Úvod do automatů

Martin Mareš

Katedra aplikované matematiky
Matematicko-fyzikální fakulta
Univerzita Karlova, Praha

Toto je stručný úvod do teorie formálních jazyků, automatů a gramatik. Vznikl jako studijní text k předmětu Algoritmy a automaty pro učitele na MFF UK, ale může se hodit i dalším zájemcům o teoretickou informatiku. Text navazuje na Průvodce labyrintem algoritmů.⟨1⟩

Milý čtenáři, buď varován, že se jedná o pracovní verzi, která jistě není dokonalá. Najdeš-li libovolnou chybu (což zahrnuje i těžko srozumitelné pasáže), dej prosím vědět autorovi na adrese *mj@ucw.cz*. Díky!

⟨1⟩ Viz *http://pruvodce.ucw.cz/*.

1 Regulární jazyky

1.1 Definice a značení

- *Abeceda* Σ je nějaká konečná množina, jejím prvkům budeme říkat *znaky* (někdy též *písmena*).
- Σ^* je množina všech *slov* neboli *řetězců* nad abecedou Σ, což jsou konečné posloupnosti znaků ze Σ.
- *Slova* budeme značit malými písmenky řecké abecedy α, β, ...
- *Znaky* abecedy označíme malými písmeny latinky x, y, ... Konkrétní znaky budeme psát `psacím strojem.` Znak budeme používat i ve smyslu jednoznakového řetězce.
- *Délka slova* $|\alpha|$ udává, kolika znaky je slovo tvořeno.
- *Prázdné slovo* značíme písmenem ε, je to jediné slovo délky 0.
- *Zřetězení* $\alpha\beta$ vznikne zapsáním slov α a β za sebe. Platí $|\alpha\beta| = |\alpha|+|\beta|$, $\alpha\varepsilon = \varepsilon\alpha = \alpha$.
- *Mocnina* řetězce α^k pro $k \in \mathbb{N}$[1] vznikne zřetězením k kopií řetězce α. Tedy $\alpha^0 = \varepsilon$, $\alpha^{k+1} = \alpha^k\alpha$.
- $\alpha[k]$ je k-tý znak slova α, indexujeme od 0 do $|\alpha| - 1$.
- $\alpha[k : \ell]$ je *podslovo* začínající k-tým znakem a končící těsně před ℓ-tým. Tedy $\alpha[k : \ell] = \alpha[k]\alpha[k + 1] \dots \alpha[\ell - 1]$. Pokud $k \geq \ell$, je podslovo prázdné. Pokud některou z mezí vynecháme, míní se $k = 0$ nebo $\ell = |\alpha|$.
- $\alpha[: \ell]$ je *prefix* (předpona) tvořený prvními ℓ znaky řetězce.
- $\alpha[k :]$ je *suffix* (přípona) od k-tého znaku do konce řetězce.
- $\alpha[:] = \alpha$.
- $\#x \in \alpha$ je počet výskytů znaku x v řetězci α. Je to tedy počet všech i takových, že $\alpha[i] = x$.
- *Otočení* α^{R} je slovo α „čtené pozpátku“. Tedy pro $\alpha = x_1 \dots x_n$ máme $\alpha^{\mathrm{R}} = x_n \dots x_1$.
- *Jazyk* říkáme jakékoliv množině slov. Jazyky jsou tedy podmnožiny Σ^*.

[1] V tomto textu považujeme 0 za přirozené číslo.

Pozorování: Jazyky jsou podobné *rozhodovacím problémům,* které definujeme[2] jako funkce ze Σ^* do $\{0,1\}$. Rozhodovacímu problému P můžeme přiřadit jazyk $L_P = \{\alpha \in \Sigma^* \mid P(\alpha) = 1\}$. Naopak jazyku $L \subseteq \Sigma^*$ přiřadíme problém P_L takový, že $P_L(\alpha) = 1 \Leftrightarrow \alpha \in L$.[3]

1.2 Konečné automaty

Definice: *Deterministický konečný automat* (DFA) je uspořádaná pětice $(Q, \Sigma, \delta, q_0, F)$, kde:

- Q je konečná neprázdná množina *stavů,*
- Σ je konečná neprázdná množina znaků – *abeceda,*
- $\delta : Q \times \Sigma \to Q$ je *přechodová funkce,*[4]
- $q_0 \in Q$ je *počáteční stav,*
- $F \subseteq Q$ je množina *přijímacích (neboli koncových) stavů.*

Ve zbytku tohoto oddílu budeme říkat prostě *automat.*

Definice: *Výpočet* automatu pro vstup $\alpha \in \Sigma^*$ je posloupnosti stavů $s_0, s_1, \ldots, s_{|\alpha|}$ taková, že:

- $s_0 = q_0$,
- $s_{i+1} = \delta(s_i, \alpha[i])$.

Poznámka: Automat si také můžeme představit jako orientovaný graf. Jeho vrcholy odpovídají stavům, hrany přechodům mezi stavy. Každá hrana je označena jedním znakem abecedy, přičemž platí, že z každého vrcholu vede pro každý znak abecedy právě jedna hrana. Výpočet pro vstup α je pak sled[5] začínající v počátečním stavu, jehož označení hran dávají po řadě slovo α. Tento sled je slovem α jednoznačně určen.

Definice: *Rozšířená přechodová funkce* $\delta^* : Q \times \Sigma^* \to Q$ je definována takto:

---

[2] Viz Průvodce, kapitola Těžké problémy.

[3] Bystrý čtenář v tom poznává charakteristickou funkci podmnožiny.

[4] Zde se omlouváme za nekonsistenci: malá řecká písmena jsme si vyhradili pro slova, ale toto značení pro přechodovou funkci je natolik zažité, že ho je marno měnit.

[5] Připomínáme, že *sled* v grafu je posloupnost na sebe navazujících vrcholů a hran. Od cesty se liší tím, že se v něm mohou vrcholy i hrany opakovat.

- $\delta^*(q, \varepsilon) = q$ pro všechna $q \in Q$,
- $\delta^*(q, \alpha x) = \delta(\delta^*(q, \alpha), x)$ pro všechna $q \in Q$, $\alpha \in \Sigma^*$, $x \in \Sigma$.

Pozorování: Pro výpočet automatu platí $s_i = \delta^*(q_0, \alpha[:i])$.

Definice: Slovo α je *přijímáno automatem*, pokud příslušný výpočet skončí v přijímacím stavu, tedy $\delta^*(q_0, \alpha) \in F$.

Definice: *Jazyk přijímaný (rozpoznávaný) automatem* je množina všech přijímaných slov, tedy $\{\alpha \in \Sigma^* \mid \delta^*(q_0, \alpha) \in F\}$. Pro automat A tento jazyk označíme $L(A)$.

Definice: Jazyk L je *regulární,* pokud je rozpoznávaný nějakým konečným automatem. Tedy existuje-li konečný automat A takový, že $L = L(A)$.

Notace: V obrázcích značíme počáteční stav šipkou z okolního prostoru do stavu, koncové stavy mají tučný okraj.

Příklad (počítání jedniček): Uvažme jazyk $L_3 = \{\alpha \in \{\mathtt{0},\mathtt{1}\} \mid (\#\mathtt{1} \in \alpha) \bmod 3 = 0\}$, tedy jazyk slov, která obsahují počet jedniček dělitelný třemi. Tento jazyk je regulární. O tom se snadno přesvědčíme sestrojením automatu: bude mít stavy $\{0, 1, 2\}$ odpovídající možným zbytkům po dělení počtu zatím přečtených jedniček třemi. Stav 0 bude jak počáteční, tak jediný přijímající.

Příklad (konečné jazyky): Ukážeme, že libovolný konečný jazyk L je regulární. Automat bude mít tvar písmenkového stromu (trie) pro množinu L. Stavy tedy budou prefixy všech slov z L a navíc jeden univerzální zamítací stav z. Přechodová funkce $\delta(\alpha, x)$ pro prefix α a znak x povede do prefixu αx, pokud existuje, jinak do z. Ze z povedou všechny přechody zase do z. Počátečním stavem bude prázdný prefix ε, přijímacími stavy všechna slova z L.

Příklad (vyhledávací automaty): Vyhledávací automat[6] typu Knuth-Morris-Pratt jde upravit na konečný automat. Množinu stavů zachováme, první stav automatu (prázdný prefix) se stane počátečním, poslední stav (prefix rovný jehle) jediným přijímajícím. Přechodovou funkci definujeme pomocí funkce na jeden krok automatu, která se sama postará o použití dopředných a zpětných hran. Jazyk rozpoznávaný tímto automatem bude tvořen všemi slovy, která končí jehlou. Podobně můžeme upravit automat typu Aho-Corasicková.

Příklad (neregulární jazyk): Jazyk $L_{01} = \{\mathtt{0}^n\mathtt{1}^n \mid n \in \mathbb{N}\}$ není regulární. Dokážeme to sporem. Předpokládejme, že existuje automat rozpoznávající tento jazyk. Označme n počet jeho stavů. Automat spustíme na vstupech $\mathtt{0}^k$ pro $k = 0, \ldots, n$ a nazveme $s_0, \ldots, s_n$ stavy, ve kterých jednotlivé výpočty skončí. Bude tedy $s_i = \delta^*(q_0, \mathtt{0}^i)$.

[6] Viz Průvodce, kapitola Vyhledávání v textu.

Posloupnost $s_0, \ldots, s_n$ má $n+1$ prvků, ale ty nabývají nejvýše n hodnot. Pro to se některá nutně zopakuje: $s_i = s_j$ pro $0 \leq i < j \leq n$. Výpočty pro slova $\mathtt{0}^i$ a $\mathtt{0}^j$ tedy oba shodně dojdou do nějakého stavu s. Pokud za tato dvě slova přidáme libovolný suffix β, musí tedy pokračovat shodně do $\delta^*(s, \beta)$.

Takže slova $\mathtt{0}^i\mathtt{1}^i$ a $\mathtt{0}^j\mathtt{1}^i$ buďto automat obě přijme, nebo obě zamítne. Jenže první do jazyka L_{01} patří, zatímco druhé nikoliv. To je spor s předpokladem, že automat rozpoznává jazyk L_{01}.

Iterační lemma

Obrat s opakováním stavů se hodí v důkazu následujícího lemmatu:

Lemma (iterační; angl. pumping lemma): Mějme regulární jazyk L. Potom existuje číslo $n \in \mathbb{N}$ takové, že každé slovo $\omega \in \Sigma^*$ délky alespoň n můžeme rozložit na $\omega = \alpha\beta\gamma$, přičemž:

1. $\beta \neq \varepsilon$ ◁ *prostřední část není prázdná*
2. $|\alpha\beta| \leq n$ ◁ *první a druhá část jsou krátké*
3. Slova $\alpha\beta^t\gamma$ pro $t \geq 0$ leží všechna v L. ◁ *druhou část lze libovolně opakovat*

Důkaz: Jelikož jazyk L je regulární, existuje nějaký automat A rozpoznávající L. Za n zvolíme počet stavů tohoto automatu.

Uvažme nyní nějaké slovo $\omega \in L$ délky $m \geq n$. Označíme $s_0, \ldots, s_m$ výpočet automatu pro toto slovo, tedy $s_i = \delta^*(q_0, \omega[:i])$. Tato posloupnost má délku větší než n, ale vyskytuje se v ní nejvýše n různých stavů. Proto se nějaký stav musí opakovat: bude $s_i = s_j = s$ pro nějaké $i < j$ a stav s. Navíc k opakování nutně dojde v prvních $n+1$ prvcích, takže $j \leq n$.

Nyní zvolíme $\alpha = \omega[:i]$, $\beta = \omega[i:j]$, $\gamma = \omega[j:]$. Jelikož $\delta^*(q_0, \alpha) = s$ a $\delta^*(s, \beta) = s$, musí být $\delta^*(q_0, \alpha\beta^t) = s$ pro každé t. Proto jsou všechny stavy $\delta^*(q_0, \alpha\beta^t\gamma)$ stejné a jelikož ten pro $t = 1$ je přijímající, musí být pro všechna t přijímající. □

Příklad: Neregularitu jazyka L_{01} z předchozího příkladu můžeme snadno dokázat pumpováním. Kdyby byl regulární, uvažme slovo $\omega = \mathtt{0}^n\mathtt{1}^n$, kde n je konstanta z lemmatu. Podle lemmatu existuje rozklad $\omega = \alpha\beta\gamma$. Jelikož α a β mají dohromady nanejvýš n znaků, skládají se jenom z nul. Přidání další kopie β (nebo odstranění) by mělo vytvořit jiné slovo jazyka. Jenže přidání β zvýší počet nul, ale zachová počet jedniček, takže slovo v jazyku ležet nemůže. To je spor.

Příklad (prvočísla): Jazyk $L_P = \{0^p \mid p \text{ je prvočíslo}\}$ také nemůže být regulární. Kdyby byl, uvažme konstantu n z lemmatu a zvolme libovolné prvočíslo $p \geq n+2$. Slovo $\mathtt{0}^p \in L_P$

tedy můžeme rozložit na části $\alpha = \mathtt{0}^i$, $\beta = \mathtt{0}^j$, $\gamma = 0^k$, přičemž $i + j + k = p$, $j > 0$ a $i + j \leq n$.

Všechna slova $\mathtt{0}^i\mathtt{0}^{jt}\mathtt{0}^k$ pro $t \geq 0$ mají také ležet v L_P, takže $i + jt + k$ musí být pro všechna t prvočíslo. Jenže zvolíme-li $t = i + k$, je $i + jt + k = (i + k)(1 + t)$. Jelikož $i + k$ i $1 + t$ jsou větší než 1 (to první díky tomu, že $k \geq 2$), nemůže se jednat o prvočíslo. Došli jsme ke sporu.

Součin automatů

Už jsme zjistili, že otázka, zda počet jedniček ve slově je dělitelný třemi, je regulární. Podobně je regulární otázka, zda je počet nul sudý. Co kdybychom chtěli rozpoznávat slova, která splňují obě podmínky současně? K tomu se hodí představa, že oba automaty spustíme paralelně a slovo přijmeme v případě, že ho oba přijaly. To můžeme formálně popsat následující konstrukcí.

Definice: Mějme dva automaty se společnou abecedou Σ: $A_1 = (Q_1, \Sigma, \delta_1, q_{01}, F_1)$ a $A_2 = (Q_2, \Sigma, \delta_2, q_{02}, F_2)$. Jejich *součin* $A_1 \times A_2$ je automat $A = (Q, \Sigma, \delta, q_0, F)$ definovaný takto:

- $Q = Q_1 \times Q_2$, ◁ *ve stavu si pamatujeme si stav obou automatů*
- $\delta((s_1, s_2), x) = (\delta_1(s_1, x), \delta_2(s_2, x))$, ◁ *simulujeme jeden krok každého automatu*
- $q_0 = (q_{01}, q_{02})$, ◁ *oba automaty začínají ve svých počátečních stavech*
- $F = F_1 \times F_2$. ◁ *přijmeme, pokud oba přijaly*

Snadno nahlédneme, že automat A přijme právě ta slova, která jsou přijata jak automatem A_1, tak A_2. Proto $L(A) = L(A_1) \cap L(A_2)$.

Důsledek: Průnik dvou regulárních jazyků je zase regulární jazyk.

Cvičení

1. *Pro tento a následující jazyky rozhodněte, zda jsou regulární – kladnou odpověď můžete zdůvodnit sestrojením automatu, zápornou třeba pomocí iteračního lemmatu.* Jazyk $\{x\alpha x \mid x \in \Sigma, \alpha \in \Sigma^*\}$ pro abecedu $\Sigma = \{\mathtt{a}, \mathtt{b}\}$. Tedy jazyk slov délky aspoň 2, jejichž první a poslední znak je stejný.

2. Jazyk všech neklesajících posloupností číslic `0` ... `3`.

3. Jazyk dvojkových zápisů přirozených čísel dělitelných 5.

4. Jazyk slov nad abecedou $\{\mathtt{a}, \mathtt{b}, \mathtt{r}\}$, která začínají na některý z řetězců `abba`, `ara`, `bar`.

5. Jazyk slov nad abecedou $\{\mathtt{a}, \mathtt{b}, \mathtt{r}\}$, která obsahují některý z podřetězců `ara`, `bar`, `arab`.

6. Jazyk $\{\alpha\alpha \mid \alpha \in \{\mathtt{0}, \mathtt{1}\}^*\}$.

7. Jazyk $\{\mathtt{0}^{n^2} \mid n \in \mathbb{N}\}$.

8. Jazyk $\{\mathtt{0}^{2^n} \mid n \in \mathbb{N}\}$.

9. Dokažte, že doplněk regulárního jazyka je zase regulární. Tedy pro každý regulární jazyk $L \subseteq \Sigma^*$ je $\Sigma^* \setminus L$ také regulární.

10. Dokažte, že sjednocení regulárních jazyků je regulární.

11. Dokažte, že jazyk $\{\alpha \in \{\mathtt{0}, \mathtt{1}\}^* \mid (\#\mathtt{0} \in \alpha) = (\#\mathtt{1} \in \alpha)\}$ není regulární. Využijte toho, že $\{\mathtt{0}^n\mathtt{1}^n \mid n \in \mathbb{N}\}$ není regulární, a že průnik dvou regulárních jazyků je regulární.

12. Definujme *uzávorkování* jako posloupnost závorek `(` a `)`, které lze spárovat tak, aby se páry nekřížily a v každém páru byla `(` před `)`. Ukažte, že jazyk všech uzávorkování není regulární. Může se hodit postup z předchozího cvičení.

13. Co kdybychom automatu dovolili mít nekonečně mnoho stavů? Jak by se změnilo, které jazyky můžeme rozpoznávat?

14. Vymyslete algoritmus, který pro daný automat A rozhodne, zda jazyk $L(A)$ je neprázdný.

15.* Vymyslete algoritmus, který pro daný automat A rozhodne, zda jazyk $L(A)$ je konečný.

1.3 Nedeterministické automaty

Uvažme následující příklad. Chceme popsat jazyk všech slov nad abecedou $\{\mathtt{0}, \mathtt{1}\}$, která končí na `01`. Taková slova můžeme složit ze dvou částí α a β, kde α je libovolný řetězec nul a jedniček a $\beta = \mathtt{01}$. Obě tyto části umíme rozpoznat konečnými automaty A a B, takže by bylo pěkné umět tyto automaty složit dohromady a získat tak automat pro požadovaný jazyk.

Nabízí se ztotožnit přijímající stav automatu A s počátečním stavem automatu B a dovolit tak výpočtu přejít z A do B. Jenže vzniklý „slepený“ stav má dva přechody pro znak `0`, což naše definice konečného automatu nedovoluje. Co kdybychom definici zobecnili, aby to bylo možné, a během výpočtu si pak z možných přechodů jeden vybrali? To vede k myšlence nedeterminismu.

Definice: *Nedeterministický konečný automat* (NFA) je uspořádaná pětice $(Q, \Sigma, \delta, Q_0, F)$, kde:

- Q je konečná neprázdná množina *stavů,*
- Σ je konečná neprázdná množina znaků – *abeceda,*
- $\delta : Q \times \Sigma \to 2^Q$ je *přechodová funkce,* která každé dvojici (*stav*, *znak*) přiřadí množinu všech stavů, do kterých je možné přejít,
- $Q_0 \subseteq Q$ je množina *počátečních stavů,*
- $F \subseteq Q$ je množina *přijímacích stavů.*

Z jednoho stavu tedy může vést více přechodů označených stejným znakem abecedy, ale také nemusí vést žádný.

Výpočet automatu opět odpovídá nějakému sledu v grafu, který začíná některým z počátečních stavů a jeho hrany jsou označeny znaky vstupního slova. Oproti deterministickému automatu ale tento sled není jednoznačně určen a nemusí ani existovat. Totéž můžeme popsat jako posloupnost stavů.

Definice: *Výpočet* nedeterministického automatu pro vstup $\alpha \in \Sigma^*$ je posloupnosti stavů $s_0, s_1, \ldots, s_{|\alpha|}$ taková, že:

- $s_0 \in Q_0$,
- $s_{i+1} \in \delta(s_i, \alpha[i])$.

Definice: Slovo α je *přijímáno automatem*, pokud existuje alespoň jeden výpočet se vstupem α, který končí v jednom z přijímajících stavů.

Výpočtů pro jedno slovo může být exponenciálně mnoho. Ale všimneme si, že pokud pro nějaký prefix vstupu dva výpočty skončí ve stejném stavu, mají stejnou množinu možných pokračování. Proto je nemusíme rozlišovat – stačí pro každý prefix vstupu určit množinu stavů, v nichž se může automat nacházet. To popíšeme pomocí rozšířené přechodové funkce.

Definice: *Rozšířená přechodová funkce* $\delta^* : 2^Q \times \Sigma^* \to 2^Q$ je definována takto:

- $\delta^*(S, \varepsilon) = Q$ pro všechny $S \subseteq Q$,
- $\delta^*(S, \alpha x) = \bigcup_{s \in \delta^*(S,\alpha)} \delta(s, x)$ pro všechny $S \subseteq Q$, $\alpha \in \Sigma^*$, $x \in \Sigma$.

Poznámka: Slovo α je přijato automatem právě tehdy, když $\delta^*(Q_0, \alpha) \cap F \neq \emptyset$.

Příklad ze začátku oddílu ukazuje, že někdy je pohodlnější sestrojit nedeterministický automat. Nyní ukážeme, že nedeterminismu se je vždy možné zbavit:

Věta: Ke každému NFA A existuje DFA A' takový, že $L(A') = L(A)$.

Důkaz: Budeme simulovat rozšířenou přechodovou funkci automatu A automatem A'. Stavy A' tedy budou odpovídat množinám stavů automatu A a do přechodů mezi nimi zakódujeme indukční krok z definice rozšířené přechodové funkce. Nyní precizně.

Mějme NFA $A = (Q, \Sigma, \delta, Q_0, F)$. Sestrojíme DFA $A' = (Q', \Sigma, \delta', q_0', F')$, přičemž:

- $Q' = 2^Q$,
- $\delta'(S, x) = \bigcup_{s \in S} \delta(s, x)$,
- $q_0' = Q_0$,
- $F' = \{S \in Q' \mid S \cap F \neq \emptyset\}$.

Nyní si všimneme, že výpočet automatu A' pro vstup α skončí ve stavu, který je roven $\delta^*(Q_0, \alpha)$. Tento stav je přijímající právě tehdy, když automat A přijme slovo α. □

Důsledek: Jazyk je rozpoznatelný nedeterministickým automatem právě tehdy, je-li regulární.

Příklad: Větu vyzkoušíme na NFA ze začátku kapitoly. Stavy DFA budou

$$Q' = \{\emptyset, \{a\}, \{b\}, \{c\}, \{a, b\}, \{a, c\}, \{b, c\}, \{a, b, c\}\},$$

přičemž stav $\{a\}$ je počáteční a stavy $\{c\}$, $\{a, c\}$, $\{b, c\}$ a $\{a, b, c\}$ přijímající. Přechodová funkce bude vypadat následovně:

S	$\delta(S, \mathtt{0})$	$\delta(S, \mathtt{1})$
$\emptyset$	$\emptyset$	$\emptyset$
$\{a\}$	$\{a, b\}$	$\{a\}$
$\{b\}$	$\emptyset$	$\{c\}$
$\{c\}$	$\emptyset$	$\emptyset$
$\{a, b\}$	$\{a, b\}$	$\{a, c\}$
$\{a, c\}$	$\{a, b\}$	$\{a\}$
$\{b, c\}$	$\emptyset$	$\{c\}$
$\{a, b, c\}$	$\{a, b\}$	$\{a, c\}$

Přitom pouze stavy $\{a\}$, $\{a,b\}$ a $\{a,c\}$ budou dosažitelné z počátečního stavu, takže všechny ostatní stavy můžeme vynechat.

Sestrojenému automatu můžeme rozumět tak, že aktuální stav obsahuje a vždy, b jen tehdy, končí-li vstup na `0`, a c jen tehdy, končí-li vstup na `01`.

Lambda-přechody

Zapojení dvou automatů „sériově“ ztotožněním stavů má jeden potenciální háček: pokud z přijímajícího stavu prvního automatu vedly nějaké přechody, může výpočet přejít z druhého automatu zpátky do prvního. Lepší by bylo zavést z konce prvního automatu do začátku druhého nějaký speciální přechod, který nepoužije žádný znak ze vstupu. Takovým přechodům se říká λ-přechody a můžeme o ně definici NFA rozšířit.

Definice: *Nedeterministický konečný automat s λ-přechody (λ-NFA)* je definován stejně jako klasický NFA, jen přechodovou funkci rozšíříme na $\delta : Q \times (\Sigma \cup \{\lambda\}) \to 2^Q$.

Výpočet automatu bude opět sled v grafu, jehož označení hran dají po vynechání všech λ vstupní slovo. Pozor na to, že je-li v grafu cyklus z λ-hran, může pro jeden vstup existovat nekonečně mnoho výpočtů. Popis rozšířenou přechodovou funkcí nadále funguje podobně a dá se zavést pomocí λ-uzávěrů:

Definice: *λ-uzávěr stavu $s \in Q$* značíme $U_\lambda(s)$ a je to množina všech stavů, do kterých se dá ze stavu s dostat po λ-hranách. Uzávěr rozšíříme na množiny stavů: $U_\lambda(S) = \bigcup_{s \in S} U_\lambda(s)$.

Pozorování: Stav je vždy dosažitelný sám ze sebe, takže vždy platí $s \in U_\lambda(s)$ a $S \subseteq U_\lambda(S)$.

Definice: *Rozšířená přechodová funkce pro λ-NFA* $\delta^* : 2^Q \times \Sigma \to 2^Q$ je definována takto:

- $\delta^*(S, \varepsilon) = U_\lambda(Q)$,
- $\delta^*(S, \alpha x) = U_\lambda\left(\bigcup_{s \in \delta^*(S,\alpha)} \delta(s, x)\right)$.

Jinými slovy na samém začátku výpočtu a po průchodu každou obyčejnou hranou jsme přidali průchod po libovolně mnoha λ-hranách.

Přijímání slova a jazyk rozpoznávaný automatem definujeme stejně jako pro NFA. Důležité je, že λ-přechody je možné eliminovat a získat tak obyčejný NFA.

Věta: Pro každý λ-NFA A existuje NFA A' takový, že $L(A') = L(A)$.

Důkaz: Množinu stavů ponecháme. Množinu počátečních stavů nahradíme jejím λ-uzávěrem. Přechodovou funkci také zkombinujeme s λ-uzávěrem, tedy $\delta'(S, x) = U_\lambda(\delta(S, x))$. Koncové stavy ponecháme.

Rozšířené přechodové funkce obou automatů se budou rovnat, takže se automaty shodnou na přijetí každého slova. □

Důsledek: Jazyk je rozpoznatelný λ-NFA právě tehdy, je-li regulární.

Lemma: Každý λ-NFA je možné (při zachování rozpoznávaného jazyka) upravit tak, aby měl jediný počáteční a jediný přijímající stav.

Důkaz: Přidáme nový počáteční stav, z nějž povedou λ-přechody do původních počátečních stavů. Také přidáme nový přijímající stav, do kterého zavedeme λ-přechody z původních přijímajících stavů. □

Algoritmické otázky

Jak těžké je zjistit, zda slovo patří do regulárního jazyka? Jak to závisí na délce slova n a počtu stavů automatu p? A jak na druhu automatu?

- DFA můžeme odsimulovat v čase $\mathcal{O}(n)$.
- U NFA můžeme simulovat rozšířenou přechodovou funkci v čase $\mathcal{O}(p^2)$ na krok, celkem tedy $\mathcal{O}(p^2 n)$. Nebo můžeme NFA převést podmnožinovou konstrukcí na DFA – to potrvá $\mathcal{O}(2^p \cdot p^2)$, ale pak už vstup zpracujeme v čase $\mathcal{O}(n)$.
- λ-NFA převedeme v čase $\mathcal{O}(p^2)$ na klasický NFA.

Cvičení

1. U DFA platilo, že prohodíme-li přijímající a nepřijímající stavy (tedy nahradíme F za $Q \setminus F$), dostaneme automat přijímající doplněk původního jazyka. Co se stane, když totéž provedeme u NFA?

2.* Podmnožinová konstrukce produkuje automaty s exponenciálně mnoha stavy vzhledem k počtu stavů původního automatu. I když často budou některé z nich nedosažitelné, nemusí tomu tak být vždy. Sestrojte pro každé t jazyk, který lze rozpoznat pomocí NFA s $\mathcal{O}(t)$ stavy, ale každý DFA, který ho rozpoznává, má aspoň 2^t stavů.

1.4 Regulární výrazy

Hlavní motivací pro zavádění NFA byla touha po vytváření komplikovaných regulárních jazyků z jednodušších. Ukážeme, že s λ-NFA je možné sestrojit praktickou „stavebnici regulárních jazyků“. Dokonce pak zjistíme, že z ní jdou sestavit úplně všechny regulární jazyky.

Operace s jazyky

Definice: Pro jazyky X a Y nad abecedou Σ definujeme:

- *sjednocení* $X \cup Y$ a *průnik* $X \cap Y$ jako běžné množinové operace
- *doplněk* $\overline{X} = \Sigma^* \setminus X$
- *zřetězení (konkatenaci)* $X \cdot Y = \{\alpha\beta \mid \alpha \in X \wedge \beta \in Y\}$, často tečku vynecháváme a píšeme prostě XY.
- *mocninu* X^k: $X^0 = \{\varepsilon\}$, $X^{k+1} = X^k \cdot X$. (Zjevně $X^1 = X$, $X^2 = XX$, $X^3 = XXX$, přičemž uzávorkování není třeba určit, neboť zřetězení je asociativní.)
- *iteraci* $X^* = \bigcup_{n \geq 0} X^n$
- *pozitivní iteraci* $X^+ = \bigcup_{n \geq 1} X^n$. (Platí tedy $X^* = \{\varepsilon\} \cup X^+$.)
- *otočení* $X^{\mathrm{R}} = \{\alpha^{\mathrm{R}} \mid \alpha \in X\}$.

Věta: Pokud X a Y jsou regulární, všechny operace produkují opět regulární jazyky.

Důkaz: Automat pro *průnik* regulárních jazyků už umíme získat součinovou konstrukcí. *Doplněk* jsme vyřešili v cvičení 1.2.9, *otočení* vyřešíme ve cvičení 1.

Pro zbývající operace ukážeme, že z automatů pro X a Y umíme sestrojit automat pro výsledný jazyk. Budou se nám k tomu hodit λ-automaty s jednoznačným počátečním i koncovým (přijímajícím) stavem. Automat pro X označme A_X, jeho počáteční stav a_X a koncový stav z_X. Podobně pro Y.

Pro *sjednocení* vytvoříme nový počáteční stav a a nový koncový z. Přidáme λ-přechody $a \to a_X$, $a \to a_Y$, $z_X \to z$, $z_Y \to z$.

Pro *zřetězení* stačí přidat λ-přechod ze z_X do a_Y. Počátečním stavem bude a_X, koncovým z_Y.

Mocninu realizujeme jako k-násobné zřetězení (pro $k = 0$ stačí užít fakt, že každý konečný jazyk je regulární).

Pro *pozitivní iteraci* stačí přidat λ-přechod z koncového stavu do počátečního. Obecná *iterace* potřebuje přijímat navíc slovo ε: vytvoříme nový počáteční stav a a koncový z, přidáme λ-přechody $a \to z$, $a \to a_X$, $z_X \to z$, $z_X \to a_X$. □

Regulární výrazy

Postup, jak jazyk získat z jednodušších pomocí jazykových operací, můžeme popsat regulárním výrazem. To je buďto konstanta vyjadřující nějaký elementární jazyk, nebo opera-

ce aplikovaná na jednodušší regulární výrazy. Každému regulárnímu výrazu pak můžeme přiřadit nějaký jazyk *generovaný výrazem*, který budeme značit obvyklým $L(\ldots)$.

Výčet operací najdete na obrázku 1.1.

$\emptyset$	*prázdný jazyk*	$L(\emptyset) = \emptyset$
ε	*prázdné slovo*	$L(\varepsilon) = \{\varepsilon\}$
x	*znak abecedy*	$L(x) = \{x\}$
$X \mid Y$	*sjednocení*	$L(X \mid Y) = L(X) \cup L(Y)$
XY	*zřetězení*	$L(XY) = L(X) \cdot L(Y)$
X^*	*iterace*	$L(X^*) = L(X)^*$
X^+	*pozitivní iterace*	zkratka za XX^*
$X?$	*možnost*	zkratka za $X \mid \varepsilon$

Obrázek 1.1: Regulární výrazy a jimi generované jazyky

Příklad: Slova z $\{\mathtt{0},\mathtt{1}\}^*$ končící na `01` můžeme popsat regulárním výrazem $(\mathtt{0} \mid \mathtt{1})^*\mathtt{01}$.

Příklad: Slova z $\{\mathtt{0},\mathtt{1}\}^*$, ve kterých se pravidelně střídají `0` a `1`, můžeme popsat výrazem $(\mathtt{0} \mid \mathtt{1})^* \mid (\mathtt{1} \mid \mathtt{0})^* \mid (\mathtt{0} \mid \mathtt{1})^*\mathtt{0} \mid (\mathtt{1} \mid \mathtt{0})^*\mathtt{1}$, případně jednodušeji $\mathtt{1}?(\mathtt{0} \mid \mathtt{1})^*\mathtt{0}?$.

Už víme, že jazyky generované regulárnimi výrazy jsou vždy regulární. Překvapivě tak jde vygenerovat úplně každý regulární jazyk:

Věta (Kleeneho): Jazyk je generovaný regulárním výrazem právě tehdy, je-li regulární.

Důkaz: TODO □

Poznámka: UNIXové nástroje (například `grep` a `sed` používají nejrůznější varianty regulárních výrazů, které se od těch našich liší pouze detaily notace. Jinde (například v Perlu) ale najdeme „regulární“ výrazy vybavené i schopností zpětných odkazů a rekurze. Ty dokáží popsat i mnohé neregulární jazyky, ale algoritmy používané k jejich vyhodnocování nemají polynomiální časovou složitost.

Cvičení

1. Dokažte, že pro regulární jazyk X je jeho otočení X^{R} také regulární.

2. Jak vypadá jazyk X^{**}?

3. Napište regulární výraz, který generuje jazyk všech slov nad abecedou $\{\mathtt{0},\mathtt{1}\}$, jejichž počet jedniček je sudý nebo dělitelný třemi. Sestrojte odpovídající λ-NFA, ten převeďte na NFA a ten konečně na DFA.

4. Napište regulární výraz pro jazyk všech desítkových zápisů přirozených čísel (nedovolujeme nestandardní zápisy typu `0123` nebo ε). Podobně pro desetinná čísla (např. `3.1415`) a desetinná s periodou (značíme `0.0[01]`).

5. Napište regulární výraz pro jazyk všech slov nad abecedou $\{\texttt{0}, \ldots, \texttt{9}\}$, která jsou neklesající (`1377` v tomto jazyce leží, `735` nikoliv).

6. Vytvořte regulární výraz pro dvojkové zápisy čísel dělitelných třemi. Doporučujeme nejprve sestrojit DFA a pak použít konstrukci z důkazu Kleeneho věty.

7. Navrhněte co nejefektivnější algoritmus, který zjistí, zda zadané slovo je generované zadaným regulárním výrazem.