$

Postup:

1 $N_\epsilon^0 = \emptyset, i = 0$

Konstrukce algoritmu zpřesňujícího odhad výsledku

Pro **BKG** $G = (N, \Sigma, P, S)$ **sestrojte algoritmus počítající množinu**
 $N_\epsilon = \{A \in N \mid A \Rightarrow_G^+ \epsilon\}$

Analýza

- doména $D = 2^N$
- uspořádání $\sqsubseteq = \subseteq$
- výchozí hodnota $\perp = \emptyset$
- úplný svaz $(2^N, \subseteq, \cup, \cap, N, \emptyset)$
- zpřesňující funkce
 $f : 2^N \rightarrow 2^N$ mon. rostoucí
- $f(X) = \{A \in N \mid \exists (A, \alpha) \in P : \alpha \in X^*\}$
- hledáme nejmenší pevný bod $f^k(\emptyset)$ funkce f , $k \in \mathbb{N}$

Algoritmus

Vstup: $G = (N, \Sigma, P, S)$

Výstup: $N_\epsilon = \{A \in N \mid A \Rightarrow_G^+ \epsilon\}$

Postup:

- 1 $N_\epsilon^0 = \emptyset, i = 0$
- 2 $N_\epsilon^{i+1} = \{A \in N \mid \exists (A, \alpha) \in P : \alpha \in (N_\epsilon^i)^*\}$

Konstrukce algoritmu zpřesňujícího odhad výsledku

Pro **BKG** $G = (N, \Sigma, P, S)$ **sestrojte algoritmus počítající množinu**
 $N_\epsilon = \{A \in N \mid A \Rightarrow_G^+ \epsilon\}$

Analýza

- doména $D = 2^N$
- uspořádání $\sqsubseteq = \subseteq$
- výchozí hodnota $\perp = \emptyset$
- úplný svaz $(2^N, \subseteq, \cup, \cap, N, \emptyset)$
- zpřesňující funkce
 $f : 2^N \rightarrow 2^N$ mon. rostoucí
- $f(X) = \{A \in N \mid \exists (A, \alpha) \in P : \alpha \in X^*\}$
- hledáme nejmenší pevný bod $f^k(\emptyset)$ funkce f , $k \in \mathbb{N}$

Algoritmus

Vstup: $G = (N, \Sigma, P, S)$

Výstup: $N_\epsilon = \{A \in N \mid A \Rightarrow_G^+ \epsilon\}$

Postup:

- 1 $N_\epsilon^0 = \emptyset, i = 0$
- 2 $N_\epsilon^{i+1} = \{A \in N \mid \exists (A, \alpha) \in P : \alpha \in (N_\epsilon^i)^*\}$
- 3 pokud $N_\epsilon^i \neq N_\epsilon^{i+1}$
 - $i = i + 1$
 - přejdi na bod 2

Konstrukce algoritmu zpřesňujícího odhad výsledku

Pro **BKG** $G = (N, \Sigma, P, S)$ **sestrojte algoritmus počítající množinu**
 $N_\epsilon = \{A \in N \mid A \Rightarrow_G^+ \epsilon\}$

Analýza

- doména $D = 2^N$
- uspořádání $\sqsubseteq = \subseteq$
- výchozí hodnota $\perp = \emptyset$
- úplný svaz $(2^N, \subseteq, \cup, \cap, N, \emptyset)$
- zpřesňující funkce
 $f : 2^N \rightarrow 2^N$ mon. rostoucí
- $f(X) = \{A \in N \mid \exists (A, \alpha) \in P : \alpha \in X^*\}$
- hledáme nejmenší pevný bod $f^k(\emptyset)$ funkce f , $k \in \mathbb{N}$

Algoritmus

Vstup: $G = (N, \Sigma, P, S)$

Výstup: $N_\epsilon = \{A \in N \mid A \Rightarrow_G^+ \epsilon\}$

Postup:

- 1 $N_\epsilon^0 = \emptyset, i = 0$
- 2 $N_\epsilon^{i+1} = \{A \in N \mid \exists (A, \alpha) \in P : \alpha \in (N_\epsilon^i)^*\}$
- 3 pokud $N_\epsilon^i \neq N_\epsilon^{i+1}$
 - $i = i + 1$
 - přejdi na bod 2
- 4 $N_\epsilon = N_\epsilon^i$

Gramatika

- Demonstrujte algoritmus na BKG $G = (N, \Sigma, P, S)$
- $N = \{S, A, B, C\}$
- $\Sigma = \{a, b, c\}$
- P :
 - 1 $S \rightarrow AAAB \mid BC \mid c$
 - 2 $A \rightarrow a \mid aB \mid \epsilon$
 - 3 $B \rightarrow cBb \mid AA$
 - 4 $C \rightarrow ccC \mid Ab$

Gramatika

- Demonstrujte algoritmus na BKG $G = (N, \Sigma, P, S)$
- $N = \{S, A, B, C\}$
- $\Sigma = \{a, b, c\}$
- P :
 - 1 $S \rightarrow AAAB \mid BC \mid c$
 - 2 $A \rightarrow a \mid aB \mid \epsilon$
 - 3 $B \rightarrow cBb \mid AA$
 - 4 $C \rightarrow ccC \mid Ab$

Algoritmus

Postup:

- 1 $N_\epsilon^0 = \emptyset, i = 0$
- 2 $N_\epsilon^{i+1} = \{A \in N \mid \exists (A, \alpha) \in P : \alpha \in (N_\epsilon^i)^*\}$
- 3 pokud $N_\epsilon^i \neq N_\epsilon^{i+1}$
 - $i = i + 1$
 - přejdi na bod 2
- 4 $N_\epsilon = N_\epsilon^i$

Gramatika

- Demonstrujte algoritmus na BKG $G = (N, \Sigma, P, S)$
- $N = \{S, A, B, C\}$
- $\Sigma = \{a, b, c\}$
- P :
 - 1 $S \rightarrow AAAB \mid BC \mid c$
 - 2 $A \rightarrow a \mid aB \mid \epsilon$
 - 3 $B \rightarrow cBb \mid AA$
 - 4 $C \rightarrow ccC \mid Ab$

Algoritmus

Postup:

- 1 $N_\epsilon^0 = \emptyset, i = 0$
- 2 $N_\epsilon^{i+1} = \{A \in N \mid \exists (A, \alpha) \in P : \alpha \in (N_\epsilon^i)^*\}$
- 3 pokud $N_\epsilon^i \neq N_\epsilon^{i+1}$
 - $i = i + 1$
 - přejdi na bod 2
- 4 $N_\epsilon = N_\epsilon^i$

Demonstrace

Gramatika

- Demonstrujte algoritmus na BKG $G = (N, \Sigma, P, S)$
- $N = \{S, A, B, C\}$
- $\Sigma = \{a, b, c\}$
- P :
 - 1 $S \rightarrow AAAB \mid BC \mid c$
 - 2 $A \rightarrow a \mid aB \mid \epsilon$
 - 3 $B \rightarrow cBb \mid AA$
 - 4 $C \rightarrow ccC \mid Ab$

Algoritmus

Postup:

- 1 $N_\epsilon^0 = \emptyset, i = 0$
- 2 $N_\epsilon^{i+1} = \{A \in N \mid \exists (A, \alpha) \in P : \alpha \in (N_\epsilon^i)^*\}$
- 3 pokud $N_\epsilon^i \neq N_\epsilon^{i+1}$
 - $i = i + 1$
 - přejdi na bod 2
- 4 $N_\epsilon = N_\epsilon^i$

Demonstrace

- $N_\epsilon^0 = \emptyset$

Gramatika

- Demonstrujte algoritmus na BKG $G = (N, \Sigma, P, S)$
- $N = \{S, A, B, C\}$
- $\Sigma = \{a, b, c\}$
- P :
 - 1 $S \rightarrow AAAB \mid BC \mid c$
 - 2 $A \rightarrow a \mid aB \mid \epsilon$
 - 3 $B \rightarrow cBb \mid AA$
 - 4 $C \rightarrow ccC \mid Ab$

Algoritmus

Postup:

- 1 $N_\epsilon^0 = \emptyset, i = 0$
- 2 $N_\epsilon^{i+1} = \{A \in N \mid \exists (A, \alpha) \in P : \alpha \in (N_\epsilon^i)^*\}$
- 3 pokud $N_\epsilon^i \neq N_\epsilon^{i+1}$
 - $i = i + 1$
 - přejdi na bod 2
- 4 $N_\epsilon = N_\epsilon^i$

Demonstrace

- $N_\epsilon^0 = \emptyset$
- $N_\epsilon^1 = \{A \in N \mid \exists (A, \alpha) \in P : \alpha \in (N_\epsilon^0)^* = \emptyset^* = \{\epsilon\}\} = \{A\}$

Gramatika

- Demonstrujte algoritmus na BKG $G = (N, \Sigma, P, S)$
- $N = \{S, A, B, C\}$
- $\Sigma = \{a, b, c\}$
- P :
 - 1 $S \rightarrow AAAB \mid BC \mid c$
 - 2 $A \rightarrow a \mid aB \mid \epsilon$
 - 3 $B \rightarrow cBb \mid AA$
 - 4 $C \rightarrow ccC \mid Ab$

Algoritmus

Postup:

- 1 $N_\epsilon^0 = \emptyset, i = 0$
- 2 $N_\epsilon^{i+1} = \{A \in N \mid \exists (A, \alpha) \in P : \alpha \in (N_\epsilon^i)^*\}$
- 3 pokud $N_\epsilon^i \neq N_\epsilon^{i+1}$
 - $i = i + 1$
 - přejdi na bod 2
- 4 $N_\epsilon = N_\epsilon^i$

Demonstrace

- $N_\epsilon^0 = \emptyset$
- $N_\epsilon^1 = \{A \in N \mid \exists (A, \alpha) \in P : \alpha \in (N_\epsilon^0)^* = \emptyset^* = \{\epsilon\}\} = \{A\}$
- $N_\epsilon^2 = \{A \in N \mid \exists (A, \alpha) \in P : \alpha \in (N_\epsilon^1)^* = \{A\}^*\} = \{A, B\}$

Gramatika

- Demonstrujte algoritmus na BKG $G = (N, \Sigma, P, S)$
- $N = \{S, A, B, C\}$
- $\Sigma = \{a, b, c\}$
- P :
 - 1 $S \rightarrow AAAB \mid BC \mid c$
 - 2 $A \rightarrow a \mid aB \mid \epsilon$
 - 3 $B \rightarrow cBb \mid AA$
 - 4 $C \rightarrow ccC \mid Ab$

Algoritmus

Postup:

- 1 $N_\epsilon^0 = \emptyset, i = 0$
- 2 $N_\epsilon^{i+1} = \{A \in N \mid \exists (A, \alpha) \in P : \alpha \in (N_\epsilon^i)^*\}$
- 3 pokud $N_\epsilon^i \neq N_\epsilon^{i+1}$
 - $i = i + 1$
 - přejdi na bod 2
- 4 $N_\epsilon = N_\epsilon^i$

Demonstrace

- $N_\epsilon^3 = \{A \in N \mid \exists (A, \alpha) \in P : \alpha \in (N_\epsilon^2)^* = \{A, B\}^*\} = \{A, B, S\}$

Gramatika

- Demonstrujte algoritmus na BKG $G = (N, \Sigma, P, S)$
- $N = \{S, A, B, C\}$
- $\Sigma = \{a, b, c\}$
- P :
 - 1 $S \rightarrow AAAB \mid BC \mid c$
 - 2 $A \rightarrow a \mid aB \mid \epsilon$
 - 3 $B \rightarrow cBb \mid AA$
 - 4 $C \rightarrow ccC \mid Ab$

Algoritmus

Postup:

- 1 $N_\epsilon^0 = \emptyset, i = 0$
- 2 $N_\epsilon^{i+1} = \{A \in N \mid \exists (A, \alpha) \in P : \alpha \in (N_\epsilon^i)^*\}$
- 3 pokud $N_\epsilon^i \neq N_\epsilon^{i+1}$
 - $i = i + 1$
 - přejdi na bod 2
- 4 $N_\epsilon = N_\epsilon^i$

Demonstrace

- $N_\epsilon^3 = \{A \in N \mid \exists (A, \alpha) \in P : \alpha \in (N_\epsilon^2)^* = \{A, B\}^*\} = \{A, B, S\}$
- $N_\epsilon^4 = \{A \in N \mid \exists (A, \alpha) \in P : \alpha \in (N_\epsilon^3)^* = \{A, B, S\}^*\} = \{A, B, S\}$

Gramatika

- Demonstrujte algoritmus na BKG $G = (N, \Sigma, P, S)$
- $N = \{S, A, B, C\}$
- $\Sigma = \{a, b, c\}$
- P :
 - 1 $S \rightarrow AAAB \mid BC \mid c$
 - 2 $A \rightarrow a \mid aB \mid \epsilon$
 - 3 $B \rightarrow cBb \mid AA$
 - 4 $C \rightarrow ccC \mid Ab$

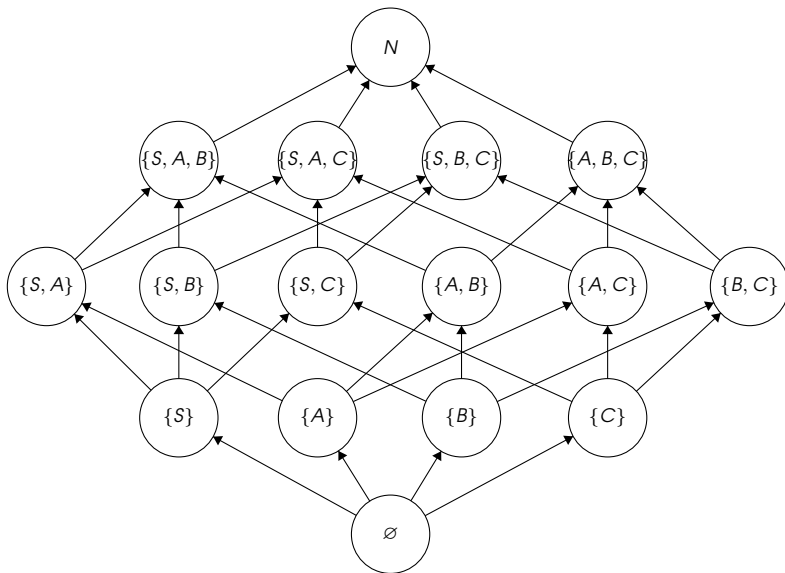
Algoritmus

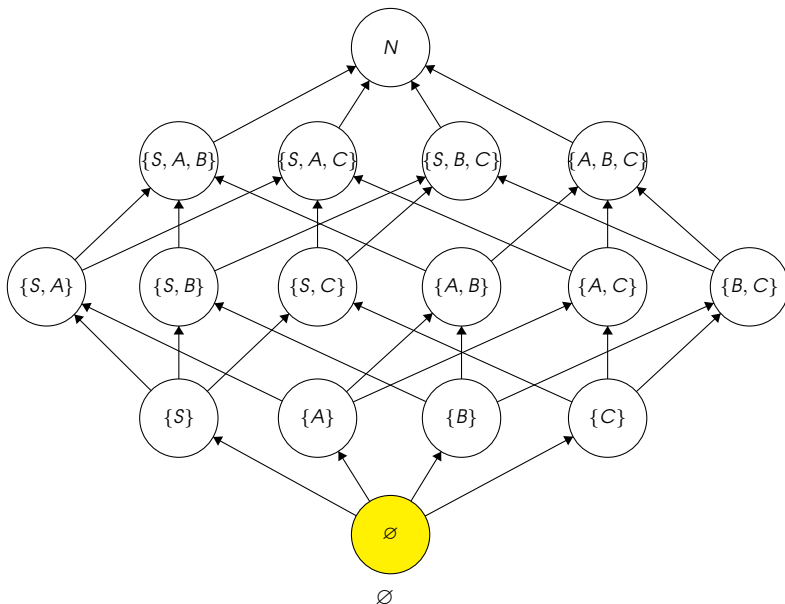
Postup:

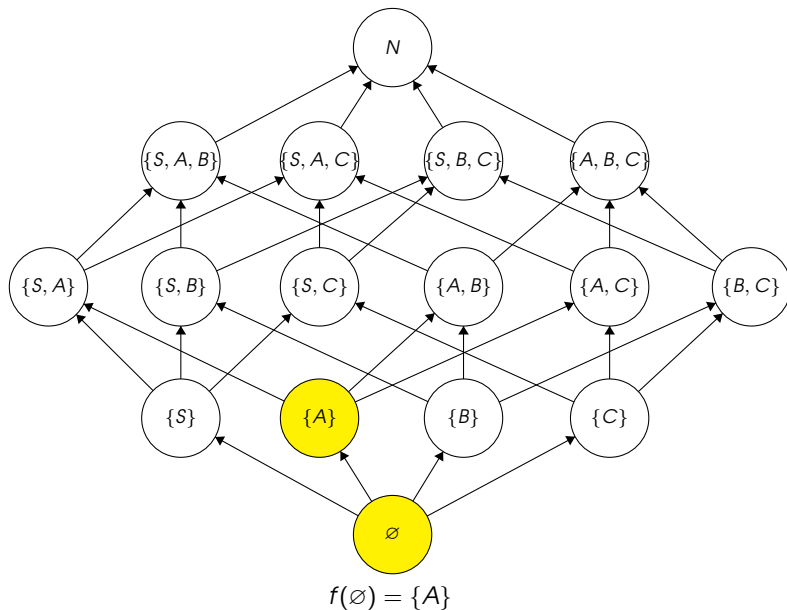
- 1 $N_\epsilon^0 = \emptyset, i = 0$
- 2 $N_\epsilon^{i+1} = \{A \in N \mid \exists (A, \alpha) \in P : \alpha \in (N_\epsilon^i)^*\}$
- 3 pokud $N_\epsilon^i \neq N_\epsilon^{i+1}$
 - $i = i + 1$
 - přejdi na bod 2
- 4 $N_\epsilon = N_\epsilon^i$

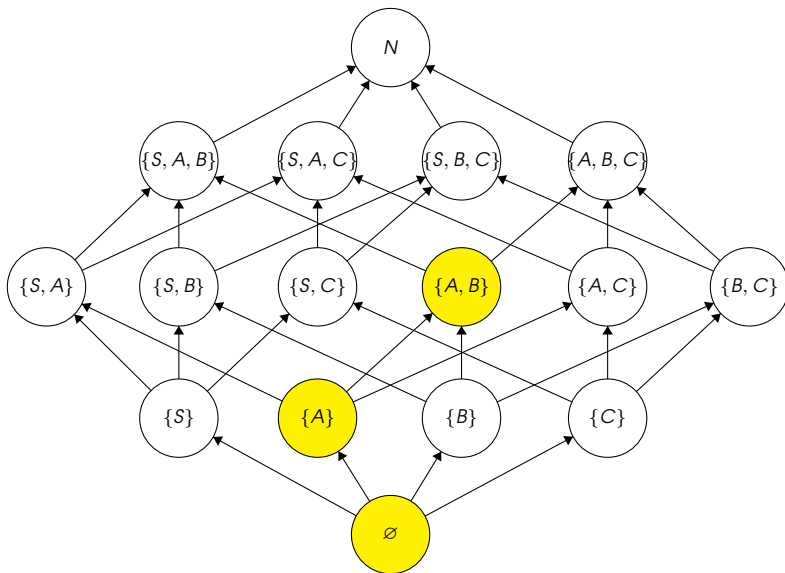
Demonstrace

- $N_\epsilon^3 = \{A \in N \mid \exists (A, \alpha) \in P : \alpha \in (N_\epsilon^2)^* = \{A, B\}^*\} = \{A, B, S\}$
- $N_\epsilon^4 = \{A \in N \mid \exists (A, \alpha) \in P : \alpha \in (N_\epsilon^3)^* = \{A, B, S\}^*\} = \{A, B, S\}$
- $N_\epsilon = N_\epsilon^3 = \{A, B, S\}$

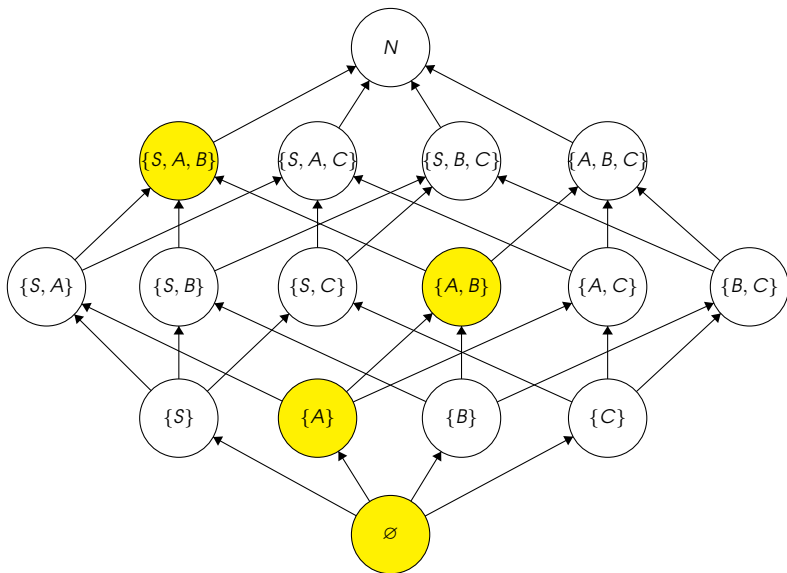




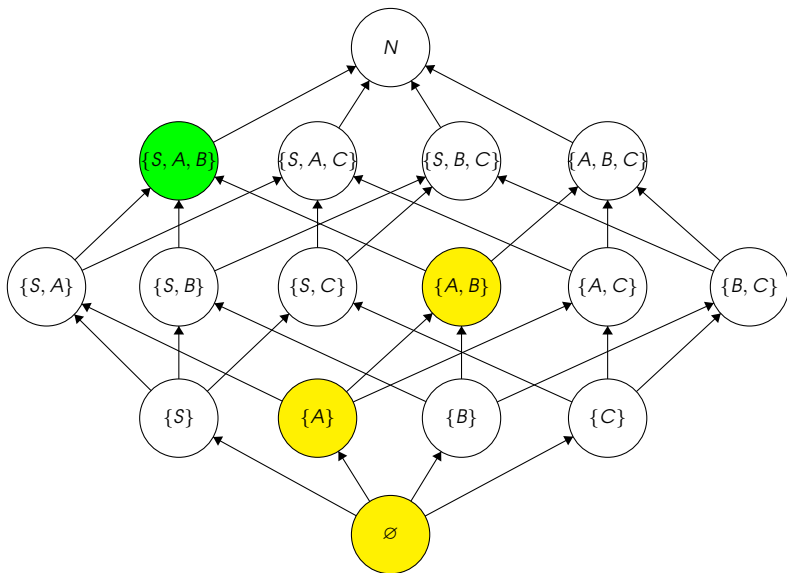




$$f^2(\emptyset) = f(\{A\}) = \{A, B\}$$



$$f^3(\emptyset) = f^2(\{A\}) = f(\{A, B\}) = \{A, B, S\}$$



$$f^4(\emptyset) = f^3(\{A\}) = f^2(\{A, B\}) = f(\{A, B, S\}) = \{A, B, S\}$$

Konstrukce algoritmu zpřesňujícího odhad výsledku

Pro **BKG** $G = (N, \Sigma, P, S)$ **sestrojte algoritmus počítající množinu**
$$N_{wa} = \{A \in N \mid \exists w \in \Sigma^* : A \Rightarrow_G^+ wa\}$$

Konstrukce algoritmu zpřesňujícího odhad výsledku

Pro **BKG** $G = (N, \Sigma, P, S)$ **sestrojte algoritmus počítající množinu**
 $N_{wa} = \{A \in N \mid \exists w \in \Sigma^* : A \Rightarrow_G^+ wa\}$

Idea

$$A \rightarrow \alpha \beta \gamma$$

- Hledáme pravidla typu $A \rightarrow \alpha\beta\gamma$, kde
 - α je přepsatelná na jakékoliv terminální symboly
 - β nám garantuje, že vygeneruje větu se symbolem a vpravo
 - γ je přepsatelná na ϵ

Konstrukce algoritmu zpřesňujícího odhad výsledku

Pro **BKG** $G = (N, \Sigma, P, S)$ **sestrojte algoritmus počítající množinu**
 $N_{wa} = \{A \in N \mid \exists w \in \Sigma^* : A \Rightarrow_G^+ wa\}$

Idea

$$A \rightarrow \alpha \beta \gamma$$

- Hledáme pravidla typu $A \rightarrow \alpha\beta\gamma$, kde
 - α je přepsatelná na jakékoliv terminální symboly
 - β nám garantuje, že vygeneruje větu se symbolem a vpravo
 - γ je přepsatelná na ϵ

Pro výpočet potřebujeme množiny

- $N_\epsilon = \{A \in N \mid A \Rightarrow^* \epsilon\}$
- $N_f = \{A \in N \mid \exists w \in \Sigma^* : A \Rightarrow^* w\}$

Konstrukce algoritmu zpřesňujícího odhad výsledku

Pro **BKG** $G = (N, \Sigma, P, S)$ sestrojte algoritmus počítající množinu
 $N_{wa} = \{A \in N \mid \exists w \in \Sigma^* : A \Rightarrow_G^+ wa\}$

Analýza

- doména $D = 2^N$

Konstrukce algoritmu zpřesňujícího odhad výsledku

Pro **BKG** $G = (N, \Sigma, P, S)$ **sestrojte algoritmus počítající množinu**
 $N_{wa} = \{A \in N \mid \exists w \in \Sigma^* : A \Rightarrow_G^+ wa\}$

Analýza

- doména $D = 2^N$
- uspořádání $\sqsubseteq = \subseteq$

Konstrukce algoritmu zpřesňujícího odhad výsledku

Pro **BKG** $G = (N, \Sigma, P, S)$ **sestrojte algoritmus počítající množinu**
 $N_{wa} = \{A \in N \mid \exists w \in \Sigma^* : A \Rightarrow_G^+ wa\}$

Analýza

- doména $D = 2^N$
- uspořádání $\sqsubseteq = \subseteq$
- výchozí hodnota $\perp = \emptyset$

Konstrukce algoritmu zpřesňujícího odhad výsledku

Pro **BKG** $G = (N, \Sigma, P, S)$ **sestrojte algoritmus počítající množinu**
 $N_{wa} = \{A \in N \mid \exists w \in \Sigma^* : A \Rightarrow_G^+ wa\}$

Analýza

- doména $D = 2^N$
- uspořádání $\sqsubseteq = \subseteq$
- výchozí hodnota $\perp = \emptyset$
- úplný svaz $(2^N, \subseteq, \cup, \cap, N, \emptyset)$

Konstrukce algoritmu zpřesňujícího odhad výsledku

Pro **BKG** $G = (N, \Sigma, P, S)$ **sestrojte algoritmus počítající množinu**
 $N_{wa} = \{A \in N \mid \exists w \in \Sigma^* : A \Rightarrow_G^+ wa\}$

Analýza

- doména $D = 2^N$
- uspořádání $\sqsubseteq = \subseteq$
- výchozí hodnota $\perp = \emptyset$
- úplný svaz $(2^N, \subseteq, \cup, \cap, N, \emptyset)$
- zpřesňující funkce $f : 2^N \rightarrow 2^N$ mon. rostoucí

Konstrukce algoritmu zpřesňujícího odhad výsledku

Pro **BKG** $G = (N, \Sigma, P, S)$ **sestrojte algoritmus počítající množinu**
 $N_{wa} = \{A \in N \mid \exists w \in \Sigma^* : A \Rightarrow_G^+ wa\}$

Analýza

- doména $D = 2^N$
- uspořádání $\sqsubseteq = \subseteq$
- výchozí hodnota $\perp = \emptyset$
- úplný svaz $(2^N, \subseteq, \cup, \cap, N, \emptyset)$
- zpřesňující funkce $f : 2^N \rightarrow 2^N$ mon. rostoucí
- $f(X) = \{A \in N \mid \exists (A, \alpha\beta\gamma) \in P : \alpha \in (N_f \cup \Sigma)^*, \beta \in (X \cup \{a\}) \wedge \gamma \in N_\epsilon^*\}$

Konstrukce algoritmu zpřesňujícího odhad výsledku

Pro **BKG** $G = (N, \Sigma, P, S)$ **sestrojte algoritmus počítající množinu**
 $N_{wa} = \{A \in N \mid \exists w \in \Sigma^* : A \Rightarrow_G^+ wa\}$

Analýza

- doména $D = 2^N$
- uspořádání $\sqsubseteq = \subseteq$
- výchozí hodnota $\perp = \emptyset$
- úplný svaz $(2^N, \subseteq, \cup, \cap, N, \emptyset)$
- zpřesňující funkce $f : 2^N \rightarrow 2^N$ mon. rostoucí
- $f(X) = \{A \in N \mid \exists (A, \alpha\beta\gamma) \in P : \alpha \in (N_t \cup \Sigma)^*, \beta \in (X \cup \{a\}) \wedge \gamma \in N_\epsilon^*\}$
- hledáme nejmenší pevný bod $f^k(\emptyset)$ funkce f , $k \in \mathbb{N}$

Konstrukce algoritmu zpřesňujícího odhad výsledku

Pro BKG $G = (N, \Sigma, P, S)$ **sestrojte algoritmus počítající množinu**
 $N_{wa} = \{A \in N \mid \exists w \in \Sigma^* : A \Rightarrow_G^+ wa\}$

Algoritmus

- **Vstup:** BKG $G = (N, \Sigma, P, S)$, $a \in \Sigma$
- **Výstup:** množina $N_{wa} = \{A \in N \mid \exists w \in \Sigma^* : A \Rightarrow^* wa\}$
- **Postup:**
 - ① $N_{wa}^0 = \emptyset, i = 0$
 - ② $N_{wa}^{i+1} = \{A \in N \mid \exists (A \rightarrow \alpha) \in P : \alpha \in (N_t \cup \Sigma)^* (N_{wa}^i \cup \{a\}) N_\epsilon^*\}$
 - ③ pokud $N_{wa}^{i+1} \neq N_{wa}^i$
 - $i = i + 1$
 - přejí na bod 2
 - ④ $N_{wa} = N_{wa}^i$

Gramatika

- Demonstrujte algoritmus na BKG $G = (N, \Sigma, P, S)$
- $N = \{S, A, B, C\}$
- $\Sigma = \{a, b, c\}$
- P :
 - 1 $S \rightarrow ABBbCAS \mid bA \mid \epsilon$
 - 2 $A \rightarrow \epsilon \mid cA$
 - 3 $B \rightarrow bba \mid cB$
 - 4 $C \rightarrow aBA \mid Bc$

Gramatika

- Demonstrujte algoritmus na BKG $G = (N, \Sigma, P, S)$
- $N = \{S, A, B, C\}$
- $\Sigma = \{a, b, c\}$
- P :
 - 1 $S \rightarrow ABBbCAS \mid bA \mid \epsilon$
 - 2 $A \rightarrow \epsilon \mid cA$
 - 3 $B \rightarrow bba \mid cB$
 - 4 $C \rightarrow aBA \mid Bc$

Algoritmus

Postup:

- 1 $N_{wa}^0 = \emptyset, i = 0$
- 2 $N_{wa}^{i+1} = \{A \in N \mid \exists (A, \alpha) \in P : \alpha \in (N_t \cup \Sigma)^*(N_{wa}^i \cup \{a\})N_\epsilon^*\}$
- 3 pokud $N_{wa}^i \neq N_{wa}^{i+1}$
 - $i = i + 1$
 - přejdi na bod 2
- 4 $N_{wa} = N_{wa}^i$

Gramatika

- Demonstrujte algoritmus na BKG $G = (N, \Sigma, P, S)$
- $N = \{S, A, B, C\}$
- $\Sigma = \{a, b, c\}$
- P :
 - 1 $S \rightarrow ABBbCAS \mid bA \mid \epsilon$
 - 2 $A \rightarrow \epsilon \mid cA$
 - 3 $B \rightarrow bba \mid cB$
 - 4 $C \rightarrow aBA \mid Bc$

Algoritmus

Postup:

- 1 $N_{wa}^0 = \emptyset, i = 0$
- 2 $N_{wa}^{i+1} = \{A \in N \mid \exists (A, \alpha) \in P : \alpha \in (N_t \cup \Sigma)^* (N_{wa}^i \cup \{a\}) N_\epsilon^*\}$
- 3 pokud $N_{wa}^i \neq N_{wa}^{i+1}$
 - $i = i + 1$
 - přejdi na bod 2
- 4 $N_{wa} = N_{wa}^i$

Množiny

$$N_\epsilon = \{A, S\}$$

$$N_t = \{A, B, C, S\}$$

Gramatika

- Demonstrujte algoritmus na BKG $G = (N, \Sigma, P, S)$
- $N = \{S, A, B, C\}$
- $\Sigma = \{a, b, c\}$
- P :
 - 1 $S \rightarrow ABBbCAS \mid bA \mid \epsilon$
 - 2 $A \rightarrow \epsilon \mid cA$
 - 3 $B \rightarrow bba \mid cB$
 - 4 $C \rightarrow aBA \mid Bc$

Algoritmus

Postup:

- 1 $N_{wa}^0 = \emptyset, i = 0$
- 2 $N_{wa}^{i+1} = \{A \in N \mid \exists (A, \alpha) \in P : \alpha \in (N_t \cup \Sigma)^*(N_{wa}^i \cup \{a\})N_\epsilon^*\}$
- 3 pokud $N_{wa}^i \neq N_{wa}^{i+1}$
 - $i = i + 1$
 - přejdi na bod 2
- 4 $N_{wa} = N_{wa}^i$

Množiny

$$N_\epsilon = \{A, S\}$$

$$N_t = \{A, B, C, S\}$$

Demonstrace

$$\bullet N_{wa}^0 = \emptyset$$

Gramatika

- Demonstrujte algoritmus na BKG $G = (N, \Sigma, P, S)$
- $N = \{S, A, B, C\}$
- $\Sigma = \{a, b, c\}$
- P :
 - 1 $S \rightarrow ABBbCAS \mid bA \mid \epsilon$
 - 2 $A \rightarrow \epsilon \mid cA$
 - 3 $B \rightarrow bba \mid cB$
 - 4 $C \rightarrow aBA \mid Bc$

Algoritmus

Postup:

- 1 $N_{wa}^0 = \emptyset, i = 0$
- 2 $N_{wa}^{i+1} = \{A \in N \mid \exists (A, \alpha) \in P : \alpha \in (N_t \cup \Sigma)^*(N_{wa}^i \cup \{a\})N_\epsilon^*\}$
- 3 pokud $N_{wa}^i \neq N_{wa}^{i+1}$
 - $i = i + 1$
 - přejdi na bod 2
- 4 $N_{wa} = N_{wa}^i$

Množiny

$$N_\epsilon = \{A, S\}$$

$$N_t = \{A, B, C, S\}$$

Demonstrace

- $N_{wa}^0 = \emptyset$
- $N_{wa}^1 = \{B\}$

Gramatika

- Demonstrujte algoritmus na BKG $G = (N, \Sigma, P, S)$
- $N = \{S, A, B, C\}$
- $\Sigma = \{a, b, c\}$
- P :
 - 1 $S \rightarrow ABBbCAS \mid bA \mid \epsilon$
 - 2 $A \rightarrow \epsilon \mid cA$
 - 3 $B \rightarrow bba \mid cB$
 - 4 $C \rightarrow aBA \mid Bc$

Algoritmus

Postup:

- 1 $N_{wa}^0 = \emptyset, i = 0$
- 2 $N_{wa}^{i+1} = \{A \in N \mid \exists (A, \alpha) \in P : \alpha \in (N_t \cup \Sigma)^*(N_{wa}^i \cup \{a\})N_\epsilon^*\}$
- 3 pokud $N_{wa}^i \neq N_{wa}^{i+1}$
 - $i = i + 1$
 - přejdi na bod 2
- 4 $N_{wa} = N_{wa}^i$

Množiny

$$N_\epsilon = \{A, S\}$$

$$N_t = \{A, B, C, S\}$$

Demonstrace

$$\bullet N_{wa}^2 = \{B, C\}$$

Gramatika

- Demonstrujte algoritmus na BKG $G = (N, \Sigma, P, S)$
- $N = \{S, A, B, C\}$
- $\Sigma = \{a, b, c\}$
- P :
 - 1 $S \rightarrow ABBbCAS \mid bA \mid \epsilon$
 - 2 $A \rightarrow \epsilon \mid cA$
 - 3 $B \rightarrow bba \mid cB$
 - 4 $C \rightarrow aBA \mid Bc$

Algoritmus

Postup:

- 1 $N_{wa}^0 = \emptyset, i = 0$
- 2 $N_{wa}^{i+1} = \{A \in N \mid \exists (A, \alpha) \in P : \alpha \in (N_f \cup \Sigma)^*(N_{wa}^i \cup \{a\})N_\epsilon^*\}$
- 3 pokud $N_{wa}^i \neq N_{wa}^{i+1}$
 - $i = i + 1$
 - přejdi na bod 2
- 4 $N_{wa} = N_{wa}^i$

Množiny

$$N_\epsilon = \{A, S\}$$

$$N_f = \{A, B, C, S\}$$

Demonstrace

- $N_{wa}^2 = \{B, C\}$
- $N_{wa}^3 = \{B, C, S\}$

Gramatika

- Demonstrujte algoritmus na BKG $G = (N, \Sigma, P, S)$
- $N = \{S, A, B, C\}$
- $\Sigma = \{a, b, c\}$
- P :
 - 1 $S \rightarrow ABBbCAS \mid bA \mid \epsilon$
 - 2 $A \rightarrow \epsilon \mid cA$
 - 3 $B \rightarrow bba \mid cB$
 - 4 $C \rightarrow aBA \mid Bc$

Algoritmus

Postup:

- 1 $N_{wa}^0 = \emptyset, i = 0$
- 2 $N_{wa}^{i+1} = \{A \in N \mid \exists (A, \alpha) \in P : \alpha \in (N_t \cup \Sigma)^*(N_{wa}^i \cup \{a\})N_\epsilon^*\}$
- 3 pokud $N_{wa}^i \neq N_{wa}^{i+1}$
 - $i = i + 1$
 - přejdi na bod 2
- 4 $N_{wa} = N_{wa}^i$

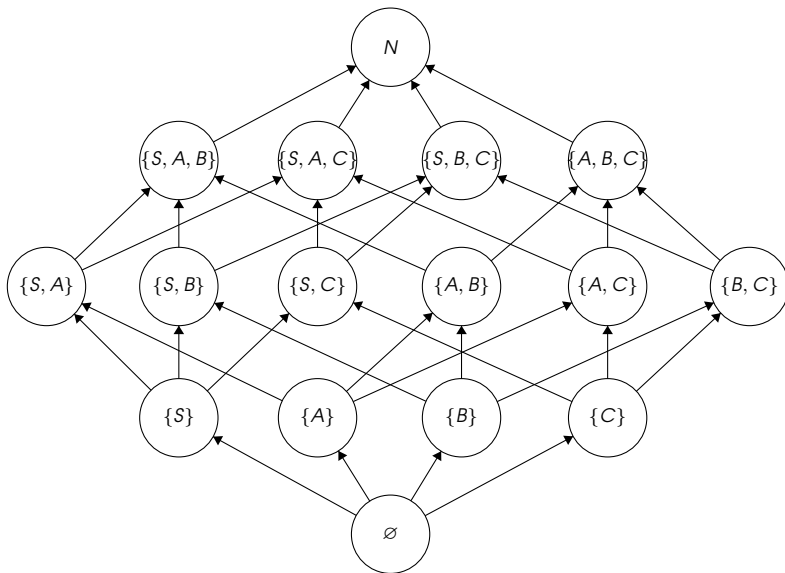
Množiny

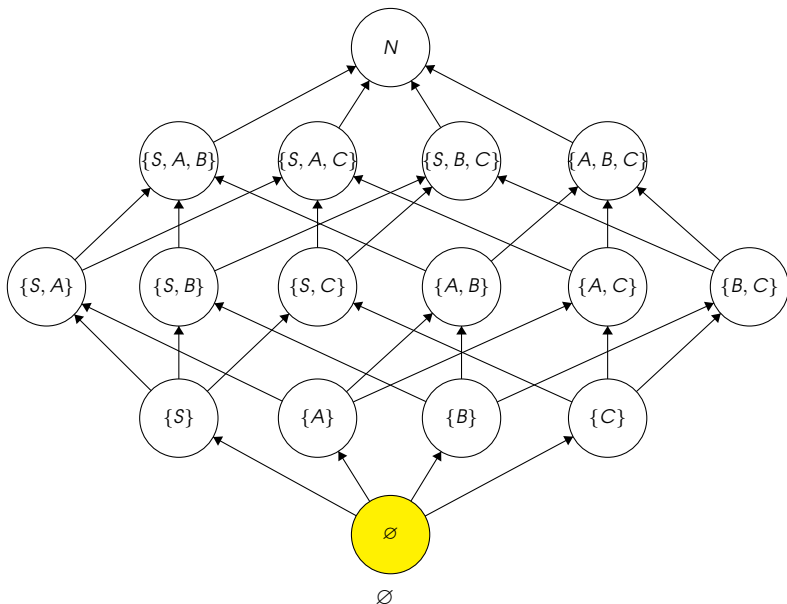
$$N_\epsilon = \{A, S\}$$

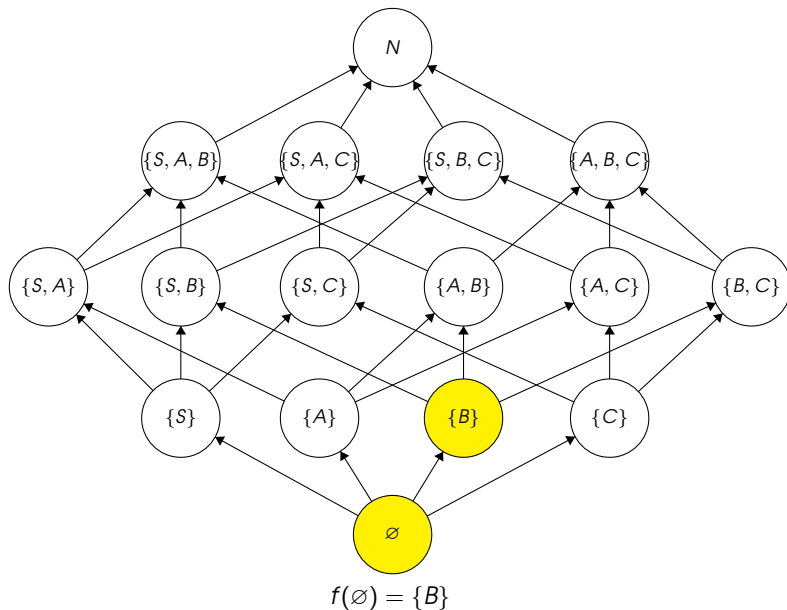
$$N_t = \{A, B, C, S\}$$

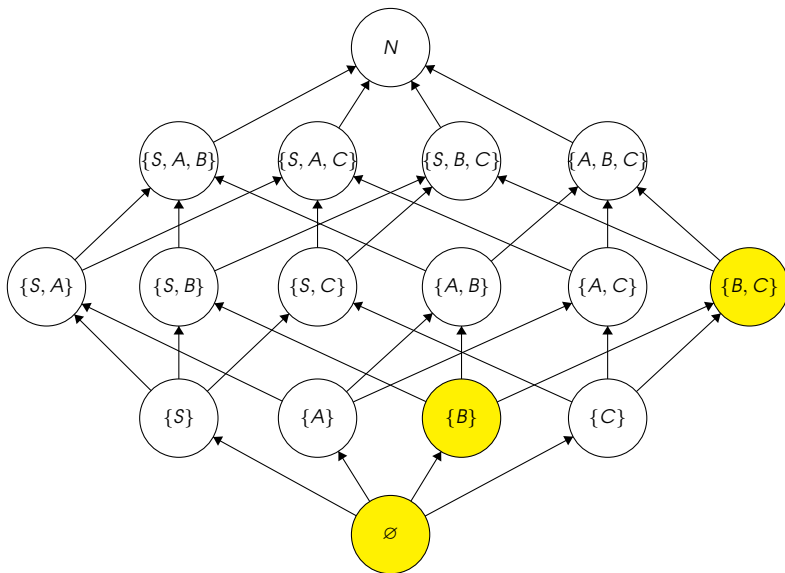
Demonstrace

$$\bullet N_{wa}^4 = \{B, C, S\} = N_{wa}^3 = N_{wa}$$

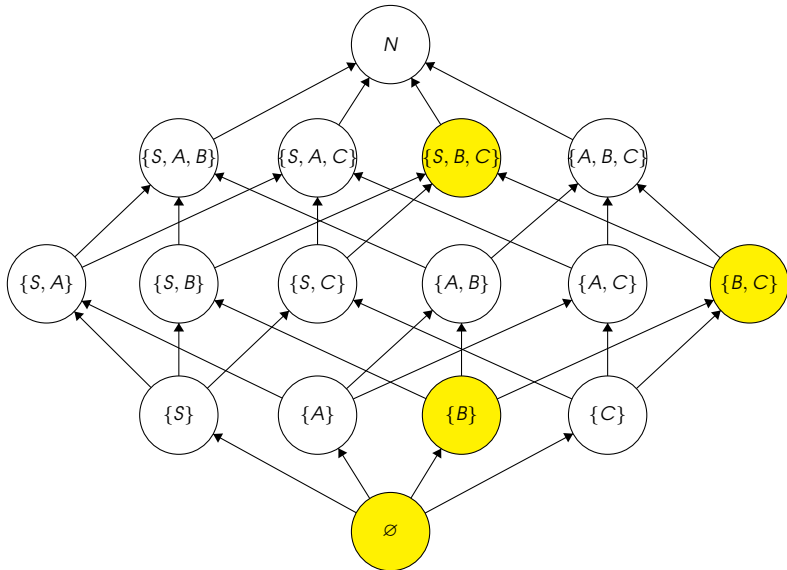




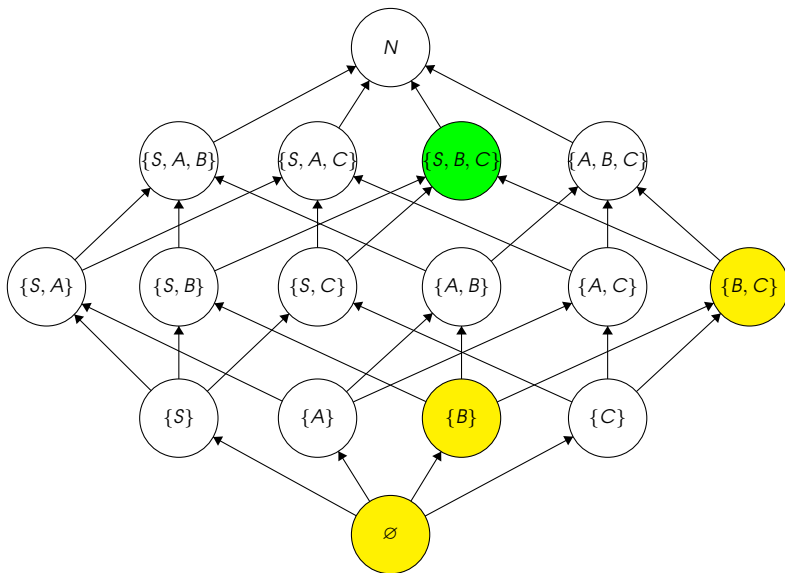




$$f^2(\emptyset) = f(\{B\}) = \{B, C\}$$



$$f^3(\emptyset) = f^2(\{B\}) = f(\{B, C\}) = \{B, C, S\}$$



$$f^4(\emptyset) = f^3(\{B\}) = f^2(\{B, C\}) = f(\{B, C, S\}) = \{B, C, S\}$$

Algoritmy využívající tranzitivní uzávěry relací

Algoritmy využívající tranzitivní uzávěry relací

Sestrojte algoritmus, jenž rozhodne, zda daná bezkontextová gramatika $G = (N, \Sigma, P, S)$ obsahuje cyklus

Algoritmy využívající tranzitivní uzávěry relací

Sestrojte algoritmus, jenž rozhodne, zda daná bezkontextová gramatika $G = (N, \Sigma, P, S)$ obsahuje cyklus

Cyklus

- $G = (N, \Sigma, P, S), A \in N$
- A má **cyklus** v G , pokud $A \Rightarrow_G^+ A$

Algoritmy využívající tranzitivní uzávěry relací

Sestrojte algoritmus, jenž rozhodne, zda daná bezkontextová gramatika $G = (N, \Sigma, P, S)$ obsahuje cyklus

Cyklus

- $G = (N, \Sigma, P, S), A \in N$
- A má **cyklus** v G , pokud $A \Rightarrow_G^+ A$

Rekurze

- $G = (N, \Sigma, P, S), A \in N$
- A má **rekurzi** v G , pokud $A \Rightarrow_G^+ \alpha A \beta, \alpha, \beta \in (N \cup \Sigma)^*$

Algoritmy využívající tranzitivní uzávěry relací

Sestrojte algoritmus, jenž rozhodne, zda daná bezkontextová gramatika $G = (N, \Sigma, P, S)$ obsahuje cyklus

Cyklus

- $G = (N, \Sigma, P, S), A \in N$
- A má **cyklus** v G , pokud $A \Rightarrow_G^+ A$

Rekurze

- $G = (N, \Sigma, P, S), A \in N$
- A má **rekurzi** v G , pokud $A \Rightarrow_G^+ \alpha A \beta, \alpha, \beta \in (N \cup \Sigma)^*$

Zdroje cyklu

- jednoduchá pravidla $A \rightarrow A$
- ϵ -pravidla $A \rightarrow AB, B \rightarrow \epsilon$

Algoritmy využívající tranzitivní uzávěry relací

Sestrojte algoritmus, jenž rozhodne, zda daná bezkontextová gramatika $G = (N, \Sigma, P, S)$ obsahuje cyklus

Cyklus

- $G = (N, \Sigma, P, S), A \in N$
- A má **cyklus** v G , pokud $A \Rightarrow_G^+ A$

Rekurze

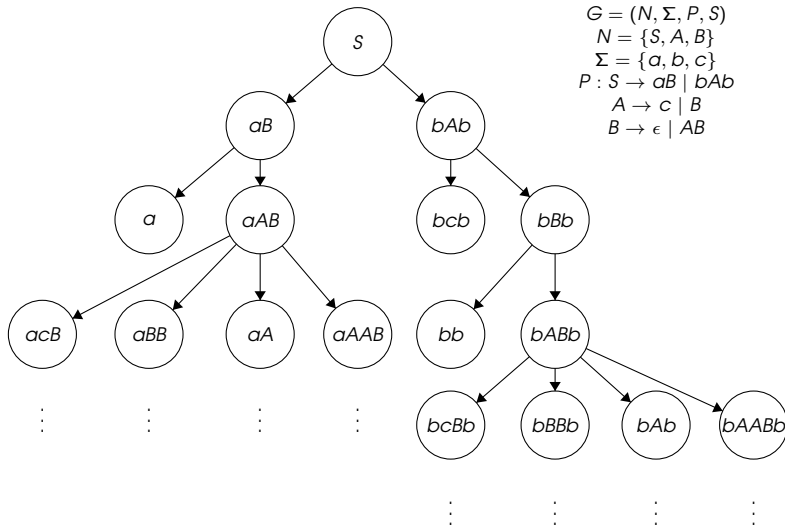
- $G = (N, \Sigma, P, S), A \in N$
- A má **rekurzi** v G , pokud $A \Rightarrow_G^+ \alpha A \beta, \alpha, \beta \in (N \cup \Sigma)^*$

Zdroje cyklu

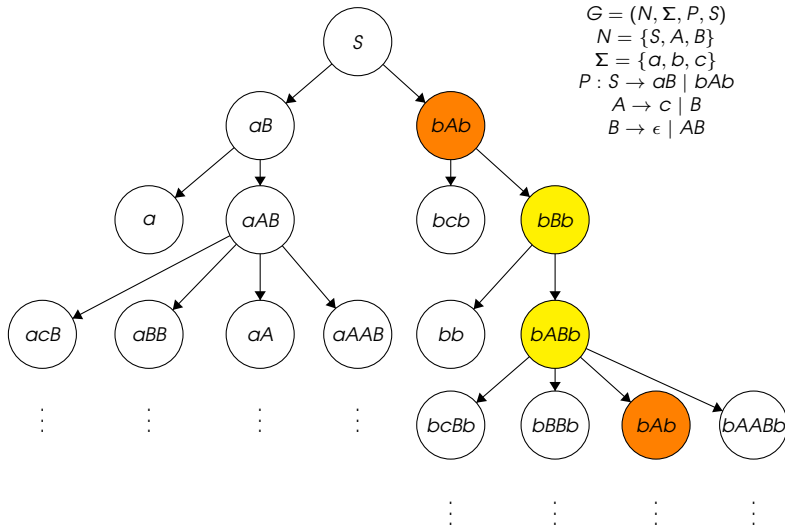
- jednoduchá pravidla $A \rightarrow A$
- ϵ -pravidla $A \rightarrow AB, B \rightarrow \epsilon$

Ukázka cyklu

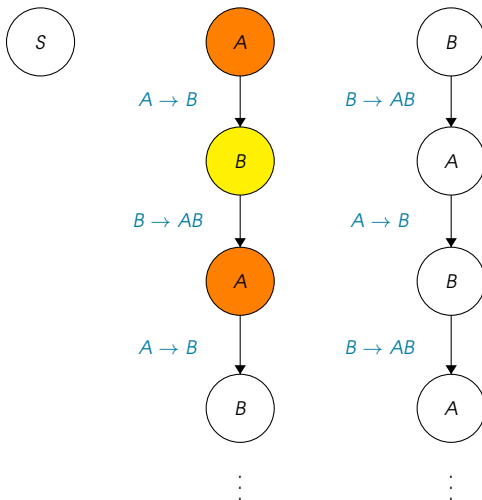
- $G = (\{S, A\}, \{a\}, P, S)$
- $P : S \rightarrow aSb \mid AA$
- $A \rightarrow \epsilon \mid A \mid AA \mid S$
- $A \Rightarrow_G A$
- $A \Rightarrow_G S \Rightarrow_G AA \Rightarrow_G A$



Obsahuje tato gramatika cyklus?



Obsahuje tato gramatika cyklus?



$$\begin{aligned}
 G &= (N, \Sigma, P, S) \\
 N &= \{S, A, B\} \\
 \Sigma &= \{a, b, c\} \\
 P : S &\rightarrow aB \mid bAb \\
 A &\rightarrow c \mid B \\
 B &\rightarrow \epsilon \mid AB
 \end{aligned}$$

Definujeme vhodnou relaci λ na množině neterminálů

Algoritmy využívající tranzitivní uzávěry relací

Sestrojte algoritmus, jenž rozhodne, zda daná bezkontextová gramatika $G = (N, \Sigma, P, S)$ obsahuje cyklus

Algoritmy využívající tranzitivní uzávěry relací

Sestrojte algoritmus, jenž rozhodne, zda daná bezkontextová gramatika $G = (N, \Sigma, P, S)$ obsahuje cyklus

Algoritmus

Vstup: bezkontextová gramatika $G = (N, \Sigma, P, S)$

Algoritmy využívající tranzitivní uzávěry relací

Sestrojte algoritmus, jenž rozhodne, zda daná bezkontextová gramatika $G = (N, \Sigma, P, S)$ obsahuje cyklus

Algoritmus

Vstup: bezkontextová gramatika $G = (N, \Sigma, P, S)$

Výstup: ANO, pokud $\exists A \in N : A \Rightarrow^+ A$, jinak NE

Algoritmy využívající tranzitivní uzávěry relací

Sestrojte algoritmus, jenž rozhodne, zda daná bezkontextová gramatika $G = (N, \Sigma, P, S)$ obsahuje cyklus

Algoritmus

Vstup: bezkontextová gramatika $G = (N, \Sigma, P, S)$

Výstup: ANO, pokud $\exists A \in N : A \Rightarrow^+ A$, jinak NE

Postup:

Algoritmy využívající tranzitivní uzávěry relací

Sestrojte algoritmus, jenž rozhodne, zda daná bezkontextová gramatika $G = (N, \Sigma, P, S)$ obsahuje cyklus

Algoritmus

Vstup: bezkontextová gramatika $G = (N, \Sigma, P, S)$

Výstup: ANO, pokud $\exists A \in N : A \Rightarrow^+ A$, jinak NE

Postup:

- 1 Najdi množinu N_ϵ

Algoritmy využívající tranzitivní uzávěry relací

Sestrojte algoritmus, jenž rozhodne, zda daná bezkontextová gramatika $G = (N, \Sigma, P, S)$ obsahuje cyklus

Algoritmus

Vstup: bezkontextová gramatika $G = (N, \Sigma, P, S)$

Výstup: ANO, pokud $\exists A \in N : A \Rightarrow^+ A$, jinak NE

Postup:

- 1 Najdi množinu N_ϵ
- 2 Spočti relaci $\lambda \subseteq N \times N$ definovanou následovně:
$$\forall A, B \in N : A \lambda B \Leftrightarrow \exists (A, \alpha B \beta) \in P : \alpha, \beta \in (N_\epsilon)^*$$

Algoritmy využívající tranzitivní uzávěry relací

Sestrojte algoritmus, jenž rozhodne, zda daná bezkontextová gramatika $G = (N, \Sigma, P, S)$ obsahuje cyklus

Algoritmus

Vstup: bezkontextová gramatika $G = (N, \Sigma, P, S)$

Výstup: ANO, pokud $\exists A \in N : A \Rightarrow^+ A$, jinak NE

Postup:

- 1 Najdi množinu N_ϵ
- 2 Spočti relaci $\lambda \subseteq N \times N$ definovanou následovně:
 $\forall A, B \in N : A \lambda B \Leftrightarrow \exists (A, \alpha B \beta) \in P : \alpha, \beta \in (N_\epsilon)^*$
- 3 Spočti λ^+ (tranzitivní uzávěr relace λ)

Algoritmy využívající tranzitivní uzávěry relací

Sestrojte algoritmus, jenž rozhodne, zda daná bezkontextová gramatika $G = (N, \Sigma, P, S)$ obsahuje cyklus

Algoritmus

Vstup: bezkontextová gramatika $G = (N, \Sigma, P, S)$

Výstup: ANO, pokud $\exists A \in N : A \Rightarrow^+ A$, jinak NE

Postup:

- 1 Najdi množinu N_ϵ
- 2 Spočti relaci $\lambda \subseteq N \times N$ definovanou následovně:
$$\forall A, B \in N : A \lambda B \Leftrightarrow \exists (A, \alpha B \beta) \in P : \alpha, \beta \in (N_\epsilon)^*$$
- 3 Spočti λ^+ (tranzitivní uzávěr relace λ)
- 4 Pokud je λ^+ ireflexivní, vrať NE, jinak vrať ANO

Demonstrace algoritmu

- Demonstrujte algoritmus na BKG $G = (N, \Sigma, P, S)$, kde

① $N = \{S, A, B, C\}$

② $\Sigma = \{a, b, c\}$

③ $P :$

- $S \rightarrow AB \mid Aa$
- $A \rightarrow \epsilon \mid aAb$
- $B \rightarrow C \mid cAc$
- $C \rightarrow AA \mid ASA$

Demonstrace algoritmu

- Demonstrujte algoritmus na BKG $G = (N, \Sigma, P, S)$, kde

① $N = \{S, A, B, C\}$

② $\Sigma = \{a, b, c\}$

③ $P :$

- $S \rightarrow AB \mid Aa$
- $A \rightarrow \epsilon \mid aAb$
- $B \rightarrow C \mid cAc$
- $C \rightarrow AA \mid ASA$

Výpočet

Demonstrace algoritmu

- Demonstrujte algoritmus na BKG $G = (N, \Sigma, P, S)$, kde

- $N = \{S, A, B, C\}$

- $\Sigma = \{a, b, c\}$

- $P :$

- $S \rightarrow AB \mid Aa$
- $A \rightarrow \epsilon \mid aAb$
- $B \rightarrow C \mid cAc$
- $C \rightarrow AA \mid ASA$

- $N_{\epsilon}^0 = \emptyset$
- $N_{\epsilon}^1 = \{A\}$
- $N_{\epsilon}^2 = \{A, C\}$
- $N_{\epsilon}^3 = \{A, C, B\}$
- $N_{\epsilon}^4 = \{A, C, B, S\}$
- $N_{\epsilon}^5 = N_{\epsilon}^4 = N_{\epsilon}$

Výpočet

- $N_{\epsilon} = \{S, A, B, C\}$

Demonstrace algoritmu

- Demonstrujte algoritmus na BKG $G = (N, \Sigma, P, S)$, kde

1 $N = \{S, A, B, C\}$

2 $\Sigma = \{a, b, c\}$

3 $P :$

- $S \rightarrow AB \mid Aa$
- $A \rightarrow \epsilon \mid aAb$
- $B \rightarrow C \mid cAc$
- $C \rightarrow AA \mid ASA$

A

B

C

S

Výpočet

1 $N_{\epsilon} = \{S, A, B, C\}$

2 $\lambda = \{ \}$

Demonstrace algoritmu

- Demonstrujte algoritmus na BKG $G = (N, \Sigma, P, S)$, kde

① $N = \{S, A, B, C\}$

② $\Sigma = \{a, b, c\}$

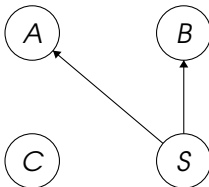
③ P :

- $S \rightarrow AB \mid Aa$
- $A \rightarrow \epsilon \mid aAb$
- $B \rightarrow C \mid cAc$
- $C \rightarrow AA \mid ASA$

Výpočet

① $N_\epsilon = \{S, A, B, C\}$

② $\lambda = \{(S, A), (S, B)\}$



Demonstrace algoritmu

- Demonstrujte algoritmus na BKG $G = (N, \Sigma, P, S)$, kde

1 $N = \{S, A, B, C\}$

2 $\Sigma = \{a, b, c\}$

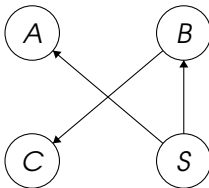
3 $P :$

- $S \rightarrow AB \mid Aa$
- $A \rightarrow \epsilon \mid aAb$
- $B \rightarrow C \mid cAc$
- $C \rightarrow AA \mid ASA$

Výpočet

1 $N_{\epsilon} = \{S, A, B, C\}$

2 $\lambda = \{(S, A), (S, B), (B, C)\}$



Demonstrace algoritmu

- Demonstrujte algoritmus na BKG $G = (N, \Sigma, P, S)$, kde

① $N = \{S, A, B, C\}$

② $\Sigma = \{a, b, c\}$

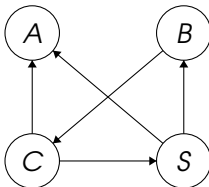
③ P :

- $S \rightarrow AB \mid Aa$
- $A \rightarrow \epsilon \mid aAb$
- $B \rightarrow C \mid cAc$
- $C \rightarrow AA \mid ASA$

Výpočet

① $N_\epsilon = \{S, A, B, C\}$

② $\lambda = \{(S, A), (S, B), (B, C), (C, A), (C, S)\}$



Demonstrace algoritmu

- Demonstrujte algoritmus na BKG $G = (N, \Sigma, P, S)$, kde

1 $N = \{S, A, B, C\}$

2 $\Sigma = \{a, b, c\}$

3 $P :$

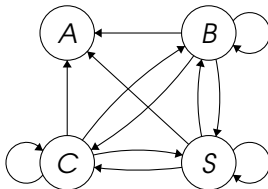
- $S \rightarrow AB \mid Aa$
- $A \rightarrow \epsilon \mid aAb$
- $B \rightarrow C \mid cAc$
- $C \rightarrow AA \mid ASA$

Výpočet

1 $N_\epsilon = \{S, A, B, C\}$

2 $\lambda = \{(S, A), (S, B), (B, C), (C, A), (C, S)\}$

3 $\lambda^+ = \lambda \cup \{(B, S), (B, B), (B, A), (S, C), (S, S), (C, B), (C, C)\}$



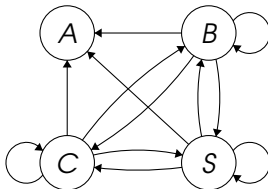
Demonstrace algoritmu

- Demonstrujte algoritmus na BKG $G = (N, \Sigma, P, S)$, kde

- $N = \{S, A, B, C\}$
- $\Sigma = \{a, b, c\}$
- P :
 - $S \rightarrow AB \mid Aa$
 - $A \rightarrow \epsilon \mid aAb$
 - $B \rightarrow C \mid cAc$
 - $C \rightarrow AA \mid ASA$

Výpočet

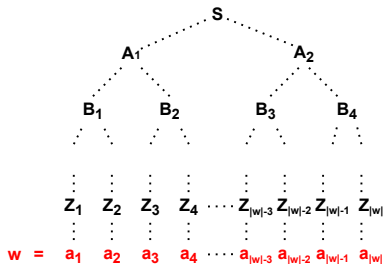
- $N_\epsilon = \{S, A, B, C\}$
- $\lambda = \{(S, A), (S, B), (B, C), (C, A), (C, S)\}$
- $\lambda^+ = \lambda \cup \{(B, S), (B, B), (B, A), (S, C), (S, S), (C, B), (C, C)\}$
- λ^+ není ireflexivní, gramatika G má cyklus v neterminálech B, C, S



Pumping lemma pro bezkontextové jazyky

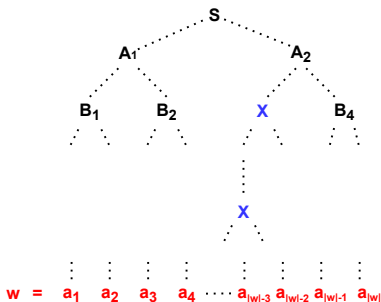
Chomského normální forma

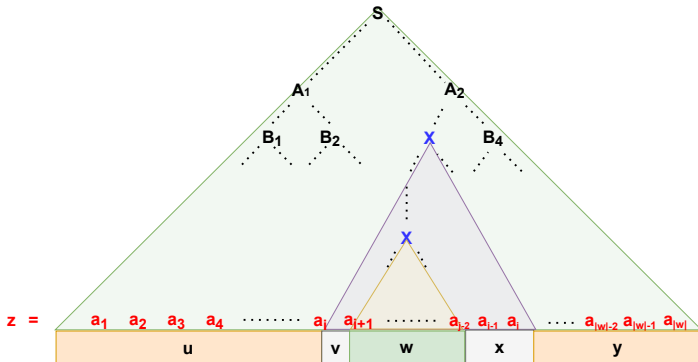
- $G = (N, \Sigma, P, S)$
- G je v Chomského normální formě (CNF), pokud jsou pravidla v P ve tvaru:
 - 1 $A \rightarrow BC, A, B, C \in N$
 - 2 $A \rightarrow a, A \in N, a \in \Sigma$
 - 3 $S \rightarrow \epsilon$, pokud S není na pravé straně žádného pravidla
- pokud má derivační strom pro slovo $w \in L(G)$ výšku $n - 1$, pak $|w| \leq 2^{n-2}$



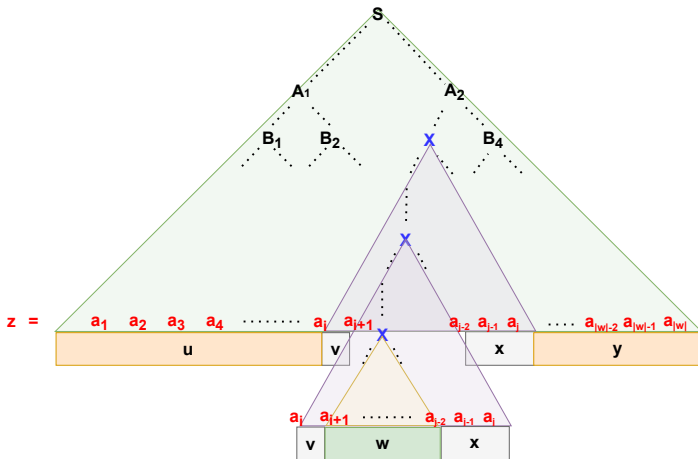
Opakování neterminálu na nejdelší cestě z kořenu k listu

- mějme $G = (N, \Sigma, P, S)$ v CNF , kde $L(G)$ je nekonečný jazyk
- mějme řetězec $w \in L(G)$, $|w| \geq 2^{|N|}$
- $|w| \leq 2^{n-2}$, kde výška derivačního stromu je $n - 1$
- $2^{|N|} \leq |w| \leq 2^{n-2} \Rightarrow |N| + 2 \leq n$
- některý neterminál se na některé cestě od kořene k listu objeví alespoň dvakrát

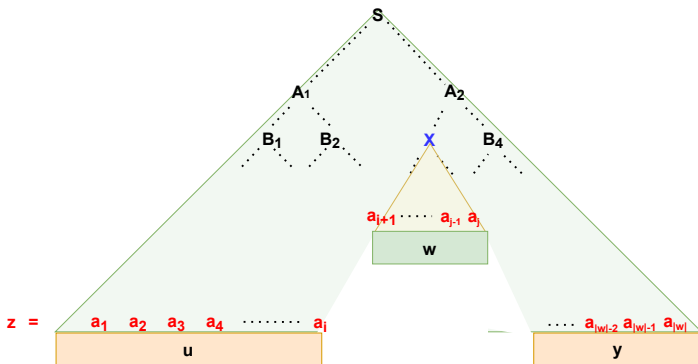




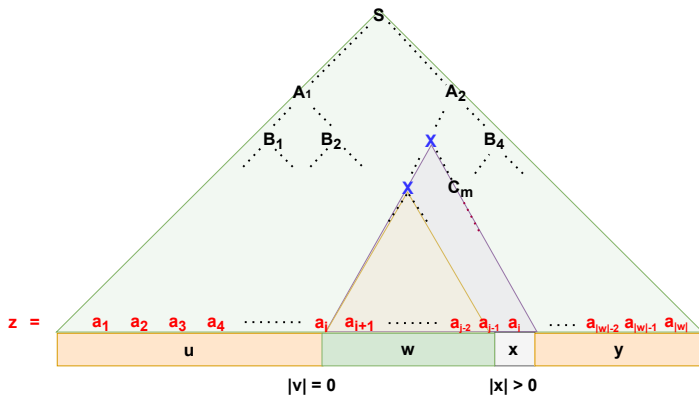
$$S \Rightarrow^+ uXy \Rightarrow^+ uvXxy \Rightarrow^+ uvwxy$$



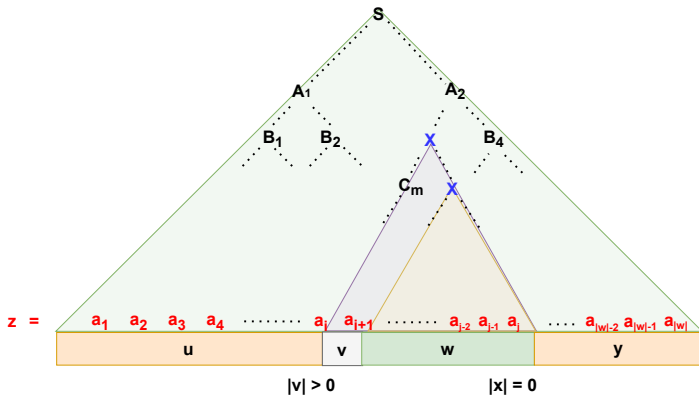
$$S \Rightarrow^+ uXy \Rightarrow^+ uvXxy \Rightarrow^+ uv^2Xx^2y \Rightarrow^+ uv^2wx^2y$$



$$S \Rightarrow^+ uXy \Rightarrow^+ uwy$$



Alespoň jedna z částí v, x bude vždy neprázdná.



Alespoň jedna z částí v, x bude vždy neprázdná.

Znění Pumping lemmatu pro $\mathcal{L}_2$

$$L \in \mathcal{L}_2 \implies (\exists k \in \mathbb{N}^+ : \forall z \in \Sigma^* : z \in L \wedge |z| \geq k \implies \\ (\exists u, v, w, x, y \in \Sigma^* : uvwxy = z \wedge |vx| > 0 \\ \wedge |vwx| \leq k \wedge \forall i \in \mathbb{N} : uv^iwx^iy \in L))$$

Obměněná věta

$$(\varphi \implies \psi) \Leftrightarrow (\neg\psi \implies \neg\varphi)$$

Znění Pumping lemmatu pro $\mathcal{L}_2$ ve tvaru obměněné věty

$$(\forall k \in \mathbb{N}^+ : \exists z \in \Sigma^* : z \in L \wedge |z| \geq k \wedge \\ (\forall u, v, w, x, y \in \Sigma^* : uvwxy = z \wedge \\ |vx| > 0 \wedge |vwx| \leq k \implies \exists i \in \mathbb{N} : \\ uv^iwx^iy \notin L)) \implies L \notin \mathcal{L}_2$$

Pumping lemma pro bezkontextové jazyky

Dokažte, že $L = \{w \in \{a, b, c\}^* \mid \#_a(w) > \#_b(w) > \#_c(w)\} \notin \mathcal{L}_2$

Pumping lemma pro bezkontextové jazyky

Dokažte, že $L = \{w \in \{a, b, c\}^* \mid \#_a(w) > \#_b(w) > \#_c(w)\} \notin \mathcal{L}_2$

Postup (neformálně)

- 1 Uvažujeme o každém kladném přirozeném čísle k

Pumping lemma pro bezkontextové jazyky

Dokažte, že $L = \{w \in \{a, b, c\}^* \mid \#_a(w) > \#_b(w) > \#_c(w)\} \notin \mathcal{L}_2$

Postup (neformálně)

- 1 Uvažujeme o každém kladném přirozeném čísle k
- 2 Zvolíme slovo $z \in \{a, b, c\}^*$ parametricky závislé na k tak, aby $|z| \geq k$ a $z \in L$

Pumping lemma pro bezkontextové jazyky

Dokažte, že $L = \{w \in \{a, b, c\}^* \mid \#_a(w) > \#_b(w) > \#_c(w)\} \notin \mathcal{L}_2$

Postup (neformálně)

- 1 Uvažujeme o každém kladném přirozeném čísle k
- 2 Zvolíme slovo $z \in \{a, b, c\}^*$ parametricky závislé na k tak, aby $|z| \geq k$ a $z \in L$
- 3 Uvažujeme o všech možných rozděleních slova z na pět částí u, v, w, x, y , kde $uvwxy = z$, $|vx| > 0$ a $|vwx| \leq k$

Pumping lemma pro bezkontextové jazyky

Dokažte, že $L = \{w \in \{a, b, c\}^* \mid \#_a(w) > \#_b(w) > \#_c(w)\} \notin \mathcal{L}_2$

Postup (neformálně)

- 1 Uvažujeme o každém kladném přirozeném čísle k
- 2 Zvolíme slovo $z \in \{a, b, c\}^*$ parametricky závislé na k tak, aby $|z| \geq k$ a $z \in L$
- 3 Uvažujeme o všech možných rozděleních slova z na pět částí u, v, w, x, y , kde $uvwxy = z$, $|vx| > 0$ a $|vwx| \leq k$
- 4 Pro každé takové rozdělení najdeme přirozené číslo i tak, aby $uv^iwx^iy \notin L$

Pumping lemma pro bezkontextové jazyky

Dokažte, že $L = \{w \in \{a, b, c\}^* \mid \#_a(w) > \#_b(w) > \#_c(w)\} \notin \mathcal{L}_2$

Postup (neformálně)

- 1 Uvažujeme o každém kladném přirozeném čísle k
- 2 Zvolíme slovo $z \in \{a, b, c\}^*$ parametricky závislé na k tak, aby $|z| \geq k$ a $z \in L$
- 3 Uvažujeme o všech možných rozděleních slova z na pět částí u, v, w, x, y , kde $uvwxy = z$, $|vx| > 0$ a $|vwx| \leq k$
- 4 Pro každé takové rozdělení najdeme přirozené číslo i tak, aby $uv^iwx^iy \notin L$
- 5 Pokud se nám to povede, ukážeme tím, že $L \notin \mathcal{L}_2$

Pumping lemma pro bezkontextové jazyky

Dokažte, že $L = \{w \in \{a, b, c\}^* \mid \#_a(w) > \#_b(w) > \#_c(w)\} \notin \mathcal{L}_2$

Pumping lemma pro bezkontextové jazyky

Dokažte, že $L = \{w \in \{a, b, c\}^* \mid \#_a(w) > \#_b(w) > \#_c(w)\} \notin \mathcal{L}_2$

Volba a rozdělení slova

zvolíme slovo $z = a^{k+2}b^{k+1}c^k$

Pumping lemma pro bezkontextové jazyky

Dokažte, že $L = \{w \in \{a, b, c\}^* \mid \#_a(w) > \#_b(w) > \#_c(w)\} \notin \mathcal{L}_2$

Volba a rozdělení slova

zvolíme slovo $z = a^{k+2}b^{k+1}c^k$

a^{k+2}	b^{k+1}	c^k
-----------	-----------	-------

u	v	w	x	y
---	---	---	---	---

Různých typů rozdělení je příliš mnoho.

Pumping lemma pro bezkontextové jazyky

Dokažte, že $L = \{w \in \{a, b, c\}^* \mid \#_a(w) > \#_b(w) > \#_c(w)\} \notin \mathcal{L}_2$

Volba a rozdělení slova

zvolíme slovo $z = a^{k+2}b^{k+1}c^k$

a^{k+2}	b^{k+1}	c^k
-----------	-----------	-------

u	v	w	x	y
---	---	---	---	---

Různých typů rozdělení je příliš mnoho.

Pumping lemma pro bezkontextové jazyky

Dokažte, že $L = \{w \in \{a, b, c\}^* \mid \#_a(w) > \#_b(w) > \#_c(w)\} \notin \mathcal{L}_2$

Volba a rozdělení slova

zvolíme slovo $z = a^{k+2}b^{k+1}c^k$

a^{k+2}	b^{k+1}	c^k
-----------	-----------	-------

u	v	w	x	y
---	---	---	---	---

Různých typů rozdělení je příliš mnoho.

Pumping lemma pro bezkontextové jazyky

Dokažte, že $L = \{w \in \{a, b, c\}^* \mid \#_a(w) > \#_b(w) > \#_c(w)\} \notin \mathcal{L}_2$

Volba a rozdělení slova

zvolíme slovo $z = a^{k+2}b^{k+1}c^k$

a^{k+2}	b^{k+1}	c^k
-----------	-----------	-------

u	v	w	x	y
---	---	---	---	---

Různých typů rozdělení je příliš mnoho.

Pumping lemma pro bezkontextové jazyky

Dokažte, že $L = \{w \in \{a, b, c\}^* \mid \#_a(w) > \#_b(w) > \#_c(w)\} \notin \mathcal{L}_2$

Volba a rozdělení slova

zvolíme slovo $z = a^{k+2}b^{k+1}c^k$

a^{k+2}	b^{k+1}	c^k
-----------	-----------	-------

u	v	w	x	y
---	---	---	---	---

Různých typů rozdělení je příliš mnoho.

Pumping lemma pro bezkontextové jazyky

Dokažte, že $L = \{w \in \{a, b, c\}^* \mid \#_a(w) > \#_b(w) > \#_c(w)\} \notin \mathcal{L}_2$

Volba a rozdělení slova

zvolíme slovo $z = a^{k+2}b^{k+1}c^k$

a^{k+2}	b^{k+1}	c^k
-----------	-----------	-------

u	v	w	x	y
---	---	---	---	---

Různých typů rozdělení je příliš mnoho.

Pumping lemma pro bezkontextové jazyky

Dokažte, že $L = \{w \in \{a, b, c\}^* \mid \#_a(w) > \#_b(w) > \#_c(w)\} \notin \mathcal{L}_2$

Volba a rozdělení slova

zvolíme slovo $z = a^{k+2}b^{k+1}c^k$

a^{k+2}	b^{k+1}	c^k
-----------	-----------	-------

u	v	w	x	y
---	---	---	---	---

Různých typů rozdělení je příliš mnoho.

Pumping lemma pro bezkontextové jazyky

Dokažte, že $L = \{w \in \{a, b, c\}^* \mid \#_a(w) > \#_b(w) > \#_c(w)\} \notin \mathcal{L}_2$

Volba a rozdělení slova

zvolíme slovo $z = a^{k+2}b^{k+1}c^k$

a^{k+2}	b^{k+1}	c^k
-----------	-----------	-------

u	v	w	x	y
-----	-----	-----	-----	-----

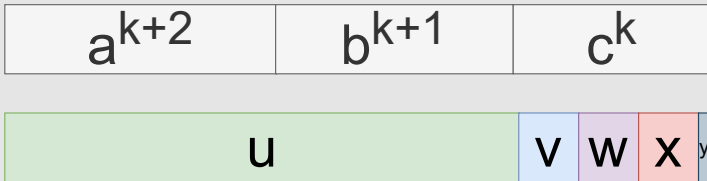
Různých typů rozdělení je příliš mnoho.

Pumping lemma pro bezkontextové jazyky

Dokažte, že $L = \{w \in \{a, b, c\}^* \mid \#_a(w) > \#_b(w) > \#_c(w)\} \notin \mathcal{L}_2$

Volba a rozdělení slova

zvolíme slovo $z = a^{k+2}b^{k+1}c^k$



Různých typů rozdělení je příliš mnoho.

Pumping lemma pro bezkontextové jazyky

Dokažte, že $L = \{w \in \{a, b, c\}^* \mid \#_a(w) > \#_b(w) > \#_c(w)\} \notin \mathcal{L}_2$

Řešení

- 1 Uvažujeme o libovolném kladném přirozeném k

Pumping lemma pro bezkontextové jazyky

Dokažte, že $L = \{w \in \{a, b, c\}^* \mid \#_a(w) > \#_b(w) > \#_c(w)\} \notin \mathcal{L}_2$

Řešení

- 1 Uvažujeme o libovolném kladném přirozeném k
- 2 Zvolíme pro každé takové k slovo $z = a^{k+2}b^{k+1}c^k$

Pumping lemma pro bezkontextové jazyky

Dokažte, že $L = \{w \in \{a, b, c\}^* \mid \#_a(w) > \#_b(w) > \#_c(w)\} \notin \mathcal{L}_2$

Řešení

- 1 Uvažujeme o libovolném kladném přirozeném k
- 2 Zvolíme pro každé takové k slovo $z = a^{k+2}b^{k+1}c^k$
- 3 Určitě platí, že $z \in L$ a $|z| \geq k$
- 4 Pro každé takové slovo uvažujme o všech možných rozděleních na 5 částí u, v, w, x, y , kde $uvwxy = z$, $|vx| > 0$ a $|vwx| \leq k$

Pumping lemma pro bezkontextové jazyky

Dokažte, že $L = \{w \in \{a, b, c\}^* \mid \#_a(w) > \#_b(w) > \#_c(w)\} \notin \mathcal{L}_2$

Řešení

- 1 Uvažujeme o libovolném kladném přirozeném k
- 2 Zvolíme pro každé takové k slovo $z = a^{k+2}b^{k+1}c^k$
- 3 Určitě platí, že $z \in L$ a $|z| \geq k$
- 4 Pro každé takové slovo uvažujme o všech možných rozděleních na 5 částí u, v, w, x, y , kde $uvwxy = z$, $|vx| > 0$ a $|vwx| \leq k$
- 5 Taková rozdělení můžeme roztrždit do tří skupin:

Pumping lemma pro bezkontextové jazyky

Dokažte, že $L = \{w \in \{a, b, c\}^* \mid \#_a(w) > \#_b(w) > \#_c(w)\} \notin \mathcal{L}_2$

Řešení

- 1 Uvažujeme o libovolném kladném přirozeném k
- 2 Zvolíme pro každé takové k slovo $z = a^{k+2}b^{k+1}c^k$
- 3 Určitě platí, že $z \in L$ a $|z| \geq k$
- 4 Pro každé takové slovo uvažujeme o všech možných rozděleních na 5 částí u, v, w, x, y , kde $uvwxy = z$, $|vx| > 0$ a $|vwx| \leq k$
- 5 Taková rozdělení můžeme roztřídit do tří skupin:
 - vx obsahuje pouze symboly a
 - vx obsahuje alespoň jeden symbol b a neobsahuje symbol c
 - vx obsahuje alespoň jeden symbol c

Pumping lemma pro bezkontextové jazyky

Dokažte, že $L = \{w \in \{a, b, c\}^* \mid \#_a(w) > \#_b(w) > \#_c(w)\} \notin \mathcal{L}_2$

Řešení

- 6 vx obsahuje pouze symboly a

Pumping lemma pro bezkontextové jazyky

Dokažte, že $L = \{w \in \{a, b, c\}^* \mid \#_a(w) > \#_b(w) > \#_c(w)\} \notin \mathcal{L}_2$

Řešení

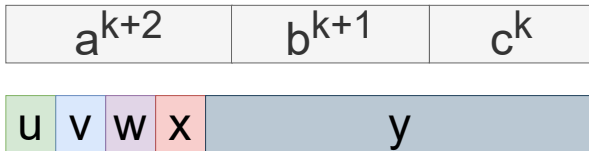
- 6 vx obsahuje pouze symboly a
- zvolíme $i = 0$
 - ve slově $uv^iwx^iy = uv^0wx^0y$ dojde k narušení podmínky $\#_a(w) > \#_b(w)$
 - pracujeme totiž s řetězcem $z = a^{k+2}b^{k+1}c^k$ a $|vx| > 0$

Pumping lemma pro bezkontextové jazyky

Dokažte, že $L = \{w \in \{a, b, c\}^* \mid \#_a(w) > \#_b(w) > \#_c(w)\} \notin \mathcal{L}_2$

Řešení

- 6 vx obsahuje pouze symboly a
- zvolíme $i = 0$
 - ve slově $uv^iwx^iy = uv^0wx^0y$ dojde k narušení podmínky $\#_a(w) > \#_b(w)$
 - pracujeme totiž s řetězcem $z = a^{k+2}b^{k+1}c^k$ a $|vx| > 0$



Pumping lemma pro bezkontextové jazyky

Dokažte, že $L = \{w \in \{a, b, c\}^* \mid \#_a(w) > \#_b(w) > \#_c(w)\} \notin \mathcal{L}_2$

Řešení

- 7 vx obsahuje alespoň jeden symbol b a neobsahuje c

Pumping lemma pro bezkontextové jazyky

Dokažte, že $L = \{w \in \{a, b, c\}^* \mid \#_a(w) > \#_b(w) > \#_c(w)\} \notin \mathcal{L}_2$

Řešení

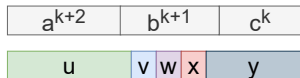
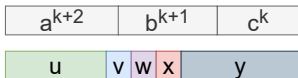
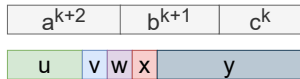
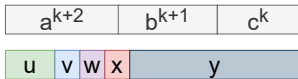
- 7 vx obsahuje alespoň jeden symbol b a neobsahuje c
- zvolíme $i = 0$
 - ve slově $uv^iwx^iy = uv^0wx^0y$ dojde k narušení podmínky $\#_b(w) > \#_c(w)$
 - pracujeme totiž s řetězcem $z = a^{k+2}b^{k+1}c^k$ a $|vx| > 0$

Pumping lemma pro bezkontextové jazyky

Dokažte, že $L = \{w \in \{a, b, c\}^* \mid \#_a(w) > \#_b(w) > \#_c(w)\} \notin \mathcal{L}_2$

Řešení

- 7 vx obsahuje alespoň jeden symbol b a neobsahuje c
- zvolíme $i = 0$
 - ve slově $uv^iwx^iy = uv^0wx^0y$ dojde k narušení podmínky $\#_b(w) > \#_c(w)$
 - pracujeme totiž s řetězcem $z = a^{k+2}b^{k+1}c^k$ a $|vx| > 0$



Pumping lemma pro bezkontextové jazyky

Dokažte, že $L = \{w \in \{a, b, c\}^* \mid \#_a(w) > \#_b(w) > \#_c(w)\} \notin \mathcal{L}_2$

Řešení

- 8 vx obsahuje alespoň jeden symbol c

Pumping lemma pro bezkontextové jazyky

Dokažte, že $L = \{w \in \{a, b, c\}^* \mid \#_a(w) > \#_b(w) > \#_c(w)\} \notin \mathcal{L}_2$

Řešení

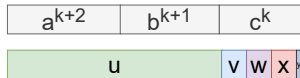
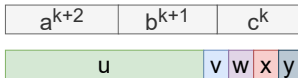
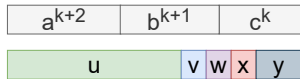
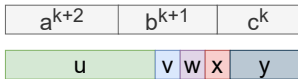
- 8 vx obsahuje alespoň jeden symbol c
- zvolíme $i = 3$
 - ve slově $uv^iwx^iy = uv^3wx^3y$ dojde k narušení podmínky $\#_a(w) > \#_c(w)$
 - pracujeme totiž s řetězcem $z = a^{k+2}b^{k+1}c^k$ a $|vx| > 0$

Pumping lemma pro bezkontextové jazyky

Dokažte, že $L = \{w \in \{a, b, c\}^* \mid \#_a(w) > \#_b(w) > \#_c(w)\} \notin \mathcal{L}_2$

Řešení

- 8 vx obsahuje alespoň jeden symbol c
- zvolíme $i = 3$
 - ve slově $uv^iwx^iy = uv^3wx^3y$ dojde k narušení podmínky $\#_a(w) > \#_c(w)$
 - pracujeme totiž s řetězcem $z = a^{k+2}b^{k+1}c^k$ a $|vx| > 0$



Pumping lemma pro bezkontextové jazyky

Dokažte, že $L = \{w \in \{a, b, c\}^* \mid \#_a(w) > \#_b(w) > \#_c(w)\} \notin \mathcal{L}_2$

Řešení

- 9 ukázali jsme, že pro každé uvažované rozdělení lze najít $i \in \mathbb{N}$ tak, že $uv^iwx^iy \notin L$

Pumping lemma pro bezkontextové jazyky

Dokažte, že $L = \{w \in \{a, b, c\}^* \mid \#_a(w) > \#_b(w) > \#_c(w)\} \notin \mathcal{L}_2$

Řešení

- 9 ukázali jsme, že pro každé uvažované rozdělení lze najít $i \in \mathbb{N}$ tak, že $uv^iwx^iy \notin L$
- 10 Pro jazyk L tedy neplatí pravá strana Pumping lemmatu pro $\mathcal{L}_2$

Pumping lemma pro bezkontextové jazyky

Dokažte, že $L = \{w \in \{a, b, c\}^* \mid \#_a(w) > \#_b(w) > \#_c(w)\} \notin \mathcal{L}_2$

Řešení

- 9 ukázali jsme, že pro každé uvažované rozdělení lze najít $i \in \mathbb{N}$ tak, že $uv^iwx^iy \notin L$
- 10 Pro jazyk L tedy neplatí pravá strana Pumping lemmatu pro $\mathcal{L}_2$
- 11 $L \notin \mathcal{L}_2$

Pumping lemma pro bezkontextové jazyky

Dokažte, že $L = \{w \in \{a, b, c\}^* \mid \#_a(w) > \#_b(w) > \#_c(w)\} \notin \mathcal{L}_2$

Příklady špatného výběru slova

1 $z = a^{k+2}b$

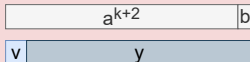
Pumping lemma pro bezkontextové jazyky

Dokažte, že $L = \{w \in \{a, b, c\}^* \mid \#_a(w) > \#_b(w) > \#_c(w)\} \notin \mathcal{L}_2$

Příklady špatného výběru slova

① $z = a^{k+2}b$

- mějme rozdělení $u = \epsilon, v = a, w = \epsilon, x = \epsilon, y = a^{k+1}b$
- pro každé $i \in \mathbb{N}$ platí $uv^iwx^iy = a^i a^{k+1}b \in L$



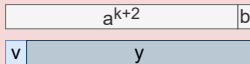
Pumping lemma pro bezkontextové jazyky

Dokažte, že $L = \{w \in \{a, b, c\}^* \mid \#_a(w) > \#_b(w) > \#_c(w)\} \notin \mathcal{L}_2$

Příklady špatného výběru slova

① $z = a^{k+2}b$

- mějme rozdělení $u = \epsilon, v = a, w = \epsilon, x = \epsilon, y = a^{k+1}b$
- pro každé $i \in \mathbb{N}$ platí $uv^iwx^iy = a^i a^{k+1}b \in L$



② $z = aa(abc)^k b$

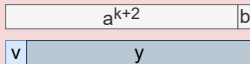
Pumping lemma pro bezkontextové jazyky

Dokažte, že $L = \{w \in \{a, b, c\}^* \mid \#_a(w) > \#_b(w) > \#_c(w)\} \notin \mathcal{L}_2$

Příklady špatného výběru slova

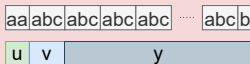
① $z = a^{k+2}b$

- mějme rozdělení $u = \epsilon, v = a, w = \epsilon, x = \epsilon, y = a^{k+1}b$
- pro každé $i \in \mathbb{N}$ platí $uv^iwx^iy = a^i a^{k+1}b \in L$



② $z = aa(abc)^k b$

- mějme rozdělení $u = aa, v = abc, w = \epsilon, x = \epsilon, y = (abc)^{k-1}aab$
- pro každé $i \in \mathbb{N}$ platí $uv^iwx^iy = (abc)^i(abc)^{k-1}b \in L$



Pumping lemma pro bezkontextové jazyky

Dokažte, že $L = \{w \in \{a, b, c\}^* \mid \#_a(w) > \#_b(w) > \#_c(w)\} \notin \mathcal{L}_2$

Příklady špatného výběru slova

3 $z = a^{k+4}b^{k+2}c^k$

Pumping lemma pro bezkontextové jazyky

Dokažte, že $L = \{w \in \{a, b, c\}^* \mid \#_a(w) > \#_b(w) > \#_c(w)\} \notin \mathcal{L}_2$

Příklady špatného výběru slova

③ $z = a^{k+4}b^{k+2}c^k$

- mějme rozdělení $u = \epsilon, v = a, w = \epsilon, x = \epsilon, y = a^{k+3}b^{k+2}c^k$
- pro každé $i \in \mathbb{N}$ platí $uv^iwx^iy = a^i a^{k+3} b^{k+2} c^k \in L$

a	a^{k+3}	b^{k+2}	c^k
---	-----------	-----------	-------

v	y
---	---

Pumping lemma pro bezkontextové jazyky

Dokažte, že $L = \{w \in \{a, b, c\}^* \mid \#_a(w) > \#_b(w) > \#_c(w)\} \notin \mathcal{L}_2$

Příklady špatného výběru slova

③ $z = a^{k+4}b^{k+2}c^k$

- mějme rozdělení $u = \epsilon, v = a, w = \epsilon, x = \epsilon, y = a^{k+3}b^{k+2}c^k$
- pro každé $i \in \mathbb{N}$ platí $uv^iwx^iy = a^i a^{k+3} b^{k+2} c^k \in L$

a	a^{k+3}	b^{k+2}	c^k
---	-----------	-----------	-------

v	y
---	---

④ $z = a^k b^k c^k$

Pumping lemma pro bezkontextové jazyky

Dokažte, že $L = \{w \in \{a, b, c\}^* \mid \#_a(w) > \#_b(w) > \#_c(w)\} \notin \mathcal{L}_2$

Příklady špatného výběru slova

③ $z = a^{k+4}b^{k+2}c^k$

- mějme rozdělení $u = \epsilon, v = a, w = \epsilon, x = \epsilon, y = a^{k+3}b^{k+2}c^k$
- pro každé $i \in \mathbb{N}$ platí $uv^iwx^iy = a^i a^{k+3} b^{k+2} c^k \in L$

a	a^{k+3}	b^{k+2}	c^k
---	-----------	-----------	-------

v	y
---	---

④ $z = a^k b^k c^k$

- vůbec nepatří do jazyka L

Pumping lemma pro bezkontextové jazyky

Dokažte, že $L = \{w \in \{a, b, c\}^* \mid \#_a(w) > \#_b(w) > \#_c(w)\} \notin \mathcal{L}_2$

Příklady špatného výběru slova

③ $z = a^{k+4}b^{k+2}c^k$

- mějme rozdělení $u = \epsilon, v = a, w = \epsilon, x = \epsilon, y = a^{k+3}b^{k+2}c^k$
- pro každé $i \in \mathbb{N}$ platí $uv^iwx^iy = a^i a^{k+3} b^{k+2} c^k \in L$

a	a^{k+3}	b^{k+2}	c^k
---	-----------	-----------	-------

v	y
---	---

④ $z = a^k b^k c^k$

- vůbec nepatří do jazyka L

⑤ $z = a^2 b$

Pumping lemma pro bezkontextové jazyky

Dokažte, že $L = \{w \in \{a, b, c\}^* \mid \#_a(w) > \#_b(w) > \#_c(w)\} \notin \mathcal{L}_2$

Příklady špatného výběru slova

③ $z = a^{k+4}b^{k+2}c^k$

- mějme rozdělení $u = \epsilon, v = a, w = \epsilon, x = \epsilon, y = a^{k+3}b^{k+2}c^k$
- pro každé $i \in \mathbb{N}$ platí $uv^iwx^iy = a^i a^{k+3} b^{k+2} c^k \in L$

a	a^{k+3}	b^{k+2}	c^k
---	-----------	-----------	-------

v	y
---	---

④ $z = a^k b^k c^k$

- vůbec nepatří do jazyka L

⑤ $z = a^2 b$

- pro $k > 3$ neplatí, že $|z| \geq k$

Tvrzení o bezkontextových jazycích

Abeceda

Uvažujme o abecedě $\Sigma = \{a, b, c\}$

Rozhodněte a dokažte, zda platí následující tvrzení

$$L_1 \in \mathcal{L}_3 \vee L_2 \in \mathcal{L}_2 \Rightarrow L_1 \cup L_2 \in \mathcal{L}_2$$

Abeceda

Uvažujme o abecedě $\Sigma = \{a, b, c\}$

Rozhodněte a dokažte, zda platí následující tvrzení

$$L_1 \in \mathcal{L}_3 \vee L_2 \in \mathcal{L}_2 \Rightarrow L_1 \cup L_2 \in \mathcal{L}_2$$

Řešení

- Tvrzení **neplatí**

Abeceda

Uvažujme o abecedě $\Sigma = \{a, b, c\}$

Rozhodněte a dokažte, zda platí následující tvrzení

$$L_1 \in \mathcal{L}_3 \vee L_2 \in \mathcal{L}_2 \Rightarrow L_1 \cup L_2 \in \mathcal{L}_2$$

Řešení

- Tvrzení **neplatí**
- V rámci protipříkladu zvolme:
 - $L_1 = \emptyset$
 - $L_2 = \{a^n b^n c^n \mid n \geq 0\}$

Abeceda

Uvažujme o abecedě $\Sigma = \{a, b, c\}$

Rozhodněte a dokažte, zda platí následující tvrzení

$$L_1 \in \mathcal{L}_3 \vee L_2 \in \mathcal{L}_2 \Rightarrow L_1 \cup L_2 \in \mathcal{L}_2$$

Řešení

- Tvrzení **neplatí**
- V rámci protipříkladu zvolme:
 - $L_1 = \emptyset$
 - $L_2 = \{a^n b^n c^n \mid n \geq 0\}$
- Potom ale $L_1 \cup L_2 = L_2 \notin \mathcal{L}_2$

Abeceda

Uvažujme o abecedě $\Sigma = \{a, b, c\}$

Rozhodněte a dokažte, zda platí následující tvrzení

$$\forall L_1 \in \mathcal{L}_2 : \forall L_2 \in \mathcal{L}_3 : L_1 \subseteq L_2 \Rightarrow \overline{L_1 \cup L_2} \in \mathcal{L}_3$$

Abeceda

Uvažujme o abecedě $\Sigma = \{a, b, c\}$

Rozhodněte a dokažte, zda platí následující tvrzení

$$\forall L_1 \in \mathcal{L}_2 : \forall L_2 \in \mathcal{L}_3 : L_1 \subseteq L_2 \Rightarrow \overline{L_1 \cup L_2} \in \mathcal{L}_3$$

Řešení

- Tvrzení platí

Abeceda

Uvažujme o abecedě $\Sigma = \{a, b, c\}$

Rozhodněte a dokažte, zda platí následující tvrzení

$$\forall L_1 \in \mathcal{L}_2 : \forall L_2 \in \mathcal{L}_3 : L_1 \subseteq L_2 \Rightarrow \overline{L_1 \cup L_2} \in \mathcal{L}_3$$

Řešení

- Tvrzení platí
- Pokud $L_1 \subseteq L_2$, pak $L_1 \cup L_2 = L_2$

Abeceda

Uvažujme o abecedě $\Sigma = \{a, b, c\}$

Rozhodněte a dokažte, zda platí následující tvrzení

$$\forall L_1 \in \mathcal{L}_2 : \forall L_2 \in \mathcal{L}_3 : L_1 \subseteq L_2 \Rightarrow \overline{L_1 \cup L_2} \in \mathcal{L}_3$$

Řešení

- Tvrzení platí
- Pokud $L_1 \subseteq L_2$, pak $L_1 \cup L_2 = L_2$
- Potom $\overline{L_1 \cup L_2} = \overline{L_2} \in \mathcal{L}_3$, neboť $\mathcal{L}_3$ jazyky jsou uzavřeny na doplněk

Abeceda

Uvažujme o abecedě $\Sigma = \{a, b, c\}$

Rozhodněte a dokažte, zda platí následující tvrzení

$$\forall L_1 \in \mathcal{L}_2 : \forall L_2 \in \mathcal{L}_3 : L_1 \subseteq L_2 \Rightarrow \overline{L_1 \cap L_2} \in \mathcal{L}_2$$

Abeceda

Uvažujme o abecedě $\Sigma = \{a, b, c\}$

Rozhodněte a dokažte, zda platí následující tvrzení

$$\forall L_1 \in \mathcal{L}_2 : \forall L_2 \in \mathcal{L}_3 : L_1 \subseteq L_2 \Rightarrow \overline{L_1 \cap L_2} \in \mathcal{L}_2$$

Řešení

- Tvrzení **neplatí**

Abeceda

Uvažujme o abecedě $\Sigma = \{a, b, c\}$

Rozhodněte a dokažte, zda platí následující tvrzení

$$\forall L_1 \in \mathcal{L}_2 : \forall L_2 \in \mathcal{L}_3 : L_1 \subseteq L_2 \Rightarrow \overline{L_1 \cap L_2} \in \mathcal{L}_2$$

Řešení

- Tvrzení **neplatí**
- V rámci protipříkladu zvolme:
 - $L_1 = \{a^n b^n c^n \mid n \geq 0\} \in \mathcal{L}_2$
 - $L_2 = \Sigma^* \in \mathcal{L}_3$

Abeceda

Uvažujme o abecedě $\Sigma = \{a, b, c\}$

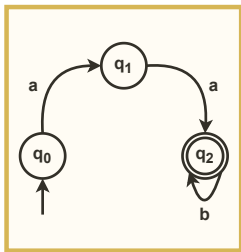
Rozhodněte a dokažte, zda platí následující tvrzení

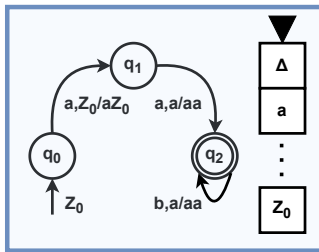
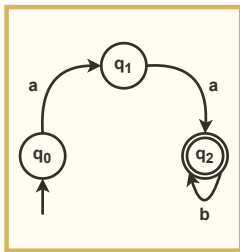
$$\forall L_1 \in \mathcal{L}_2 : \forall L_2 \in \mathcal{L}_3 : L_1 \subseteq L_2 \Rightarrow \overline{L_1 \cap L_2} \in \mathcal{L}_2$$

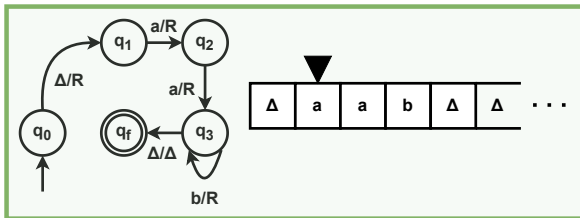
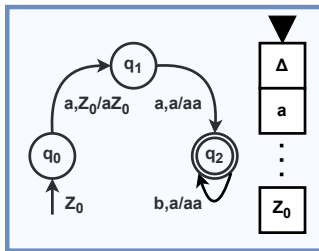
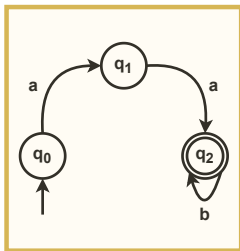
Řešení

- Tvrzení **neplatí**
- V rámci protipříkladu zvolme:
 - $L_1 = \{\overline{a^n b^n c^n} \mid n \geq 0\} \in \mathcal{L}_2$
 - $L_2 = \Sigma^* \in \mathcal{L}_3$
- Potom ale $\overline{L_1 \cap L_2} = \overline{L_1} = \{a^n b^n c^n \mid n \geq 0\} \notin \mathcal{L}_2$

Turingovy stroje

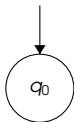






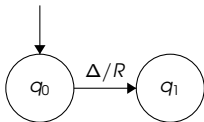
Konstrukce jednopáskového Turingova stroje

Sestrojte jednopáskový TS přijímající jazyk $\{wcw \mid w \in \{a, b\}^*\}$



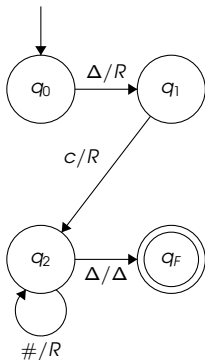
Konstrukce jednopáskového Turingova stroje

Sestrojte jednopáskový TS přijímající jazyk $\{wcw \mid w \in \{a, b\}^*\}$

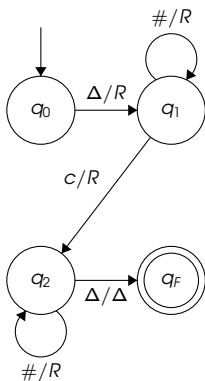


Konstrukce jednopáskového Turingova stroje

Sestrojte jednopáskový TS přijímající jazyk $\{wcw \mid w \in \{a, b\}^*\}$

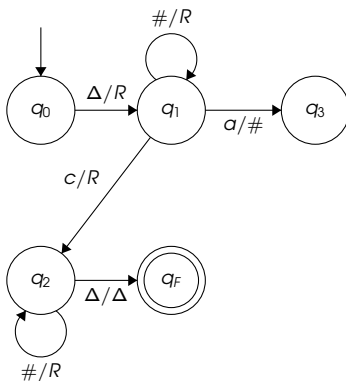


Konstrukce jednopáskového Turingova stroje

Sestrojte jednopáskový TS přijímající jazyk $\{wcw \mid w \in \{a, b\}^*\}$ 

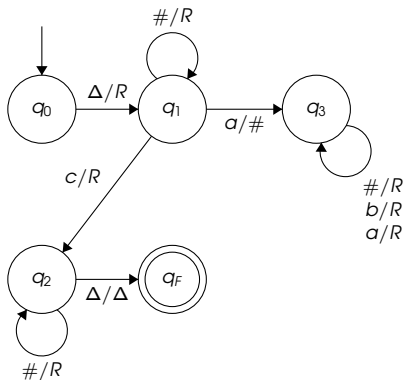
Konstrukce jednopáskového Turingova stroje

Sestrojte jednopáskový TS přijímající jazyk $\{wcw \mid w \in \{a, b\}^*\}$

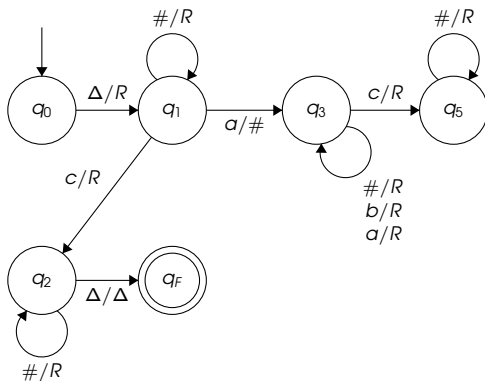


Konstrukce jednopáskového Turingova stroje

Sestrojte jednopáskový TS přijímající jazyk $\{wcw \mid w \in \{a, b\}^*\}$

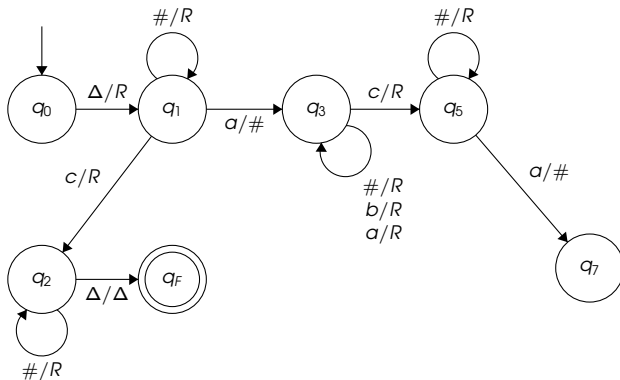


Konstrukce jednopáskového Turingova stroje

Sestrojte jednopáskový TS přijímající jazyk $\{wcw \mid w \in \{a, b\}^*\}$ 

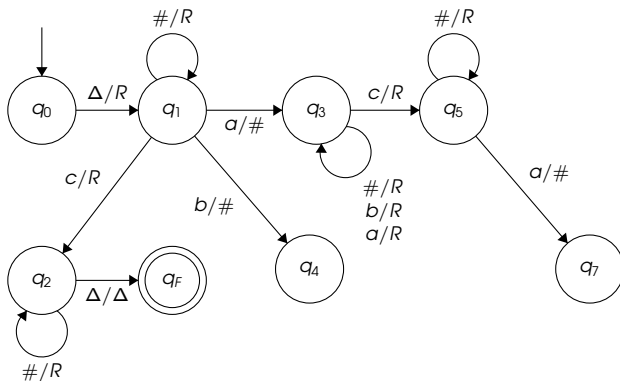
Konstrukce jednopáskového Turingova stroje

Sestrojte jednopáskový TS přijímající jazyk $\{wcw \mid w \in \{a, b\}^*\}$



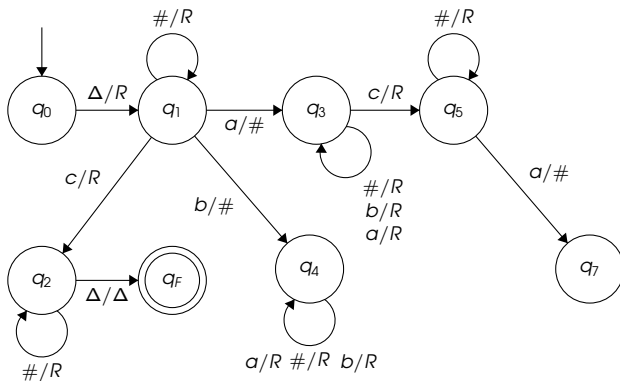
Konstrukce jednopáskového Turingova stroje

Sestrojte jednopáskový TS přijímající jazyk $\{wcw \mid w \in \{a, b\}^*\}$



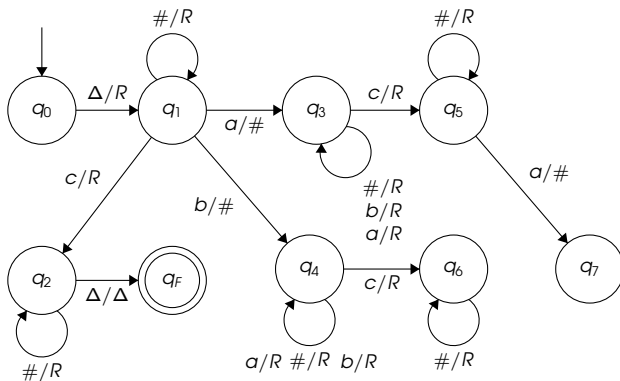
Konstrukce jednopáskového Turingova stroje

Sestrojte jednopáskový TS přijímající jazyk $\{wcw \mid w \in \{a, b\}^*\}$



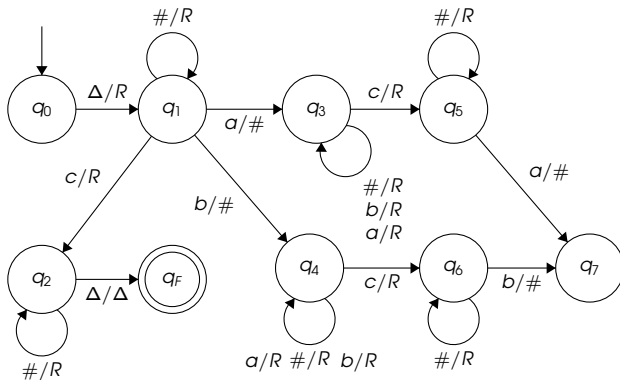
Konstrukce jednopáskového Turingova stroje

Sestrojte jednopáskový TS přijímající jazyk $\{wcw \mid w \in \{a, b\}^*\}$

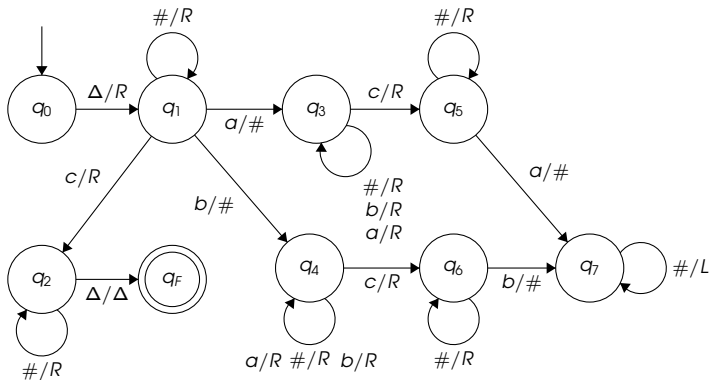


Konstrukce jednopáskového Turingova stroje

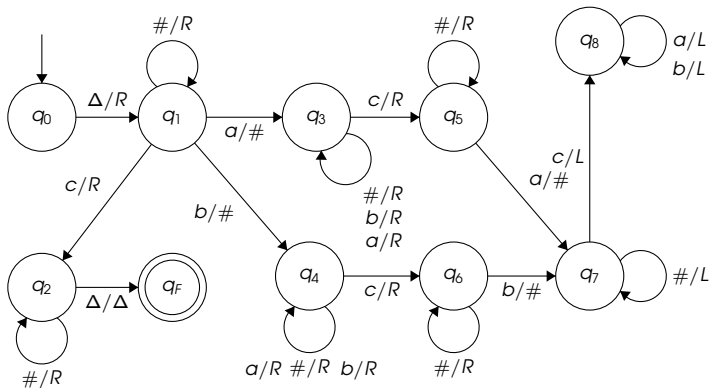
Sestrojte jednopáskový TS přijímající jazyk $\{wcw \mid w \in \{a, b\}^*\}$



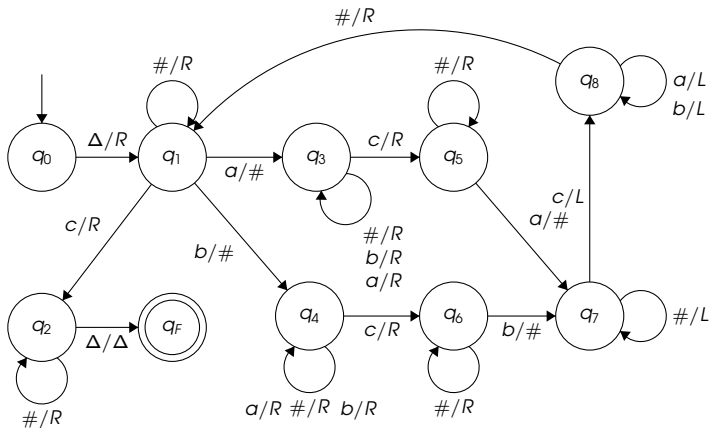
Konstrukce jednopáskového Turingova stroje

Sestrojte jednopáskový TS přijímající jazyk $\{wcw \mid w \in \{a, b\}^*\}$ 

Konstrukce jednopáskového Turingova stroje

Sestrojte jednopáskový TS přijímající jazyk $\{wcw \mid w \in \{a, b\}^*\}$ 

Konstrukce jednopáskového Turingova stroje

Sestrojte jednopáskový TS přijímající jazyk $\{wcw \mid w \in \{a, b\}^*\}$ 

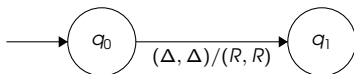
Konstrukce vícepáskového Turingova stroje

Sestrojte vícepáskový **TS** přijímající jazyk $\{wcw \mid w \in \{a, b\}^*\}$



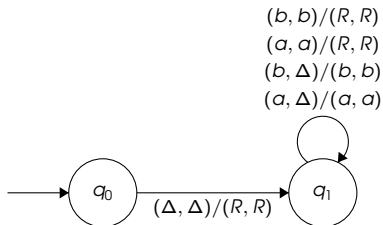
Konstrukce vícepáskového Turingova stroje

Sestrojte vícepáskový **TS** přijímající jazyk $\{wcw \mid w \in \{a, b\}^*\}$



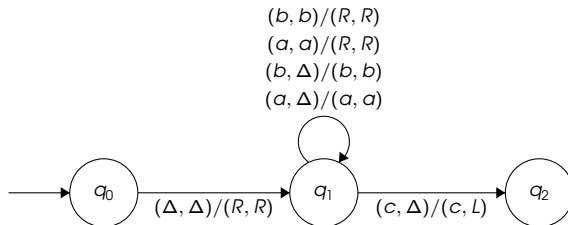
Konstrukce vícepáskového Turingova stroje

Sestrojte vícepáskový **TS** přijímající jazyk $\{wcw \mid w \in \{a, b\}^*\}$



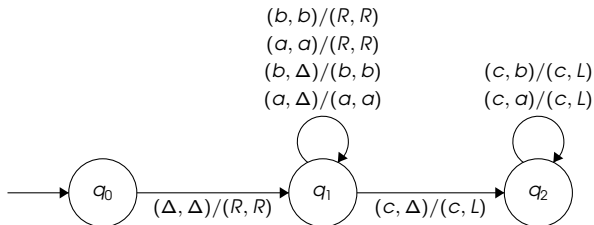
Konstrukce vícepáskového Turingova stroje

Sestrojte vícepáskový **TS** přijímající jazyk $\{wcw \mid w \in \{a, b\}^*\}$

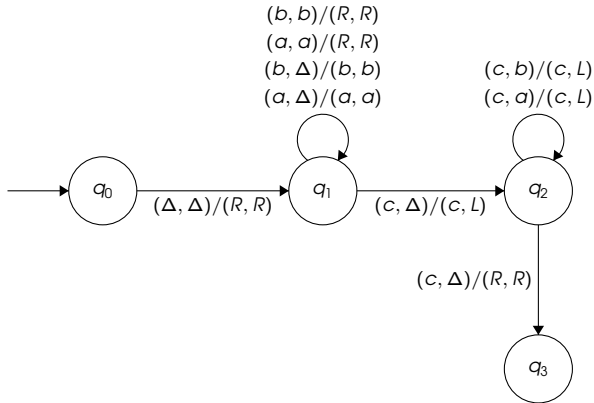


Konstrukce vícepáskového Turingova stroje

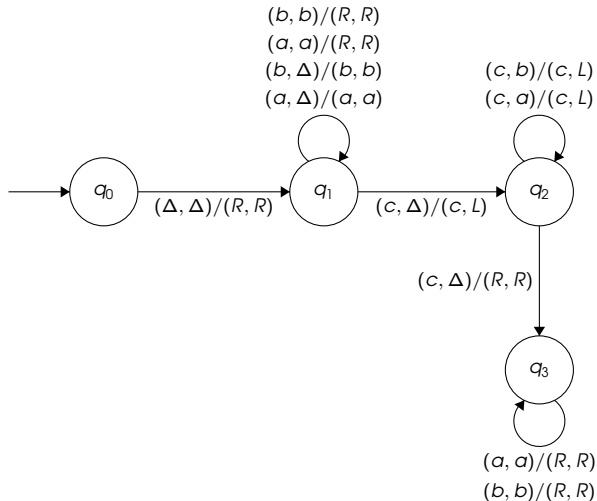
Sestrojte vícepáskový TS přijímající jazyk $\{wcw \mid w \in \{a, b\}^*\}$



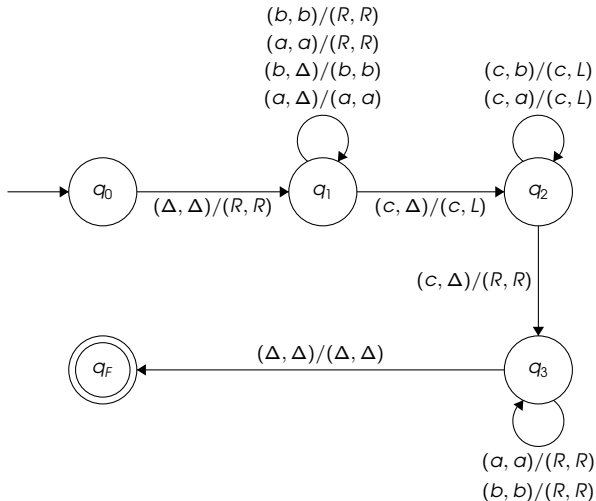
Konstrukce vícepáskového Turingova stroje

Sestrojte vícepáskový TS přijímající jazyk $\{wcw \mid w \in \{a, b\}^*\}$ 

Konstrukce vícepáskového Turingova stroje

Sestrojte vícepáskový TS přijímající jazyk $\{wcw \mid w \in \{a, b\}^*\}$ 

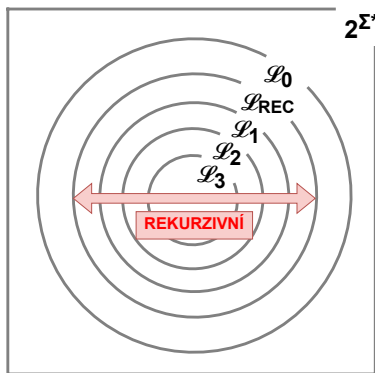
Konstrukce vícepáskového Turingova stroje

Sestrojte vícepáskový TS přijímající jazyk $\{wcw \mid w \in \{a, b\}^*\}$ 

Důkazy rekurzivní vyčísitelnosti

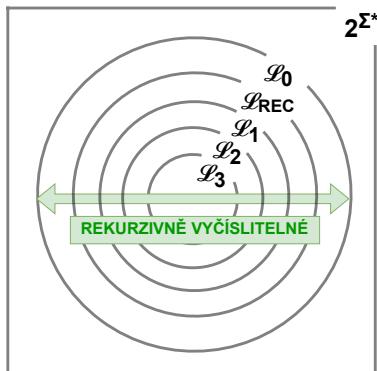
Rekurzivní jazyk

- Jazyk L nad abecedou Σ je **rekurzivní** ($L \in \mathcal{L}_{REC}$) tehdy, pokud existuje **úplný Turingův stroj** M takový, že $L(M) = L$
- Úplný Turingův stroj** je takový Turingův stroj, který **zastaví** pro každý svůj vstup



Rekurzivně vyčíslitelný jazyk

- Jazyk L nad abecedou Σ je rekurzivně vyčíslitelný ($L \in \mathcal{L}_{RE}$) tehdy, pokud existuje (obecný) Turingův stroj M takový, že $L(M) = L$
- Turingův stroj M může na některých vstupech cyklit



Důkaz rekurzivní vyčísitelnosti

- Chceme-li dokázat, že L je **rekurzivně vyčíslitelný**, stačí popsat konstrukci **(obecného) Turingova stroje M** , kde $L = L(M)$

Důkaz rekurzivní vyčísitelnosti

- Chceme-li dokázat, že L je **rekurzivně vyčísitelný**, stačí popsat konstrukci (**obecného**) Turingova stroje M , kde $L = L(M)$
- Stroj M může být
 - deterministický nebo nedeterministický
 - jednopáskový nebo vícepáskový

Důkaz rekurzivní vyčíslitelnosti

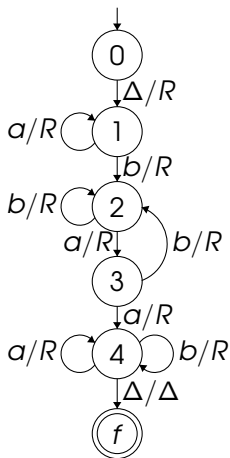
- Chceme-li dokázat, že L je **rekurzivně vyčíslitelný**, stačí popsat konstrukci **(obecného) Turingova stroje M** , kde $L = L(M)$
- Stroj **M** může být
 - deterministický nebo nedeterministický
 - jednopáskový nebo vícepáskový

Vysokoúrovňový popis chování Turingova stroje

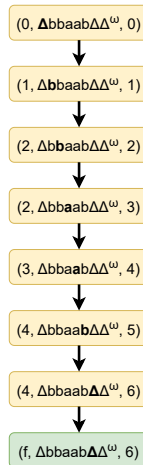
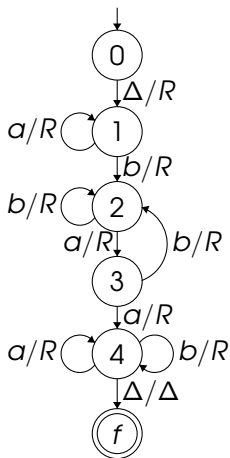
- Nebudeme popisovat chování stroje na úrovni přechodů
- O různých operacích víme, že je možné je v rámci běhu Turingova stroje provést, nebudeme je tedy v důkazu rozvádět do detailů:
 - spuštění/pozastavení simulace jiného Turingova stroje
 - nedeterministické vygenerování nějakého řetězce
 - test, zda je řetězec kódem Turingova stroje
 - ...

$$L = \{x_1baax_2 \mid x_1, x_2 \in \{a, b\}^*\}$$

$$L = \{x_1 baax_2 \mid x_1, x_2 \in \{a, b\}^*\}$$

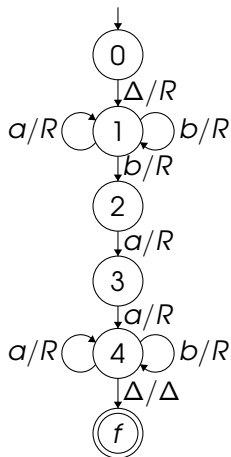


$$L = \{x_1 b a a x_2 \mid x_1, x_2 \in \{a, b\}^*\}$$

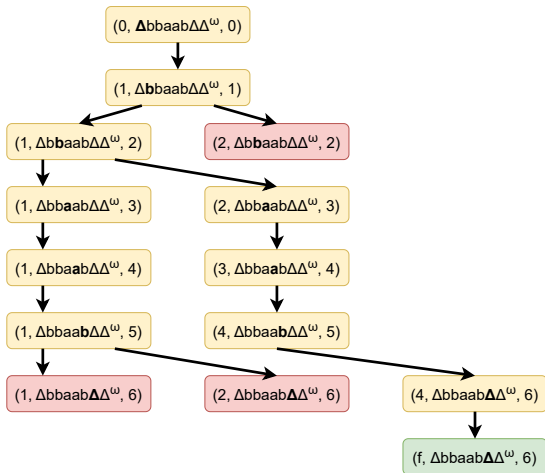
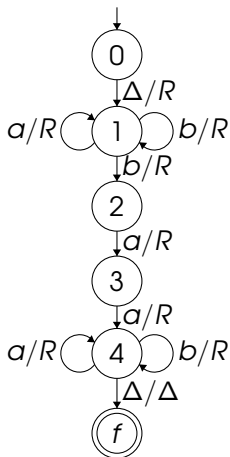


$$L = \{x_1 baax_2 \mid x_1, x_2 \in \{a, b\}^*\}$$

$$L = \{x_1baax_2 \mid x_1, x_2 \in \{a, b\}^*\}$$



$$L = \{x_1baax_2 \mid x_1, x_2 \in \{a, b\}^*\}$$



Důkaz rekurzivní vyčísitelnosti

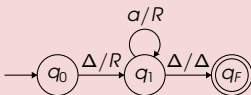
Dokažte, že následující jazyk L je rekurzivně vyčíslitelný:

$$L = \{ \langle M \rangle \mid \langle M \rangle \text{ je kód TS } M, \text{ kde } L(M) \neq \emptyset \}$$

Důkaz rekurzivní vyčísitelnosti

Dokažte, že následující jazyk L je rekurzivně vyčíslitelný:

$$L = \{ \langle M \rangle \mid \langle M \rangle \text{ je kód TS } M, \text{ kde } L(M) \neq \emptyset \}$$

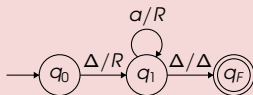
Příklady řetězců patřících do jazyka L 

- $L(M_1) = \{a^n \mid n \geq 0\}$
- $\langle M_1 \rangle \in L$

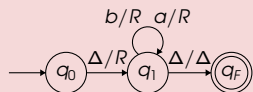
Důkaz rekurzivní vyčíslitelnosti

Dokažte, že následující jazyk L je rekurzivně vyčíslitelný:

$$L = \{ \langle M \rangle \mid \langle M \rangle \text{ je kód TS } M, \text{ kde } L(M) \neq \emptyset \}$$

Příklady řetězců patřících do jazyka L 

- $L(M_1) = \{a^n \mid n \geq 0\}$
- $\langle M_1 \rangle \in L$

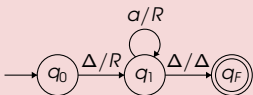


- $L(M_2) = \{a, b\}^*$
- $\langle M_2 \rangle \in L$

Důkaz rekurzivní vyčíslitelnosti

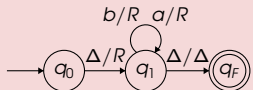
Dokažte, že následující jazyk L je rekurzivně vyčíslitelný:

$$L = \{ \langle M \rangle \mid \langle M \rangle \text{ je kód TS } M, \text{ kde } L(M) \neq \emptyset \}$$

Příklady řetězců patřících do jazyka L 

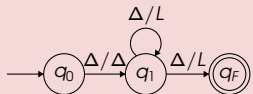
- $L(M_1) = \{a^n \mid n \geq 0\}$

- $\langle M_1 \rangle \in L$



- $L(M_2) = \{a, b\}^*$

- $\langle M_2 \rangle \in L$



- $L(M_3) = \emptyset$

- $\langle M_3 \rangle \notin L$

Důkaz rekurzivní vyčísitelnosti

Dokažte, že následující jazyk L je rekurzivně vyčíslitelný:

$$L = \{ \langle M \rangle \mid \langle M \rangle \text{ je kód TS } M, \text{ kde } L(M) \neq \emptyset \}$$

Důkaz rekurzivní vyčíslitelnosti

Dokažte, že následující jazyk **L** je rekurzivně vyčíslitelný:

$$L = \{ \langle M \rangle \mid \langle M \rangle \text{ je kód TS } M, \text{ kde } L(M) \neq \emptyset \}$$

Simulace nedeterministicky vygenerovaného slova

může skončit akceptováním

		krok simulace							
		1	2	3	4	5	6	7	
w	w ₀								
	w ₁								
	w ₂								
	w ₃	✗							
		⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

Důkaz rekurzivní vyčíslitelnosti

Dokažte, že následující jazyk **L** je rekurzivně vyčíslitelný:

$$L = \{ \langle M \rangle \mid \langle M \rangle \text{ je kód TS } M, \text{ kde } L(M) \neq \emptyset \}$$

Simulace nedeterministicky vygenerovaného slova

může skončit akceptováním

		krok simulace							
		1	2	3	4	5	6	7	
W	w ₀								
	w ₁								
	w ₂								
	w ₃	✗	✗						
		⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

Důkaz rekurzivní vyčíslitelnosti

Dokažte, že následující jazyk **L** je rekurzivně vyčíslitelný:

$$L = \{ \langle M \rangle \mid \langle M \rangle \text{ je kód TS } M, \text{ kde } L(M) \neq \emptyset \}$$

Simulace nedeterministicky vygenerovaného slova

může skončit akceptováním

		krok simulace							
		1	2	3	4	5	6	7	
W	w ₀								
	w ₁								
	w ₂								
	w ₃	✗	✗	✗					
		⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

Důkaz rekurzivní vyčíslitelnosti

Dokažte, že následující jazyk **L** je rekurzivně vyčíslitelný:

$$L = \{ \langle M \rangle \mid \langle M \rangle \text{ je kód TS } M, \text{ kde } L(M) \neq \emptyset \}$$

Simulace nedeterministicky vygenerovaného slova

může skončit akceptováním

		krok simulace							
		1	2	3	4	5	6	7	
W	w ₀								
	w ₁								
	w ₂								
	w ₃	✗	✗	✗	●				
		⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

Důkaz rekurzivní vyčíslitelnosti

Dokažte, že následující jazyk **L** je rekurzivně vyčíslitelný:

$$L = \{ \langle M \rangle \mid \langle M \rangle \text{ je kód TS } M, \text{ kde } L(M) \neq \emptyset \}$$

Simulace nedeterministicky vygenerovaného slova

může skončit odmítnutím

		krok simulace							
		1	2	3	4	5	6	7	
w	w ₀								
	w ₁								
	w ₂	✗							
	w ₃								
		⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

Důkaz rekurzivní vyčíslitelnosti

Dokažte, že následující jazyk **L** je rekurzivně vyčíslitelný:

$$L = \{ \langle M \rangle \mid \langle M \rangle \text{ je kód TS } M, \text{ kde } L(M) \neq \emptyset \}$$

Simulace nedeterministicky vygenerovaného slova

může skončit odmítnutím

		krok simulace							
		1	2	3	4	5	6	7	
w	w ₀								
	w ₁								
	w ₂	✗	✗						
	w ₃								
		⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

Důkaz rekurzivní vyčíslitelnosti

Dokažte, že následující jazyk **L** je rekurzivně vyčíslitelný:

$$L = \{ \langle M \rangle \mid \langle M \rangle \text{ je kód TS } M, \text{ kde } L(M) \neq \emptyset \}$$

Simulace nedeterministicky vygenerovaného slova

může skončit odmítnutím

		krok simulace							
		1	2	3	4	5	6	7	
w	w ₀								
	w ₁								
	w ₂	✗	✗	●					
	w ₃								
		⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

Důkaz rekurzivní vyčíslitelnosti

Dokažte, že následující jazyk **L** je rekurzivně vyčíslitelný:

$$L = \{ \langle M \rangle \mid \langle M \rangle \text{ je kód TS } M, \text{ kde } L(M) \neq \emptyset \}$$

Simulace nedeterministicky vygenerovaného slova

může cyklit

		krok simulace							
		1	2	3	4	5	6	7	
w	w ₀	✗							
	w ₁								
	w ₂								
	w ₃								
		⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

Důkaz rekurzivní vyčíslitelnosti

Dokažte, že následující jazyk **L** je rekurzivně vyčíslitelný:

$$L = \{ \langle M \rangle \mid \langle M \rangle \text{ je kód TS } M, \text{ kde } L(M) \neq \emptyset \}$$

Simulace nedeterministicky vygenerovaného slova

může cyklit

		krok simulace							
		1	2	3	4	5	6	7	
w	w ₀	×	×						
	w ₁								
	w ₂								
	w ₃								
		⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

Důkaz rekurzivní vyčíslitelnosti

Dokažte, že následující jazyk **L** je rekurzivně vyčíslitelný:

$$L = \{ \langle M \rangle \mid \langle M \rangle \text{ je kód TS } M, \text{ kde } L(M) \neq \emptyset \}$$

Simulace nedeterministicky vygenerovaného slova

může cyklit

		krok simulace							
		1	2	3	4	5	6	7	
W	w ₀	×	×	×					
	w ₁								
	w ₂								
	w ₃								
		⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

Důkaz rekurzivní vyčíslitelnosti

Dokažte, že následující jazyk **L** je rekurzivně vyčíslitelný:

$$L = \{ \langle M \rangle \mid \langle M \rangle \text{ je kód TS } M, \text{ kde } L(M) \neq \emptyset \}$$

Simulace nedeterministicky vygenerovaného slova

může cyklit

		krok simulace							
		1	2	3	4	5	6	7	
w	w ₀	×	×	×	×				
	w ₁								
	w ₂								
	w ₃								
		⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

Důkaz rekurzivní vyčíslitelnosti

Dokažte, že následující jazyk **L** je rekurzivně vyčíslitelný:

$$L = \{ \langle M \rangle \mid \langle M \rangle \text{ je kód TS } M, \text{ kde } L(M) \neq \emptyset \}$$

Simulace nedeterministicky vygenerovaného slova

může cyklit

		krok simulace							
		1	2	3	4	5	6	7	
w	w ₀	×	×	×	×	×			
	w ₁								
	w ₂								
	w ₃								
		⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

Důkaz rekurzivní vyčíslitelnosti

Dokažte, že následující jazyk **L** je rekurzivně vyčíslitelný:

$$L = \{ \langle M \rangle \mid \langle M \rangle \text{ je kód TS } M, \text{ kde } L(M) \neq \emptyset \}$$

Simulace nedeterministicky vygenerovaného slova

může cyklit

		krok simulace							
		1	2	3	4	5	6	7	
w	w ₀	×	×	×	×	×	×		
	w ₁								
	w ₂								
	w ₃								
		⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

Důkaz rekurzivní vyčíslitelnosti

Dokažte, že následující jazyk **L** je rekurzivně vyčíslitelný:

$$L = \{ \langle M \rangle \mid \langle M \rangle \text{ je kód TS } M, \text{ kde } L(M) \neq \emptyset \}$$

Simulace nedeterministicky vygenerovaného slova

může cyklit

		krok simulace							
		1	2	3	4	5	6	7	
w	w ₀	×	×	×	×	×	×	×	
	w ₁								
	w ₂								
	w ₃								
		⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

Důkaz rekurzivní vyčíslitelnosti

Dokažte, že následující jazyk **L** je rekurzivně vyčíslitelný:

$$L = \{ \langle M \rangle \mid \langle M \rangle \text{ je kód TS } M, \text{ kde } L(M) \neq \emptyset \}$$

Řešení pomocí **nedeterministického** Turingova stroje

- 1 Konstruujeme **NTS** T takový, aby $L(T) = L$

Důkaz rekurzivní vyčíslitelnosti

Dokažte, že následující jazyk **L** je rekurzivně vyčíslitelný:

$$L = \{ \langle M \rangle \mid \langle M \rangle \text{ je kód TS } M, \text{ kde } L(M) \neq \emptyset \}$$

Řešení pomocí **nedeterministického** Turingova stroje

- 1 Konstruujeme **NTS** T takový, aby $L(T) = L$
- 2 **NTS** T má na svém vstupu řetězec nad abecedou $\{0, 1\}$

Důkaz rekurzivní vyčíslitelnosti

Dokažte, že následující jazyk L je rekurzivně vyčíslitelný:

$$L = \{ \langle M \rangle \mid \langle M \rangle \text{ je kód TS } M, \text{ kde } L(M) \neq \emptyset \}$$

Řešení pomocí **nedeterministického** Turingova stroje

- 1 Konstruujeme **NTS** T takový, aby $L(T) = L$
- 2 **NTS** T má na svém vstupu řetězec nad abecedou $\{0, 1\}$
- 3 **NTS** T ověří, zda jeho vstup odpovídá validnímu kódu $\langle M \rangle$ nějakého TS M . Pokud ne, **odmítne**, jinak pokračuje

Důkaz rekurzivní vyčíslitelnosti

Dokažte, že následující jazyk L je rekurzivně vyčíslitelný:

$$L = \{ \langle M \rangle \mid \langle M \rangle \text{ je kód TS } M, \text{ kde } L(M) \neq \emptyset \}$$

Řešení pomocí **nedeterministického** Turingova stroje

- 1 Konstruujeme **NTS** T takový, aby $L(T) = L$
- 2 **NTS** T má na svém vstupu řetězec nad abecedou $\{0, 1\}$
- 3 **NTS** T ověří, zda jeho vstup odpovídá validnímu kódu $\langle M \rangle$ nějakého TS M . Pokud ne, **odmítne**, jinak pokračuje
- 4 **NTS** T nedeterministicky vygeneruje nějaký řetězec w nad abecedou Σ stroje M

Důkaz rekurzivní vyčíslitelnosti

Dokažte, že následující jazyk L je rekurzivně vyčíslitelný:

$$L = \{ \langle M \rangle \mid \langle M \rangle \text{ je kód TS } M, \text{ kde } L(M) \neq \emptyset \}$$

Řešení pomocí **nedeterministického** Turingova stroje

- 1 Konstruujeme **NTS** T takový, aby $L(T) = L$
- 2 **NTS** T má na svém vstupu řetězec nad abecedou $\{0, 1\}$
- 3 **NTS** T ověří, zda jeho vstup odpovídá validnímu kódu $\langle M \rangle$ nějakého TS M . Pokud ne, **odmítne**, jinak pokračuje
- 4 **NTS** T nedeterministicky vygeneruje nějaký řetězec w nad abecedou Σ stroje M
- 5 **NTS** T spustí simulaci stroje M na řetězci w

Důkaz rekurzivní vyčíslitelnosti

Dokažte, že následující jazyk L je rekurzivně vyčíslitelný:

$$L = \{ \langle M \rangle \mid \langle M \rangle \text{ je kód TS } M, \text{ kde } L(M) \neq \emptyset \}$$

Řešení pomocí **nedeterministického** Turingova stroje

- 1 Konstruujeme **NTS** T takový, aby $L(T) = L$
- 2 **NTS** T má na svém vstupu řetězec nad abecedou $\{0, 1\}$
- 3 **NTS** T ověří, zda jeho vstup odpovídá validnímu kódu $\langle M \rangle$ nějakého TS M . Pokud ne, **odmítne**, jinak pokračuje
- 4 **NTS** T nedeterministicky vygeneruje nějaký řetězec w nad abecedou Σ stroje M
- 5 **NTS** T spustí simulaci stroje M na řetězci w
- 6 Skončí-li simulace přijetím, **NTS** T akceptuje vstup $\langle M \rangle$

Důkaz rekurzivní vyčíslitelnosti

Dokažte, že následující jazyk L je rekurzivně vyčíslitelný:

$$L = \{ \langle M \rangle \mid \langle M \rangle \text{ je kód TS } M, \text{ kde } L(M) \neq \emptyset \}$$

Řešení pomocí **nedeterministického** Turingova stroje

- 1 Konstruujeme **NTS** T takový, aby $L(T) = L$
- 2 **NTS** T má na svém vstupu řetězec nad abecedou $\{0, 1\}$
- 3 **NTS** T ověří, zda jeho vstup odpovídá validnímu kódu $\langle M \rangle$ nějakého TS M . Pokud ne, **odmítne**, jinak pokračuje
- 4 **NTS** T nedeterministicky vygeneruje nějaký řetězec w nad abecedou Σ stroje M
- 5 **NTS** T spustí simulaci stroje M na řetězci w
- 6 Skončí-li simulace přijetím, **NTS** T akceptuje vstup $\langle M \rangle$
- 7 Skončí-li simulace odmítnutím, **NTS** T odmítne vstup $\langle M \rangle$

Důkaz rekurzivní vyčíslitelnosti

Dokažte, že následující jazyk L je rekurzivně vyčíslitelný:

$$L = \{ \langle M \rangle \mid \langle M \rangle \text{ je kód TS } M, \text{ kde } L(M) \neq \emptyset \}$$

Řešení pomocí **nedeterministického** Turingova stroje

- 1 Konstruujeme **NTS** T takový, aby $L(T) = L$
- 2 **NTS** T má na svém vstupu řetězec nad abecedou $\{0, 1\}$
- 3 **NTS** T ověří, zda jeho vstup odpovídá validnímu kódu $\langle M \rangle$ nějakého TS M . Pokud ne, **odmítne**, jinak pokračuje
- 4 **NTS** T nedeterministicky vygeneruje nějaký řetězec w nad abecedou Σ stroje M
- 5 **NTS** T spustí simulaci stroje M na řetězci w
- 6 Skončí-li simulace přijetím, **NTS** T akceptuje vstup $\langle M \rangle$
- 7 Skončí-li simulace odmítnutím, **NTS** T odmítne vstup $\langle M \rangle$
- 8 Cyklí-li simulace, cyklí i stroj **NTS** T

Determinismus

- Při použití deterministického Turingova stroje nelze provést krok nedeterministické volby řetězce w
- Je třeba systematicky generovat všechny řetězce nad abecedou simulovaného stroje

Determinismus

- Při použití **deterministického Turingova stroje** nelze provést krok **nedeterministické** volby řetězce w
- Je třeba **systematicky generovat všechny řetězce** nad abecedou simulovaného stroje

Strategie systematického provádění simulací

Spouštění simulací na jednotlivých slovech (nefunguje)

		krok simulace							
		1	2	3	4	5	6	7	
w	w ₀								
	w ₁								
	w ₂								
	w ₃								
		⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

Determinismus

- Při použití **deterministického Turingova stroje** nelze provést krok **nedeterministické** volby řetězce w
- Je třeba **systematicky generovat všechny řetězce** nad abecedou simulovaného stroje

Strategie systematického provádění simulací

Spouštění simulací na jednotlivých slovech (nefunguje)

		krok simulace							
		1	2	3	4	5	6	7	
w	w ₀	✗							
	w ₁								
	w ₂								
	w ₃								
		⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

Determinismus

- Při použití **deterministického Turingova stroje** nelze provést krok **nedeterministické** volby řetězce w
- Je třeba **systematicky generovat všechny řetězce** nad abecedou simulovaného stroje

Strategie systematického provádění simulací

Spouštění simulací na jednotlivých slovech (nefunguje)

		krok simulace							
		1	2	3	4	5	6	7	
w	w ₀	×	×						
	w ₁								
	w ₂								
	w ₃								
		⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

Determinismus

- Při použití **deterministického Turingova stroje** nelze provést krok **nedeterministické** volby řetězce w
- Je třeba **systematicky generovat všechny řetězce** nad abecedou simulovaného stroje

Strategie systematického provádění simulací

Spouštění simulací na jednotlivých slovech (nefunguje)

		krok simulace							
		1	2	3	4	5	6	7	
w	w ₀	×	×	×					
	w ₁								
	w ₂								
	w ₃								
		⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

Determinismus

- Při použití **deterministického Turingova stroje** nelze provést krok **nedeterministické** volby řetězce w
- Je třeba **systematicky generovat všechny řetězce** nad abecedou simulovaného stroje

Strategie systematického provádění simulací

Spouštění simulací na jednotlivých slovech (nefunguje)

		krok simulace							
		1	2	3	4	5	6	7	
w	w ₀	×	×	×	×				
	w ₁								
	w ₂								
	w ₃								
		⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

Determinismus

- Při použití **deterministického Turingova stroje** nelze provést krok **nedeterministické** volby řetězce w
- Je třeba **systematicky generovat všechny řetězce** nad abecedou simulovaného stroje

Strategie systematického provádění simulací

Spouštění simulací na jednotlivých slovech (nefunguje)

		krok simulace							
		1	2	3	4	5	6	7	
w	w ₀	×	×	×	×	×			
	w ₁								
	w ₂								
	w ₃								
		⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

Determinismus

- Při použití **deterministického Turingova stroje** nelze provést krok **nedeterministické** volby řetězce w
- Je třeba **systematicky generovat všechny řetězce** nad abecedou simulovaného stroje

Strategie systematického provádění simulací

Spouštění simulací na jednotlivých slovech (nefunguje)

		krok simulace							
		1	2	3	4	5	6	7	
w	w ₀	×	×	×	×	×	×		
	w ₁								
	w ₂								
	w ₃								
		⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

Determinismus

- Při použití **deterministického Turingova stroje** nelze provést krok **nedeterministické** volby řetězce w
- Je třeba **systematicky generovat všechny řetězce** nad abecedou simulovaného stroje

Strategie systematického provádění simulací

Spouštění simulací na jednotlivých slovech (nefunguje)

		krok simulace							
		1	2	3	4	5	6	7	
w	w ₀	×	×	×	×	×	×	×	
	w ₁								
	w ₂								
	w ₃								
		⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

Determinismus

- Při použití **deterministického Turingova stroje** nelze provést krok **nedeterministické** volby řetězce w
- Je třeba **systematicky generovat všechny řetězce** nad abecedou simulovaného stroje

Strategie systematického provádění simulací

Spouštění prvních kroků všech simulací (nefunguje)

		krok simulace							
		1	2	3	4	5	6	7	
w	w ₀	✗							
	w ₁								
	w ₂								
	w ₃								
		⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

Determinismus

- Při použití **deterministického Turingova stroje** nelze provést krok **nedeterministické** volby řetězce w
- Je třeba **systematicky generovat všechny řetězce** nad abecedou simulovaného stroje

Strategie systematického provádění simulací

Spouštění prvních kroků všech simulací (nefunguje)

		krok simulace							
		1	2	3	4	5	6	7	
w	w ₀	✗							
	w ₁	✗							
	w ₂								
	w ₃								
		⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

Determinismus

- Při použití **deterministického Turingova stroje** nelze provést krok **nedeterministické** volby řetězce w
- Je třeba **systematicky generovat všechny řetězce** nad abecedou simulovaného stroje

Strategie systematického provádění simulací

Spouštění prvních kroků všech simulací (nefunguje)

		krok simulace							
		1	2	3	4	5	6	7	
w	w ₀	✗							
	w ₁	✗							
	w ₂	✗							
	w ₃								
		⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

Determinismus

- Při použití **deterministického Turingova stroje** nelze provést krok **nedeterministické** volby řetězce w
- Je třeba **systematicky generovat všechny řetězce** nad abecedou simulovaného stroje

Strategie systematického provádění simulací

Spouštění prvních kroků všech simulací (nefunguje)

		krok simulace							
		1	2	3	4	5	6	7	
w	w ₀	✗							
	w ₁	✗							
	w ₂	✗							
	w ₃	✗							
		⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

Determinismus

- Při použití **deterministického Turingova stroje** nelze provést krok **nedeterministické** volby řetězce w
- Je třeba **systematicky generovat všechny řetězce** nad abecedou simulovaného stroje

Strategie systematického provádění simulací

"Diagonální" střídání simulací po krocích (funguje)

		krok simulace							
		1	2	3	4	5	6	7	
w	w ₀	✗							
	w ₁								
	w ₂								
	w ₃								
		⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

Determinismus

- Při použití **deterministického Turingova stroje** nelze provést krok **nedeterministické** volby řetězce w
- Je třeba **systematicky generovat všechny řetězce** nad abecedou simulovaného stroje

Strategie systematického provádění simulací

"Diagonální" střídání simulací po krocích (funguje)

		krok simulace							
		1	2	3	4	5	6	7	
w	w ₀	×	×						
	w ₁								
	w ₂								
	w ₃								
		⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

Determinismus

- Při použití **deterministického Turingova stroje** nelze provést krok **nedeterministické** volby řetězce w
- Je třeba **systematicky generovat všechny řetězce** nad abecedou simulovaného stroje

Strategie systematického provádění simulací

"Diagonální" střídání simulací po krocích (funguje)

		krok simulace							
		1	2	3	4	5	6	7	
w	w ₀	×	×						
	w ₁	×							
	w ₂								
	w ₃								
		⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

Determinismus

- Při použití **deterministického Turingova stroje** nelze provést krok **nedeterministické** volby řetězce w
- Je třeba **systematicky generovat všechny řetězce** nad abecedou simulovaného stroje

Strategie systematického provádění simulací

"Diagonální" střídání simulací po krocích (funguje)

		krok simulace							
		1	2	3	4	5	6	7	
w	w ₀	×	×	×					
	w ₁	×							
	w ₂								
	w ₃								
		⋮	⋮	⋮	⋮	⋮	⋮	⋮	↘

Determinismus

- Při použití **deterministického Turingova stroje** nelze provést krok **nedeterministické** volby řetězce w
- Je třeba **systematicky generovat všechny řetězce** nad abecedou simulovaného stroje

Strategie systematického provádění simulací

"Diagonální" střídání simulací po krocích (funguje)

		krok simulace							
		1	2	3	4	5	6	7	
w	w ₀	×	×	×					
	w ₁	×	×						
	w ₂								
	w ₃								
		⋮	⋮	⋮	⋮	⋮	⋮	⋮	↘

Determinismus

- Při použití **deterministického Turingova stroje** nelze provést krok **nedeterministické** volby řetězce w
- Je třeba **systematicky generovat všechny řetězce** nad abecedou simulovaného stroje

Strategie systematického provádění simulací

"Diagonální" střídání simulací po krocích (funguje)

		krok simulace							
		1	2	3	4	5	6	7	
w	w ₀	×	×	×					
	w ₁	×	×						
	w ₂	×							
	w ₃								
		⋮	⋮	⋮	⋮	⋮	⋮	⋮	↘

Determinismus

- Při použití **deterministického Turingova stroje** nelze provést krok **nedeterministické** volby řetězce w
- Je třeba **systematicky generovat všechny řetězce** nad abecedou simulovaného stroje

Strategie systematického provádění simulací

"Diagonální" střídání simulací po krocích (funguje)

		krok simulace							
		1	2	3	4	5	6	7	
w	w ₀	×	×	×	×				
	w ₁	×	×						
	w ₂	×							
	w ₃								
		⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

Determinismus

- Při použití **deterministického Turingova stroje** nelze provést krok **nedeterministické** volby řetězce w
- Je třeba **systematicky generovat všechny řetězce** nad abecedou simulovaného stroje

Strategie systematického provádění simulací

"Diagonální" střídání simulací po krocích (funguje)

		krok simulace							
		1	2	3	4	5	6	7	
w	w ₀	×	×	×	×				
	w ₁	×	×	×					
	w ₂	×							
	w ₃								
		⋮	⋮	⋮	⋮	⋮	⋮	⋮	↘

Determinismus

- Při použití **deterministického Turingova stroje** nelze provést krok **nedeterministické** volby řetězce w
- Je třeba **systematicky generovat všechny řetězce** nad abecedou simulovaného stroje

Strategie systematického provádění simulací

"Diagonální" střídání simulací po krocích (funguje)

		krok simulace							
		1	2	3	4	5	6	7	
w	w ₀	×	×	×	×				
	w ₁	×	×	×					
	w ₂	×	×						
	w ₃								
		⋮	⋮	⋮	⋮	⋮	⋮	⋮	↘

Determinismus

- Při použití **deterministického Turingova stroje** nelze provést krok **nedeterministické** volby řetězce w
- Je třeba **systematicky generovat všechny řetězce** nad abecedou simulovaného stroje

Strategie systematického provádění simulací

"Diagonální" střídání simulací po krocích (funguje)

		krok simulace							
		1	2	3	4	5	6	7	
w	w ₀	×	×	×	×				
	w ₁	×	×	×					
	w ₂	×	×						
	w ₃	×							
		⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

Determinismus

- Při použití **deterministického Turingova stroje** nelze provést krok **nedeterministické** volby řetězce w
- Je třeba **systematicky generovat všechny řetězce** nad abecedou simulovaného stroje

Strategie systematického provádění simulací

"Diagonální" střídání simulací po krocích (funguje)

		krok simulace							
		1	2	3	4	5	6	7	
w	w ₀	×	×	×	×	×			
	w ₁	×	×	×					
	w ₂	×	×						
	w ₃	×							
		⋮	⋮	⋮	⋮	⋮	⋮	⋮	↘

Determinismus

- Při použití **deterministického Turingova stroje** nelze provést krok **nedeterministické** volby řetězce w
- Je třeba **systematicky generovat všechny řetězce** nad abecedou simulovaného stroje

Strategie systematického provádění simulací

"Diagonální" střídání simulací po krocích (funguje)

		krok simulace							
		1	2	3	4	5	6	7	
w	w ₀	×	×	×	×	×			
	w ₁	×	×	×	×				
	w ₂	×	×						
	w ₃	×							
		⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

Determinismus

- Při použití **deterministického Turingova stroje** nelze provést krok **nedeterministické** volby řetězce w
- Je třeba **systematicky generovat všechny řetězce** nad abecedou simulovaného stroje

Strategie systematického provádění simulací

"Diagonální" střídání simulací po krocích (funguje)

		krok simulace							
		1	2	3	4	5	6	7	
w	w ₀	×	×	×	×	×			
	w ₁	×	×	×	×				
	w ₂	×	×	●					
	w ₃	×							
		⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

Důkaz rekurzivní vyčísitelnosti

Dokažte, že následující jazyk L je rekurzivně vyčíslitelný:

$$L = \{ \langle M \rangle \mid \langle M \rangle \text{ je kód TS } M, \text{ kde } L(M) \neq \emptyset \}$$

Důkaz rekurzivní vyčíslitelnosti

Dokažte, že následující jazyk **L** je rekurzivně vyčíslitelný:

$$L = \{ \langle M \rangle \mid \langle M \rangle \text{ je kód TS } M, \text{ kde } L(M) \neq \emptyset \}$$

Řešení pomocí **deterministického** Turingova stroje

- Konstruujeme **DTS** T takový, aby $L(T) = L$

Důkaz rekurzivní vyčíslitelnosti

Dokažte, že následující jazyk L je rekurzivně vyčíslitelný:

$$L = \{\langle M \rangle \mid \langle M \rangle \text{ je kód TS } M, \text{ kde } L(M) \neq \emptyset\}$$

Řešení pomocí **deterministického** Turingova stroje

- Konstruujeme **DTS** T takový, aby $L(T) = L$
- **DTS** T má na svém vstupu řetězec nad abecedou $\{0, 1\}$

Důkaz rekurzivní vyčíslitelnosti

Dokažte, že následující jazyk L je rekurzivně vyčíslitelný:

$$L = \{ \langle M \rangle \mid \langle M \rangle \text{ je kód TS } M, \text{ kde } L(M) \neq \emptyset \}$$

Řešení pomocí **deterministického** Turingova stroje

- Konstruujeme **DTS** T takový, aby $L(T) = L$
- **DTS** T má na svém vstupu řetězec nad abecedou $\{0, 1\}$
- **DTS** T ověří, zda jeho vstup odpovídá validnímu kódu $\langle M \rangle$ nějakého TS M . Pokud ne, **odmítne**, jinak pokračuje.

Důkaz rekurzivní vyčíslitelnosti

Dokažte, že následující jazyk L je rekurzivně vyčíslitelný:

$$L = \{ \langle M \rangle \mid \langle M \rangle \text{ je kód TS } M, \text{ kde } L(M) \neq \emptyset \}$$

Řešení pomocí **deterministického** Turingova stroje

- Konstruujeme **DTS** T takový, aby $L(T) = L$
- **DTS** T má na svém vstupu řetězec nad abecedou $\{0, 1\}$
- **DTS** T ověří, zda jeho vstup odpovídá validnímu kódu $\langle M \rangle$ nějakého TS M . Pokud ne, **odmítne**, jinak pokračuje.
- **DTS** T postupně generuje veškeré řetězce nad abecedou Σ stroje M v lexikografickém pořadí $w_0, w_1, w_2, \dots$

Důkaz rekurzivní vyčíslitelnosti

Dokažte, že následující jazyk L je rekurzivně vyčíslitelný:

$$L = \{ \langle M \rangle \mid \langle M \rangle \text{ je kód TS } M, \text{ kde } L(M) \neq \emptyset \}$$

Řešení pomocí **deterministického** Turingova stroje

- Konstruujeme **DTS** T takový, aby $L(T) = L$
- **DTS** T má na svém vstupu řetězec nad abecedou $\{0, 1\}$
- **DTS** T ověří, zda jeho vstup odpovídá validnímu kódu $\langle M \rangle$ nějakého TS M . Pokud ne, **odmítne**, jinak pokračuje.
- **DTS** T postupně generuje veškeré řetězce nad abecedou Σ stroje M v lexikografickém pořadí $w_0, w_1, w_2, \dots$
- **DTS** T spustí první krok simulace stroje M na řetězci w_0 , simulaci poté pozastaví a uloží si řídicí informace na pomocnou pásku

Důkaz rekurzivní vyčíslitelnosti

Dokažte, že následující jazyk L je rekurzivně vyčíslitelný:

$$L = \{ \langle M \rangle \mid \langle M \rangle \text{ je kód TS } M, \text{ kde } L(M) \neq \emptyset \}$$

Řešení pomocí **deterministického** Turingova stroje

- **DTS** T následně opakovaně provádí následující postup:
 - provede **jeden** krok všech rozpracovaných simulací
 - provede **první** krok simulace následujícího dosud nepoužitého řetězce v uvedeném lexikografickém pořadí

Důkaz rekurzivní vyčíslitelnosti

Dokažte, že následující jazyk L je rekurzivně vyčíslitelný:

$$L = \{ \langle M \rangle \mid \langle M \rangle \text{ je kód TS } M, \text{ kde } L(M) \neq \emptyset \}$$

Řešení pomocí **deterministického** Turingova stroje

- **DTS** T následně opakovně provádí následující postup:
 - provede **jeden** krok všech rozpracovaných simulací
 - provede **první** krok simulace následujícího dosud nepoužitého řetězce v uvedeném lexikografickém pořadí
- Pokud některá simulace skončí odmítnutím, napříště již není spouštěna

Důkaz rekurzivní vyčíslitelnosti

Dokažte, že následující jazyk L je rekurzivně vyčíslitelný:

$$L = \{ \langle M \rangle \mid \langle M \rangle \text{ je kód TS } M, \text{ kde } L(M) \neq \emptyset \}$$

Řešení pomocí **deterministického** Turingova stroje

- **DTS** T následně opakovaně provádí následující postup:
 - provede **jeden** krok všech rozpracovaných simulací
 - provede **první** krok simulace následujícího dosud nepoužitého řetězce v uvedeném lexikografickém pořadí
- Pokud některá simulace skončí odmítnutím, napříště již není spouštěna
- Pokud některá simulace skončí akceptováním, **DTS** T akceptuje svůj vstup $\langle M \rangle$

Důkaz rekurzivní vyčíslitelnosti

Dokažte, že následující jazyk L je rekurzivně vyčíslitelný:

$$L = \{ \langle M \rangle \mid \langle M \rangle \text{ je kód TS } M, \text{ kde } L(M) \neq \emptyset \}$$

Řešení pomocí **deterministického** Turingova stroje

- **DTS** T následně opakovaně provádí následující postup:
 - provede **jeden** krok všech rozpracovaných simulací
 - provede **první** krok simulace následujícího dosud nepoužitého řetězce v uvedeném lexikografickém pořadí
- Pokud některá simulace skončí odmítnutím, napříště již není spouštěna
- Pokud některá simulace skončí akceptováním, **DTS** T akceptuje svůj vstup $\langle M \rangle$
- Pokud je jazyk stroje M prázdný, **DTS** T cyklí

Důkaz rekurzivní vyčíslitelnosti

Dokažte, že následující jazyk **L** je rekurzivně vyčíslitelný:

$$L = \{\langle M \rangle \# \langle n \rangle \mid \forall w \in \Sigma^* : |w| \leq n \Rightarrow w \in L(M)\}$$

		krok simulace							
		1	2	3	4	5	6	7	
w	w ₀								
	w ₁								
	w ₂								
	w ₃								
	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

Důkaz rekurzivní vyčíslitelnosti

Dokažte, že následující jazyk L je rekurzivně vyčíslitelný:

$$L = \{\langle M \rangle \# \langle n \rangle \mid \forall w \in \Sigma^* : |w| \leq n \Rightarrow w \in L(M)\}$$

Důkaz rekurzivní vyčíslitelnosti

Dokažte, že následující jazyk L je rekurzivně vyčíslitelný:

$$L = \{\langle M \rangle \# \langle n \rangle \mid \forall w \in \Sigma^* : |w| \leq n \Rightarrow w \in L(M)\}$$

Řešení pomocí **deterministického** Turingova stroje

- Konstruujeme **DTS** T takový, aby $L(T) = L$

Důkaz rekurzivní vyčíslitelnosti

Dokažte, že následující jazyk L je rekurzivně vyčíslitelný:

$$L = \{\langle M \rangle \# \langle n \rangle \mid \forall w \in \Sigma^* : |w| \leq n \Rightarrow w \in L(M)\}$$

Řešení pomocí **deterministického** Turingova stroje

- Konstruujeme **DTS** T takový, aby $L(T) = L$
- **DTS** T má na svém vstupu řetězec nad abecedou $\{0, 1, \#\}$

Důkaz rekurzivní vyčíslitelnosti

Dokažte, že následující jazyk L je rekurzivně vyčíslitelný:

$$L = \{\langle M \rangle \# \langle n \rangle \mid \forall w \in \Sigma^* : |w| \leq n \Rightarrow w \in L(M)\}$$

Řešení pomocí **deterministického** Turingova stroje

- Konstruujeme **DTS** T takový, aby $L(T) = L$
- **DTS** T má na svém vstupu řetězec nad abecedou $\{0, 1, \#\}$
- **DTS** T ověří, zda jeho vstup je ve tvaru $\langle M \rangle \# \langle n \rangle$, kde $\langle M \rangle$ je kód Turingova stroje M a $\langle n \rangle$ je kód přirozeného čísla n

Důkaz rekurzivní vyčíslitelnosti

Dokažte, že následující jazyk L je rekurzivně vyčíslitelný:

$$L = \{\langle M \rangle \# \langle n \rangle \mid \forall w \in \Sigma^* : |w| \leq n \Rightarrow w \in L(M)\}$$

Řešení pomocí **deterministického** Turingova stroje

- Konstruujeme **DTS** T takový, aby $L(T) = L$
- **DTS** T má na svém vstupu řetězec nad abecedou $\{0, 1, \#\}$
- **DTS** T ověří, zda jeho vstup je ve tvaru $\langle M \rangle \# \langle n \rangle$, kde $\langle M \rangle$ je kód Turingova stroje M a $\langle n \rangle$ je kód přirozeného čísla n
- Pokud tomu tak není, **DTS** T odmítne, jinak pokračuje

Důkaz rekurzivní vyčíslitelnosti

Dokažte, že následující jazyk L je rekurzivně vyčíslitelný:

$$L = \{ \langle M \rangle \# \langle n \rangle \mid \forall w \in \Sigma^* : |w| \leq n \Rightarrow w \in L(M) \}$$

Řešení pomocí **deterministického** Turingova stroje

- Konstruujeme **DTS** T takový, aby $L(T) = L$
- **DTS** T má na svém vstupu řetězec nad abecedou $\{0, 1, \#\}$
- **DTS** T ověří, zda jeho vstup je ve tvaru $\langle M \rangle \# \langle n \rangle$, kde $\langle M \rangle$ je kód Turingova stroje M a $\langle n \rangle$ je kód přirozeného čísla n
- Pokud tomu tak není, **DTS** T odmítne, jinak pokračuje
- **DTS** T postupně generuje veškeré řetězce $w_0, w_1, \dots$ nad abecedou stroje M , které mají délku nejvýše n

Důkaz rekurzivní vyčíslitelnosti

Dokažte, že následující jazyk L je rekurzivně vyčíslitelný:

$$L = \{ \langle M \rangle \# \langle n \rangle \mid \forall w \in \Sigma^* : |w| \leq n \Rightarrow w \in L(M) \}$$

Řešení pomocí **deterministického** Turingova stroje

- Konstruujeme **DTS** T takový, aby $L(T) = L$
- **DTS** T má na svém vstupu řetězec nad abecedou $\{0, 1, \#\}$
- **DTS** T ověří, zda jeho vstup je ve tvaru $\langle M \rangle \# \langle n \rangle$, kde $\langle M \rangle$ je kód Turingova stroje M a $\langle n \rangle$ je kód přirozeného čísla n
- Pokud tomu tak není, **DTS** T odmítne, jinak pokračuje
- **DTS** T postupně generuje veškeré řetězce $w_0, w_1, \dots$ nad abecedou stroje M , které mají délku nejvýše n
- **DTS** T postupně spouští simulace stroje M nad všemi těmito řetězci

Důkaz rekurzivní vyčíslitelnosti

Dokažte, že následující jazyk L je rekurzivně vyčíslitelný:

$$L = \{\langle M \rangle \# \langle n \rangle \mid \forall w \in \Sigma^* : |w| \leq n \Rightarrow w \in L(M)\}$$

Řešení pomocí **deterministického** Turingova stroje

- Pokud některá simulace skončí odmítnutím, odmítne i stroj T

Důkaz rekurzivní vyčíslitelnosti

Dokažte, že následující jazyk L je rekurzivně vyčíslitelný:

$$L = \{\langle M \rangle \# \langle n \rangle \mid \forall w \in \Sigma^* : |w| \leq n \Rightarrow w \in L(M)\}$$

Řešení pomocí **deterministického** Turingova stroje

- Pokud některá simulace skončí odmítnutím, odmítne i stroj T
- Pokud některá simulace cyklí, cyklí i stroj T

Důkaz rekurzivní vyčíslitelnosti

Dokažte, že následující jazyk L je rekurzivně vyčíslitelný:

$$L = \{\langle M \rangle \# \langle n \rangle \mid \forall w \in \Sigma^* : |w| \leq n \Rightarrow w \in L(M)\}$$

Řešení pomocí **deterministického** Turingova stroje

- Pokud některá simulace skončí odmítnutím, odmítne i stroj T
- Pokud některá simulace cyklí, cyklí i stroj T
- Pokud veškeré simulace skončily akceptováním, stroj T akceptuje

Dotazy 🐱