2. Konečné automaty

Formální jazyky a automaty

Jiří Balun

Obsah

Deterministický konečný automat (DFA)

Motivační příklad: emailové adresy

Příklad (část 1/2)

Chceme ověřit, zda řetězec w nad $\Sigma = \{\mathbf{.}, @, a, \ldots, z\}$ obsahuje emailovou adresu. Minule jsme si představili regulární výraz, který byl podobný tomuto (trochu jej zjednodušíme):

$$(\mathbf{.} + \boldsymbol{a} + \ldots + \boldsymbol{z})^*\mathbf{@}(\boldsymbol{a} + \ldots + \boldsymbol{z})^*\mathbf{.}(\boldsymbol{a} + \ldots + \boldsymbol{z})^*$$

Návrh algoritmu, který čte w znak po znaku, a rozhoduje odpovídající jazyk $L_@$:

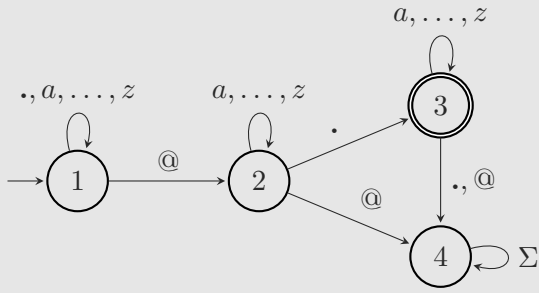
- všimněme si, že emailová adresa má několik částí (lokální část, znak @ a doménu),
- přečtení korektního podřetězce v každé části posune algoritmus do další fáze ověřování,
- řetězec w náleží do $L_@$, pokud takto ověříme všechny části emailové adresy.

Namísto psaní kódu navrhneme jednoduchý výpočetní model, který bude reprezentovat průběh výpočtu pomocí diskrétních stavů (podobně jako ve vývojových diagramech).

Motivační příklad: první návrh

Příklad (část 2/2)

Konečný automat ekvivalentní s RE $(\boldsymbol{.} + \boldsymbol{a} + \ldots + \boldsymbol{z})^*\boldsymbol{@}(\boldsymbol{a} + \ldots + \boldsymbol{z})^*\boldsymbol{.}(\boldsymbol{a} + \ldots + \boldsymbol{z})^*$, který ověřuje, zda řetězce nad $\Sigma = \{., @, a, \ldots, z\}$ obsahují emailovou adresu:



Grafická reprezentace

- uzly reprezentují stavy, ve kterých se můžeme během výpočtu nacházet
- šipka u stavu 1 (zleva) označuje počáteční stav a zdvojené stavy jsou akceptující
- hrany a jejich popisy reprezentují přechody mezi stavy – definují průběh výpočtu

Deterministický konečný automat

Intuice: automat je jednoduchý abstraktní výpočetní model (algoritmus), který realizuje rozhodovací problém náležení do regulárního jazyka.

Definice

Deterministický konečný automat (zkráceně DFA z *deterministic finite automaton*) je reprezentován uspořádanou pěticí $A = (Q, \Sigma, \delta, q_0, F)$, kde:

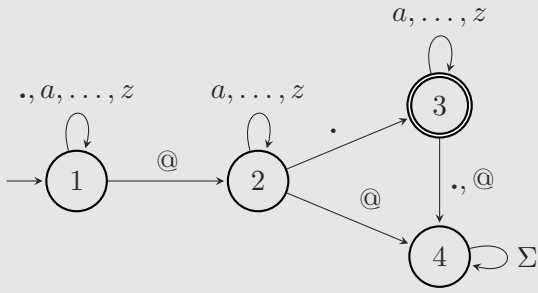
1. Q je konečná množina stavů (proto název *konečný* automat),
2. Σ je abeceda,
3. δ je přechodová funkce ve tvaru $\delta : Q \times \Sigma \to Q$, která stavu a symbolu přiřadí stav,
4. q_0 je počáteční stav (platí $q_0 \in Q$),
5. F je množina koncových/akceptujících stavů (platí $F \subseteq Q$).

- přechodová funkce δ je **úplná** (je definovaná pro každou dvojici stavu a znaku ze Σ)
  - pro řetězec nad Σ se nestane, že by automat nevěděl, jak pokračovat ve výpočtu
- přechodová funkce δ je **deterministická** (výsledkem je vždy jeden stav)
  - máme zaručeno, že automat je vždy právě v jednom stavu

Motivační příklad: popis automatu

Příklad

Chceme ověřovat zda řetězce nad $\Sigma = \{., @, a, \ldots, z\}$ obsahují emailovou adresu.



Popíšeme jednotlivé části DFA $A_1 = (Q, \Sigma, \delta, q_0, F)$:

- $Q = \{1, 2, 3, 4\}$, $\Sigma = \{., @, a, \ldots, z\}$, $q_0 = 1$ a $F = \{3\}$,
- přechodovou funkci δ lze reprezentovat jako relaci $\delta \subseteq Q \times \Sigma \times Q$:
  - $\{(1, ., 1), (1, a, 1), \ldots, (1, z, 1), (1, @, 2), (2, a, 2), \ldots, (2, z, 2), (2, ., 3), (2, @, 4), \ldots\}$

Výpočet DFA

Intuice: výpočet si lze představit jako průchod grafem automatu, v němž hrany volíme podle přechodové funkce (pro každý řetězec existuje právě jedna cesta z každého stavu).

Definice

Konfigurace je dvojice stavu a řetězce $\langle q, w \rangle \in Q \times \Sigma^*$, která popisuje aktuální stav DFA.

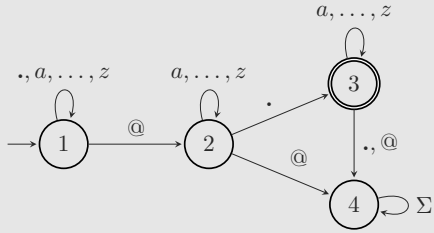
Průběh výpočtu DFA

- výpočet začíná v konfiguraci $\langle q_0, w \rangle$ s řetězcem $w = w_1 \dots w_n \in \Sigma^*$ na vstupu
- v každém kroku DFA, který je v konfiguraci $\langle q, w_i \dots w_n \rangle$, zpracuje první znak vstupu
  - přečte první symbol w_i a ze vstupu jej odebereme
  - na základě přechodové funkce se vypočítá nový stav $p = \delta(q, w_i)$
  - DFA přejde do nové konfigurace $\langle p, w_{i+1} \dots w_n \rangle$
- výpočet končí v konfiguraci s prázdným vstupem, tj. ve tvaru $\langle q, \varepsilon \rangle$
  - pokud $q \in F$ je koncový stav, pak daný DFA **přijímá** w
  - v opačném případě $q \notin F$ daný DFA **zamítá** w

Motivační příklad: výpočet DFA

Příklad

Výpočet DFA $A_1 = (\{1,2,3,4\}, \{\texttt{.}, @, a, \ldots, z\}, \delta, 1, \{3\})$, který ověřuje emailové adresy:



Mějme na vstupu řetězec $w = a@b.c$, a tedy A_1 začíná v konfiguraci $\langle 1, a@b.c\rangle$:

- $\delta(1, a) = 1$, a proto A_1 přejde přečtením symbolu a do konfigurace $\langle 1, @b.c\rangle$,
- $\delta(1, @) = 2$, a proto A_1 přejde přečtením symbolu $@$ do konfigurace $\langle 2, b.c\rangle$,
- $\delta(2, b) = 2$, a proto A_1 přejde přečtením symbolu b do konfigurace $\langle 2, .c\rangle$,
- $\delta(2, .) = 3$, a proto A_1 přejde přečtením symbolu $.$ do konfigurace $\langle 3, c\rangle$,
- $\delta(3, c) = 3$, a proto A_1 přejde přečtením symbolu c do **přijímací** konfigurace $\langle 3, \varepsilon\rangle$.

Rozšířená přechodová funkce

Intuice: Zobecníme přechodovou funkci DFA, tak aby na pracovala s řetězcem namísto jednoho symbolu.

Definice

Rozšířenou přechodovou funkci $\hat{\delta}\colon Q \times \Sigma^* \to Q$ definujeme pro dvojici stavu $q \in Q$ a řetězce $s = s_1 \dots s_n \in \Sigma^*$ následovně:

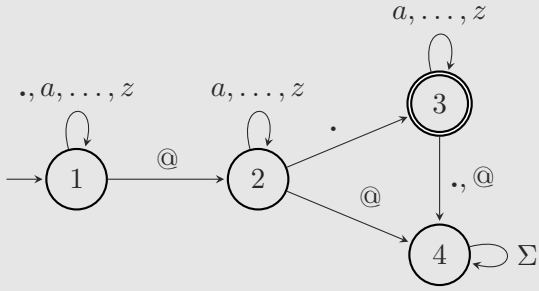
$$\hat{\delta}(q, s) = \begin{cases} q & \text{pro } s = \varepsilon \\ \hat{\delta}(\delta(q, s_1), s_2 \dots s_n) & \text{pro } s \neq \varepsilon \end{cases}$$

- jedná se pouze o syntaktický cukr, který nám usnadní zápis
- například místo $\delta(\delta(\delta(q, s_1), s_2), s_3)$ můžeme napsat $\hat{\delta}(q, s_1 s_2 s_3)$
- většinou nebudeme rozlišovat mezi δ a $\hat{\delta}$, tj. budeme psát rovnou $\delta(q, s_1 s_2 s_3)$

Motivační příklad: průběh výpočtu

Příklad

Výpočet DFA $A_1 = (\{1,2,3,4\}, \{., @, a, \ldots, z\}, \delta, 1, \{3\})$, který ověřuje emailové adresy

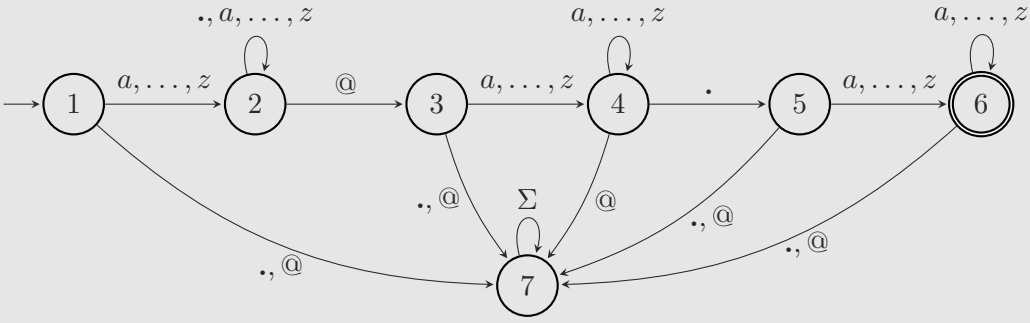


- A_1 zamítá řetězec $@b@$, protože $\delta(\delta(\delta(1, @), b), @) = \delta(\delta(2, b), @) = \delta(2, @) = 4 \notin F$
- A_1 přijímá řetězec $john.doe@upol.cz$, protože $\hat{\delta}(1, john.doe@upol.cz) = 3 \in F$
- A_1 přijímá i řetězec $@.$, protože $\hat{\delta}(1, @.) = 3 \in F$ ... to ale není emailová adresa

Motivační příklad: lepší návrh

Příklad

Nový DFA $A_2 = (\{1, \ldots, 7\}, \{., @, a, \ldots, z\}, \delta, 1, \{6\})$ ověřující emailové adresy



- nyní A_2 zamítá řetězec .@, protože $\hat{\delta}(1, .@) = 7 \notin F$
- A_2 vyžaduje aby každá část adresy (lokální i domény) byla neprázdná
- odpovídající RE: $(\boldsymbol{a} + \ldots + \boldsymbol{z})(\boldsymbol{.} + \boldsymbol{a} + \ldots + \boldsymbol{z})^*\boldsymbol{@}(\boldsymbol{a} + \ldots + \boldsymbol{z})^+\boldsymbol{.}(\boldsymbol{a} + \ldots + \boldsymbol{z})^+$

Jazyk DFA

Intuice: DFA lze chápat jako další formalismus pro definování regulárních jazyků.

Definice

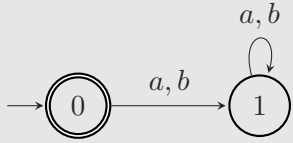
Jazyk DFA $A = (Q, \Sigma, \delta, q_0, F)$ definujeme jako $L(A) = \{w \in \Sigma^* \mid \hat{\delta}(q_0, w) \in F\}$.

- říkáme, že A **rozhoduje/rozpoznává** jazyk $L(A)$
- **regulární jazyky** odpovídají přesně jazykům, které jsou rozhodovány pomocí DFA
  - každý DFA rozhoduje nějaký regulární jazyk
  - každý regulární jazyk je rozhodován nějakým DFA
  - regulární výrazy a DFA jsou ekvivalentní – toto tvrzení zatím ponecháme bez důkazu
- implementace DFA je velice jednoduchá (zkuste si to sami)
  - verifikaci regulárních jazyků můžeme provádět na počítači

Příklad: jazyk obsahující jen prázdný řetězec

Příklad

Mějmě jazyk $L_\varepsilon = \{\varepsilon\}$ nad $\Sigma = \{a, b\}$, který obsahuje pouze prázdný řetězec.



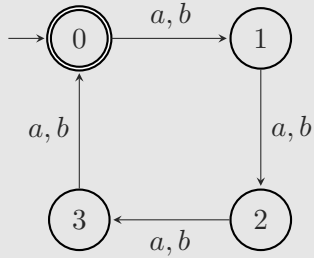
DFA, který jej rozpoznává, označíme $A_\varepsilon = (Q_\varepsilon, \{a, b\}, \delta_\varepsilon, 0, \{0\})$**:**

- A_ε zamítne každý vstupní řetězec, pokud obsahuje alespoň jeden znak,
- stav 1 je tzv. *sink state* – zamítací stav, ze kterého se nelze nijak dostat.

Příklad: počítání délky slova

Příklad

Mějme jazyk $L_{mod} = \{w \in \{a,b\}^* \mid \text{platí, že } |w| \equiv 0 \,(\text{mod } 4)\}$.



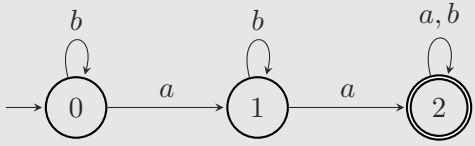
DFA, který jej rozpoznává, označíme $A_{mod} = (Q_{mod}, \{a,b\}, \delta_{mod}, 0, \{0\})$**:**

- každý stav (jeho pojmenování) odpovídá aktuální délce slova modulo 4,
- přidáním/odebráním stavů lze zobecnit pro jazyky slov délky pro obecné $n \in \mathbb{N}$.

Příklad: počítání výskytů znaku

Příklad

Mějmě funkci $\#_a : \Sigma^* \to \mathbb{N}_0$, která pro daný řetězec w vrátí počet výskytů znaků a ve w, a definujeme jazyk $L_\# = \{w \in \{a, b\}^* \mid w \text{ obsahuje alespoň 2 znaky } a, \text{ tj. } \#_a(w) \geq 2\}$.

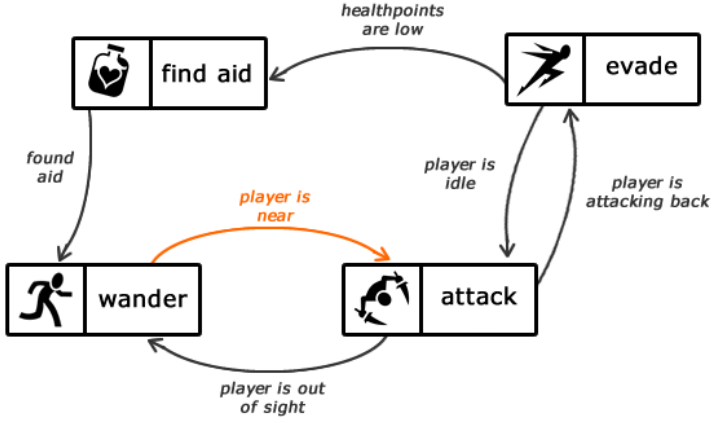


DFA, který jej rozpoznává, označíme $A_\# = (Q_\#, \{a, b\}, \delta_\#, 0, \{2\})$**:**

- každý stav (jeho pojmenování) odpovídá aktuálnímu počtu přečtených symbolů a.

Příklad: stavový systém chování NPC

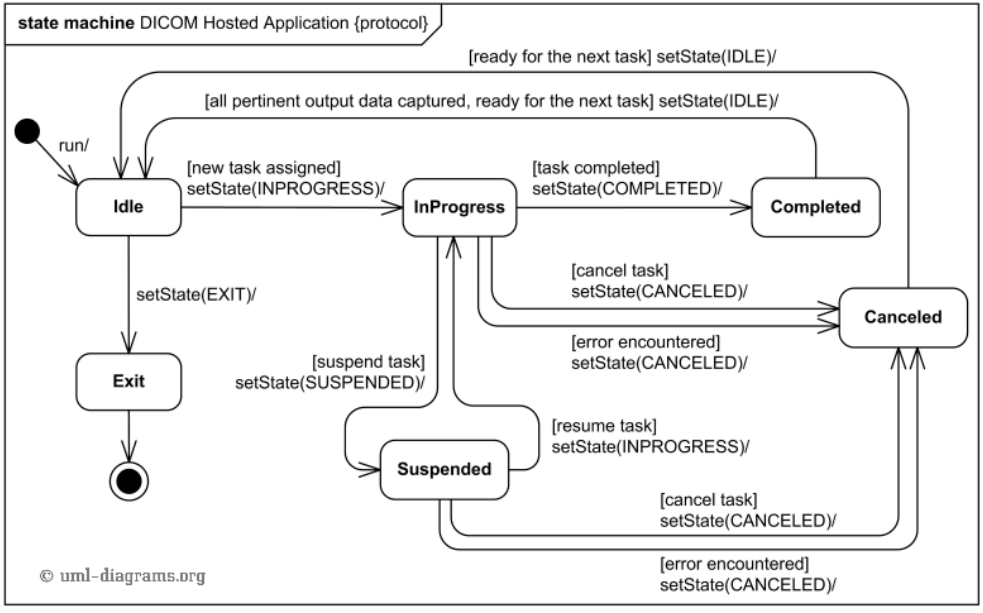
- pomocí konečných automatů lze například modelovat chování postav ve hrách
- nejde o to přímo implementovat DFA, ale umět si usnadnit návrh svého kódu



Obrázek je převzatý z popisu knihovny *Roblox State Machine*.

Příklad: stavové diagramy z UML

- při návrhu a popisu systémů se běžně používají modely podobné konečným automatům



Základní vlastnosti regulárních jazyků

Motivace studovat modely a třídy jazyků

Uzávěrové vlastnosti jazyků

- operace je uzavřená na množině M, pokud pro argumenty z M vrátí opět hodnotu z M
- aplikováním libovolného počtu uzavřených operací na jazyk nezvýšíme sílu modelu, který je potřeba k jeho rozhodování
- například třída regulárních jazyků je uzavřená na doplněk, průnik, atd...

Vlastnosti modelů

- jak modifikovat model, aniž bychom změnili jazyk, který rozhoduje?
  - příklady: odstranění nedosažitelných stavů DFA, minimalizace DFA, atd...
- zobecnění modelu:
  - dostaneme silnější model, pokud oslabíme některé podmínky v modelu DFA?
  - příklad: neúplná přechodová funkce, nedeterminismus, počet počátečních stavů, ...
- specializace modelu, tedy opak k zobecnění (toto se nás moc týkat nebude):
  - definici modelu můžeme i zesílit a vymezit tím nějakou specifickou podmnožinu
  - příklad: DFA bez cyklů má jiné vlastnosti nez obecný DFA

Neúplná přechodová funkce

- definici DFA lze zobecnit tak, že povolíme neúplnou přechodovou funkci
  - δ je praciální funkce, tj. přechod nemusí být definován pro všechny dvojice stavu a symbolu
  - pokud má automat provést nedefinovaný přechod, okamžitě daný řetězec zamítne
  - pro jednoduchost budeme DFA i nadále uvažovat s úplnou přechodovou funkcí

Věta

Ke každému DFA A s neúplnou přechodovou funkcí existuje DFA A' s úplnou přechodou funkcí takový, že $L(A) = L(A')$.

Důkaz

Zkonstruujeme A' dodefinováním chybějících přechodů v původním $A = (Q, \Sigma, \delta, q_0, F)$.

1. Do A přidáme nový nekoncový stav $q^\star$ (*sink state*),
2. pro každý nedefinovaný přechod $\delta(q, a)$ definujeme $\delta'(q, a) = q^\star$,
3. definujeme přechod $\delta'(q^\star, a) = q^\star$ pro každý znak $a \in \Sigma$,
4. získali jsme požadovaný automat $A' = (Q \cup \{q^\star\}, \Sigma, \delta \cup \delta', q_0, F)$.

Očividně platí $L(A) = L(A')$, protože upravujeme jen výpočet pro zamítnuté řetězce. □

Doplněk regulárního jazyka

Intuice: prohozením koncových a nekoncových stavů v automatu A získáme automat $\bar{A}$, který rozpoznává doplněk jazyka $L(A)$.

Věta

Nechť $L \subseteq \Sigma^*$ je regulární jazyk, pak jazyk $\bar{L} = \Sigma^* \setminus L$, který je jeho doplňkem vzhledem k univerzu Σ^*, je také regulární.

Důkaz

Jazyk L je regulární, proto existuje DFA $A = (Q, \Sigma, \delta, q_0, F)$ takový, že $L(A) = L$.
Sestrojíme nový DFA $\bar{A} = (Q, \Sigma, \delta, q_0, Q \setminus F)$ a ukážeme, že $\bar{L} = L(\bar{A}) = \Sigma^* \setminus L(A)$:

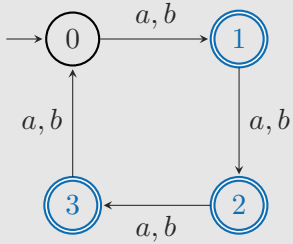
- dle definice $w \in L(\bar{A})$ právě tehdy, když $\delta(q_0, w) \in Q \setminus F$,
- to je ekvivaletní s $\delta(q_0, w) \notin F$, a tedy platí $w \notin L(A)$. □

- **Důsledek:** třída regulárních jazyků je uzavřená na doplněk
- všimněte si, že důkaz nefunguje pokud by DFA A měl neúplnou přechodovou funkci

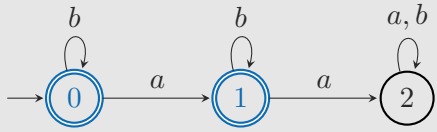
Příklad: doplněk

Příklad

DFA rozpoznávající jazyky $\bar{L}_{mod}$ a $\bar{L}_{\#}$:



a) DFA $\bar{A}_{mod}$



b) DFA $\bar{A}_{\#}$

a) $\bar{A}_{mod} = (Q_{mod}, \{a, b\}, \delta_{mod}, 0, \{1, 2, 3\})$
- rozpoznává jazyk $\bar{L}_{mod} = \{w \in \{a, b\}^* \mid \text{platí, že } |w| \not\equiv 0 \pmod 4\}$

b) $\bar{A}_{\#} = (Q_{\#}, \{a, b\}, \delta_{\#}, 0, \{0, 1\})$
- rozpoznává jazyk $\bar{L}_{\#} = \{w \in \{a, b\}^* \mid w \text{ obsahuje nanejvýš 1 znak } a, \text{ tj. } \#_a(w) \leq 1\}$

Nedosažitelné stavy

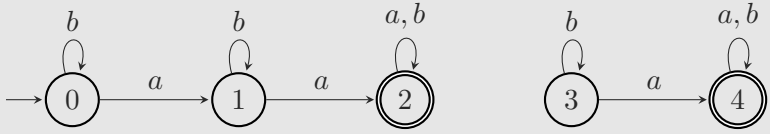
Intuice: stav automatu je dosažitelný, pokud k němu vede cesta v grafu automatu, která začíná v počátečním stavu.

Definice

Mějme DFA $A = (Q, \Sigma, \delta, q_0, F)$, pak stav $q \in Q$ je **dosažitelný** pokud existuje $w \in \Sigma^*$ takové, že $\delta(q_0, w) = q$. V opačném případě je q nedosažitelný.

Příklad

Mějmě DFA $A'_{\#}$, který má oproti $A_{\#}$ navíc dva nedosažitelné stavy 3 a 4.



Odstranění nedosažitelných stavů

Intuice: nedosažitelné stavy nijak neovlivňují jazyk automatu, a proto je lze odstranit.

Věta

Ke každému DFA $A = (Q, \Sigma, \delta, q_0, F)$ existuje DFA A^{acc} takový, že všechny stavy A^{acc} jsou dosažitelné a zároveň platí $L(A) = L(A^{acc})$.

Důkaz

Zkonstruujeme automat A^{acc} tak, že "zmenšíme" A:

1. Množinu dosažitelných stavů v A definujeme jako $S = \{q \mid \exists w \in \Sigma^* \colon \delta(q_0, w) = q\}$,
2. poté definujeme $A^{acc} = (S, \Sigma, \delta^{acc}, q_0, F \cap S)$, kde $\delta^{acc} = \delta \cap (S \times \Sigma \times S)$,
3. evidentně platí $L(A^{acc}) \subseteq L(A)$, jelikož A^{acc} vznikl odebráním stavů a přechodů z A,
4. $L(A) \subseteq L(A^{acc})$ platí, protože pro každé $w \in L(A)$ je $\delta(q_0, w) = q_f \in F$, kde q_f je dosažitelný stav, a tedy $q_f \in S$. Z toho dostaneme $\delta^{acc}(q_0, w) \in F \cap S$. □

Test na neprázdnost jazyka

Intuice: jazyk jakéhokoli automatu je neprázdný, pokud je dosažitelný alespoň jeden jeho koncový stav.

Věta

Mějme DFA $A = (Q, \Sigma, \delta, q_0, F)$, pak $L(A) \neq \emptyset$ právě tehdy, když existuje řetězec $w \in \Sigma^*$ takový, že platí $\delta(q_0, w) \in F$.

Důkaz

Stačí vypočítat A^{acc} a ověřit, zda je jeho množina koncových stavů F^{acc} prázdná:

- pokud $F^{acc} \neq \emptyset$, pak libovolný stav $q_f \in F^{acc}$ je dosažitelný, a tedy dle definice existuje řetězec $w \in \Sigma^*$ takový, že $\delta^{acc}(q_0, w) \in F^{acc}$,
- v opačném případě $F^{acc} = \emptyset$, a tedy platí $L(A^{acc}) = L(A) = \emptyset$. □

- výpočet A^{acc} lze použít jako algoritmus na ověření neprázdnosti jazyka DFA
  - toho využijeme později k rozhodování inkluze a rovnosti regulárních jazyků
  - A^{acc} lze vypočítat v polynomiálním čase pomocí algoritmů na průchod grafu