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 18, 2024

Bezkontextové jazyky

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

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

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

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

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 $\overline{\{a^n b^n c^n \mid n \geq 1\}}$

ZA

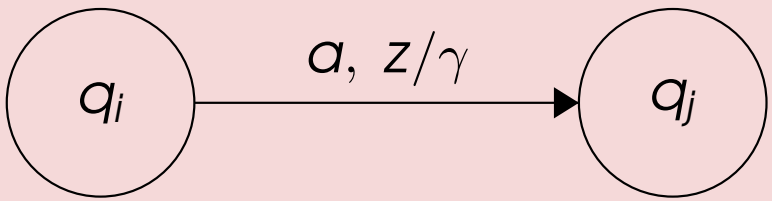
- $\delta : Q \times (\Sigma \cup \{\epsilon\}) \times \Gamma \rightarrow 2^{Q \times \Gamma^*}$
- Odstraňujeme právě jeden symbol z vrcholu zásobníku

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

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

ZA

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



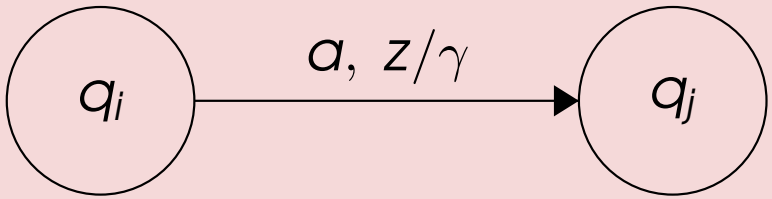
Příklad 1

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

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

ZA

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



RZA

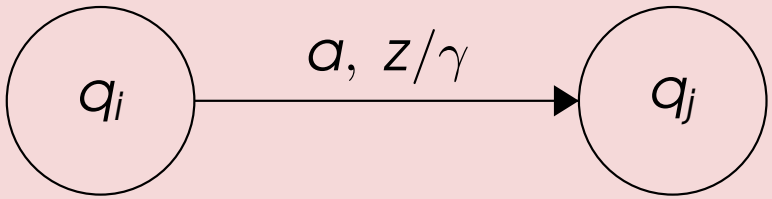
- $\delta : Q \times (\Sigma \cup \{\epsilon\}) \times \Gamma^* \rightarrow 2^{Q \times \Gamma^*}$

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

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

ZA

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



RZA

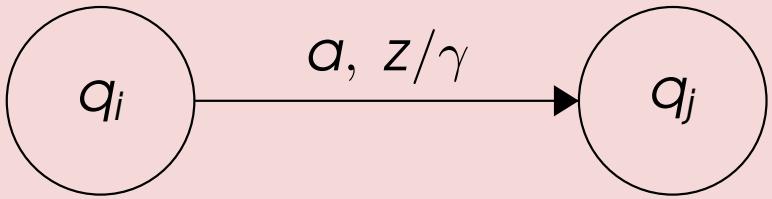
- $\delta : Q \times (\Sigma \cup \{\epsilon\}) \times \Gamma^* \rightarrow 2^{Q \times \Gamma^*}$
- Odstraňujeme libovolný počet symbolů z vrcholu zásobníku

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

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

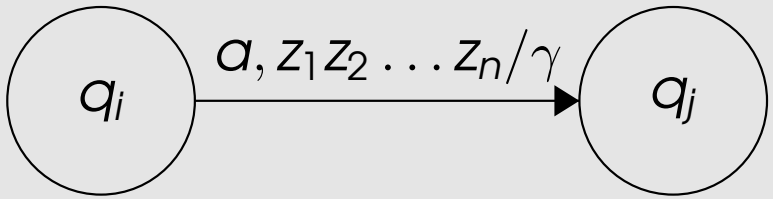
ZA

- $\delta : Q \times (\Sigma \cup \{\epsilon\}) \times \Gamma \rightarrow 2^{Q \times \Gamma^*}$
- 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 $\overline{\{a^n b^n c^n \mid n \geq 1\}}$

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

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

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

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

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

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

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

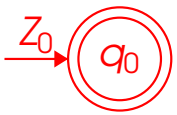
- Pracujeme s abecedou $\Sigma = \{a, b, c\}$
- $\overline{\{a^n b^n c^n \mid n \geq 1\}}$ =
  1. $\{xyz \in \{a, b, c\}^* \mid y \in \{ac, ba, ca, cb\}\} \cup$
  2. $\{w \in \{a, b, c\}^* \mid \#_a(w) \neq \#_b(w)\} \cup$
  3. $\{w \in \{a, b, c\}^* \mid \#_a(w) \neq \#_c(w)\} \cup$
  4. $\{w \in \{a,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

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

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

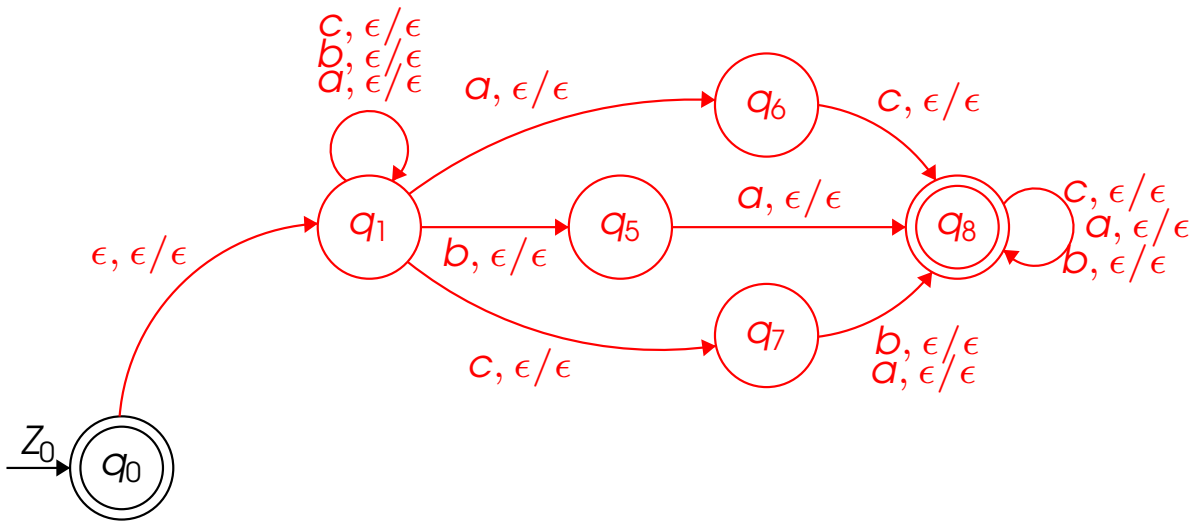
Konstrukce RZA nad abecedou $\Sigma = \{a, b, c\}$

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



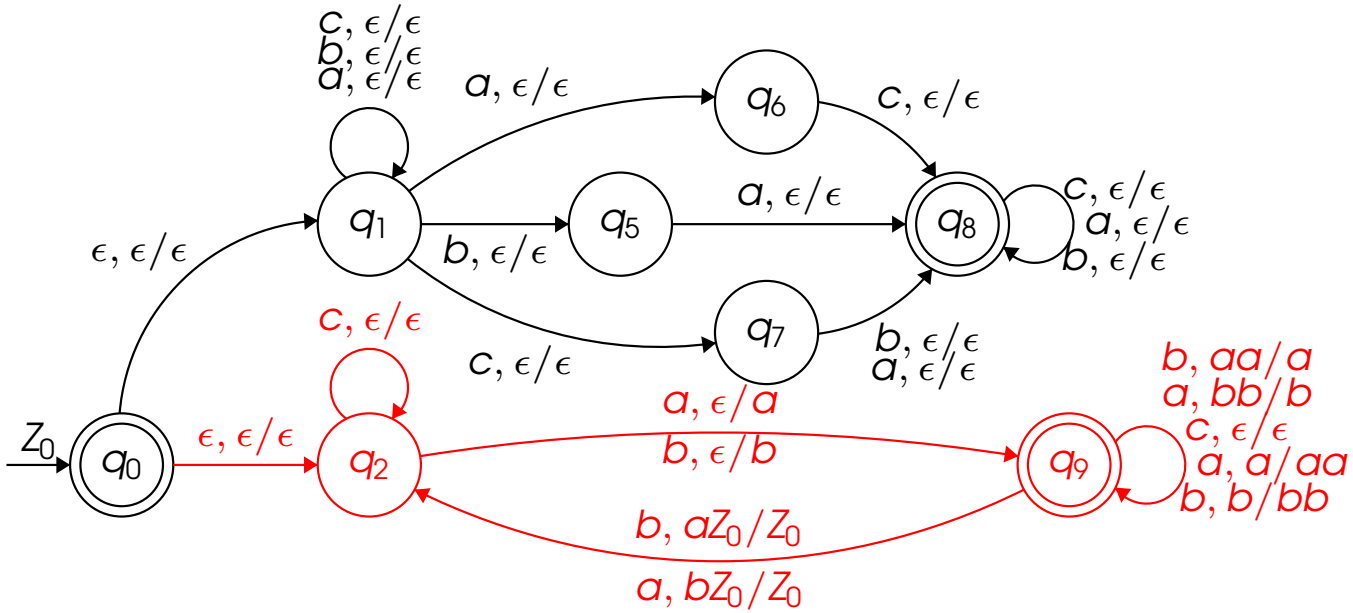
Konstrukce RZA nad abecedou $\Sigma = \{a, b, c\}$

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



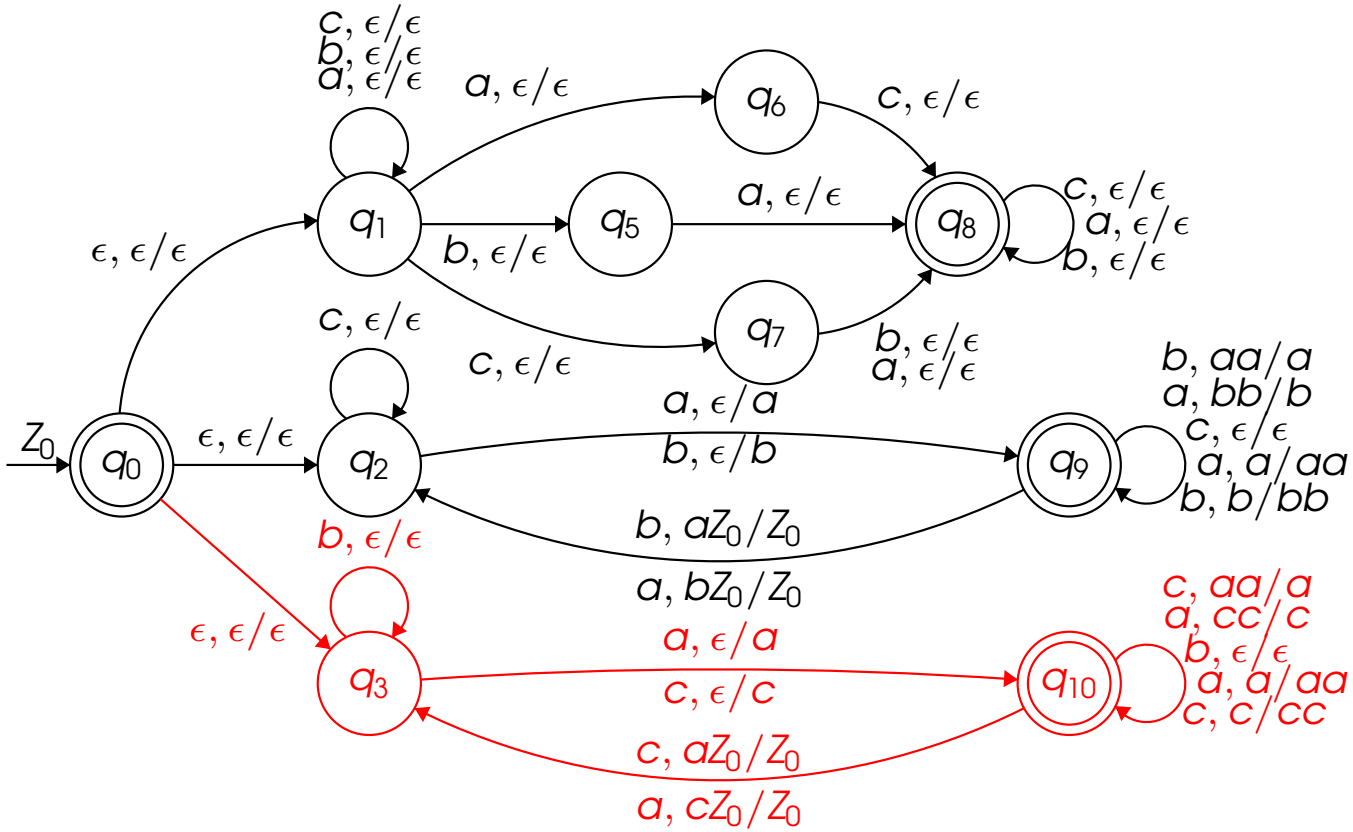
Konstrukce RZA nad abecedou $\Sigma = \{a, b, c\}$

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



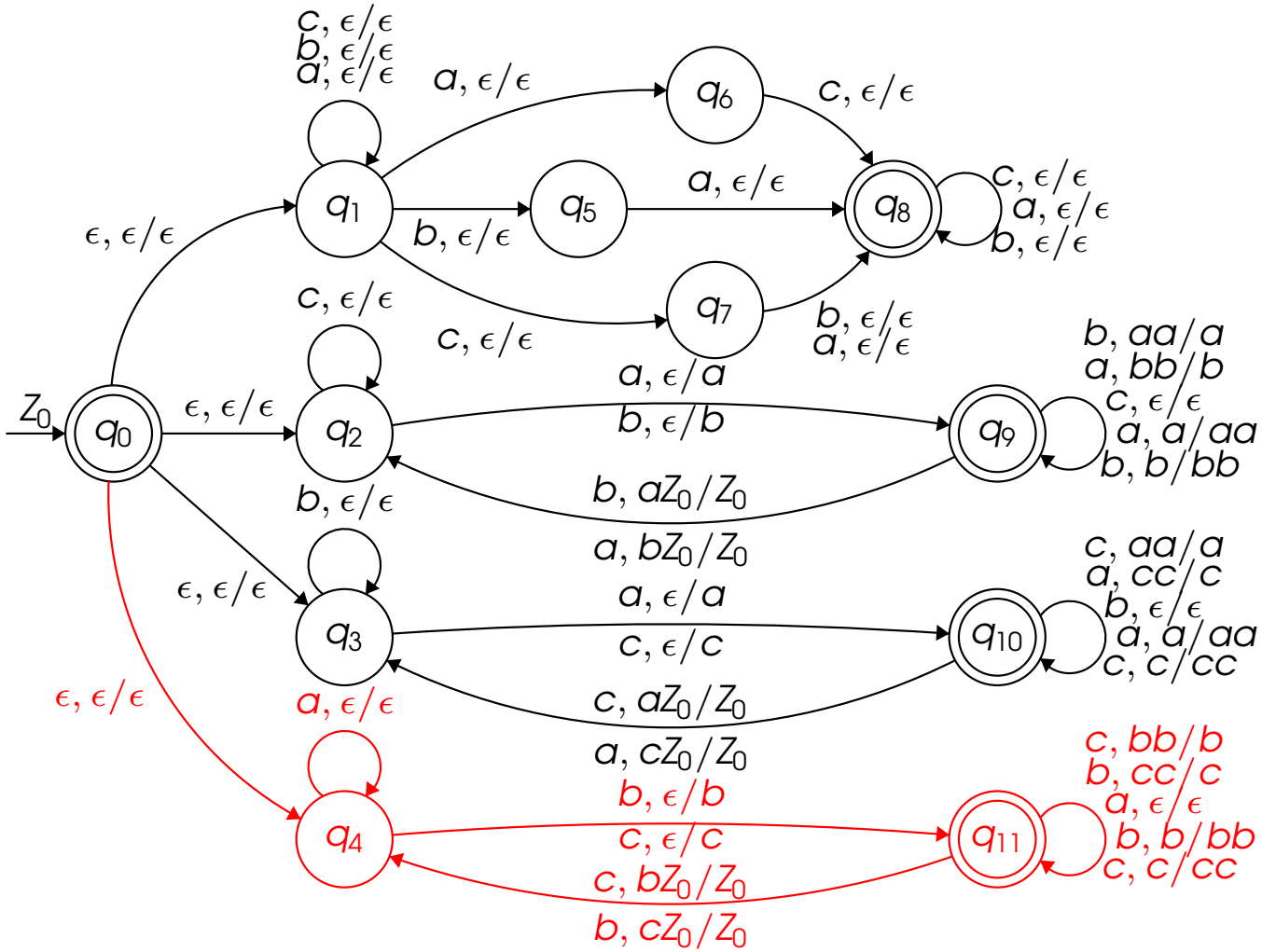
Konstrukce RZA nad abecedou $\Sigma = \{a, b, c\}$

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



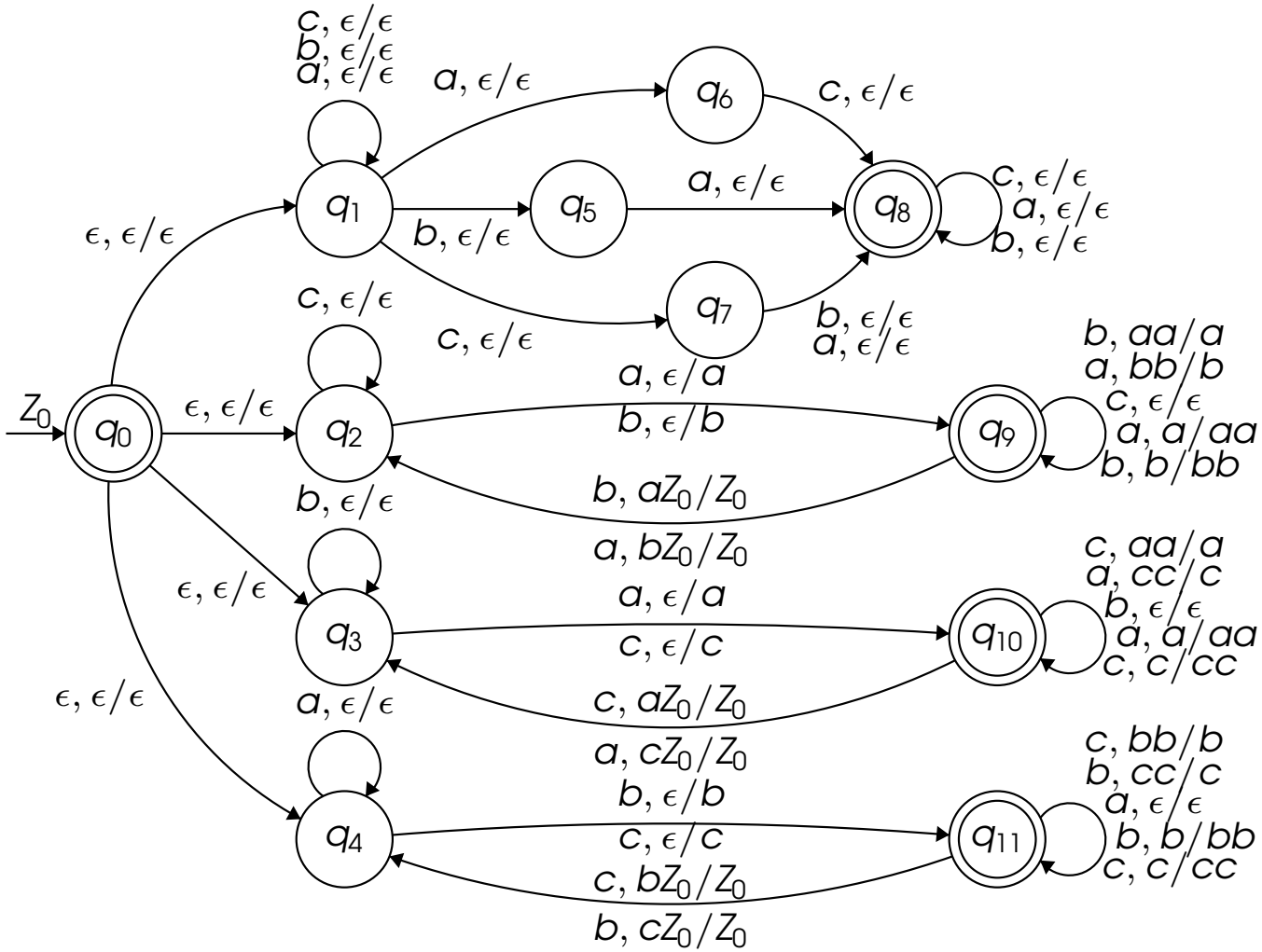
Konstrukce RZA nad abecedou $\Sigma = \{a, b, c\}$

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



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

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



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

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

- 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 $\overline{\{a^n b^n c^n \mid n \geq 1\}}$

- 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 $\overline{\{a^n b^n c^n \mid n \geq 1\}}$

- 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 $\overline{\{a^n b^n c^n \mid n \geq 1\}}$

- 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 $\overline{\{a^n b^n c^n \mid n \geq 1\}}$

- 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 $\overline{\{a^n b^n c^n \mid n \geq 1\}}$

- 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 $\overline{\{a^n b^n c^n \mid n \geq 1\}}$

- 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 zapište algoritmus pro sestrojení průniku ZA a KA

Průnik KA

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

Algoritmy nad zásobníkovými automaty

Formálně zapište algoritmus pro sestrojení průniku ZA a KA

Průnik KA

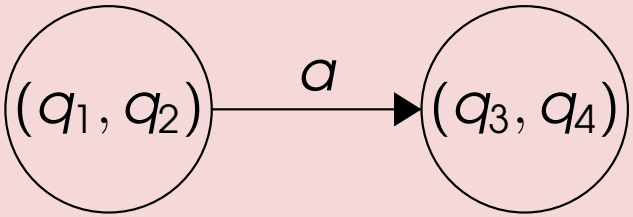
- třída $\mathcal{L}_3$ je uzavřena na průnik
- $L_1 \cap L_2 = \overline{\overline{L_1} \cup \overline{L_2}}$

Algoritmy nad zásobníkovými automaty

Formálně zapište algoritmus pro sestrojení průniku ZA a KA

Průnik KA

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

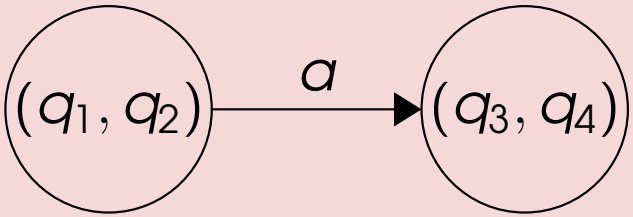


Algoritmy nad zásobníkovými automaty

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

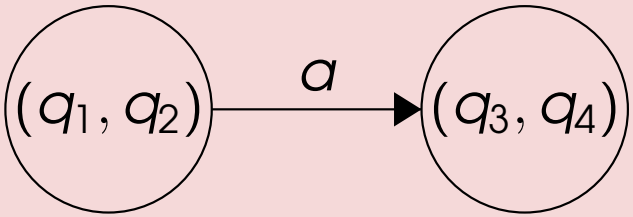
- třída $\mathcal{L}_2$ není uzavřena na průnik

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

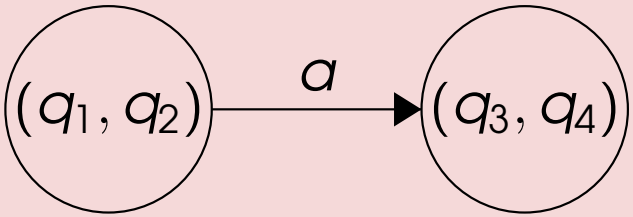
- třída $\mathcal{L}_2$ není uzavřena na průnik
- $L_1 = \{a^n b^n c^m \mid m, n \geq 0\}$

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

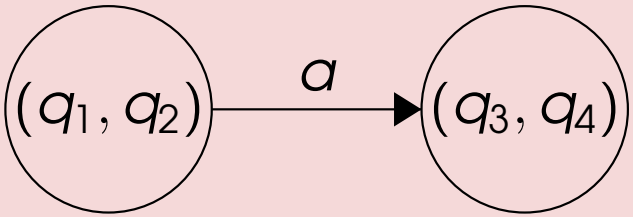
- třída $\mathcal{L}_2$ není uzavřena na průnik
- $L_1 = \{a^n b^n c^m \mid m, n \geq 0\}$
- $L_2 = \{a^m b^n c^n \mid m, n \geq 0\}$

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

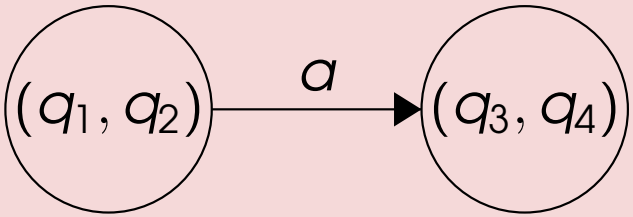
- třída $\mathcal{L}_2$ není uzavřena na průnik
- $L_1 = \{a^n b^n c^m \mid m, n \geq 0\}$
- $L_2 = \{a^m b^n c^n \mid m, n \geq 0\}$
- $L_3 = L_1 \cap L_2$

Algoritmy nad zásobníkovými automaty

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

Algoritmy nad zásobníkovými automaty

Formálně zapište algoritmus pro sestrojení průniku ZA a KA

- Vstup:

Algoritmy nad zásobníkovými automaty

Formálně zapište algoritmus pro sestrojení průniku ZA a KA

- Vstup:
  1. ZA $A_1 = \left(Q_1, \Sigma_1, \Gamma_1, \delta_1, q_0^1, Z_0^1, F_1\right)$

Algoritmy nad zásobníkovými automaty

Formálně zapište algoritmus pro sestrojení průniku ZA a KA

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

Algoritmy nad zásobníkovými automaty

Formálně zapište algoritmus pro sestrojení průniku ZA a KA

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

Algoritmy nad zásobníkovými automaty

Formálně zapište algoritmus pro sestrojení průniku ZA a KA

- Vstup:
  1. ZA $A_1 = \left(Q_1, \Sigma_1, \Gamma_1, \delta_1, q_0^1, Z_0^1, F_1\right)$
  2. KA $A_2 = \left(Q_2, \Sigma_2, \delta_2, q_0^2, F_2\right)$
- 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 zapište algoritmus pro sestrojení průniku ZA a KA

- Vstup:
  1. ZA $A_1 = \left(Q_1, \Sigma_1, \Gamma_1, \delta_1, q_0^1, Z_0^1, F_1\right)$
  2. KA $A_2 = \left(Q_2, \Sigma_2, \delta_2, q_0^2, F_2\right)$
- 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:
  1. ZA $A_1 = \left(Q_1, \Sigma_1, \Gamma_1, \delta_1, q_0^1, Z_0^1, F_1\right)$
  2. KA $A_2 = \left(Q_2, \Sigma_2, \delta_2, q_0^2, F_2\right)$
- Výstup:
  - ZA $A = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$, kde $L(A) = L(A_1) \cap L(A_2)$
- Algoritmus:
  1. $Q = Q_1 \times Q_2$

Algoritmy nad zásobníkovými automaty

Formálně zapište algoritmus pro sestrojení průniku ZA a KA

- Vstup:
  1. ZA $A_1 = \left(Q_1, \Sigma_1, \Gamma_1, \delta_1, q_0^1, Z_0^1, F_1\right)$
  2. KA $A_2 = \left(Q_2, \Sigma_2, \delta_2, q_0^2, F_2\right)$
- 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 $\Sigma_1 \cap \Sigma_2 = \varnothing$

Algoritmy nad zásobníkovými automaty

Formálně zapište algoritmus pro sestrojení průniku ZA a KA

- Vstup:
  1. ZA $A_1 = \left(Q_1, \Sigma_1, \Gamma_1, \delta_1, q_0^1, Z_0^1, F_1\right)$
  2. KA $A_2 = \left(Q_2, \Sigma_2, \delta_2, q_0^2, F_2\right)$
- 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 $\Sigma_1 \cap \Sigma_2 = \varnothing$
  3. $\Gamma = \Gamma_1$

Algoritmy nad zásobníkovými automaty

Formálně zapište algoritmus pro sestrojení průniku ZA a KA

- Vstup:
  1. ZA $A_1 = \left(Q_1, \Sigma_1, \Gamma_1, \delta_1, q_0^1, Z_0^1, F_1\right)$
  2. KA $A_2 = \left(Q_2, \Sigma_2, \delta_2, q_0^2, F_2\right)$
- 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 $\Sigma_1 \cap \Sigma_2 = \varnothing$
  3. $\Gamma = \Gamma_1$
  4. $q_0 = \left(q_0^1, q_0^2\right)$

Algoritmy nad zásobníkovými automaty

Formálně zapište algoritmus pro sestrojení průniku ZA a KA

- Vstup:
  1. ZA $A_1 = \left(Q_1, \Sigma_1, \Gamma_1, \delta_1, q_0^1, Z_0^1, F_1\right)$
  2. KA $A_2 = \left(Q_2, \Sigma_2, \delta_2, q_0^2, F_2\right)$
- 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 $\Sigma_1 \cap \Sigma_2 = \varnothing$
  3. $\Gamma = \Gamma_1$
  4. $q_0 = \left(q_0^1, q_0^2\right)$
  5. $Z_0 = Z_0^1$

Algoritmy nad zásobníkovými automaty

Formálně zapište algoritmus pro sestrojení průniku ZA a KA

- Vstup:
  1. ZA $A_1 = \left(Q_1, \Sigma_1, \Gamma_1, \delta_1, q_0^1, Z_0^1, F_1\right)$
  2. KA $A_2 = \left(Q_2, \Sigma_2, \delta_2, q_0^2, F_2\right)$
- 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 $\Sigma_1 \cap \Sigma_2 = \varnothing$
  3. $\Gamma = \Gamma_1$
  4. $q_0 = \left(q_0^1, q_0^2\right)$
  5. $Z_0 = Z_0^1$
  6. $F = F_1 \times F_2$

Algoritmy nad zásobníkovými automaty

Formálně zapište algoritmus pro sestrojení průniku ZA a KA

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

Algoritmy nad zásobníkovými automaty

Formálně zapište algoritmus pro sestrojení průniku ZA a KA

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

Algoritmy nad zásobníkovými automaty

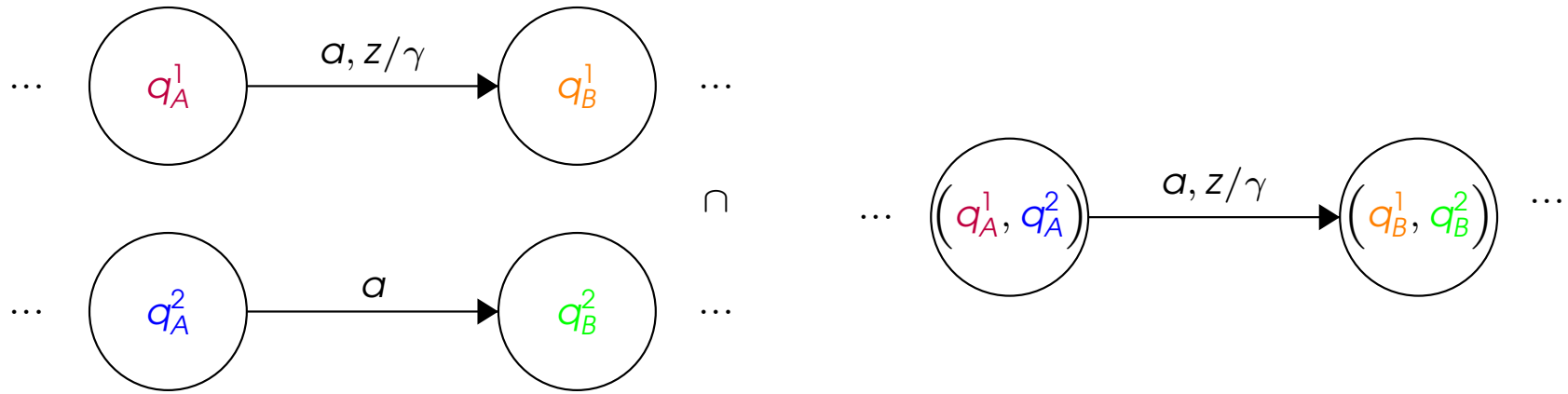
Formálně zapište algoritmus pro sestrojení průniku ZA a KA

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

Algoritmy nad zásobníkovými automaty

Formálně zapište algoritmus pro sestrojení průniku ZA a KA

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



Algoritmy nad zásobníkovými automaty

Formálně zapište algoritmus pro sestrojení průniku ZA a KA

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

Algoritmy nad zásobníkovými automaty

Formálně zapište algoritmus pro sestrojení průniku ZA a KA

8. $\forall q_A^1, q_B^1 \in Q_1 : \forall q_A^2, q_B^2 \in Q_2 : \forall z \in \Gamma : \forall \gamma \in \Gamma^* :$
$\left(\left(q_B^1, q_B^2\right), \gamma\right) \in \delta\left(\left(q_A^1, q_A^2\right), \epsilon, z\right) \Leftrightarrow$

Algoritmy nad zásobníkovými automaty

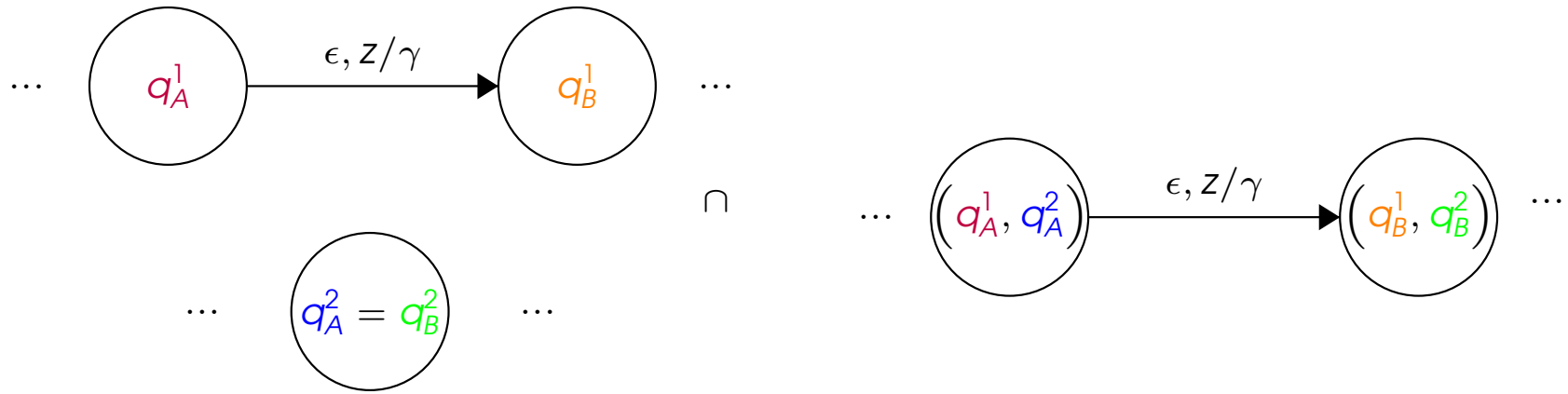
Formálně zapište algoritmus pro sestrojení průniku ZA a KA

8. $\forall q_A^1, q_B^1 \in Q_1 : \forall q_A^2, q_B^2 \in Q_2 : \forall z \in \Gamma : \forall \gamma \in \Gamma^* :$
$\left(\left(q_B^1, q_B^2\right), \gamma\right) \in \delta\left(\left(q_A^1, q_A^2\right), \epsilon, z\right) \Leftrightarrow$
$\left(\left(q_B^1, \gamma\right) \in \delta(q_A^1, \epsilon, z) \wedge q_A^2 = q_B^2\right)$

Algoritmy nad zásobníkovými automaty

Formálně zapište algoritmus pro sestrojení průniku ZA a KA

8. $\forall q_A^1, q_B^1 \in Q_1 : \forall q_A^2, q_B^2 \in Q_2 : \forall z \in \Gamma : \forall \gamma \in \Gamma^* :$
$\left(\left(q_B^1, q_B^2\right), \gamma\right) \in \delta\left(\left(q_A^1, q_A^2\right), \epsilon, z\right) \Leftrightarrow$
$\left(\left(q_B^1, \gamma\right) \in \delta(q_A^1, \epsilon, z) \wedge q_A^2 = q_B^2\right)$



Algoritmy nad zásobníkovými automaty

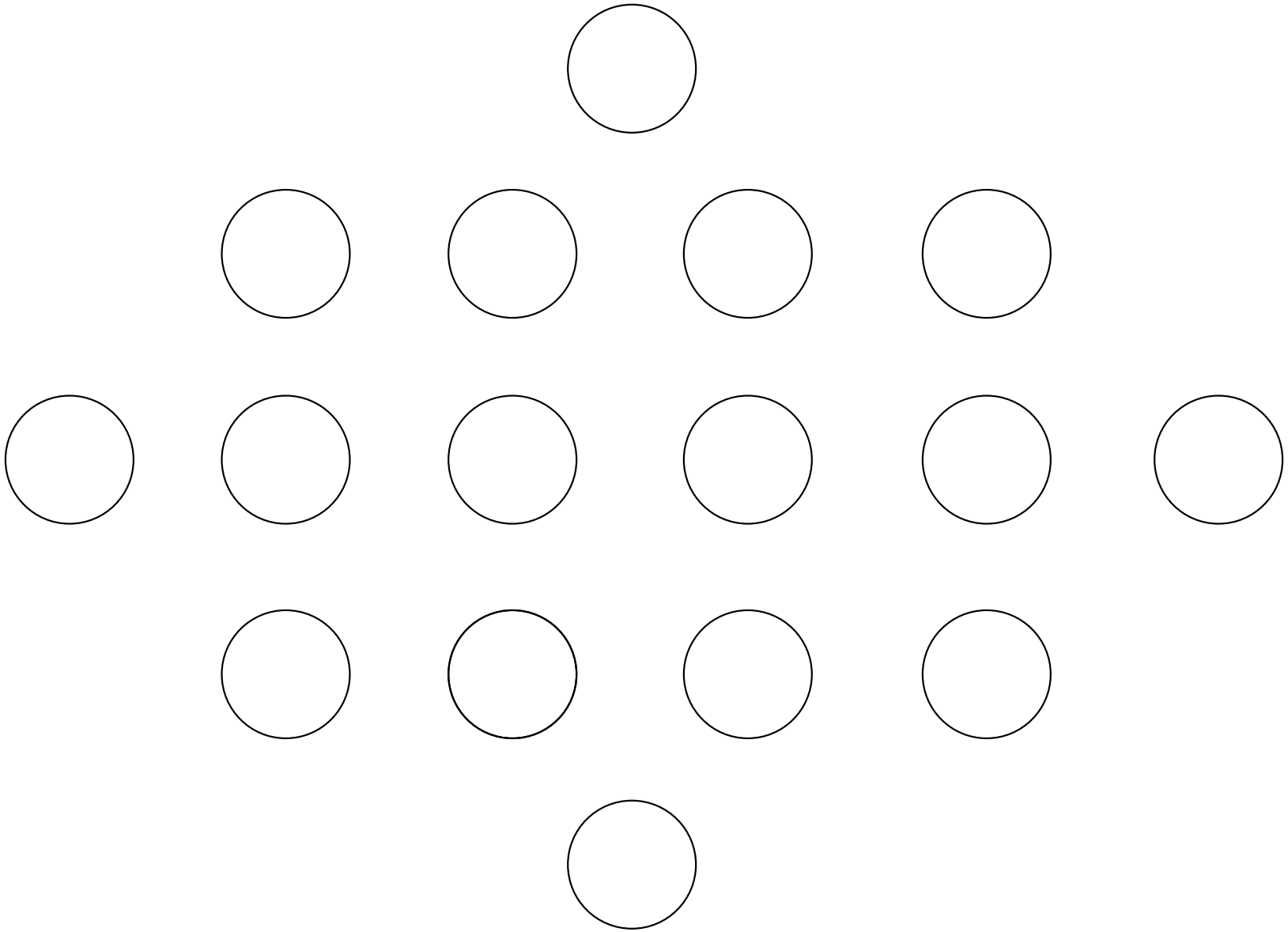
Formálně zapište algoritmus pro sestrojení průniku ZA a KA

Pozn.: nelze napsat

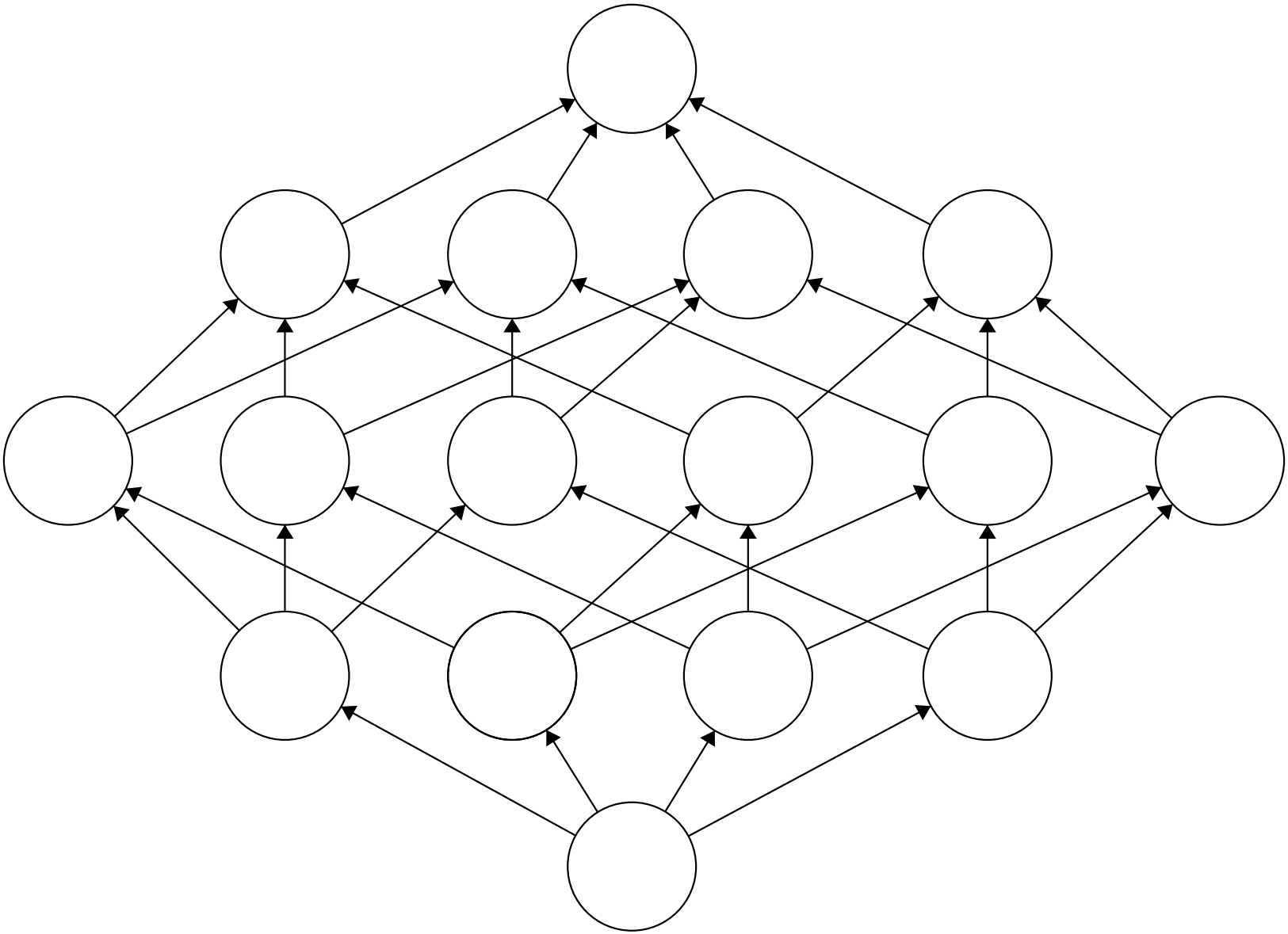
- $\forall q_A^1, q_B^1 \in Q_1 : \forall q \in Q_2 : \forall z \in \Gamma : \forall \gamma \in \Gamma^* :$
  $\left(\left(q_B^1, q\right), \gamma\right) \in \delta\left(\left(q_A^1, q\right), \epsilon, z\right) \Leftrightarrow \left(q_B^1, \gamma\right) \in \delta(q_A^1, \epsilon, z)$

Algoritmy zpřesňující odhad výsledku

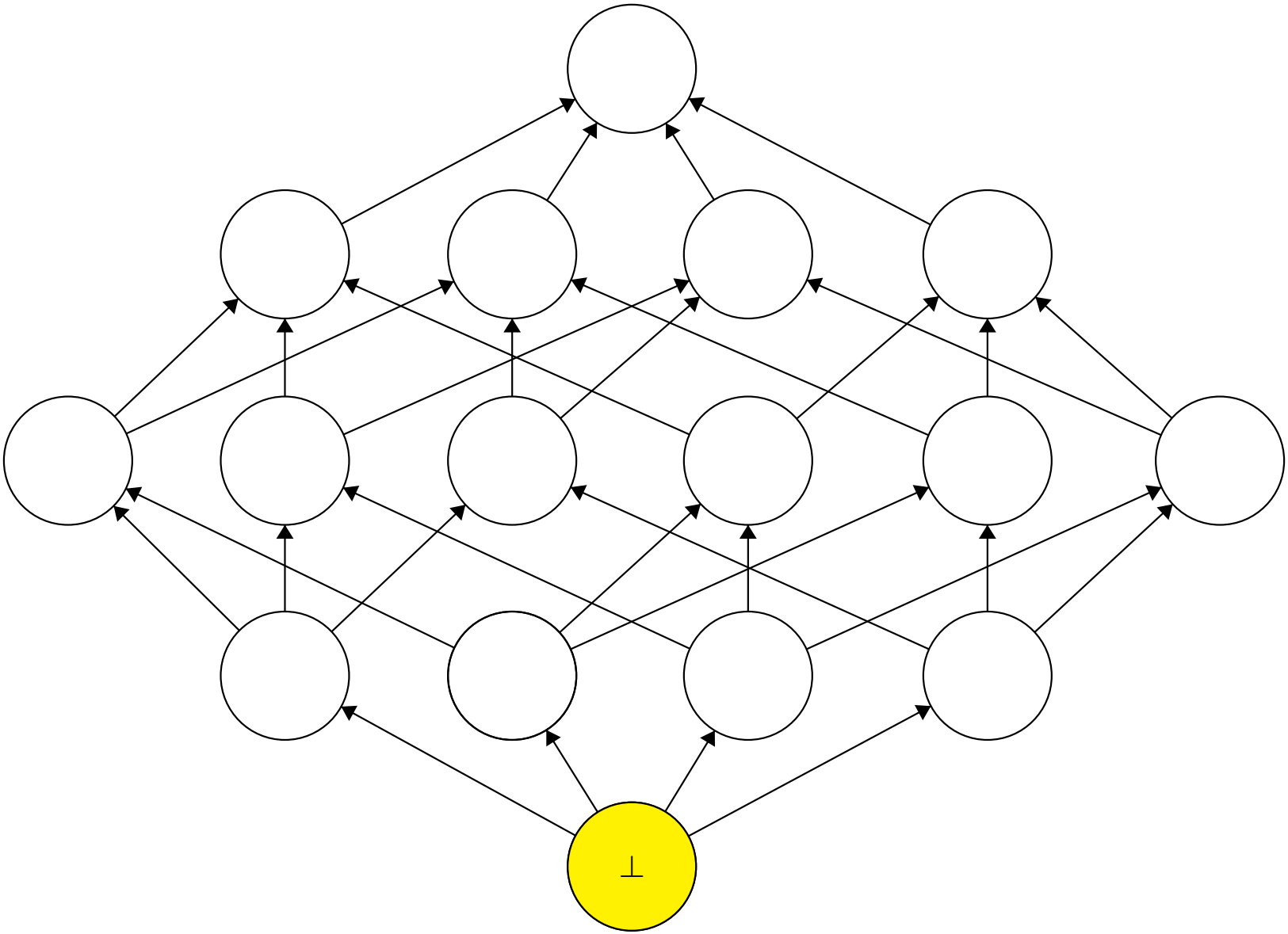
Algoritmy zpřesňující odhad výsledku



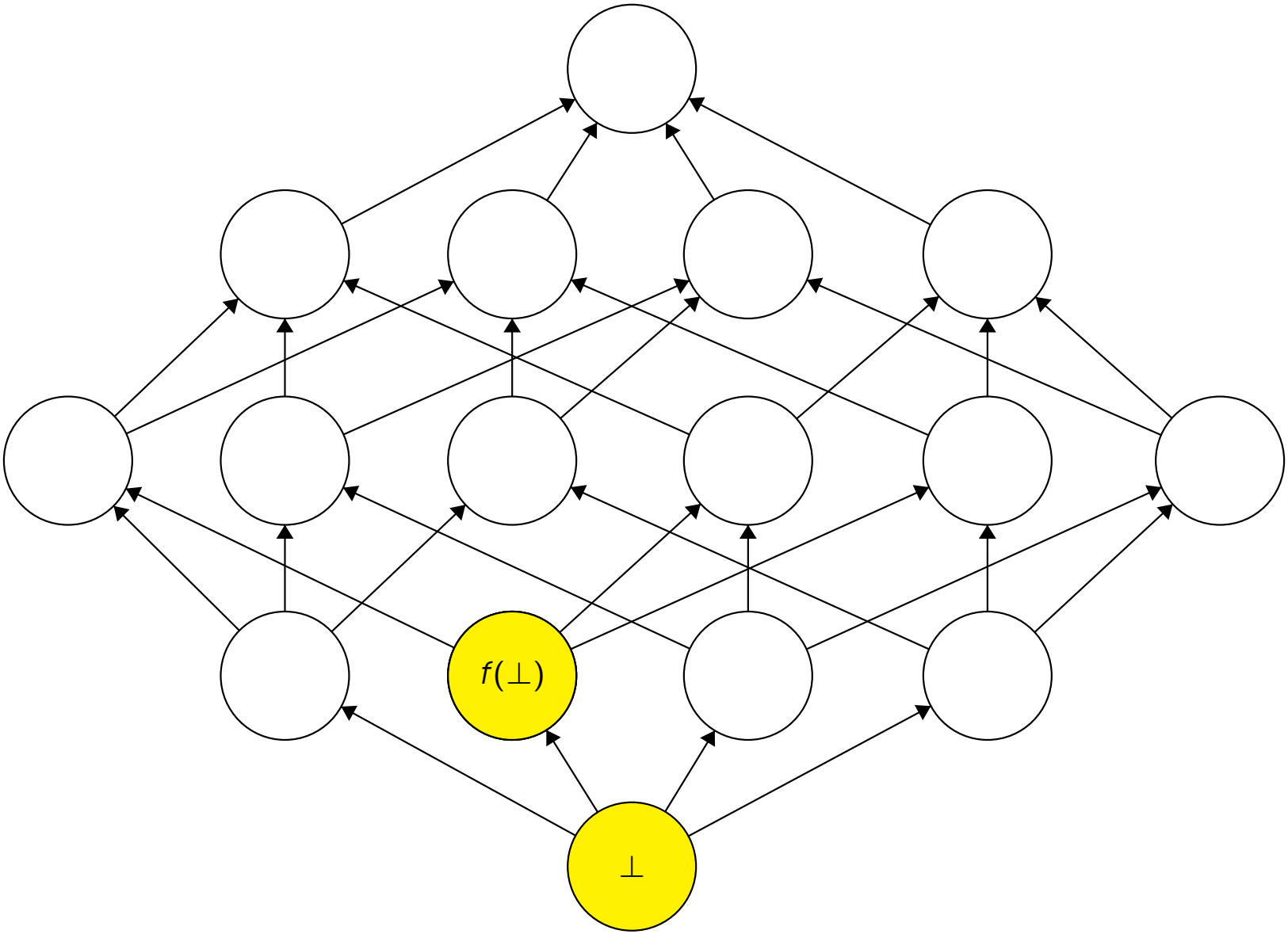
Algoritmy zpřesňující odhad výsledku



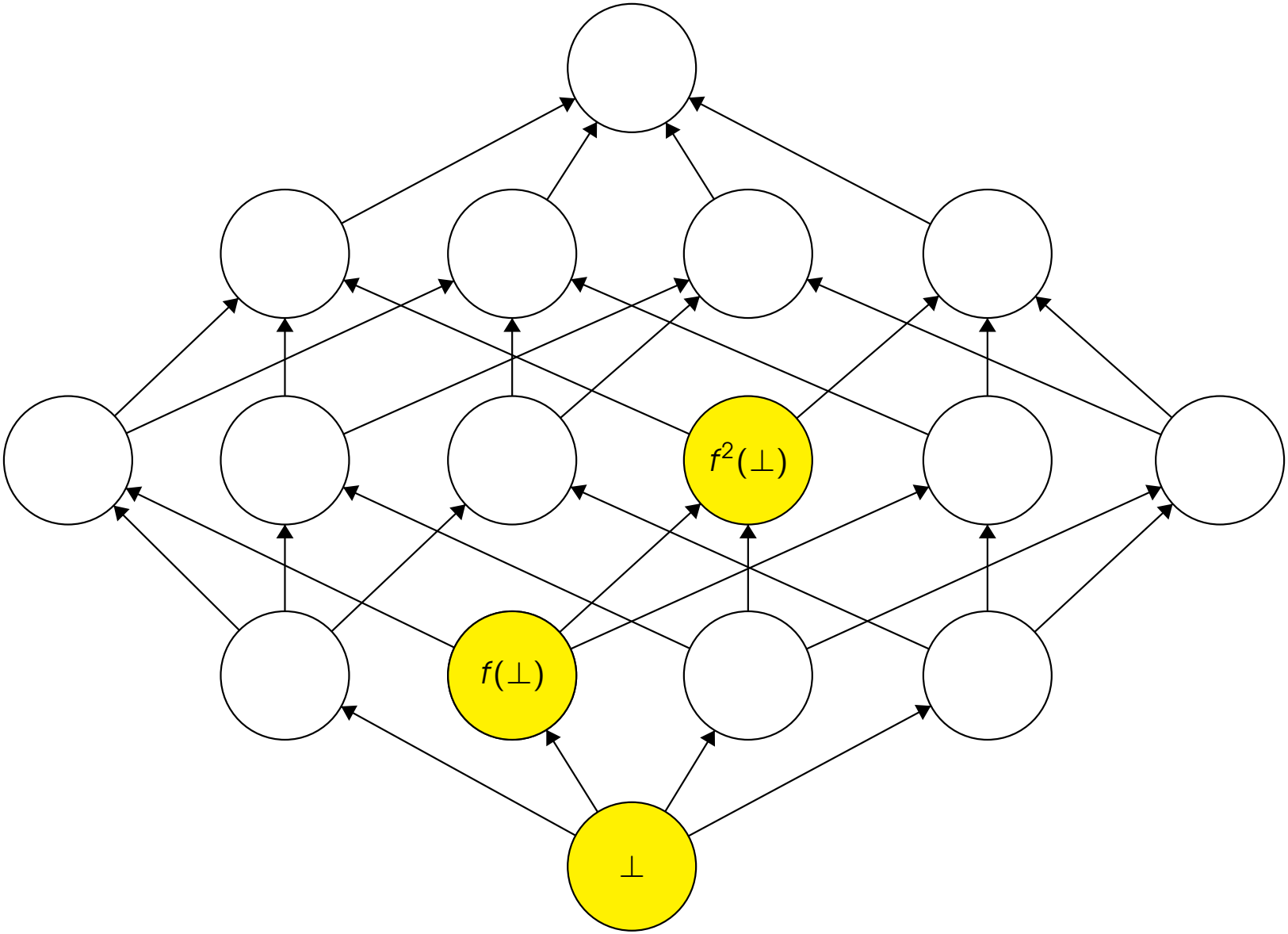
Algoritmy zpřesňující odhad výsledku



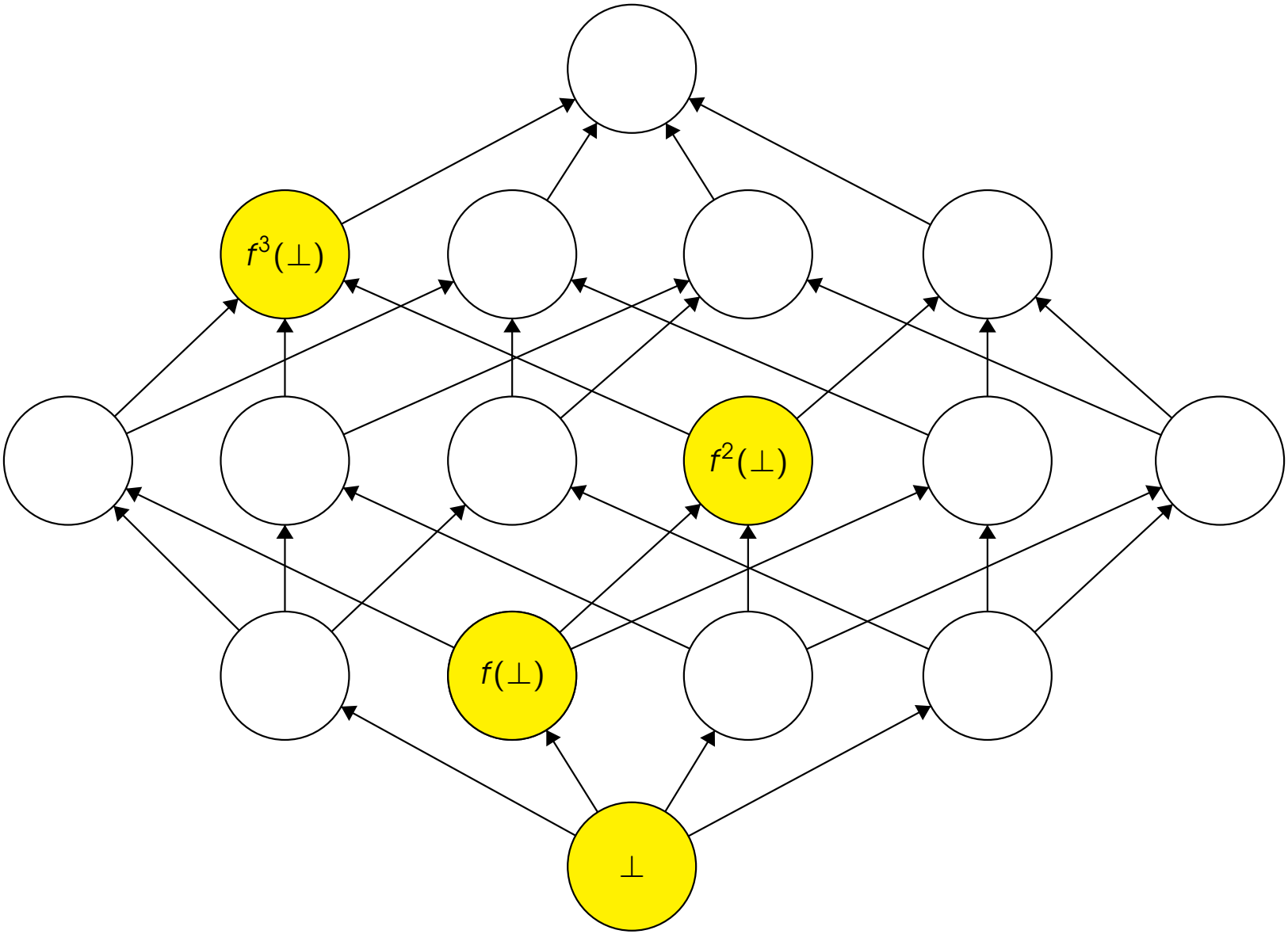
Algoritmy zpřesňující odhad výsledku



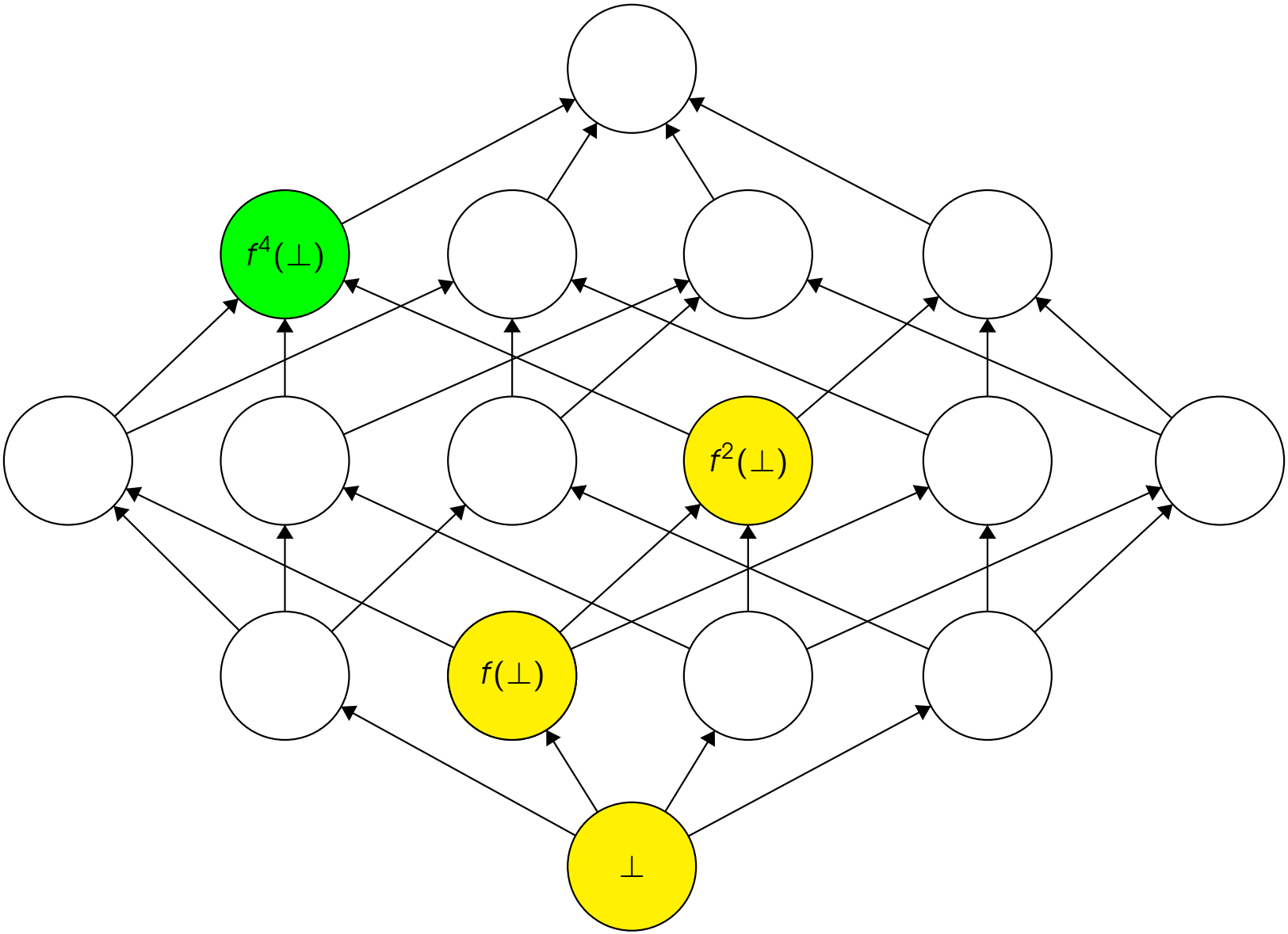
Algoritmy zpřesňující odhad výsledku



Algoritmy zpřesňující odhad výsledku



Algoritmy zpřesňující odhad výsledku



Poset

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

Algoritmy zpřesňující odhad výsledku

Poset

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

Pevný bod funkce

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

Algoritmy zpřesňující odhad výsledku

Poset

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

Úplný svaz

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

Pevný bod funkce

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

Algoritmy zpřesňující odhad výsledku

Poset

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

Úplný svaz

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

Pevný bod funkce

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

Monotónní funkce

- $(D, \sqsubseteq)$ je poset
- $f : D \to D$ 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 $f : D \to D$ **nad úplným svazem** $(D, \sqsubseteq, \sqcup, \sqcap, \top, \bot)$ **tvoří opět úplný svaz.**
(To garantuje existenci nejmenšího/největšího pevného bodu f)

Knaster-Tarskiho teorém

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

Nejmenší pevný bod

Knaster-Tarskiho teorém

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

Nejmenší pevný bod

- iniciální hodnota $\bot$

Knaster-Tarskiho teorém

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

Nejmenší pevný bod

- iniciální hodnota $\bot$
- f je mon. rostoucí

Knaster-Tarskiho teorém

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

Nejmenší pevný bod

- iniciální hodnota $\bot$
- f je mon. rostoucí
- $\bot \sqsubseteq f(\bot) \Rightarrow f(\bot) \sqsubseteq f(f(\bot))$

Knaster-Tarskiho teorém

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

Nejmenší pevný bod

- iniciální hodnota $\bot$
- f je mon. rostoucí
- $\bot \sqsubseteq f(\bot) \Rightarrow f(\bot) \sqsubseteq f(f(\bot))$
- postupně počítáme $f(\bot), f(f(\bot)), f(f(f(\bot))), \ldots$

Knaster-Tarskiho teorém

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

Nejmenší pevný bod

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

Knaster-Tarskiho teorém

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

Nejmenší pevný bod

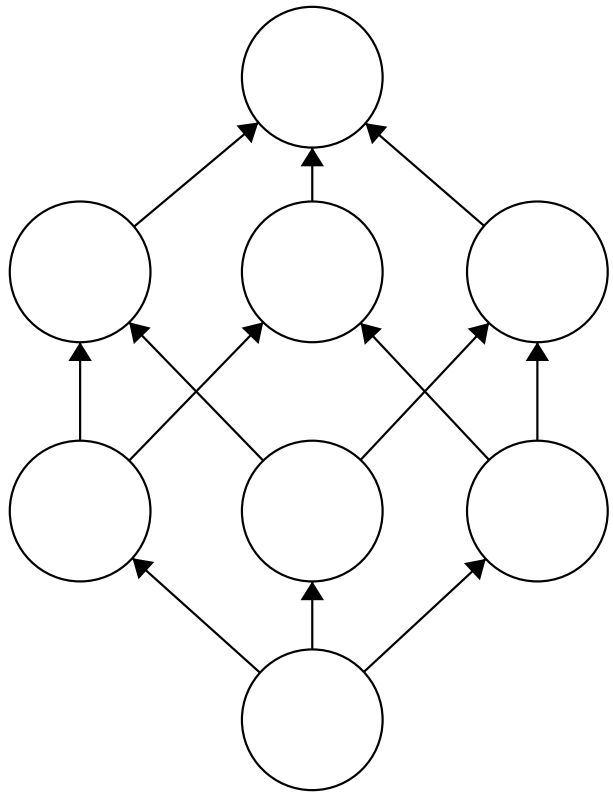
- iniciální hodnota $\bot$
- f je mon. rostoucí
- $\bot \sqsubseteq f(\bot) \Rightarrow f(\bot) \sqsubseteq f(f(\bot))$
- postupně počítáme $f(\bot), f(f(\bot)), f(f(f(\bot))), \ldots$
- $\exists k \in \mathbb{N} : f^k(\bot) = f^{k+1}(\bot)$
- $f^k(\bot)$ je nejmenší pevný bod zobrazení f

Knaster-Tarskiho teorém

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

Nejmenší pevný bod

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

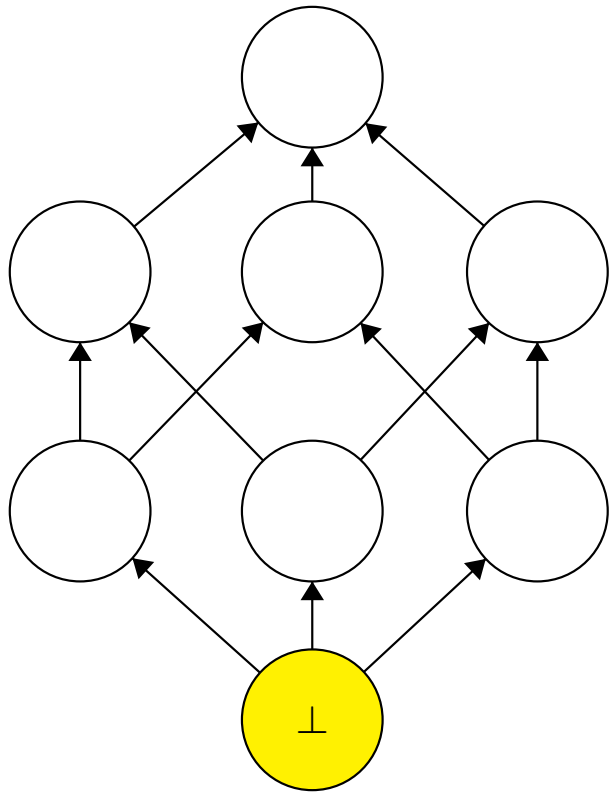


Knaster-Tarskiho teorém

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

Nejmenší pevný bod

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

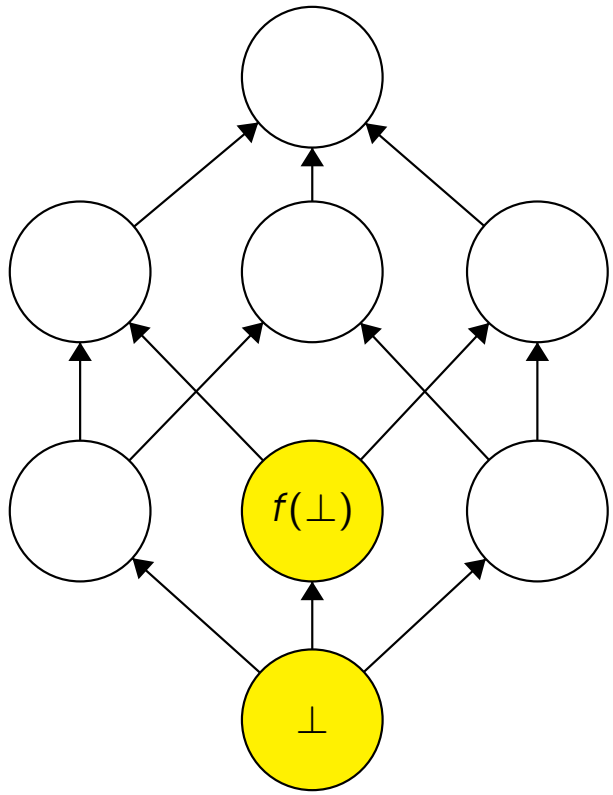


Knaster-Tarskiho teorém

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

Nejmenší pevný bod

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

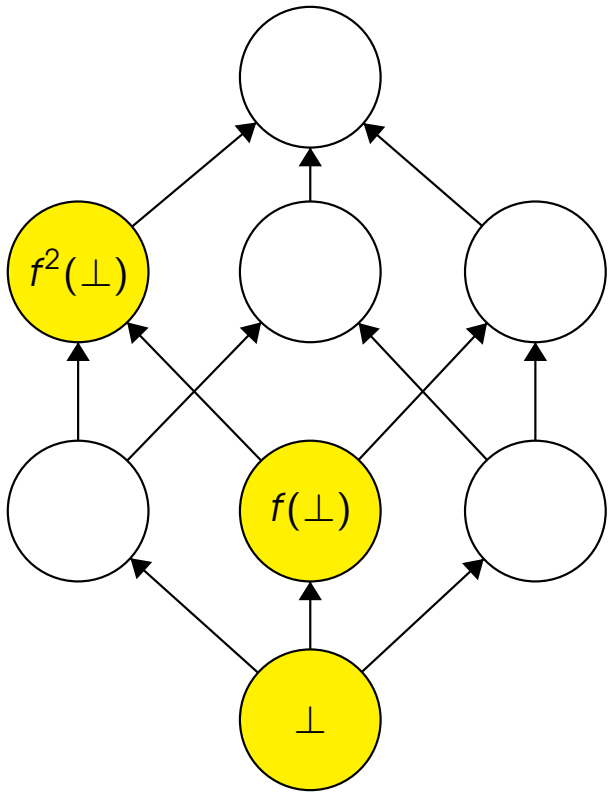


Knaster-Tarskiho teorém

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

Nejmenší pevný bod

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

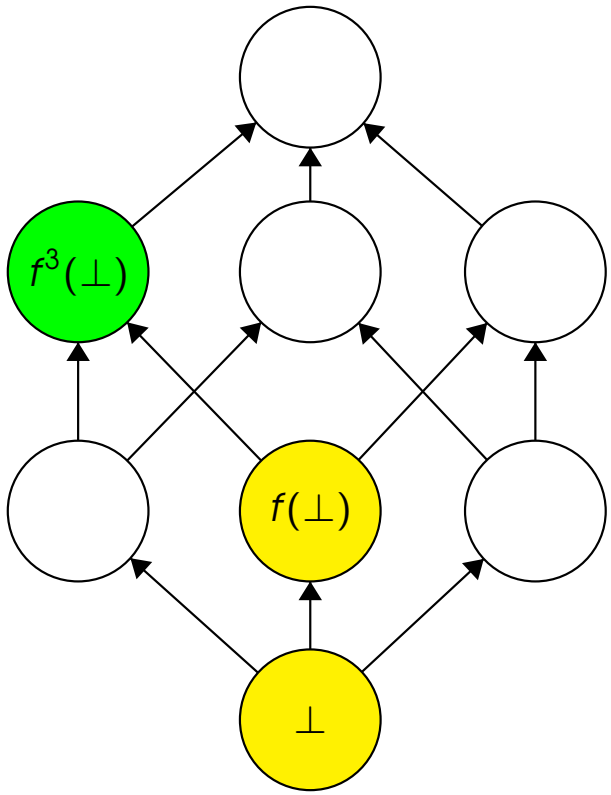


Knaster-Tarskiho teorém

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

Nejmenší pevný bod

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

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 $\bot = \varnothing$
- úplný svaz $(2^N, \subseteq, \cup, \cap, N, \varnothing)$

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 $\bot = \varnothing$
- úplný svaz $(2^N, \subseteq, \cup, \cap, N, \varnothing)$
- zpřesňující funkce
  $f : 2^N \to 2^N$ 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 $\bot = \varnothing$
- úplný svaz $(2^N, \subseteq, \cup, \cap, N, \varnothing)$
- zpřesňující funkce $f : 2^N \to 2^N$ mon. rostoucí
- $f(X) = \{A \in N \mid \exists (A, \alpha) \in P : \alpha \in X^*\}$

Konstrukce algoritmu zpřesňujícího odhad výsledku

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

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

Analýza

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

Příklad 3

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 $\perp = \varnothing$
- úplný svaz $(2^N, \subseteq, \cup, \cap, N, \varnothing)$
- zpřesňující funkce $f : 2^N \to 2^N$ mon. rostoucí
- $f(X) = \{A \in N \mid \exists (A, \alpha) \in P : \alpha \in X^*\}$
- hledáme nejmenší pevný bod $f^k(\varnothing)$ 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 $\perp = \varnothing$
- úplný svaz $(2^N, \subseteq, \cup, \cap, N, \varnothing)$
- zpřesňující funkce $f : 2^N \to 2^N$ mon. rostoucí
- $f(X) = \{A \in N \mid \exists (A, \alpha) \in P : \alpha \in X^*\}$
- hledáme nejmenší pevný bod $f^k(\varnothing)$ 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 $\bot = \varnothing$
- úplný svaz $(2^N, \subseteq, \cup, \cap, N, \varnothing)$
- zpřesňující funkce $f : 2^N \to 2^N$ mon. rostoucí
- $f(X) = \{A \in N \mid \exists (A, \alpha) \in P : \alpha \in X^*\}$
- hledáme nejmenší pevný bod $f^k(\varnothing)$ 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 $\perp = \varnothing$
- úplný svaz $(2^N, \subseteq, \cup, \cap, N, \varnothing)$
- zpřesňující funkce $f : 2^N \to 2^N$ mon. rostoucí
- $f(X) = \{A \in N \mid \exists (A, \alpha) \in P : \alpha \in X^*\}$
- hledáme nejmenší pevný bod $f^k(\varnothing)$ 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 $\bot = \varnothing$
- úplný svaz $(2^N, \subseteq, \cup, \cap, N, \varnothing)$
- zpřesňující funkce $f : 2^N \to 2^N$ mon. rostoucí
- $f(X) = \{A \in N \mid \exists (A, \alpha) \in P : \alpha \in X^*\}$
- hledáme nejmenší pevný bod $f^k(\varnothing)$ 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. $N_\epsilon^0 = \varnothing, i = 0$

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 $\bot = \varnothing$
- úplný svaz $(2^N, \subseteq, \cup, \cap, N, \varnothing)$
- zpřesňující funkce $f : 2^N \to 2^N$ mon. rostoucí
- $f(X) = \{A \in N \mid \exists (A, \alpha) \in P : \alpha \in X^*\}$
- hledáme nejmenší pevný bod $f^k(\varnothing)$ 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. $N_\epsilon^0 = \varnothing, i = 0$
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 $\perp = \varnothing$
- úplný svaz $(2^N, \subseteq, \cup, \cap, N, \varnothing)$
- zpřesňující funkce $f : 2^N \to 2^N$ mon. rostoucí
- $f(X) = \{A \in N \mid \exists (A, \alpha) \in P : \alpha \in X^*\}$
- hledáme nejmenší pevný bod $f^k(\varnothing)$ 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. $N_\epsilon^0 = \varnothing, i = 0$
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 $\bot = \varnothing$
- úplný svaz $(2^N, \subseteq, \cup, \cap, N, \varnothing)$
- zpřesňující funkce $f : 2^N \to 2^N$ mon. rostoucí
- $f(X) = \{A \in N \mid \exists (A, \alpha) \in P : \alpha \in X^*\}$
- hledáme nejmenší pevný bod $f^k(\varnothing)$ 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. $N_\epsilon^0 = \varnothing, i = 0$
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. $N_\epsilon = N_\epsilon^i$

Příklad 3

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

Příklad 3

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. $N_\epsilon^0 = \varnothing, i = 0$
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

Příklad 3

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. $N_\epsilon^0 = \varnothing, i = 0$
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

- $N_\epsilon^0 = \varnothing$

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

Demonstrace

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

Příklad 3

Gramatika

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

Algoritmus

Postup:

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

Demonstrace

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

Příklad 3

Gramatika

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

Algoritmus

Postup:

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

Demonstrace

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

Příklad 3

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

Demonstrace

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

Gramatika

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

Algoritmus

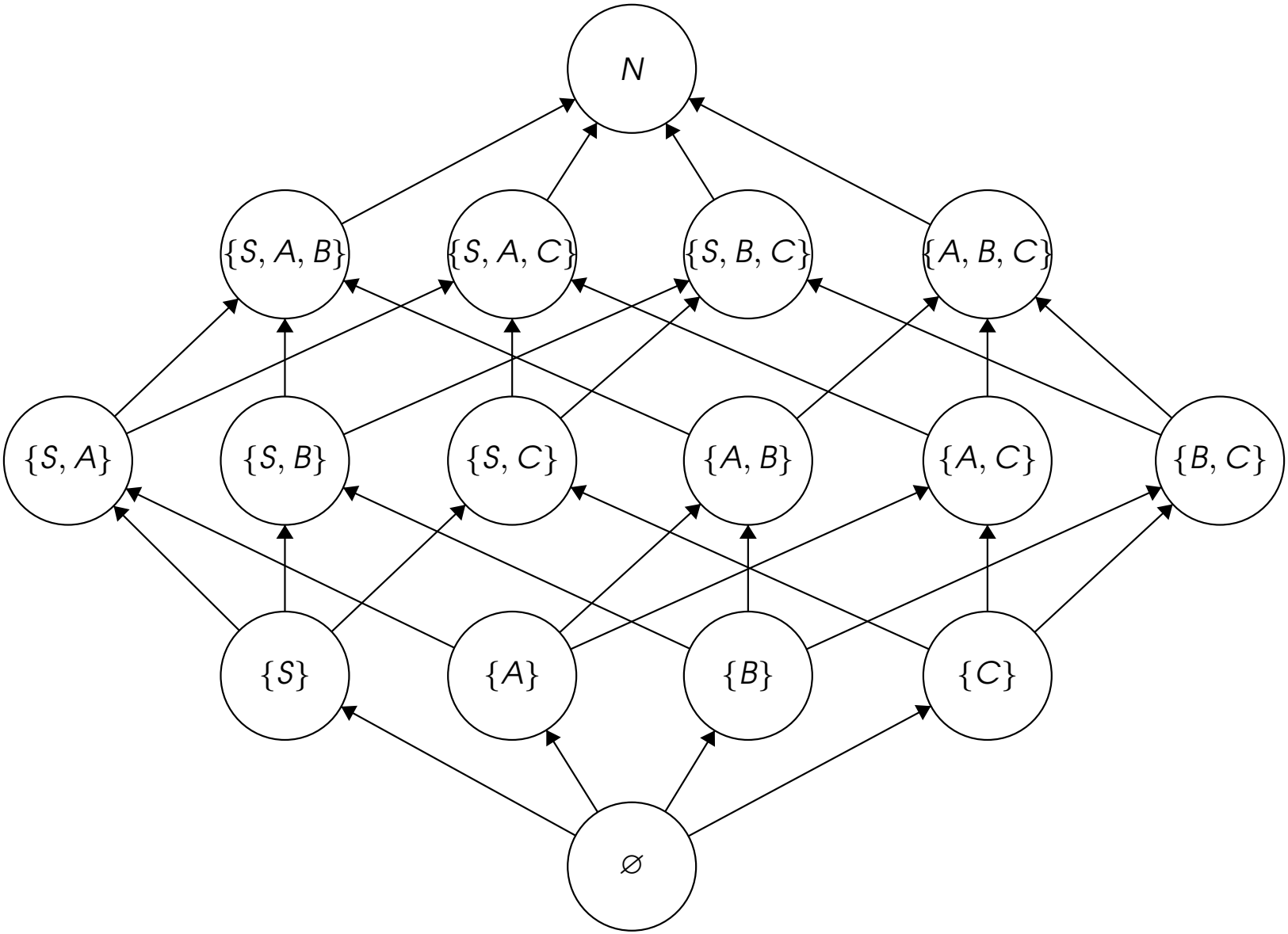
Postup:

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

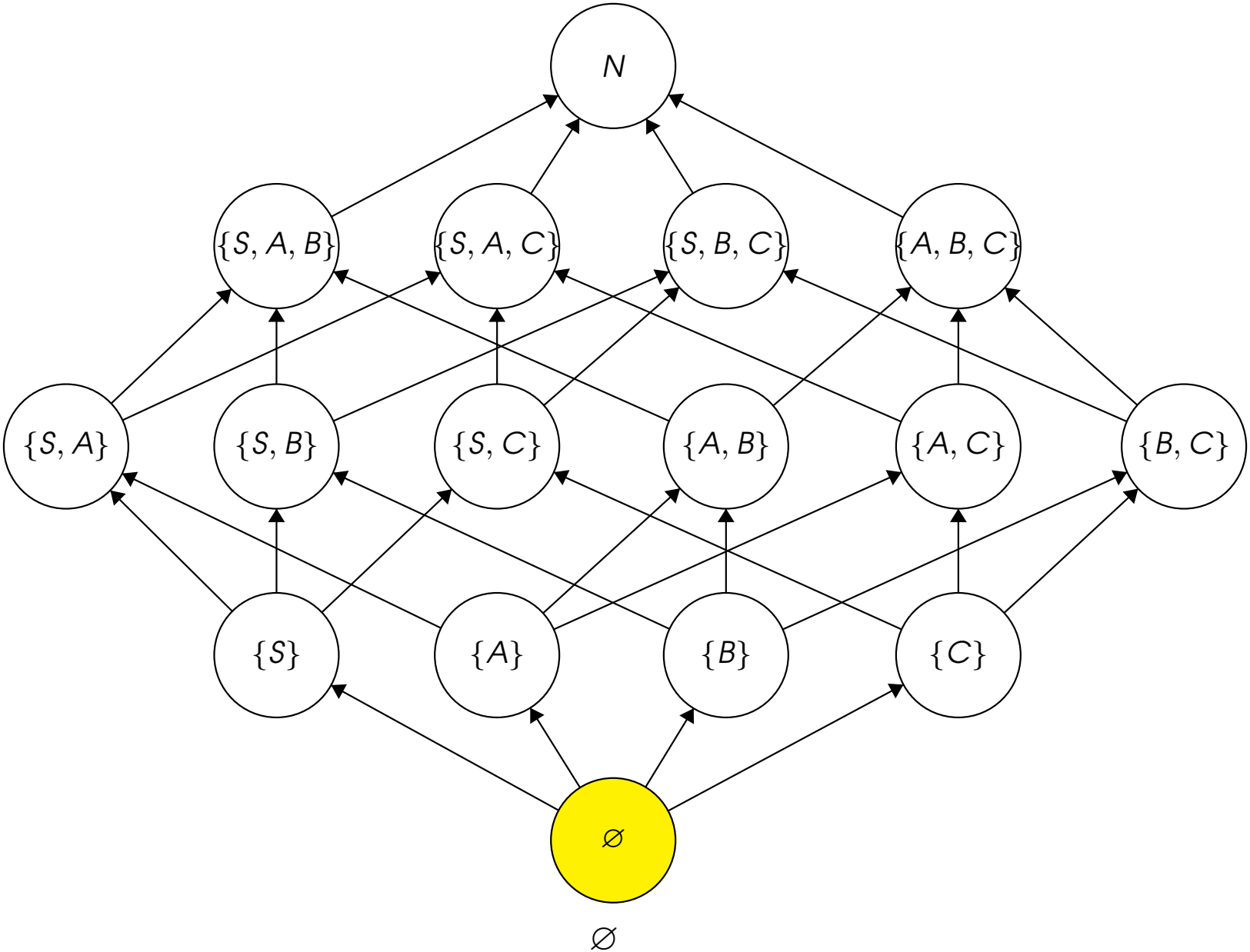
Demonstrace

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

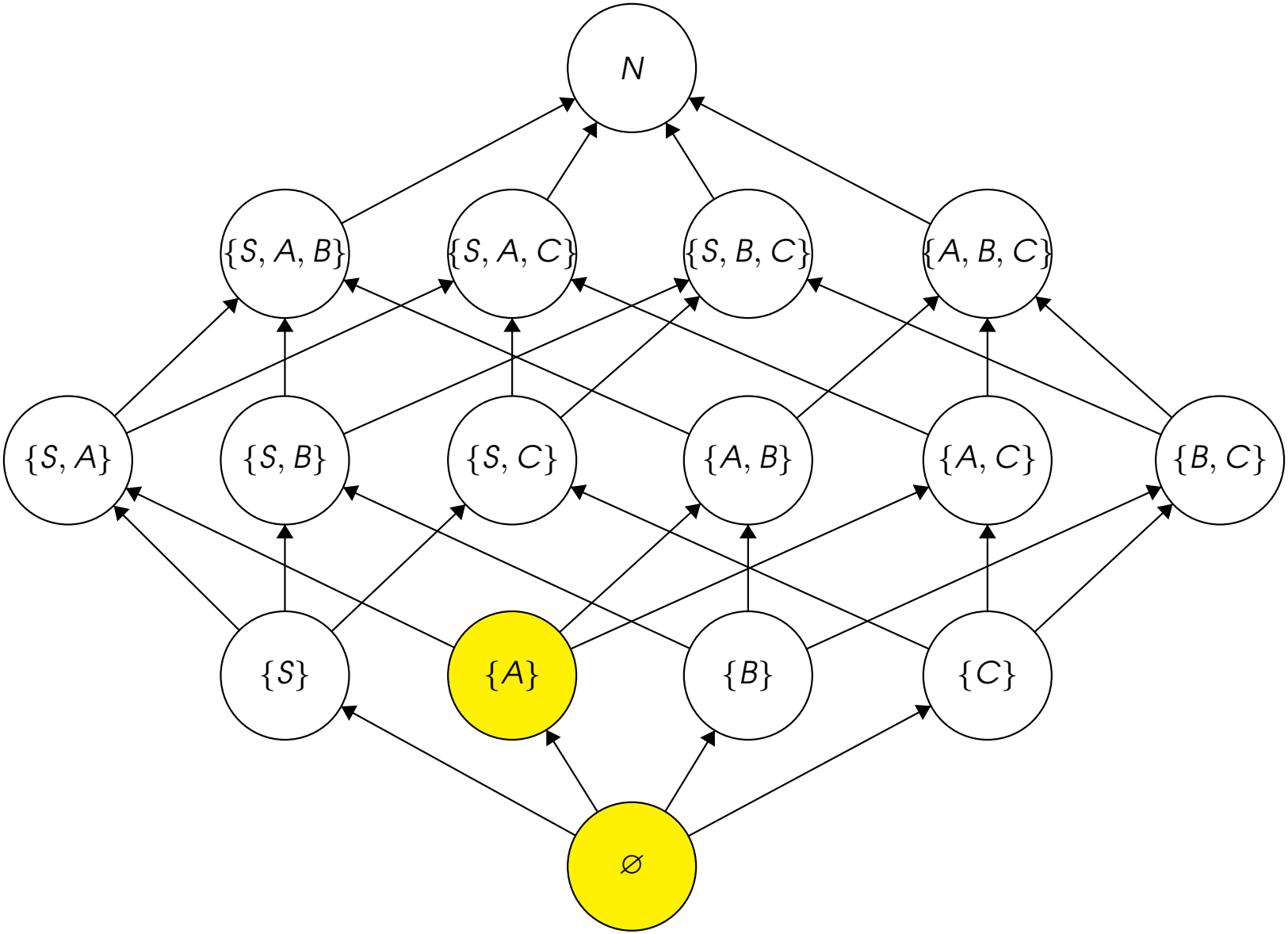
Příklad 3



Příklad 3

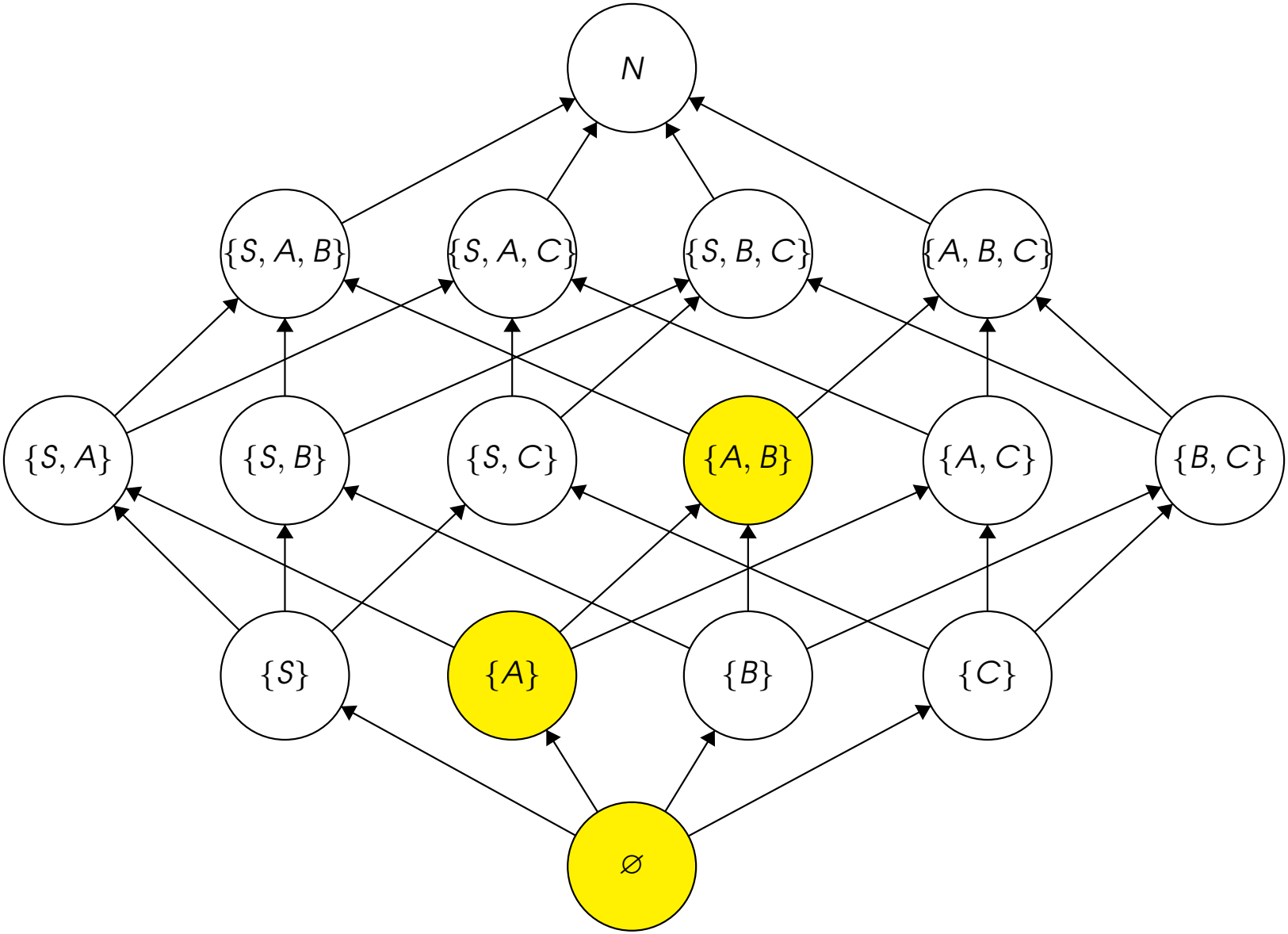


Příklad 3



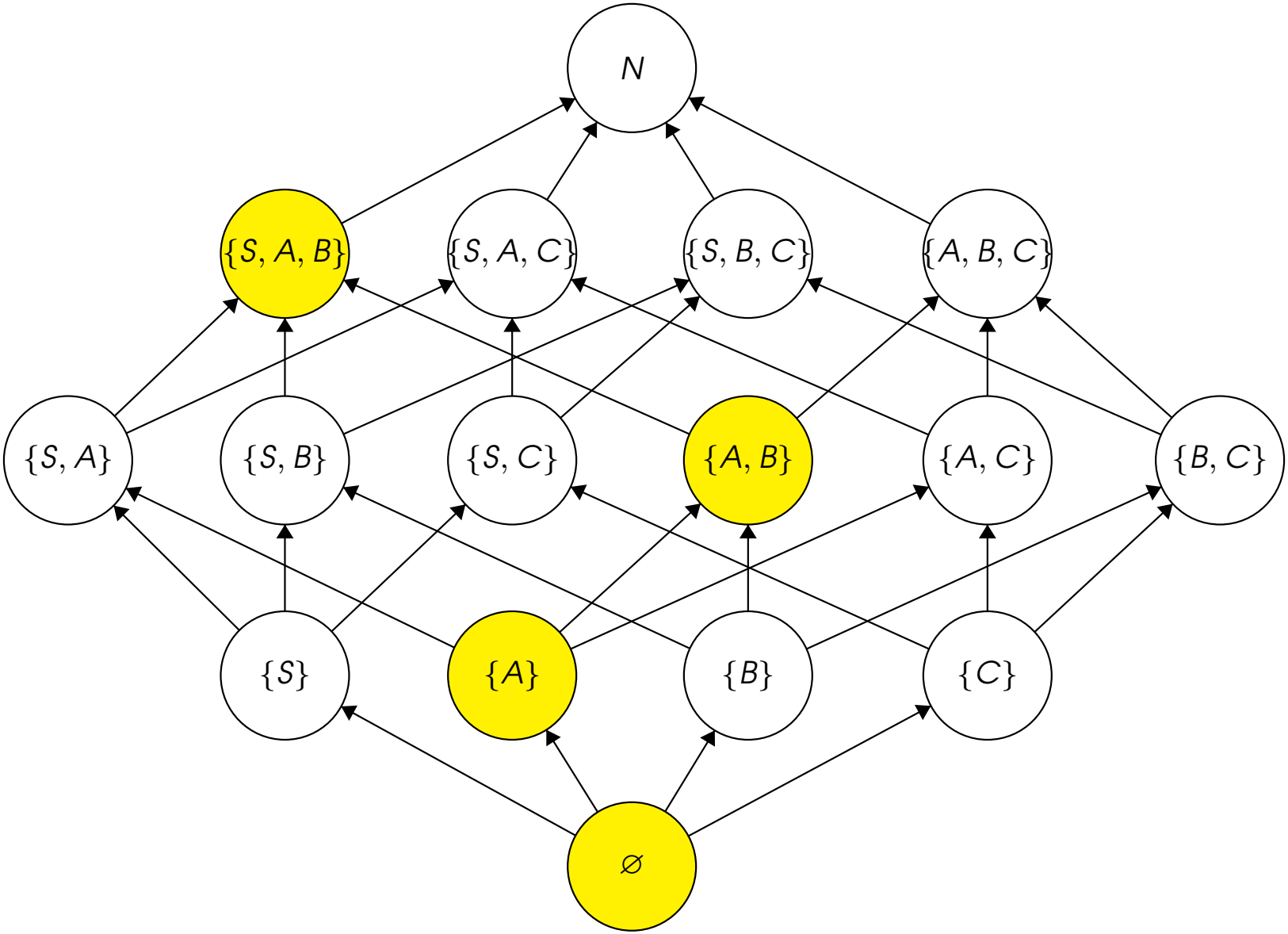
$$f(\varnothing) = \{A\}$$

Příklad 3



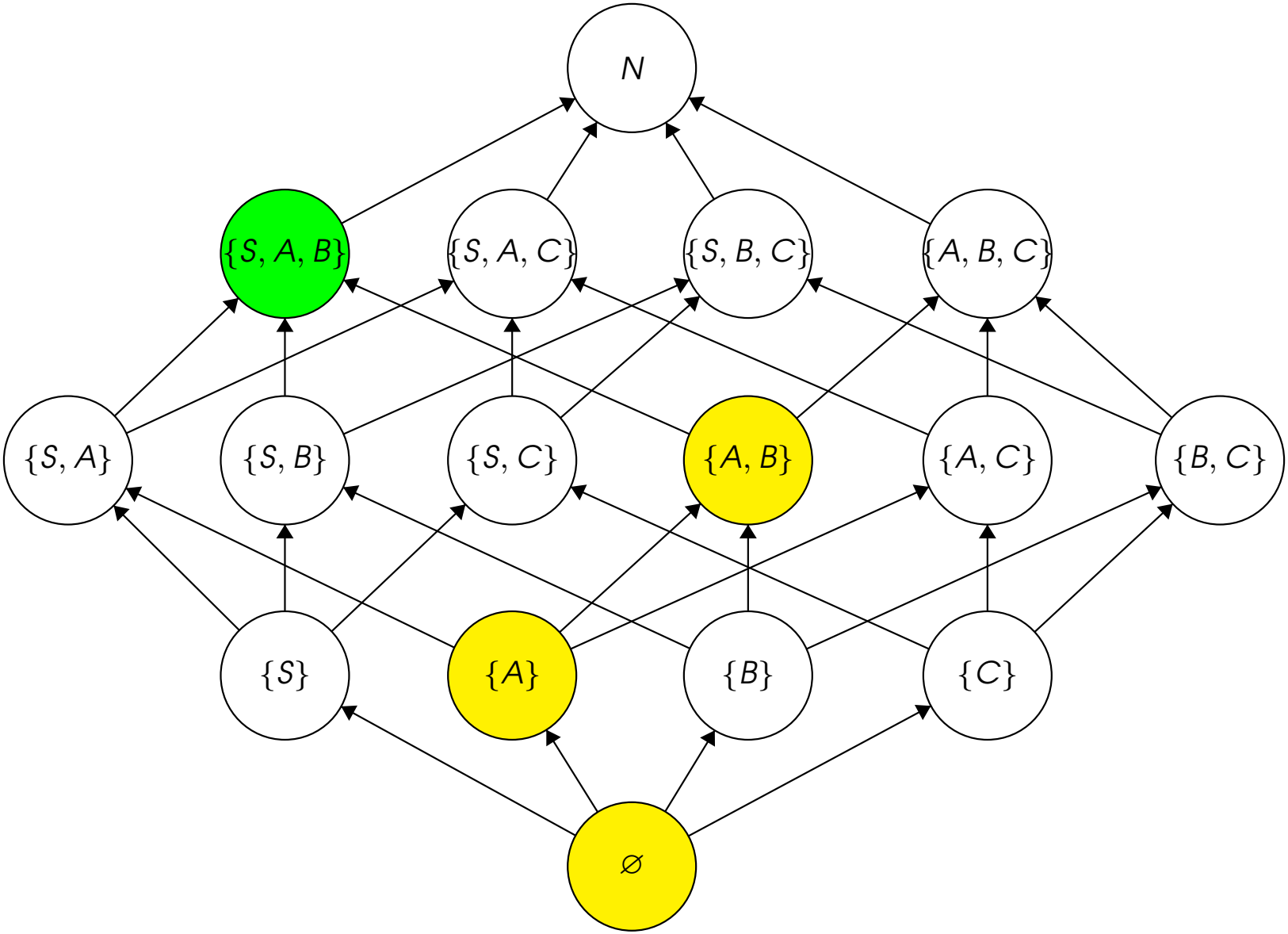
$$f^2(\varnothing) = f(\{A\}) = \{A, B\}$$

Příklad 3



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

Příklad 3



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

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

- Hledáme pravidla typu $A \to \alpha\beta\gamma$, 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

- Hledáme pravidla typu $A \to \alpha\beta\gamma$, 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

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

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 $\bot = \varnothing$

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 $\bot = \varnothing$
- úplný svaz $(2^N, \subseteq, \cup, \cap, N, \varnothing)$

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 $\bot = \varnothing$
- úplný svaz $(2^N, \subseteq, \cup, \cap, N, \varnothing)$
- zpřesňující funkce $f : 2^N \to 2^N$ 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 $\bot = \varnothing$
- úplný svaz $(2^N, \subseteq, \cup, \cap, N, \varnothing)$
- zpřesňující funkce $f : 2^N \to 2^N$ mon. rostoucí
- $f(X) = \{A \in N \mid \exists (A, \alpha\beta\gamma) \in P : \alpha \in (N_t \cup \Sigma)^*, \beta \in (X \cup \{a\}) \wedge \gamma \in N_\epsilon^*\}$

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 $\perp = \varnothing$
- úplný svaz $(2^N, \subseteq, \cup, \cap, N, \varnothing)$
- zpřesňující funkce $f : 2^N \to 2^N$ 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 $f^k(\varnothing)$ 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. $N_{wa}^0 = \varnothing, i = 0$
  2. $N_{wa}^{i+1} = \{A \in N \mid \exists (A \to \alpha) \in P : \alpha \in (N_t \cup \Sigma)^* (N_{wa}^i \cup \{a\}) N_\epsilon^*\}$
  3. pokud $N_{wa}^{i+1} \neq N_{wa}^i$
     - $i = i + 1$
     - přeji 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. $C \rightarrow aBA \mid Bc$

Příklad 4

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

Příklad 4

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. $C \rightarrow aBA \mid Bc$

Množiny

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

Algoritmus

Postup:

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

Příklad 4

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

Množiny

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

Demonstrace

- $N^0_{wa} = \varnothing$

Příklad 4

Gramatika

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

Algoritmus

Postup:

1. $N_{wa}^0 = \varnothing, i = 0$
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

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

Příklad 4

Gramatika

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

Algoritmus

Postup:

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

Množiny

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

Demonstrace

- $N^2_{wa} = \{B, C\}$

Příklad 4

Gramatika

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

Algoritmus

Postup:

1. $N_{wa}^0 = \varnothing, i = 0$
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

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

Příklad 4

Gramatika

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

Algoritmus

Postup:

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

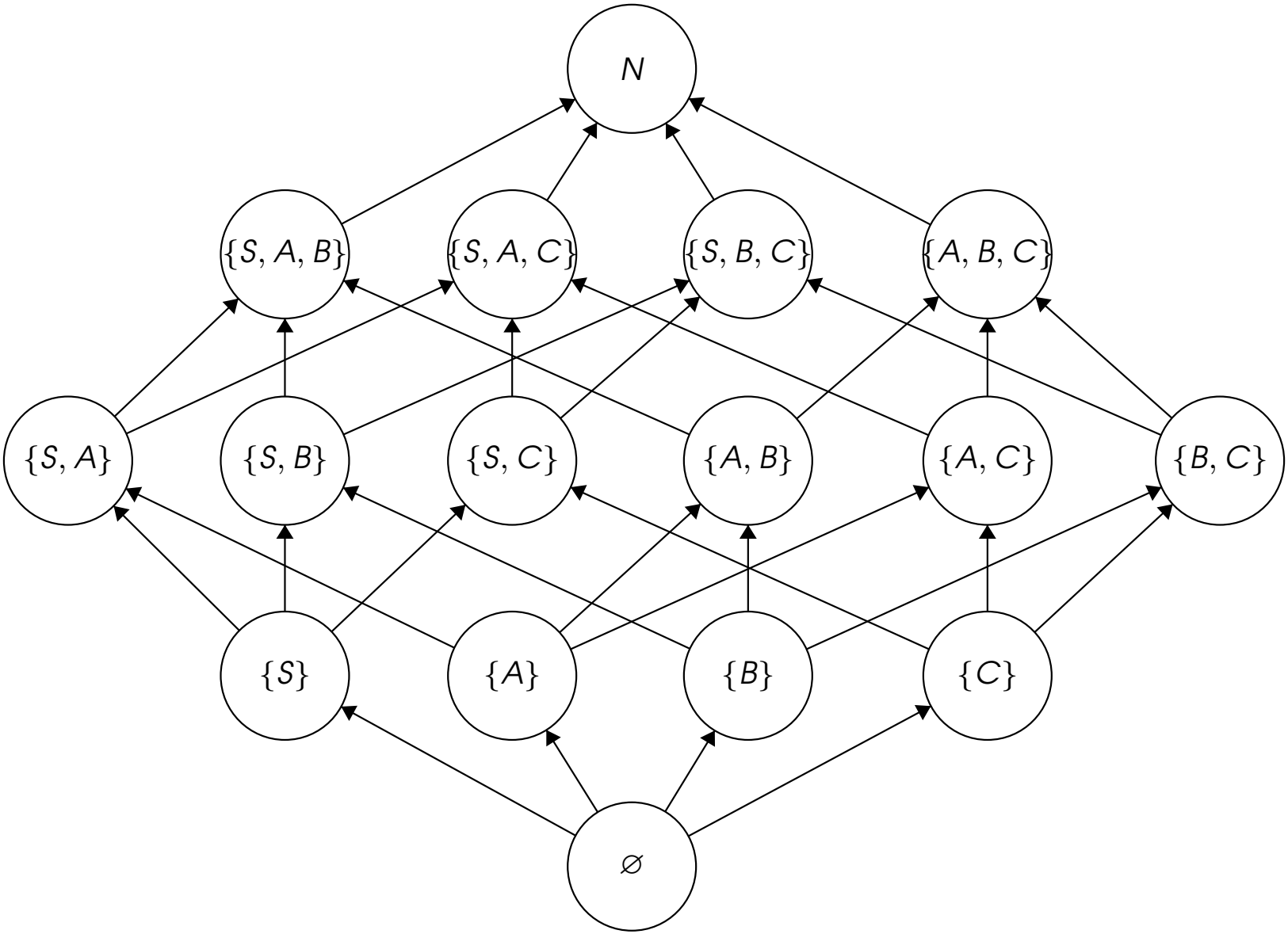
Množiny

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

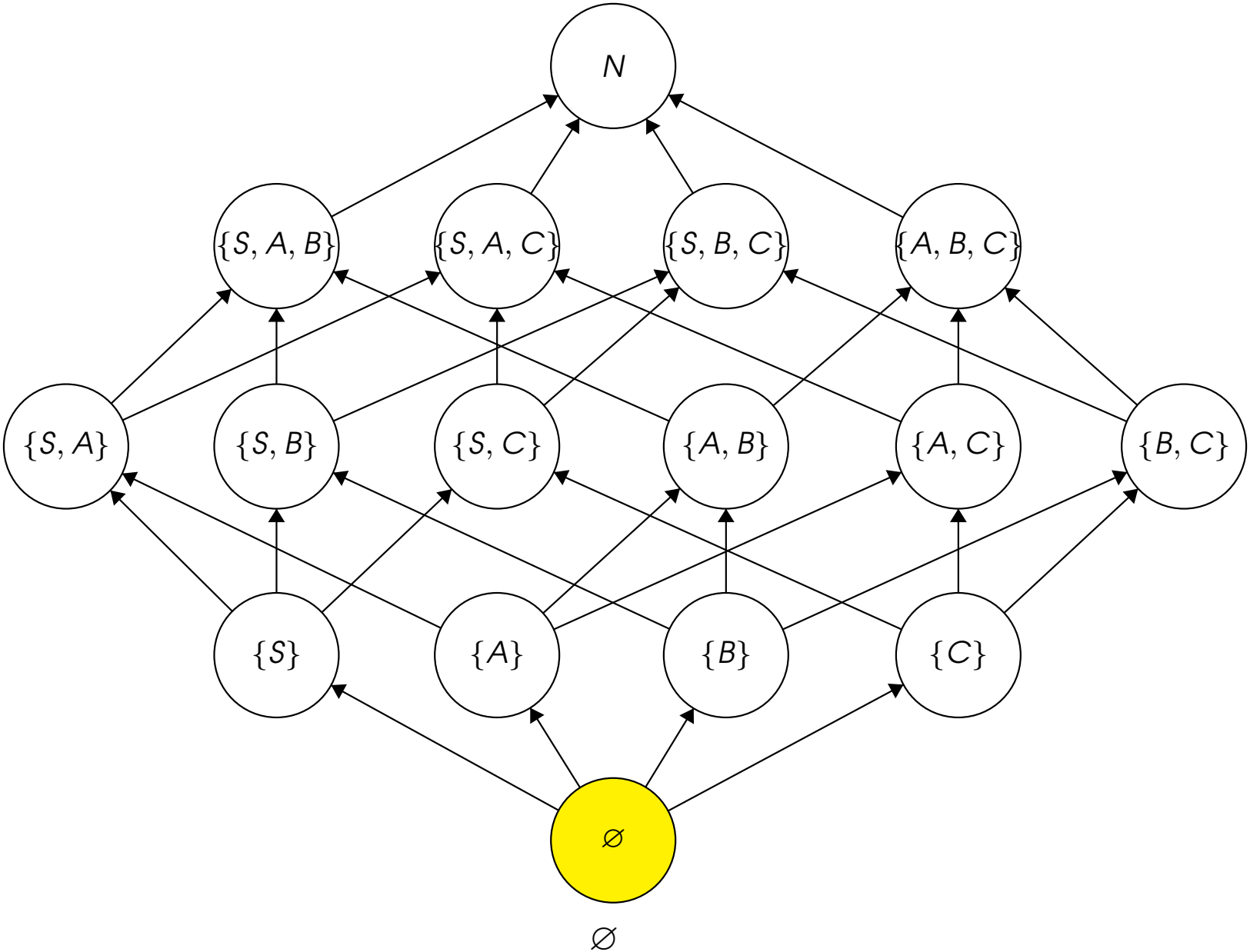
Demonstrace

- $N^4_{wa} = \{B, C, S\} = N^3_{wa} = N_{wa}$

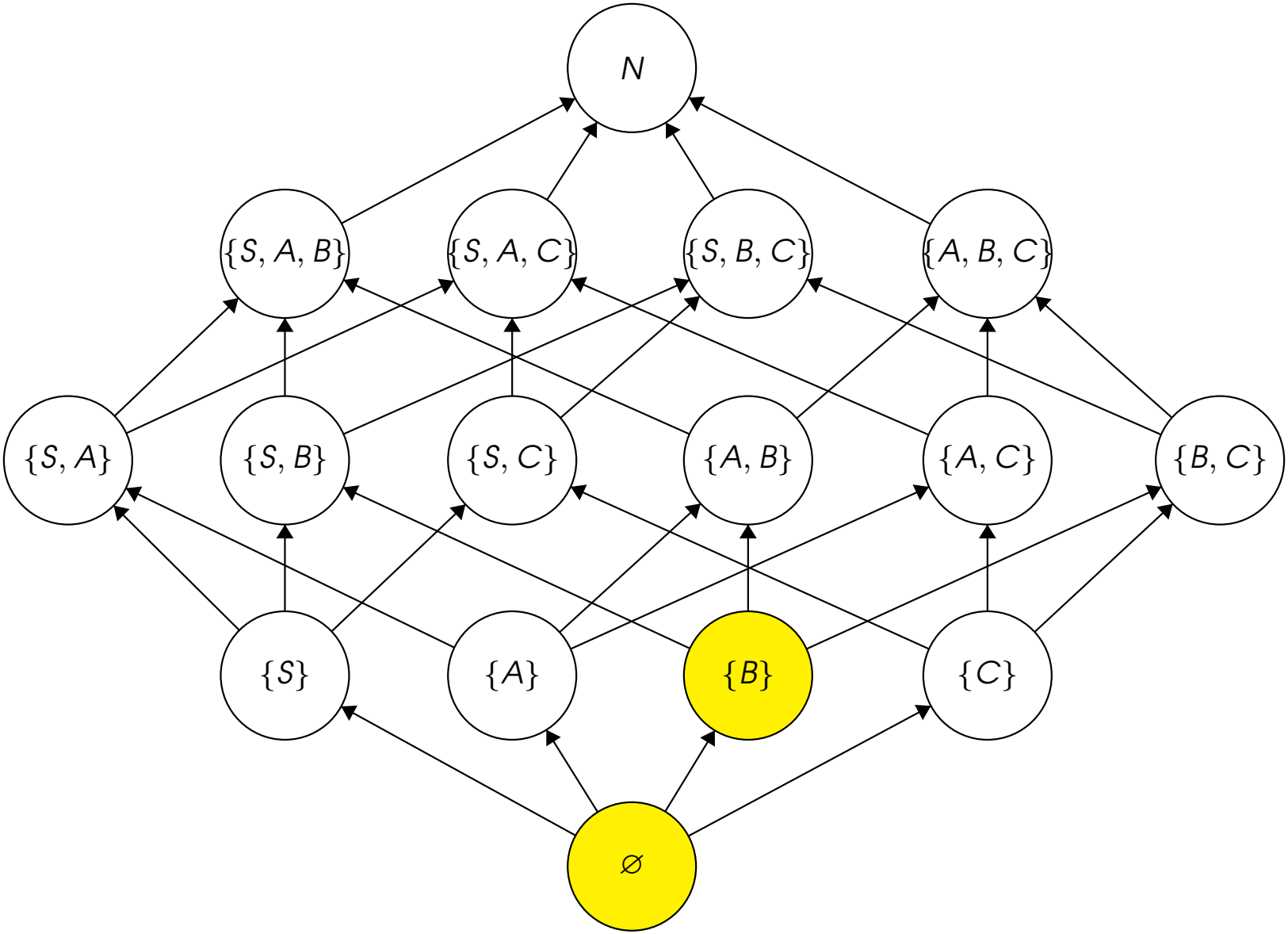
Příklad 4



Příklad 4

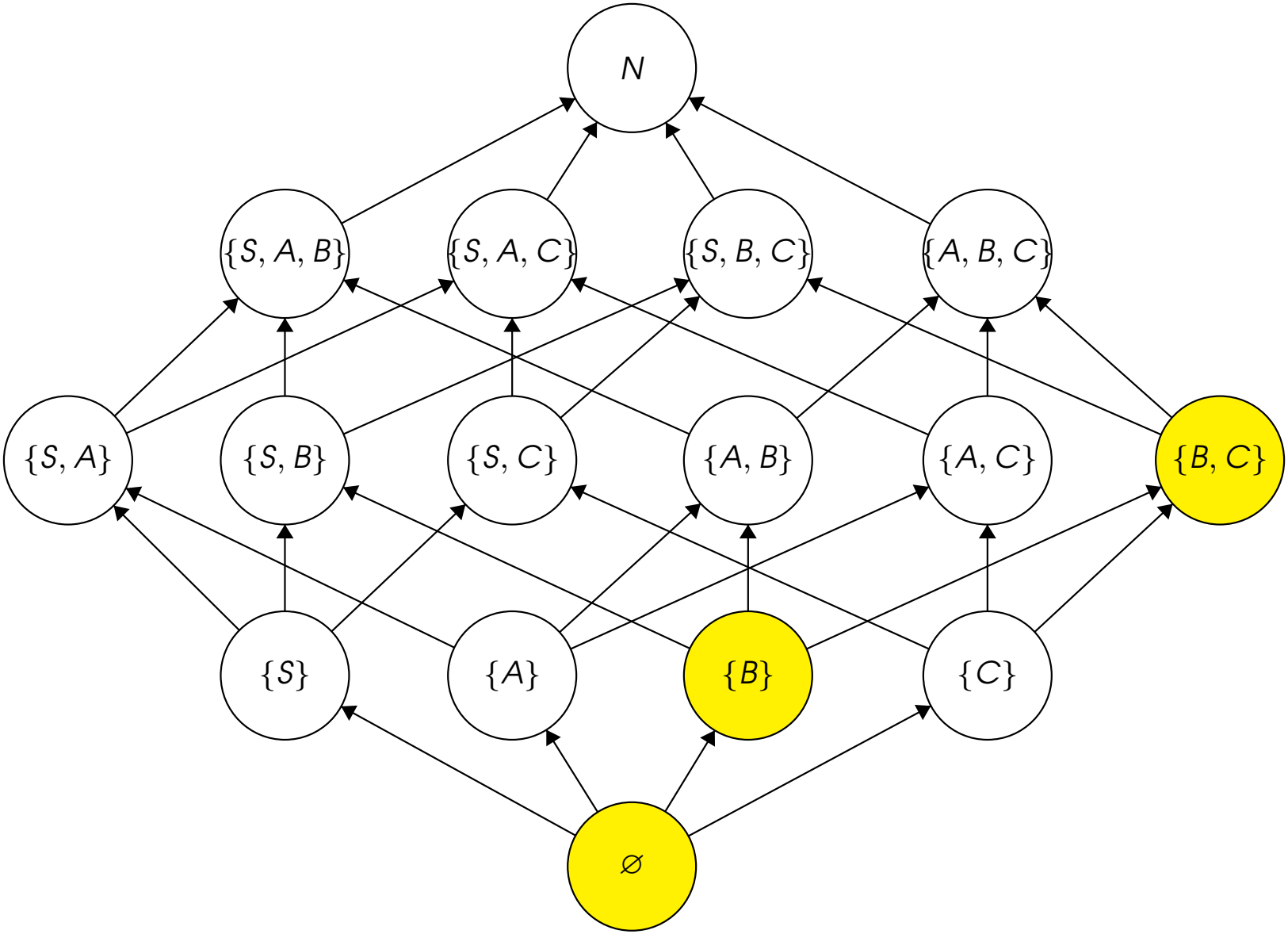


Příklad 4



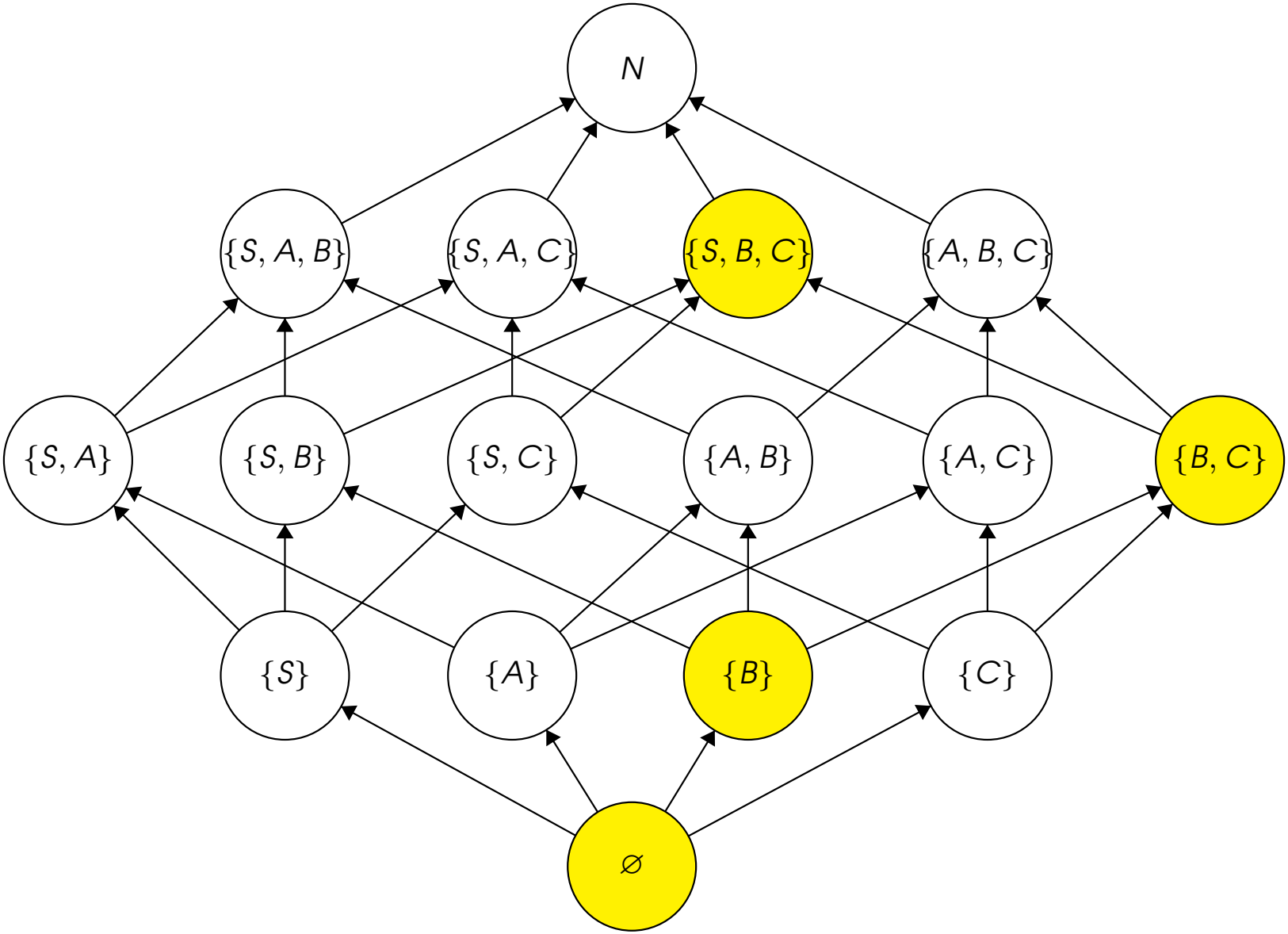
$$f(\varnothing) = \{B\}$$

Příklad 4



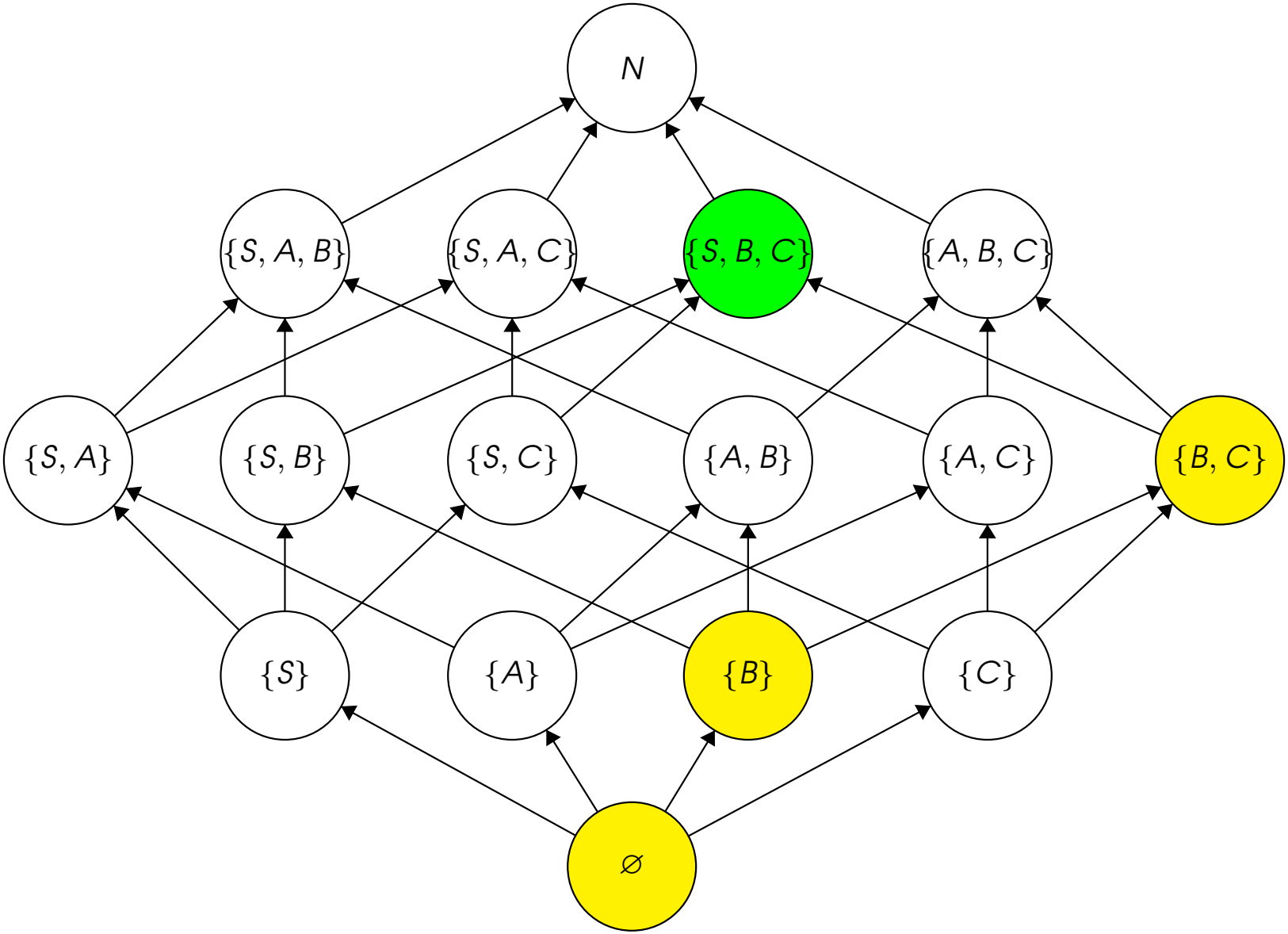
$$f^2(\varnothing) = f(\{B\}) = \{B, C\}$$

Příklad 4



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

Příklad 4



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

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 $A \to A$
- ϵ-pravidla $A \to AB, B \to \epsilon$

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

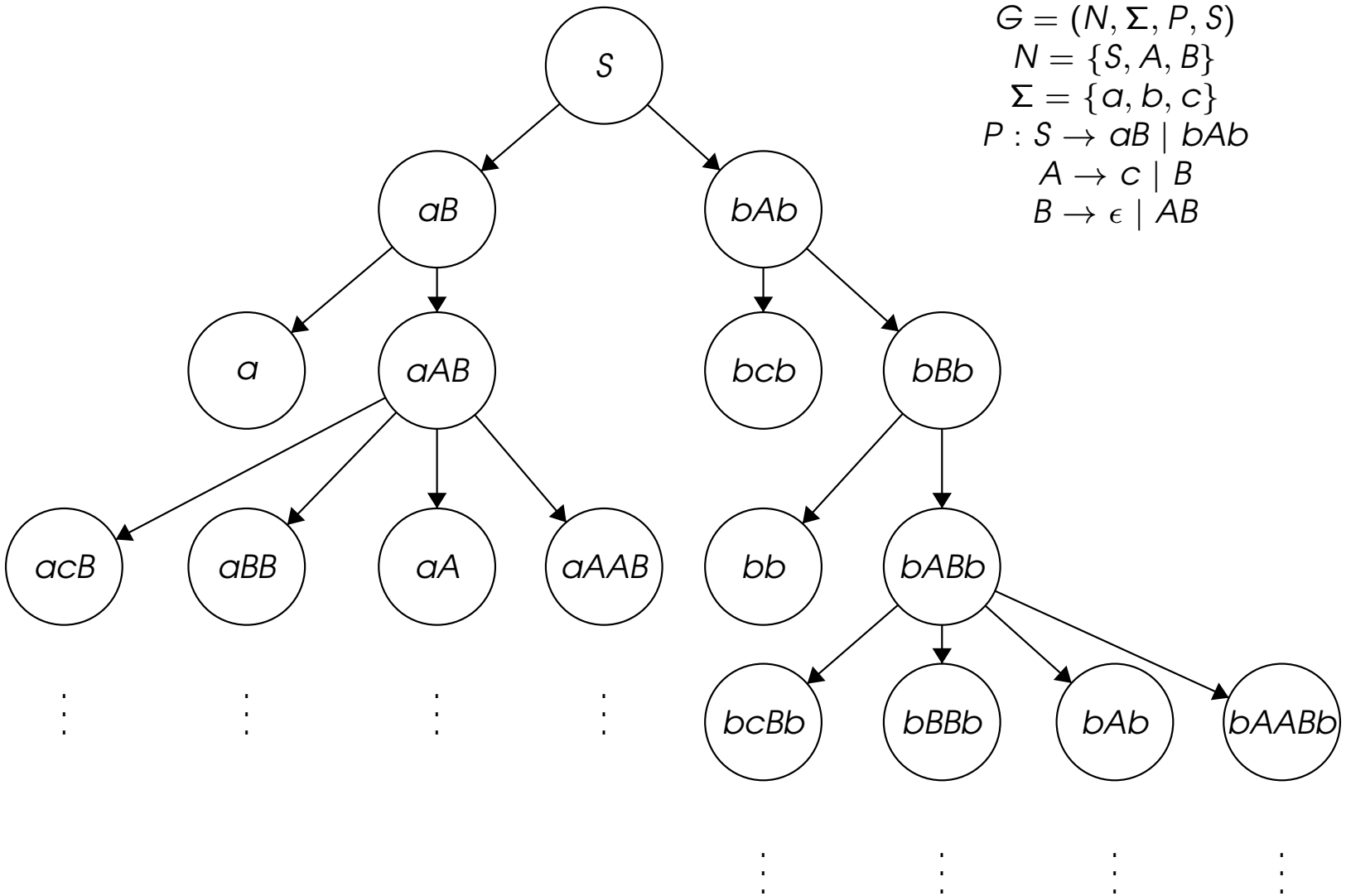
- jednoduchá pravidla $A \to A$
- ϵ-pravidla $A \to AB, B \to \epsilon$

Ukázka cyklu

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

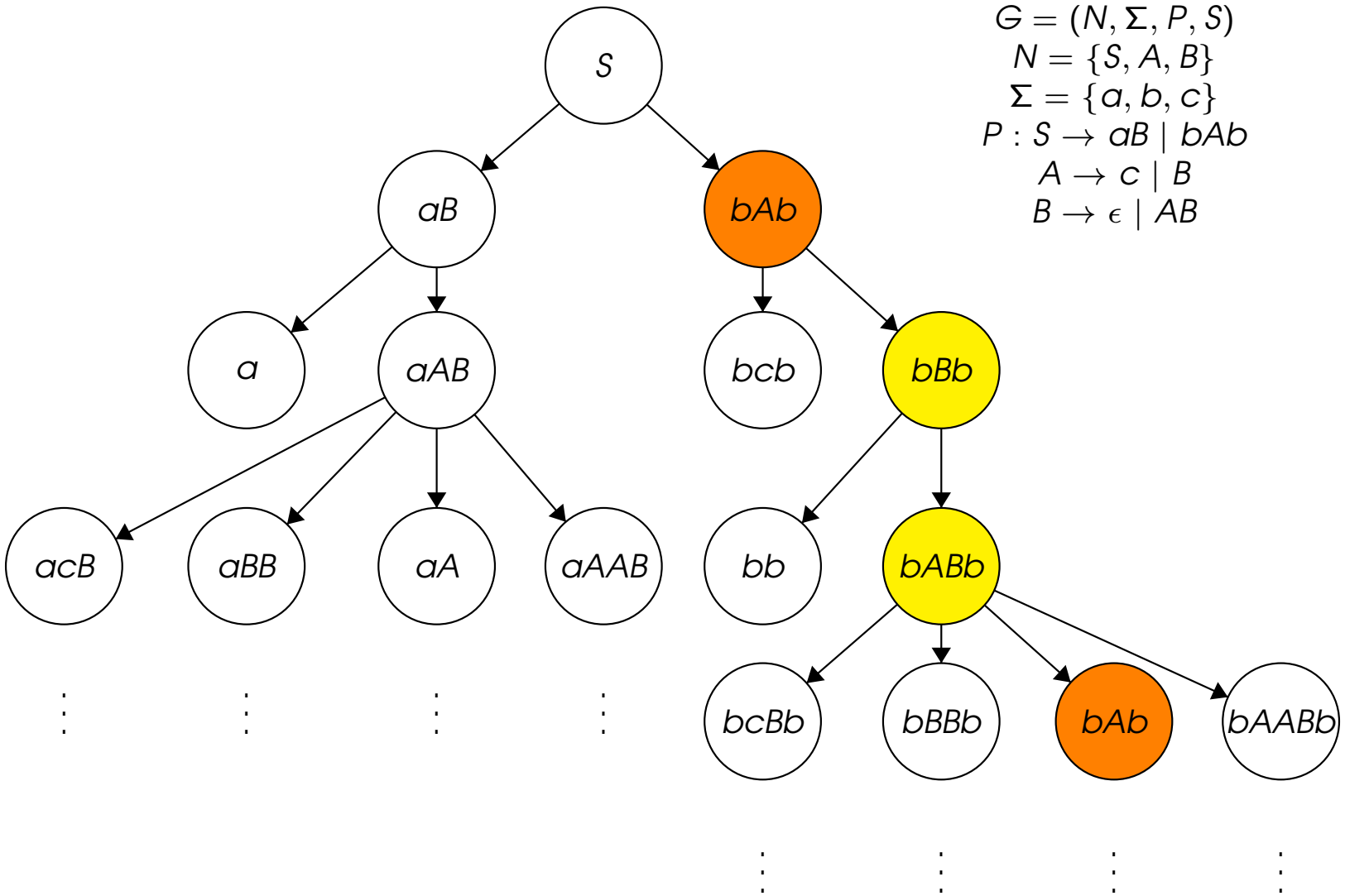
Příklad 5

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



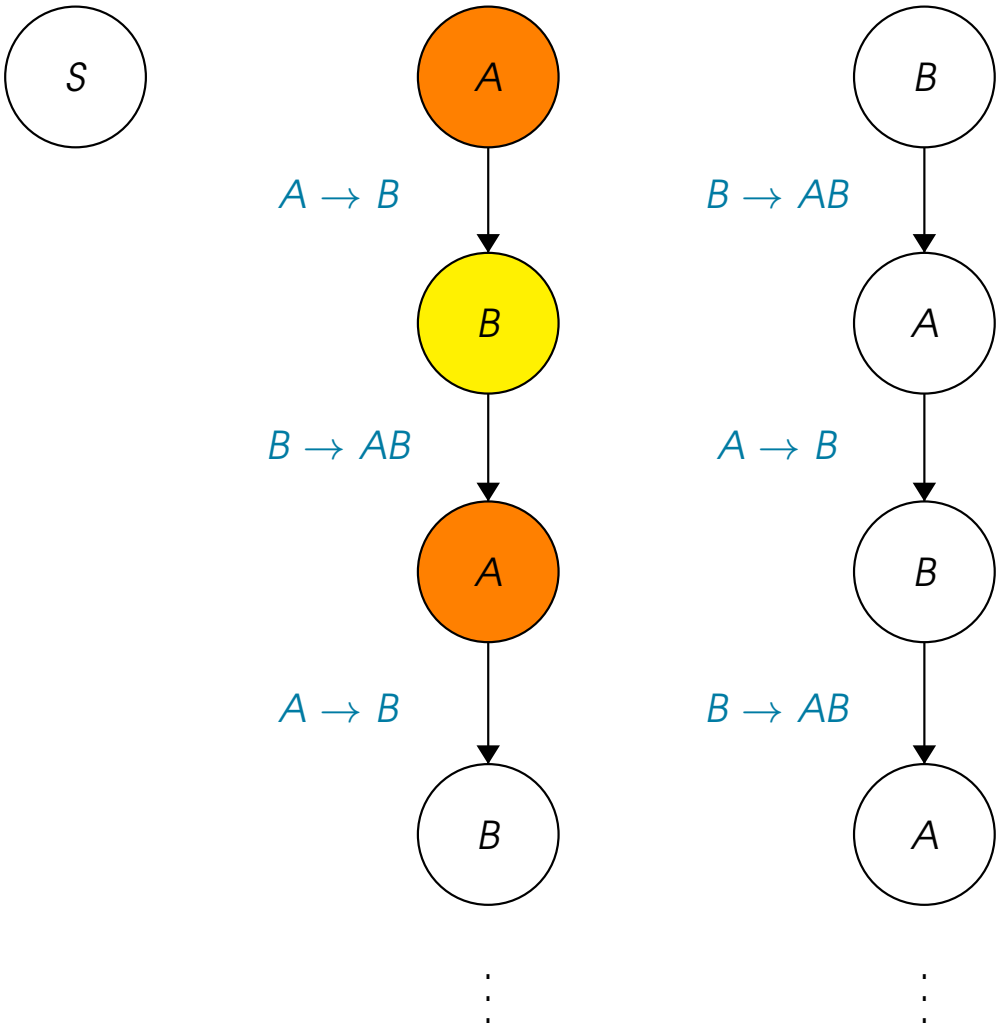
Obsahuje tato gramatika cyklus?

Příklad 5



Obsahuje tato gramatika cyklus?

Příklad 5



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

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

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

Výpočet

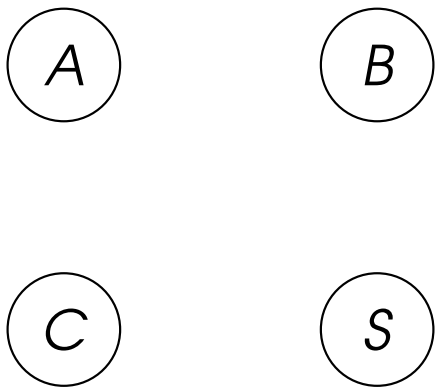
1. $N_\epsilon = \{S, A, B, C\}$

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

Výpočet

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

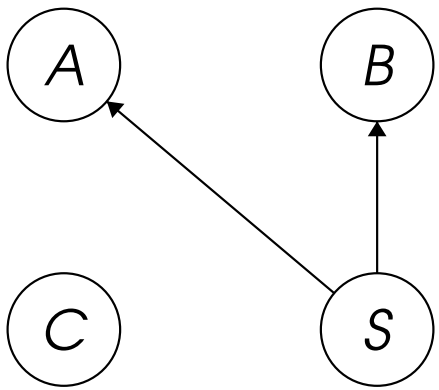


Demonstrace algoritmu

- 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. $\lambda = \{(S, A), (S, B)\}$

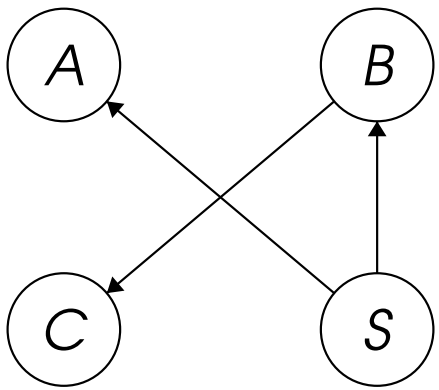


Demonstrace algoritmu

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

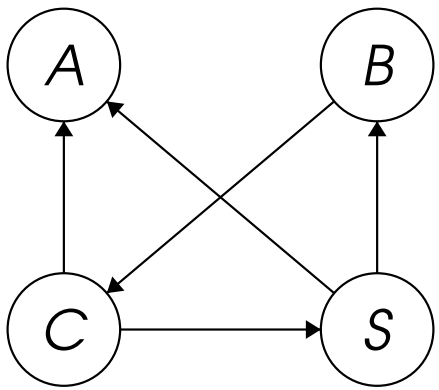


Demonstrace algoritmu

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

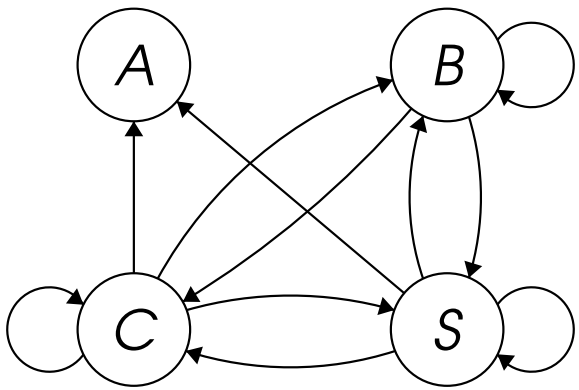


Demonstrace algoritmu

- 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. $\lambda^+ = \lambda \cup \{(B, S), (B, B), (S, C), (S, S), (C, B), (C, C)\}$

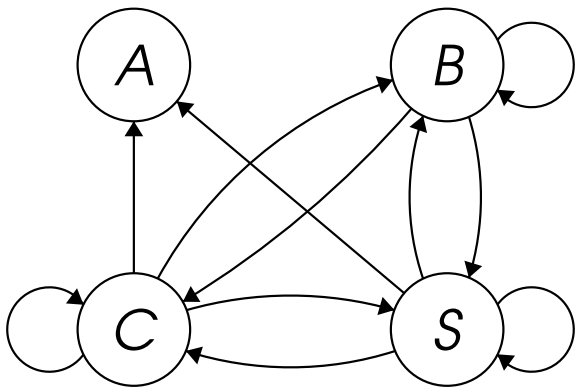


Demonstrace algoritmu

- Demonstrujte algoritmus na BKG $G = (N, \Sigma, P, S)$, kde
  1. $N = \{S, A, B, C\}$
  2. $\Sigma = \{a, b, c\}$
  3. P :
     - $S \to AB \mid Aa$
     - $A \to \epsilon \mid aAb$
     - $B \to C \mid cAc$
     - $C \to AA \mid ASA$

Výpočet

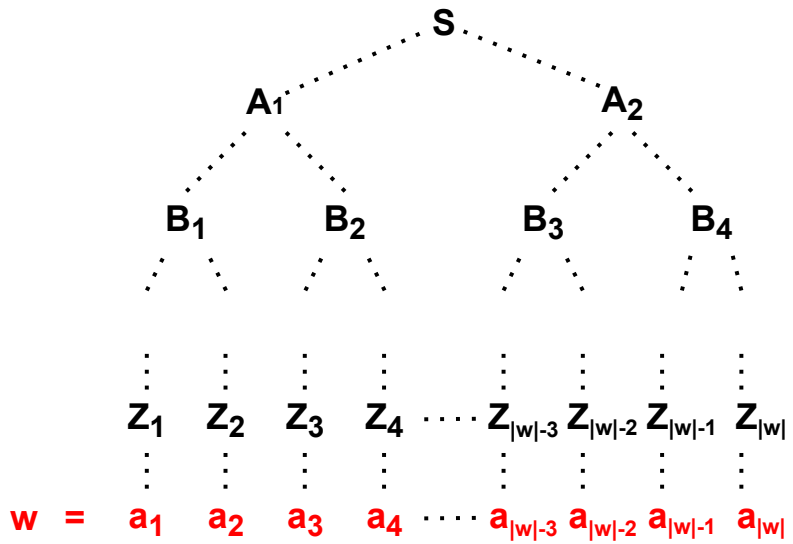
1. $N_\epsilon = \{S, A, B, C\}$
2. $\lambda = \{(S, A), (S, B), (B, C), (C, A), (C, S)\}$
3. $\lambda^+ = \lambda \cup \{(B, S), (B, B), (S, C), (S, S), (C, B), (C, C)\}$
4. λ^+ není ireflexivní, gramatika G má cyklus v neterminálech B, C, S



Pumping lemma pro bezkontextové jazyky

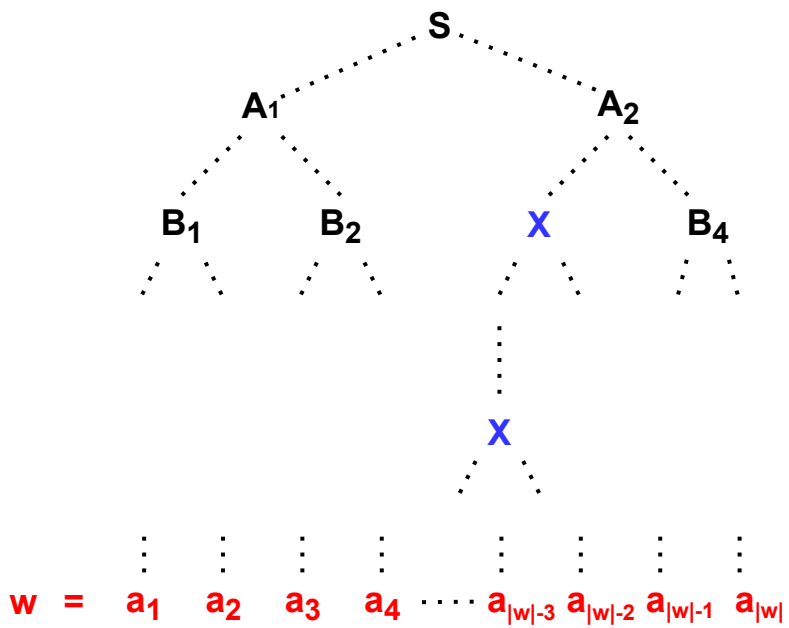
Chomského normální forma

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

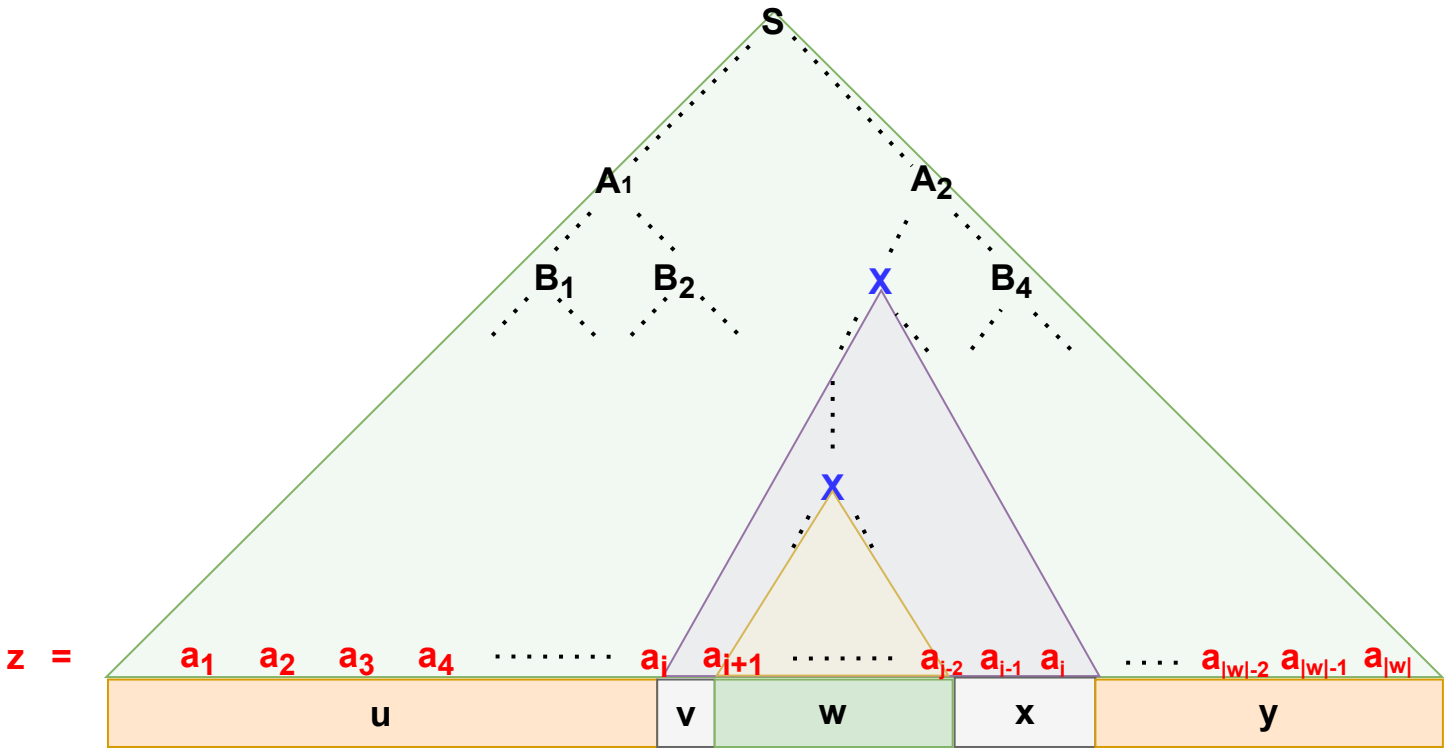


Opakování neterminálu na nejdelší cestě z kořenu k listu

- 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

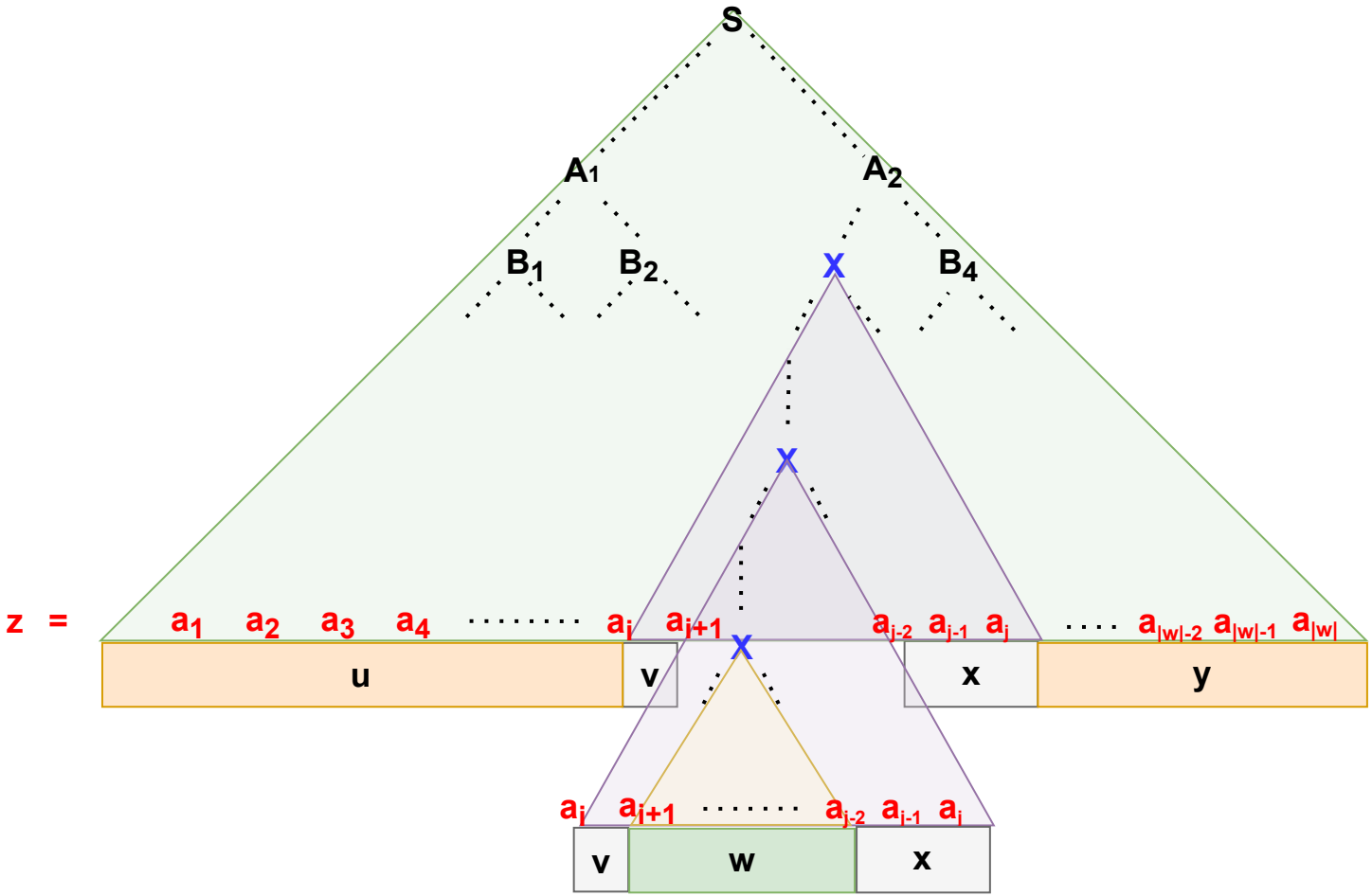


Pumping lemma pro bezkontextové jazyky

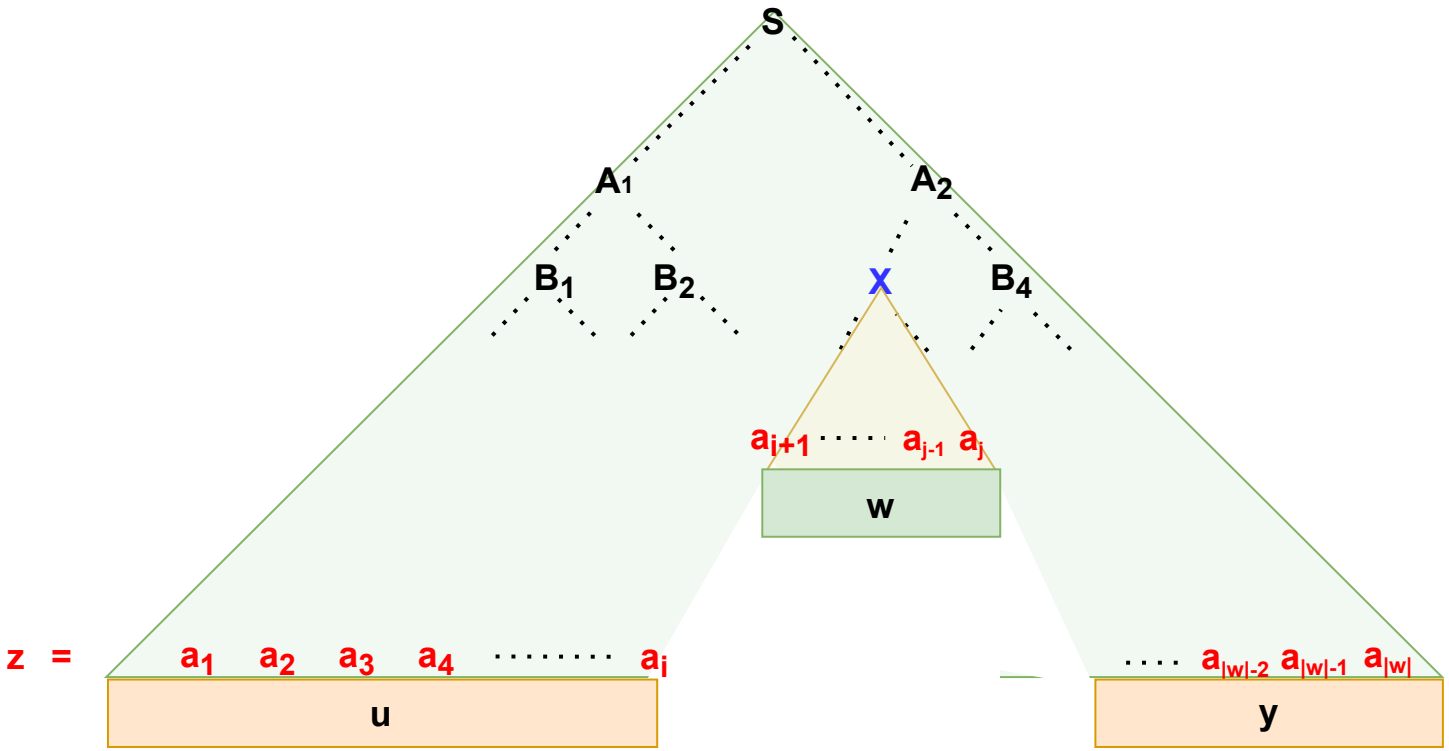


$$S \Rightarrow^+ uXy \Rightarrow^+ uvXxy \Rightarrow^+ uvwxy$$

Pumping lemma pro bezkontextové jazyky

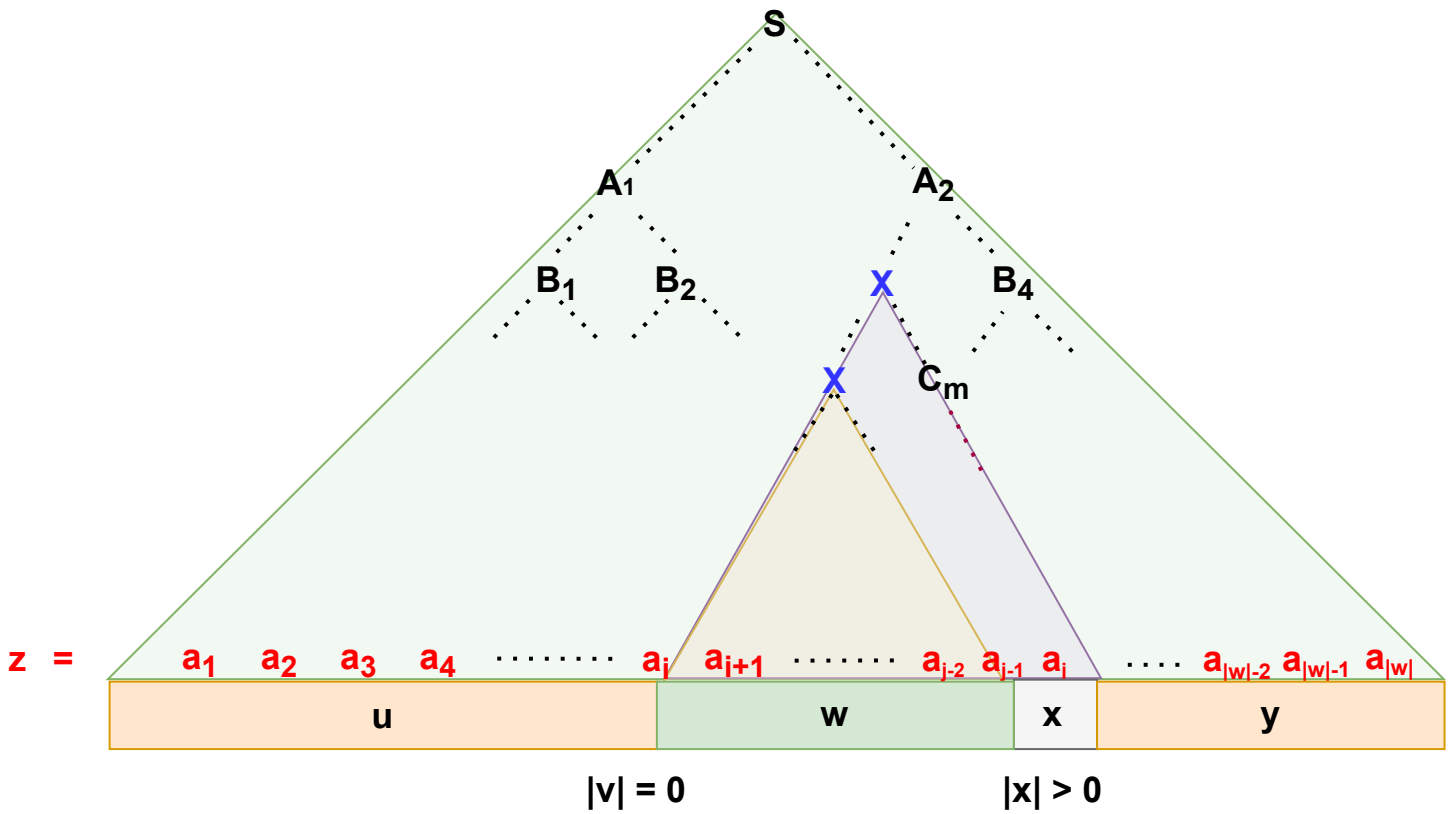


$$S \Rightarrow^{+} uXy \Rightarrow^{+} uvXxy \Rightarrow^{+} uv^2Xx^2y \Rightarrow^{+} uv^2wx^2y$$

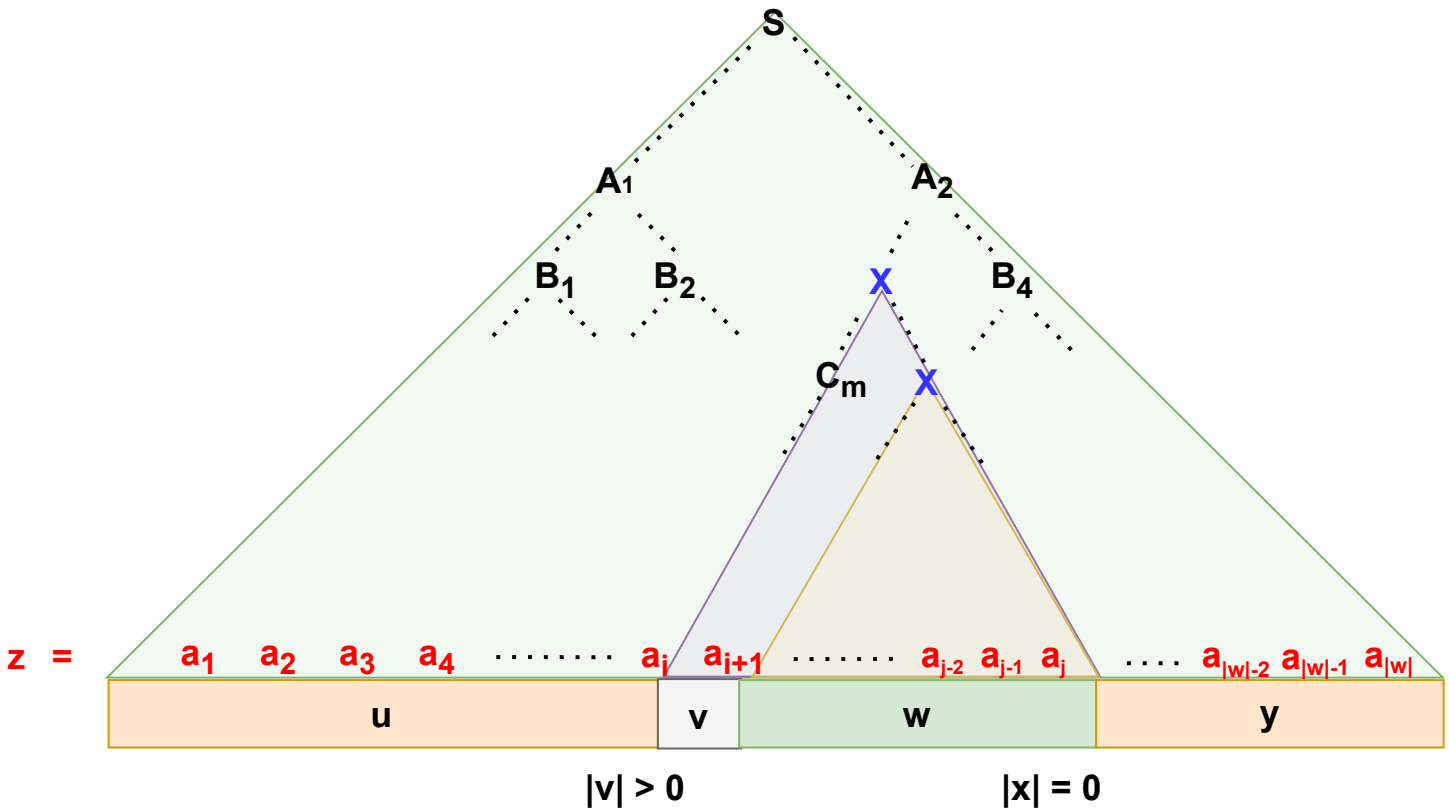


$$S \Rightarrow^+ uXy \Rightarrow^+ uwy$$

Pumping lemma pro bezkontextové jazyky



Alespoň jedna z částí v, x bude vždy neprázdná.



Alespoň jedna z částí v, x bude vždy neprázdná.

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

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

Obmněněná věta

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

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

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

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 $uv^i wx^i y \notin L$

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 $uv^i wx^i y \notin L$
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 $z = a^{k+2} b^{k+1} c^k$

Ruzných 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 $z = a^{k+2} b^{k+1} c^k$

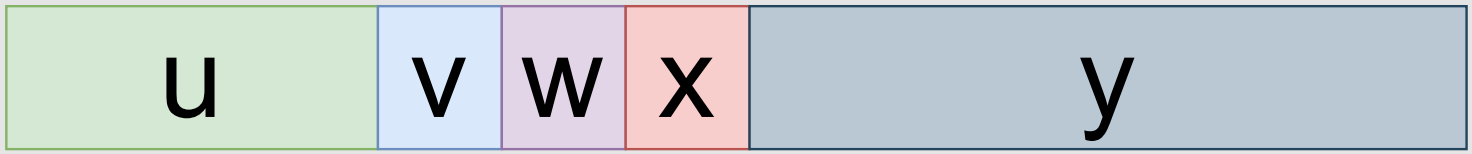
Ruzných typů rozdělení je příliš mnoho.

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



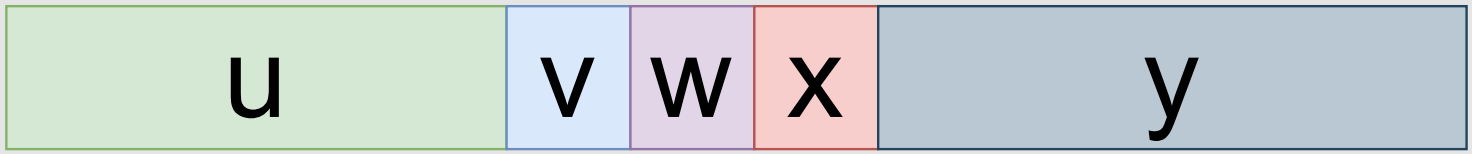
Ruzných typů rozdělení je příliš mnoho.

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



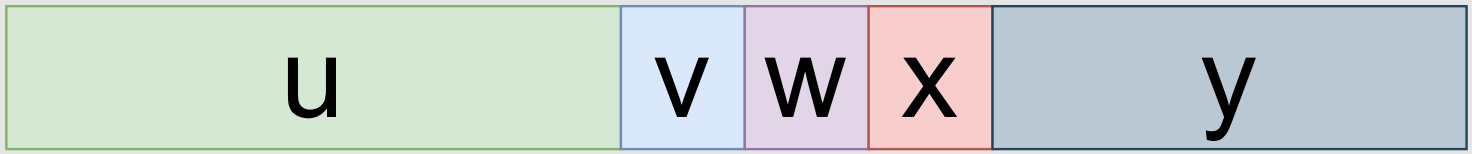
Ruzných typů rozdělení je příliš mnoho.

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



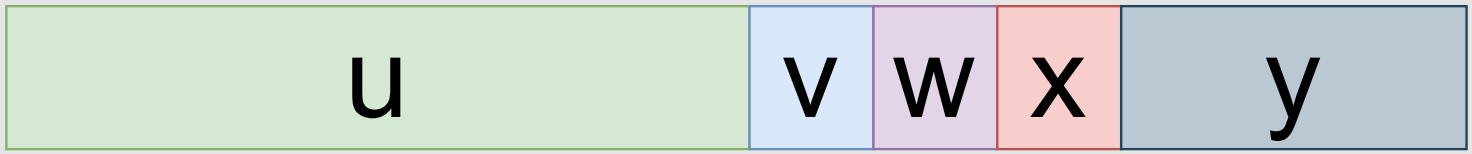
Ruzných typů rozdělení je příliš mnoho.

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



Ruzných 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 $z = a^{k+2} b^{k+1} c^k$

Ruzných 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 $z = a^{k+2} b^{k+1} c^k$

Ruzných 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 $z = a^{k+2} b^{k+1} c^k$

Ruzných 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 roztřídit 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 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 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

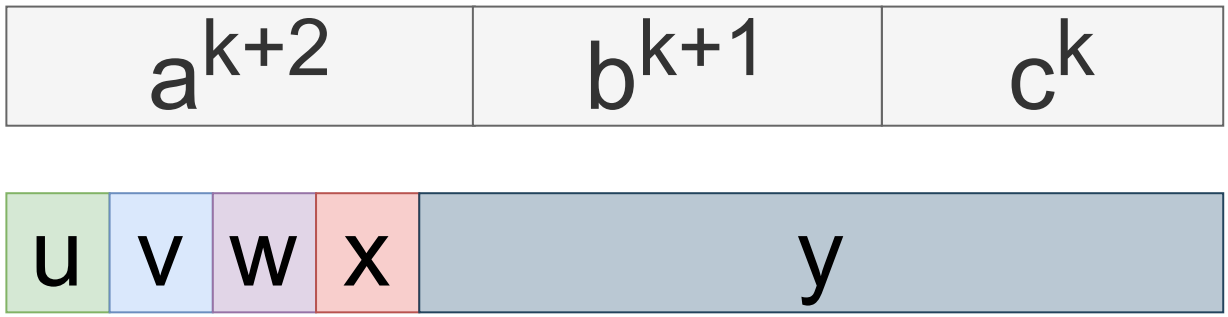
6. vx obsahuje pouze symboly a
   - zvolíme $i = 0$
   - ve slově $uv^iwx^iy = uv^0wx^0y$ dojde k narušení podmínky $\#_a(w) > \#_b(w)$
   - pracujeme totiž s řetězcem $z = a^{k+2}b^{k+1}c^k$ a $|vx| > 0$

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

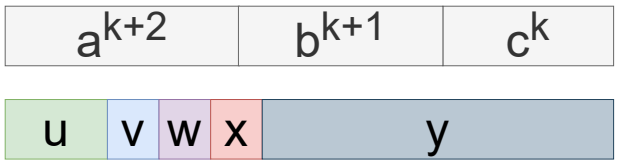
7. vx obsahuje alespoň jeden symbol b a neobsahuje c
   - zvolíme $i = 0$
   - ve slově $uv^i wx^i y = uv^0 wx^0 y$ dojde k narušení podmínky $\#_b(w) > \#_c(w)$
   - pracujeme totiž s řetězcem $z = a^{k+2} b^{k+1} c^k$ a $|vx| > 0$

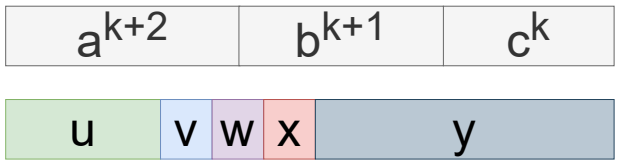
Pumping lemma pro bezkontextové jazyky

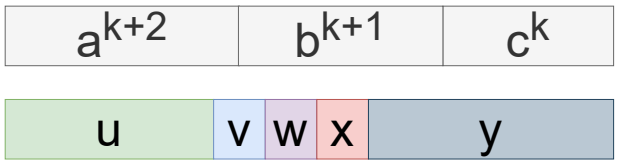
Dokažte, že $L = \{w \in \{a, b, c\}^* \mid \#_a(w) > \#_b(w) > \#_c(w)\} \notin \mathcal{L}_2$

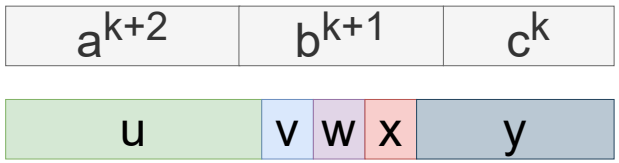
Řešení

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

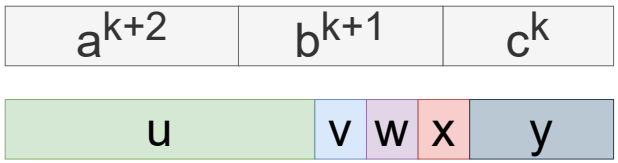
8. vx obsahuje alespoň jeden symbol c
   - zvolíme $i = 3$
   - ve slově $uv^i wx^i y = uv^3 wx^3 y$ dojde k narušení podmínky $\#_a(w) > \#_c(w)$
   - pracujeme totiž s řetězcem $z = a^{k+2} b^{k+1} c^k$ a $|vx| > 0$

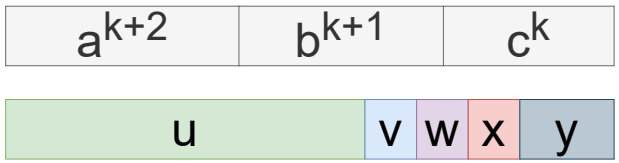
Pumping lemma pro bezkontextové jazyky

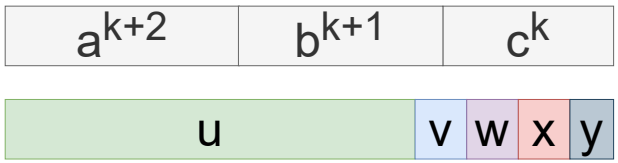
Dokažte, že $L = \{w \in \{a, b, c\}^* \mid \#_a(w) > \#_b(w) > \#_c(w)\} \notin \mathcal{L}_2$

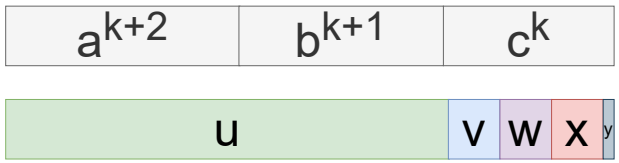
Řešení

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 $uv^i wx^i y \notin L$

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 $uv^i wx^i y \notin L$
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 $uv^i wx^i y \notin L$
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

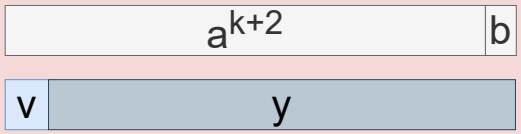
1. $z = a^{k+2}b$

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 $uv^iwx^iy = a^ia^{k+1}b \in L$

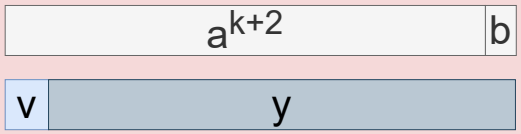


Pumping lemma pro bezkontextové jazyky

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 $uv^iwx^iy = a^ia^{k+1}b \in L$



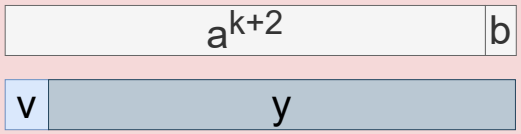
2. $z = aa(abc)^kb$

Pumping lemma pro bezkontextové jazyky

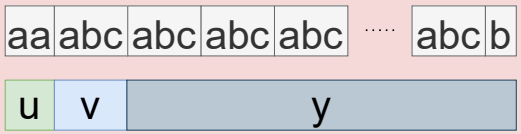
Dokažte, že $L = \{w \in \{a, b, c\}^* \mid \#_a(w) > \#_b(w) > \#_c(w)\} \notin \mathcal{L}_2$

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 $uv^iwx^iy = a^ia^{k+1}b \in L$



2. $z = aa(abc)^kb$
   - 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

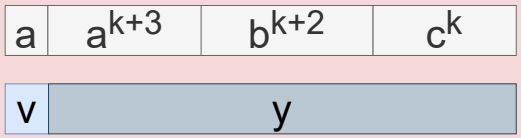
3. $z = a^{k+4}b^{k+2}c^k$

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 $uv^iwx^iy = a^ia^{k+3}b^{k+2}c^k \in L$

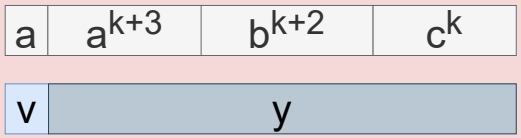


Pumping lemma pro bezkontextové jazyky

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

Příklady špatného výběru slova

3. $z = a^{k+4}b^{k+2}c^k$
   - 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 $uv^iwx^iy = a^ia^{k+3}b^{k+2}c^k \in L$



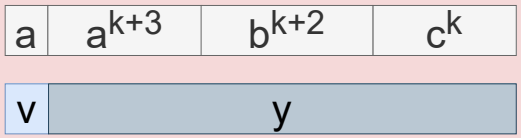
4. $z = a^kb^kc^k$

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 $uv^iwx^iy = a^ia^{k+3}b^{k+2}c^k \in L$



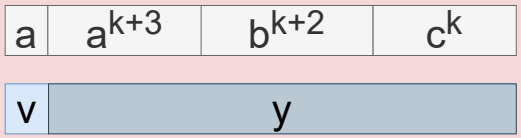
4. $z = a^kb^kc^k$
   - vůbec nepatří do jazyka L

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 $uv^iwx^iy = a^ia^{k+3}b^{k+2}c^k \in L$



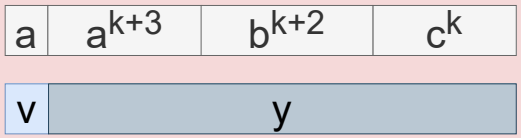
4. $z = a^kb^kc^k$
   - vůbec nepatří do jazyka L
5. $z = a^2b$

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 $uv^iwx^iy = a^ia^{k+3}b^{k+2}c^k \in L$



4. $z = a^kb^kc^k$
   - vůbec nepatří do jazyka L
5. $z = a^2b$
   - 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

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

Příklad 7

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:
  - $L_1 = \varnothing$
  - $L_2 = \{a^n b^n c^n \mid n \geq 0\}$

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:
  - $L_1 = \varnothing$
  - $L_2 = \{a^n b^n c^n \mid n \geq 0\}$
- 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

$$\forall L_1 \in \mathcal{L}_2 : \forall L_2 \in \mathcal{L}_3 : L_1 \subseteq L_2 \Rightarrow \overline{L_1 \cup L_2} \in L_3$$

Příklad 8

Abeceda

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

Rozhodněte a dokažte, zda platí následující tvrzení

$$\forall L_1 \in \mathcal{L}_2 : \forall L_2 \in \mathcal{L}_3 : L_1 \subseteq L_2 \Rightarrow \overline{L_1 \cup L_2} \in L_3$$

Řešení

- Tvrzení platí

Abeceda

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

Rozhodněte a dokažte, zda platí následující tvrzení

$$\forall L_1 \in \mathcal{L}_2 : \forall L_2 \in \mathcal{L}_3 : L_1 \subseteq L_2 \Rightarrow \overline{L_1 \cup L_2} \in L_3$$

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

$$\forall L_1 \in \mathcal{L}_2 : \forall L_2 \in \mathcal{L}_3 : L_1 \subseteq L_2 \Rightarrow \overline{L_1 \cup L_2} \in L_3$$

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

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

Příklad 9

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:
  - $L_1 = \overline{\{a^n b^n c^n \mid n \geq 0\}} \in \mathcal{L}_2$
  - $L_2 = \Sigma^* \in \mathcal{L}_3$

Příklad 9

Abeceda

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

Rozhodněte a dokažte, zda platí následující tvrzení

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

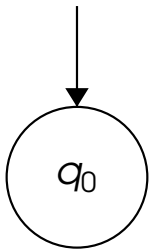
Řešení

- 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 $\overline{L_1 \cap L_2} = \overline{L_1} = \{a^n b^n c^n \mid \geq 0\} \notin \mathcal{L}_2$

Turingovy stroje

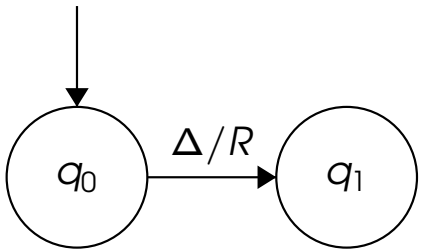
Konstrukce jednopáskového Turingova stroje

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



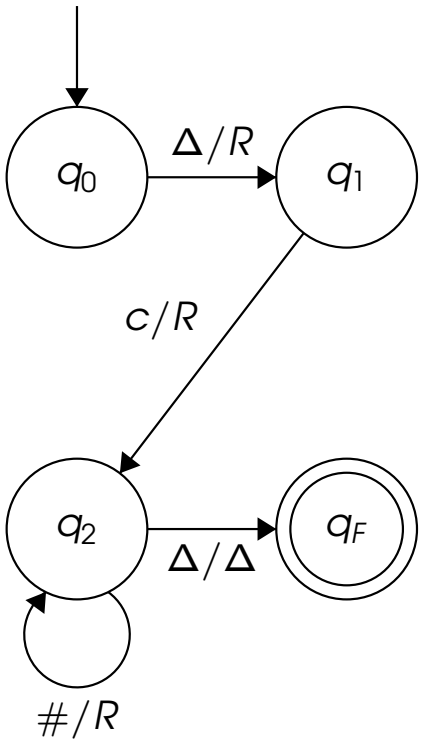
Konstrukce jednopáskového Turingova stroje

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



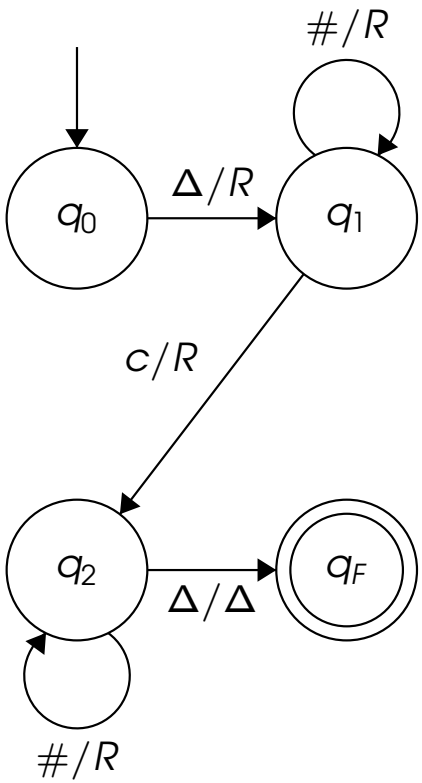
Konstrukce jednopáskového Turingova stroje

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



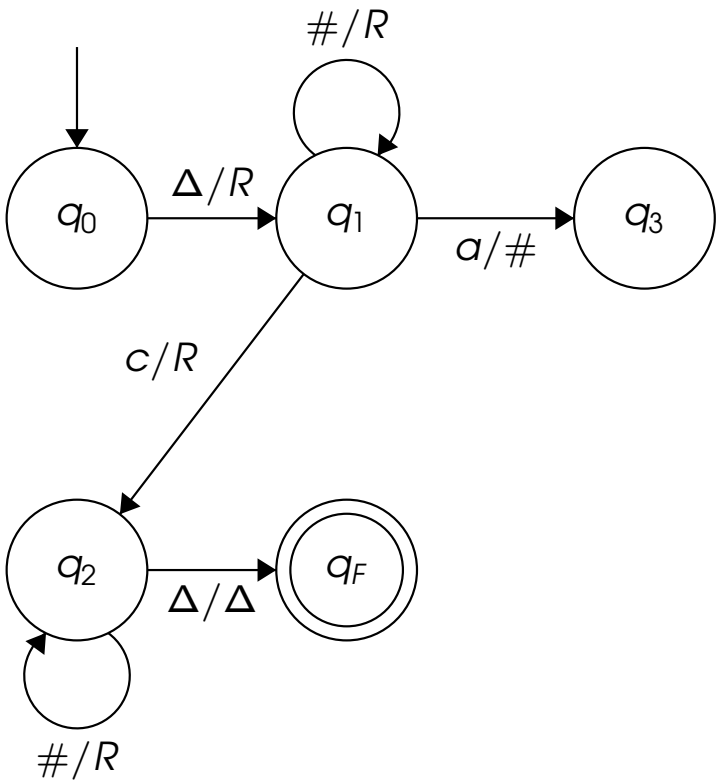
Konstrukce jednopáskového Turingova stroje

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



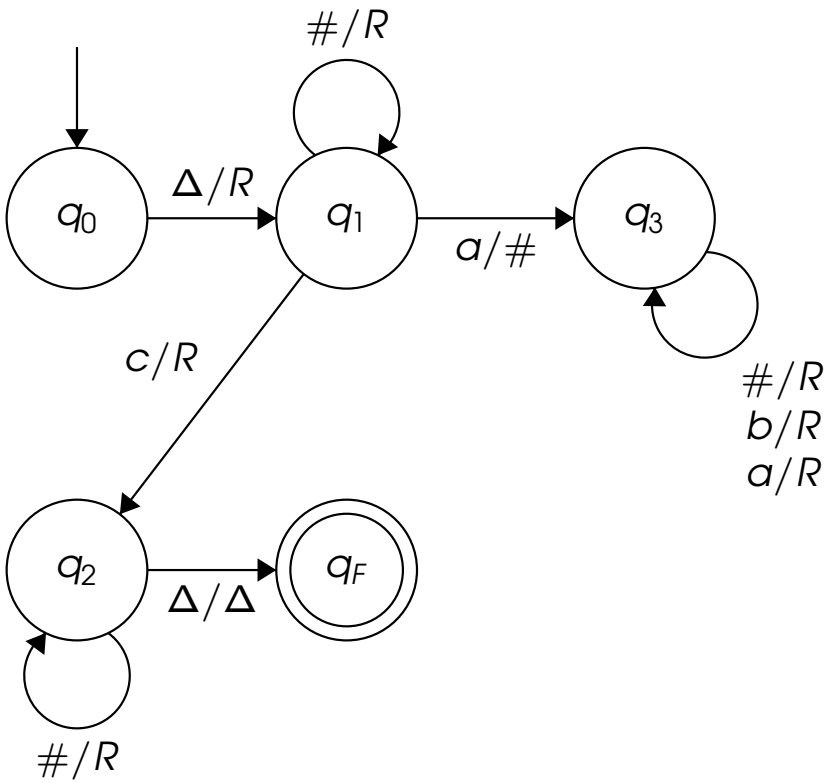
Konstrukce jednopáskového Turingova stroje

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



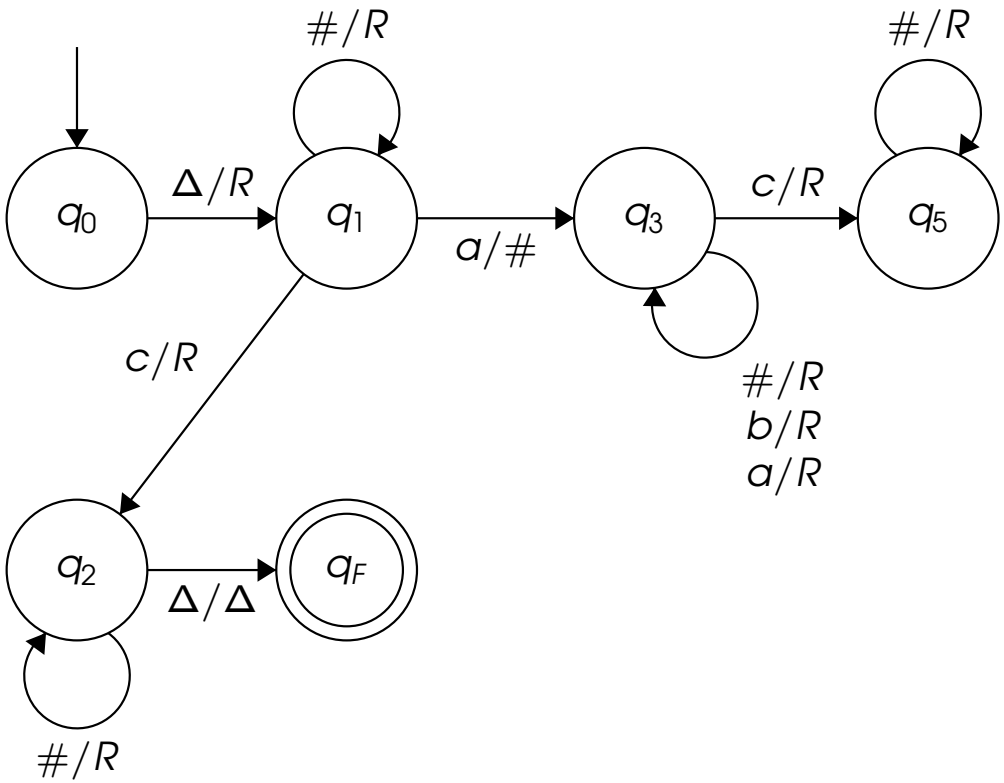
Konstrukce jednopáskového Turingova stroje

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



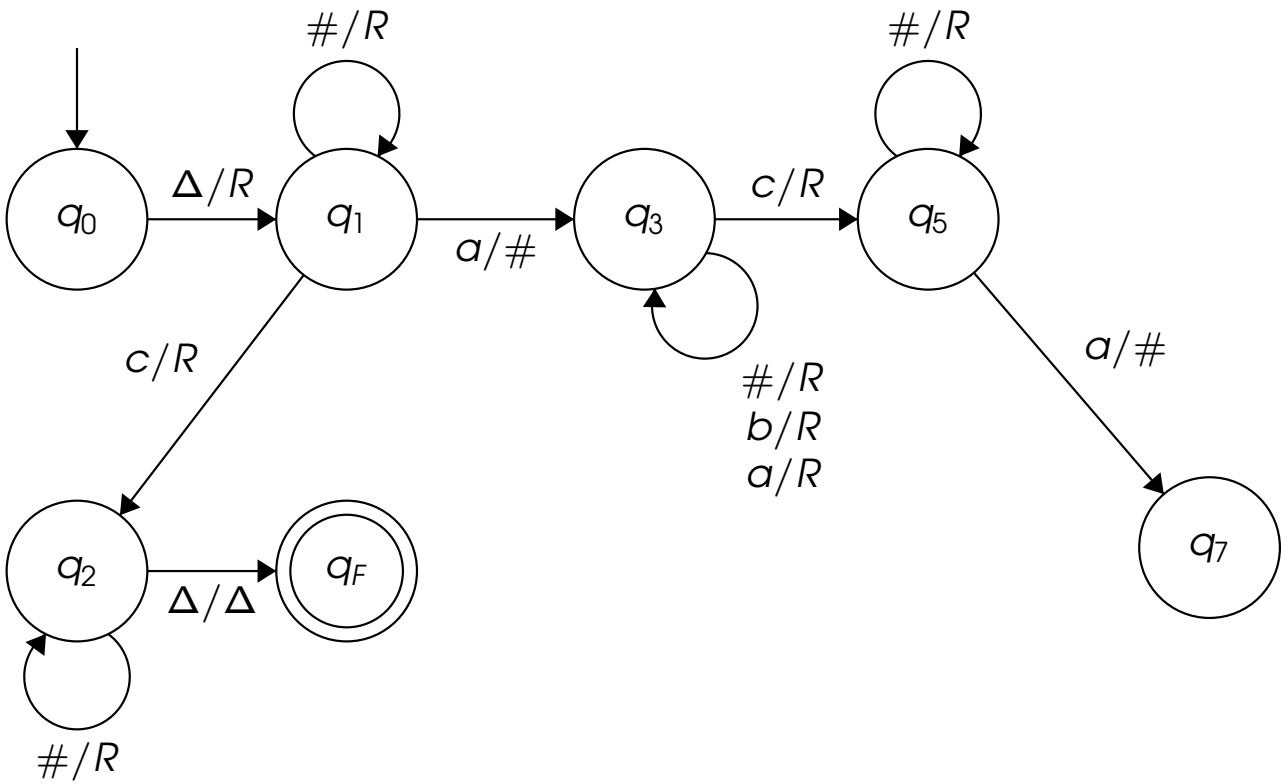
Konstrukce jednopáskového Turingova stroje

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



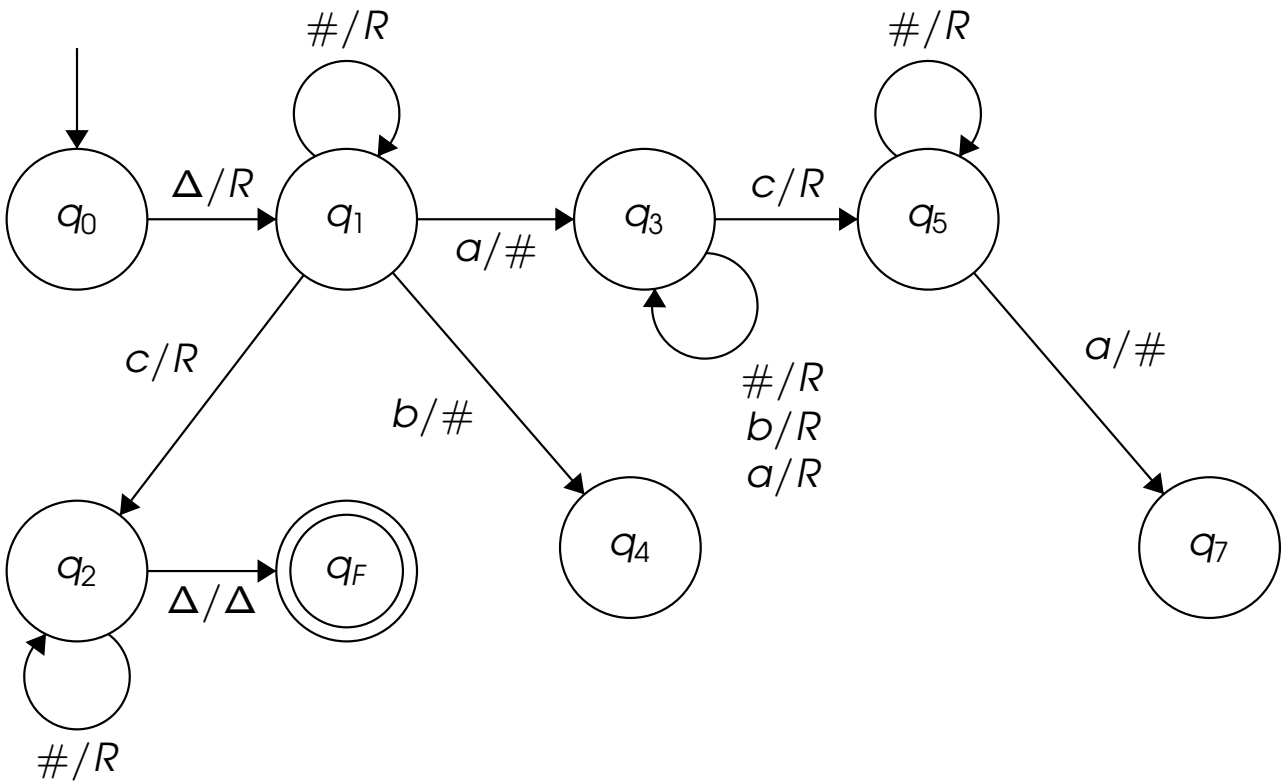
Konstrukce jednopáskového Turingova stroje

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



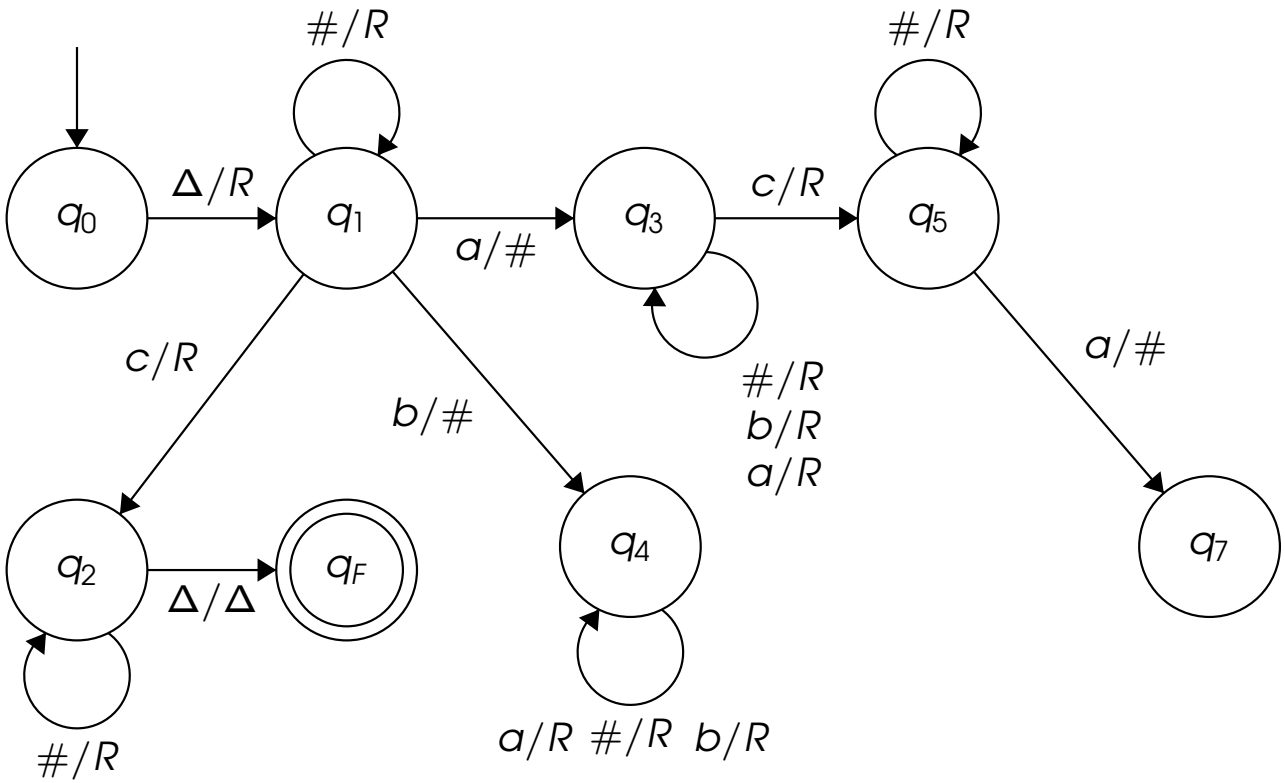
Konstrukce jednopáskového Turingova stroje

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



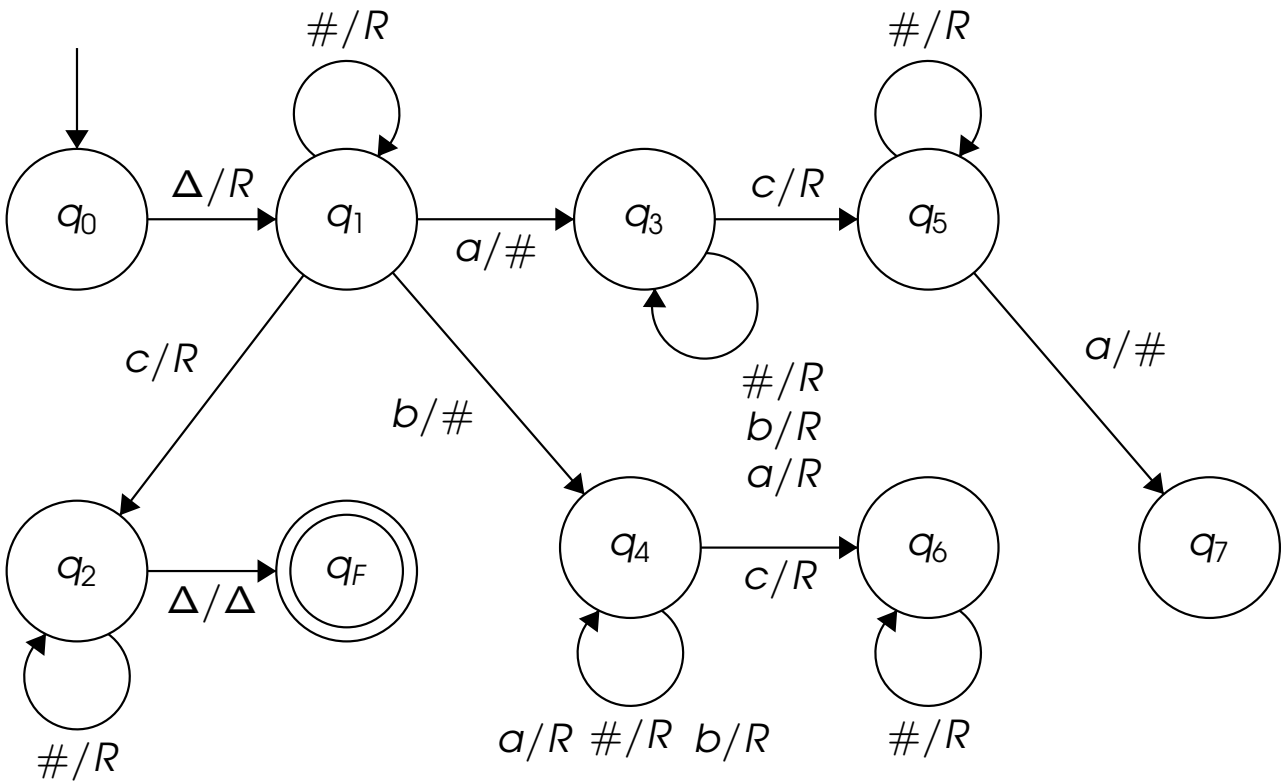
Konstrukce jednopáskového Turingova stroje

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



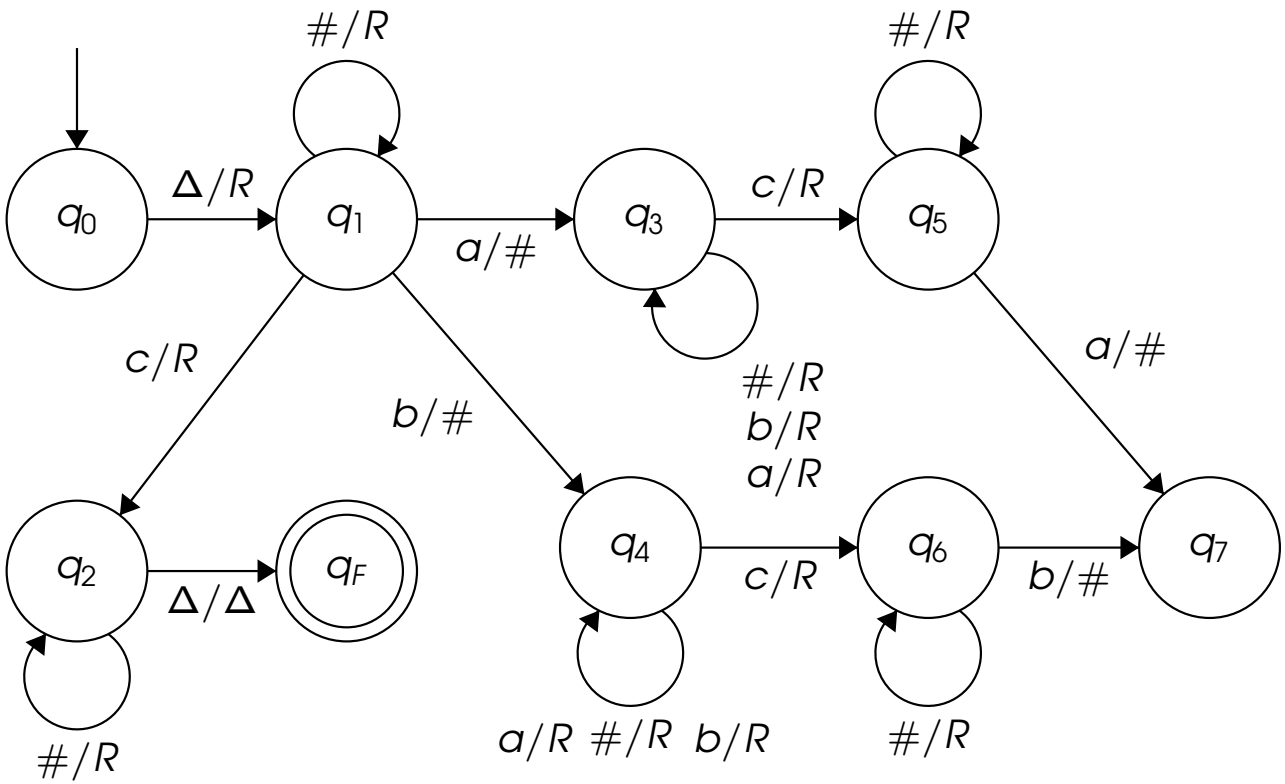
Konstrukce jednopáskového Turingova stroje

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



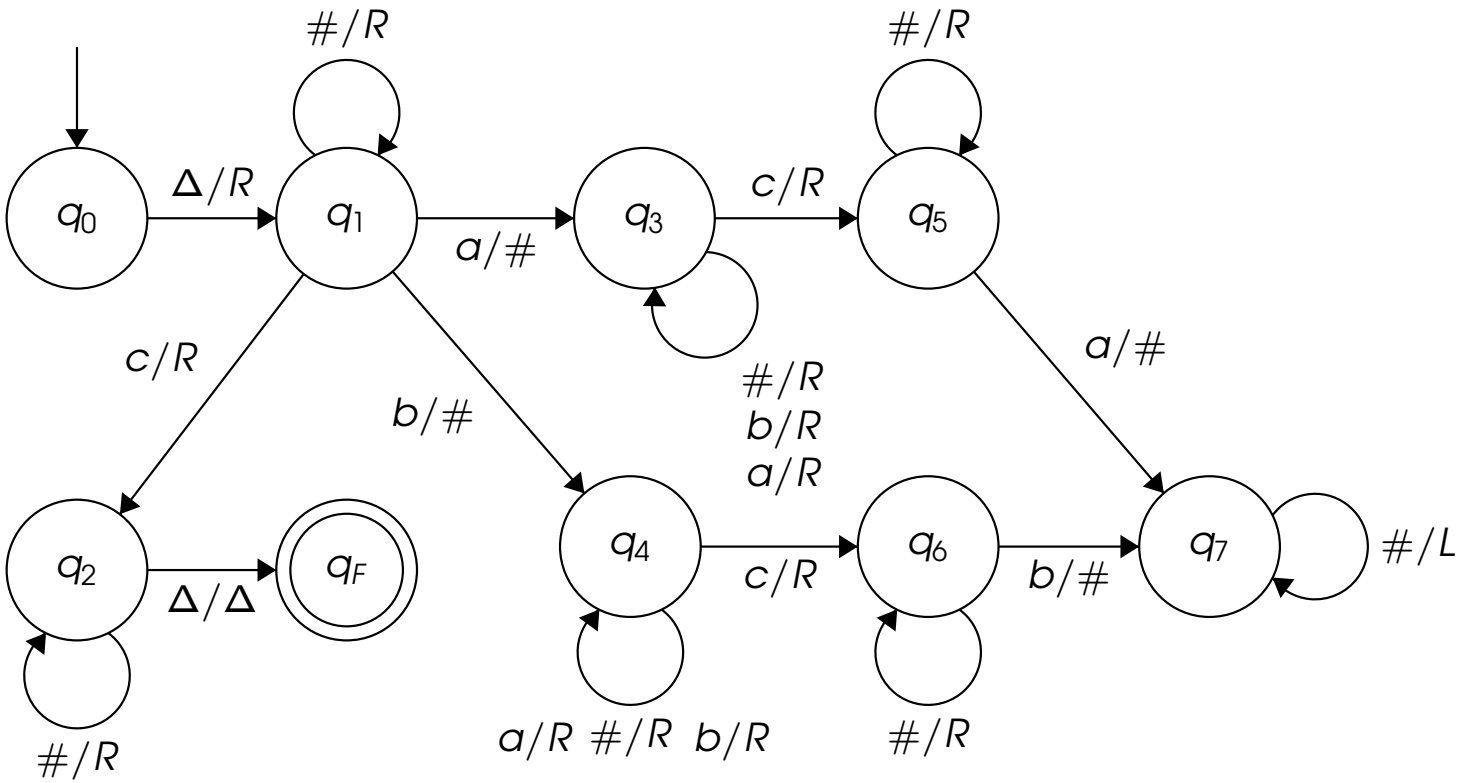
Konstrukce jednopáskového Turingova stroje

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



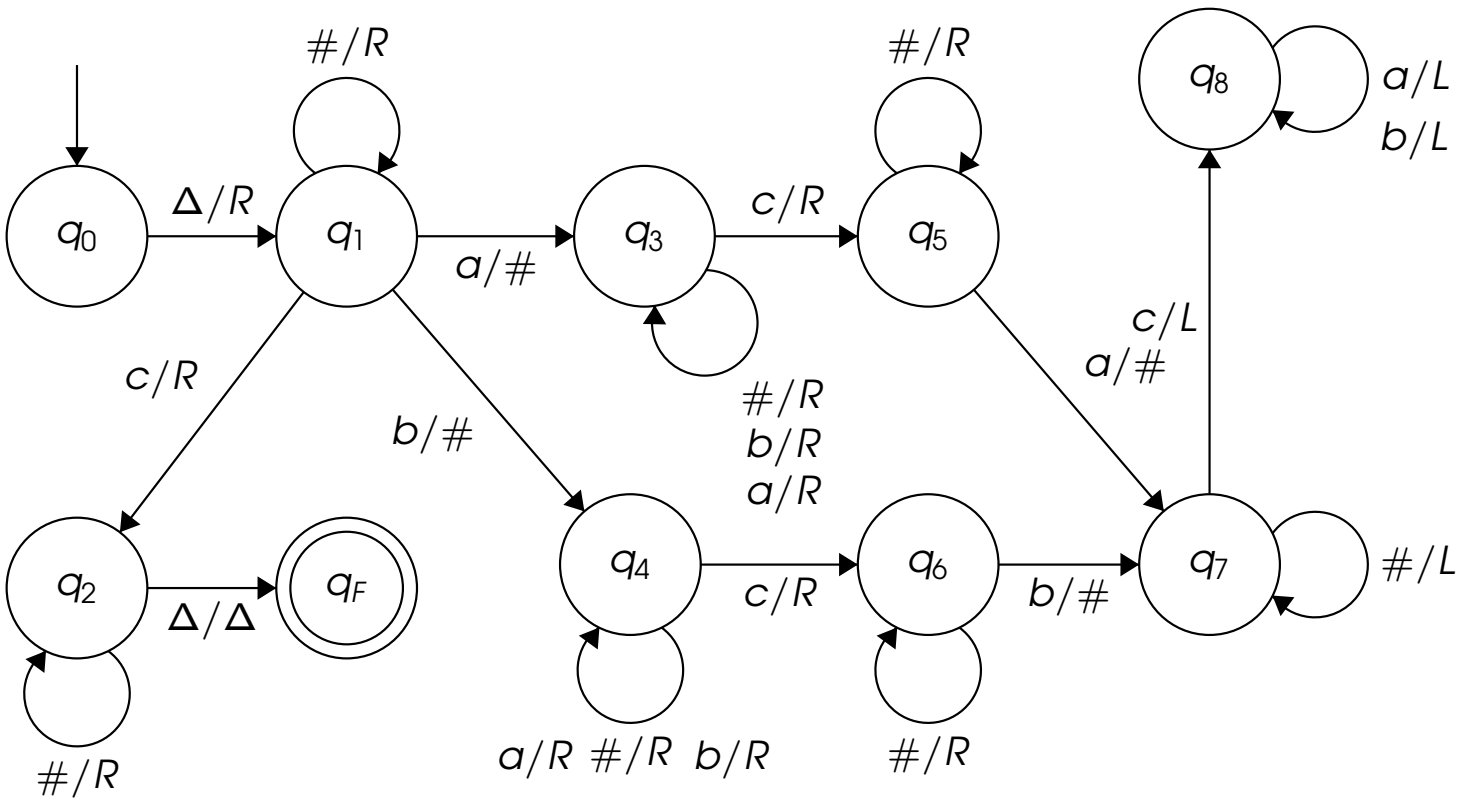
Konstrukce jednopáskového Turingova stroje

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



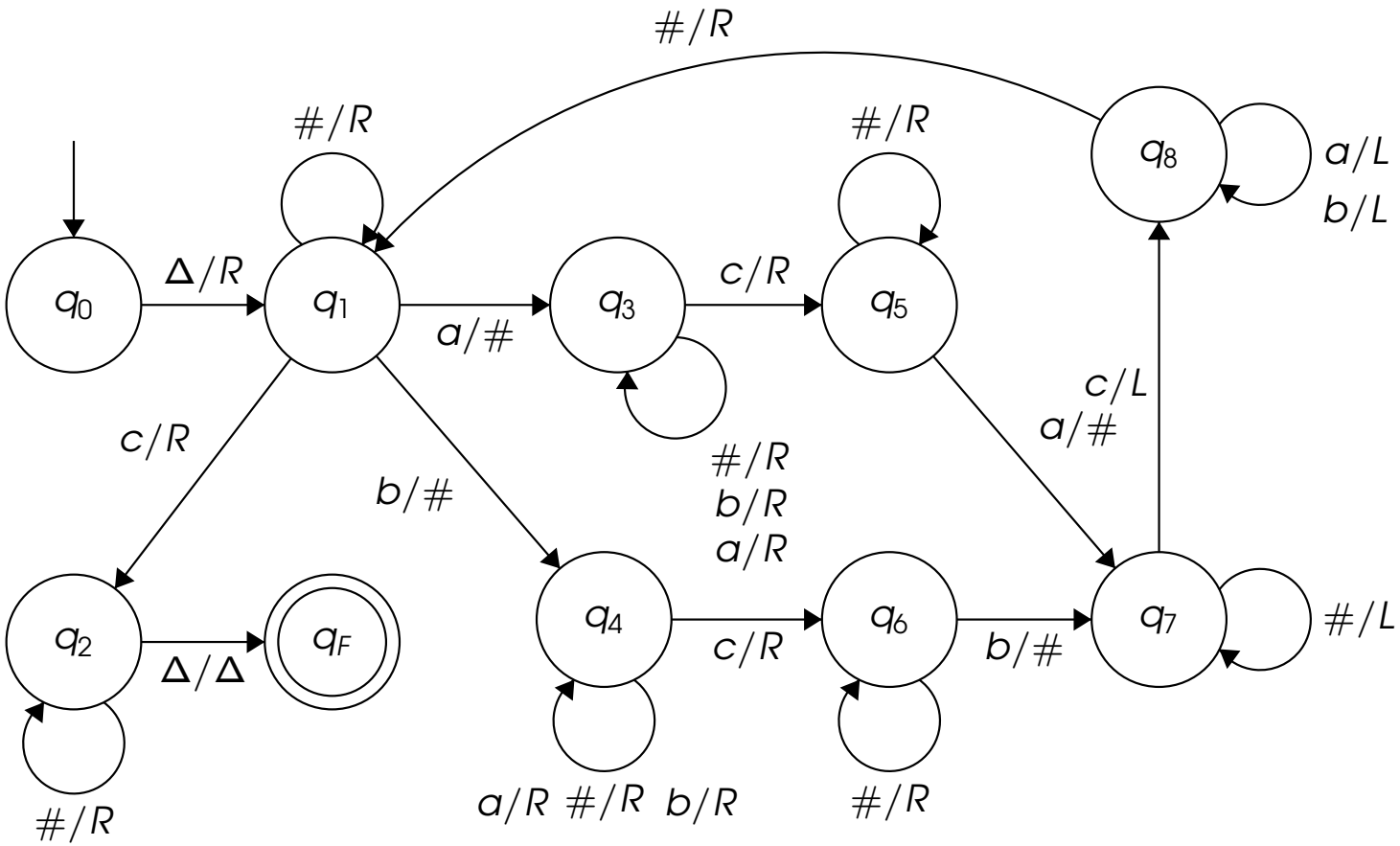
Konstrukce jednopáskového Turingova stroje

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



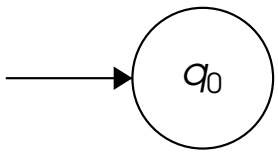
Konstrukce jednopáskového Turingova stroje

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



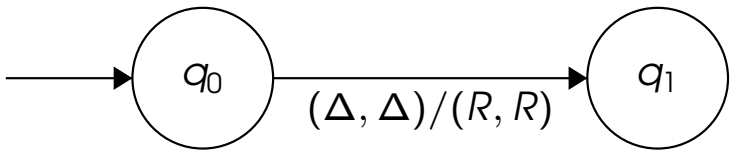
Konstrukce vícepáskového Turingova stroje

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



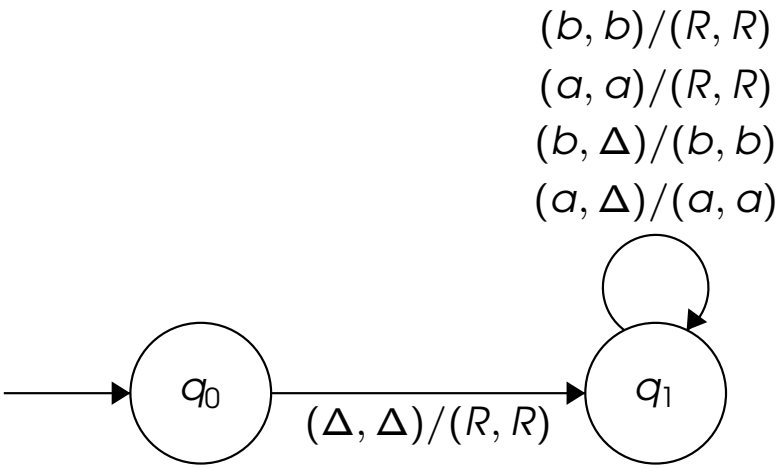
Konstrukce vícepáskového Turingova stroje

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



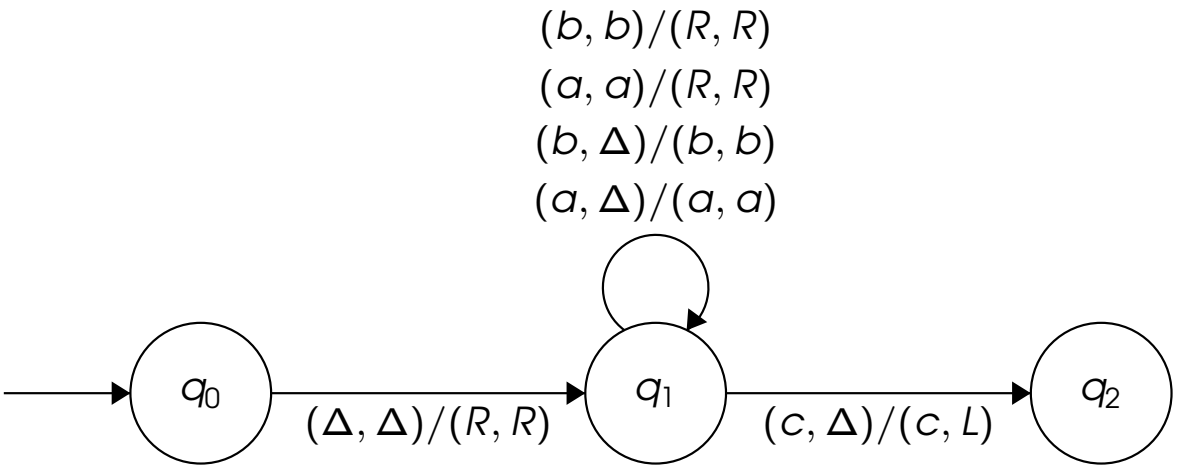
Konstrukce vícepáskového Turingova stroje

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



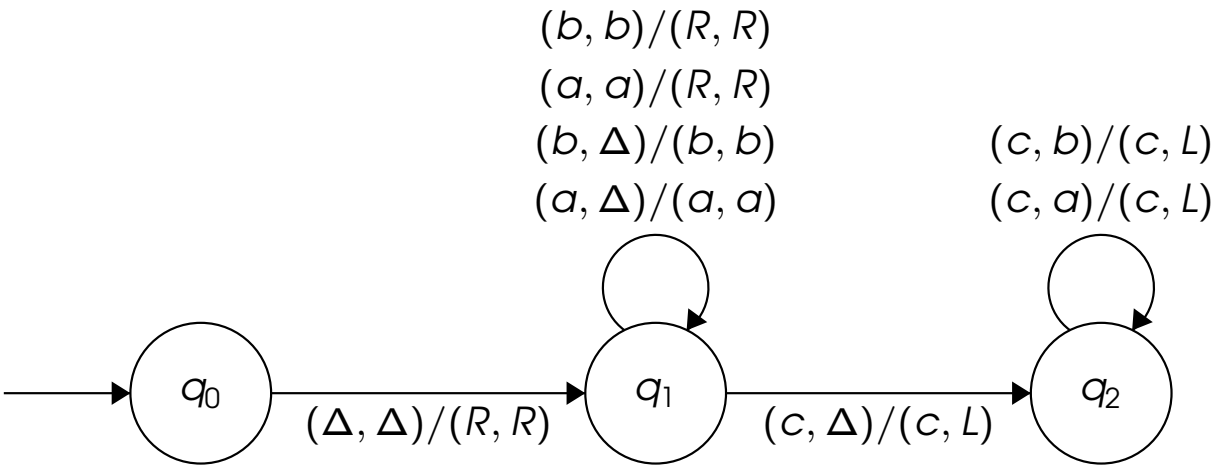
Konstrukce vícepáskového Turingova stroje

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



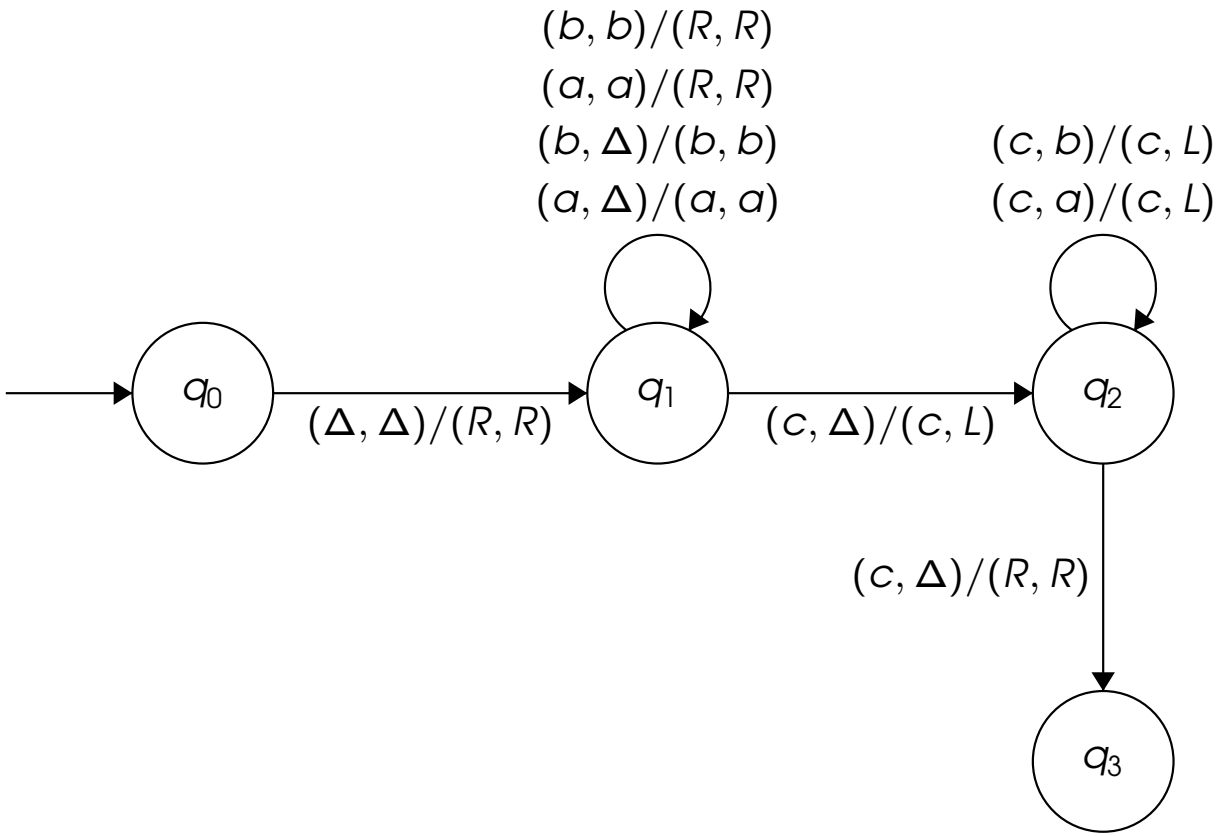
Konstrukce vícepáskového Turingova stroje

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



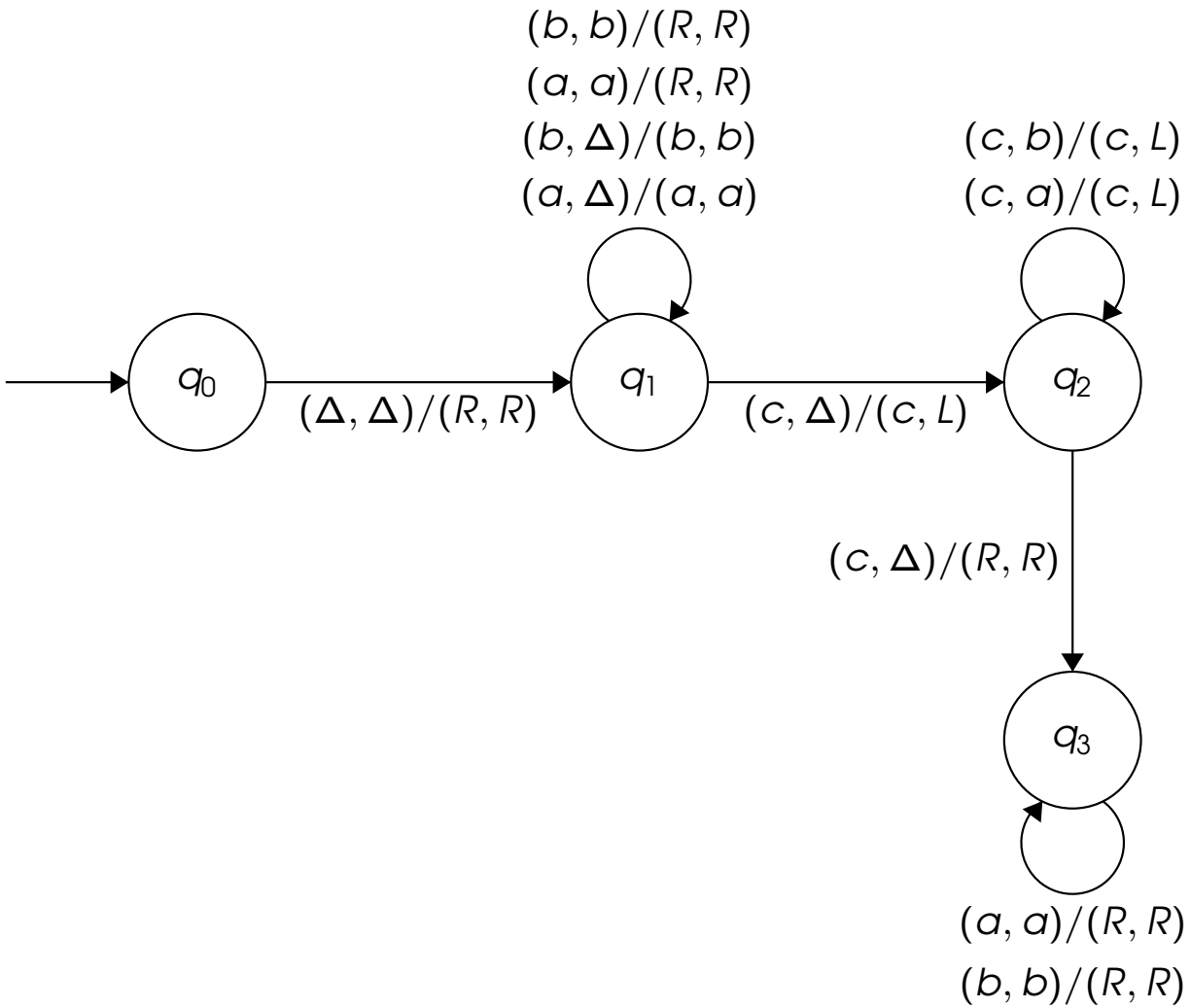
Konstrukce vícepáskového Turingova stroje

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



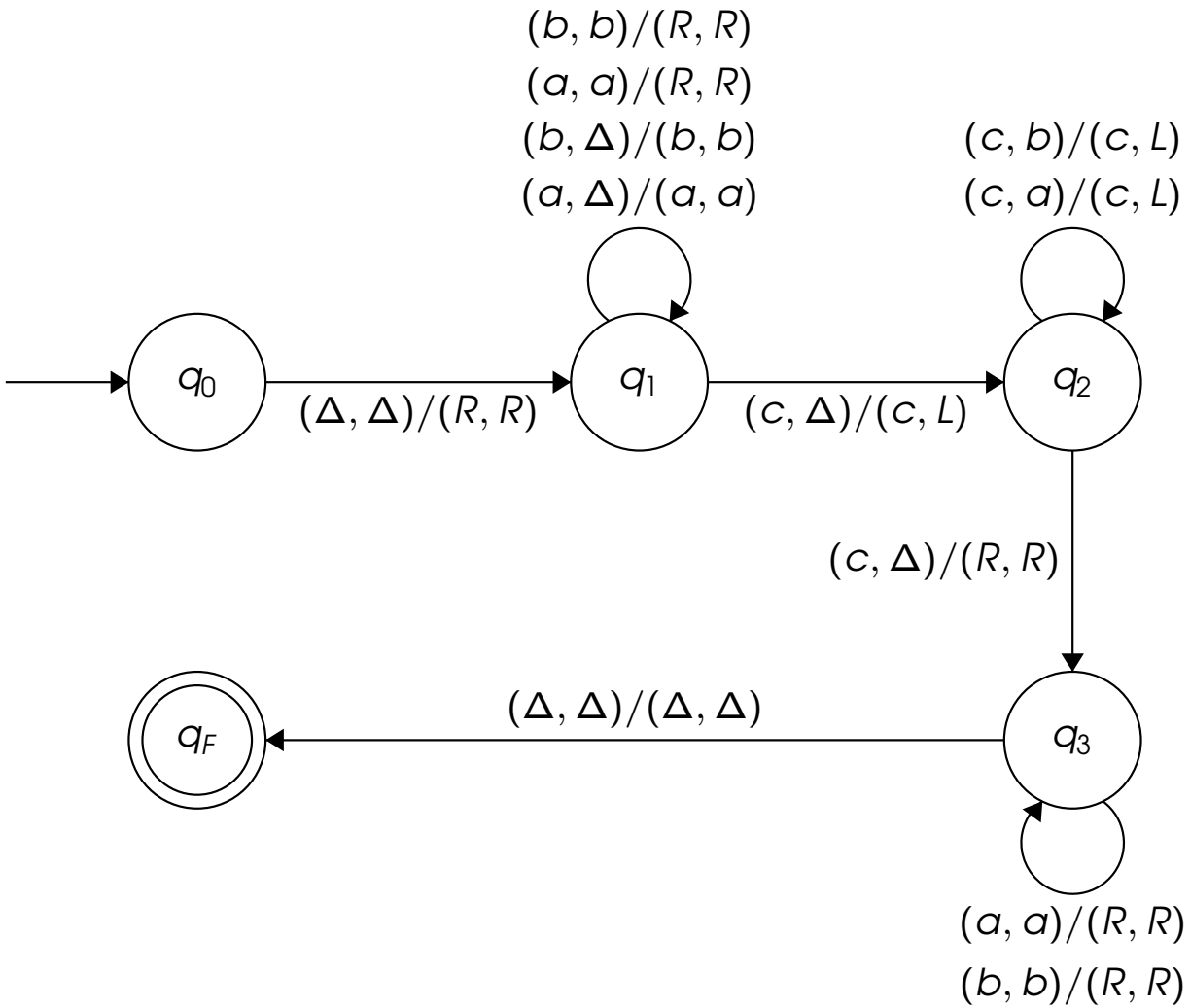
Konstrukce vícepáskového Turingova stroje

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



Konstrukce vícepáskového Turingova stroje

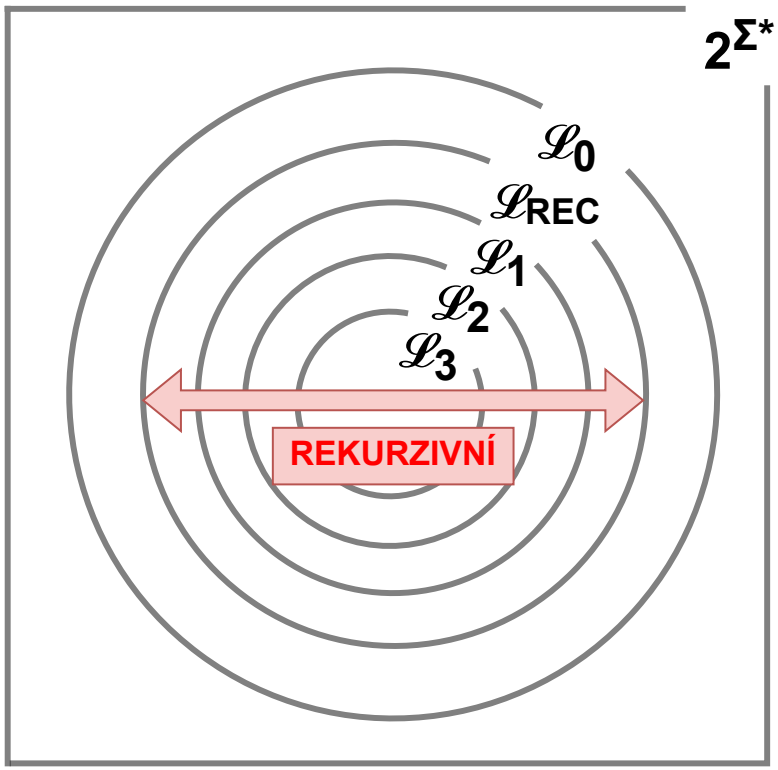
Sestrojte vícepáskový TS přijímající jazyk $\{wcw \mid w \in \{a, b\}^*\}$



Důkazy rekurzivní vyčíslitelnosti

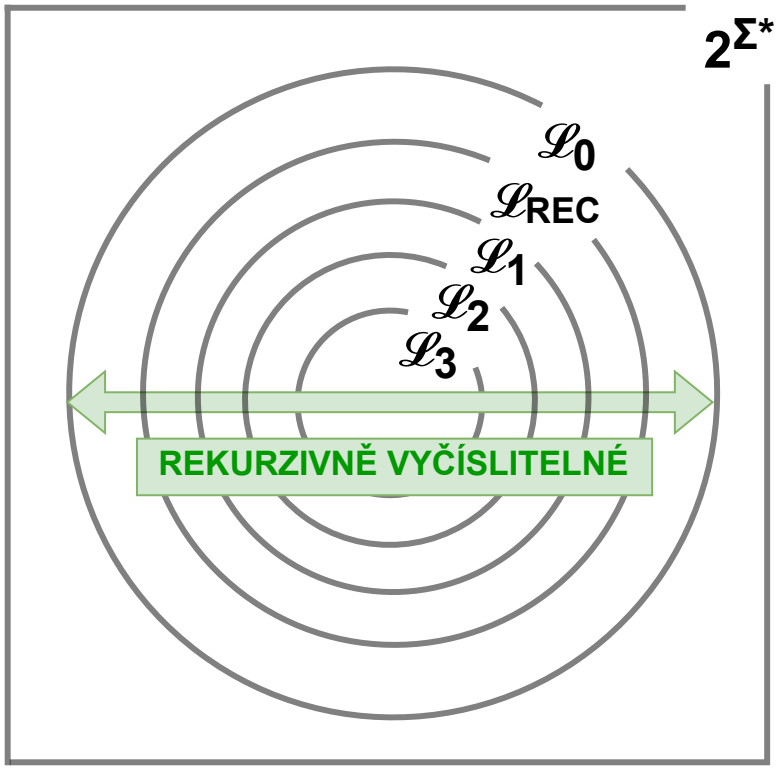
Rekurzivní jazyk

- 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 vyčíslitelnosti

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

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

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
  - ...

Důkaz rekurzivní vyčíslitelnosti

Dokažte, že následující jazyk L je rekurzivně vyčíslitelný:

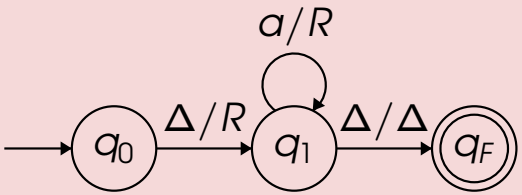
$L = \{\langle M \rangle \mid \langle M \rangle$ **je kód TS** M**, kde** $L(M) \neq \varnothing\}$

Důkaz rekurzivní vyčíslitelnosti

Dokažte, že následující jazyk L je rekurzivně vyčíslitelný:

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

Příklady řetězců patřících do jazyka L



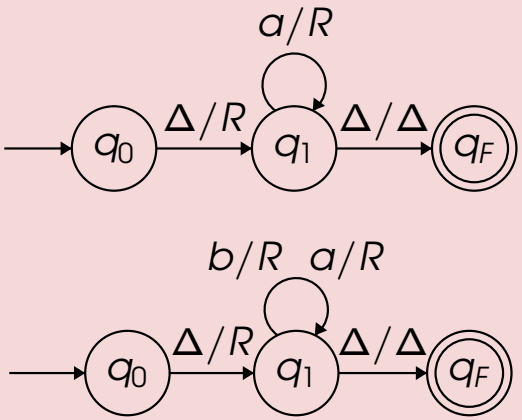
- $L(M_1) = \{a^n \mid n \geq 0\}$
- $\langle M_1 \rangle \in L$

Důkaz rekurzivní vyčíslitelnosti

Dokažte, že následující jazyk L je rekurzivně vyčíslitelný:

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

Příklady řetězců patřících do jazyka L



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

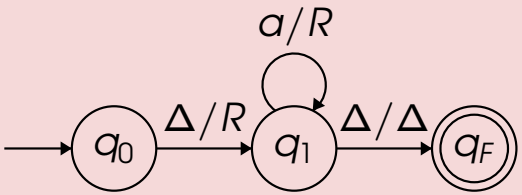
- $L(M_2) = \{a, b\}^*$
- $\langle M_2 \rangle \in L$

Důkaz rekurzivní vyčíslitelnosti

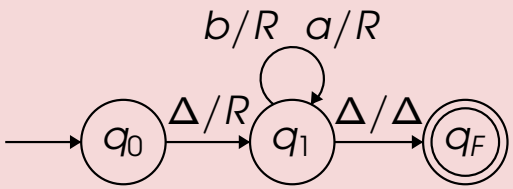
Dokažte, že následující jazyk L je rekurzivně vyčíslitelný:

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

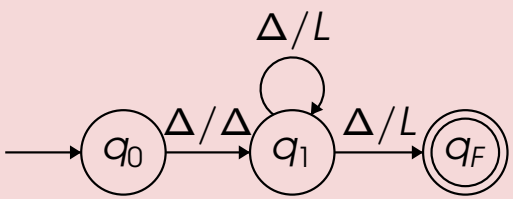
Příklady řetězců patřících do jazyka L



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



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



- $L(M_3) = \varnothing$
- $\langle M_3 \rangle \notin L$

Důkaz rekurzivní vyčíslitelnosti

Dokažte, že následující jazyk L je rekurzivně vyčíslitelný:

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

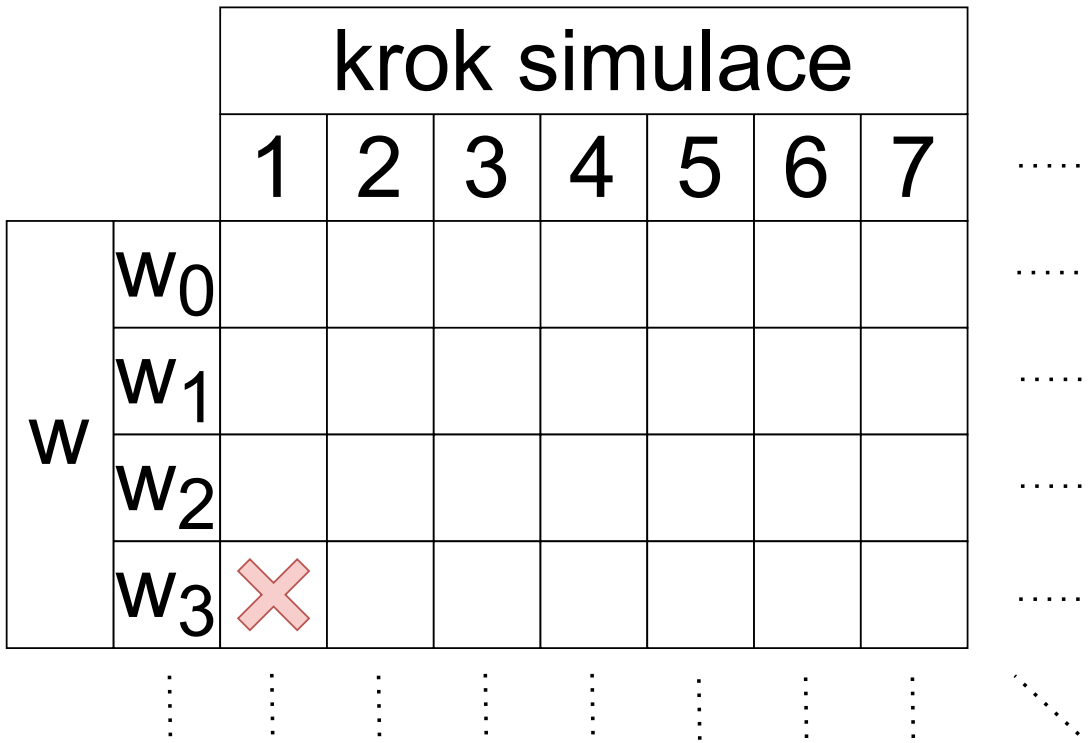
Důkaz rekurzivní vyčíslitelnosti

Dokažte, že následující jazyk L je rekurzivně vyčíslitelný:

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

Simulace nedeterministicky vygenerovaného slova

může skončit akceptováním



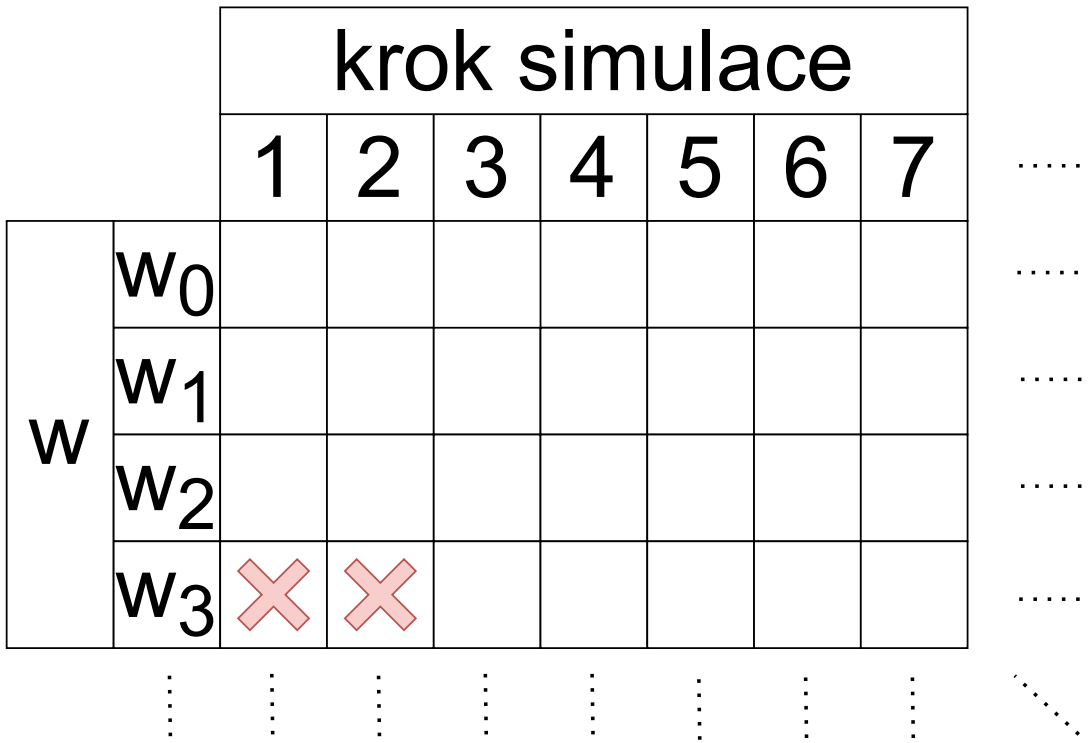
Důkaz rekurzivní vyčíslitelnosti

Dokažte, že následující jazyk L je rekurzivně vyčíslitelný:

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

Simulace nedeterministicky vygenerovaného slova

může skončit akceptováním



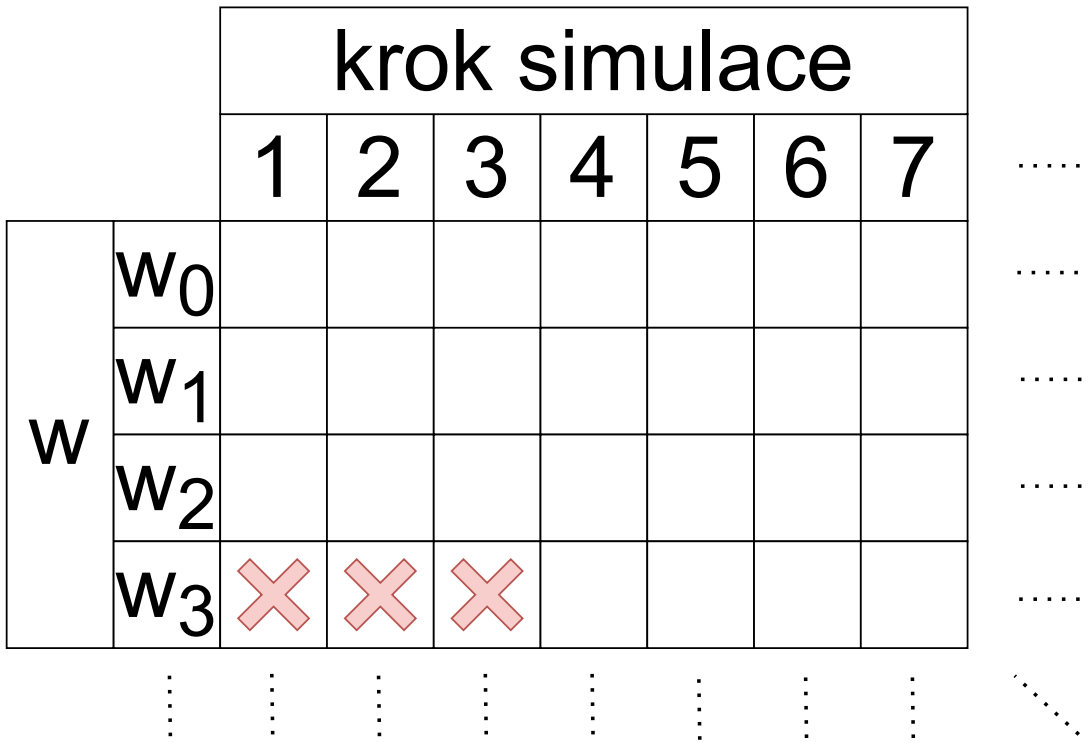
Důkaz rekurzivní vyčíslitelnosti

Dokažte, že následující jazyk L je rekurzivně vyčíslitelný:

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

Simulace nedeterministicky vygenerovaného slova

může skončit akceptováním



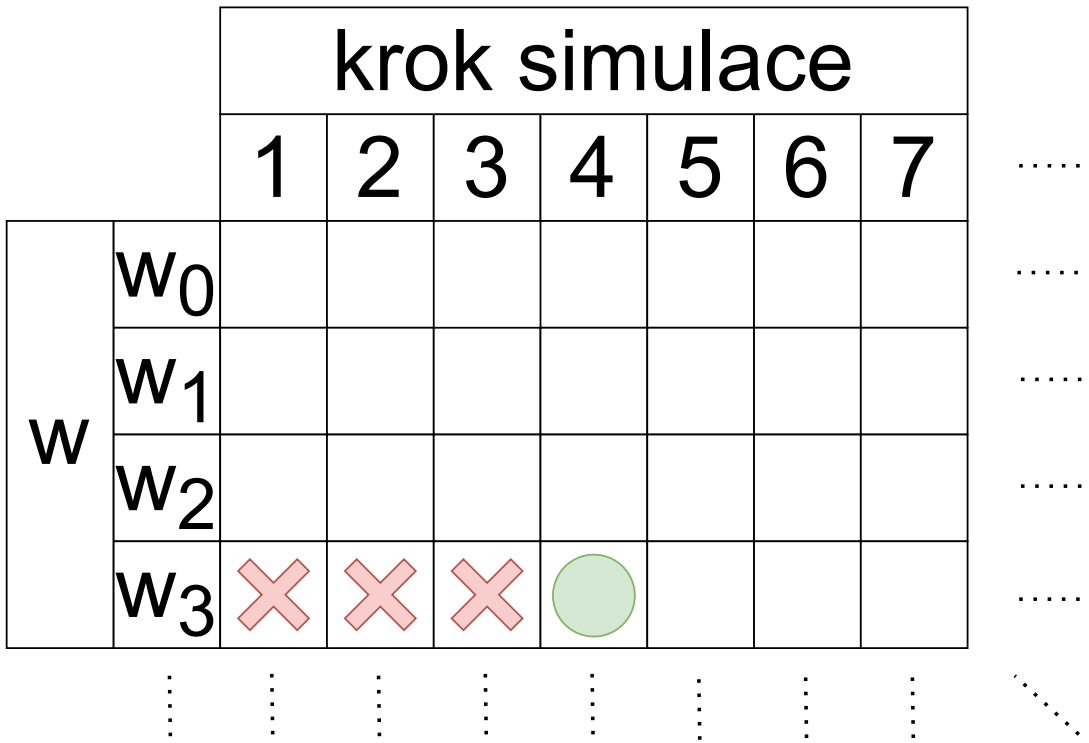
Důkaz rekurzivní vyčíslitelnosti

Dokažte, že následující jazyk L je rekurzivně vyčíslitelný:

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

Simulace nedeterministicky vygenerovaného slova

může skončit akceptováním



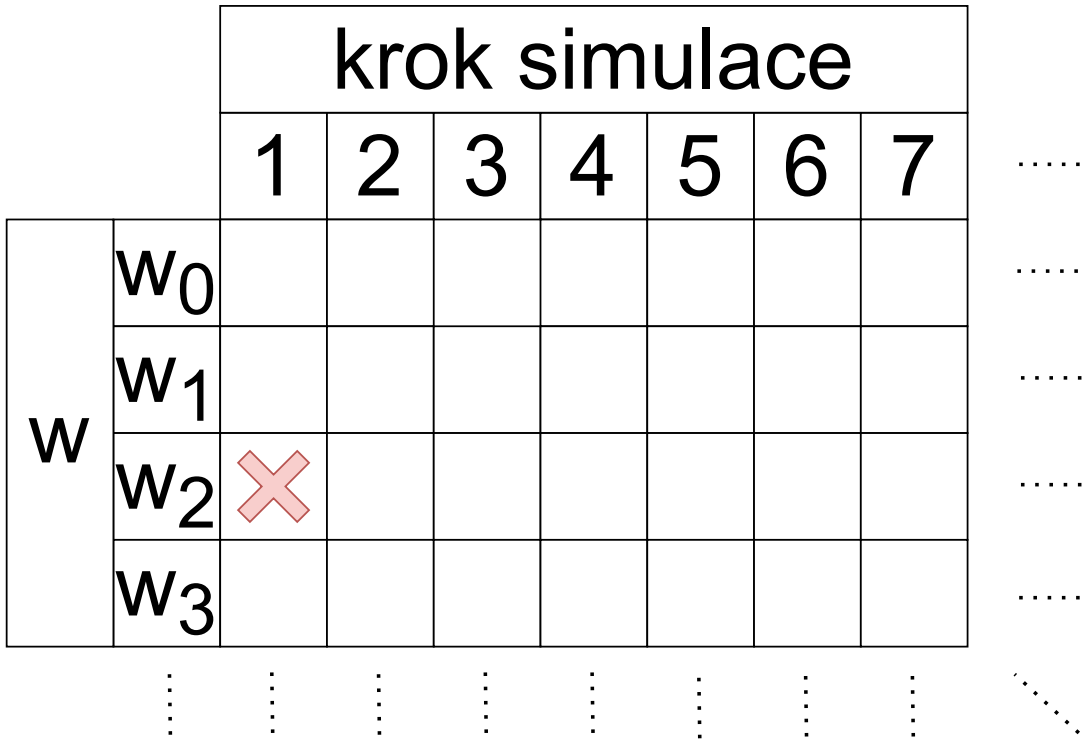
Důkaz rekurzivní vyčíslitelnosti

Dokažte, že následující jazyk L je rekurzivně vyčíslitelný:

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

Simulace nedeterministicky vygenerovaného slova

může skončit odmítnutím



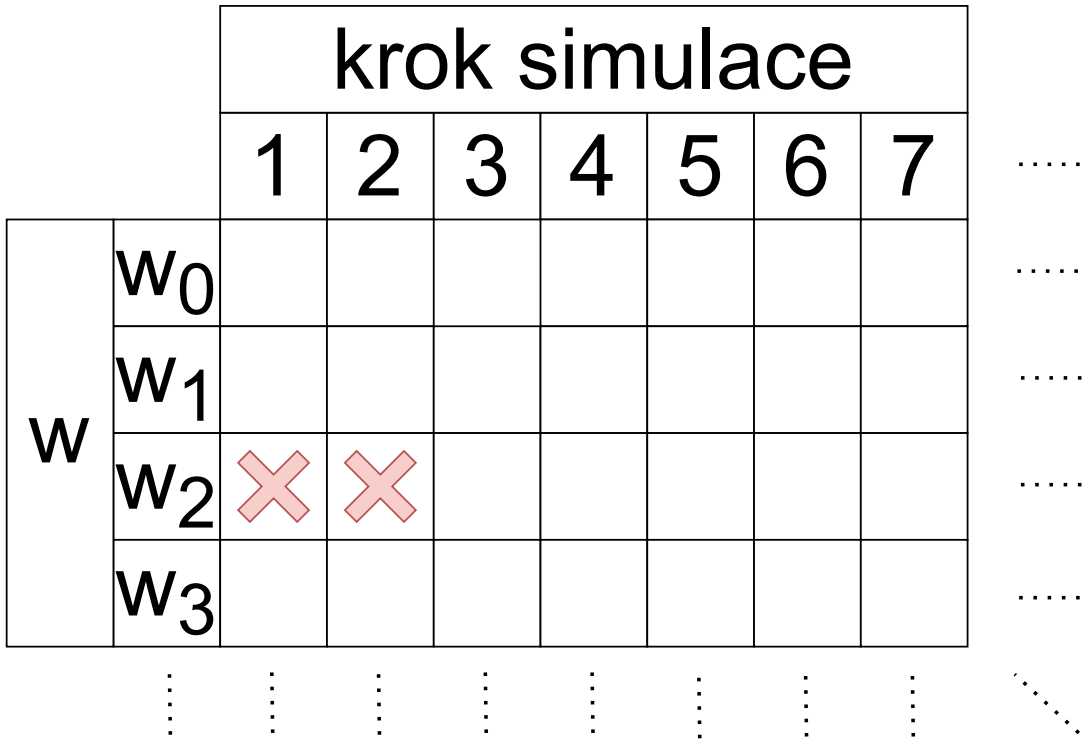
Důkaz rekurzivní vyčíslitelnosti

Dokažte, že následující jazyk L je rekurzivně vyčíslitelný:

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

Simulace nedeterministicky vygenerovaného slova

může skončit odmítnutím



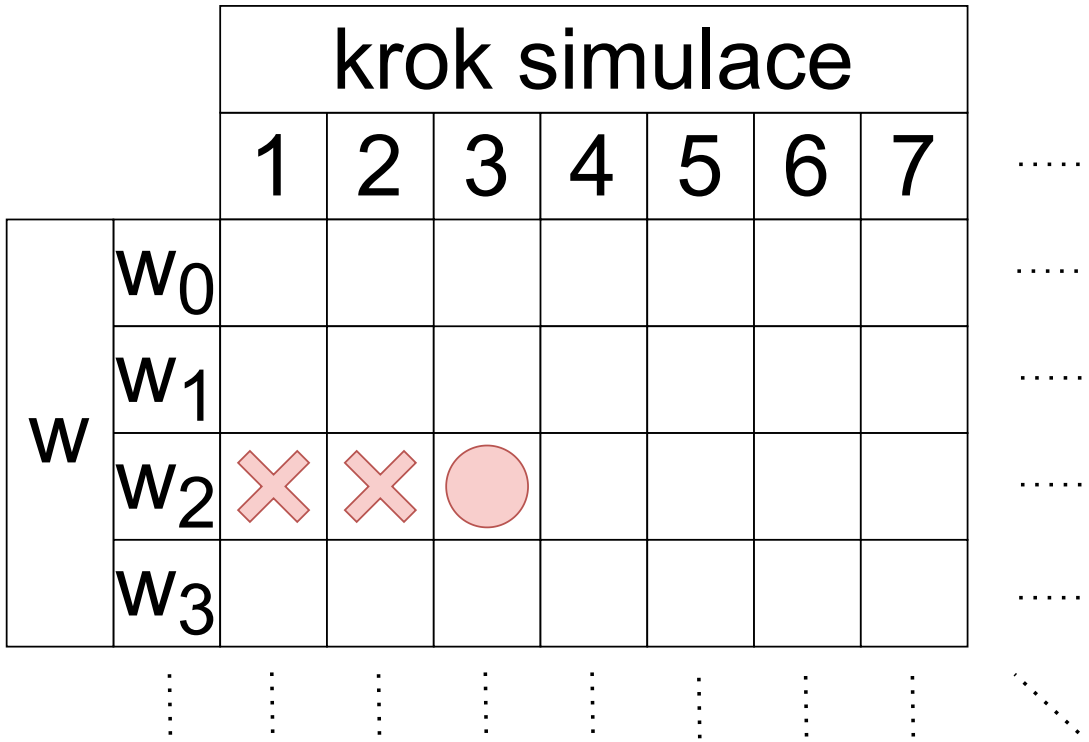
Důkaz rekurzivní vyčíslitelnosti

Dokažte, že následující jazyk L je rekurzivně vyčíslitelný:

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

Simulace nedeterministicky vygenerovaného slova

může skončit odmítnutím



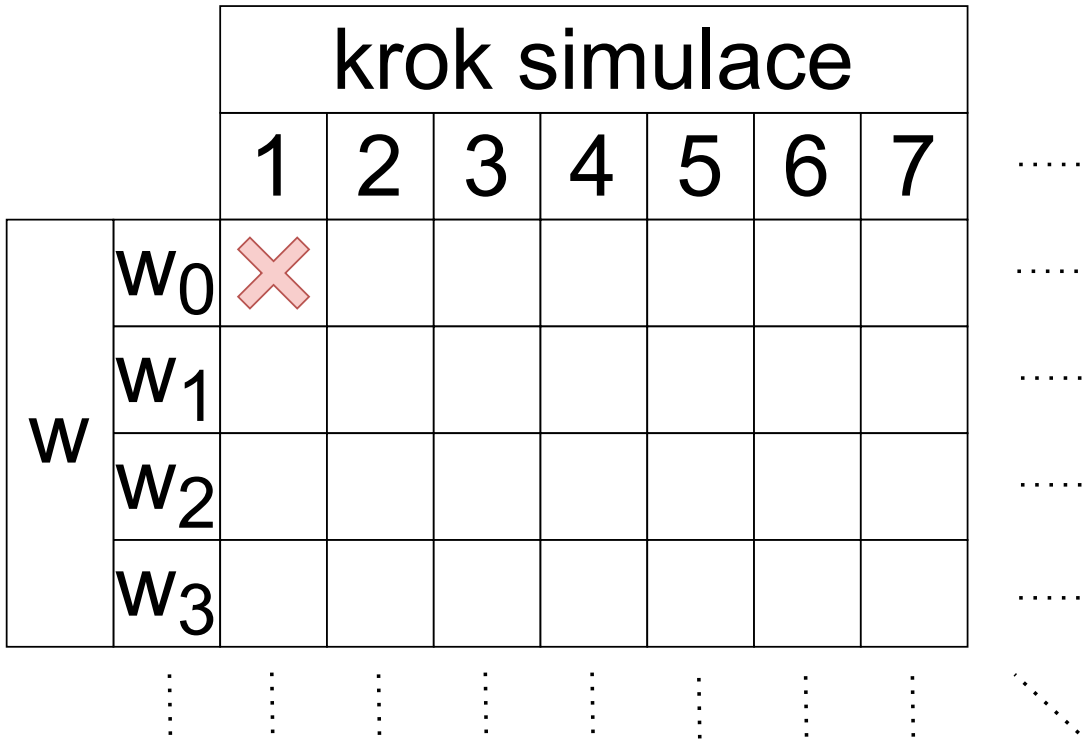
Důkaz rekurzivní vyčíslitelnosti

Dokažte, že následující jazyk L je rekurzivně vyčíslitelný:

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

Simulace nedeterministicky vygenerovaného slova

může cyklit



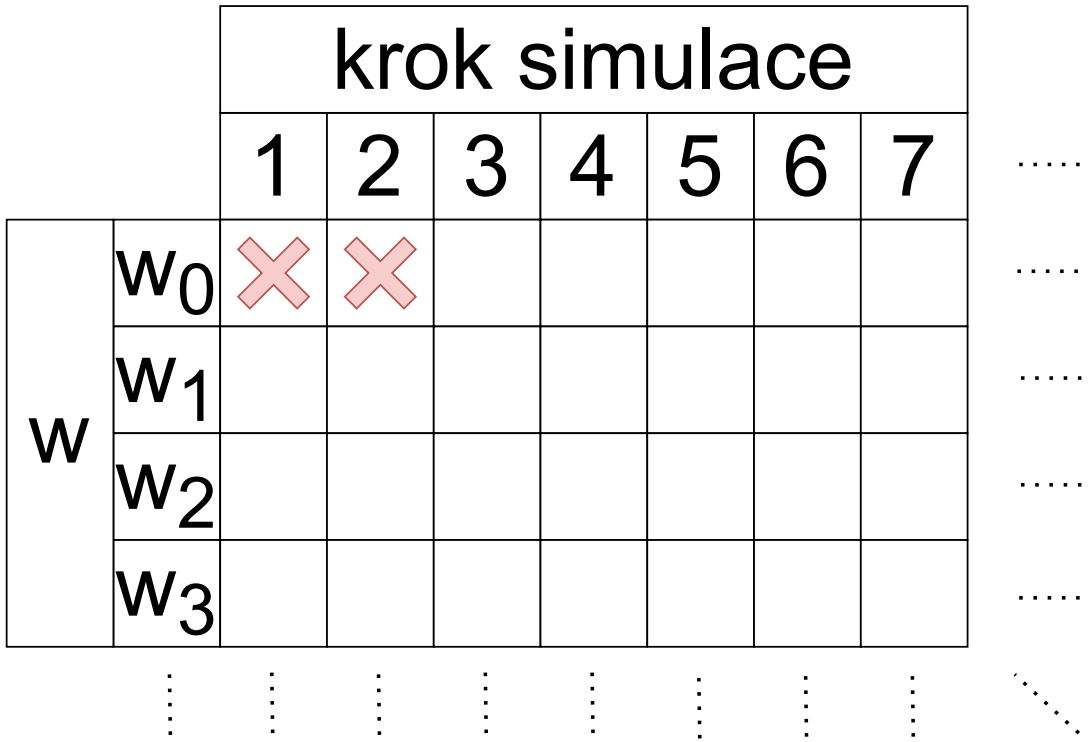
Důkaz rekurzivní vyčíslitelnosti

Dokažte, že následující jazyk L je rekurzivně vyčíslitelný:

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

Simulace nedeterministicky vygenerovaného slova

může cyklit



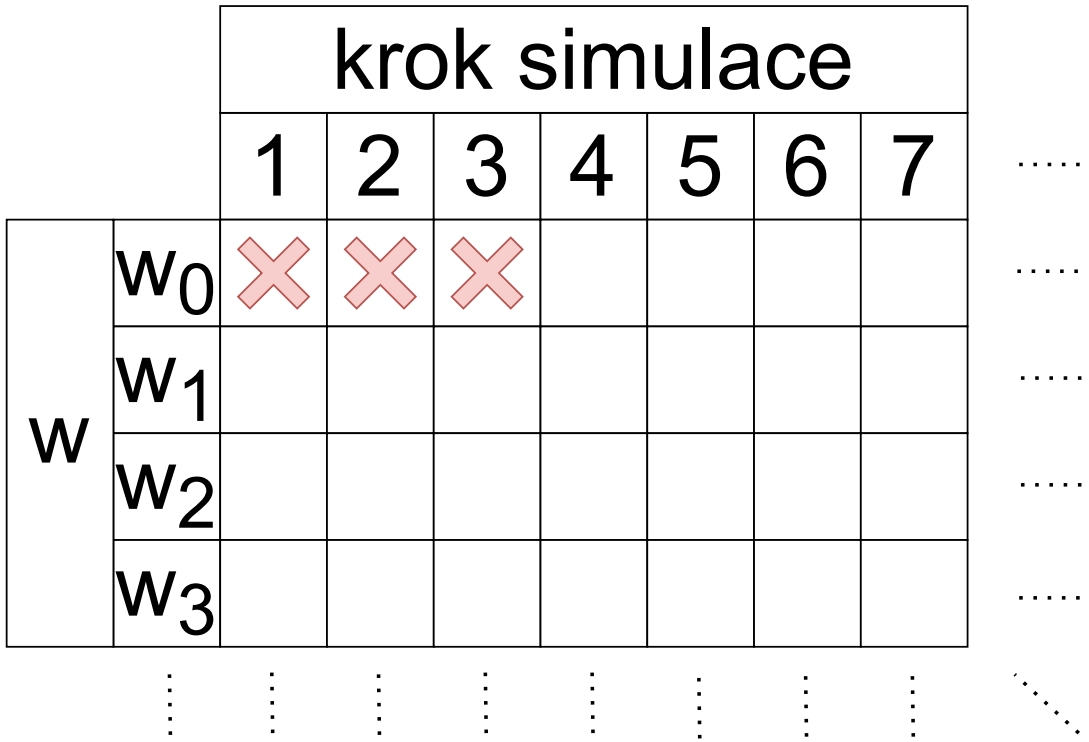
Důkaz rekurzivní vyčíslitelnosti

Dokažte, že následující jazyk L je rekurzivně vyčíslitelný:

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

Simulace nedeterministicky vygenerovaného slova

může cyklit



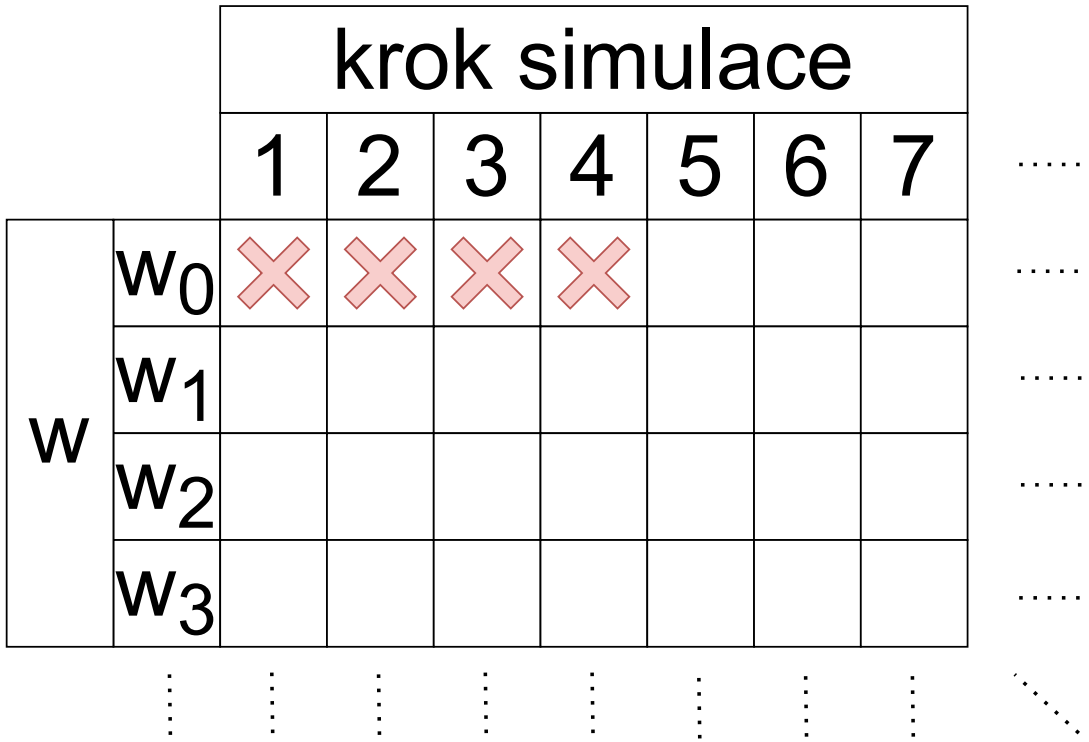
Důkaz rekurzivní vyčíslitelnosti

Dokažte, že následující jazyk L je rekurzivně vyčíslitelný:

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

Simulace nedeterministicky vygenerovaného slova

může cyklit



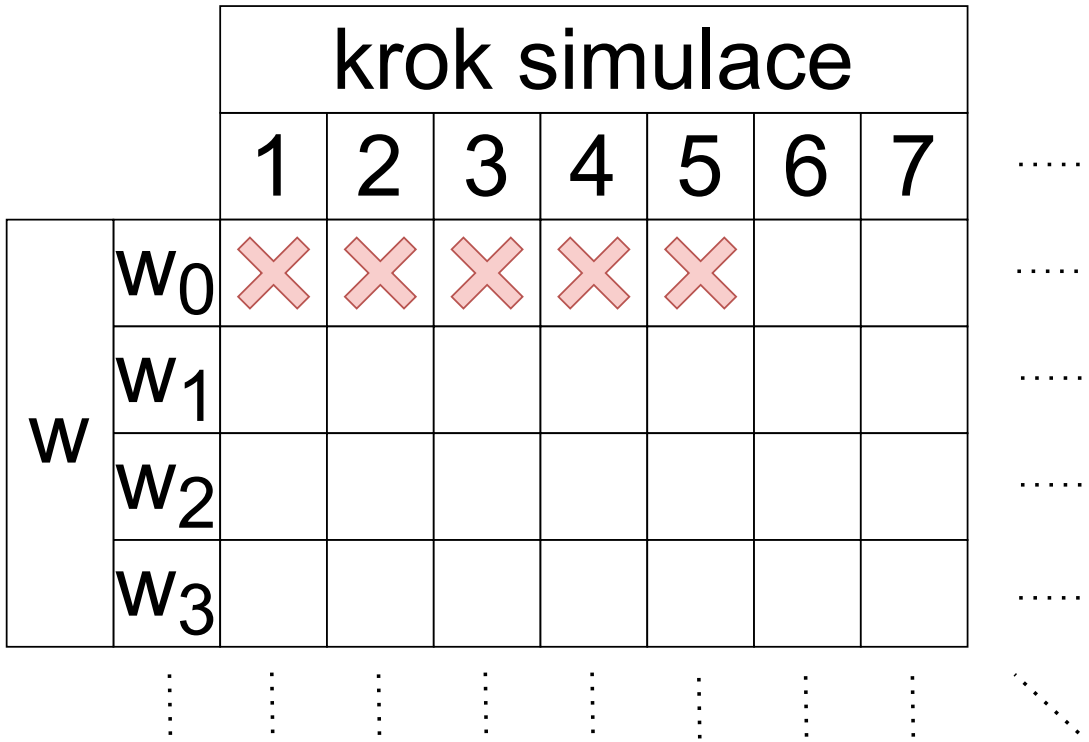
Důkaz rekurzivní vyčíslitelnosti

Dokažte, že následující jazyk L je rekurzivně vyčíslitelný:

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

Simulace nedeterministicky vygenerovaného slova

může cyklit



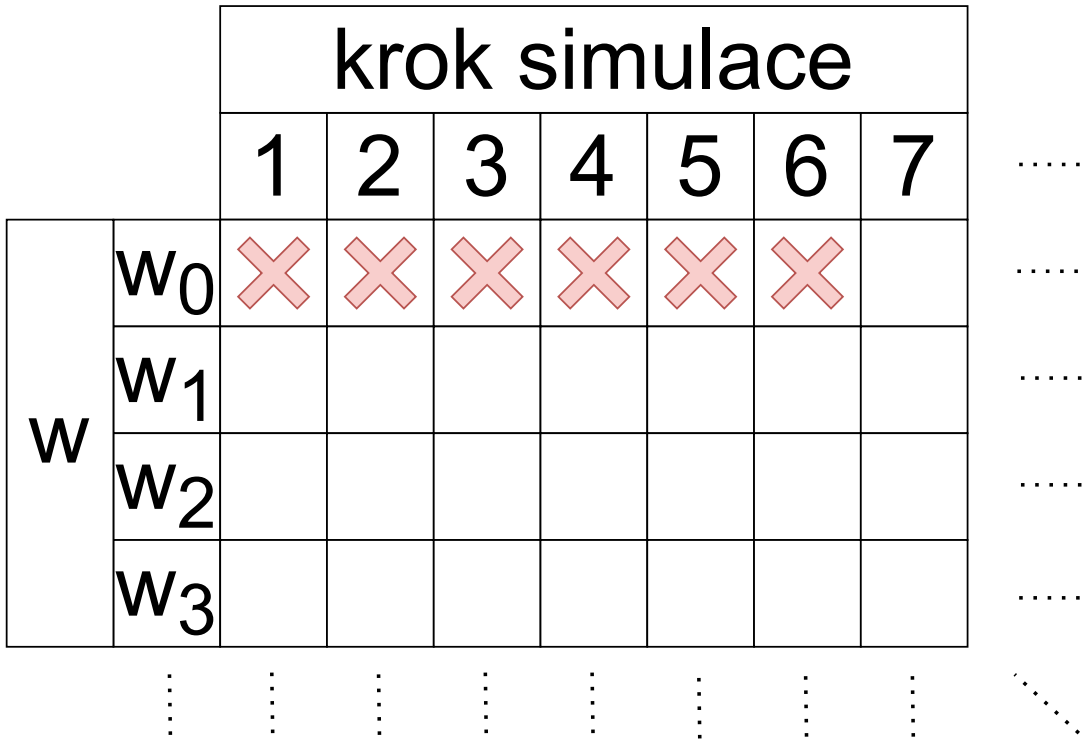
Důkaz rekurzivní vyčíslitelnosti

Dokažte, že následující jazyk L je rekurzivně vyčíslitelný:

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

Simulace nedeterministicky vygenerovaného slova

může cyklit



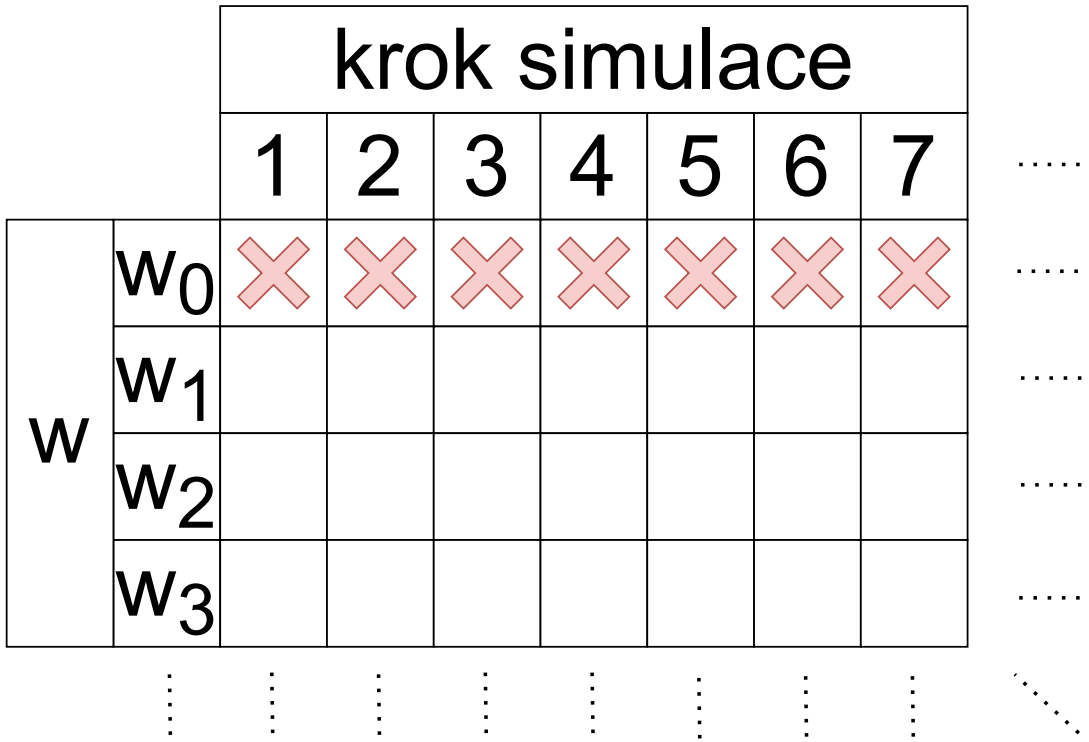
Důkaz rekurzivní vyčíslitelnosti

Dokažte, že následující jazyk L je rekurzivně vyčíslitelný:

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

Simulace nedeterministicky vygenerovaného slova

může cyklit



Důkaz rekurzivní vyčíslitelnosti

Dokažte, že následující jazyk L je rekurzivně vyčíslitelný:

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

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:

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

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:

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

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:

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

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:
$L = \{\langle M \rangle \mid \langle M \rangle$ **je kód TS** M**, kde** $L(M) \neq \varnothing\}$

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:

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

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:

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

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:
$L = \{\langle M \rangle \mid \langle M \rangle$ **je kód TS** M**, kde** $L(M) \neq \varnothing\}$

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

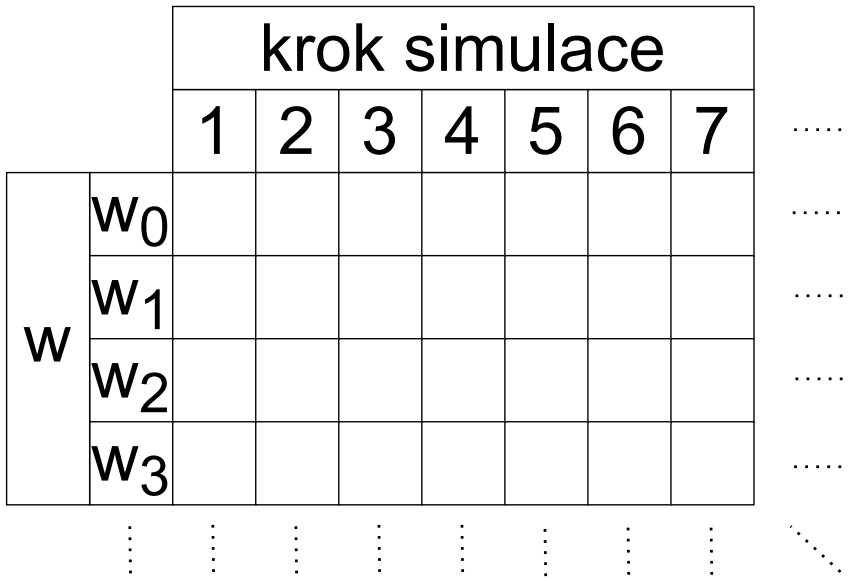
- Při použití deterministického Turingova stroje nelze provést krok nedeterministické volby řetězce w
- Je třeba systematicky generovat všechny řetězce nad abecedou simulovaného stroje

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)

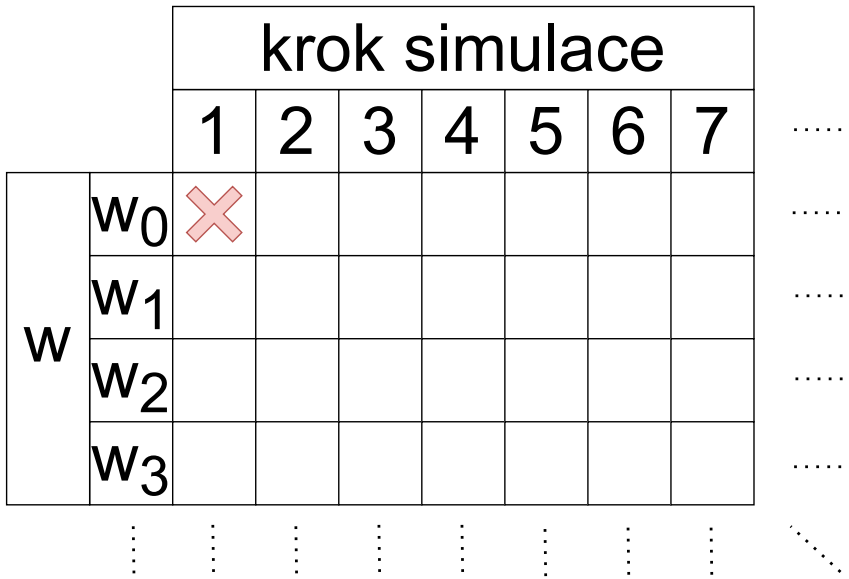


Determinismus

- 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)

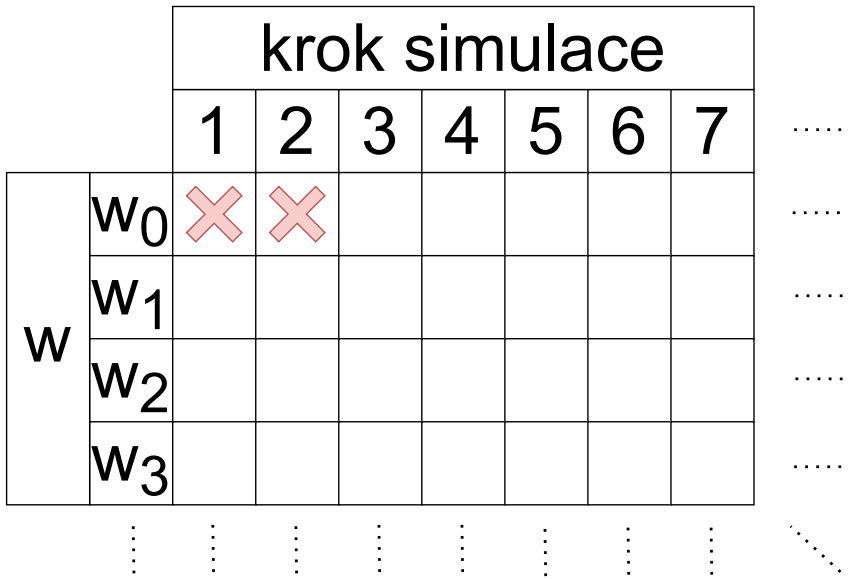


Determinismus

- 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)

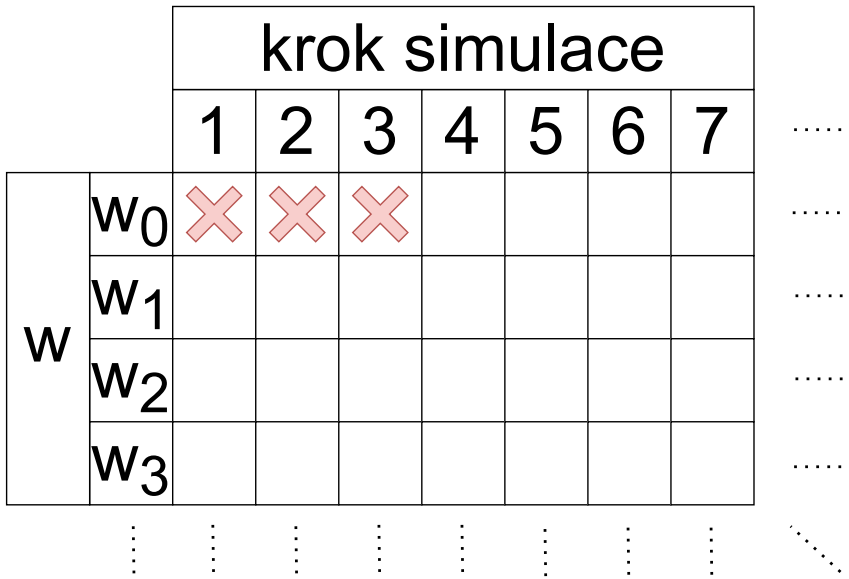


Determinismus

- 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)

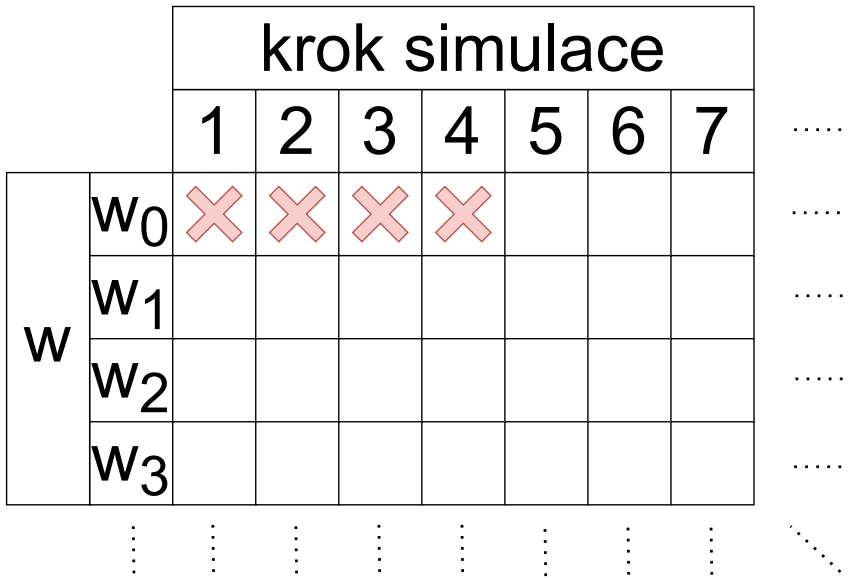


Determinismus

- 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)

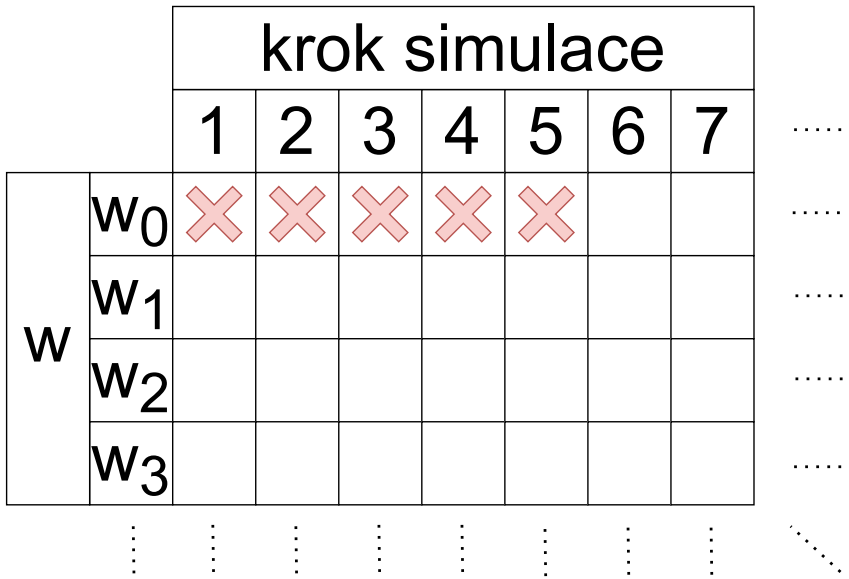


Determinismus

- 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)

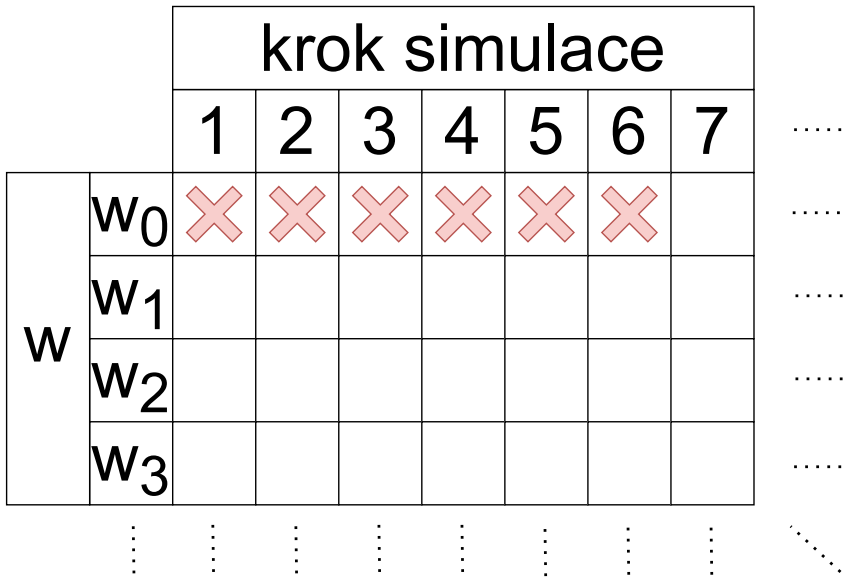


Determinismus

- 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)

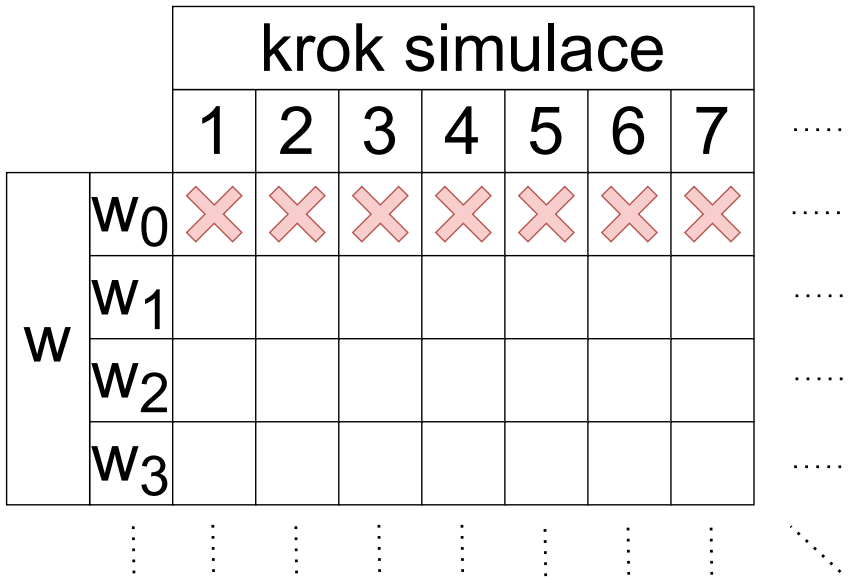


Determinismus

- 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)

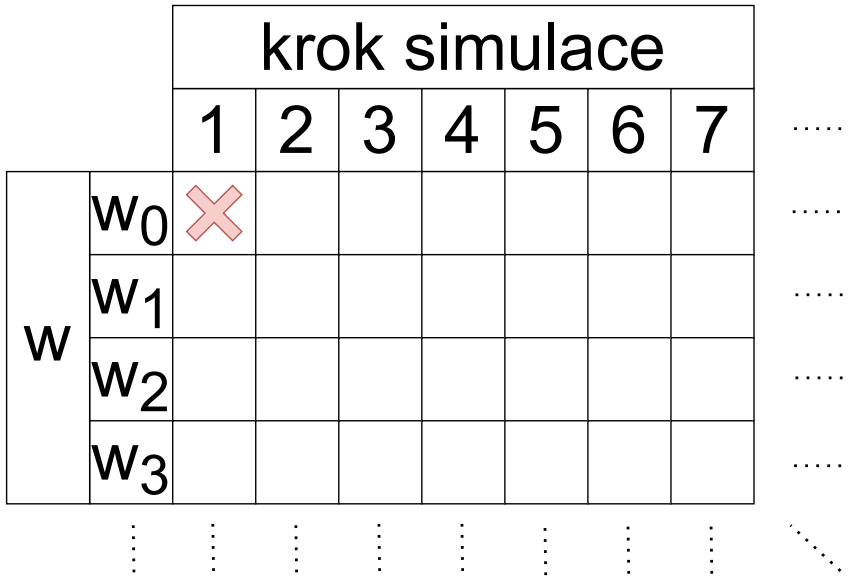


Determinismus

- 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)

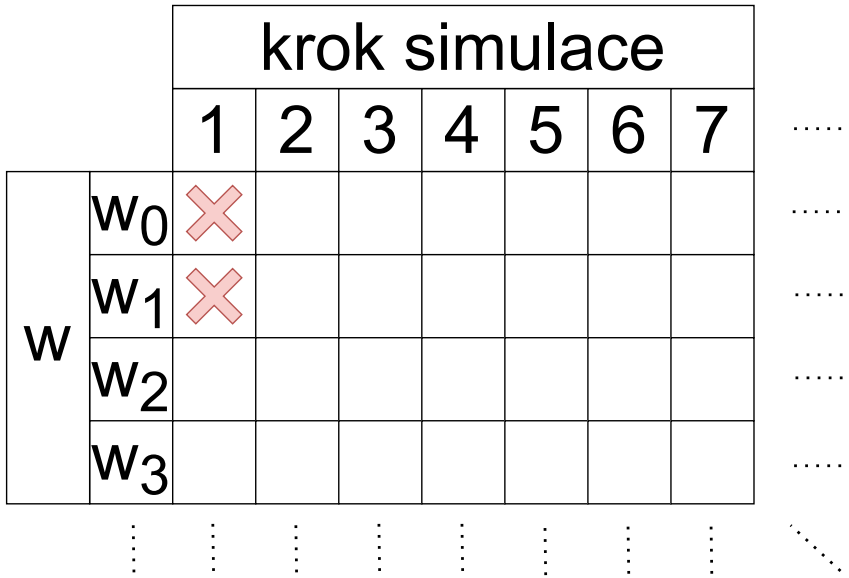


Determinismus

- 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)

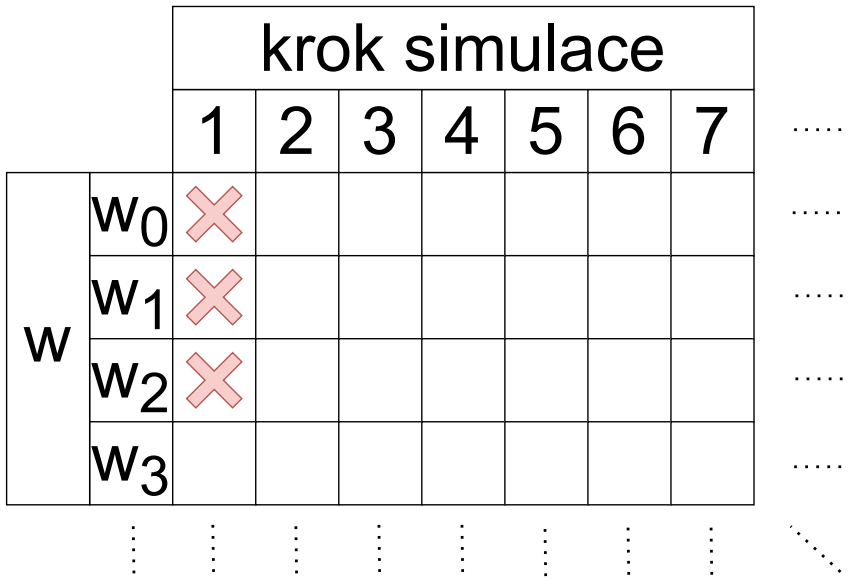


Determinismus

- 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)

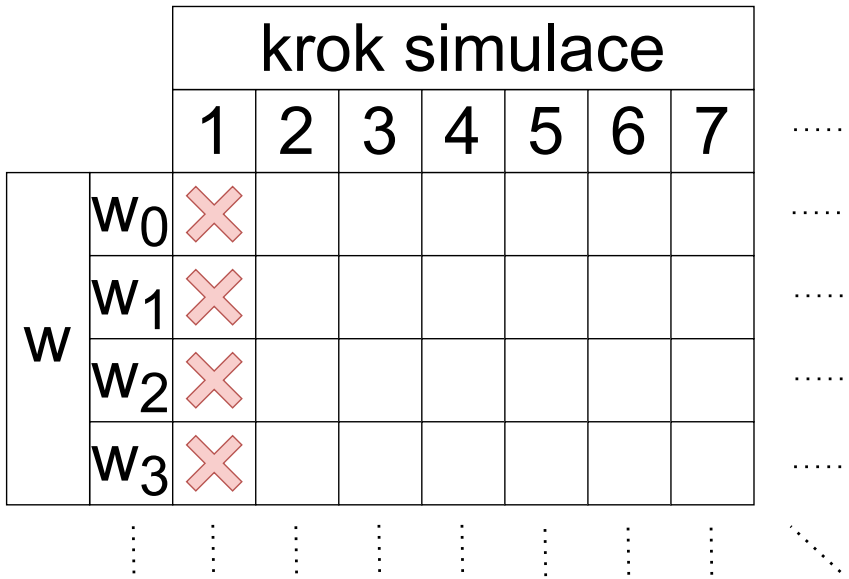


Determinismus

- 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)

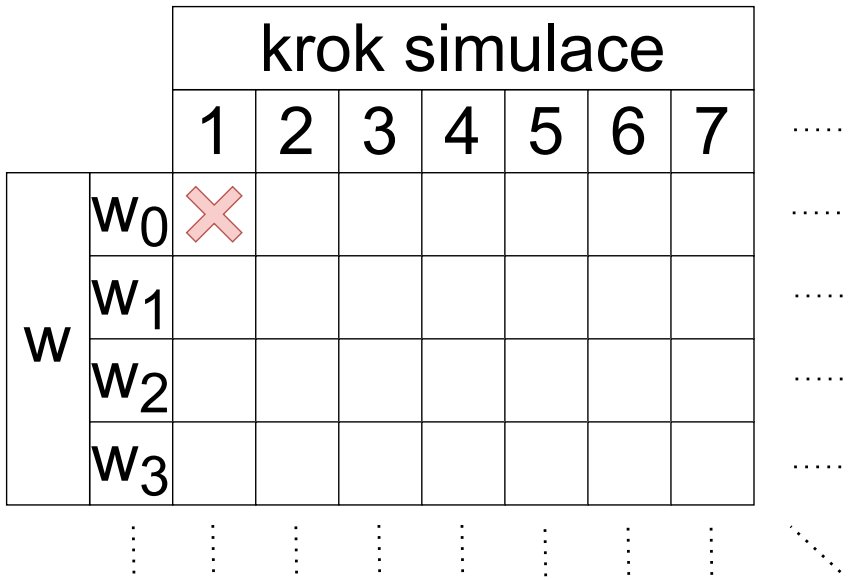


Determinismus

- 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)

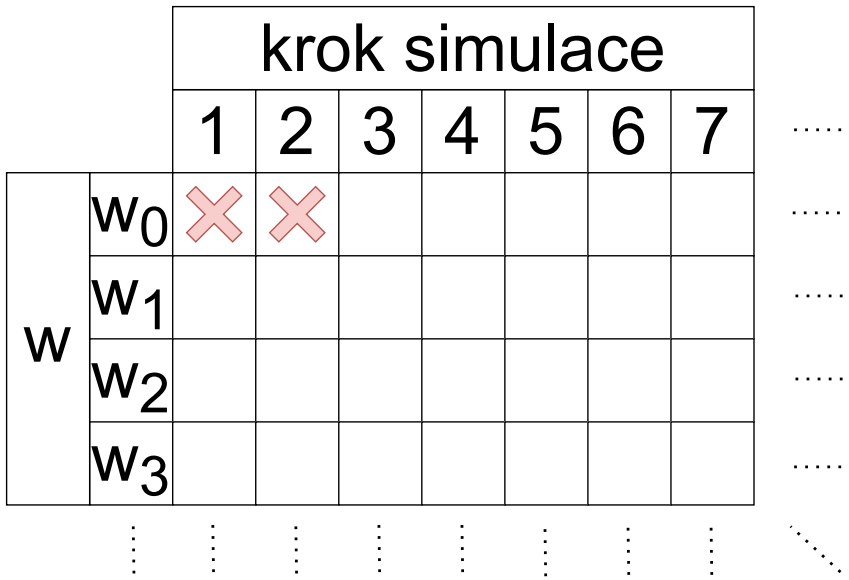


Determinismus

- 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)

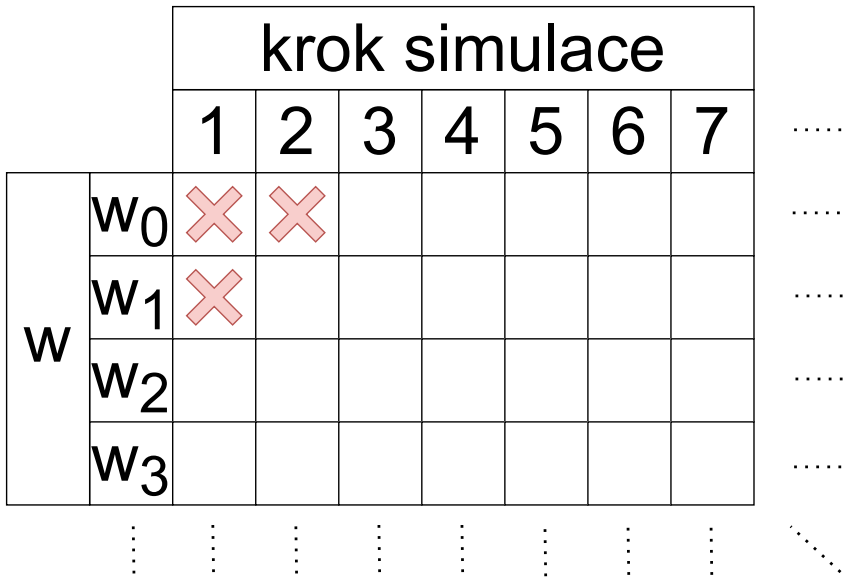


Determinismus

- 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)

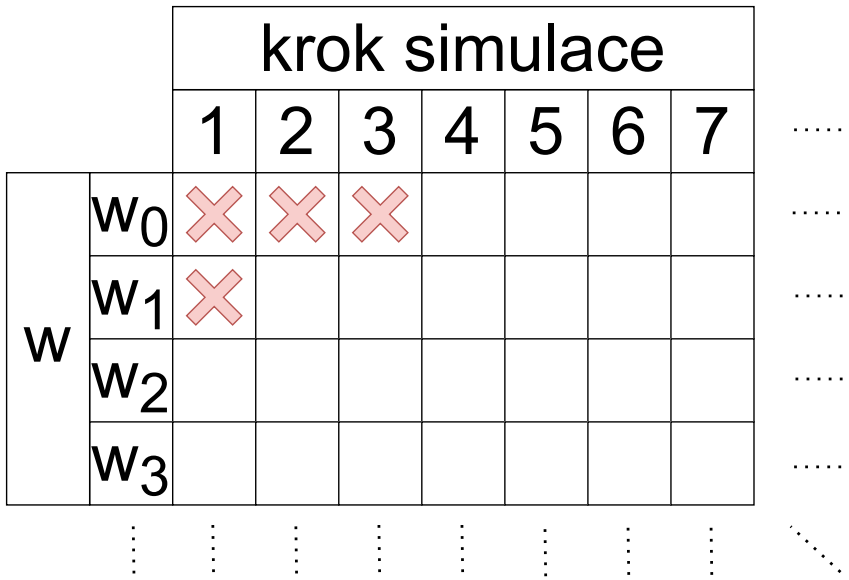


Determinismus

- 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)

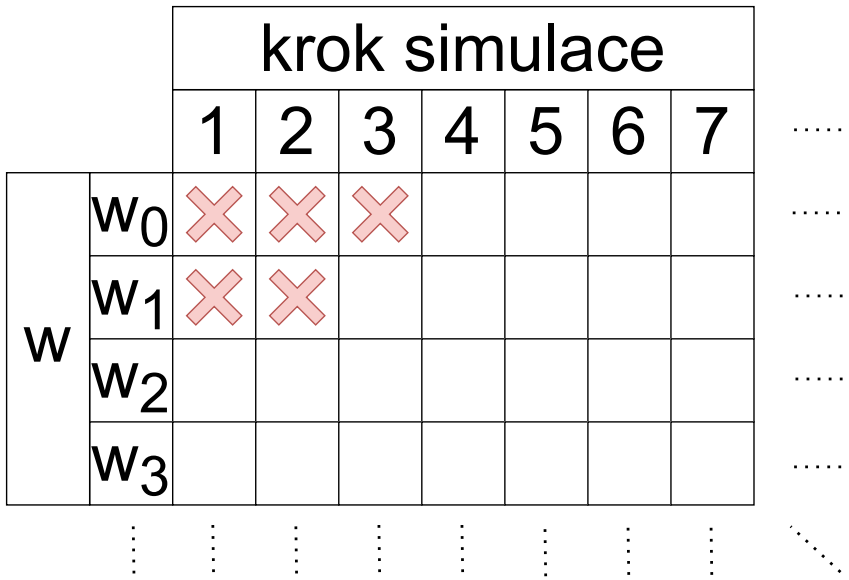


Determinismus

- 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)

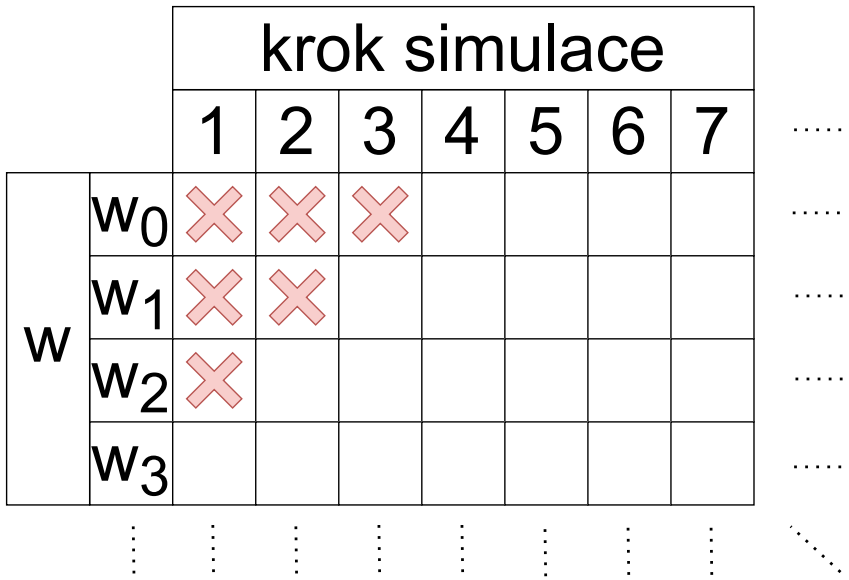


Determinismus

- 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)

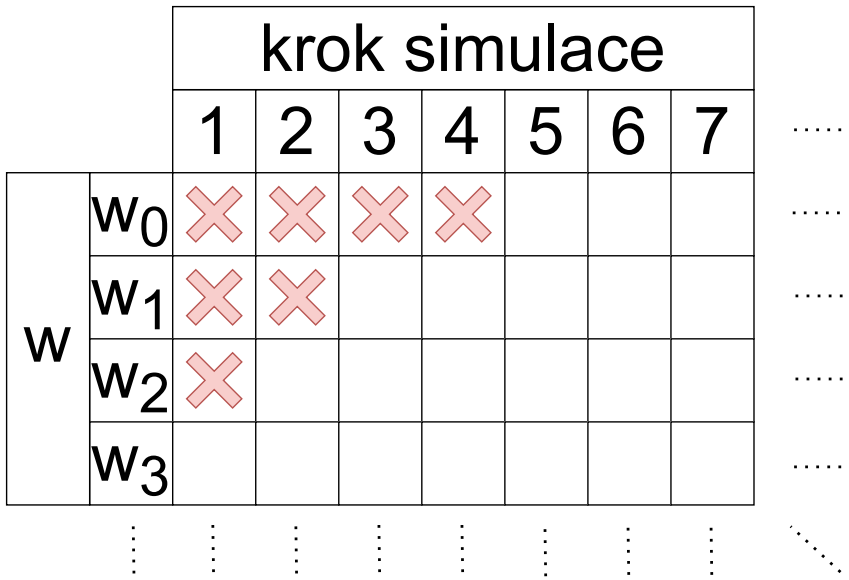


Determinismus

- 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)

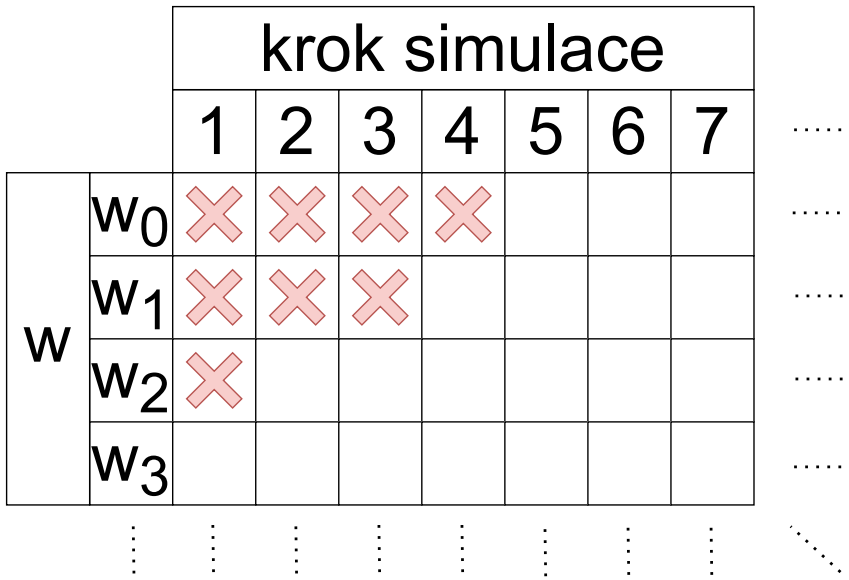


Determinismus

- 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)

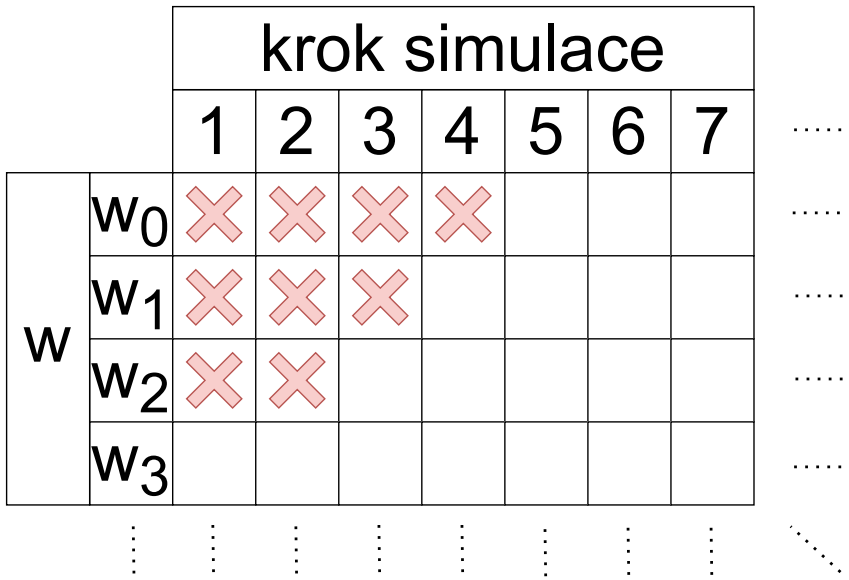


Determinismus

- 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)

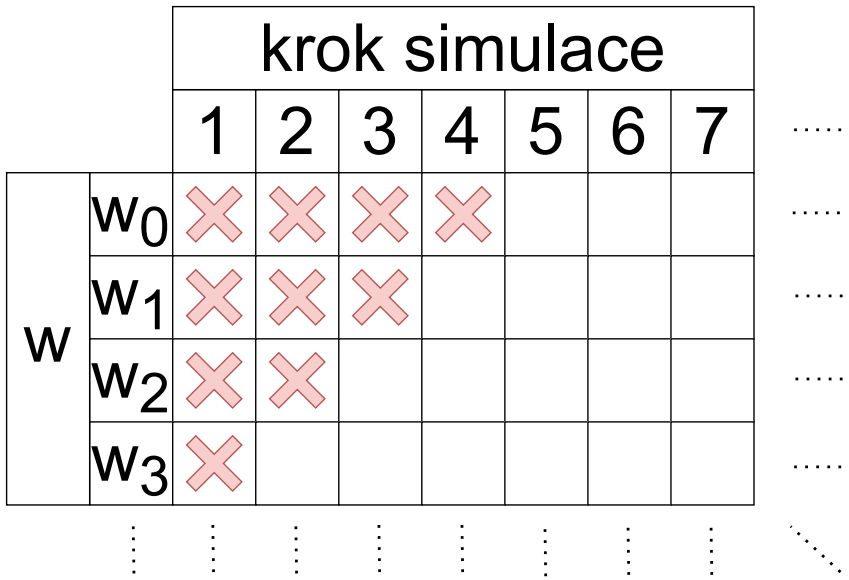


Determinismus

- 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)

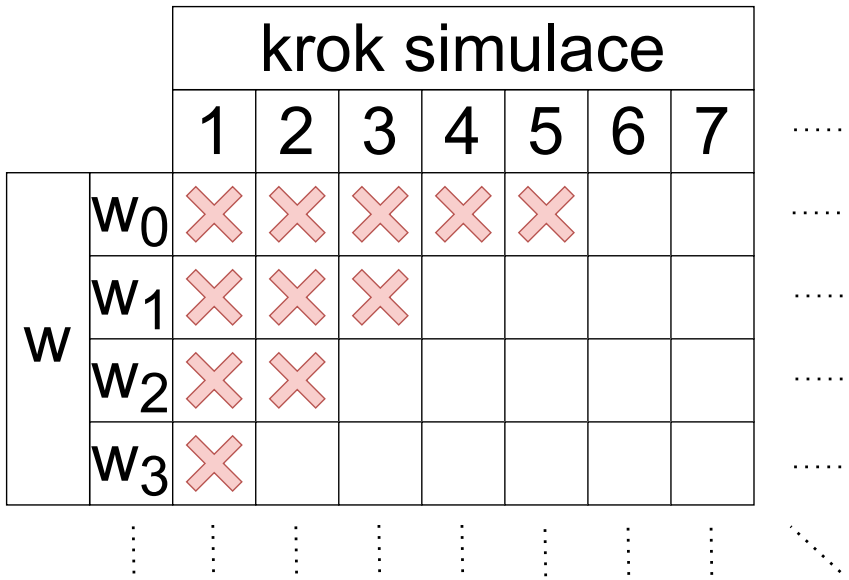


Determinismus

- 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)

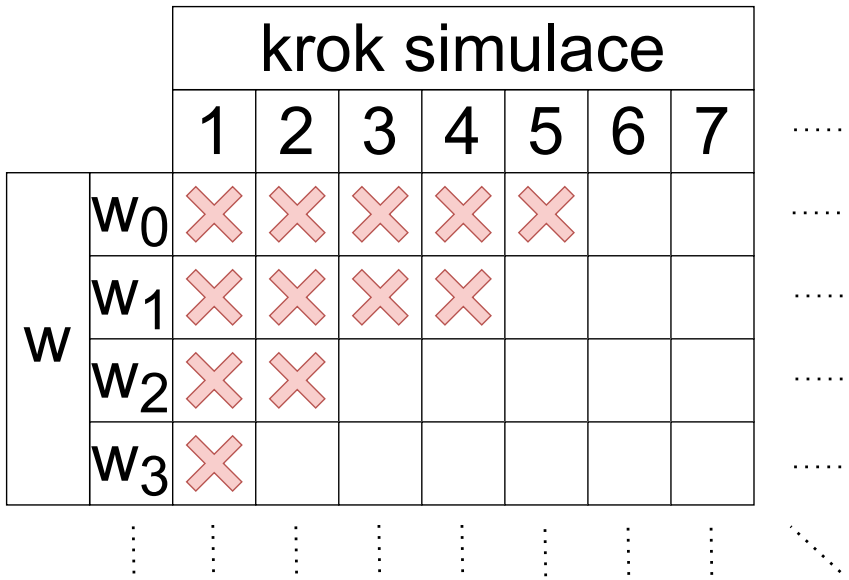


Determinismus

- 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)

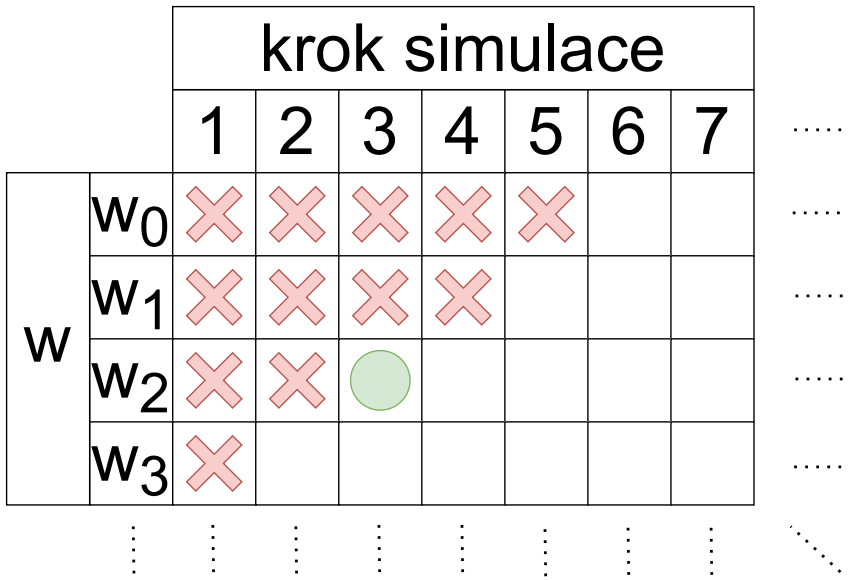


Determinismus

- 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)



Důkaz rekurzivní vyčíslitelnosti

Dokažte, že následující jazyk L je rekurzivně vyčíslitelný:

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

Důkaz rekurzivní vyčíslitelnosti

Dokažte, že následující jazyk L je rekurzivně vyčíslitelný:

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

Řešení pomocí **deterministického** Turingova stroje

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

Důkaz rekurzivní vyčíslitelnosti

Dokažte, že následující jazyk L je rekurzivně vyčíslitelný:

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

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:

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

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:

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

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 $w_0, w_1, w_2, \ldots$

Důkaz rekurzivní vyčíslitelnosti

Dokažte, že následující jazyk L je rekurzivně vyčíslitelný:

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

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 $w_0, w_1, w_2, \ldots$
- **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:

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

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

Důkaz rekurzivní vyčíslitelnosti

Dokažte, že následující jazyk L je rekurzivně vyčíslitelný:

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

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:

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

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
- Pokud některá simulace skončí akceptováním, **DTS** T akceptuje svůj vstup $\langle M \rangle$

Důkaz rekurzivní vyčíslitelnosti

Dokažte, že následující jazyk L je rekurzivně vyčíslitelný:

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

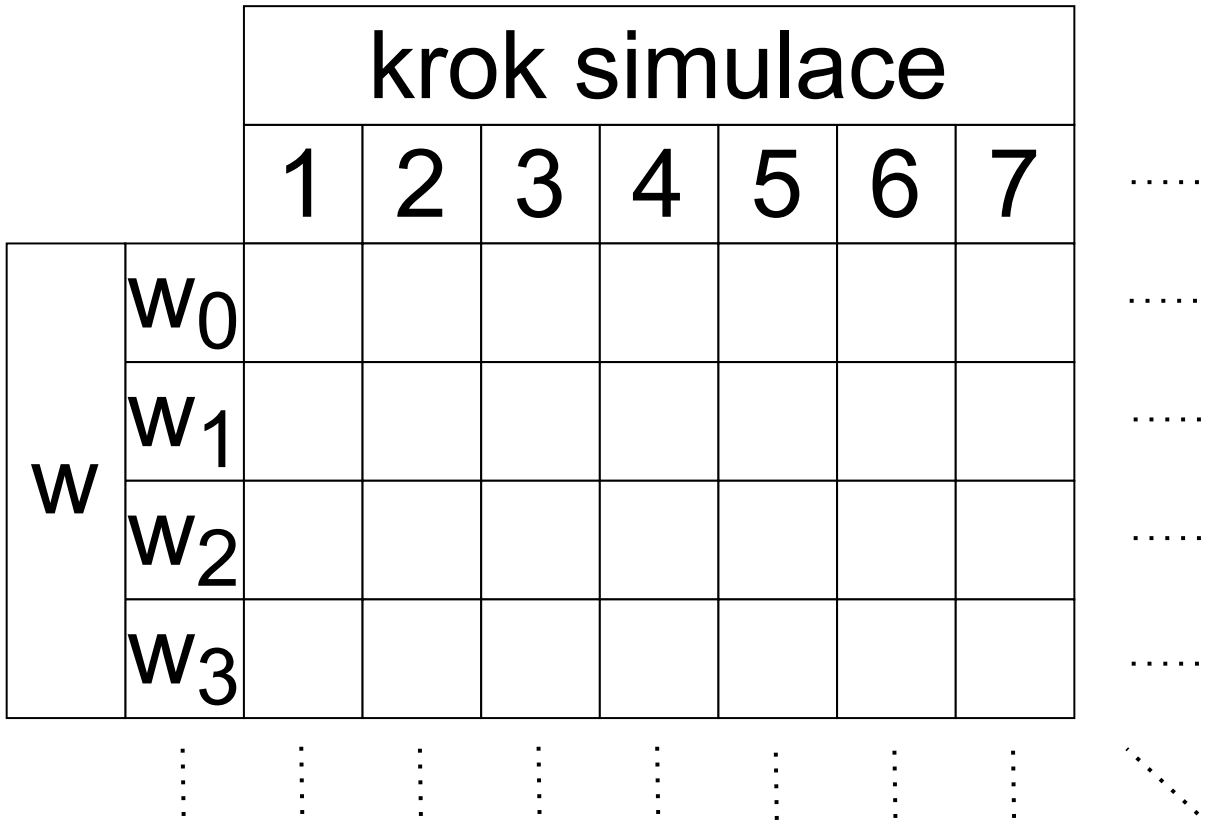
Řešení pomocí **deterministického** Turingova stroje

- **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
- 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:

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



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 $w_0, w_1, \ldots$ 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 $w_0, w_1, \ldots$ 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

Příklad 12

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