Týden 2

Připomeňme nejdříve algoritmus zachycený následujícím pseudokódem, který jsme použili minule při konstrukci kon. automatu (vlastně dvourozměrného pole $Next$) k danému vzorku $patt$: array[1..d] of char (pro $d \geq 1$):

$Next[0, patt[1]] := 1$; $\forall x \in \Sigma \setminus \{patt[1]\} : Next[0, x] := 0$; $Sec[1] := 0$;

for $i := 1$ to $d-1$ do

$Next[i, patt[i+1]] := i+1$;
$\forall x \in \Sigma \setminus \{patt[i+1]\} : Next[i, x] := Next[Sec[i], x]$;
$Sec[i+1] := Next[Sec[i], patt[i+1]]$;

$\forall x \in \Sigma : Next[d, x] := Next[Sec[d]; x]$;

Zde $Sec[i]$ je "sekundární možnost", tedy délka nejdelšího vlastního sufixu řetězce $patt[1..i]$, který je rovněž prefixem vzorku $patt$. Vypadá to na první pohled zamotaně, ale ilustrovali jsme si, že je to ve skutečnosti vlastně jednoduché.

(Ne)regulární jazyky

Připomněli jsme větu
Věta. Jazyk $L \subseteq \Sigma^*$ je regulární (tzn. přijímaný konečným automatem) právě tehdy, když je množina kvocientů $\{w\backslash L \mid w \in \Sigma^*\}$ konečná.
(Její důkaz je za bonusové body, jak je uvedeno na webu u průběhu výuky.)
Schopnost (alespoň intuitivní) aplikace věty jsme ověřovali na poznání (ne)regularity jazyků (pro $w \in \Sigma^*$, $|w|$ označuje délku slova w, $|w|_a$ označuje počet výskytů symbolu a ve w ("délka w v a-čkách")]):

1. $\{w \in \{a, b\}^*; |w|_a \bmod 2 = 0\}$
2. $\{w \in \{a, b\}^*; w$ začíná nebo končí dvojicí stejných písmen $\}$
3. $\{w \in \{a, b\}^*; |w|_a < |w|_b\}$
4. $\{w \in \{a, b, c\}^*$; jestliže w neobsahuje podřetězec abc pak končí řetězcem $bca\}$
5. $\{w \in \{a, b\}^*; |w|_a > |w|_b$ nebo w končí $baa\}$
6. $\{w \in \{a, b\}^*; |w|_a > |w|_b$ nebo $|w|_b \geq 2\}$

Redukce konečného automatu

Podívali jsme se na následující automat (který by vznikl po dotažení modulární konstrukce z první přednášky: má $3 \cdot 4 = 12$ stavů, jež vznikly přejmenováním původních dvojic).

	0	1
$\leftrightarrow s_1$	s_1	s_2
s_2	s_3	s_4
s_3	s_5	s_6
$\leftarrow s_4$	s_7	s_2
s_5	s_8	s_4
s_6	s_9	s_6
$\leftarrow s_7$	s_1	s_9
s_8	s_5	s_{10}
s_9	s_6	s_{11}
s_{10}	s_{12}	s_{10}
s_{11}	s_{11}	s_9
s_{12}	s_8	s_{11}

Rychle jsme zjistili, že všechny stavy jsou dosažitelné a promýšleli jsme, jak zjistit, zda existují dva různé stavy $q, q' \in Q = \{s_1, s_2, \ldots, s_{12}\}$ pro něž $L_q = L_{q'}$. Jde tedy o zjištění, zda pro q, q' existuje (či neexistuje) rozlišující slovo.
Uvědomili jsme si, že se nabízí induktivní postup, při němž konstruujeme ekvivalence $\sim_0, \sim_1, \sim_2, \ldots$ na množině Q definované takto:

$$q \sim_i q' \Leftrightarrow \text{pro stavy } q, q' \text{ neexistuje rozlišující slovo délky} \leq i.$$

Každé ekvivalenci $\sim_i$ odpovídá příslušný rozklad R_i na množině Q.
Výchozí R_0 jsme hravě sestrojili; má dvě třídy, přijímající a nepřijímající stavy.

$$R_0\text{:}\quad \mathrm{I} = \{s_1, s_4, s_7\},\ \ \mathrm{II} = \{s_2, s_3, s_5, s_6, s_8, s_9, s_{10}, s_{11}, s_{12}\}$$

Pak jsme začali sestrojovat R_1 (postupem popsaným ve studijním textu).

$$R_1\text{:}\quad \mathrm{I} = \{s_1, s_4, s_7\},\ \ \mathrm{II} = \{s_2, s_5\},\quad \mathrm{III} = \{s_3, s_6, s_8, s_9, s_{10}, s_{11}, s_{12}\}.$$

Uvědomili jsme si, že postup se opírá o toto pozorování:

> pokud pro q, q' existuje rozlišující slovo délky $i+1$, tak nutně existuje $b \in \Sigma$ (v našem případě $\Sigma = \{0, 1\}$) tak, že $q \xrightarrow{b} q''$, $q' \xrightarrow{b} q'''$ a pro q'', q''' existuje rozlišující slovo délky i.

(Pozn.: použili jsme symbol b pro proměnnou, ať si uvědomíme, že nemusíme vždy stereotypně používat a.)
Kdybychom pokračovali, zjistili bychom, že

R_2: $\mathrm{I} = \{s_1, s_4\}$, $\mathrm{II} = \{s_7\}$, $\mathrm{III} = \{s_2, s_5\}$, $\mathrm{IV} = \{s_3, s_8\}$, $\mathrm{V} = \{s_6, s_9, s_{10}, s_{11}, s_{12}\}$,

atd., až bychom dospěli k pevnému bodu, tedy k rozkladu, který se již nezjemní. Ten má 3-prvkovou třídu $\{s_6, s_9, s_{11}\}$ a pak už jen jednoprvkové třídy.
Redukcí (původně 12-stavového) automatu tedy vznikne ekvivalentní automat s 10 stavy (jeho stavy odpovídají třídám závěrečného rozkladu).

Ekvivalence konečných automatů; minimální automaty

Přemýšleli jsme, zda bychom uměli navrhnout (a naprogramovat) algoritmus řešící následující problém.

NÁZEV: *Evivalence konečných automatů*

VSTUP: dva konečné automaty A_1, A_2

VÝSTUP: ANO – jestliže $L(A_1) = L(A_2)$,
NE – jestliže $L(A_1) \neq L(A_2)$.

K jednomu možnému algoritmu nás přímo přivedla myšlenka, že u negativního případu, tedy $L(A_1) \neq L(A_2)$, by bylo dobré také ukázat protipříklad, tedy (co nejkratší) slovo w, které patří jen do jednoho z jazyků $L(A_1), L(A_2)$. Uvědomili jsme si, že takové slovo patří do jazyka

$$L = (L(A_1) - L(A_2)) \cup (L(A_2) - L(A_1)) = (L(A_1) \cap \overline{L(A_2)}) \cup (L(A_2) \cap \overline{L(A_1)})$$

a že díky dříve probraným algoritmům snadno sestrojíme A tak, že $L(A) = L$. Je nám tedy jasný algoritmus, který k zadaným A_1, A_2 sestrojí A tak, že

$$L(A_1) = L(A_2) \;\Leftrightarrow\; L(A) = \emptyset\,.$$

No a napsat proceduru (algoritmus), která o zadaném konečném automatu A zjistí, zda $L(A) = \emptyset$, hravě zvládneme (když si vzpomeneme na algoritmus pro zjišťování dosažitelných stavů; stačí prostě zjistit, zda nějaký přijímající stav je dosažitelný [z počátečního]).
Ekvivalence konečných automatů se ovšem dá algoritmicky (rychle) zjišťovat i jinak. Nejprve jsme si uvedli následující přirozenou definici.

> Konečný automat A je *minimální*, jestliže neexistuje automat A', který je ekvivalentní s A (pro nějž je tedy $L(A) = L(A')$) a který má méně stavů než A.

Pak jsme si uvedli tyto věty.
Věta. Je-li automat redukovaný, pak je minimální.
Věta. Dva minimální automaty, které přijímají tentýž jazyk, jsou izomorfní, tedy stejné až na pojmenování stavů; to také znamená, že mají stejný normovaný tvar.

Platnost vět jsme si naznačili jen intutivně, důkazům se věnujeme na rozšiřujících přednáškách. Uvědomili jsme si ale, že algoritmus rozhodující ekvivalenci konečných automatů může být tedy založen na odstranění nedosažitelných stavů, redukci (ztotožnění stavů se stejným L_q) a převodu do (přirozeně definovaného) *normovaného tvaru*. (Dva automaty jsou ekvivalentní právě tehdy, když po odstranění nedosažitelných stavů a následné redukci mají stejný normovaný tvar.)

Partie textu k prostudování

V návaznosti na část 3.3. (modulární návrh) se jedná zejména o část 3.4. (dosažitelné stavy, normovaný tvar), část 3.6. (minimalizace konečných automatů) a část 3.7. (ekvivalence konečných automatů, minimální automaty). (Máte si udělat přinejmenším dobrou první představu a zamyslet se nad příklady, speciálně těmi plánovanými na cvičení, ať se můžete na cvičení aktivně účastnit a případné problémy si tam objasnit.)

Cvičení

Příklad 2.1

Definujte jazyk
$L = \{w \in \{a,b\}^* \mid w$ začíná a a končí b nebo začíná b a končí $a\}$
jako (přirozené) sjednocení $L = L_1 \cup L_2$, kde
$L_1 = \{w \in \{a,b\}^* \mid \dots\}$
$L_2 = \{w \in \{a,b\}^* \mid \dots\}$
Zkonstruujte (co nejmenší) automat A_1 tak, že $L(A_1) = L_1$, a automat A_2 tak, že $L(A_2) = L_2$.
Aplikujte algoritmickou konstrukci vedoucí k automatu pro sjednocení $L_1 \cup L_2$. Přitom ovšem konstruujte jen dosažitelné stavy, podle definice
1/ $q_0 \in Reach(A)$,
2/ $(q \in Reach(A) \wedge q \longrightarrow q') \Longrightarrow q' \in Reach(A)$.
Definujte pak *normovaný tvar* automatu, při němž jsou stavy označeny $1, 2, 3, \dots$ a jsou uspořádány podle „vzdálenosti“ od počátečního stavu. Jak přesně vyjádříte onu vzdálenost, aby poskytla úplné uspořádání (dosažitelných) stavů?
Uveďte sestrojený automat do normovaného tvaru.

Příklad 2.2

O konečném automatu řekneme, že je *redukovaný*, jestliže

- všechny jeho stavy jsou dosažitelné
- a pro každé dva různé stavy q, q' platí $L_q \neq L_{q'}$.

Zkuste u vašeho (5-stavového?) automatu z předchozího příkladu nalézt pro každou dvojici stavů $q \neq q'$ slovo w tak, že $q \xrightarrow{w} F$ a $q' \not\xrightarrow{w} F$ (tedy $q' \xrightarrow{w} (Q-F)$) či naopak $q \not\xrightarrow{w} F$ a $q' \xrightarrow{w} F$; takovému slovu w budeme říkat

rozlišující slovo pro stavy q, q'.

Povedlo se? Pak je váš automat redukovaný.

Příklad 2.3

Dokončete redukci automatu z přednášky (tam jsme jen začali)

	0	1
$\leftrightarrow s_1$	s_1	s_2
s_2	s_3	s_4
s_3	s_5	s_6
$\leftarrow s_4$	s_7	s_2
s_5	s_8	s_4
s_6	s_9	s_6
$\leftarrow s_7$	s_1	s_9
s_8	s_5	s_{10}
s_9	s_6	s_{11}
s_{10}	s_{12}	s_{10}
s_{11}	s_{11}	s_9
s_{12}	s_8	s_{11}

Příklad 2.4

Připomeňme si větu:
Věta. Jazyk $L \subseteq \Sigma^*$ je regulární (tzn. přijímaný konečným automatem) právě tehdy, když je množina kvocientů $\{w\backslash L \mid w \in \Sigma^*\}$ konečná.
Vysvětlete, proč pro jazyk

$$L = \{a^n b^n \mid n \geq 0\}$$

jsou jazyky (kvocienty) $a\backslash L$, $aa\backslash L$, $aaa\backslash L$, ... navzájem různé. Vyvoďte, že L není regulární.

Příklad 2.5

Vybudujte si intuici o (ne)regulárních jazycích promyšleným zodpovězením otázek na s. 88 a 89. Ty se týkají (ne)regularity následujících jazyků. (Zápisem w^R značíme zrcadlový obraz [reversal] slova w; induktivně lze definovat takto: $\varepsilon^R = \varepsilon$; $(au)^R = u^R a$.)
$\{w \in \{a,b\}^*; |w|_a \bmod 2 = 0\}$

$\{w \in \{a,b\}^*; w$ začíná nebo končí dvojicí stejných písmen $\}$
$\{w \in \{a,b\}^*; |w|_a < |w|_b\}$
$\{w \in \{a,b,c\}^*$; jestliže w neobsahuje podřetězec abc, pak končí $bca\}$
$\{w \in \{a,b\}^*; |w|_a > |w|_b$ nebo w končí $baa\}$
$\{w \in \{a,b\}^*; |w|_a > |w|_b$ nebo $|w|_b \geq 2\}$
$\{u$; ex. $w \in \{a,b\}^*$ tak, že $u = ww^R\}$, stručněji také psáno $\{ww^R; w \in \{a,b\}^*\}$,
$\{w \in \{a,b\}^*; w = w^R\}$
$\{w \in \{a\}^*; w = w^R\}$
$\{ww; w \in \{a,b\}^*\}$
$\{ww; w \in \{a\}^*\}$
$\{w \in \{a,b\}^*$; rozdíl počtů znaků a a znaků b ve w je větší než 100$\}$
$\{w \in \{a,b\}^*$; součin $|w|_a$ a $|w|_b$ je větší nebo roven 100$\}$
$\{w \in \{a\}^*; |w|$ je prvočíslo $\}$

Příklad 2.6

Připomeňme problém

NÁZEV: *Evivalence konečných automatů*

VSTUP: dva konečné automaty A_1, A_2

VÝSTUP: ANO – jestliže $L(A_1) = L(A_2)$,
NE – jestliže $L(A_1) \neq L(A_2)$.

a uvažujme následující algoritmus pro jeho řešení:

> generuj postupně (všechna) slova $w_0, w_1, w_2, \ldots$ v abecedě daných automatů; pro každé w_i přitom zjisti, zda je oběma automaty přijímáno či oběma nepřijímáno – když je jedním přijímáno a druhým nepřijímáno, skonči s výsledkem $L(A_1) \neq L(A_2)$.

Je zřejmé, že běh tohoto algoritmu neskončí v případě $L(A_1) = L(A_2)$. Lze tento nedostatek opravit tím, že necháme algoritmus probírat jen slova do délky n, kde n je součtem počtu stavů A_1 a A_2, a nenajde-li se rozlišující slovo do té doby, pak algoritmus zahlásí $L(A_1) = L(A_2)$? Pokud je tomu tak (tedy odpověď algoritmu je vždy správná), v čem je tento algoritmus zřejmě horší než algoritmy probrané na přednášce?

Příklad 2.7

Nalezněte všechny dvojice stavů q, q', pro něž platí $q \sim q'$, tedy $L_q = L_{q'}$.

	a	b
→0	0	1
←1	1	2
←2	3	1
3	2	4
4	2	3

	a	b
→5	5	6
6	7	5
←7	7	9
8	9	8
←9	8	7

Příklad 2.8

Pro následující vztahy vysvětlete, proč obecně platí či neplatí.

- $L_1 \cdot L_2 = L_2 \cdot L_1$
- $L_1(L_2 \cup L_3) = L_1L_2 \cup L_1L_3$
- $(L_1 \cup L_2)^* = L_1^*(L_2 \cdot L_1^*)^*$
- $(L_1 \cap L_2)^* = L_1^* \cap L_2^*$
- $w \cdot (w \backslash L) = L$