

Týden 3

Přednáška - první část

(viz také slidy k této přednášce ...)

Ekvivalence konečných automatů; minimální automaty

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

NÁZEV: *Ekvivalence 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 intuitivně, důkazům se věnujeme na přednáškách pro hlubší zájemce. 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^{toAcc}) a převodu do normovaného tvaru. (Dva automaty jsou ekvivalentní právě tehdy, když po odstranění nedosažitelných stavů a následné redukci mají stejný normovaný tvar.)

Nedeterministické konečné automaty

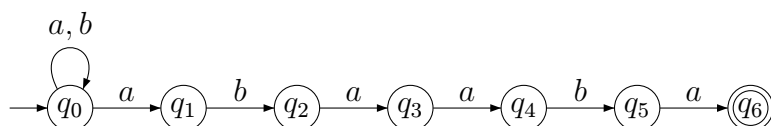
Definici nedeterministického konečného automatu

$$A = (Q, \Sigma, \delta, I, F),$$

kde $\delta : Q \times \Sigma \rightarrow \mathcal{P}(Q)$ ($\mathcal{P}(Q)$ značí množinu všech podmnožin množiny Q),

$$I \subseteq Q,$$

jsme si osvětlili na následujícím příkladu (kde $I = \{q_0\}$, $\delta(q_0, a) = \{q_0, q_1\}$, $\delta(q_2, b) = \emptyset$, atd.)



a uvědomili jsme si, že pro automat A na obrázku je $L(A) = \{w \mid \exists q \in I : q \xrightarrow{w} F\}$ roven jazyku, pro nějž jsme konstruovali (deterministický) konečný automat na první přednášce, totiž jazyku

$$L(A) = \{w \in \{a, b\}^* \mid w \text{ má sufix } abaaba\}.$$

Připomněli jsme si, že existuje algoritmus, který převede zadaný nedeterministický konečný automat na ekvivalentní deterministický; tento algoritmus je popsán ve studijním textu (formou „knoflíkové hry“).

Zobecněné nedeterministické konečné automaty (s ε -přechody)

Zatím neformálně jsme uvedli příklad regulárního výrazu, konkrétně $(aa)^*(bb)^*(cc)^*$ a shodli se, že ho chápeme jako reprezentaci jazyka $L = \{a^{2k}b^{2\ell}c^{2m} \mid k, \ell, m \geq 0\}$.

Pak jsme si uvědomili, jak elegantně lze z automatů pro jazyky $\{a^{2k} \mid k \geq 0\}$, $\{b^{2\ell} \mid \ell \geq 0\}$, $\{c^{2m} \mid m \geq 0\}$ vytvořit automat přijímající jazyk L , když použijeme ε -přechody.

Uvedli jsme takto zobecněný nedeterministický konečný automat (ZNKKA), tedy strukturu $A = (Q, \Sigma, \delta, I, F)$, kde je patřičně rozšířen definiční obor přechodové funkce:

$$\delta : Q \times (\Sigma \cup \{\varepsilon\}) \rightarrow \mathcal{P}(Q).$$

Přirozeně jsme došli k tomu, jak zde budeme chápat značení $q \xrightarrow{w} q'$ (či $q \xrightarrow{w} F$ apod.).

Stručná induktivní definice může vypadat takto:

1. $q \xrightarrow{\varepsilon} q$,
2. když $\delta(q, a) \ni q'$ (pro $a \in \Sigma \cup \{\varepsilon\}$) a $q' \xrightarrow{v} q''$, tak $q \xrightarrow{av} q''$.

I pro ZNKA A lze tedy definovat $L(A) = \{w \in \Sigma^* \mid \exists q \in I : q \xrightarrow{w} F\}$.

Myšlenku důkazu věty 3.24. (s. 103)

Existuje algoritmus, který ke každému zobecněnému nedeterministickému konečnému automatu A sestrojí ekvivalentní (deterministický) konečný automat A' (tedy $L(A) = L(A')$).

jsme si demonstrovali na výše sestrojeném automatu (a de facto jsme tak provedli Cvičení 3.64, s. 102).

Uzavěrové vlastnosti třídy regulárních jazyků

Uvědomili jsme si, že umíme snadno dokázat věty typu 3.26, 3.27, 3.28 (sekce 3.10), tedy

Třída REG je uzavřena vůči sjednocení, průniku, doplňku. (Je-li tedy $L_1, L_2 \in \text{REG}$, pak také $L_1 \cup L_2$, $L_1 \cap L_2$, $\overline{L_1}$ jsou v REG.)

Třída REG je uzavřena vůči zřetězení a iteraci. (Je-li tedy $L_1, L_2 \in \text{REG}$, pak také $L_1 \cdot L_2$, a $(L_1)^*$ jsou v REG.)

Třída REG je uzavřena vůči operaci zrcadlového obrazu. (Je-li tedy $L \in \text{REG}$, pak také $L^R \in \text{REG}$.)

Také jsme si uvědomili konstruktivnost těchto tvrzení (tedy existenci příslušných algoritmů). Speciálně jsme si odvodili konstrukce zachycené na obrázku 3.15 na s. 114.

Regulární výrazy

Přesněji jsme uvedli, co myslíme pojmem *regulární výraz* (RV) v našem kontextu, a jakým způsobem reprezentuje výraz α určitý jazyk, označovaný $[\alpha]$. Uvedli jsme tedy Definici 3.30 a 3.31 (v sekci 3.12) a demonstrovali na příkladu (z Cvičení 3.80)

$$((01^*0 + 101)^*100 + (11)^*0)^*01,$$

na němž jsme rovněž ilustrovali standardní domluvu o prioritě operátorů, vynechávání znaku “.”, vynechávání nepotřebných závorek apod.

Uvědomili jsme si, že máme de facto k dispozici všechny prostředky pro návrh algoritmu z Věty 3.22

Ke každému regulárnímu výrazu α lze sestrojít konečný automat přijímající jazyk $[\alpha]$.

za předpokladu, že umíme (algoritmicky) rozebrat syntaktickou strukturu zadaného výrazu. To znamená, že víme, jak sestavit příslušný syntaktický strom; např. pro výraz

$$(1*0 + 10)*1$$

může být takový strom zachycen lineárním zápisem

$$\text{Conc}(\text{Iter}(\text{Union}(\text{Conc}(\text{Iter}(1), 0), \text{Conc}(1, 0))), 1).$$

Začali jsme řešení Cvičení 3.80, naznačili jsme tedy průběh převodu $RV \rightarrow ZNKA$ na výše uvedeném výrazu $((01*0 + 101)*100 + (11)*0)*01$.

Regulární a neregulární jazyky

Připomněli jsme si stručně intuici budovanou na minulém cvičení (otázky na s. 88-89). Uvědomili jsme si, že kombinace ‘regulární’ a ‘neregulární’ podmínky vždy vyžaduje hlubší zamyšlení. (Např. Otázky 3.51 a 3.52 demonstrují, že sjednocení $L_1 \cup L_2$ dvou jazyků, z nichž jeden je regulární a druhý neregulární, je v některých případech regulární a v jiných neregulární.)

Přednáška - druhá část

Dokončili jsme úvahy z minula, mj. jsme ukázali, že pro konečnou abecedu Σ je množina Σ^* spočetná, zatímco množina všech jazyků nad abecedou Σ je nespočetná.

Vrátili jsme se k pojmu podílový automat (definovaný při redukci konečných automatů) ... a také se pobavili o *koinduktivním* přístupu k jeho konstrukci ...

Dále jsme se věnovali (matematickému) popisu algoritmu, který převede zadaný nedeterministický konečný automat na ekvivalentní deterministický, a důkazu jeho korektnosti.

Zamysleli jsme se nad větami

Věta. Je-li automat redukováný a nemá nedosažitelné stavy, pak je minimální.

Věta. Dva minimální automaty, které přijímají tentýž jazyk (a jsou tedy ekvivalentní), jsou izomorfní, tedy stejné až na pojmenování stavů.

Věta. Regulárními výrazy lze reprezentovat právě regulární jazyky.

(Podrobnější poznámky lze nalézt v průběhu výuky v loňském roce ...)

Partie textu k prostudování

Jedná se zejména o části 3.7. (ekvivalence konečných automatů, minimální automaty), 3.8. (regulární a neregulární jazyky), 3.9. (nedeterministické konečné automaty), 3.10. (uzávěrové vlastnosti třídy regulárních jazyků), 3.12. (regulární výrazy).

(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

Prezentace referátů 3 a 4

Příklad 3.1

Připomeňme problém

NÁZEV: *Ekvivalence 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, \dots 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 3.2

	0	1
→ 1	1,2	1
2	3	-
3	-	4
④	-	-
→ 5	5	5,6
6	-	7
7	-	8
8	-	9
⑨	9	9

NKA (nedeterministický konečný automat) A zadaný uvedenou tabulkou zadejte jako (matematickou) strukturu $A = (Q, \Sigma, \delta, I, F)$ a pak zakreslete grafem. Charakterizujte co nejjednodušeji jazyk $L(A)$, tedy množinu $\{w \in \Sigma^* \mid \exists q \in I : q \xrightarrow{w} F\}$.

Pak zkonstruuje ekvivalentní DKA (deterministický konečný automat). Dokončete tedy konstrukci následující tabulky.

		0	1	
→	K_0	K_1	K_2	$K_0 = \{1, 5\}$
	K_1	K_3		$K_1 = \{1, 2, 5\}$
	K_2			$K_2 = \{1, 5, 6\}$
	K_3			$K_3 = \{1, 2, 3, 5\}$

(Pro procvičení můžete výsledný DKA zredukovat a převést do normovaného tvaru.)

Příklad 3.3

Připomeňme si, co jsou regulární výrazy (standardní v teorii). Zjistěte, zda platí

$$[(011 + (10)^*1 + 0)^*] = [011(011 + (10)^*1 + 0)^*]$$

$$[((1 + 0)^*100(1 + 0)^*)^*] = [((1 + 0)100(1 + 0)^*100)^*]$$

Příklad 3.4

Proveďte syntaktický rozbor výrazu $(01^*0 + 10)^*10$ (konstrukcí syntaktického stromu a odpovídajícího lineárního zápisu používajícího „procedury“ *Union*, *Conc*, *Iter*) a aplikujte algoritmus převodu $RV \rightarrow ZNKA$.

K výslednému ZNKA alespoň započnete konstrukci ekvivalentního DKA.

Příklad 3.5

(Nepovinně.)

Zadejte regulárním výrazem jazyk

$$L = \{ w \in \{0, 1\}^* \mid \text{ve } w \text{ je sudý počet nul a každá jednička je bezprostředně následována nulou} \}$$

Příklad 3.6

(Nepovinně.)

Při konstrukci ekvivalence $\sim \subseteq Q \times Q$ definované vztahem $q \sim q' \Leftrightarrow L_q^{toAcc} = L_{q'}^{toAcc}$ (pro automat $A = (Q, \Sigma, \delta, q_0, F)$), jsme de facto postupovali *koinduktivně*: vzali jsme (největší) relaci $R_0 = Q \times Q$, na ni jsme aplikovali určitý monotónní funkcionál \mathcal{F} , dostali jsme tedy (menší) relaci $R_1 = \mathcal{F}(R_0)$, pak jsme dostali relaci $R_2 = \mathcal{F}(R_1)$, atd., až jsme dospěli k (největšímu) pevnému bodu R (pro nějž $R = \mathcal{F}(R)$). (Monotónností funkcionálu $\mathcal{F} : 2^{Q \times Q} \rightarrow 2^{Q \times Q}$ zde rozumíme vlastnost, že pro $T_1 \subseteq T_2$ máme $\mathcal{F}(T_1) \subseteq \mathcal{F}(T_2)$.) Snažte se funkcionál \mathcal{F} přesně popsat.