Teoretické základy informatických vied

Konečné automaty

Mgr. Martin Bobák, PhD.

Ústav informatiky
Slovenská akadémia vied

2022/2023

Pôvodný autor: doc. Mgr. Daniela Chudá, PhD. Teoretické základy informatických vied, FIIT STU, 2020.

Teória formálnych jazykov

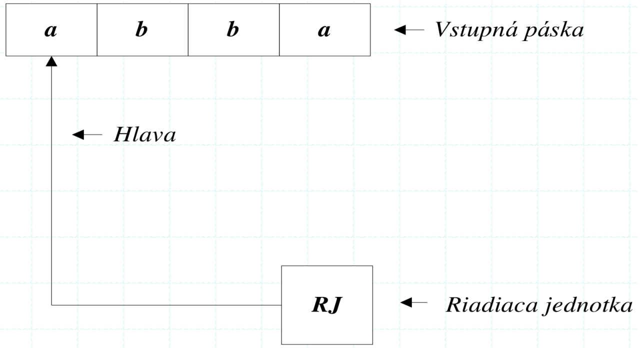
Reprezentácia jazyka:

- Matematický opis jazyka – **množina**.
- Systematické generovanie slov z jazyka (generovanie jazyka) – **gramatika**.
- Zostrojenie algoritmu, ktorý určí, či dané slovo patrí do jazyka (rozpoznanie jazyka) – **automat**

Teória jazykov:

- štúdium množín znakov, reťazcov, ich reprezentácií, štruktúr a vlastností

Výpočtový model



Zdroj: Daniela Chudá:Teoretické základy informatických vied, FIIT STU, 2020.

- riadiaca jednotka sa nachádza v jednom zo stavov (ich počet je konečný)
- hlava číta slovo zo vstupnej pásky (napr. zľava doprava)
- podľa prečítaných symbolov mení svoj stav (nevie sa vrátiť na už prečítaný symbol)
- stav, v ktorom skončí po dočítaní slova rozhoduje, či slovo je akceptované alebo zamietnuté

Konečný automat

- konečný automat dostane vstupné slovo nad abecedou Σ na páske
- riadiaca jednotka sa nachádza na začiatku v počiatočnom stave (automat má práve jeden počiatočný stav)
- automat vie spracovať v jednom kroku práve jeden symbol. Prečíta ho čítacou hlavou. Na základe prečítaného symbolu a aktuálneho stavu, prejde podľa prechodovej funkcie do nasledujúceho stavu.
- čítacia hlava prečíta nasledujúci symbol. Posun na predchádzajúce symboly nie je možný.
- Ak po prečítaní celého vstupného slova skončí automat v akceptačnom stave, vstupné slovo akceptuje. Inak slovo neakceptuje.

Konečný automat

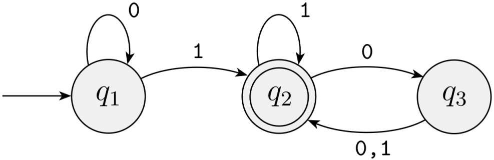
Typy konečných automatov:

- deterministický
- nedeterministický

Reprezentácia konečných automatov:

- stavový diagram
- maticou prechodovej funkcie
- formálnym zápisom

Konečný automat



Zdroj: M. Sipser. Introduction to the Theory of Computation. Third Boston, 2013.

Popis:

- Stavy: q_1, q_2, q_3
- Vstupná abeceda: $\{0, 1\}$
- Počiatočný stav: q_1
- Koncové (akceptačné) stavy: q_2

Prechodová funkcia δ („program automatu“):

	0	1
q_1	q_1	q_2
q_2	q_3	q_2
q_3	q_2	q_2,

Deterministický konečný automat

Definícia (Definícia deterministického konečného automatu)

Deterministický konečný automat *(DFA[a]) A pätica*
$A = (K, \Sigma, \delta, q_0, F)$, *kde:*
K je konečná množina stavov,
Σ je vstupná abeceda,
$\delta : K \times \Sigma \to K$ je prechodová funkcia
$q_0 \in K$ je počiatočný stav,
$F \subseteq K$ je množina koncových (akceptačných) stavov.

[a] ang. Deterministic finite automaton

Deterministický konečný automat

Definícia (Definícia konfigurácie DFA)

Konfigurácia *deterministického konečného automatu je usporiadaná dvojica*

$$(q, w) \in K \times \Sigma^*,$$

kde:
q je aktuálny stav automatu
w je nespracovaná časť vstupného slova.

Poznámky:

- (q_0, w) je počiatočná konfigurácia práve vtedy keď, q_0 je počiatočný stav a w je celé vstupné slovo.
- (q_F, ε) je akceptačná konfigurácia práve vtedy keď $q_F \in F$.
- konfigurácia automatu jednoznačne charakterizuje situáciu, v ktorej sa automat nachádza v danom okamihu výpočtu (niečo ako obsah celej pamäte počítača)

Deterministický konečný automat

Definícia (Definícia kroku výpočtu DFA)

Krok výpočtu *deterministického konečného automatu je relácia* $\vdash_A$ *na konfigurácii definovaná nasledovne:*

$$(q, au) \vdash_A (p, u) \iff \delta(q, a) = p,$$

pričom $p, q \in K$, $a \in \Sigma \cup \{\varepsilon\}$, $u \in \Sigma^*$.

Poznámky:

- krok výpočtu charakterizuje nasledujúcu konfiguráciu automatu A
- výpočet automatu je postupnosť konfigurácií, pričom každé dve po sebe nasledujúce konfigurácie sú v relácií krok výpočtu.
- automat postupne číta písmená vstupného slova a v stave si prenáša konečnú informáciu.

Definícia (Definícia jazyka akceptovaného DFA)

Jazyk *rozpoznávaný deterministickým konečným automatom (koncovým stavom) je množina*

$$L(A) = \{w \in \Sigma^* | \exists q \in F : (q_0, w) \vdash_A^* (q, \varepsilon)\},$$

Poznámky:

- $\vdash^*$ je reflexívno-tranzitívny uzáver relácie $\vdash$
- $Q_1 \vdash_A^* Q_2$ znamená, že automat A prejde z konfigurácie Q_1 do konfigurácie Q_2 na ľubovoľný konečný počet krokov.

Nech A = (K, Σ, δ, q_0, F), kde:
K = $\{q_0, q_1\}$,
$\Sigma = \{a, b\}$,
$\delta(q_0, a) = q_0$
$\delta(q_0, b) = q_1$
$\delta(q_1, a) = q_1$
$\delta(q_1, b) = q_0$
F = $\{q_1\}$.

$L(A) = \{w \in \{a, b\} | \sharp_b(w) = 2k + 1, k \in \mathbb{N}\}$

$(q_0, abbb) \vdash_A$
$(q_0, bbb) \vdash_A$
$(q_1, bb) \vdash_A$
$(q_0, b) \vdash_A (q_1, \varepsilon)$
$q_1 \in F \Rightarrow$ slovo abbb je **akceptované**

$(q_0, abba) \vdash_A$
$(q_0, bba) \vdash_A$
$(q_1, ba) \vdash_A$
$(q_0, a) \vdash_A (q_0, \varepsilon)$
$q_0 \notin F \Rightarrow$ slovo abba je **zamietnuté**

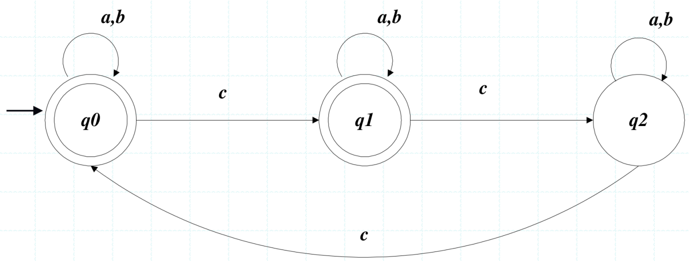
$$L_2 = \{w \in \{a, b, c\}^* | \sharp_c w = 3k \vee \sharp_c w = 3k + 1, k \in \mathbb{N}\}$$

$L = \{w \in \Sigma^* | f(a_1, a_2, ..., a_s) = \alpha k + \beta, k \in \mathbb{N}\}, s \leq |\Sigma|, 0 \leq \beta < \alpha$,

potom $A = (K, \Sigma, \delta, q_0, F), |K| = \alpha, F = \{q_\beta\}, |\delta| = \alpha \cdot |\Sigma|$.

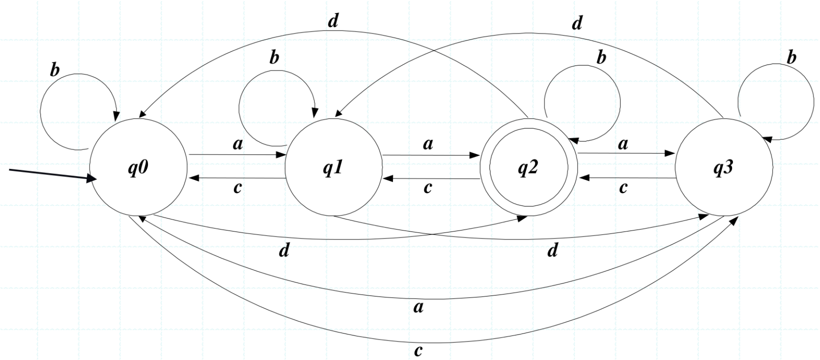
- počet stavov je rovný koeficientu α.
- majme stavy $q_0, q_1, ..., q_{\alpha-1}$, koncový stav je určený koeficientom β, tak, že $q_\beta \in F$
- konštruujeme prechodovú funkciu, realizujú sa prechody pre jednotlivé znaky. V prípade, že v ľavej strane obmedzujúcej podmienky sa daný znak vyskytuje s koeficientom nula, prechodová funkcia zostáva v tom istom stave, ak sa daný znak vyskytuje s koeficientom +1 prechodová funkcia sa preklápa do nasledovného stavu, ak sa ak sa daný znak vyskytuje s koeficientom -1 prechodová funkcia sa preklápa do predošlého stavu, ak sa daný znak vyskytuje s koeficientom +2 prechodová funkcia sa preklápa ob 2 do nasledovného stavu, s výskytmi ďalších znakov konštruujeme prechodovú funkciu analogicky.

$$L_2 = \{w \in \{a, b, c\}^* | \sharp_c w = 3k \vee \sharp_c w = 3k + 1, k \in \mathbb{N}\}$$



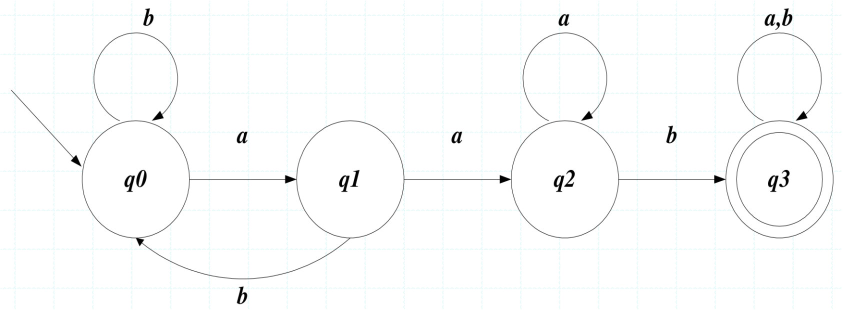
Zdroj: Daniela Chudá:Teoretické základy informatických vied, FIIT STU, 2020.

$$L_3 = \{w \in \{a,b,c,d\}^* | \sharp_a w - \sharp_c w + 2\sharp_d w = 4k+2, k \in \mathbb{N}\}$$



Zdroj: Daniela Chudá:Teoretické základy informatických vied, FIIT STU, 2020.

$$L_4 = \{xaaby | x, y \in \{a, b\}^*\}$$



Zdroj: Daniela Chudá:Teoretické základy informatických vied, FIIT STU, 2020.

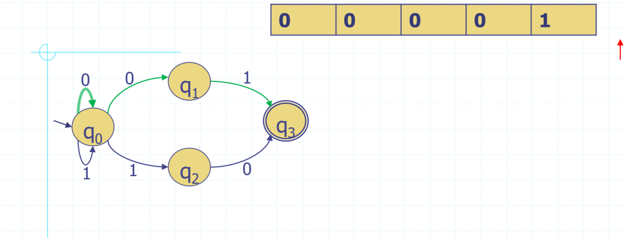
Nedeterministický konečný automat

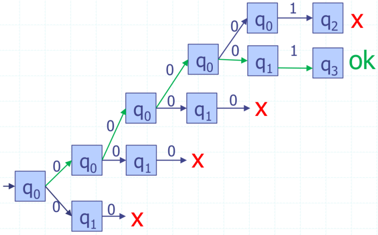
Nedeterminizmus – formálna abstrakcia takých výpočtov, v ktorých nie je jednoznačne určený nasledujúci krok (napr. konkurentné procesy, distribuované výpočty).

Základné vlastnosti nedeterministických konečných automatov:

- všeobecnejší typ deterministických konečných automatov
- umožníme automatu sa rozhodovať, čím môže vzniknúť na jednom slove viacero rôznych výpočtov:
  - aktuálny stav a načítaný vstupný symbol neurčujú jednoznačne nasledujúci stav
  - epsilonové kroky (t.j. zmeniť stav bez prečítania symbolu)
- zmena nastane v definícii automatu (prechodová funkcia) a kroku výpočtu.

Nedeterministický konečný automat





Zdroj: Daniela Chudá:Teoretické základy informatických vied, FIIT STU, 2020.

Definícia (Definícia nedeterministického konečného automatu)

Nedeterministický konečný automat *(NFA[a]) A pätica*
$A = (K, \Sigma, \delta, q_0, F)$, *kde:*
K *je konečná množina stavov,*
Σ *je vstupná abeceda,*
$\delta : K \times (\Sigma \cup \{\varepsilon\}) \to 2^K$ *je prechodové zobrazenie*
$q_0 \in K$ *je počiatočný stav,*
$F \subseteq K$ *je množina koncových (akceptačných) stavov.*

[a] ang. Nondeterministic finite automaton

Nedeterministický konečný automat

Definícia (Definícia kroku výpočtu NFA)

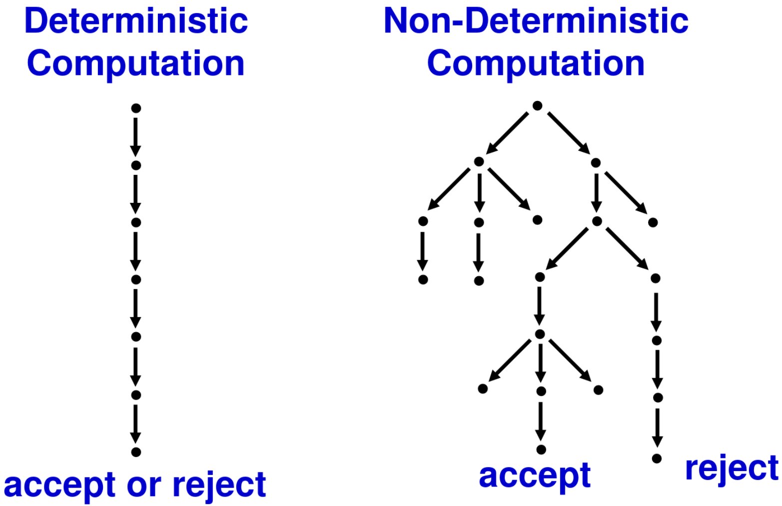
Krok výpočtu *nedeterministického konečného automatu je relácia* $\vdash_A$ *na konfigurácii definovaná nasledovne:*

$$(q, au) \vdash_A (p, u) \iff p \in \delta(q, a),$$

pričom $p, q \in K$, $a \in \Sigma \cup \{\varepsilon\}$, $u \in \Sigma^*$.

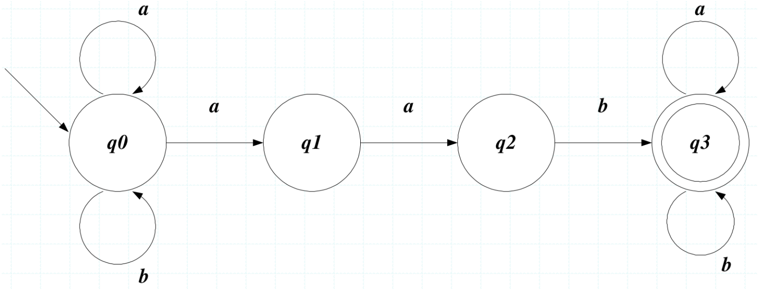
Zvyšné pojmy (konfigurácia, jazyk rozpoznaný konečným automatom) zostávajú nezmenené.

- Ak **existuje akceptujúci výpočet**, tak nedeterminizmus zabezpečí, že **konečný automat nájde tento výpočet**.
- Ak pre dané slovo akceptujúci výpočet neexistuje, nesmie existovať spôsob, ako sa do koncového stavu dostať (žiadnou kombináciou prechodov).
- V skutočnosti sa to rieši prehľadávaním všetkých možností – backtracking.



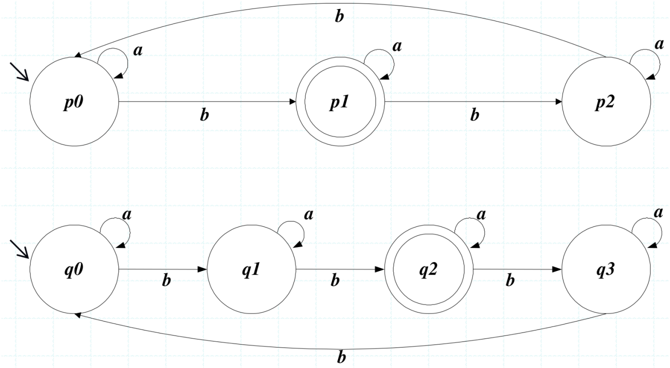
Zdroj: Sanjit A. Seshia:Computability and Complexity, UC Berkeley, 2010

$$L_4 = \{xaaby | x, y \in \{a, b\}^*\}$$



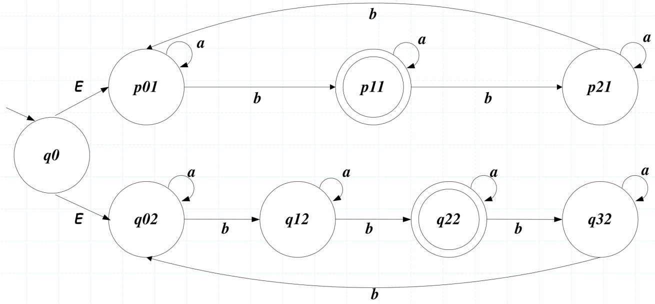
Zdroj: Daniela Chudá:Teoretické základy informatických vied, FIIT STU, 2020.

$$L = L_{2b} \cup L_{3b} \text{ pričom}$$
$$L_{2b} = \{w \in \{a, b\}^* | \sharp_b w = 3k + 1, k \in \mathbb{N}\}$$
$$L_{3b} = \{w \in \{a, b\}^* | \sharp_b w = 4k + 2, k \in \mathbb{N}\}$$



Zdroj: Daniela Chudá:Teoretické základy informatických vied, FIIT STU, 2020.

$$L = L_{2b} \cup L_{3b} \text{ pričom}$$
$$L_{2b} = \{w \in \{a,b\}^* | \sharp_b w = 3k+1, k \in \mathbb{N}\}$$
$$L_{3b} = \{w \in \{a,b\}^* | \sharp_b w = 4k+2, k \in \mathbb{N}\}$$



Zdroj: Daniela Chudá:Teoretické základy informatických vied, FIIT STU, 2020.

Trieda jazykov rozpoznávaných konečnými automatmi

Pre skúmanie triedy jazykov rozpoznávaných konečnými automatmi zaveďme nasledovné označenia:

- trieda jazykov rozpoznávaných deterministickými konečnými automatmi $\mathcal{L}(\mathrm{DFA})$,
- trieda jazykov rozpoznávaných nedeterministickými konečnými automatmi $\mathcal{L}(\mathrm{NFA})$,
- trieda jazykov rozpoznávaných konečnými automatmi $\mathcal{L}(\mathrm{FA})$.

Nedeterministický konečný automat

Veta

Trieda jazykov rozpoznávaných deterministickými konečnými automatmi je ekvivalentná s triedou jazykov rozpoznávaných nedeterministickými konečnými automatmi.

$$\mathcal{L}(DFA) = \mathcal{L}(NFA)$$

Ku každému nedeterministickému konečnému automatu môžeme zostrojiť deterministický konečný automat.

Veta

Trieda jazykov rozpoznávaných konečnými automatmi je zhodná s triedou všetkých regulárnych jazykov.

$$\mathcal{L}(FA) = \mathcal{R}$$

Nedeterministický konečný automat

Ku každému nedeterministickému konečnému automatu môžeme zostrojiť deterministický konečný automat.

Hlavná idea – odstránenie zdrojov nedeterminizmu:

1. odstránenie epsilonových krokov pomocou konštrukcie epsilonového uzáveru
2. transformácia prechodovej relácie na prechodovú funkciu

Definícia (Definícia epsilonový uzáver stavu NFA)

Majme NFA $A = (K, \Sigma, \delta, q_0, F)$. Nech $q \in K$, potom pre epsilonový uzáver stavu q, platí:

$$U(q)_\varepsilon = \{p | (q, \varepsilon) \vdash_A^* (p, \varepsilon)\}$$

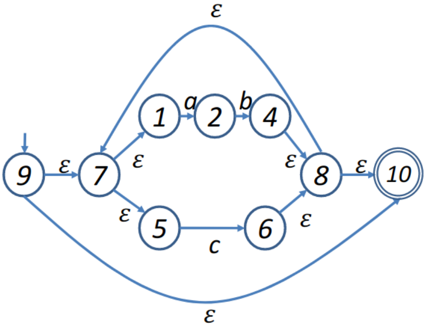
(množina stavov, do ktorých sa z q vieme v A dostať bez čítania písmena)

Vstup: Množina stavov T, $T \subseteq K$
Výstup: $U(T)_\varepsilon$
Premenné: Z - zásobník
begin

$Z \leftarrow T$ ▷ Ulož všetky stavy z T do Z
$U(T)_\varepsilon \leftarrow T$
while Z nie je prázdny **do**
  $s \leftarrow Z.Pop()$
  for $\forall$ stav t do ktorého existuje ε-krok zo stavu s **do**
    if $t \notin U(T)_\varepsilon$ **then**
      $U(T)_\varepsilon \leftarrow U(T)_\varepsilon \cup \{t\}$
      Z.Push(t)
    end if
  end for
end while

end

Vstup: NFA, $A = (K, \Sigma, \delta, q_0, F)$
Výstup: DFA $A_D = (K_D, \Sigma, \delta_D, {q_0}_D, F_D)$.
Poznámka: Pracujeme s dvoma typmi stavov: 1) neoznačené q a 2) označené $\bar{q}$.
begin
  $K_D \leftarrow U(q_0)_\varepsilon$ ▷ Ulož $U(q_0)_\varepsilon$ do K_D ako neoznačený podmnožinový stav
  while $\exists$ neoznačený (podmnožinový) stav t $\in K_D$ **do**
    označ t v K_D ako $\bar{t}$
    for $\forall a \in \Sigma$ **do**
      $u \leftarrow U(\delta(t, a))_\varepsilon$ ▷ podmnožinový stav
      if $u \notin K_D$ **then**
        $K_D \leftarrow K_D \cup \{u\}$ ▷ Neoznačené
      end if
      $\delta_D(t, a) \leftarrow u$ ▷ nová prechodová funkcia
    end for
  end while
  ${q_0}_D \leftarrow U(q_0)_\varepsilon$
  $F_D \leftarrow \{q \in K_D | F \cap q \neq \emptyset\}$
end

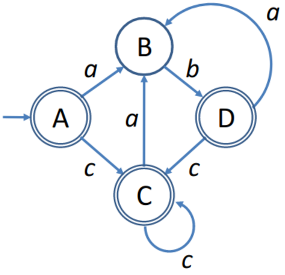


D_{states}

NFA States	DFA State	Next State a	Next State b	Next State c
{9,7,1,5,10}	A ✓	B	-	C
{2}	B ✓	-	D	-
{6,8,10,7,1,5}	C ✓	B	-	C
{4,8,7,1,5,10}	D ✓	B	-	C

Zdroj: Matthew Naylor:Lexical and Syntax Analysis of Programming Languages, University of York, 2011.

NFA States	DFA State	*Next State*		
		a	*b*	*c*
{9,7,1,5,10}	A ✓	B	-	C
{2}	B ✓	-	D	-
{6,8,10,7,1,5}	C ✓	B	-	C
{4,8,7,1,5,10}	D ✓	B	-	C



Zdroj: Matthew Naylor:Lexical and Syntax Analysis of Programming Languages, University of York, 2011.

Veta (Ekvivalencia NFA s epsilonovými krokmi a NFA bez epsilonových krokov)

K ľubovolnému nedeterministickému konečnému automatu $A = (K, \Sigma, \delta, q_0, F)$ existuje nedeterministický konečný automat A' bez epsilonových prechodov (čiže taký, že $\forall q \in K_{A'} : \delta_{A'}(q, \varepsilon) = \emptyset$), pre ktorý L(A') = L(A).

$A' = (K_{A'}, \Sigma, \delta_{A'}, q_0, F_{A'})$, kde:

$\boldsymbol{K_{A'}} \subseteq 2^K$ (stavy v tvare $\{q_1, q_2, ..., q_k\}$, kde $1 \leq i \leq k : q_i \in K$)

$\boldsymbol{\delta_{A'}(q, x)} = \bigcup_{p \in U(q)_\varepsilon} \delta(p, x)$ – všetky prechody (stavy $p \in K$), ktoré mohol spraviť pôvodný automat A, keď prečítal symbol x a bol v stave $q \in K_{A'}$.

$\boldsymbol{q_0'} = U(q_0)_\varepsilon$

$\boldsymbol{F_{A'}} = \{q \in K_{A'} | F \cap q \neq \emptyset\}$ – stavmi z $K_{A'}$ ktoré obsahujú aspoň 1 akceptujúci stav z pôvodného automatu A

Nedeterministický konečný automat

Veta (Ekvivalencia NFA a DFA)

K ľubovolnému nedeterministickému konečnému automatu $A = (K, \Sigma, \delta, q_0, F)$[a] existuje deterministický konečný automat A' pre ktorý L(A') = L(A).

[a] bez ε-krokov

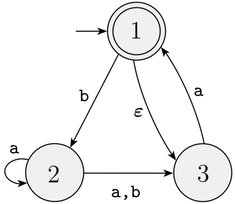
$A' = (K_{A'}, \Sigma, \delta_{A'}, q_0, F_{A'})$, kde:
$\boldsymbol{K_{A'}} \subseteq 2^K$ (stavy v tvare $\{q_1, q_2, ..., q_k\}$, kde $1 \leq i \leq k : q_i \in K$)
$\boldsymbol{\delta_{A'}(Q, x)} = \bigcup\limits_{p \in Q} \delta(p, x)$ – všetky prechody (stavy $p \in K$), ktoré mohol spraviť pôvodný automat A, keď prečítal symbol x a bol v stavoch $Q \in K_{A'}$.
$\boldsymbol{q'_0} = \{q_0\}$
$\boldsymbol{F_{A'}} = \{q \in K_{A'} | F \cap q \neq \emptyset\}$ – stavmi z $K_{A'}$ ktoré obsahujú aspoň 1 akceptujúci stav z pôvodného automatu A

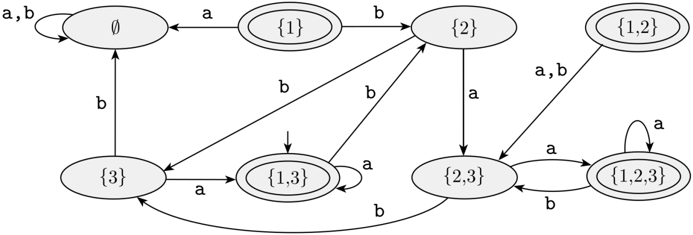
Vstup: NFA, $A = (K, \Sigma, \delta, q_0, F)$ ▷ bez epsilonových prechodov
Výstup: DFA $A_D = (K_D, \Sigma, \delta_D, q_{0D}, F_D)$.
Poznámka: Pracujeme s dvoma typmi stavov: 1) neoznačené q a 2) označené $\bar{q}$.

begin
 $K_D \leftarrow \{q_0\}$
 while $\exists$ neoznačený (podmnožinový) stav t $\in K_D$ **do**
  označ t v K_D ako $\bar{t}$
  for $\forall a \in \Sigma$ **do**
   $u \leftarrow \delta(t, a)$ ▷ podmnožinový stav
   if $u \notin K_D$ **then**
    $K_D \leftarrow K_D \cup \{u\}$ ▷ Neoznačené
   end if
   $\delta_D(t, a) \leftarrow u$ ▷ nová prechodová funkcia
  end for
 end while
 $q_{0D} \leftarrow \{q_0\}$
 $F_D \leftarrow \{q \in K_D | F \cap q \neq \emptyset\}$
end

Vlastnosti:

- exponenciálny nárast stavov
- simulácia – prehľadávanie do šírky





Zdroj: M. Sipser. Introduction to the Theory of Computation. Third Boston, 2013.

Ďakujem vám za pozornosť.

Dotazník k prednáške:
`https://forms.gle/gPpbqYUYSnGfbG9X9`