Automaty a gramatiky

Jasoň
korektury: Miloš Teodorovič
odborné rady a garance: Václav Koubek

10. listopadu 1996

Tato skripta slouží jako pomůcka k přednášce „Automaty a gramatiky“ na MFF UK. Obsahují tedy pouze základní úvod do dané problematiky. Většina kapitol zde uvedených je zestručněním (a také „polidštěním“, jak doufám) knihy [2].

Obsah

1 Základní pojmy z teorie jazyků

- Abeceda A je libovolná konečná množina A
- prvky abecedy A budeme nazývat písmena, resp. symboly
- Slovem nazveme každou konečnou posloupnost písmen, její délku nazveme délkou slova
- prázdné slovo (slovo délky 0) označíme Λ
- množinu všech slov nad abecedou A označíme A^*
- $A^+ = A^* \setminus \{\Lambda\}$ (množina všech neprázdných slov nad A)
- libovolnou podmnožinu L množiny A^* nazveme jazykem nad A ($L \subseteq A^*$)
- pro slova $\alpha = a_1 \dots a_m$ a $\beta = b_1 \dots b_n$ definujeme operaci conc tak, že $conc(\alpha, \beta) = \alpha\beta = a_1 \dots a_m b_1 \dots b_n$ (pro $m, n \in N$) (jedná se o složení řetězců, resp. slov) [1]

Poznámka: Na tomto místě považuji za velmi vhodné si připomenout, co znamená *konečná* posloupnost, resp. množina. Konečnou množinou máme na mysli množinu konečné velikosti, neboli velikost této množiny je rovna některému přirozenému číslu[2]. Podobně posloupnost je konečná, pokud je její délka rovna přirozenému číslu. Přirozených čísel je ale nekonečně mnoho, z čehož například vidíme, že množina A^* všech slov nad abecedou A je nekonečná, kdykoliv $A \neq \emptyset$. Věřím, že se k této poznámce rádi vrátí všichni, kdo v průběhu čtení následujících kapitol zabloudí v pojmech *konečno*, *nekonečno* a podobně. Pro případ nejvyšší nouze připojuji místo záchranného kruhu fakt shrnující celou poznámku:

> Přirozených čísel je nekonečně mnoho. Každé přirozené číslo je konečné.

Naprosto stejná tvrzení platí samozřejmě i pro N_0 a Z (celá čísla).

Teorie jazyků nachází široké uplatnění v lingvistice, teorii programovacích jazyků, biologii a dalších oborech.

2 Úvod do teorie automatů

2.1 Historie

Teorie automatů je poměrně mladá věda. Vznikla počátkem druhé čtvrtiny 20. století na popud logiky jako obor, hledající možnosti konečného popisu nekonečných objektů. U jejího zrodu stáli zejména Turing, Kleene, Post, Church a Markov. Definice Turingova stroje z roku 1936 natrvalo zařadila výzkum automatů mezi významná odvětví moderní vědy. K dalšímu rozvoji přispěli v roce

[1] Lehce nahlédneme, že operace skládání řetězců je asociativní, Λ je její jednotkový prvek, a že tedy $A^*(conc, \Lambda)$ je monoid (dokonce volný nad množinou A).

[2] značíme N, resp. N_0 pro přirozená čísla obohacená o 0.

Tabulka 1: Třídy jazyků

Typ akceptoru (A)	$\longleftrightarrow$	Třída jazyků rozpoznávaná akceptory typu A
Typ gramatiky (G)	$\longleftrightarrow$	Třída jazyků generovaná gramatikami typu G

Tabulka 2: Schéma práce gramatik a akceptorů

Vstupní slovo	$\longrightarrow$	Akceptor A	$\longrightarrow$	Výstup (slovo patří, resp. nepatří do jazyka rozpoznávaného akceptorem A)
Vstup (bázová slova jazyka L)	$\longrightarrow$	Gramatika G	$\longrightarrow$	Jazyk L

1943 biologové McGullcik a Pitts. V letech 1954–1960 byly konečně položeny základy formální teorie konečných automatů (Huffmann, Kleene), což vedlo ke zvýšenému zájmu o tento obor v 60-tých letech.

2.2 Automaty a jazyky

Teorie automatů se zabývá konstrukcí zařízení na rozpoznávání jazyků (akceptory, resp. automaty) a mechanismů na generování jazyků (gramatiky). Každému typu akceptoru (gramatiky) lze přiřadit třídu jazyků, které dané zařízení rozpozná (generuje). Vlastnostmi tříd jazyků rozpoznávaných (generovaných) daným zařízením a vztahy mezi jazyky rozpoznávanými (generovanými) různými akceptory (gramatikami) se zabývá teorie jazyků (viz tab. 1). Schématické znázornění práce rozpoznávacích zařízení a generujících mechanismů naleznete v tab. 2.

2.3 Klasifikace automatů

Automaty[1] lze klasifikovat (třídit) podle nejrůznějších hledisek. Mezi nejpoužívanější patří

- použitá paměť[2]
- způsob určení následující konfigurace[3]
- přístup ke vstupním datům[4]

[1] nebo akceptory, protože to je pro naše potřeby jedno a to samé. Automaty je poněkud obecnější slovo, akceptory výstižnější

[2] žádná, zásobník, atd.

[3] jak je zadán způsob přechodu automatu z jednoho stavu do druhého (pevně daný, výběr z množiny možných přechodů)

[4] způsob čtení vstupních dat (pohyb ve vstupním slově, které má automat prověřit)

V následujících kapitolách se nejprve budeme trochu zabývat automaty obecně, potom přejdeme k akceptorům a vztahům mezi akceptory a jazyky.

2.4 Mooreův automat

Definice 2.1 *Mooreovým automatem rozumíme šestici* $(Q, X, \delta, q_0, Y, \beta)$*, resp. pětici* (Q, X, δ, Y, β)[5]*, kde*

- X *označuje vstupní abecedu*
- Q *(konečnou) množinu stavů automatu*
- $\delta : Q \times X \to Q$ *přechodovou funkci*
- $q_0 \in Q$ *počáteční stav*
- Y *výstupní abecedu*
- $\beta : Q \to Y$ *výstupní funkci*

Mooreův automat si představujeme jako zařízení, které má kontrolní jednotku, která může nabývat konečně mnoha stavů, vstupní a výstupní pásku. Pásky jsou rozdělené na jednotlivé políčka, která jsou očíslována kladnými celými čísly. Každé políčko může obsahovat jen jeden znak abecedy nebo může být prázdné. Vstupní slovo $\alpha = a_1 \dots a_n$ je napsané na prvních n políčkách vstupní pásky, tj. a_i je na i-tém políčku pro $i \leq n$ a pro $i > n$ je i-té políčko prázdné. Po vstupní pásce se pohybuje hlava, která přečte vždy obsah jednoho políčka. Po výstupní pásce se pohybuje hlava, která zapisuje na tuto pásku výstup automatu. Automat tedy pracuje v jednotlivých taktech. Na počátku své činnosti se automat nachází ve stavu q_0 a obě hlavy jsou na prvních políčkách své pásky. V jednom taktu hlava na vstupní pásce přečte písmeno ze vstupu, které je napsáno na prohlíženém i-tém políčku, a v závislosti na přečteném písmenu a funkci δ přechází do dalšího stavu. Hlava na vstupní pásce se přesune na $i+1$-ní políčko. Zároveň hlava na výstupní pásce zapíše výstup automatu, který závisí pouze na stavu automatu. Tuto závislost určuje funkce β. Po zapsání výstupu se hlava na výstupní pásce přesune také na další políčko.

Je přirozené definovat rozšíření funkcí δ (přechod z jednoho stavu do jiného v závislosti na přečteném písmenu) na $\delta^* : Q \times X^* \to Q$ (přechod z jednoho stavu do jiného v závislosti na přečteném slově) a β (výstup v závislosti na momentálním stavu automatu) na $\beta^* : Q \times X^* \to Y$ (výstup v závislosti na přečteném slově) následovně:

(1) $\delta^*(q, \Lambda) \stackrel{df}{=} q \qquad \delta^*(q, \alpha a) \stackrel{df}{=} \delta(\delta^*(q, \alpha), a),$

(2) $\beta^*(q, \alpha) \stackrel{df}{=} \beta(\delta^*(q, \alpha)),$

pro $q \in Q$, $a \in X$ a $\alpha \in X^*$.
Dále definujme překladovou funkci $\beta_p : Q \times X^* \to Y^*$

(3) $\beta_p(q, \Lambda) \stackrel{df}{=} \Lambda \qquad \beta_p(q, \alpha a) \stackrel{df}{=} \beta_p(q, \alpha)\beta^*(q, \alpha a)$

[5] někdy nás nezajímá počáteční stav, ale práce automatu bez závislosti na něm

pro $q \in Q$, $a \in X$ a $\alpha \in X^*$. Překladová funkce, jak je zřejmé z definice, vrátí pro dané vstupní slovo odpovídající slovo výstupní[6].

Tvrzení 2.2 $(\forall\, \alpha, \gamma \in X^*)(\forall\, q \in Q)$

$$\begin{aligned} \delta^*(q, \alpha\gamma) &= \delta^*(\delta^*(q, \alpha), \gamma) \\ \beta^*(q, \alpha\gamma) &= \beta^*(\delta^*(q, \alpha), \gamma) \\ \beta_p(q, \alpha\gamma) &= \beta_p(q, \alpha)\beta_p(\delta^*(q, \alpha), \gamma). \end{aligned}$$

Důkaz: Provedeme pro první rovnost (ostatní jsou analogické) indukcí dle délky γ. Pro $\gamma = \Lambda$ rovnost plyne z definice. Předpokládejme tedy $\gamma \neq \Lambda$. Buď $\gamma = \gamma' b$ (pro nějaké $\gamma' \in X^*, b \in X$). Potom postupně z definice, indukčního předpokladu a opět z definice dostáváme

$$\delta^*(q, \alpha\gamma) = \delta(\delta^*(q, \alpha\gamma'), b) = \delta(\delta^*(\delta^*(q, \alpha), \gamma'), b) = \delta^*(\delta^*(q, \alpha), \gamma)$$

Čímž je důkaz proveden. CBD

2.5 Mealyho automat

Definice Mealyho automatu je velmi blízká Mooreovu automatu (a jak se později ukáže, dá se jeden na druhý převést).

Definice 2.3 *Mealyho automat je šestice $(Q, X, \delta, q_0, Y, \beta)$, resp. při vynechání iniciálního stavu pětice (Q, X, δ, Y, β). Jednotlivé prvky odpovídají popisu v kapitole 2.4 (Mooreův automat), až na β, které je zde definováno jako zobrazení $\beta : Q \times X \to Y$. Výstup Mealyho automatu tedy závisí na momentálním stavu a znaku na vstupu.*

I pro Mealyho automat definujeme zobrazení δ^* a β^*, a to stejně jako pro Mooreův automat.[7]

2.6 Zadávání automatů v praxi

Nejčastějším způsobem zadání Mooreova automatu $M = (Q, X, \delta, q_0, Y, \beta)$ je tabulka $|\,Q\,| \times (|\,X\,| + 1)$, kde každý řádek přísluší jednomu stavu a každý sloupec jednomu vstupnímu symbolu. V posledním sloupci jsou uvedeny hodnoty výstupní funkce β. Ostatní sloupce definují funkci δ, tzn. danému stavu z Q přiřazují pro daný vstup opět nějaký stav z Q. Vstupní stav q_0 (pokud je nutné jej zadat) se označuje šipkou.

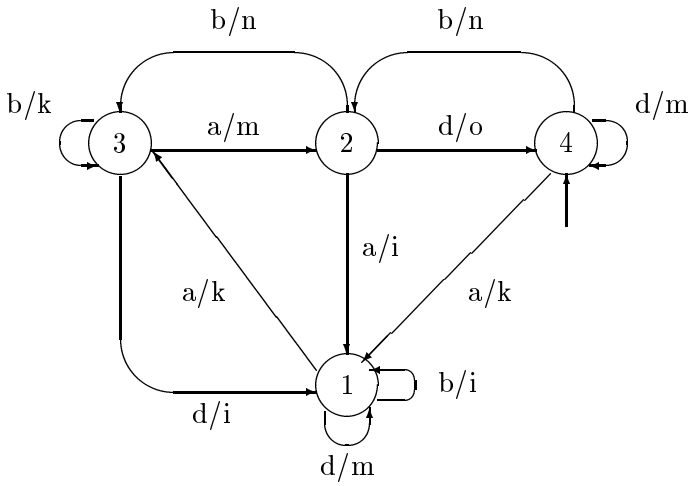
Mealyho automat se zadává podobnou tabulkou, jen poslední sloupec chybí a položky tabulky jsou tvořeny dvojicemi δ/β, kde δ je opět hodnota přechodové a β výstupní funkce.

Následující tabulky ukazují příklady zadání

- Mooreova automatu $M^o = (Q^o, X^o, \delta^o, Y^o, \beta^o)$ (vlevo),

[6] přičemž práce automatu nad slovem α nemusí začínat v q_0; překladová funkce je definována pro všechny stavy z Q

[7] až na $\beta^*(q, \Lambda)$, což v případě Mealyho automatu nemá smysl!



Obr. 1: Automat M^e zadaný graficky

- Mealyho automatu $M^e = (Q^e, X^e, \delta^e, Y^e, \beta^e)$ (vpravo).

M^o	a	b	c	d	
$\Rightarrow q_1$	q_1	q_5	q_2	q_4	1
q_2	q_5	q_2	q_1	q_3	0
q_3	q_1	q_1	q_2	q_1	1
q_4	q_3	q_2	q_5	q_4	1
q_5	q_2	q_4	q_3	q_2	0

M^e	a	b	d
1	$3/k$	$1/i$	$1/m$
2	$1/i$	$3/n$	$4/o$
3	$2/m$	$3/k$	$1/i$
$\Rightarrow$ 4	$1/k$	$2/n$	$4/m$

Z tabulek lze například vyčíst, že automat M^o přejde ze stavu q_3 při vstupu b do stavu q_1 a na výstupu je 1. Automat M^e ze stavu 2 při vstupu d přejde do stavu 4, na výstupu bude o.[8]

Jiným poměrně běžným způsobem zadání automatu je orientovaný graf, ve kterém stavy tvoří uzly a hrany znázorňují přechody mezi stavy. Obrázek 1 ukazuje grafickou reprezentaci automatu M^e zadaného výše uvedenou tabulkou.

2.7 Ekvivalence Mooreova a Mealyho automatu

Definice 2.4 *Mějme dva Mooreovy automaty*[9] $(Q_i, X, \delta_i, Y, \beta_i)$, $i = 1, 2$ *se stejnou vstupní a výstupní abecedou. Řekneme, že stavy* q_1 *a* q_2 $(q_i \in Q_i)$ *jsou ekvivalentní (píšeme* $q_1 \sim q_2$*), pokud pro všechna* $\alpha \in X^*$ *platí*

(4) $\beta_1^*(q_1, \alpha) = \beta_2^*(q_2, \alpha)$.

[8] Jako cvičení by nebylo špatné zkusit si vypsat vstupní a výstupní abecedy a množiny stavů obou automatů

[9] ne nutně různé

Řekneme, že stavy q_1 a q_2 jsou Λ-ekvivalentní (píšeme $q_1 \overset{\Lambda}{\sim} q_2$), pokud (4) *platí pro všechna $\alpha \in X^+$.*[10]

Pro Mealyho automaty je Λ-ekvivalence definována stejně jako pro Mooreovy. Ekvivalence na Mealyho automatech nemá smysl, neboť v její definici vystupuje libovolné $\alpha \in X^*$, tedy i $\alpha = \Lambda$. Výstupní funkce β, potřebná pro zavedení pojmu ekvivalence, však nad prázdným slovem v Mealyho automatu není definována. Pojem Λ-ekvivalence definujeme analogicky také pro jeden Mealyho a jeden Mooreův automat.

Definice 2.5 *Řekneme, že dva Mooreovy automaty $(Q_i, X, \delta_i, Y, \beta_i)$, $i = 1, 2$ mají stejné chování (resp. stejné Λ-chování), pokud*

$$(\forall\, q_1 \in Q_1)(\exists\, q_2 \in Q_2)(q_1 \sim q_2)$$
$$(\forall\, q_2 \in Q_2)(\exists\, q_1 \in Q_1)(q_1 \sim q_2),$$

resp. $q_1 \overset{\Lambda}{\sim} q_2$, jde-li o Λ-chování.

Podobně jako u předchozí definice 2.4, i zde je nutno poznamenat, že pro Mealyho automaty je stejné Λ-chování definováno jako pro Mooreovy. Definovat pro Mealyho automaty stejné chování však nemá smysl. Stejné Λ-chování definujeme beze změny také pro jeden Mealyho a jeden Mooreův automat.

Věta 2.6 *(Moore→Mealy) Mějme Mooreův automat $(Q, X, \delta, Y, \beta_1)$ a Mealyho automat $(Q, X, \delta, Y, \beta_2)$, pro které platí*

(5) $\quad (\forall\, q \in Q)(\forall\, a \in X)(\beta_2(q, a) = \beta_1(\delta(q, a)))$

Potom mají tyto dva automaty stejné Λ-chování.

Důkaz: Podle definice 2.5 je třeba najít ke každému stavu jednoho automatu Λ-ekvivalentní stav automatu druhého a naopak. Relace Λ-ekvivalence se nám zde přímo nabízí coby identita, tzn. stav $q \in Q$ Mealyho automatu označíme za Λ-ekvivalentní s týmž stavem q v Mooreově a naopak. Musíme však ověřit, že identita id_Q[11] je zde skutečně Λ-ekvivalence (viz definice 2.4). Vezměme libovolný stav $q \in Q$, nějaké $\alpha = \alpha' a$ pro $\alpha' \in X^*$, $a \in X$. Z definice β^*, předpokladu (5), definice δ^* a opět definice β^* dostáváme

$$\begin{aligned} \beta_2^*(q, \alpha) &= \beta_2(\delta^*(q, \alpha'), a) = \\ &= \beta_1(\delta(\delta^*(q, \alpha'), a)) = \beta_1(\delta^*(q, \alpha' a)) = \beta_1^*(q, \alpha) \end{aligned}$$

Za podmínky (5) jsme ke každému stavu Mooreova automatu našli (identický) Λ-ekvivalentní stav Mealyho automatu a naopak, tedy tyto automaty mají stejné Λ-chování. CBD

[10] Jinými slovy, dva stavy jsou ekvivalentní, pokud výpočet započatý v každém z nich nad stejným slovem dává stejný výstup

[11] pod zápisem id_Q se skrývá identická relace nad množinou Q, tzn. každý prvek je ekvivalentní právě sám se sebou

Věta 2.7 *(Mealy→Moore) Mějme Mealyho automat* $(Q, X, \delta_1, Y, \beta_1)$ *a Mooreův automat* $(Q \times Y, X, \delta_2, Y, \beta_2)$*, pro které platí* $(\forall\, q \in Q)(\forall\, a \in X)(\forall\, y \in Y)$

$$(6) \quad \delta_2((q,y),a) = (\delta_1(q,a), \beta_1(q,a))$$

$$(7) \quad \beta_2((q,y)) = y$$

Potom mají tyto dva automaty stejné Λ*-chování.*

Důkaz: Abychom prokázali stejné Λ-chování dvou automatů, musíme najít příslušné Λ-ekvivalentní stavy. Jak by asi každý čekal, za relaci Λ-ekvivalence vezmeme

$$(\forall\, q \in Q)(\forall\, y \in Y)((q,y) \overset{\Lambda}{\sim} q)$$

Ověřit, že je to skutečně Λ-ekvivalence, znamená dokázat rovnost

$$(8) \quad (\forall\, \alpha \in X^+)(\beta_2^*((q,y),\alpha) = \beta_1^*(q,\alpha))$$

Dokážeme nejprve vztah

$$(9) \quad (\forall\, \alpha \in X^+)(\forall\, q \in Q)(\forall\, y \in Y)(\delta_2^*((q,y),\alpha) = (\delta_1^*(q,\alpha), \beta_1^*(q,\alpha)))$$

který později využijeme. Důkaz vztahu (9) provedeme indukcí podle délky α: pro $\alpha = a \in X$ je z definice δ^*, předpokladu (6) a z definice δ^* a β^*

$$\delta_2^*((q,y),a) = \delta_2((q,y),a) = (\delta_1(q,a), \beta_1(q,a)) = (\delta_1^*(q,a), \beta_1^*(q,a))$$

Pro slovo délky jedna tedy vztah (9) platí. Předpokládejme nyní $\alpha = \alpha' a$ a nechť pro α' platí (9). Potom z definice δ^*, indukčního předpokladu, předpokladu (6) a definice δ^* a β^* můžeme psát

$$\begin{aligned} \delta_2^*((q,y),a) &= \delta_2(\delta_2^*((q,y),\alpha'),a) = \delta_2((\delta_1^*(q,\alpha'), \beta_1^*(q,\alpha')),a) = \\ &= (\delta_1(\delta_1^*(q,\alpha'),a), \beta_1(\delta_1^*(q,\alpha'),a)) = (\delta_1^*(q,a), \beta_1^*(q,a)) \end{aligned}$$

Tím jsme dokázali vztah (9) a můžeme přikročit k důkazu věty (tedy rovnice (8)). Postupným použitím definice β^*, vztahu (9) a předpokladu (7) dostáváme

$$\beta_2^*((q,y),\alpha) = \beta_2(\delta_2^*((q,y),\alpha)) = \beta_2(\delta_1^*(q,\alpha), \beta_1^*(q,\alpha)) = \beta_1^*(q,\alpha)$$

Tím je dokázána rovnost (8), zvolená relace $\overset{\Lambda}{\sim}$ je opravdu Λ-ekvivalence. Podle definice $\overset{\Lambda}{\sim}$ pro každý stav Mealyho automatu existuje alespoň jeden stav Mooreova automatu s ním Λ-ekvivalentní a naopak, tedy automaty popsané v předpokladech věty mají stejné Λ-chování. CBD

Uvedené věty nám říkají nejen, že lze Mooreův automat převést na Mealyho (a naopak) se stejným Λ-chováním, ale nadto poskytují přesný návod, jak to máme provést. Neboť teď už umíme převést jeden automat na druhý, platí definice, věty a tvrzení, vyslovená a dokázaná pro jeden typ automatů, okamžitě i pro druhý[12].

Od této chvíle, pokud nebude řečeno jinak, budeme pracovat jenom s Mooreovými automaty s tím, že předchozí odstavec si podržíme stále v paměti.

[12] až na to, že Λ na vstupu Mealyho automatu nemá smysl(!), což je ale triviálně ošetřitelný problém. Nicméně je třeba si uvědomit, že kdykoliv používáme $\beta^*(q,\alpha)$ bez určení, zda se jedná o Mealyho nebo Mooreův automat, a píšeme $\alpha \in X^*$, mělo by být pod čarou poznamenáno, že pro Mealyho je $\alpha \in X^+$. Vzhledem k tomuto odstavci to nadále považuji za samozřejmost a poznámky pod čarou na účet β^* v Mealyho automatech si s laskavým dovolením odpustím.

Příklad 2.8 *(Převedení Moore→Mealy) Je dán automat M^o ze strany 7. Sestrojte Mealyho automat se stejným Λ-chováním jako M^o.*

Věta 2.6 nám říká, jak takový automat máme sestrojit. Mealyho automat $M_1^e = (Q, X, \delta, Y, \beta_1)$ se bude od M^o lišit pouze výstupní funkcí, kterou definujeme pomocí výstupní funkce β předpisem (5). Výsledek je zřejmý z tabulky.

	M_1^e	a	b	c	d
⇒	q_1	$q_1/1$	$q_5/0$	$q_2/0$	$q_4/1$
	q_2	$q_5/0$	$q_2/0$	$q_1/1$	$q_3/1$
	q_3	$q_1/1$	$q_1/1$	$q_2/0$	$q_1/1$
	q_4	$q_3/1$	$q_2/0$	$q_5/0$	$q_4/1$
	q_5	$q_2/0$	$q_4/1$	$q_3/1$	$q_2/0$

FINE[13]

Příklad 2.9 *(Převedení Mealy→Moore) Buď $M_2^e = (Q_2, X_2, \delta_2, Y_2, \beta_2)$ Mealyho automat zadaný tabulkou (viz níže). Sestrojte Mooreův automat se stejným Λ-chováním.*

Z věty 2.7 víme, jak takový automat sestrojit. Definujme Mooreův automat $M_3^o = (Q_2 \times Y_2, X_2, \delta_3, Y_2, \beta_3)$. Jeho stavy budou stavy uspořádané dvojice z kartézského součinu $Q_2 \times Y_2$, přechodová funkce δ_3 bude definována vztahem (6) a výstupní funkce β_3 dána rovnicí (7). Následují tabulky automatů M_2^e a M_3^o.

M_2^e	x	y
1	1/b	2/a
2	2/a	1/b

M_3^o	x	y	
(1,a)	(1,b)	(2,a)	a
(1,b)	(1,b)	(2,a)	b
(2,a)	(2,a)	(1,b)	a
(2,b)	(2,a)	(1,b)	b

FINE

3 Propojování a dekompozice automatů

3.1 Automatová a stavová kongruence

Definice 3.1 *Řekneme, že Mooreův automat je redukovaný, pokud žádné dva různé stavy nejsou ekvivalentní.*

Definice 3.2 *Ekvivalenci $\approx$ na množině Q nazveme automatovou kongruencí automatu (Q, X, δ, Y, β), pokud platí*

$$(\forall\, a \in X)(\forall\, q, q' \in Q)(q \approx q' \;\Rightarrow\; \delta(q,a) \approx \delta(q',a)) \tag{10}$$

$$(\forall\, q, q' \in Q)(q \approx q' \;\Rightarrow\; \beta(q) = \beta(q')) \tag{11}$$

Pokud platí alespoň podmínka (10)*, hovoříme o stavové kongruenci*[1]

[13] konec příkladu

[1] je důležité si uvědomit, že automatová kongruence je ekvivalence definovaná na množině Q jednoho automatu, kdežto ekvivalence popsaná v definici 2.4 je na množinách Q_1 a Q_2 dvou automatů (mohou být různé)

O automatových a stavových kongruencích si řekneme něco bližšího až po objasnění algoritmu na jejich hledání (algoritmus 1).

Algoritmus 1 *(Konstrukce automatových a stavových kongruencí)*

Mějme automat (Q, X, δ, Y, β).

1. Hledání nejhrubší[2] automatové kongruence. Na automatu (Q, X, δ, Y, β) definujeme ekvivalence $\overset{i}{\sim}$ pro $i \in N_0$:

$$(\forall\, q, q' \in Q)\Big(q \overset{i+1}{\sim} q' \overset{df}{\equiv} (q \overset{i}{\sim} q') \wedge (\forall\, x \in X)(\delta(q, x) \overset{i}{\sim} \delta(q', x))\Big)$$

   Inicializační krok $\overset{0}{\sim}$ se pro Mooreův a Mealyho automat liší:

   - $q \overset{0}{\sim} q' \overset{df}{\equiv} (\beta(q) = \beta(q'))$ pro Mooreův automat
   - $q \overset{0}{\sim} q' \overset{df}{\equiv} (\forall\, x \in X)(\beta(q, x) = \beta(q', x))$ pro Mealyho automat

   Jinak řečeno, stavy q a q' jsou ekvivalentní v $\overset{i}{\sim}$, pokud výpočty[3] v nich započaté mají pro všechna slova délky menší nebo rovné i stejné výstupy.

   Algoritmus končí, jakmile $\overset{n}{\sim} = \overset{n-1}{\sim}$.

   Algoritmus je deterministický a vychází přímo z definice automatové kongruence. Lze nahlédnout, že pokud se $\overset{n}{\sim}$ dále nerozpadá, je nejhrubší automatovou kongruencí daného automatu. Nejjemnější je zřejmě id_Q. Všechny ostatní automatové kongruence získáme zjemněním $\overset{n}{\sim}$, tzn. rozpadem některých tříd $\overset{n}{\sim}$ na menší. Přitom je však nutné dbát na to, aby získaná relace byla opět automatová kongruence (tj. splňovala podmínku (10))!

2. Hledání elementárních stavových kongruencí. Stavová kongruence $\sim$ je elementární, pokud existují dva různé stavy q a q' takové, že $\sim$ je nejjemnější[4] kongruence, pro kterou platí $q \sim q'$. Elementární stavové kongruence dostáváme opačným postupem než nejhrubší automatovou kongruenci. Vyjdeme od ekvivalence lepící právě dva prvky[5] a postupně vytváříme ekvivalence hrubší a hrubší. Definujme opět ekvivalenci

$$\overset{0}{\sim} = \{T_1^0, T_2^0, \ldots, T_{|Q|-1}^0\},$$

   kde T_i^0 jsou ekvivalenční třídy a dále platí $|T_1^0| = 2$, tzn. třída T_1^0 lepí právě dva prvky[6] a ostatní třídy obsahují po jednom prvku (stavu z Q). Nyní indukcí

$$q \overset{i+1}{\sim} q' \overset{df}{\equiv} (\exists\, p, p' \in Q)(\exists\, a \in X)(p \overset{i}{\sim} p' \wedge \delta(p, a) = q \wedge \delta(p', a) = q')$$

[2] tzn. má nejméně kongruenčních tříd

[3] výpočtem nad slovem $a_1 \ldots a_n$ započatém ve stavu q_1 mám na mysli posloupnost stavů $q_1, \ldots, q_{n+1}$, která splňuje $\delta(q_i, a_i) = q_{i+1}$ pro všechna $1 \leq i \leq n$. Myslím, že pojem výpočtu je zde intuitivně zřejmý a považuji za zbytečné věnovat mu zvláštní definici.

[4] tzn. má nejvíce kongruenčních tříd

[5] Definice: ekvivalence ρ lepí prvky x_1 a x_2, právě když $x_1 \rho x_2$

[6] vybíráme libovolné dvojice stavů do T_1^0 a podle toho dostáváme různé stavové kongruence

Jinými slovy vezmeme všechny stavy z jedné třídy a zjistíme přechody funkce δ ze všech těchto stavů pro daný vstup. Stavy, získané jako výsledky této operace, prohlásíme za ekvivalentní.

Algoritmus končí, jakmile $\overset{n}{\sim}=\overset{n-1}{\sim}$. Ekvivalence $\overset{n}{\sim}$ je stavovou kongruencí.

Praktické použití algoritmu je předvedeno v příkladu 3.18.

Nebojíme-li se ponořit do algebraických výrazů, můžeme označit K množinu všech stavových kongruencí a K' množinu všech automatových kongruencí na daném automatu a na K definovat operace spojení ($\cup$) a průsek ($\cap$) vztahy (pro $\pi, \tau \in K$)

$$\begin{aligned}(q,q') \in (\pi \cap \tau) &\Leftrightarrow (q,q') \in \pi \wedge (q,q') \in \tau \\ (q,q') \in (\pi \cup \tau) &\Leftrightarrow (\exists\, q_1, \ldots, q_n, q_1 = q, q_n = q') \\ &\quad (\forall\, i \in \langle 1, n-1\rangle \subset N)(q_i \pi q_{i+1} \vee q_i \tau q_{i+1})\end{aligned}$$

Zájemci si mohou dokázat, že $K(\cup,\cap)$ je svaz a $K'(\cup,\cap)$ je jeho podsvaz. První tvrzení obnáší ověřit pro $\cap$ a $\cup$ komutativitu, asociativitu, absorbci a idempotenci[7]. Druhé tvrzení také není žádná věda. Každá automatová kongruence je z definice i stavová, tedy $K' \subseteq K$. Dále každá automatová kongruence je zjemněním nejhrubší automatové kongruence získané algoritmem 1, tzn. K' je uzavřen na operace a je tudíž podsvazem K. Označíme-li nejhrubší automatovou kongruenci $\overset{max}{\sim}$, potom $K'(\cup,\cap,id_Q,\overset{max}{\sim})$ je dokonce svaz s nejmenším a největším prvkem. Navíc každý prvek $\sim$ svazu K je spojením elementárních stavových kongruencí.

3.2 Podílový automat

Definice 3.3 *Mějme automat $M = (Q, X, \delta, Y, \beta)$ a na něm automatovou kongruenci $\approx$. Podílovým automatem automatu M rozumíme automat*

$$M/_\approx = (Q, X, \delta, Y, \beta)/_\approx = (Q/_\approx, X, \delta', Y, \beta'),$$

kde definujeme:

$$\begin{aligned}(12)\quad \delta'([q]_\approx, a) &= [\delta(q,a)]_\approx \\ (13)\quad \beta'([q]_\approx) &= \beta(q)\end{aligned}$$

Nezávislost této definice na volbě reprezentanta třídy kongruence $\approx$ (stav q) je třeba ověřit: Vezměme dva stavy $q_1, q_2 \in Q$ takové, že $q_1 \approx q_2$ a ověřme podmínky (12) a (13):

Pro podmínku (12) dostáváme definice automatové kongruence (10) a ekvivalence stavů q_1 a q_2

$$\delta'([q_1]_\approx, a) = [\delta(q_1,a)]_\approx = [\delta(q_2,a)]_\approx = \delta'([q_2]_\approx, a),$$

tedy $\delta'([q_1]_\approx, a) \approx \delta'([q_2]_\approx, a)$ pro všechna $a \in X$.

Pro podmínku (13) obdobně aplikací (11) a ekvivalence q_1 a q_2

$$\beta'([q_1]_\approx) = \beta(q_1) = \beta(q_2) = \beta'([q_2]_\approx)$$

Čímž je korektnost definice δ' a β' prokázána.

[7] Všechny čtyři vlastnosti lze ověřit přímo z definice.

Tvrzení 3.4 *Ekvivalence z definice 2.4 na automatu* (Q, X, δ, Y, β) *je automatová kongruence.*

Důkaz: Vezměme dva stavy $q_1, q_2 \in Q$ takové, že $q_1 \sim q_2$. Pro $\sim$ dokážeme nejprve podmínku (10) z definice automatové kongruence. Užitím druhého řádku tvrzení 2.2[8], ekvivalence $q_1 \sim q_2$ a opět tvrzení 2.2 dostáváme pro všechna $a \in X, \alpha \in X^*$

$$\beta^*(\delta(q_1, a), \alpha) = \beta^*(q_1, a\alpha) = \beta^*(q_2, a\alpha) = \beta^*(\delta(q_2, a), \alpha),$$

a tedy dle definice 2.4 je $\delta(q_1, a) \sim \delta(q_2, a)$.

Podmínka (11) z definice automatové kongruence vyplývá z definice β^* a ekvivalence stavů q_1 a q_2:

$$\beta(q_1) = \beta^*(q_1, \Lambda) = \beta^*(q_2, \Lambda) = \beta(q_2)$$

Dokázali jsme, že ekvivalence $\sim$, jak jsme ji definovali na začátku kapitoly v definici 2.4, splňuje všechny požadavky, aby mohla být považována za automatovou kongruenci[9]. CBD

Do konce kapitoly 3.2 považujme $\sim$ za ekvivalenci stavů automatu (viz definice 2.4).

Lemma 3.5 *Pro podílový automat* $(Q', X, \delta', Y, \beta') = (Q, X, \delta, Y, \beta)/_\sim$ *platí*

$$(14) \quad (\forall\, q \in Q)(\forall\, \alpha \in X^*)((\delta')^*([q]_\sim, \alpha) = [\delta^*(q, \alpha)]_\sim)$$

Důkaz: Provedeme indukcí podle délky α. Pro $\alpha = \Lambda$ dostáváme

$$[\delta^*(q, \Lambda)]_\sim = [q]_\sim = (\delta')^*([q]_\sim, \Lambda)$$

pro všechna $q \in Q$. Pokud $\alpha \neq \Lambda$, platí pro všechna $q \in Q$ a $\alpha = \alpha' a \in X^+$ z definice δ^*, δ', indukčního předpokladu a definice δ^*

$$\begin{aligned} [\delta^*(q, \alpha)]_\sim &= [\delta(\delta^*(q, \alpha'), a)]_\sim = \delta'([\delta^*(q, \alpha')]_\sim, a) = \\ &= \delta'((\delta')^*([q]_\sim, \alpha'), a) = (\delta')^*([q]_\sim, \alpha) \end{aligned}$$

CBD

Věta 3.6 *(O podílovém automatu) Podílový automat* $M/_\sim$ *je redukovaný a má stejné chování jako automat* M.

[8] mám na mysli řádek

$$\beta^*(q, \alpha\gamma) = \beta^*(\delta^*(q, \alpha), \gamma),$$

resp. speciální případ

$$\beta^*(q, a\alpha) = \beta^*(\delta(q, a), \alpha)$$

[9] s tím, že porovnáváme stavy jednoho automatu, kdežto definice ekvivalence stavů pracuje s automaty dvěma

Důkaz: Nechť $M/_\sim = (Q', X, \delta', Y, \beta')$ a $M = (Q, X, \delta, Y, \beta)$. Nejprve ukážeme, že $M/_\sim$ má stejné chování jako M, tzn. že odpovídající stavy q a $[q]_\sim$ jsou ekvivalentní, což okamžitě pro všechna $q \in Q, \alpha \in X^*$ dostáváme z definic β^*, β' a δ' a lemmatu 3.5

$$(15) \quad \begin{aligned} (\beta')^*([q]_\sim, \alpha) &= \beta'((\delta')^*([q]_\sim, \alpha)) = \beta'([\delta^*(q,\alpha)]_\sim) = \\ &= \beta(\delta^*(q,\alpha)) = \beta^*(q,\alpha) \end{aligned}$$

Nyní dokážeme, že M' je redukovaný. Buďte $[q_1]_\sim$, $[q_2]_\sim$ ekvivalentní stavy v $M/_\sim$. Potom podle (15), definice 2.4 a opět (15) pro všechna $\alpha \in X^*$

$$\beta^*(q_1, \alpha) = (\beta')^*([q_1]_\sim, \alpha) = (\beta')^*([q_2]_\sim, \alpha) = \beta^*(q_2, \alpha),$$

tedy $q_1 \sim q_2$, což dává $[q_1]_\sim = [q_2]_\sim$. CBD

Definice 3.7 *Mějme dva automaty $M_i = (Q_i, X_i, \delta_i, Y_i, \beta_i)$, $i = 1, 2$. Pak trojice*

$$(f : X_1 \to X_2, g : Q_1 \to Q_2, h : Y_1 \to Y_2)$$

se nazývá homomorfismus[10] *automatů z M_1 do M_2, pokud pro všechna $q \in Q_1$, $a \in X_1$ platí*

$$(16) \quad \delta_2(g(q), f(a)) = g(\delta_1(q, a))$$
$$(17) \quad \beta_2(g(q)) = h(\beta_1(q))$$

Věta 3.8 *Když automaty $M_i = (Q_i, X, \delta_i, Y, \beta_i)$, $i = 1, 2$ mají stejné chování a automat M_2 je redukovaný, pak existuje homomorfismus automatů z M_1 do M_2*[11]

$$(id_X, g : Q_1 \to Q_2, id_Y)$$

takový, že g je surjektivní (na).

Důkaz: Definujme $g : Q_1 \to Q_2$ vztahem

$$(\forall\, q \in Q_1)(g(q) \overset{df}{\sim} q) \qquad \text{CBD}$$

Věta 3.9 *Máme-li Mooreův automat $M = (Q, X, \delta, Y, \beta)$, pak existuje Mooreův automat M' se stejným chováním, který má nejmenší počet stavů mezi automaty, které mají stejné chování jako M*[12]*. Pro automat M' dále platí, že je*

[10] Pro zájemce tu máme exkurzi do algebry: homomorfismus dvou algeber $A(\alpha_1, \ldots, \alpha_n)$, $B(\beta_1, \ldots, \beta_n)$ je jakékoliv zobrazení $f : A \to B$, které je slučitelné se všemi operacemi algeber A, B, tzn. platí

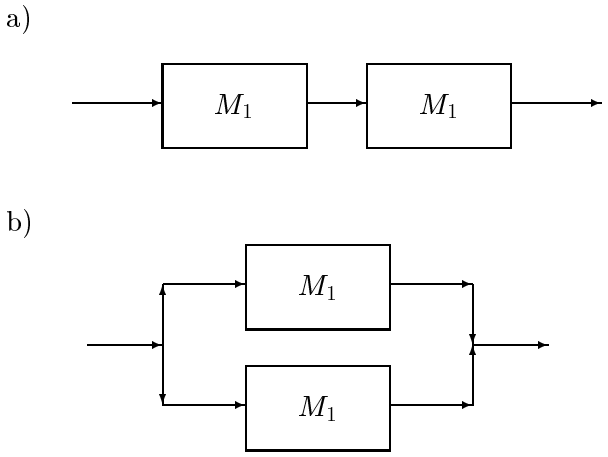
$$(\forall\, i \in \langle 1; n\rangle)(\forall\, x_1, \ldots, x_m \in A)(f(\alpha_i(x_1, \ldots, x_m)) = \beta_i(f(x_1), \ldots, f(x_m))),$$

kde m je arita operací α_i, β_i.

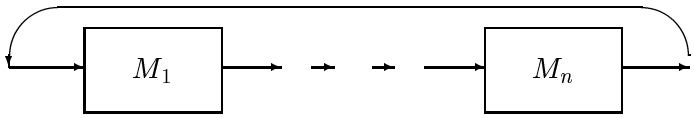
V našem případě jsou algebrami automaty, které ovšem nemají pouze jednu nosnou množinu, ale trojici množin X, Q, Y. Je tedy potřeba přesně určit, co si představujeme pod pojmem slučitelnost s operacemi δ, β (viz rovnice (16),(17)).

[11] zápis id_X značí identickou funkci na množině X, tzn. $(\forall\, x \in X)(x = id_X(x))$; srovnejte s definicí identické relace

[12] což je poměrně pochopitelné



Obr. 2: a) sériové, b) paralelní zapojení automatů



Obr. 3: zpětná vazba nastává, pokud se v kaskádním spojení automatů vytvoří (orientovaná) kružnice

- *jednoznačně určen až na izomorfismus*[13]
- *podílovým automatem automatu M podle kongruence $\sim$*[14]

3.3 Sériová a paralelní dekompozice

V této kapitole ukážeme, jak z jednodušších automatů vytvářet složitější. K tomuto účelu budeme propojovat Mealyho automaty.

Definice 3.10 *Kaskádní spojení automatů je síť tvořená sériovými a paralelními spojeními automatů (viz obr. 2) bez zpětné vazby (viz obr. 3).*

[13] dvě algebry (automaty) M_1, M_2 jsou z definice izomorfní, jakmile existuje homomorfismus (automatů) z M_1 do M_2, který je prostý a na (injektivní a surjektivní)

[14] ekvivalence stavů — viz definici 2.4

Definice 3.11 *Mějme dva automaty $M_i = (Q_i, X_i, \delta_i, Y_i, \beta_i)$, $i = 1, 2$ takové, že $Y_1 \subseteq X_2$. Pak sériovým spojením automatů M_1, M_2 rozumíme automat*

$$M = M_1(\Longrightarrow)M_2 = (Q_1 \times Q_2, X_1, \delta, Y_2, \beta),$$

kde pro všechna $a \in X_1, q_1 \in Q_1$ a $q_2 \in Q_2$

$$(18)\quad \delta((q_1, q_2), a) = (\delta_1(q_1, a), \delta_2(q_2, \beta_1(q_1, a)))$$
$$(19)\quad \beta((q_1, q_2), a) = \beta_2(q_2, \beta_1(q_1, a))$$

Definice 3.12 *Řekneme, že automat M_1 realizuje automat M_2, pokud existují prostá zobrazení $f : X_2 \to X_1$ a $g : Q_2 \to Q_1$ a zobrazení $h : Y_1 \to Y_2$ taková, že pro všechna $a \in X_2, q \in Q_2$*

$$(20)\quad \delta_1(g(q), f(a)) = g(\delta_2(q, a))$$
$$(21)\quad \beta_2(q, a) = h(\beta_1(g(q), f(a)))$$

Definice 3.13 *Řekneme, že automat M má sériovou dekompozici, když existují automaty M_1 a M_2 takové, že $M_1(\Longrightarrow)M_2$ realizuje M a počet stavů každého z automatů M_1 a M_2 je menší než počet stavů M.*

Definice 3.14 *Mějme dva automaty $M_i = (Q_i, X_i, \delta_i, Y_i, \beta_i)$, $i = 1, 2$ takové, že $X_1 = X_2 = X$. Pak paralelním spojením automatů M_1, M_2 rozumíme automat $M = (M_1 \parallel M_2) = (Q_1 \times Q_2, X, \delta, Y_1 \times Y_2, \beta)$, kde pro všechna $a \in X_1$,$q_1 \in Q_1$ a $q_2 \in Q_2$*

$$(22)\quad \delta((q_1, q_2), a) = (\delta_1(q_1, a), \delta_2(q_2, a))$$
$$(23)\quad \beta((q_1, q_2), a) = (\beta_1(q_1, a), \beta_2(q_2, a))$$

Definice 3.15 *Automat M má paralelní dekompozici, pokud existují automaty M_1, M_2 takové, že $M_1 \parallel M_2$ realizuje M a počet stavů každého z automatů M_1 a M_2 je menší než počet stavů M.*

Poznámka: Pro následující větu je důležité si osvětlit, co je chápáno jako triviální ekvivalence. Triviální ekvivalencí na množině o n prvcích je myšlena ekvivalence o jedné nebo o n třídách, tzn. buď jsou ekvivalentní všechny prvky, nebo žádné dva různé prvky. Tentýž pojem budeme ve stejném smyslu používat i pro kongruenci.

Věta 3.16 *(Sériová dekompozice) Automat M má sériovou dekompozici, právě když má netriviální stavovou kongruenci.*

Důkaz: Dokážeme nejprve implikaci zleva doprava. Předpokládejme, že M má sériovou dekompozici $M' = M_1(\Longrightarrow)M_2$[15], tedy existují f, g, h taková, že platí (20) a (21) (neboť M' realizuje M), a dále platí (18), neboť M' je sériové spojení[16]. Definujme ekvivalenci $\rho \subseteq Q \times Q$ následovně:

$$(24)\quad (\forall\, q_i \in Q)(q_1 \rho q_2 \overset{df}{\equiv} (r_1 = r_2))$$

[15] $M' = (Q_1 \times Q_2, X_1, \delta', Y_2, \beta')$, $M_i = (Q_i, X_i, \delta_i, Y_i, \beta_i)$, $M = (Q, X, \delta, Y, \beta)$

[16] samozřejmě platí i (19), ale to zde nebudeme potřebovat

kde $g(q_i) = (r_i, s_i)$, $i = 1, 2$) (neboli stavy q_1 a q_2 jsou v ρ, jakmile se první složky $g(q_1)$ a $g(q_2)$ rovnají). O relaci ρ ukážeme, že je netriviální stavovou kongruencí, tzn. ověříme podmínku (10) a potom netriviálnost ρ. Jinými slovy, musí platit

$$q_1 \rho q_2 \Rightarrow (\forall\, a \in X)(\delta(q_1, a) \rho \delta(q_2, a)),$$

tedy

$$\begin{aligned} g(\delta(q_1, a)) &= (r_1', s_1') \\ g(\delta(q_2, a)) &= (r_2', s_2') \\ r_1' &= r_2' \end{aligned}$$

Jak na to? Zkusme vyjádřit $g(\delta(q_i, a))$. Z (20) a (18) dostáváme

$$\begin{aligned} g(\delta(q_1, a)) &= \delta'(g(q_1), f(a)) = \Big(\delta_1(r_1, f(a)), \delta_2(s_1, \beta_1(r_1, f(a)))\Big) \\ &= (r_1', s_1') \\ g(\delta(q_2, a)) &= \delta'(g(q_2), f(a)) = \Big(\delta_1(r_2, f(a)), \delta_2(s_2, \beta_1(r_2, f(a)))\Big) \\ &= (r_2', s_2'), \end{aligned}$$

z čehož je vidět, že $(r_1 = r_2) \Rightarrow (r_1' = r_2')$, tedy $(q_1 \rho q_2 \Rightarrow \delta(q_1, a) \rho \delta(q_2, a))$ pro každé $a \in X$ a proto ρ je stavová kongruence. Protože dva stavy $q_1, q_2 \in Q$ jsou v ρ, právě když se první složky funkce $g : Q \to Q_1 \times Q_2$ na těchto stavech rovnají, vytváří ρ nejvýše tolik ekvivalenčních tříd, kolik je stavů v Q_1. Protože podle definice sériové dekompozice $|Q_1| < |Q|$, má ρ méně tříd než $|Q|$. Kdyby ρ měla jedinou třídu, ve všech $g(q)$ by musela být stejná první složka. Protože g je prostá, musela by nutně platit první nerovnost ze vztahu

$$(25) \quad |Q| \leq |\{s; (\exists q \in Q)(g(q) = (r, s))\}| \leq |Q_2| \ .$$

Druhá nerovnost je dána faktem, že druhé složky $g(q)$ bereme z Q_2. Podle definice sériové dekompozice platí $|Q_2| < |Q|$, což je ale ve sporu s nerovnostmi (25) a tím i s předpokladem triviality ρ. ρ je tedy netriviální.

Ale teď ...[17]

Nechť nyní M má netriviální stavovou kongruenci ρ. Označíme-li k maximální velikost tříd kongruence ρ, je $k < |Q|$. Předpokládejme, že pro každou třídu V jsou její prvky seřazeny, tedy $V = (q_{1,V}, q_{2,V}, \ldots, q_{|V|,V})$. Definujme automaty M_1 a M_2 následovně:[18] $M_1 = (Q/_\rho, X, \delta_1, Y_1, \beta_1)$, nechť je „podílovým" automatem[19] automatu M s tím, že $Y_1 = Q/_\rho \times X$ a pro každé $V \in Q/_\rho$ a $a \in X$ je $\beta_1(V, a) = (V, a)$. Dále $M_2 = (Q_2, Y_1, \delta_2, Y, \beta_2)$, kde

$$Q_2 = \{1, 2, \ldots, k\},$$

[17] druhá implikace

[18] Idea definice M_1, M_2 není nepochopitelná. Oba automaty kódují ve svých stavech stav z M, přičemž M_1 uchovává třídu ekvivalence ρ, ve které se stav z M vyskytuje, a M_2 kóduje jeho pořadí v této třídě.

[19] spíš je podílovému automatu představou blízký, protože ρ není automatová, ale jen stavová kongruence

$$\delta_2(i,(V,a)) = \begin{cases} i & \text{pro } i > |V| \\ j & \text{když } \delta(q_{i,V},a) = q_{j,\delta_1(V,a)} \text{ pro } i \leq |V| \end{cases}$$

$$\beta_2(i,(V,a)) = \begin{cases} \text{lib. prvek} & \text{pro } i > |V| \\ \beta(q_{i,V},a) & \text{pro } i \leq |V| \end{cases}$$

Chceme ukázat, že $M_1(\Longrightarrow)M_2 = (Q_1 \times Q_2, X, \delta', Y, \beta')$ je sériovou dekompozicí M, tedy v první řadě, že vůbec M realizuje (tzn. najít prostá zobrazení $f : X \to X$ a $g : Q \to Q_1 \times Q_2$ a zobrazení $h : Y \to Y$ takové, aby platily vztahy (20) a (21) — viz obrázek 4). Položme $h = id_Y$, $f = id_X$ a $g(q_{i,V}) = (V,i)$ pro všechna $q_{i,V} \in Q$, kde V je rozkladová třída ρ, obsahující stav $q_{i,V}$ na pozici i. Důkaz vztahu (21) plyne z definice β_2, β_1, definice sériového spojení a definice zobrazení g

$$\beta(q_{i,V},a) = \beta_2(i,(V,a)) = \beta_2(i,\beta_1(V,a)) = \beta'((V,i),a) = \beta'(g(q),f(a)).$$

Rovnici (20) dostaneme pro $a \in X, q_{i,V} \in Q$ a $\delta(q_{i,V},a) = q_{j,U}$ porovnáním složek výrazů[20]

$$\begin{aligned} \delta'(g(q_{i,V}),a) = \delta'((V,i),a) &= (\delta_1(V,a), \delta_2(i,(V,a))) \\ g(\delta(q_{i,V},a)) &= (U,j) \end{aligned}$$

Z definice δ_1 platí $\delta_1(V,a) = [\delta(q_{i,V},a)]_\rho = U$, tedy první složky výrazů na pravých stranách se rovnají. Druhé složky se rovnají přímo z definice δ_2, neboť $q_{i,V}$ je i-tý prvek ve V a $\delta(q_{i,V},a)$ je j-tý prvek v U, tedy $\delta_2(i,(V,a)) = j$. Z definice M_1 a z faktu, že ρ je netriviální, plyne $|Q_1| < |Q|$. Protože $k = |Q_2|$ a ρ je netriviální, víme $|Q_2| < |Q|$, tedy $M_1(\Longrightarrow)M_2$ je sériovou dekompozicí M.[21] CBD

Věta 3.17 *(Paralelní dekompozice) Automat M má paralelní dekompozici právě tehdy, když existují dvě netriviální stavové kongruence ρ_1, ρ_2 takové, že pro každé dva různé stavy $q, q' \in Q$ platí buď $(q_1,q_2) \notin \rho_1$ nebo $(q_1,q_2) \notin \rho_2$*[22].

Důkaz: Dokažme implikaci zleva doprava. Předpokládejme, že $M' = M_1 \parallel M_2$ realizuje M[23] a zobrazení $f : X \to X_1$, $g : Q \to Q_1 \times Q_2$, $h : Y_1 \times Y_2 \to Y$ (f,g prostá) tuto realizaci určují. Definujme ekvivalence ρ_1 a ρ_2. Pro každé dva stavy $q_1, q_2 \in Q$ označme $g(q_1) = (r_1,s_1)$ a $g(q_2) = (r_2,s_2)$. Pak

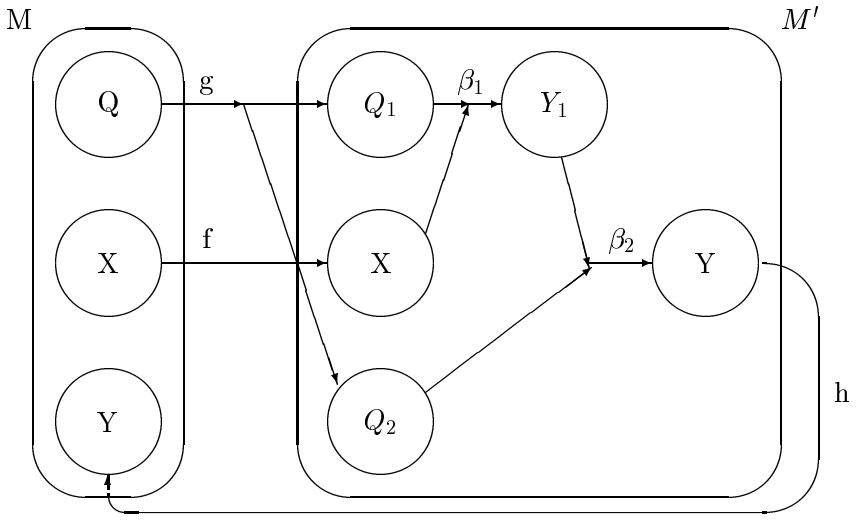
$$(26) \quad (q_1,q_2) \in \rho_1 \overset{df}{\equiv} r_1 = r_2$$

$$(27) \quad (q_1,q_2) \in \rho_2 \overset{df}{\equiv} s_1 = s_2$$

Z definic ρ_1, ρ_2 okamžitě vidíme, že obě relace jsou symetrické, reflexivní i tranzitivní a jsou tudíž ekvivalencemi. Dokážeme, že ρ_1 a ρ_2 jsou netriviální stavové kongruence a $\rho_1 \cap \rho_2 = id_Q$. Vezměme nějaké $q_1, q_2 \in Q$. Nechť

$$\begin{aligned} g(q_1) &= (r_1,s_1) \\ g(q_2) &= (r_2,s_2) \end{aligned}$$

[20] první výraz lze odvodit z definice g a definice sériového spojení, druhý je triviální
[21] zde je dobré si povšimnout, že M_1 i M_2 mají nejméně dva stavy
[22] nebo obojí
[23] $M' = (Q_1 \times Q_2, X, \delta', Y_1 \times Y_2, \beta')$, $M_i = (Q_i, X, \delta_i, Y_i, \beta_i)$, $M = (Q, X, \delta, Y, \beta)$



Obr. 4: Schéma realizujících funkcí sériové dekompozice

Z definice realizace a paralelního spojení platí

$$g(\delta(q_1,a)) = \delta'(g(q_1),f(a)) = (\delta_1(r_1,f(a)),\delta_2(s_1,f(a))) = (r_1',s_1')$$
$$g(\delta(q_2,a)) = \delta'(g(q_2),f(a)) = (\delta_1(r_2,f(a)),\delta_2(s_2,f(a))) = (r_2',s_2')$$

Z uvedených rovností podle definice ρ_1 a ρ_2 vyplývá

1. $(q_1,q_2) \in \rho_1 \Rightarrow (r_1 = r_2) \Rightarrow (r_1' = r_2') \Rightarrow (\delta(q_1,a),\delta(q_2,a)) \in \rho_1$, tedy ρ_1 je stavová kongruence

2. $(q_1,q_2) \in \rho_2 \Rightarrow (s_1 = s_2) \Rightarrow (s_1' = s_2') \Rightarrow (\delta(q_1,a),\delta(q_2,a)) \in \rho_2$, tzn. ρ_2 je stavová kongruence

Všimněme si nejprve, že ρ_1 má nejvýše $|Q_1|$ tříd a ρ_2 má nejvýše $|Q_2|$ tříd. Protože M' je paralelní dekompozice, platí $|Q_1|<|Q|$ a $|Q_2|<|Q|$, tedy ρ_1 a ρ_2 nejsou identické kongruence. Dokážeme sporem, že ρ_1 má více než jednu třídu. Kdyby ρ_1 lepila všechny prvky Q, pak odstraněním „zbytečných“ prvků dostaneme $|Q_1|= 1$, a tedy $|Q_1 \times Q_2|=|Q_2|$. Protože ale g je prosté, muselo by platit $|Q_2|\geq|Q|$, což je spor s požadavky paralelní dekompozice. Obdobně se ukáže, ře ρ_2 nelepí všechny prvky Q. Ekvivalence ρ_1 a ρ_2 jsou tedy netriviální stavové kongruence. Nyní je ještě třeba ověřit, že $\rho_1 \cap \rho_2 = id_Q$, tzn. dokázat implikaci

$$(\forall\, q_1,q_2 \in Q)((q_1,q_2) \in \rho_1 \wedge (q_1,q_2) \in \rho_2 \Rightarrow q_1 = q_2)$$

Vezměme libovolné q_1, q_2, nechť $g(q_1) = (r_1,s_1)$ a $g(q_2) = (r_2,s_2)$. Pak

$$\left.\begin{array}{lcl} q_1\rho_1 q_2 & \Rightarrow & r_1 = r_2 \\ q_1\rho_2 q_2 & \Rightarrow & s_1 = s_2 \end{array}\right\} \Rightarrow (g(q_1) = g(q_2)) \Rightarrow (q_1 = q_2),$$

přičemž poslední implikace plyne z předpokladu, že g je prosté.

Dokázali jsme, že relace ρ_1 a ρ_2, jak jsme je definovali na začátku této části důkazu, vyhovují podmínkám věty.

Nyní dokážeme druhou implikaci. Předpokládejme, že ρ_1 a ρ_2 jsou netriviální stavové kongruence automatu M, pro které navíc platí

(28) $\rho_1 \cap \rho_2 = id_Q$

Budeme definovat automaty M_1 a M_2 a zobrazení f, g, h a ukážeme, že

$$M' = M_1 \parallel M_2 = (Q_1 \times Q_2, X, \delta', Y_1 \times Y_2, \beta')$$

pomocí f, g, h[24] paralelně dekomponuje M. Definujme automaty

$$M_1 = (Q_1, X, \delta_1, Y_1, \beta_1) \text{ a } M_2 = (Q_2, X, \delta_2, Y_2, \beta_2)$$

následujícím předpisem

$$\begin{array}{rclrcl} Q_1 & = & Q/_{\rho_1} & Q_2 & = & Q/_{\rho_2} \\ \delta_1([q]_{\rho_1}, a) & = & [\delta(q,a)]_{\rho_1} & \delta_2([q]_{\rho_2}, a) & = & [\delta(q,a)]_{\rho_2} \\ Y_1 & = & Q_1 \times X & Y_2 & = & Q_2 \times X \\ \beta_1 & = & id_{Y_1} & \beta_2 & = & id_{Y_2} \end{array}$$

Představou jsou M_1, M_2 blízké podílovým automatům automatu M podle ρ_1 a ρ_2. Dále definujme zobrazení $h : Y_1 \times Y_2 \to Y$, $f : X \to X$, $g : Q \to Q_1 \times Q_2$ takto:

- $f = id_X$
- $g(q) = ([q]_{\rho_1}, [q]_{\rho_2})$
- zvolme libovolné $y_0 \in Y$

$$h(([q_1]_{\rho_1}, x_1), ([q_2]_{\rho_2}, x_2)) = \begin{cases} \beta(q, x_1) \text{ pro } q \in [q_1]_{\rho_1} \cap [q_2]_{\rho_2} \text{ a } x_1 = x_2 \\ y_0 \text{ pro } [q_1]_{\rho_1} \cap [q_2]_{\rho_2} = \emptyset \text{ nebo } x_1 \neq x_2 \end{cases}$$

Přechody funkcí f, g, h jsou schématicky znázorněny na obrázku 5.

Ze všeho nejdříve musíme ověřit, že h je zobrazení a f, g jsou prostá zobrazení. Pro daná q_1, q_2 je h určeno jednoznačně (a tedy korektně definováno), neboť z (28) plyne $|\, [q_1]_{\rho 1} \cap [q_2]_{\rho_2} \,| \leq 1$ pro libovolná q_1, q_2. Je nasnadě, že f je prosté. Pro g a pro libovolné $q_1, q_2 \in Q$ z (28) dostáváme

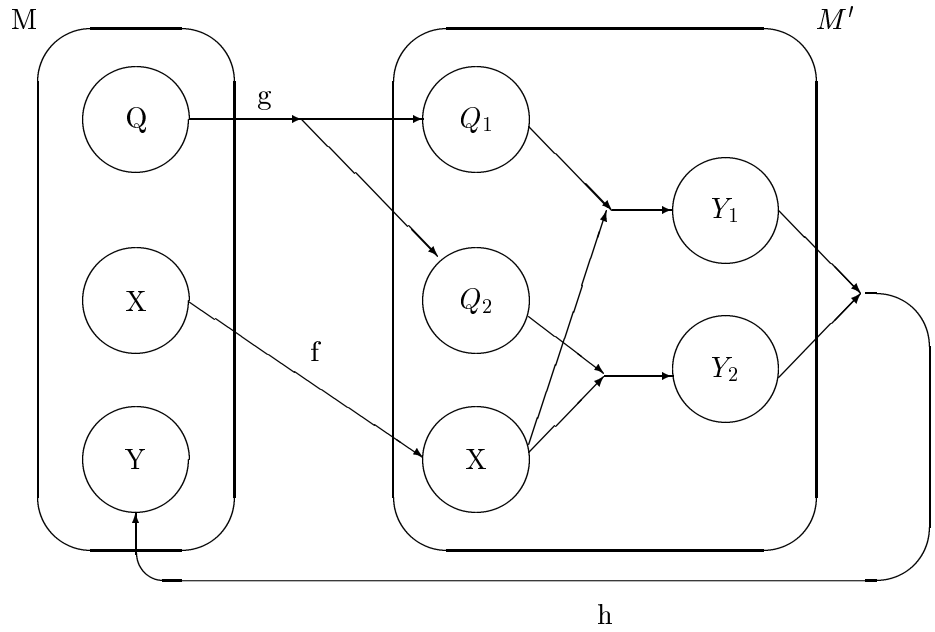
$$\begin{array}{rcl} (q_1 \neq q_2) & \Rightarrow & ((q_1, q_2) \notin \rho_1 \vee (q_1, q_2) \notin \rho_2) \Rightarrow \\ & \Rightarrow & (([q_1]_{\rho_1} \neq [q_2]_{\rho_1}) \vee ([q_1]_{\rho_2} \neq [q_2]_{\rho_2})) \Rightarrow g(q_1) \neq g(q_2), \end{array}$$

tedy g je prosté zobrazení (jednoznačnost definice g plyne z vlastností ekvivalence).

Aby M' mohlo být paralelní dekompozicí M, musí M' realizovat M a dále počet stavů M_1 a M_2 musí být menší než počet stavů M. Druhý požadavek je splněn díky netriviálnosti ρ_1 a ρ_2. První bude splněn, pokud pro M', M a f, g, h dokážeme vztahy (20) a (21) z definice realizace. K důkazu (20) použijeme postupně definice g,f,δ',δ_1,δ_2 a opět g

$$\begin{array}{rcl} \delta'(g(q), f(x)) & = & \delta'(([q]_{\rho_1}, [q]_{\rho_2}), x) = \\ & = & (\delta_1([q]_{\rho_1}, x), \delta_2([q]_{\rho_2}, x)) = \\ & = & ([\delta(q,x)]_{\rho_1}, [\delta(q,x)]_{\rho_2}) = g(\delta(q,x)) \end{array}$$

[24] značení zobrazení zde odpovídá definici realizace, definice funkcí δ' a β' viz (22) a (23)



Obr. 5: Přechody realizujících funkcí paralelní dekompozice

Vztah (21) platí podobně z definic g,f,β',β_1,β_2 a h a faktu $[q]_{\rho_1} \cap [q]_{\rho_2} = q$

$$\begin{aligned} h(\beta'(g(q), f(x))) &= h(\beta'([q]_{\rho_2}, [q]_{\rho_2}), x) = \\ &= h(\beta_1([q]_{\rho_1}, x), \beta_2([q]_{\rho_2}, x)) = \\ &= h(([q]_{\rho_1}, x), ([q]_{\rho_2}, x)) = \beta(q, x) \end{aligned}$$

Dokázali jsme, že M' realizuje M a počet stavů automatů M_1, M_2 je menší než počet stavů M, tedy M' je paralelní dekompozicí automatu M.[25] CBD

Příklad 3.18 *(Hledání kongruencí automatu) Najděte nejhrubší automatovou a alespoň dvě netriviální stavové kongruence automatu M, zadaného tabulkou. Dále sestrojte podílový automat M' automatu M podle nejhrubší automatové kongruence.*

M	0	1
A	$B/1$	$C/0$
B	$B/1$	$A/0$
C	$A/0$	$B/0$
D	$E/1$	$A/0$
E	$D/1$	$E/0$
F	$B/1$	$C/0$

V první řadě sestrojíme nejhrubší automatovou kongruenci. Použijeme algoritmus ze strany 11. Podle něj musíme nejprve definovat ekvivalenci $\overset{0}{\sim}$, kde dva

[25] pokud některý krok důkazu není zcela zřejmý, doporučuji přesně si rozepsat všechny předpoklady (definice paralelního spojení, automatů M_1, M_2 a funkcí f, g, h) a cílové rovnice (definice realizace) pro automaty a jejich komponenty, které v důkazu vystupují

stavy jsou ekvivalentní, právě když mají pro všechny vstupy stejné výstupy. Pro nás to znamená, že C je ekvivalentní samo se sebou a ostatní stavy každý s každým

$$\overset{0}{\sim} = \{\{A, B, D, E, F\}, \{C\}\}$$

V ekvivalenci $\overset{1}{\sim}$ budou ve stejné třídě ty stavy, které jsou ekvivalentní v $\overset{0}{\sim}$ a i přechody pro všechny vstupy pomocí funkce δ jsou ekvivalentní v $\overset{0}{\sim}$

$$\overset{1}{\sim} = \{\{A, F\}, \{B, D, E\}, \{C\}\}$$

Postupně dostáváme

$$\begin{aligned}
\overset{2}{\sim} &= \{\{A, F\}, \{B, D\}, \{E\}, \{C\}\} \\
\overset{3}{\sim} &= \{\{A, F\}, \{B\}, \{D\}, \{E\}, \{C\}\} \\
\overset{4}{\sim} &= \{\{A, F\}, \{B\}, \{D\}, \{E\}, \{C\}\}
\end{aligned}$$

Našli jsme automatovou kongruenci $\overset{3}{\sim}=\overset{4}{\sim}=\sim_1$. Protože každá automatová kongruence je také stavová, je i $\sim_1$ stavová kongruence. Navíc je $\sim_1$ netriviální, takže nám zbývá najít jen jednu netriviální stavovou kongruenci. Algoritmus 1 nám radí, abychom vzali nějaké dva stavy, prohlásili je za ekvivalentní (dostaneme $\overset{0}{\sim}$), přechody z těchto stavů přes stejné vstupy opět prohlásili za ekvivalentní atd. Vezměme například $A \overset{0}{\sim} B$. Dostáváme

$$\overset{0}{\sim} = \{\{A, B\}, \{C\}, \{D\}, \{E\}, \{F\}\}$$

Neboť $\delta(A, 1) = C$ a $\delta(B, 1) = A$, bude $A \overset{1}{\sim} C$:

$$\overset{1}{\sim} = \{\{A, B, C\}, \{D\}, \{E\}, \{F\}\}$$

Ze stavů A, B, C se již nikam jinam nedostaneme, tedy $\overset{2}{\sim}=\overset{1}{\sim}=\sim_2$. Ekvivalence $\overset{1}{\sim}=\overset{2}{\sim}=\sim_2$ je netriviální stavovou kongruencí a spolu s $\sim_1$ vyhovuje podmínkám první části úlohy.

Přestože se říká, že student by si neměl přidělávat práci a dělat víc, než se v zadání požaduje, v rámci lepšího zažití konstrukce stavových kongruencí uvedu některé další:

$$\begin{aligned}
\overset{0}{\sim} &= \{\{A, E\}, \{B\}, \{C\}, \{D\}, \{F\}\}\} \\
\overset{1}{\sim} &= \{\{A, C, E\}, \{B, D\}, \{F\}\} \\
\overset{2}{\sim} &= \{\{A, B, C, D, E\}, \{F\}\}
\end{aligned}$$

$$\begin{aligned}
\overset{0}{\sim} &= \{\{C, F\}, \{A\}, \{B\}, \{D\}, \{E\}\}\} \\
\overset{1}{\sim} &= \{\{A, B, C, F\}, \{D\}, \{E\}\} \\
\overset{2}{\sim} &= \{\{A, B, C, F\}, \{D\}, \{E\}\}
\end{aligned}$$

To už by snad stačilo, ne?

Najděme nyní podílový automat $M' = (Q', X, \delta', Y, \beta')$ automatu M podle $\sim_1$. Automat M je Mealyho automat, tedy i M' bude Mealyho automat. Definice podílového automatu říká, že množina stavů Q' je množinou tříd automatové kongruence $\sim_1$. Označme si jednotlivé třídy po řadě $T_1, \ldots, T_5$, aby manipulace s nimi byla přehlednější

$$\sim_1 = \{\{A, F\}, \{B\}, \{C\}, \{D\}, \{E\}\} = \{T_1, T_2, T_3, T_4, T_5\}$$

Automat M' má stejnou vstupní a výstupní abecedu jako automat M, tedy $X = \{0, 1\}$, $Y = \{0, 1\}$. Přechodovou a výstupní funkci δ' a β' dostaneme tak, že z každé třídy T_i vezmeme libovolný prvek $q \in T_i$ a definujeme $\delta'(T_i, a) = [\delta(q, a)]_{\sim_1}$, $\beta'(T_i, a) = \beta(q, a)$ pro všechna $a \in X$. Automat M' zapíšeme tabulkou a budeme hotovi.

M'	0	1
T_1	$T_2/1$	$T_3/0$
T_2	$T_2/1$	$T_1/0$
T_3	$T_1/0$	$T_2/0$
T_4	$T_5/1$	$T_1/0$
T_5	$T_4/1$	$T_5/0$

FINE

Příklad 3.19 *(Sestrojení dekompozic) Sestrojte sériovou a paralelní dekompozici automatu $M = (Q, X, \delta, Y, \beta)$, zadaného tabulkou*

M	a	b	c
1	$6/y$	$3/z$	$2/u$
2	$5/z$	$4/u$	$1/y$
3	$2/y$	$5/y$	$4/y$
4	$1/z$	$6/z$	$3/z$
5	$4/u$	$1/u$	$6/u$
6	$3/u$	$2/y$	$5/z$

Z tabulky lze o automatu M vyčíst, že

$$\begin{aligned} Q &= \{1, 2, 3, 4, 5, 6\} \\ X &= \{a, b, c\} \\ Y &= \{u, y, z\} \end{aligned}$$

Sestrojíme nejprve sériovou dekompozici podle postupu, který nám nabízí důkaz věty 3.16. Nejprve budeme potřebovat jednu stavovou kongruenci (označme ji ρ_1 automatu M. Podle algoritmu 1 volbou $1 \overset{0}{\sim} 2$ dostáváme

$$\begin{aligned} \overset{0}{\sim} &= \{\{1, 2\}, \{3\}, \{4\}, \{5\}, \{6\}\} \\ \overset{1}{\sim} &= \{\{1, 2\}, \{3, 4\}, \{5, 6\}\} \\ \overset{2}{\sim} &= \{\{1, 2\}, \{3, 4\}, \{5, 6\}\} \end{aligned}$$

Algoritmus se zastavil a my jsme obdrželi stavovou kongruenci

(29) $\rho_1 = \{\{1, 2\}, \{3, 4\}, \{5, 6\}\} = \{T_1, T_2, T_3\}$.

Kongruence ρ_1 a volba pořadí prvků v jednotlivých třídách kongruence ρ_1[26] nám jednoznačně určuje automaty

$$M_1 = (Q_1, X, \delta_1, Q_1 \times X, \beta_1)$$
$$M_2 = (Q_2, Q_1 \times X, \delta_2, Y, \beta_2)$$

z důkazu věty 3.16 tak, že $M' = M_1(\Longrightarrow)M_2$ je sériovou dekompozicí M. V automatech M_1 a M_2 je

- $Q_1 = \{T_1, T_2, T_3\}$
- $Q_2 = \{1, 2\}$, neboť třídy kongruence ρ_1 obsahují maximálně dva prvky
- $\delta_1(T, x) = [\delta(t, x)]_{\rho_1}$ pro $x \in X$, $T \in \rho_1$, $t \in T$
- $\delta_2(i, (T, x)) = j$, když funkce δ zobrazuje i-tý prvek třídy T na j-tý prvek třídy $S = \delta_1(T, x)$[27], pro $i, j \in Q_2$, $T, S \in Q_1$,$x \in X$
- β_1 je identická funkce
- $\beta_2(i, (T, x)) = y$, když výstup funkce β z i-tého prvku třídy T při vstupu x je y, pro $i \in Q_2, T \in Q_1, x \in X, y \in Y$

Funkce f, g, h, které zprostředkovávají realizaci, jsou definovány následovně

- $f = id_X$, $h = id_Y$
- $g(q) = (T, i)$, kde $T \in Q_1$ obsahuje prvek $q \in T$ na pozici i

Funkce g je popsána následující tabulkou

q	1	2	3	4	5	6
$g(q)$	$(T_1, 1)$	$(T_1, 2)$	$(T_2, 1)$	$(T_2, 2)$	$(T_3, 1)$	$(T_3, 2)$

Tabulky automatů M_1 a M_2 vypadají takto

M_1	a	b	c
T_1	$T_3/(T_1, a)$	$T_2/(T_1, b)$	$T_1/(T_1, c)$
T_2	$T_1/(T_2, a)$	$T_3/(T_2, b)$	$T_2/(T_2, c)$
T_3	$T_2/(T_3, a)$	$T_1/(T_3, b)$	$T_3/(T_3, c)$

M_2	(T_1, a)	(T_1, b)	(T_1, c)	(T_2, a)	(T_2, b)	(T_2, c)	(T_3, a)	(T_3, b)	(T_3, c)
1	$2/y$	$1/z$	$2/u$	$2/y$	$1/y$	$2/y$	$2/u$	$1/u$	$2/u$
2	$1/z$	$2/u$	$1/y$	$1/z$	$2/z$	$1/z$	$1/u$	$2/y$	$1/z$

Získali jsme tedy sériovou dekompozici automatu M. Nyní získejme ještě paralelní podle návodu z důkazu věty 3.17. Automat M_1 můžeme vzít nezměněn z předchozí části úlohy. Pro konstrukci druhého automatu (označme ho M_3) potřebujeme nějakou netriviální stavovou kongruenci ρ_2 takovou, aby bylo splněno $\rho_1 \cap \rho_2 = id_Q$. Vezměme například[28]

$$\rho_2 = \{\{1, 3, 5\}, \{2, 4, 6\}\} = \{S_1, S_2\}$$

[26] v našem případě je pořadí dáno zápisem (29)

[27] třída S nás sama o sobě nezajímá, pouze pořadí prvků v ní

[28] algoritmus pro získání ρ_2 nastartujeme volbou $1 \overset{0}{\sim} 3$

Pro automat $M_3 = (Q_3, X_3, \delta_3, Y_3, \beta_3)$ platí podobně jako pro M_1 (neboť jejich konstrukce je podle důkazu věty 3.17 stejná)

- $X_3 = X$
- $Q_3 = Q/_{\rho_2} = \{S_1, S_2\}$
- $Y_3 = Q_3 \times X_3$
- δ_3 a β_3 jsou zřejmé z tabulky

M_3	a	b	c
S_1	$S_2/(S_1, a)$	$S_1/(S_1, b)$	$S_2/(S_1, c)$
S_2	$S_1/(S_2, a)$	$S_2/(S_2, b)$	$S_1/(S_2, c)$

Funkce f', g', h', které realizují M pomocí $M'' = M_1 \parallel M_3$, jsou popsány v důkazu věty 3.17. Pro naši úlohu:

- f' je identita
- $g' : Q \to Q_1 \times Q_3$; $g'(q) = (T, S)$,
  kde $q \in T \in Q_1, q \in S \in Q_2, q \in T \cap S$
- $h' : Y_1 \times Y_3 \to Y$; $h'((T, x), (S, x)) = \beta(q, x)$,
  kde $q \in T \in Q_1, q \in S \in Q_2, x \in X, q \in T \cap S$

Tabulky funkcí g, h vypadají takto (s tím, že pro zbytek definičního oboru h, tedy při $T \cap S = \emptyset$ nebo pro $x \neq x'$, definujeme $h((T, x), (S, x')) =$ lib. prvek Y a do tabulky nezapisujeme):[29]

q	1	2	3	4	5	6
$(g(q))_1$	T_1		T_2		T_3	
$(g(q))_2$	S_1	S_2	S_1	S_2	S_1	S_2

T	S	$T \cap S$	$h((T, a), (S, a))$	$h((T, b), (S, b))$	$h((T, c), (S, c))$
T_1	S_1	1	y	z	u
T_1	S_2	2	z	u	y
T_2	S_1	3	y	y	y
T_2	S_2	4	z	z	z
T_3	S_1	5	u	u	u
T_3	S_2	6	u	y	z

FINE

4 Regulární jazyky

4.1 Akceptory

Zařízení, které rozpoznává nějaký jazyk L, tzn. pro každé vstupní slovo α rozhodne o tom, zda α patří do L či nikoliv, nazveme *akceptorem*. Jedná se

[29] obecně zápisem $(x)_i$ mám na mysli i-tou složku vektoru x; konkrétně na tomto místě je konstrukce použita na vektor $g(q)$

ve skutečnosti o Mooreův automat $\mathcal{A} = (Q, X, \delta, q_0, Y, \beta)$ s výstupní abecedou $Y = \{true, false\}$. Na výstupu je *true*, pokud vstupní slovo patří do jazyka rozpoznávaného akceptorem $\mathcal{A}$.

Definice 4.1 *Akceptorem* $\mathcal{A}$ *obvykle rozumíme pětici* $\mathcal{A} = (Q, X, \delta, q_0, F)$, *kde*

- Q *je konečná množina stavů*
- X *je konečná vstupní abeceda*
- $\delta : Q \times X \to Q$ *je přechodová funkce*
- $q_0 \in Q$ *je iniciální (nebo také počáteční) stav*
- $F \subseteq Q$ *je množina koncových (nebo také přijímacích) stavů* [1]

Posloupnost stavů $r_0, \ldots, r_n$ *se nazývá výpočtem nad slovem* $\alpha = a_1 a_2 \ldots a_n$, *když* $\delta(r_{i-1}, a_i) = r_i$ *pro každé* $i = 1, 2, \ldots, n$. *Řekneme, že výpočet je přijímací, když* $r_0 = q_0$ *a* $r_n \in F$. *Slovo* α *je přijímáno akceptorem* $\mathcal{A}$, *když existuje přijímací výpočet nad* α, *což je právě když* $\delta^*(q_0, \alpha) \in F$.

Jazyk přijímaný akceptorem $\mathcal{A}$ *je tvořen množinou všech slov přijímaných akceptorem* $\mathcal{A}$, *značíme ho* $L(\mathcal{A})$. *Tedy* $L(\mathcal{A}) = \{\alpha \in X^*; \delta^*(q_0, \alpha) \in F\}$.

O dvou akceptorech $\mathcal{A}_1, \mathcal{A}_2$ *řekneme, že jsou ekvivalentní, když přijímají stejný jazyk, tzn.* $L(\mathcal{A}_1) = L(\mathcal{A}_2)$.

Poznámka: Akceptor (obecně jakýkoliv automat) nazveme konečný, pokud si může zaznamenat pouze konečné množství informací, tedy např. když má jen konečnou množinu stavů.

Definice 4.2 *Jazyk* L *nad abecedou* X *nazveme regulární, je-li přijímán nějakým konečným akceptorem.*

Definice 4.3 *Stav* q *akceptoru* $\mathcal{A} = (Q, X, \delta, q_0, F)$ *je dosažitelný, když existuje* $\alpha \in X^*$ *takové, že* $q = \delta^*(q_0, \alpha)$.

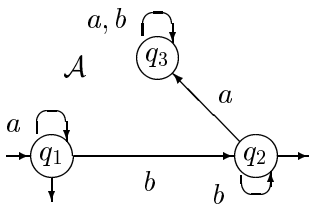
Poznámka: Podobně jako Mooreovy a Mealyho automaty, i akceptory mohou být zadány graficky, tabulkou funkce δ, případně slovně nebo jiným vhodným způsobem. Vyhnu se složitým formálním popisům zadání akceptoru. Raději uvedu příklad, ze kterého by mělo být všechno jasné.

Příklad 4.4 *(Akceptor) Sestrojte akceptor* $\mathcal{A} = (Q, X, \delta, q_0, F)$, *který přijímá jazyk*

$$L = \{a^i b^j; i, j = 0, 1, \ldots\}.$$

Jazyk L je množinou posloupností z abecedy $\{a, b\}$, ve kterých po znaku b již nenásleduje žádné a. Měli bychom tedy definovat akceptor, který takové posloupnosti přijímá. Jediné, co musíme zajistit, je: „Pokud po b přijde a, akceptor již vstupní slovo nesmí přijmout“. Akceptor $\mathcal{A}$ na následujícím obrázku vlevo tento požadavek jistě splňuje.

[1] Jinými slovy, máme-li Mooreův automat $\mathcal{A} = (Q, X, \delta, q_0, Y, \beta)$ s výstupní abecedou $Y = \{true, false\}$, liší se od akceptoru (Q, X, δ, q_0, F) pouze formálním zápisem, pokud platí $F = \{q; \beta(q) = true\}$



$\mathcal{A}$	a	b
$\leftrightarrow q_1$	q_1	q_2
$\leftarrow q_2$	q_3	q_2
q_3	q_3	q_3

V pravé polovině obrázku je potom tabulka akceptoru $\mathcal{A}$. Z obrázku, stejně jako z tabulky, lze všechny potřebné informace[2] o akceptoru $\mathcal{A}$ vyčíst. Pro pořádek uvedu ještě jeden způsob zadání $\mathcal{A}$. Tato varianta však vyžaduje nějaký doplňující popis funkce δ.

$$\mathcal{A} = (\{q_1, q_2, q_3\}, \{a, b\}, \delta_{\mathcal{A}}, q_1, \{q_1, q_2\})$$

FINE

Poznámka: Jistě jste si povšimli, že pokud se jednou dostaneme do q_3, již v něm zůstaneme a celý výpočet není přijímací. Stavům s touto vlastností se někdy říká „garbage“ (odpadové) stavy.

Povšimněme si, že vzhledem k definici akceptoru je třeba těchto stavů použít, kdykoli chceme zajistit, aby výpočet nad zbytkem slova již nikdy nebyl přijímající, neboť funkce δ musí být definovaná pro každou dvojici z $Q \times X$, tzn. akceptor musí v každém stavu znát pokračování přes každé písmeno vstupní abecedy[3].

Lemma 4.5 *Mějme akceptory $\mathcal{A}_1 = (Q_1, X, \delta_1, q_1, F_1)$, $\mathcal{A}_2 = (Q_2, X, \delta_2, q_2, F_2)$ takové, že platí*

1. $Q_2 = \{q \in Q_1; q$ *je dosažitelný v* $\mathcal{A}_1\}$
2. $\delta_2 : Q_2 \times X \to Q_2$ *je restrikce (zúžení)* δ_1 *na* Q_2, *tj.* $\delta_2(q, x) = \delta_1(q, x)$ *pro všechna* $x \in X$ *a* $q \in Q_2$
3. $F_2 = F_1 \cap Q_2$

Pak jsou tyto akceptory ekvivalentní.

Důkaz: Zřejmě $L(\mathcal{A}_2) \subseteq L(\mathcal{A}_1)$ je triviální fakt, který vyplývá z definice akceptoru $\mathcal{A}_2$. Nechť nyní $\alpha \in L(\mathcal{A}_1)$ je libovolné slovo. Pro toto slovo existuje v akceptoru $\mathcal{A}_1$ výpočet, který ovšem neobsahuje stavy z množiny $Q_1 \setminus Q_2$, neboť tyto jsou nedosažitelné. To znamená, že výpočet probíhá pouze stavy množiny Q_2, a tedy vzhledem k definici $\mathcal{A}_2$ je slovo α přijímáno v $\mathcal{A}_1$, právě když je přijímáno v $\mathcal{A}_2$. CBD

Podobně jako v automatech, i v akceptorech se definuje ekvivalence stavů. Následující definice je zřejmě analogická definici 2.4.

Definice 4.6 *Řekneme, že dva stavy q_1, q_2 akceptoru A jsou ekvivalentní (píšeme $q_1 \sim q_2$), když pro všechna $\alpha \in X^*$ platí*

$$\delta^*(q_1, \alpha) \in F \Leftrightarrow \delta^*(q_2, \alpha) \in F$$

[2] vstupní abeceda, množina stavů, iniciální stav (označen šipkou dovnitř), přijímací stavy (šipka ven), přechodová funkce

[3] neformálně řečeno... nicméně formálně by nebylo nutné říkat nic :)

Tvrzení 4.7 *Relace $\sim$ z definice 4.6 na akceptoru $\mathcal{A} = (Q, X, \delta, q_0, F)$ je automatovou kongruencí.*

Důkaz: Je třeba ověřit podmínky (10) a (11) z definice automatové kongruence 3.2. Vezměme dva stavy $q_1, q_2 \in Q$ takové, že $q_1 \sim q_2$. Z definice δ^* a ekvivalence stavů q_1, q_2 pro libovolné $a \in X$ a $\alpha \in X^*$ dostáváme

$$\delta^*(\delta(q_1, a), \alpha) = \delta^*(q_1, a\alpha) \in F \iff F \ni \delta^*(q_2, a\alpha) = \delta^*(\delta(q_2, a), \alpha),$$

tudíž $\delta(q_1, a) \sim \delta(q_2, a)$ a je ověřena podmínka (10). Podmínka (11) má pro akceptory formálně poněkud jiný zápis[4]:

$$q_1 \sim q_2 \Rightarrow (q_1 \in F \Leftrightarrow q_2 \in F)$$

Protože vztah z definice 4.6 platí i pro $\alpha = \Lambda$, z $q_1 \sim q_2$ dostáváme ekvivalenci $\delta^*(q_1, \Lambda) \in F \Leftrightarrow \delta^*(q_2, \Lambda) \in F$ a následně z definice δ^* platí $q_1 \in F \Leftrightarrow q_2 \in F$, čímž je ověřena podmínka (11). Relace $\sim$ je tedy automatovou kongruencí. CBD

Definice 4.8 *Je-li $\sim$ automatová kongruence, pak podílovým akceptorem akceptoru $\mathcal{A} = (Q, X, \delta, q_0, F)$ rozumíme akceptor*

$$\mathcal{A}/_\sim = (Q/_\sim, X, \delta_\sim, [q_0]_\sim, F/_\sim),$$

kde pro všechna $q \in Q, x \in X$ definujeme

$$\delta_\sim([q]_\sim, x) = [\delta(q, x)]_\sim$$

Jako cvičení doporučuji rozmyslet si, že pro $F/_\sim$ z předcházející definice platí

$$F/_\sim = \{[q]_\sim; (q \in Q) \wedge ([q]_\sim \cap F \neq \emptyset)\} = \{[q]_\sim; (q \in Q) \wedge ([q]_\sim \subset F)\},$$

protože dva stavy akceptoru jsou ekvivalentní v $\sim$ jen tehdy, pokud jsou buď oba dva v F, nebo žádný z nich není v F[5].

Tvrzení 4.9 *Akceptory $\mathcal{A}$ a $\mathcal{A}/_\sim$ jsou ekvivalentní.*

Důkaz: Pro akceptory $\mathcal{A} = (Q, X, \delta, q_0, F)$ a $\mathcal{A}/_\sim$ platí

$$\begin{aligned} \alpha \in L(\mathcal{A}) \quad &\Leftrightarrow \quad \delta^*(q_0, \alpha) \in F \Leftrightarrow [\delta^*(q_0, \alpha)]_\sim \in F/_\sim \Leftrightarrow \\ &\Leftrightarrow \quad \delta^*_\sim([q_0]_\sim, \alpha) \in F/_\sim \Leftrightarrow \alpha \in L(\mathcal{A}/_\sim), \end{aligned}$$

tedy $L(\mathcal{A}) = L(\mathcal{A}/_\sim)$. CBD

Definice 4.10 *Akceptor $\mathcal{A}$ je redukovaný, jestliže žádné dva stavy nejsou ekvivalentní.*

Akceptor $\mathcal{A}$ je dosahující, jestliže všechny jeho stavy jsou dosažitelné.

[4] že tato formule je ekvivalentní s (11), ponechávám čtenáři na rozvážení

[5] viz definice 4.6 pro $\alpha = \Lambda$

Poznámka: Stejně jako u automatů, i v případě akceptorů lze snadno nahlédnout, že podílový akceptor $\mathcal{A}/_\sim$ je redukovaný, jakmile $\sim$ je automatová kongruence z definice 4.6[6].

Definice 4.11 *Nechť $\mathcal{A}_1 = (Q_1, X, \delta_1, q_1, F_1)$, $\mathcal{A}_2 = (Q_2, X, \delta_2, q_2, F_2)$ jsou akceptory. Bijekci[7] $g : Q_1 \to Q_2$ nazveme izomorfismem akceptorů $\mathcal{A}_1$ a $\mathcal{A}_2$, platí-li následující podmínky:*

1. $g(q_1) = q_2$
2. $g(F_1) = F_2$
3. $(\forall\, q \in Q_1)(\forall\, a \in X)(g(\delta_1(q, a)) = \delta_2(g(q), a))$

Přestože definice izomorfismu akceptorů není sama o sobě symetrická, pojem „být izomorfní“ symetrický samozřejmě je. Vezmeme-li totiž g izomorfismus akceptorů $\mathcal{A}_1$ a $\mathcal{A}_2$, je g^{-1} zřejmě izomorfismus $\mathcal{A}_2$ a $\mathcal{A}_1$.

Tvrzení 4.12 *Pro každý akceptor $\mathcal{A}$ existuje s ním ekvivalentní redukovaný dosahující akceptor.*

Důkaz: Vezměme akceptor $\mathcal{A} = (Q, X, \delta, q_0, F)$. Označme

$$M = \{q; (\forall\, \alpha \in X^*)(\delta^*(q_0, \alpha) \neq q)\},$$

tedy M je množina nedosažitelných stavů v $\mathcal{A}$. Z lemmatu 4.5 víme, že jeli $Q' = Q \setminus M$, $F' = F \cap Q'$ a $\delta' : Q' \times X \to Q'$ zúžení δ na Q', pak akceptor $\mathcal{A}' = (Q', X, \delta', q_0, F')$ je dosahující a navíc ekvivalentní s $\mathcal{A}$. Dále z tvrzení 4.9 víme, že podílový akceptor $\mathcal{A}'/_\sim$[8] je ekvivalentní s $\mathcal{A}'$, tedy i s $\mathcal{A}$. Navíc $\mathcal{A}'/_\sim$ je redukovaný[9] a jednoduše dostaneme, že $\mathcal{A}'/_\sim$ je dosahující, protože $\mathcal{A}'$ je dosahující, tudíž jsme k $\mathcal{A}$ našli požadovaný redukovaný dosahující akceptor. CBD

Věta 4.13 *Každé dva ekvivalentní redukované dosahující akceptory jsou izomorfní.*

Důkaz: Vezměme $\mathcal{A}_1 = (Q_1, X, \delta_1, q_1, F_1)$, $\mathcal{A}_2 = (Q_2, X, \delta_2, q_2, F_2)$ dva akceptory. Neboť jsme nenároční, požadujme pouze, aby byly ekvivalentní, redukované a dosahující. Dokážeme, že jsou izomorfní, tzn. najdeme bijekci $g : Q_1 \to Q_2$ splňující podmínky definice 4.11. Definujme

$$(\forall\, \alpha \in X^*)(g(\delta_1^*(q_1, \alpha)) \stackrel{df}{=} \delta_2^*(q_2, \alpha))$$

Proč? To se uvidí! Ve třech vlnách ověříme, že g je izomorfismus akceptorů:

[6] viz věta 3.6

[7] vzájemně jednoznačné zobrazení, tzn. zobrazení *prosté* a *na*

[8] $\sim$ viz definice 4.6

[9] poměrně triviální, lze formálně nahlédnout z tvrzení 4.7 a přeformulováním věty 3.6 pro akceptory

1. Nejprve ukážeme, že g je funkce, tedy že je definovaná na celém Q_1 a jednomu $q \in Q_1$ přísluší nejvýše jedno $g(q) \in Q_2$.

   (a) Vezměme libovolné $q \in Q_1$. Neboť $\mathcal{A}_1$ je dosahující,

   $$(\exists\, \alpha \in X^*)(\delta_1^*(q_1, \alpha) = q)$$

   (b) Chceme ukázat $(\forall\, q, q' \in Q_1)((q = q') \Rightarrow (g(q) = g(q')))$. Mějme slova $\alpha, \beta \in X^*$ taková, že $\delta_1^*(q_1, \alpha) = \delta_1^*(q_1, \beta)$. Potom pro všechna $\gamma \in X^*$ platí

   $$\delta_1^*(q_1, \alpha\gamma) = \delta_1^*(\delta_1^*(q_1, \alpha), \gamma) = \delta_1^*(\delta_1^*(q_1, \beta), \gamma) = \delta_1^*(q_1, \beta\gamma),$$

   neboli

   $$\alpha\gamma \in L(\mathcal{A}_1) \Leftrightarrow \beta\gamma \in L(\mathcal{A}_1).$$

   Z ekvivalence akceptorů $\mathcal{A}_1, \mathcal{A}_2$ víme $L(\mathcal{A}_1) = L(\mathcal{A}_2)$, můžeme tedy psát $\alpha\gamma \in L(\mathcal{A}_2) \Leftrightarrow \beta\gamma \in L(\mathcal{A}_2)$, což je ekvivalentní zápisu

   $$\delta_2^*(q_2, \alpha\gamma) \in F_2 \Leftrightarrow \delta_2^*(q_2, \beta\gamma) \in F_2$$

   pro libovolné γ, odkud z definice ekvivalence stavů akceptoru platí $\delta_2^*(q_2, \alpha) \sim \delta_2^*(q_2, \beta)$. Z toho $\delta_2^*(q_2, \alpha) = \delta_2^*(q_2, \beta)$, neboť $\mathcal{A}_2$ je redukovaný. Označíme-li $q = \delta_1^*(q_1, \alpha), q' = \delta_1^*(q_1, \beta)$, dokázali jsme

   $$(q = q') \Rightarrow (g(q) = g(q')).$$

   Zobrazení g je tedy funkcí.

2. Nyní dokážeme, že g splňuje tři podmínky z definice izomorfismu (definice 4.11).

   (a) $g(q_1) = g(\delta_1^*(q_1, \Lambda)) = \delta_2^*(q_2, \Lambda) = q_2$

   (b) Vezměme $q' \in F_2$. Neboť $\mathcal{A}_2$ je dosahující, $(\exists\, \alpha \in X^*)(q' = \delta_2^*(q_2, \alpha))$. Potom slovo α musí být z ekvivalence akceptorů $\mathcal{A}_1, \mathcal{A}_2$ přijímáno i v $\mathcal{A}_1$, tzn. $(\exists\, q \in F_1)(q = \delta_1^*(q_1, \alpha))$, přičemž z definice g platí $g(q) = q'$. Z uvedeného plyne $F_2 \subseteq g(F_1)$. Podobně z předpokladu, že $\mathcal{A}_1$ je dosahující, vyplývá $(\forall\, q \in F_1)(\exists\, \alpha \in X^*)(q = \delta_1^*(q_1, \alpha))$ a dále z ekvivalence $\mathcal{A}_1, \mathcal{A}_2$ platí $(\exists\, q' \in F_2)(q' = \delta_2^*(q_2, \alpha))$. Proto $g(F_1) \subseteq F_2$ a s první inkluzí $g(F_1) = F_2$.

   (c) Nechť $q \in Q_1, a \in X$. Z dosažitelnosti všech stavů $\mathcal{A}_1$

   $$(\exists\, \alpha \in X^*)(\delta_1^*(q_1, \alpha) = q).$$

   Nyní z definic δ^* a g

   $$\begin{aligned} g(\delta_1(q, a)) &= g(\delta_1(\delta_1^*(q_1, \alpha), a)) = g(\delta_1^*(q_1, \alpha a)) = \\ &= \delta_2^*(q_2, \alpha a) = \delta_2(\delta_2^*(q_2, \alpha), a) = \delta_2(g(q), a) \end{aligned}$$

3. Zbývá dokázat, že g je bijektivní

   (a) Nejprve ukážeme, že g je surjektivní (na). Protože $\mathcal{A}_2$ je dosahující,

   $$(\forall\, q \in Q_2)(\exists\, \alpha \in X^*)(q = \delta_2^*(q_2, \alpha) = g(\delta_1^*(q_1, \alpha))),$$

   tzn. existuje $q' = \delta_1^*(q_1, \alpha) \in Q_1$ takové, že $g(q') = q$.

(b) Naším orgasmem bude důkaz injektivity g[10]. Chceme, aby pro všechna $q, q' \in Q_1$ platilo $g(q) = g(q') \Rightarrow q = q'$. Neboť $\mathcal{A}_1$ je redukovaný, stačí ukázat, že $q \sim q'$, tzn.

$$(\forall\, \gamma \in X^*)(\delta_1^*(q,\gamma) \in F_1 \Leftrightarrow \delta_1^*(q',\gamma) \in F_1).$$

Tak do toho!!! Vezměme $q, q' \in Q_1, g(q) = g(q')$. Neboť $\mathcal{A}_1$ je dosahující, $(\exists\, \alpha, \beta \in X^*)(q = \delta_1^*(q_1,\alpha), q' = \delta_1^*(q_1,\beta))$. Z definice δ^*, q, q' pro libovolné $\gamma \in X^*$ platí

$$\begin{array}{rcl} \delta_1^*(q_1,\alpha\gamma) & = & \delta_1^*(\delta_1^*(q_1,\alpha),\gamma) = \delta_1^*(q,\gamma) \\ & \Rightarrow & \alpha\gamma \in L(\mathcal{A}_1) \Leftrightarrow \delta_1^*(q,\gamma) \in F_1 \\ \delta_1^*(q_1,\beta\gamma) & = & \delta_1^*(\delta_1^*(q_1,\beta),\gamma) = \delta_1^*(q',\gamma) \\ & \Rightarrow & \beta\gamma \in L(\mathcal{A}_1) \Leftrightarrow \delta_1^*(q',\gamma) \in F_1 \end{array} \tag{30}$$

Dále z $g(q) = g(q')$ platí

$$\begin{array}{rcl} \alpha\gamma \in L(\mathcal{A}_2) & \Leftrightarrow & F_2 \ni \delta_2^*(q_2,\alpha\gamma) = \delta_2^*(g(q),\gamma) = \\ & = & \delta_2^*(g(q'),\gamma) = \delta_2^*(q_2,\beta\gamma) \in F_2 \Leftrightarrow \\ & \Leftrightarrow & \beta\gamma \in L(\mathcal{A}_2), \end{array}$$

tedy z ekvivalence akceptorů $\mathcal{A}_1, \mathcal{A}_2$ je

$$\alpha\gamma \in L(\mathcal{A}_2) = L(\mathcal{A}_1) \Leftrightarrow \beta\gamma \in L(\mathcal{A}_2) = L(\mathcal{A}_1).$$

Užitím této ekvivalence a ekvivalencí (30) dostáváme

$$\delta_1^*(q,\gamma) \in F_1 \Leftrightarrow \delta_1^*(q',\gamma) \in F_1,$$

tudíž $q \sim q'$ a dále $q = q'$, neboť $\mathcal{A}_1$ je redukovaný. CBD

Dokázali jsme, že pokud dva ekvivalentní akceptory $\mathcal{A}_1, \mathcal{A}_2$ jsou redukované a dosahující, je g definované na začátku důkazu izomorfismem těchto akceptorů.

Důsledek 4.14 *Pro každý regulární jazyk existuje až na izomorfismus* jediný *akceptor s minimálním počtem stavů, který ho přijímá.*[11]

4.2 Syntaktické kongruence jazyků

Definice 4.15 *Pravou kongruencí nad X^* nazveme každou ekvivalenci $\sim$ na X^*, pro kterou platí $(\forall\, \alpha, \beta, \gamma \in X^*)(\alpha \sim \beta \Rightarrow \alpha\gamma \sim \beta\gamma)$.*

Kongruencí nad X^* nazveme každou ekvivalenci $\sim$ na X^*, pro kterou platí $(\forall\, \alpha, \beta, \gamma, \delta \in X^*)(\alpha \sim \beta \Rightarrow \delta\alpha\gamma \sim \delta\beta\gamma)$.

Mějme dále jazyk $L \subseteq X^*$. Pak relace $\sim$ na X^* se nazývá pravá syntaktická kongruence jazyka L, kdykoliv

$$(\forall\, \alpha, \beta \in X^*)(\alpha \sim \beta \Longleftrightarrow (\forall\, \gamma \in X^*)(\alpha\gamma \in L \Leftrightarrow \beta\gamma \in L))$$

Obdobně relace $\sim$ na X^* se nazývá syntaktická kongruence jazyka L, kdykoliv

$$(\forall\, \alpha, \beta \in X^*)(\alpha \sim \beta \Longleftrightarrow (\forall\, \gamma, \delta \in X^*)(\delta\alpha\gamma \in L \Leftrightarrow \delta\beta\gamma \in L))$$

[10] tzn. budeme dokazovat, že g je prosté

[11] vyplývá přímo z tvrzení 4.12 a věty 4.13

Jak okamžitě vidíme z předcházející definice, (pravá) syntaktická kongruence je reflexivní, symetrická i tranzitivní a je tedy ekvivalencí.

Podobně jako pravé (syntaktické) kogruence můžeme samozřejmě definovat i levé (syntaktické) kongruence. V dalším textu s nimi však pracovat nebudeme, neobtěžuji proto sebe (ani Vás) jejich formální definicí.

Tvrzení 4.16 *(Pravá) syntaktická kongruence jazyka L je (pravou) kongruencí nad X^* pro každý jazyk L nad X.*

Důkaz: Provedeme pro syntaktickou kongruenci (pro pravé syntaktické kongruence je důkaz obdobný). Je-li $\sim$ syntaktická kongruence, pak pro libovolná $\alpha, \beta \in X^*$ z definice 4.15 plyne pro všechna $\gamma_1, \gamma_2, \delta_1, \delta_2 \in X^*$

$$\begin{aligned}(\alpha \sim \beta) &\iff (\delta_1\delta_2\alpha\gamma_2\gamma_1 \in L \Leftrightarrow \delta_1\delta_2\beta\gamma_2\gamma_1 \in L) \iff \\ &\iff (\delta_1(\delta_2\alpha\gamma_2)\gamma_1 \in L \Leftrightarrow \delta_1(\delta_2\beta\gamma_2)\gamma_1 \in L) \Longrightarrow \\ &\Longrightarrow (\delta_2\alpha\gamma_2 \sim \delta_2\beta\gamma_2) \qquad \boxed{\text{CBD}}\end{aligned}$$

Poznámka: Je-li $\sim$ (pravá) syntaktická kongruence jazyka $L \subseteq X^*$, pak

$$(\forall\, \alpha, \beta \in X^*)(\alpha \sim \beta \Longrightarrow (\alpha \in L \Leftrightarrow \beta \in L)).$$

Tato trivialita plyne přímo z definice pravé syntaktické kongruence ($\gamma = \Lambda$).

Věta 4.17 *(Myhill-Nerode) Pro každý jazyk L nad abecedou X jsou následující podmínky ekvivalentní:*

1. *L je regulární.*
2. *Pravá syntaktická kongruence jazyka L má konečně mnoho tříd.*
3. *Syntaktická kongruence jazyka L má konečně mnoho tříd.*

Důkaz provedeme cyklicky. Dokážeme postupně implikace $2 \Rightarrow 1, 1 \Rightarrow 3, 3 \Rightarrow 2$.

$2 \Rightarrow 1$. Máme X^* a $L \subseteq X^*$, pravá syntaktická kongruence $\sim$ jazyka L má konečně mnoho tříd. Chceme najít konečný akceptor $\mathcal{A} = (Q, X, \delta, q_0, F)$, který přijímá jazyk L ($L = L(\mathcal{A})$, pak L bude regulární). Definujme $\mathcal{A}$ následovně:

$$\begin{aligned} & & Q &= X^*/_\sim \\ (\forall\, \alpha \in X^*)(\forall\, a \in X) \qquad & (\delta([\alpha]_\sim, a) &= [\alpha a]_\sim) \\ & & q_0 &= [\Lambda]_\sim \\ & & F &= \{[\alpha]_\sim ; \alpha \in L\}\end{aligned}$$

V první řadě je třeba si uvědomit, že $\mathcal{A}$ je konečný akceptor, protože $|X^*/_\sim|$ je počet tříd pravé syntaktické kongruence $\sim$, který je dle předpokladu konečný. Hned nato je nutné dokázat korektnost definice přechodové funkce δ, tzn. dokázat implikaci

$$\alpha_1 \sim \alpha_2 \Rightarrow (\forall\, a \in X)(\delta([\alpha_1]_\sim, a) = \delta([\alpha_2]_\sim, a)),$$

která ale okamžitě plyne z triviality $\alpha_1 \sim \alpha_2 \Rightarrow [\alpha_1]_\sim = [\alpha_2]_\sim$.

Jazyk přijímaný akceptorem $\mathcal{A}$ je $L(\mathcal{A}) = \{\alpha \in X^*; \delta^*(q_0, \alpha) \in F\}$. Pro všechna $\alpha \in L$ platí

$$\delta^*(q_0, \alpha) = [\Lambda\alpha]_\sim = [\alpha]_\sim \in F,$$

tedy $\alpha \in L(\mathcal{A})$ a $L \subseteq L(\mathcal{A})$. Naopak, pro $\alpha \in L(\mathcal{A})$ platí

$$\delta^*(q_0, \alpha) \in F \Rightarrow (\exists\, \beta \in L)(\alpha \sim \beta) \Rightarrow \alpha \in L \Rightarrow L(\mathcal{A}) \subseteq L$$

a podle předchozí části důkazu $L = L(\mathcal{A})$.

Poznámka: O akceptoru $\mathcal{A}$ dokážeme, že je navíc dosahující a redukovaný a podle důsledku 4.13 má ze všech akceptorů[12], přijímajících jazyk L, minimální počet stavů, který je roven počtu rozkladových tříd pravé syntaktické kongruence L!!! Množina Q je tvořena právě všemi třídami $[\alpha]_\sim = \delta^*(q_0, \alpha), \alpha \in X^*$, tzn. do libovolného stavu $q \in Q$ se dostaneme z q_0, jakmile na vstupu je reprezentant třídy q. Odtud $\mathcal{A}$ je dosahující. Nechť dále $[\alpha]_\sim, [\beta]_\sim$ jsou ekvivalentní ve smyslu definice ekvivalence stavů akceptoru[13]. Pak pro všechna $\gamma \in X^*$ platí

$$(\delta^*([\alpha]_\sim, \gamma) = [\alpha\gamma]_\sim \in F) \Longleftrightarrow (\delta^*([\beta]_\sim, \gamma) = [\beta\gamma]_\sim \in F),$$

proto $\alpha\gamma \in L \Leftrightarrow \beta\gamma \in L$, tudíž z definice pravé syntaktické kongruence platí $\alpha \sim \beta$ a $[\alpha]_\sim = [\beta]_\sim$, neboli $\mathcal{A}$ je redukovaný.

$1 \Rightarrow 3$. Máme konečný redukovaný dosahující akceptor $\mathcal{A} = (Q, X, \delta, q_0, F)$ a chceme ukázat, že syntaktická kongruence na $L(\mathcal{A})$ má konečně mnoho tříd. Definujme relaci $\approx$ na X^* předpisem

$$\alpha \approx \beta \Leftrightarrow (\forall\, q \in Q)(\delta^*(q, \alpha) = \delta^*(q, \beta))$$

pro $\alpha, \beta \in X^*$. Dokážeme, že $\approx$ je syntaktickou kongruencí $L(\mathcal{A})$ s konečně mnoha třídami. Reflexivita, symetrie a tranzitivita relace $\approx$ je zřejmá z definice. Počet rozkladových tříd $\approx$ je nejvýše $|Q|^{|Q|}$[14], tedy konečné číslo. Zbývá dokázat, že $\approx$ je syntaktickou kongruencí jazyka $L(\mathcal{A})$. Vezměme

[12] Máme teď na mysli deterministické akceptory. Protože zatím jiné neznáme, může se tato poznámka zdát nemístná. Nicméně až poznáme nedeterministické akceptory, ukáže se být velmi podstatnou!!!

[13] ve smyslu definice 4.6

[14] Označme $n = |Q|$. Když pro slovo $\alpha \in X^*$ definujeme zobrazení $f_\alpha : Q \to Q$ předpisem $f_\alpha(q) = \delta^*(q, \alpha)$ pro každé $q \in Q$, pak $\alpha \approx \beta$ právě když $f_\alpha = f_\beta$, ale zobrazení z Q do Q je n^n, proto má $\approx$ nejvýše n^n tříd; Jiný možný způsob odvození počítáním možných ukončení výpočtu nad libovolným slovem započatého v libovolném stavu. Z každého stavu skončí výpočet v jednom z n stavů, neboli pro jeden pevný stav q vznikne n tříd. V definici $\approx$ je ale pro všechna $q \in Q$, tedy každý z n stavů vytvoří n tříd nezávisle na ostatních stavech. Proto n^n.

$\alpha \approx \beta$. Potom pro všechna $\gamma, \sigma \in X^*$ a $q \in Q$, když položíme $q' = \delta^*(q, \sigma)$, z definic δ^* a $\approx$ platí

$$\begin{aligned}\delta^*(q, \sigma\alpha\gamma) &= \delta^*(\delta^*(\delta^*(q, \sigma), \alpha), \gamma) = \delta^*(\delta^*(q', \alpha), \gamma) = \\ &= \delta^*(\delta^*(q', \beta), \gamma) = \delta^*(\delta^*(\delta^*(q, \sigma), \beta), \gamma) = \\ &= \delta^*(q, \sigma\beta\gamma),\end{aligned}$$

tedy $\sigma\alpha\gamma \approx \sigma\beta\gamma$ a $\approx$ je kongruencí s konečně mnoha třídami.

Zbývá ukázat, že $\approx$ je syntaktickou kongruencí, tzn. pro všechna $\alpha, \beta \in X^*$

$$\alpha \approx \beta \iff (\forall \sigma, \gamma \in X^*)(\sigma\alpha\gamma \in L(\mathcal{A}) \Leftrightarrow \sigma\beta\gamma \in L(\mathcal{A}))$$

Implikace zleva doprava plyne z faktu, že $\approx$ je kongruence, což jsme již dokázali. Zůstává opačná implikace. Vezměme libovolné $q \in Q$ a označme $q_1 = \delta^*(q, \alpha), q_2 = \delta^*(q, \beta)$, čímž jsme požadovaný vztah $\alpha \approx \beta$ přepsali na $q_1 = q_2$. Neboť $\mathcal{A}$ je redukovaný, stačí dokázat $q_1 \sim q_2$[15], neboli problém jsme opět převedli, tentokrát na

$$(31) \quad (\forall \gamma \in X^*)(\delta^*(q_1, \gamma) \in F \Leftrightarrow \delta^*(q_2, \gamma) \in F),$$

což už se dá vydržet. Protože $\mathcal{A}$ je dosahující, jistě existuje σ takové, že $\delta^*(q_0, \sigma) = q$. Potom platí

$$\begin{aligned}F \ni \delta^*(q_1, \gamma) = \delta^*(q, \alpha\gamma) = \delta^*(q_0, \sigma\alpha\gamma) &\quad \Leftrightarrow \quad \sigma\alpha\gamma \in L(\mathcal{A}) \\ F \ni \delta^*(q_2, \gamma) = \delta^*(q, \beta\gamma) = \delta^*(q_0, \sigma\beta\gamma) &\quad \Leftrightarrow \quad \sigma\beta\gamma \in L(\mathcal{A})\end{aligned}$$

Protože dokazujeme implikaci zprava doleva, předpokládáme pravdivost pravé strany ekvivalence, neboli víme

$$(\forall \sigma, \gamma \in X^*)(\sigma\alpha\gamma \in L(\mathcal{A}) \Leftrightarrow \sigma\beta\gamma \in L(\mathcal{A}))$$

a z předchozích dvou rovnic proto platí (31), čímž je implikace zprava doleva dokázána a $\approx$ je syntaktickou kongruencí $L(\mathcal{A})$.

$3 \Rightarrow 2$. Má-li syntaktická kongruence konečný počet rozkladových tříd, má nutně i pravá syntaktická kongruence konečný počet rozkladových tříd, neboť počet tříd pravé kongruence je zřejmě menší nebo roven počtu tříd syntaktické kongruence. CBD

Příklad 4.18 *(Syntaktické kongruence) Najděte synt. kongruence jazyků*

$$L_1 = \{a^i b^j; i, j = 0, 1, \ldots\}$$
$$L_2 = \{a^i b^i; i = 0, 1, \ldots\}$$

Sestrojme nejprve syntaktickou kongruenci $\sim_{L_1}$ jazyka L_1. Zjistíme, která slova jsou v L_1 navzájem kongruentní, a poté dokážeme, že žádné dvě z těchto tříd nelze již spojit v jednu. Všechna slova typu $\sigma_{ij} = a^i abb^j$ jsou mezi sebou kongruentní, protože vezmeme-li σ_{ij} a σ_{kl}, potom

[15] ve smyslu definice 4.6

- přidáme-li oběma slovům suffix[16] neobsahující a a prefix neobsahující b, patří obě nová slova do L_1.
- přidáme-li oběma slovům suffix obsahující a nebo prefix obsahující b, žádné ze získaných slov do L_1 nepatří.

Slova typu σ_{ij} pro $i, j = 0, 1, \ldots$ jsou tedy mezi sebou navzájem kongruentní v L_1 a mohou tvořit třídu σ hledané syntaktické kongruence $\sim_{L_1}$. Obdobně se ukáže kongruence všech slov typu

$$\begin{aligned} \alpha_i &= a^i \\ \beta_j &= b^j \\ \gamma_{k_a k_b k \ell_a \ell_b \ell} &= (a^{k_a} b^{k_b})^k ba (a^{\ell_a} b^{\ell_b})^\ell \end{aligned}$$

pro $i, k_a, k_b, k, \ell_a, \ell_b, \ell \geq 0$, $j > 0$[17] (tedy α, β, γ jsou další množiny, které budou kandidovat na třídy v $\sim_{L_1}$). Zbývá dokázat, že $\alpha, \beta, \gamma, \sigma$ jsou po dvou různé. Třída γ se vyznačuje tím, že přidáním jakéhokoliv prefixu nebo suffixu k jejím slovům vytvoříme slova, která nepatří do L_1. Slova α_i nejsou kongruentní se slovy β_j (stačí volit suffix a)[18], analogicky $\alpha \neq \sigma$ (volbou suffixu a) a $\beta \neq \sigma$ (prefix b). Syntaktická kongruence jazyka L_1 má tedy čtyři (po dvou navzájem různé) třídy $\alpha, \beta, \gamma, \sigma$.

Ukážeme nyní, že syntaktická kongruence $\sim_{L_2}$ jazyka L_2 má nekonečně mnoho tříd. Lehce nahlédneme, že slova $a^i b^j$ a $a^i b^k$ pro $j, k \leq i$, $j \neq k$, i pevné[19], nejsou kongruentní (stačí přidat suffix $b^{|i-j|}$) a tvoří tedy i tříd $\sim_{L_2}$. Po tomto krátkém úvodu snad každý uzná, že

$$\sim_{L_2} = \{\{a^i b^j; (i-j) = c\}; c \in Z\} \cup \{\{a, b\}^* \setminus \{a^i b^j; i, j \in N_0\}\},$$

tedy $\sim_{L_2}$ má nekonečně mnoho tříd[20].

Z Myhill-Nerodovy věty (4.17) vyplývá, že jazyk L_1 je regulární, kdežto L_2 regulární není[21]. FINE

Nyní podáme charakterizaci syntaktické kongruence pomocí jazyka.

Tvrzení 4.19 *Buď $L \subseteq X^*$ jazyk. Pak ekvivalence $\sim$ na X^* je syntaktickou kongruencí (resp. pravou syntaktickou kongruencí) jazyka L, právě když $\sim$ je nejhrubší kongruence (resp. pravá kongruence) taková, že platí*

(32) $(\forall \alpha, \beta \in X^*)(\alpha \sim \beta \Rightarrow (\alpha \in L \Leftrightarrow \beta \in L))$.

---

[16] suffixem délky n slova α se rozumí posledních n znaků slova α; podobně prefix délky n je prvních n znaků; vlastní prefix (suffix) slova α je prefix (suffix) délky menší než $|\alpha|$ a různý od Λ.

[17] povšimněte si, že prázdné slovo Λ patří právě do α_i

[18] $\alpha_i a \in L_1, \beta_j a \notin L_1$

[19] leč libovolné

[20] Zkuste se zamyslet nad tím, proč jsou uvedené třídy skutečně třídami $\sim_{L_2}$, a pochopit to *jako trivialitu*. Není to samozřejmě nic těžkého, ale přispěje to k pochopení myšlenky kongruencí. Malá nápověda: třída za znakem sjednocení ($\cup$) je třídou těch slov, která nelze žádným způsobem doplnit na slova z L_2.

[21] regularita L_1 již byla jednou prokázána v příkladu 4.4, neregularita L_2 bude ještě jednou ověřena v příkladu 4.25

Důkaz: Tvrzení dokážeme pro syntaktické kongruence. Nejprve implikaci zleva doprava. Jak jsme již ukázali v tvrzení 4.16, syntaktická kongruence $\sim$ jazyka L je kongruence a platí pro ni (32). Zbývá ukázat, že $\sim$ je nejhrubší kongruencí splňující (32), tzn. pro libovolnou kongruenci $\approx$ na X^* splňující (32) ověřit pro každé $\alpha, \beta \in X^*$ vztah $\alpha \approx \beta \Rightarrow \alpha \sim \beta$. Mějme $\alpha, \beta \in X^*$ takové, že $\alpha \approx \beta$. Pak pro každé $\gamma, \sigma \in X^*$ platí $\gamma\alpha\sigma \approx \gamma\beta\sigma$, protože $\approx$ je kongruence. Z vlastnosti (32) pro $\approx$ pak plyne $(\gamma\alpha\sigma \in L \Leftrightarrow \gamma\beta\sigma \in L)$, odtud dostáváme $\alpha \sim \beta$, protože $\sim$ je syntaktická kongruence.

Syntaktická kongruence $\sim$ je pro daný jazyk a danou abecedu vždy právě jedna. Protože nejhrubší kongruence splňující (32) je také právě jedna, je podle první implikace rovna $\sim$[22]. Tvrzení je pro syntaktické kongruence dokázané.

Důkaz pro pravé syntaktické kongruence je analogický. CBD

Definice 4.20 *Doplňkem jazyka $L \subseteq X^*$ rozumíme jazyk $\bar{L} = \{\alpha \in X^*; \alpha \notin L\}$*

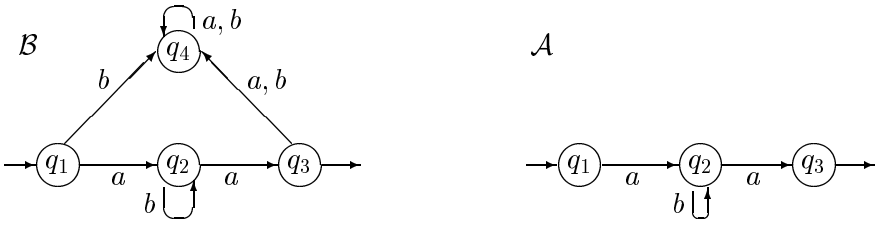
Lemma 4.21 *Regulární jazyky jsou uzavřeny na doplňky.*

Důkaz: Buď $L(\mathcal{A})$ regulární jazyk, přijímaný akceptorem $\mathcal{A} = (Q, X, \delta, q_0, F)$. Dokážeme, že akceptor $\bar{\mathcal{A}} = (Q, X, \delta, q_0, Q \setminus F)$ přijímá doplněk $L(\bar{\mathcal{A}})$ jazyka $L(\mathcal{A})$. To je ovšem triviální, neboť končí-li přijímací výpočet akceptoru $\mathcal{A}$, resp. $\bar{\mathcal{A}}$ pro nějaké slovo $\alpha \in L$ ve stavu q, pak $q \in F \Leftrightarrow q \notin Q \setminus F$, tedy

$$\alpha \in L(\mathcal{A}) \Leftrightarrow \alpha \notin L(\bar{\mathcal{A}})$$

z definice automatu $\bar{\mathcal{A}}$. Doplněk regulárního jazyka je přijímán konečným akceptorem, je tedy regulární. CBD

Vzhledem k předchozímu důkazu je vhodné si trochu připomenout definici (4.1) akceptoru a poznámku o garbage stavech ze strany 27. Zkuste si jako cvičení porovnat jazyky přijímané akceptory $\mathcal{A}$, $\mathcal{B}$ na následujícím obrázku:



Oba akceptory přijímají zřejmě stejný jazyk

$$L = \{ab^n a; n \in N_0\}.$$

Odlišně se ale chovají ke slovům mimo L. $\mathcal{A}$ není podle definice akceptor — přijde-li např. slovo aaa, nemá definovaný přechod z q_3 přes a. $\mathcal{B}$ akceptorem je, tedy má definované všechny přechody — nad slovem aaa skončí v q_4. Budeme-li nyní podle důkazu lemmatu 4.21 konstruovat akceptory přijímající $\bar{L}$, pak vskutku $L(\mathcal{B}) = \bar{L}$, ale $L(\mathcal{A}) \neq \bar{L}$, protože nepřijímá např. slovo aaa. Příčinu hledejte v neúplnosti definice funkce δ v $\mathcal{A}$.

Poučení: Používejte garbage stavy a čtěte řádně definice[23]!

[22] tím jsme se vyhnuli důkazu druhé implikace

[23] abyste nedopadli jako já, když jsem odsoudil důkaz lemmatu 4.21, protože jsem byl upozorněn, že krachuje právě na příkladu jazyka L; není tomu tak, důkaz je dobře!

Lemma 4.22 *Průnik i sjednocení dvou regulárních jazyků je opět regulární jazyk.*

Důkaz: Mějme dva konečné akceptory $\mathcal{A}_i = (Q_i, X_i, \delta_i, q_i, F_i)$, $i = 1, 2$, pro které $X_1 = X_2 = X$. Definujme akceptor

$$\mathcal{A} = (Q_1 \times Q_2, X, \delta, (q_1, q_2), F),$$

kde $\delta((q, q'), x) = (\delta_1(q, x), \delta_2(q', x))$. Vezměme nějaké slovo $\alpha \in X^*$. Přijímací výpočet akceptoru $\mathcal{A}$ nad slovem α nechť skončí ve stavu (q, q'). Lze snadno nahlédnout, že α je přijímáno akceptorem $\mathcal{A}_1$, právě když $q \in F_1$. Obdobně $\alpha \in L(\mathcal{A}_2) \Leftrightarrow q' \in F_2$.

Položíme-li $F = F_1 \times F_2$, pak $\mathcal{A}$ přijme α jen tehdy, pokud α přijme $\mathcal{A}_1$ i $\mathcal{A}_2$, tzn. $\mathcal{A}$ přijímá průnik jazyků $L(\mathcal{A}_1)$ a $L(\mathcal{A}_2)$.

Položíme-li $F = F_1 \times Q_2 \cup Q_1 \times F_2$, pak $\mathcal{A}$ přijme α, pokud přijme α alespoň jeden z akceptorů $\mathcal{A}_1$, $\mathcal{A}_2$, tedy přijímá sjednocení jazyků $L(\mathcal{A}_1)$ a $L(\mathcal{A}_2)$.

Protože $\mathcal{A}$ je konečný, jsou jazyky $L(\mathcal{A}_1) \cup L(\mathcal{A}_2)$ a $L(\mathcal{A}_1) \cap L(\mathcal{A}_2)$ regulární. CBD

Důsledek 4.23 *Regulární jazyky jsou uzavřené na doplňky, konečná sjednocení a konečné průniky.*

Důkaz: Plyne z lemmatu 4.21 a indukcí z lemmatu 4.22. CBD

Povšimněme si nyní následující situace. Mějme akceptor $\mathcal{A} = (Q, X, \delta, q_0, F)$ a slovo $\alpha = a_1 a_2 \dots a_h \in L(\mathcal{A}), |\alpha| = h$ takové, že $h > |Q|$. Je zřejmé, že přijímací výpočet nad slovem α musí projít některými stavy vícekrát. Nechť

$$q_0, q_1, \dots, q_i, \dots, q_j, \dots, q_h$$

je přijímacím výpočtem nad slovem α, tzn. $q_h \in F$. Pak existují čísla i, j taková, že $i < j$ a $q_i = q_j$ (musí nastat, neboť $h > |Q|$). Označme $\alpha_1 = a_1 \dots a_i$, $\alpha_2 = a_{i+1} \dots a_j$, $\alpha_3 = a_{j+1} \dots a_h$. Potom

$$\delta^*(q_0, \alpha_1) = q_i = q_j = \delta^*(q_i, \alpha_2)$$
$$\delta^*(q_j, \alpha_3) = q_h \in F.$$

Lehce nahlédneme, že každé slovo tvaru $\alpha_1 \alpha_2^k \alpha_3$, kde k probíhá N_0, patří do jazyka $L(\mathcal{A})$. Tento intuitivní poznatek formalizuje následující věta:

Věta 4.24 *(Nutná podmínka regularity jazyka) Ke každému regulárnímu jazyku L existuje $n \in N$ takové, že pro všechna slova $\alpha \in L$ délky větší než n existují slova $\alpha_1, \alpha_2, \alpha_3$ taková, že $\alpha_2 \neq \Lambda$, $|\alpha_2| \leq n$, $\alpha = \alpha_1 \alpha_2 \alpha_3$ a*

$$(\forall\, i \in N_0)(\alpha_1 \alpha_2^i \alpha_3 \in L).$$

Důkaz: Vyplývá z předcházejícího odstavce. CBD

Otázka konkrétní (minimální) hodnoty čísla n pro konkrétní jazyk L zůstává však otevřená. Odvození věty napovídá, že bychom měli hledat akceptor $\mathcal{A}$, který by přijímal jazyk L a měl by minimální počet stavů. Podle první části důkazu Myhill-Nerodovy věty má ovšem akceptor $\mathcal{A}$ s minimálním počtem stavů

právě tolik stavů, kolik je rozkladových tříd pravé syntaktické kongruence $\sim$ jazyka L. Je tedy n rovno počtu stavů akceptoru $\mathcal{A}$ (= počet tříd $\sim$).

Poznámka: Větu 4.24 lze s úspěchem použít, chceme-li dokázat, že nějaký jazyk není regulární (ukážeme, že žádné n nevyhovuje). Není však možné tvrdit, že najdeme-li pro nějaký jazyk L číslo n vyhovující větě 4.24, je L regulární!!![24]

Příklad 4.25 *(Důkaz neregularity jazyka) Použitím věty 4.24 dokažte, že jazyk*

$$L = \{a^n b^n; n = 0, 1, \ldots\}$$

není regulární.

Důkaz provedeme sporem. Předpokládejme, že L je regulární. Pak pro něj musí platit věta 4.24, čili existuje n takové, že libovolné slovo α délky větší než n lze rozložit na $\alpha = \alpha_1\alpha_2\alpha_3$ $(0 < |\alpha_2| \leq n)$ tak, že $(\forall\, i \in N_0)(\alpha_1\alpha_2^i\alpha_3 \in L)$. Proč si nevzít třeba $\alpha_0 = a^{n+1}b^{n+1}$. Jak může vypadat rozklad slova α_0? Neboť $0 < |\alpha_2| \leq n$, musí nastat jeden z případů

- $\alpha_2 = a^i, \quad 0 < i \leq n$
- $\alpha_2 = a^i b^j, \quad 0 < i, j,\ i + j \leq n$
- $\alpha_2 = b^j, \quad 0 < j \leq n$

Pokud $\alpha_2 = a^i$, pak $\alpha_1\alpha_3 \notin L$, neboť v $\alpha_1\alpha_3$ je více b než a. Je-li $\alpha_2 = a^i b^j$, pak $\alpha_1\alpha_2\alpha_2\alpha_3 = a^{n+1}b^j a^i b^{n+1} \notin L$. Konečně pro $\alpha_2 = b^j$ je $\alpha_1\alpha_3 \notin L$, neboť počet b je menší než počet a. Pro libovolné n jsme našli α_0 délky větší než n, které nelze rozložit na $\alpha_1\alpha_2\alpha_3$ tak aby všechna slova $\alpha_1\alpha_2^i\alpha_3$ patřila do L, neboli L nesplňuje předpoklady věty 4.24 a tedy není regulární. FINE

4.3 Nedeterministické akceptory

Dosud jsme hovořili pouze o tzv. *deterministických* akceptorech, kde z každého stavu existoval pro každé písmeno vstupní abecedy právě jeden přechod.

Definice 4.26 *Nedeterministický akceptor je pětice (Q, X, δ, I, F), kde*

- *X vstupní abeceda*
- *Q je konečná množina stavů*
- *$\delta : Q \times X \to \mathcal{P}(Q)$ přechodová funkce*[25]
- *$I \subseteq Q$ množina iniciálních stavů*

[24] Takovým neregulárním jazykem je například

$$\{a_1 \ldots a_n \in \{0,1\}^*; a_1 \ldots a_n = a_n \ldots a_1; n \in N_0\},$$

řečeno jedním slovem jazyk palindromů, tzn. slov čtených odpředu stejně jako odzadu. Uvádím tento příklad jen jako doklad předchozí poznámky a bez důkazu, který si zájemci mohou provést jako cvičení na použití Myhill-Nerodovy věty.

[25] někdy je δ definována jako přechodová relace $\delta \subseteq Q \times X \times Q$, což je totéž; množina $\mathcal{P}(Q)$ zde značí potenční množinu množiny Q, neboli množinu všech podmnožin Q

- $F \subseteq Q$ *množina koncových (přijímacích) stavů*

Povšimněme si, že každý deterministický akceptor je zároveň nedeterministický. Naopak to pochopitelně neplatí. Neuvedeme-li proto, zda se jedná o deterministický či nedeterministický akceptor, rozumí se vždy nedeterministický[26].

Dále si povšimněme, že obrazem funkce δ pro nějakou dvojici z $Q \times X$ může být — a také často je — prázdná množina. Narozdíl od deterministických akceptorů (def. 4.1) tedy nemusíme v nedeterministických používat garbage stavy.

Definice 4.27 *Řekneme, že slovo $\alpha = a_1 \dots a_k \in X^*$ je přijímáno N-akceptorem $\mathcal{A}$, jakmile existuje posloupnost stavů $r_0, \dots, r_k \in Q$ taková, že*

- $r_0 \in I$
- $(\forall\, i \in \langle 0, k-1 \rangle)(r_{i+1} \in \delta(r_i, a_{i+1}))$
- $r_k \in F$

Posloupnost $r_0, r_1, \dots, r_n$ se nazývá přijímající výpočet akceptoru $\mathcal{A}$ nad slovem α. Pokud posloupnost $r_0, r_1, \dots, r_n$ splňuje alespoň první dvě podmínky, mluvíme o ní jako o výpočtu akceptoru $\mathcal{A}$ nad α.

Jazyk $L(\mathcal{A})$ přijímaný akceptorem $\mathcal{A}$ je množina všech slov přijímaných akceptorem $\mathcal{A}$ (tzn. $L(\mathcal{A}) = \{\alpha; \alpha$ je přijímáno $\mathcal{A}\}$)

Tvrzení 4.28 *Nedeterministické akceptory přijímají právě regulární jazyky.*

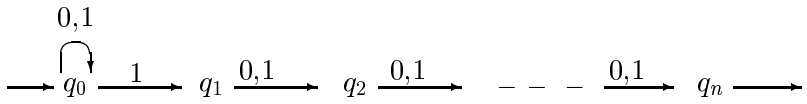
Důkaz: Každý regulární jazyk je přijímán deterministickým akceptorem (viz definice 4.2) a každý D-akceptor je zároveň nedeterministický. Zbývá dokázat implikace zleva doprava, totiž že pro libovolný N-akceptor $\mathcal{A} = (Q, X, \delta, I, F)$ je $L(\mathcal{A})$ regulární jazyk. Jinými slovy, převedeme N-akceptor $\mathcal{A}$ na D-akceptor $\mathcal{A}'$, který je s ním ekvivalentní. Pro dané vstupní slovo $\alpha = a_1 \dots a_k$ uvažujme posloupnost množin $A_i \subseteq Q$:

$$\begin{aligned} A_0 &= I \\ A_1 &= \{q; (\exists\, r \in A_0)(q \in \delta(r, a_1))\} \\ &\vdots \\ A_k &= \{q; (\exists\, r \in A_{k-1})(q \in \delta(r, a_k))\} \end{aligned}$$

Je zřejmé, že $\alpha \in L(\mathcal{A})$, právě když $A_k \cap F \neq \emptyset$. Definujme proto D-akceptor $\mathcal{A}' = (Q', X, \delta', I, F')$, kde

$$\begin{aligned} Q' &= \mathcal{P}(Q) \\ F' &= \{A \subseteq Q; A \cap F \neq \emptyset\} \\ \delta'(A, x) &= \{q \in Q; (\exists\, r \in A)(q \in \delta(r, x))\}. \end{aligned}$$

[26] vzhledem k neskladnosti slov (ne)deterministický budeme napříště používat zkratek N-akceptor=nedeterministický akceptor, D-akceptor=deterministický akceptor



Obr. 6: N-akceptor přijímající L_n

Akceptor $\mathcal{A}'$ je nesporně deterministický[27] a konečný a přijímá stejný jazyk jako $\mathcal{A}$[28], tedy jazyk $L(\mathcal{A}) = L(\mathcal{A}')$ je regulární. CBD

Zamysleme se nyní nad složitostí, resp. počtem stavů D- a N-akceptorů. Na příkladě ukážeme, že N-akceptory jsou značně jednodušší[29]. Vezměme jazyk $L_n = \{\alpha \in \{0,1\}^*, n\text{-té písmeno od konce je } 1\}$. Tvrdím, že existuje N-akceptor s $n+1$ stavy přijímající jazyk L_n, kdežto nejmenší D-akceptor pro jazyk L_n má 2^n stavů. Důkaz je snadný. Zmíněný N-akceptor ukazuje obrázek 6, kde $F = \{q_n\}$.

Zbývá ukázat, že libovolný D-akceptor, který má přijímat L_n, musí mít nejméně 2^n stavů, tzn. podle první části důkazu Myhill-Nerodovy věty (4.17) stačí dokázat, že pravá syntaktická kongruence $\sim$ jazyka L_n má alespoň 2^n tříd. Vezměme dvě slova $\alpha = a_1, \ldots, a_k, \beta = b_1, \ldots, b_\ell \in \{0,1\}^*$ a jejich suffixy $\gamma_\alpha, \gamma_\beta$ délky n[30]. Dokážeme ekvivalenci

$$(33)\quad (\gamma_\alpha = \gamma_\beta) \Longleftrightarrow (\alpha \sim \beta)$$

To je snadné. Je-li $\gamma_1 = \gamma_2$, pak se posledních n písmen slov α, β shoduje. Přidáme-li za obě slova α, β suffix σ libovolné délky, n-tá písmena slov α, β od konce se zřejmě shodují, neboli

$$(\forall\, \sigma \in \{0,1\}^*)(\alpha\sigma \in L_n \Leftrightarrow \beta\sigma \in L_n),$$

tedy $\alpha \sim \beta$.

Je-li $\gamma_\alpha \neq \gamma_\beta$, pak $(\exists\, i < n)(a_{k-n+i} \neq b_{\ell-n+i})$. Zvolíme-li slovo σ velikosti i, pak n-tá písmena slov $\alpha\sigma, \beta\sigma$ od konce jsou různá, tedy $\alpha \not\sim \beta$.

Dokázali jsme ekvivalenci (33), ze které již snadno plyne, že $\sim$ má 2^n tříd. Stačí si uvědomit, že navzájem různých suffixů délky n nad abecedou $\{0,1\}$ je právě 2^n. Obrázek 7 ukazuje D-akceptor s osmi stavy přijímající L_3[31].

Definice 4.29 *Mějme jazyky L_1, L_2 nad abecedou X. Složením (konkatenací) jazyků L_1, L_2 rozumíme jazyk $L_1L_2 = \{\alpha\beta; \alpha \in L_1, \beta \in L_2\}$.*

Lemma 4.30 *Mějme L_1, L_2 regulární jazyky. Potom L_1L_2 je také regulární.*

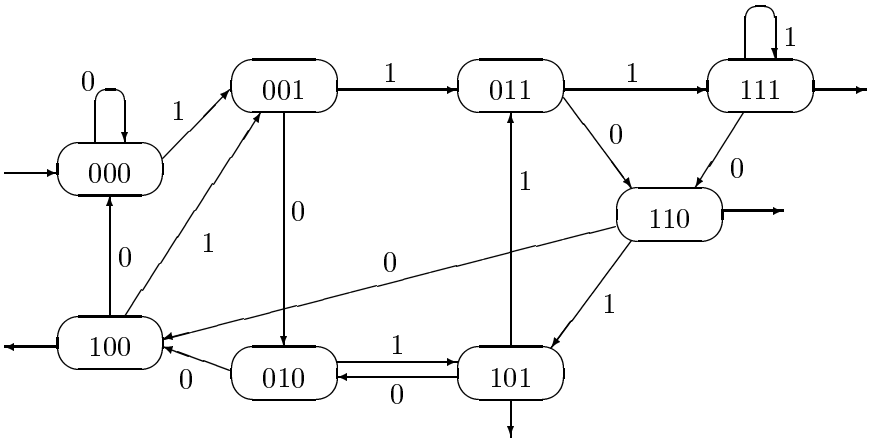
[27] pro každou množinu $A \subseteq Q$ je jednoznačně definována přechodová funkce

[28] přijímací výpočet v $\mathcal{A}$ je vlastně transverzálou přijímacího výpočtu v $\mathcal{A}'$

[29] Ne že by každý N-akceptor měl méně stavů než D-akceptor nebo něco podobného. To samozřejmě neplatí! Ale sestrojíme-li pro nějaký jazyk N- a D-akceptor s nejmenším možným počtem stavů, má zpravidla N-akceptor stavů výrazně méně.

[30] Jsou-li α nebo β kratší než n, doplníme je zpředu písmenem 0.

[31] stavy znázorňují (kódují) aktuální příponu čteného slova



Obr. 7: D-akceptor přijímající L_3

Důkaz: Nechť $L_i = L(\mathcal{A}_i)$, kde pro $i = 1, 2$ jsou $\mathcal{A}_i = (Q_i, X, \delta_i, I_i, F_i)$ nedeterministické akceptory, splňující podmínku $Q_1 \cap Q_2 = \emptyset$. Definujme akceptor $\mathcal{A} = (Q_1 \cup Q_2, X, \delta, I, F_2)$, kde

- $I = I_1$, pokud $\Lambda \notin L_1$,
- $I = I_1 \cup I_2$, pokud $\Lambda \in L_1$.

Funkce δ je pro všechna $x \in X$ definována následovně:

$$(\forall\, q \in Q_2)(\delta(q, x) = \delta_2(q, x))$$
$$(\forall\, q \in Q_1)(\delta_1(q, x) \cap F_1 = \emptyset \Rightarrow \delta(q, x) = \delta_1(q, x))$$
$$(\forall\, q \in Q_1)(\delta_1(q, x) \cap F_1 \neq \emptyset \Rightarrow \delta(q, x) = \delta_1(q, x) \cup I_2)$$

Dokážeme $L(\mathcal{A}) = L_1 L_2$. Vezměme nejprve $\gamma = a_1 \dots a_k \in L(\mathcal{A})$. Pro γ existuje přijímací výpočet

$$\overbrace{r_0, r_1, \dots, r_{i-1}}^{\in R}, \overbrace{r_i, \dots, r_k \in F_2}^{\in Q_2},$$

kde $r_\ell \in \delta(r_{\ell-1}, a_\ell)$ a

- buď $R = Q_1$, $r_0 \in I_1$ a $r_i \in I_2$,
- nebo $R = Q_2$[32], $q_0 \in I_2$ a $\Lambda \in L_1$.

V druhém případě zřejmě $\gamma \in L_2$. Pak ale také $\gamma \in L_1 L_2$, neboť $\Lambda \in L_1$. V prvním případě probíhá první část výpočtu podle definice δ pouze v části Q_1, druhá pouze v Q_2. Přechod mezi Q_1 a Q_2 je možný jen když $\delta_1(r_{i-1}, a_i) \cap F_1 \neq \emptyset$, tedy slovo $\alpha = a_1, \dots, a_i$ je přijímáno v $\mathcal{A}_1$. Zbytek slova γ, tzn. $\beta = a_{i+1}, \dots, a_k$ je přijímáno v $\mathcal{A}_2$, neboť výpočet nad β začal v $r_i \in I_2$, procházel pouze stavy Q_2 a skončil v množině F_2[33]. Pro γ jsme našli rozklad $\alpha\beta$ takový, že $\alpha \in L_1, \beta \in L_2$,

[32] tzn. do výpočtu se žádný stav z Q_1 nezapojí.

[33] Pokud by platilo $\Lambda \in L_1$, pak při použití startovacího stavu q_2 celý výpočet probíhá v Q_2 a získáme tak rozklad $\gamma = \beta = \Lambda\beta$

neboli $L(\mathcal{A}) \subseteq L_1 L_2$.

Opačná inkluze není o mnoho složitější. Vezměme slova $\alpha = a_1 \dots a_k \in L_1$ a $\beta = b_1 \dots b_\ell \in L_2$. Je-li $\alpha = \Lambda$, pak přijímací výpočet $\mathcal{A}_2$ nad β je také přijímací výpočet $\mathcal{A}$ nad $\beta = \Lambda\beta \in L(\mathcal{A})$. Buď *alpha* $\neq \Lambda$. Dále nechť $r_0, \dots, r_{k-1}, r'_k$ je pro nějaké $r'_k \in F_1$, $r_0 \in I_1$ přijímací výpočet α v $\mathcal{A}_1$ a $r_k, \dots, r_{k+\ell}$ je pro nějaké $r_k \in I_2$, $r_{k+\ell} \in F_2$ přijímací výpočet β v $\mathcal{A}_2$. Potom $r_0, \dots, r_{k-1}, r_k, \dots, r_{k+\ell}$ je z definice δ přijímacím výpočtem slova $\alpha\beta$ v $\mathcal{A}$[34]. Dokázali jsme $L_1 L_2 \subseteq L(\mathcal{A})$, což s opačnou inkluzí z první části důkazu dává kýženou rovnost $L(\mathcal{A}) = L_1 L_2$. Neboť $\mathcal{A}$ je nesporně konečný N-akceptor, je $L_1 L_2$ regulární jazyk. CBD

Když teď máme operaci konkatenace, je přirozené definovat také „mocninu“ nějakého jazyka:

Definice 4.31 *Definujme následující jazyky:*

$$\begin{aligned} L^0 &= \{\Lambda\} \\ L^1 &= L \end{aligned}$$

i-tou mocninou jazyka L rozumíme jazyk definovaný předpisem

$$L^i = LL^{i-1}$$

Dále definujme $L^* = \bigcup_{i \in N_0} L^i$. Operaci * budeme nazývat iterací jazyka L.

Lemma 4.32 *Je-li L regulární jazyk, pak i L^* je regulární.*

Proč je třeba toto lemma dokazovat, když máme důsledek 4.23, který zajišťuje uzavřenost regulárních jazyků na konečná sjednocení? Protože sjednocení v definici L^* běží přes přirozená čísla a je tedy nekonečné[35].

Důkaz: Nechť $L = L(\mathcal{A})$, $\mathcal{A} = (Q, X, \delta, I, F)$ a existuje prvek $s \notin Q$. Definujme akceptor $\mathcal{A}_1 = (Q \cup \{s\}, X, \delta_1, I \cup \{s\}, F \cup \{s\})$, kde pro všechna $x \in X$

$$\begin{aligned} \delta_1(q, x) &= \begin{cases} \delta(q, x) & \text{pokud } (\forall\, q \in Q)(\delta(q, x) \cap F = \emptyset) \\ \delta(q, x) \cup I & \text{pokud } (\forall\, q \in Q)(\delta(q, x) \cap F \neq \emptyset) \end{cases} \\ \delta_1(s, x) &= \emptyset \end{aligned}$$

Tvrdím, že $L^* = L(\mathcal{A}_1)$. Buď nejprve $\alpha \in L^*$. Je-li $\alpha = \Lambda$, pak α je přijímáno v $\mathcal{A}_1$ stavem s[36]. Je-li $\alpha = \alpha_1 \alpha_2 \dots \alpha_k, \alpha_i \in L$, pak pro každé α_i existuje v $\mathcal{A}$ přijímací výpočet $I \ni r_{i1}, r_{i2}, \dots, r_{in_i} \in F$, kde $r_{i(j+1)} \in \delta(r_{ij}, a_{ij})$. Potom ale z definice δ_1 je $r_{11}, r_{12}, \dots, r_{1(n_1-1)}, r_{21}, \dots, r_{kn_k}$ přijímacím výpočtem α, tzn. $\alpha \in L(\mathcal{A}_1)$.

Na druhou stranu. Buď $\alpha = a_1 \dots a_n \in L(\mathcal{A}_1)$. Je-li $\alpha = \Lambda$, je $\alpha \in L^*$. Není-li $\alpha = \Lambda$, existuje přijímací výpočet $r_0, \dots, r_n$, kde $r_{j+1} \in \delta_1(r_j, a_{j+1})$, $r_0 \in I$,

[34] pro ty, kteří to nevidí jako trivialitu, připojuji vysvětlení: výpočet začíná v $r_0 \in I_1$. Až do stavu r_{k-1} se pohybuje pouze v Q_1. Potom ale $\delta_1(r_{k-1}, a_k) = r'_k \in F_1$, tedy z definice δ můžeme přejít do $r_k \in I_2$ v části Q_2, ve které už zůstaneme až do $r_{k+\ell} \in F_2$. Platí $(\forall\, 0 < i \leq k + \ell)(r_i \in \delta(r_{i-1}, a_i))$, tedy r_i je přijímací výpočet $\mathcal{A}$ nad slovem $\alpha\beta$

[35] Zamyslete se nad tím, proč je L^* opravdu jazykem, neboli proč jsou všechna jeho slova konečná. Je to sice velmi jednoduché, ale možná vhodné explicitně si uvědomit.

[36] od toho tam ten stav taky máme, že ...

$r_n \in F$. Vezměme všechny indexy $i_1 < \ldots < i_k$, pro něž $r_{i_j+1} \notin \delta(r_{i_j}, a_{i_j+1})$, dodefinujme $i_0 = 0, i_{k+1} = n$ a rozdělme přijímací výpočet slova α na $k+1$ částí (j-tá část obsahuje stavy $r_{i_j+1}, r_{i_j+2}, \ldots, r_{i_{j+1}-1}; j \in \langle 0, k\rangle$). Přechody v rámci jedné části vždy patří do funkce δ, přechody mezi jednotlivými částmi nikoli. Musely tedy z definice δ_1 nastat v případě, kdy $\delta(r_{i_j-1}, a_{i_j}) \cap F \neq \emptyset$, tzn. pro každé r_{i_j-1} existuje stav $r'_{i_j} \in F$ takový, že

$$(\forall\, j \in N)(j \leq k+1 \Rightarrow (r'_{i_j} \in \delta(r_{i_j-1}, a_{i_j}))).$$

Dále lze tvrdit $(\forall\, j \in \langle 1, k\rangle)(r_{i_j} \in I)$, opět z definice δ_1. Potom ale posloupnosti $r_{i_j}, r_{i_j+1}, \ldots, r_{i_{j+1}-1}, r'_{i_{j+1}}$ jsou[37] pro $j \in \langle 0, k\rangle$ přijímacími výpočty akceptoru $\mathcal{A}$ nad slovy

$$\begin{aligned} \alpha_1 &= a_1 \ldots a_{i_1} \\ \alpha_2 &= a_{i_1+1} \ldots a_{i_2} \\ &\vdots \\ \alpha_{k+1} &= a_{i_k+1} \ldots a_{i_{k+1}}. \end{aligned}$$

Slovo α jsme rozložili na $\alpha_1 \ldots \alpha_{k+1}, \alpha_i \in L(\mathcal{A}) = L$, kde $k+1 \leq |\alpha|$, tedy konečné číslo, neboli $\alpha \in L^*$. Máme opačnou inkluzi, tedy $L^* = L(\bar{\mathcal{A}})$. CBD

Definice 4.33 *Zrcadlovým obrazem slova* $\alpha = a_1 \ldots a_n$ *rozumíme slovo*

$$\alpha^R = a_n \ldots a_1$$

Zrcadlovým obrazem jazyka L *rozumíme jazyk*

$$L^R = \{\alpha^R; \alpha \in L\}$$

Lemma 4.34 *Zrcadlový obraz regulárního jazyka je regulární.*

Důkaz: Provedeme pouze neformálně (formální zápis je možné zkusit jako cvičení). Máme-li akceptor $\mathcal{A}$, který přijímá jazyk L, obrátíme v něm všechny šipky na opačně orientované a zaměníme množiny vstupních a výstupních stavů. Takto upravený akceptor je zřejmě konečný a přijímá jazyk L^R. CBD

Poznatky předcházejících lemmat shrnuje následující věta

Věta 4.35 *(Kleene) Regulární jazyky tvoří nejmenší třídu*[38] *jazyků, která*

1. *obsahuje všechny konečné jazyky*[39]
2. *je uzavřena na konečná sjednocení, konkatenaci a iteraci*

[37] při definici $r'_n = r_n$

[38] ve smyslu teorie množin

[39] Tato podmínka je ekvivalentní podmínce „pro každou abecedu X obsahuje tzv. elementární jazyky $\emptyset, \{\Lambda\}$ a $\{x\}$ pro všechna $x \in X$“, neboť za předpokladu platnosti bodu 2 lze každý konečný jazyk vyjádřit z uvedených jazyků operacemi z bodu 2

Důkaz: Že každý jazyk, patřící do třídy specifikované body 1 a 2, je regulární, vyplývá z důsledku 4.23 a lemmat 4.30 a 4.32. S opačnou inkluzí je to poněkud složitější. Vezměme si L regulární jazyk a snažme se dokázat, že se dá vyjádřit operacemi z bodu 2 z jazyků $\emptyset$, $\{\Lambda\}$, $\{x\}$ z bodu 1. Nechť $L \subseteq X^*$, $L = L(\mathcal{A})$, $\mathcal{A} = (Q, X, \delta, q_0, F)$ nechť je D-akceptor s n stavy, očíslovanými indexy $0, \ldots, n-1$[40]. Vyjádříme $L(\mathcal{A})$ pomocí elementárních jazyků. Definujme pro tento účel jazyky typu

$$L_{ij} = \{\alpha \in X^*; \delta^*(q_i, \alpha) = q_j\}$$

Je okamžitě vidět, že $L = \bigcup_{q_j \in F} L_{0j}$. Dále definujme

$$\begin{aligned} L_{ij}^k &= \{\alpha \in X^*; \delta^*(q_i, \alpha) = q_j, \\ &\quad (\forall\, \alpha' \text{ vlastní prefix } \alpha)(\delta^*(q_i, \alpha') = q_\ell \Rightarrow \ell < k)\} \end{aligned}$$

Řečeno lidsky, v L_{ij}^k jsou všechna slova, která převedou q_i na q_j přes stavy q_ℓ takové, že $\ell < k$. Zřejmě platí $L_{ij}^{n+1} = L_{ij}$. Indukcí podle k dokážeme, že pro libovolné k je L_{ij}^k možné vytvořit z elementárních jazyků operacemi z bodu 2.

1. $(k = 0)$:
   (a) $i = j$, potom $L_{ii}^0 = \{a \in X; \delta(q_i, a) = q_i\} \cup \{\Lambda\}$
   (b) $i \neq j$, potom $L_{ij}^0 = \{a \in X; \delta(q_i, a) = q_j\}$

   Zřejmě tedy L_{ij}^0 je konečným sjednocením elementárních jazyků.

2. $(k \to k+1)$: tvrdím, že

$$L_{ij}^{k+1} = \underbrace{L_{ij}^k \cup L_{ik}^k (L_{kk}^k)^* L_{kj}^k}_{L\#}$$

   Dokážeme tento vztah postupným důkazem obou inkluzí

   (a) $L^\# \subseteq L_{ij}^{k+1}$. Mějme $\alpha \in L^\#$. Potom
      i. buď $\alpha \in L_{ij}^k$. Pak pro všechna α' libovolné vlastní prefixy α platí $\delta^*(q_i, \alpha') = q_\ell$, kde $\ell < k < k+1$, tedy $\alpha \in L_{ij}^{k+1}$,
      ii. nebo $\alpha = \beta\alpha_1 \ldots \alpha_m\gamma$, kde $\beta \in L_{ik}^k, \alpha_i \in L_{kk}^k, \gamma \in L_{kj}^k$. Je-li α' vlastní prefix α, může nastat právě jeden z případů
         A. α' je vlastní prefix β, potom $\delta^*(q_i, \alpha') = q_\ell, \ell < k < k+1$.
         B. $\alpha' = \beta$, pak $\delta^*(q_i, \alpha') = q_k, k < k+1$.
         C. $\alpha' = \beta\alpha_1 \ldots \alpha_j\alpha''$, kde α'' je vlastní prefix α_{j+1}. Potom
$$\begin{aligned} &\delta^*(q_i, \alpha') = \delta^*(\delta^*(q_i, \beta), \alpha_1 \ldots \alpha_j\alpha'') = \\ &\delta^*(q_k, \alpha_1 \ldots \alpha_j\alpha'') = \ldots = \\ &\delta^*(q_k, \alpha'') = q_\ell, \ell < k < k+1. \end{aligned}$$
         D. $\alpha' = \beta\alpha_1 \ldots \alpha_j$, pak $\delta^*(q_i, \alpha') = \delta^*(q_k, \alpha_j) = q_k, k < k+1$.

[40] tedy $Q = \{q_0, q_1, \ldots, q_{n-1}\}$

E. $\alpha' = \beta\alpha_1 \ldots \alpha_m\gamma'$, γ' vlastní prefix γ. Potom platí
$$\delta^*(q_i, \alpha') = \delta^*(q_k, \gamma') = q_\ell,$$
kde $\ell < k < k+1$.

Slovo α patří do L_{ij}^{k+1}, neboť $\delta^*(q_i, \alpha) = q_j$ a výpočty všech vlastních prefixů končí ve stavech $q_\ell, \ell < k+1$. Víme $L^\# \subseteq L_{ij}^{k+1}$.

(b) $L_{ij}^{k+1} \subseteq L^\#$. Mějme $\alpha \in L_{ij}^{k+1}$. Označme α_s prefix α délky s. Zřejmě můžeme najít posloupnost všech indexů $s_1 < s_2 < \ldots < s_t$ takových, že $(\forall\, m \leq t)(\delta^*(q_i, \alpha_{s_m}) = q_k)$. Nyní položme

i. $\beta = \alpha_{s_1} \Rightarrow \beta \in L_{ik}^k$

ii. $(\forall\, m < t)(\alpha_{s_{m+1}} = \alpha_{s_m}\sigma_m \Rightarrow \sigma_m \in L_{kk}^k)$

iii. $\alpha = \alpha_{s_t}\gamma \Rightarrow \gamma \in L_{kj}^k$

Zjevně $\alpha = \beta\sigma_1 \ldots \sigma_{t-1}\gamma$, tedy
$$\alpha \in L_{ik}^k (L_{kk}^k)^{t-1} L_{kj}^k \subset L^\#,$$
kdykoli $t > 0$. Je-li $t = 0$, potom výpočet neprošel stavem q_k a musel tedy projít pouze stavy $q_\ell, \ell < k$, tzn. $\alpha \in L_{ij}^k \subset L^\#$. Víme nyní i $L_{ij}^{k+1} \subseteq L^\#$

Obě inkluze dávají rovnost $L^\# = L_{ij}^{k+1}$, tedy z indukčního předpokladu lze L_{ij}^{k+1} sestavit z elementárních jazyků operacemi z bodu 2 znění věty. Neboť důkaz indukcí platí pro všechna k, tím spíš pro $k = n+1$, a dále víme $L_{0j}^{n+1} = L_{0j}$ a $L = \bigcup_{q_j \in F} L_{0j}$, kteréžto sjednocení je konečné a jazyky v tomto sjednocení získány jsou jen a pouze operacemi z bodu 2 znění věty, je i jazyk L sestrojitelný zmíněnými operacemi z elementárních jazyků. Protože kromě regularity na jazyk L požadavků nižádných kladeno nebylo, lze říci, že každý regulární jazyk lze složit z elementárních jazyků operacemi konečného sjednocení, iterací a konkatenací. CBD

K popisu regulárních jazyků můžeme teď používat pohodlnější nástroj, než jsou složité množinové konstrukce:

Definice 4.36 *Regulární výraz je definován indukcí*

1. $\emptyset, \Lambda$ *a každá proměnná jsou regulární výrazy*
2. α, β *regulární výrazy* $\Longrightarrow \alpha + \beta, \alpha \cdot \beta, \alpha^*$ *jsou regulární výrazy*
3. *jiné regulární výrazy nejsou*

Každému regulárnímu výrazu α odpovídá jazyk nad abecedou proměnných, který označíme $L(\alpha)$. Operaci $+$ odpovídá sjednocení, $\cdot$ konkatenace a * iterace. Kleeneova věta říká, že regulárnímu výrazu odpovídá regulární jazyk a naopak, neboli že regulární výrazy popisují právě regulární jazyky.

Poznámka: Kleeneovu větu lze vyslovit pro regulární jazyky nad konkrétní abecedou: „Regulární jazyky nad libovolnou abecedou X tvoří ...“. Jak je ale snadno vidět z důkazu Kleeneovy věty, jsou obě znění navzájem ekvivalentní.

Mimochodem, Kleeneova věta (resp. její důkaz) poskytuje návod, jak k danému akceptoru $\mathcal{A}$ sestrojit regulární výraz popisující jazyk $L(\mathcal{A})$. Stačí pro všechna $q_j \in F$ sestrojit indukcí regulární výrazy pro jazyky L_{0j} a pospojovat je mezi sebou operací $+$. Netvrdím však, že tento postup je realizovatelný ve všech případech v „rozumném“ čase.

Příklad 4.37 *(Regulární výraz) Určete jazyk, který odpovídá regulárnímu výrazu*

$$v = a^* \cdot (b \cdot c)^* + b \cdot (a \cdot c^*)^* \tag{34}$$

Výrazu v odpovídá regulární jazyk $L(v)$ nad abecedou $X = \{a, b, c\}$. K popisu $L(v)$ použijeme formuli (34), kde $+$ nahradíme sjednocením a $\cdot$ konkatenací. Sekvence x^* se nahradí sekvencemi $x^i, i = 0, 1, \ldots$[41]. A výsledek? Tady je:

$$L(v) = \{a^i(bc)^j; i, j \in N_0\} \cup \{b(ac^i)^j; i, j \in N_0\}$$

FINE

Poznámka: Vzhledem ke korespondenci operací $\cdot$ a konkatenace se zpravidla $\cdot$ nahrazuje operací konkatenace již v regulárních výrazech (vynechává se). Regulární výraz v z předchozího příkladu by pak měl tvar

$$v = a^*(bc)^* + b(ac^*)^*.$$

Když teď víme, že regulární jazyky a regulární výrazy jsou v jedno–jednoznačném vztahu, můžeme se zajímat o to jak k regulárnímu výrazu sestrojit akceptor, který by přijímal jemu odpovídající regulární jazyk.

Algoritmus 2 *(Konstrukce automatů pro regulární výrazy)*

Řekněme, že máme dány regulární výrazy $\alpha_1, \ldots, \alpha_n$ nad abecedou X. Zkonstruujeme k nim akceptory $\mathcal{A}_i = (Q, X, \delta, q_0, F_i)$ takové, že $L(\mathcal{A}_i)$ bude jazyk popsaný výrazem α_i. Obecný postup posléze ozřejmíme na příkladu:

1. Vytvoříme nové regulární výrazy $\alpha'_1, \ldots, \alpha'_n$, takže k i-té proměnné ve výrazu α_j přidáme index $i + n_j$, kde n_j je součet výskytů všech proměnných ve výrazech $\alpha_1, \ldots, \alpha_{j-1}$, pro všechna i, j. Když i-tá proměnná v α_j byla x, pak řekneme, že proměnná x_{i+n_j} vznikla z proměnné x. Abecedu tvořenou všemi indexovanými proměnnými označíme X'.

2. Vytvoříme množiny[42]

   - $C_I = \{yz; (\exists\, i \in \langle 1, n\rangle)(\exists\, u, v \in (X')^*)(uyzv \in L(\alpha'_i))\}$
   - $C_{II} = \{y; (\exists\, i \in \langle 1, n\rangle)(\exists\, u \in (X')^*)(yu \in L(\alpha'_i))\}$
   - $C^i_{III} = \{y; (\exists\, u \in (X')^*)(uy \in L(\alpha'_i))\}$ pro $i = 1, \ldots, n$

3. Sestrojíme akceptory $\mathcal{A}_i = (Q, X, \delta, q_0, F_i)$ pro $i = 1, \ldots, n$:

[41] iterace jednoprvkového jazyka $\{x\}$ jsou všechny posloupnosti x^i pro $i \in N_0$; x značí libovolné písmeno z X

[42] pro $y, z \in X'$

- $Q' = \{q_0\} \cup \mathcal{P}(X')$
- $\delta' : Q' \times X \to Q'$
  - $\delta'(q_0, x) = \{y \in C_{II}; y \text{ vzniklo z } x\}$
  - $\delta'(A, x) = \{z; z \text{ vzniklo z } x, (\exists\, y \in A)(yz \in C_I)\}$
- Q je podmnožina Q' dosažitelná z q_0
- $\delta : Q \times X \to Q$ je restrikce δ' na $Q \times X$
- $$F_i = \begin{cases} \{A \in Q; A \cap C^i_{III} \neq \emptyset\} \text{ když } \Lambda \notin L(\alpha_i) \\ \{A \in Q; A \cap C^i_{III} \neq \emptyset\} \cup \{q_0\} \text{ když } \Lambda \in L(\alpha_i) \end{cases}$$
  pro $i = 1, \ldots, n$

Příklad 4.38 *(Regulární výrazy) Sestrojte akceptory k regulárním výrazům*

$\alpha_1 = ab^*a$
$\alpha_2 = ac + b^*ab^*$

K řešení použijeme algoritmu 2:

1. $X = \{a, b, c\}, \alpha'_1 = a_1 b^*_2 a_3, \alpha'_2 = a_4 c_5 + b^*_6 a_7 b^*_8$;
   $X' = \{a_1, b_2, a_3, a_4, c_5, b_6, a_7, b_8\}$
2. - $C_I = \{a_1b_2, b_2a_3, a_1a_3, a_4c_5, b_6a_7, a_7b_8, b_2b_2, b_6b_6, b_8b_8\}$
   - $C_{II} = \{a_1, a_4, b_6, a_7\}$
   - $C^1_{III} = \{a_3\}, C^2_{III} = \{c_5, a_7, b_8\}$
3. Definujme
   - $Q = \{q_0, \emptyset, \{a_1, a_4, a_7\}, \{b_6\}, \{a_3\}, \{b_2, b_8\}, \{c_5\}, \{a_7\}, \{b_8\}\}$
   - funkci δ tabulkou

$Q \backslash X$	a	b	c
$\to q_0$	$\{a_1, a_4, a_7\}$	$\{b_6\}$	$\emptyset$
$\{a_1, a_4, a_7\}$	$\{a_3\}$	$\{b_2, b_8\}$	$\{c_5\}$
$\{b_6\}$	$\{a_7\}$	$\{b_6\}$	$\emptyset$
$\{a_3\}$	$\emptyset$	$\emptyset$	$\emptyset$
$\{b_2, b_8\}$	$\{a_3\}$	$\{b_2, b_8\}$	$\emptyset$
$\{c_5\}$	$\emptyset$	$\emptyset$	$\emptyset$
$\{a_7\}$	$\emptyset$	$\{b_8\}$	$\emptyset$
$\{b_8\}$	$\emptyset$	$\{b_8\}$	$\emptyset$
$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$

   - Položme
     - $F_1 = \{\{a_3\}\}$
     - $F_2 = \{\{a_1, a_4, a_7\}, \{b_2, b_8\}, \{c_5\}, \{a_7\}, \{b_8\}\}$

Tím je příklad vyřešen! FINE

Poznámka: Máme-li dva regulární výrazy α, β, pak jazyk

$$L = L(\alpha) \cap L(\beta) = \{\gamma; \gamma \in L(\alpha) \wedge \gamma \in L(\beta)\}$$

je zřejmě také regulární. Akceptor, přijímající L, si jistě každý dokáže navrhnout sám jako cvičení[43].

[43] Nehledě na to, že zmíněný akceptor byl již sestrojen v důkazu lemmatu 4.22 na straně 37

4.4 Dvoucestné automaty

Poznámka: Velmi často se používá také označení *dvousměrné automaty*, které je možná výstižnější.

Viděli jsme, že konečný automat je kontrolní jednotka spolu s hlavou na vstupní pásce. Hlava čte vstupní slovo a podle toho se mění stav kontrolní jednotky. V každém taktu hlava přečte jedno políčko a pak se posune o jedno políčko vpravo.

Nedeterminismus umožnil současné prohledávání více výpočtů najednou a nedeterministický automat přijímá vstupní slovo, když je alespoň jeden ze sledovaných výpočtů přijímající.

Jiné důležité zobecnění se týká způsobu získávání informace ze vstupní pásky. Toto zobecnění umožní, aby se čtecí hlava na vstupní páskce pohybovala oběma směry. Takové automaty budeme nazývat dvoucestné automaty a ukážeme, že přijímají opět jen regulární jazyky. Uvedené zobecnění dovoluje efektivnější získávání informací a tím i zmenšení řídící jednotky (počtu stavů), ale neumožní zpracovávat více informací[44]. Dvoucestný automat přijímá dané slovo, pokud existuje alespoň jeden přijímající výpočet, tj. výpočet, který začíná výpočet na prvním políčku v nějakém iniciálním stavu a končí v nějakém přijímajícím stavu a hlava je na prvním políčku napravo od vstupního slova. Formálně definujme:

Definice 4.39 *Dvoucestný nedeterministický automat je pětice (Q, X, δ, I, F), kde Q, X, I, F má stejný význam jako pro nedeterministický automat. Funkce δ je definována obdobně jako u N-akceptorů $\delta : Q \times X \to \mathcal{P}(Q \times \{-1, 0, 1\})$.*

Řekneme, že $\mathcal{A}$ je deterministický dvoucestný automat, když $|I| = 1$ a pro každé $q \in Q$ a $x \in X$ je $|\delta(q, x)| = 1$. Proto zapisujeme dvoucestný deterministický automat jako pětici (Q, X, δ, q_0, F), kde Q, X, q_0 a F má stejný význam jako pro deterministický akceptor a $\delta : Q \times X \to Q \times \{-1, 0, 1\}$ je přechodová funkce.

Na vstupní pásce je každá pozice mimo vstupní slovo prázdná[45]*.*

Výpočet dvoucestného automatu nad slovem $\alpha = a_1 \dots a_n$ je posloupnost dvojic $(r_0, i_0), \dots, (r_k, i_k)$, kde r_j jsou stavy automatu, i_j jsou celá čísla a platí $(r_{j+1}, i_{j+1} - i_j) \in \delta(r_j, i_j)$ pro každé $j = 0, \dots, k-1$. Výpočet je přijímající, když $i_0 = 1$, $i_k = n + 1$, $r_0 \in I$ a $r_k \in F$. Slovo α je přijímáno dvoucestným automatem $\mathcal{A}$, když existuje přijímající výpočet automatu $\mathcal{A}$ nad slovem α. Jazyk přijímaný dvoucestným automatem $\mathcal{A}$ je množina $L(\mathcal{A})$ všech slov přijímaných automatem $\mathcal{A}$.

Neformálně řečeno, dvojice (r_j, i_j) ve výpočtu automatu $\mathcal{A}$ nad slovem α říká, že $\mathcal{A}$ je v j-tém taktu ve stavu r_j a čte i_j-té políčko.

Věta 4.40 *Jazyk L je přijímán nějakým dvoucestným automatem, právě když je regulární.*

[44] podobně jako N-akceptory byly efektivnější než D-akceptory

[45] Pro prázdné políčko vstupní pásky není přechodová funkce definována. Později (v kapitole o Turingových strojích) uvidíme, že pro některé implementace automatů je vhodné definovat „prázdný symbol“, který bude stát na všech políčkách mimo vstupní slovo.

Důkaz: Mezi dvoucestné automaty patří i nám dobře známé „normální“ (jednocestné) D-akceptory, takže každý regulární jazyk je přijímán dvoucestným automatem. Dokážeme opačnou implikaci, tj. že každý jazyk, který je přijímán nějakým dvoucestným automatem, je regulární. Nejdřív budeme uvažovat, že $\mathcal{A} = (Q, X, \delta, q_0, F)$ je deterministický dvousměrný automat. Zkonstruujeme pro něj nedeterministický akceptor $\mathcal{B}$ takový, že $L(\mathcal{A}) = L(\mathcal{B})$ a tím ukážeme, že $L(\mathcal{A})$ je regulární jazyk. Pak rozšíříme uvedenou konstrukci na obecné dvousměrné automaty. Tak do toho ...

Vezměme vstupní slovo $\alpha = a_1 \dots a_n \in L(\mathcal{A})$ a nechť $(r_0, i_0), \dots, (r_k, i_k)$ je přijímající výpočet $\mathcal{A}$ nad slovem α. Pozorujme, jak se chová výpočet na přechodu mezi p-tou buňkou a $(p+1)$-ní buňkou pro nějaké $p = 0, \dots, n$. Nechť $j_1, \dots, j_m$ je rostoucí posloupnost všech indexů t takových, že buď $i_t = p$ a $i_{t+1} = p+1$ nebo $i_t = p+1$ a $i_{t+1} = p$. Pak posloupnost $r_{j_1+1}, \dots, r_{j_m+1}$ nazveme přechodovou posloupností $\mathcal{A}$ nad α mezi p-tou a $(p+1)$-ní pozicí (viz obr. 8a). Všimněme si, že platí:

- m je liché
- $(\forall\, t, s)(1 \leq 2t+1, 2s+1 \leq m, t \neq s \Rightarrow (q_{2t+1} \neq q_{2s+1}) \wedge (q_{2t} \neq q_{2s}))$[46]

Tedy $m \leq 2\,|Q|$. Každou posloupnost stavů $q_1, \dots, q_m$, splňující předchozí dvě podmínky[47], budeme nazývat platnou posloupností $\mathcal{A}$. Dále definujeme proces *Škrt*(A, B, x), kde A a B jsou platné posloupnosti a $x \in X$. Tento proces simuluje činnost $\mathcal{A}$ na přechodové posloupnosti A mezi p-tou a $(p+1)$-ní pozicí a přechodovou posloupností B mezi $(p+1)$-ní a $(p+2)$-ou pozicí a pro $x = a_{p+1}$. Proces bude postupně zaškrtávat prvky v A i B. Přitom v každém kroku bude platit

- počet zaškrtnutých prvků v A bude sudý, právě když počet zaškrtnutých prvků v B bude sudý[48],
- zaškrtnuté prvky (v A i v B) tvoří počáteční interval (tj. před zaškrtnutým prvkem v A, resp. v B jsou všechny prvky zaškrtnuté).

Proces funguje takto:

1. Na začátku není zaškrtnuto nic[49].
2. Dokud lze zaškrtávat[50], provádíme

[46] pokud by se např. q_ℓ vyskytl dvakrát v přechodové posloupnosti ve stejném směru, $\mathcal{A}$ by se zacyklil. Jestliže totiž před druhým průchodem stavem q_ℓ automat nedosáhl konce vstupního slova, po druhém průchodu q_ℓ ho již nedosáhne, neboť díky tomu, že je deterministický, bude při dalším výpočtu opakovat tu posloupnost stavů a poloh ve vstupním slově, která jej převádí z prvního průchodu q_ℓ do druhého, tedy na konec vstupního slova se nikdy nedostane. Pokud před druhým průchodem q_ℓ dosáhl konce vstupního slova v nepřijímacím stavu, je situace obdobná. Jestliže dosáhl konce vstupního slova ve stavu přijímacím, není třeba pokračovat ve výpočtu — slovo je přijato (do q_ℓ se již podruhé nejde).

[47] nyní bez ohledu na vstupní slovo

[48] neboli součet všech zaškrtnutých prvků v A i B je sudý

[49] zatím je to snadné, že ... ☺

[50] pochopitelně nelze zaškrtávat (mimo jiné) v případě, že je již celé A a celé B zaškrtnuté

(a) je-li v A zaškrtnutý sudý počet členů, vezmeme první nezaškrtnutý stav q v A a zaškrtneme ho. Dokud $\delta(q,x) = (q',0)$, nahrazujeme q prvkem q'[51]. Když $\delta(q,x) = (q',1)$ a q' je první nezaškrtnutý prvek v B, pak zaškrtneme q' v B a opakujeme krok 2. Když $\delta(q,x) = (q',-1)$ a q' je první nezaškrtnutý prvek v A, pak zaškrtneme q' v A a opakujeme krok 2. V ostatních případech (tj. např. když cyklus daný podmínkou $\delta(q,x) = (q',0)$ nikdy neskončí) proces zaškrtávání skončí neúspěšně.

(b) je-li v B zaškrtnutý lichý počet členů, vezmeme první nezaškrtnutý stav q v B a zaškrtneme ho. Dokud $\delta(q,x) = (q',0)$ nahrazujeme q prvkem q'. Když $\delta(q,x) = (q',1)$ a q' je první nezaškrtnutý prvek v B, pak zaškrtneme q' v B a opakujeme krok 2. Když $\delta(q,x) = (q',-1)$ a q' je první nezaškrtnutý prvek v A, pak zaškrtneme q' v A a opakujeme krok 2. V ostatních případech proces zaškrtávání skončí neúspěšně.

3. pokud jsou obě posloupnosti A i B celé zaškrtnuté, algoritmus úspěšně skončil[52].

Není od věci se zamyslet nad tím, co vlastně tento algoritmus dělá, případně jako cvičení pro něj zkusit sestavit program. Je třeba si uvědomit, že stavy z posloupností A a B jsou právě všechny stavy, které mohou výpočet posunout z(do) buňky obsahující x. Dále si všimněme, že když A je přechodová posloupnost nad α mezi p-tou a $(p+1)$-ní pozicí, B je přechodová posloupnost nad α mezi $(p+1)$-ní a $(p+2)$-ou pozicí a když $x = a_{p+1}$, pak proces $\check{S}krt(A,B,x)$ skončí úspěšně, viz obrázek 8b. Naopak, pokud proces $\check{S}krt(A,B,x)$ skončí úspěšně, pak A a B mohou být přechodové posloupnosti nějakého výpočtu $\mathcal{A}$ nad nějakým slovem α mezi p-tou pozicí a $(p+1)$-ní pozicí a mezi $(p+1)$-ní pozicí a $(p+2)$-ou pozicí, kde x je $(p+1)$-ní písmeno α. Tato fakta zachytíme v následujícím lemmatu, které potřebujeme v důkazu věty.

Lemma 4.41 *Mějme automat $\mathcal{A}$ z počátku důkazu věty. Když $\alpha = a_1 \dots a_n$, $\alpha \in L(\mathcal{A})$ a jsou-li $A_0, \dots, A_n$ přechodové posloupnosti $\mathcal{A}$ nad α takové, že A_p je přechodová posloupnost mezi p-tou a $(p+1)$-ní pozicí, pak $A_0 = \{q_0\}$, $A_n = \{q\}$ pro nějaké $q \in F$ a proces $\check{S}krt(A_p, A_{p+1}, a_{p+1})$ skončí úspěšně pro každé $p = 0, \dots, n-1$.*

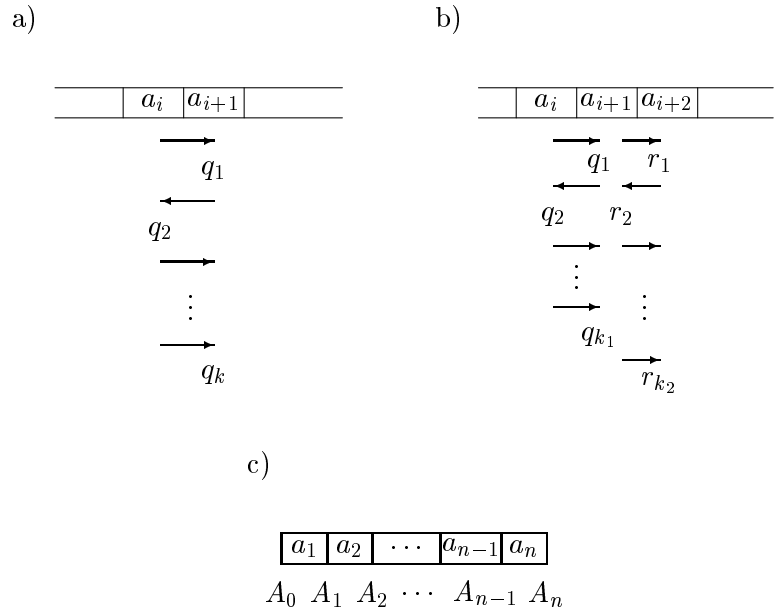
Naopak, když $\alpha = a_1 \dots a_n \in X^*$ a $A_0, \dots, A_n$ jsou platné posloupnosti nad $\mathcal{A}$ takové, že $A_0 = \{q_0\}$, $A_n = \{q\}$ pro nějaké $q \in F$ a $\check{S}krt(A_p, A_{p+1}, a_{p+1})$ skončí úspěšně pro každé $p = 0, \dots, n-1$, pak existuje přijímací výpočet $\mathcal{A}$ nad α, tj. $\alpha \in L(\mathcal{A})$.

Důkaz: Když $\alpha \in L(\mathcal{A})$, pak podle předchozích úvah plyne požadované tvrzení z definice přechodových posloupností a procesu *Škrt* poměrně triviálně.

Dokážeme opačné tvrzení. Budeme postupně vytvářet přijímací výpočet $\mathcal{A}$ nad α a zaškrtávat prvky $A_0, \dots, A_n$ takže zaškrtnuté stavy v posloupnostech

[51] to jsou přechody z buňky x zpět do x, což nás teď nezajímá

[52] Pokud skončil, aniž by byly A i B celé zaškrtnuté, jde pochopitelně o neúspěch.



Obr. 8: Přechodové posloupnosti

$A_0, \ldots, A_n$ tvoří přechodové posloupnosti doposud zkonstruovaného výpočtu. Dále bude dán aktivní prvek, bude to poslední stav ve výpočtu a zároveň poslední zaškrtnutý prvek v posloupnostech $A_0, \ldots, A_n$. Navíc bude platit:

- když A_p ani A_{p+1} neobsahuje aktivní prvek, pak zaškrtnuté prvky v posloupnostech A_p a A_{p+1} odpovídají zaškrtnutým prvkům po ukončení některého kroku 2. v procesu $\check{S}krt(A_p, A_{p+1}, a_{p+1})$.
- když aktivní prvek je v A_p a počet zaškrtnutých prvků v A_p je sudý, pak zaškrtnuté prvky v posloupnostech A_p a A_{p+1} odpovídají zaškrtnutým prvkům po ukončení některého kroku 2. v procesu $\check{S}krt(A_p, A_{p+1}, a_{p+1})$ a zároveň zaškrtnuté prvky v posloupnostech A_{p-1} a A_p po odstranění aktivního prvku odpovídají zaškrtnutým prvkům po ukončení některého kroku 2. v procesu $\check{S}krt(A_{p-1}, A_p, a_p)$
- když aktivní prvek je v A_p a počet zaškrtnutých prvků v A_p je lichý, pak zaškrtnuté prvky v posloupnostech A_{p-1} a A_p odpovídají zaškrtnutým prvkům po ukončení některého kroku 2. v procesu $\check{S}krt(A_{p-1}, A_p, a_p)$ a zároveň zaškrtnuté prvky v posloupnostech A_p a A_{p+1} po odstranění aktivního prvku odpovídají zaškrtnutým prvkům po ukončení některého kroku 2. v procesu $\check{S}krt(A_p, A_{p+1}, a_{p+1})$
- když aktivní prvek q je v A_p, pak
  - je-li počet zaškrtnutých prvků A_p sudý, poslední prvek doposud zkonstruovaného výpočtu je (q, p)
  - je-li počet zaškrtnutých prvků A_p lichý, poslední prvek doposud zkonstruovaného výpočtu je $(q, p + 1)$

Proces vytváření přijímacího výpočtu probíhá takto:

1. na počátku je zaškrtnutý pouze prvek $q_0 \in A_0$, který je také aktivní
2. nechť $q \in A_p$ je aktivní prvek
   - když počet zaškrtnutých prvků v A_p je lichý, pak do výpočtu přidáváme $(q', p+1)$ a nahrazujeme q prvkem q', dokud $\delta(q, a_{p+1}) = (q', 0)$. Když před přidáním jsme měli výpočet $\mathcal{A}$ nad α, pak po přidání opět dostaneme výpočet nad α, protože původní výpočet končil dvojicí $(q, p+1)$. Protože proces $\check{S}krt(A_p, A_{p+1}, a_{p+1})$ úspěšně skončí, existuje okamžik, kdy $\delta(q, a_{p+1}) = (q', \ell)$ pro $\ell \neq 0$. Pak buď $\ell = -1$ a q' je první nezaškrtnutý prvek v A_p — zaškrtneme ho a do výpočtu přidáme (q', p), nebo $\ell = 1$ a q' je první nezaškrtnutý prvek v A_{p+1} — zaškrtneme ho a do výpočtu přidáme $(q', p+2)$. V obou případech obdržíme výpočet nad α, protože před přidáním jsme měli výpočet nad α s posledním prvkem $(q, p+1)$, a q' bude nový aktivní prvek.
   - když počet zaškrtnutých prvků v A_p je sudý, pak přidáváme (q', p) do výpočtu a nahrazujeme q prvkem q', dokud $\delta(q, a_p) = (q', 0)$. Když před přidáním jsme měli výpočet $\mathcal{A}$ nad α, pak po přidání opět dostaneme výpočet nad α, protože původní výpočet končil dvojicí (q, p). Protože proces $\check{S}krt(A_{p-1}, A_p, a_p)$ úspěšně skončí, existuje okamžik, kdy $\delta(q, a_p) = (q', \ell)$ pro $\ell \neq 0$. Pak buď $\ell = -1$ a q' je první nezaškrtnutý prvek v A_{p-1} — zaškrtneme ho a do výpočtu přidáme $(q', p-1)$, nebo $\ell = 1$ a q' je první nezaškrtnutý prvek v A_p — zaškrtneme ho a do výpočtu přidáme $(q', p+1)$. V obou případech obdržíme výpočet nad α, protože před přidáním jsme měli výpočet s posledním prvkem (q, p), a q' bude nový aktivní prvek.
3. protože proces $\check{S}krt(A_p, A_{p+1}, a_{p+1})$ úspěšně skončí pro všechna p z intervalu $\langle 0, n-1 \rangle$, existuje okamžik, kdy aktivním prvkem je $q \in A_n$, pak výpočet skončil prvkem $(q, n+1)$. Protože $q \in F$, obdrželi jsme přijímající výpočet nad α a tedy $\alpha \in L(\mathcal{A})$.

Tím je důkaz lemmatu kompletní. CBD

Toto lemma nám radí, jak konstruovat N-akceptor $\mathcal{B} = (Q', X, \delta', I', F')$, kde

- Q' je množina všech platných posloupností
- $\delta'(A, x) = \{B \in Q'; \check{S}krt(A, B, x)$ úspěšně skončí$\}$ pro každou platnou posloupnost A a $x \in X$
- I' obsahuje jedinou platnou posloupnost $\{q_0\}$
- F' obsahuje všechny platné posloupnosti $\{q\}$ pro $q \in F$

Zřejmě z pomocného lemmatu 4.41 platí $L(\mathcal{A}) = L(\mathcal{B})$, tedy $L(\mathcal{A})$ je podle tvrzení 4.28 regulární jazyk.

Uvažujme nyní, že $\mathcal{A} = (Q, X, \delta, I, F)$ je obecný nedeterministický dvoucestný automat. Nemůžeme sice použít ideu z tvrzení 4.28 na odstranění nedeterminismu, ale detailním prohlédnutím předchozí konstrukce zjistíme, že determinismus se používá ve dvou místech — aby přechodové posloupnosti měly omezenou délku a aby proces $\check{S}krt(A, B, x)$ byl deterministický. Oba problémy lze však obejít a tuto konstrukci použít i pro obecný nedeterministický dvoucestný automat.

Nejprve si všimněme, když $(r_0, i_0), \ldots, (r_k, i_k)$ je přijímající výpočet $\mathcal{A}$ nad slovem α a $(r_t, i_t) = (r_s, i_s)$ pro $t < s$, pak

$$(r_0, i_0), \ldots, (r_t, i_t), (r_{s+1}, i_{s+1}), \ldots, (r_k, i_k)$$

je opět přijímající výpočet $\mathcal{A}$. Proto pro každé slovo $\alpha \in L(\mathcal{A})$ existuje prostý přijímající výpočet[53]. Definujeme-li přechodové posloupnosti stejně jako pro deterministický dvoucestný automat, pak pro prostý přijímající výpočet jsou přechodové posloupnosti platné. To vede k odstranění prvního problému.

Za druhé v definici procesu $\check{S}krt(A, B, x)$ nahradíme příkaz $\delta(q, x) = (q', \ell)$ příkazem „zvolme $(q', \ell) \in \delta(q, x)$“. Řekneme, že proces $\check{S}krt(A, B, x)$ úspěšně skončí, když existují volby $(q', \ell) \in \delta(q, x)$ takové, že se zaškrtávání povede úspěšně zakončit.

Nyní definujme N-akceptor $\mathcal{B} = (Q', X, \delta', I', F')$ stejně jako pro deterministický dvoucestný automat s jedinou výjimkou: I' je množina všech platných posloupností $\{q\}$ pro $q \in I$. Pak stejné argumenty jako v deterministickém případě nám dají $L(\mathcal{A}) = L(\mathcal{B})$ a tedy $L(\mathcal{A})$ je regulární jazyk, protože $\mathcal{B}$ je N-akceptor. CBD

Příklad 4.42 *(Dvoucestný automat) Sestrojte dvoucestný automat, přijímající jazyk*

$$L'_n = \{\#v\#; v \in \{0,1\}^*, n\text{-té písmeno slova } v \text{ od konce je } 1\}$$

Jazyk L'_n je velmi podobný jazyku L_n na straně 40, kde jsme ukazovali, že nejlepší deterministický akceptor, který L_n přijímá, má 2^n stavů. Každý jistě dokáže nahlédnout, že totéž (až na konstantu) platí i pro L'_n, a sestrojit si nedeterministický akceptor přijímající L'_n. My teď máme jiný úkol: sestrojit pro L'_n deterministický dvoucestný automat[54]. Je to snadné[55]...

Náš automat $\mathcal{A}$ bude pracovat tak, že přečte celé vstupní slovo, vrátí se o n písmen zpět, ověří, zda n-té písmeno od konce je 1, v případě úspěchu dočte vstupní slovo do konce a přijme, jinak výpočet selže. Automat $\mathcal{A}$ s $(n+3)$-mi stavy ukazuje obrázek 9. $q_0, \ldots, q_{n+2}$ jsou stavy $\mathcal{A}$, ohodnocení hran kóduje přechodovou funkci a posun čtecí hlavy na vstupní pásce[56]. Není příliš obtížné přesvědčit se, že $\mathcal{A}$ přijímá právě L'_n. FINE

Poznámka: O dvoucestných automatech, které ke své práci používají „zarážky“ (v $\mathcal{A}$ to byl znak #), se ještě později[57] zmíníme a ukážeme, že přijímají také regulární jazyky.

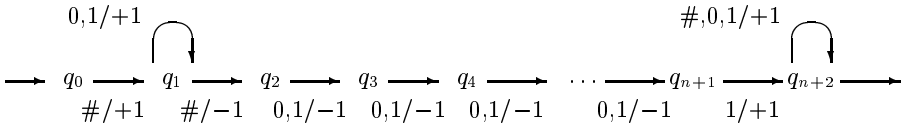
[53] prostý přijímací výpočet je takový, ve kterém se žádná konfigurace neopakuje

[54] pochopitelně s co nejméně stavy, jinak to asi nemá valný význam

[55] jako vždycky

[56] např. ohodnocení $0, 1/+1$ značí přechod přes 0 nebo 1 a posun hlavy o $+1$, tedy vpravo

[57] v kapitole 4.6 o kvocientech jazyků



Obr. 9: Dvoucestný automat $\mathcal{A}$ přijímající L'_n

Poznámka: Zdůrazňuji, že dvoucestný automat sestrojený v předchozím příkladu má $n+3$ stavů a je deterministický. Nejlepší jednocestný automat přijímající L'_n má také $n+3$ stavů, ale je nedeterministický!!! Lze tedy vyslovit hypotézu „dvoucestné automaty sice nerozšiřují možnosti jednocestných[58], ale jsou efektivnější[59]“. Uvedená hypotéza je samozřejmě pravdivá, nicméně její ověření není jednoduché a přesahuje rámec těchto skript. Je možné ukázat (viz [5],[2]), že existuje jazyk

$$L_n^1 = \{\#a^{i_1}ba^{i_2}b\ldots ba^{i_n}c^k d^{i_k}\#; 1 \leq k \leq n \wedge (\forall\, j \leq n)(1 \leq i_j \leq n)\},$$

pro který nejmenší dvoucestný deterministický automat, který ho přijímá, má $5n+6$ stavů a každý jednocestný D-akceptor, který ho přijímá, má alespoň n^n stavů. V [5] je navíc uveden důkaz, že převod dvoucestného D-automatu s n stavy na jednocestný N-akceptor vyžaduje v obecném případě $\sum_{k=1}^{n} \binom{n}{k}\binom{n}{k-1}$ stavů.

Konkrétní jazyky a automaty, na kterých je efektivnost dvoucestných automatů proti jednocestným prokázana, nejsou podstatné. Důležité je, že hypotéza z předchozího odstavce platí včetně slůvka *řádově* ve vysvětlující poznámce pod čarou.

Poznámka: Protože jsme se již seznámili s několika druhy automatů, bylo by asi vhodné uvést pro jednotlivé druhy nějaké ustálené zkratky. Z možností, se kterými jsem se v literatuře doposud setkal, mi připadaly nejlogičtější zkratky typu 1DFA, 1NFA, 2DFA, 2NFA s významem „jednocestný deterministický konečný automat“, „jednocestný nedeterministický konečný automat“, podobně pro dvoucestné, nekonečné a další a další.[60]

4.5 Substituce

Definice 4.43 *Substitucí z abecedy A do abecedy B nazveme každé zobrazení $\sigma : A \to \mathcal{P}(B^*)$.*

Je-li L jazyk nad A a M jazyk nad B, pak definujeme

$$\begin{aligned}\sigma(L) \;=\; &\{\alpha_1\ldots\alpha_n;\\ &(\exists\, a_1\ldots a_n \in L)(\forall\, i = 1,2,\ldots,n)(\alpha_i \in \sigma(a_i))\} \subseteq B^*\end{aligned}$$

[58] v tom smyslu, že přijímají stejnou třídu jazyků

[59] tedy pro daný jazyk mají buď řádově méně stavů, nebo alespoň odstaňují nedeterminismus

[60] Tato poznámka byla připojena pouze proto, aby se čtenář, který by se náhodou dostal k nějaké literatuře o automatech, nezalekl neznámých shluků písmen.

$$\begin{aligned}\sigma^{-1}(M) &= \{a_1 \dots a_n; \\ &\quad (\forall\, i = 1, 2, \dots, n)(\exists\, \alpha_i \in \sigma(a_i))(\alpha_1 \dots \alpha_n \in M)\} \subseteq A^*\end{aligned}$$

Zobrazení σ^{-1} nazýváme inverzní substitucí.

Substituce σ je regulární, platí-li, že $\sigma(a)$ je regulární jazyk pro všechna $a \in A$.

Abychom zápis v definici trochu polidštili. Substituce σ z A do B přiřazuje každému písmenu a z A množinu slov (jazyk) z B, kterými lze písmeno a nahradit. Aplikujeme-li všechny možné substituce na všechna slova z jazyka L nad A, dostaneme $\sigma(L)$. Naopak pro daný jazyk M nad B každé slovo z A^*, které lze alespoň jedním způsobem přepsat substitucí σ na slovo z M, padne do $\sigma^{-1}(M)$.

Příklad 4.44 *(Substituce) Jsou dány abecedy $A = \{a, b, c\}$, $B = \{0, 1\}$ a substituční zobrazení $\sigma : A \to \mathcal{P}(B^*)$ o hodnotách*

$$\begin{aligned}\sigma(a) &= \{010\} \\ \sigma(b) &= \{0110\} \\ \sigma(c) &= \{1\}\end{aligned}$$

Jazyk $L \subseteq A^*$ nechť je zadán regulárním výrazem b^*ca^*. Určete jazyky $\sigma(L)$ a $\sigma^{-1}(\sigma(L))$.

Protože každá z množin $\sigma(x)$[61] má jen jeden prvek, stačí v regulárním výrazu pro L zaměnit jednotlivá písmena jejich obrazy a máme regulární výraz pro $\sigma(L)$, tzn.

$$\sigma(L) = (0110)^*1(010)^* = N$$

za předpokladu, že regulární výraz ztotožníme s množinou slov, které popisuje[62].

Najít $\sigma^{-1}(N)$ není tak docela triviální. Tvrdím

$$L = \sigma^{-1}(N).$$

Každý jistě snadno nahlédne, že inverzní substitucí z jazyka N dostaneme všechna slova z L (neboli $L \subseteq \sigma^{-1}(N)$). Že nedostaneme žádné slovo mimo L, je třeba při formálním důkazu ověřit. Lze postupovat např. tak, že dokážeme následující invarianty, jejichž platnost již zaručuje inkluzi $\sigma^{-1}(N) \subseteq L$, o kterou nám jde:

- v každém slově z $\sigma^{-1}(N)$ se vyskytuje právě jedno c.
- v žádném slově z $\sigma^{-1}(N)$ po c již nenásleduje b.
- v žádném slově z $\sigma^{-1}(N)$ se před c nevyskytuje a.

[61] $x \in \{a, b, c\}$

[62] tento předpoklad je velmi přirozený a budeme jej používat i nadále

Tyto invarianty nechávám již čtenáři na rozvážení. Je možné samozřejmě použít jiné invarianty nebo celý důkaz konstruovat jinak. Po prokázání platnosti invariantů spolu s triviální první inkluzí dostáváme $L = \sigma^{-1}(N)$, což jsme chtěli ukázat.

Tím je příklad dokončen, ...

... ale není každá substituce tak průhledná. Jednoduchou změnou množin $\sigma(a), \sigma(b), \sigma(c)$ už můžeme získat značně složitější jazyky, nemluvě pak o jazycích $\sigma^{-1}(\sigma(L))$. Takovým zdánlivě jen o málo složitějším příkladem změny σ je substituční zobrazení σ'

$$\begin{aligned} \sigma'(a) &= \{010, 0\} \\ \sigma'(b) &= \{0110, 1001\} \\ \sigma'(c) &= \{10, 01\} \end{aligned}$$

Zřejmě platí

$$\sigma'(L) = (0110 + 1001)^*(01 + 10)(010 + 0)^* = N'$$

Poněkud náročnější je odvodit $(\sigma')^{-1}(N')$. Označíme-li

$$\begin{aligned} O &= (0110 + 1001)^* \\ P &= 01 + 10 \\ R &= (010 + 0)^*, \end{aligned}$$

můžeme se pokusit úlohy poněkud zjednodušit. Podívejme se, jak vypadají množiny $\sigma^{-1}(O)$, $\sigma^{-1}(P)$, $\sigma^{-1}(R)$. To není problém[63]:

$$\begin{aligned} \sigma^{-1}(O) &= (b + cc)^* \\ \sigma^{-1}(P) &= c \\ \sigma^{-1}(R) &= (a + aba + ac + ca)^* \end{aligned}$$

Někomu by se možná zdálo, že

$$\sigma^{-1}(N') = \sigma^{-1}(O)\sigma^{-1}(P)\sigma^{-1}(R) \tag{35}$$

(neboli $\sigma^{-1}(N')$ je konkatenací obrazů množin O, P, R v σ^{-1}) a že jsme tedy hotovi, ale *není* tomu tak! Jako protipříklad uveďme slovo $cb \in \sigma^{-1}(011001)$, které patří do $\sigma^{-1}(N')$, neboť $011001 \in N'$, ale v množině $\sigma^{-1}(O)\sigma^{-1}(P)\sigma^{-1}(R)$ zahrnuto není, jak si jistě každý snadno ověří. Chyba úvahy (35) je v tom, že nebereme v potaz slova z N', která začínají v jedné z množin O, P, R a končí v jiné z nich (např. 011001 začíná prefixem $0110 \in O$ a končí suffixem $01 \in P$). Podívejme se tedy, jak vypadají tyto případy[64]:

$$\begin{aligned} \sigma^{-1}(OP) &= \sigma^{-1}(O)(\sigma^{-1}(P) + bc + cb) \\ \sigma^{-1}(PR) &= (\sigma^{-1}(P) + acc + ac + ba + a + aba)\sigma^{-1}(R), \end{aligned}$$

[63] opět ztotožňujeme množinu a reg. výraz, který ji popisuje

[64] případ $\sigma^{-1}(OR)$ není třeba rozebírat, neboť nějaký prvek množiny P musí být ve slovech z N' vždy přítomen

což po rozepsání σ^{-1} a vynechání zbytečných členů dává

$$\begin{aligned}\sigma^{-1}(OP) &= (b+cc)^*(c+cb)\\ \sigma^{-1}(PR) &= (c+acc+ba)(a+aba+ac+ca)^*.\end{aligned}$$

Nyní jsme již skoro u konce. Zřejmě platí

$$\sigma^{-1}(OPR) = \overbrace{\sigma^{-1}(OP)\sigma^{-1}(R)+\sigma^{-1}(O)\sigma^{-1}(PR)}^{M}+\sigma^{-1}(O)Q\sigma^{-1}(R),$$

kde Q je množina všech slov λ z $\{a,b,c\}^*$, pro která existuje nějaké slovo $\alpha \in OPR$ s rozkladem $\alpha = \alpha_1 \ldots \alpha_n$ (kde $\alpha_i \in \sigma(a) \cup \sigma(b) \cup \sigma(c)$) takovým, že pro nějaké i_0 $(1 \leq i_0 \leq n)$

$$(\alpha_1 \ldots \alpha_{i_0} \notin OP) \vee (\alpha_{i_0+1} \ldots \alpha_n \notin PR)$$

a zároveň

$$(\alpha_1 \ldots \alpha_{i_0} \notin O) \vee (\alpha_{i_0+1} \ldots \alpha_n \notin R)$$

Po této ukázce síly matematického aparátu připojuji vysvětlení. Problém je stejný jako při konstrukci $\sigma^{-1}(OP)$ a $\sigma^{-1}(PR)$. Mohou se vyskytnout slova, která patří do OPR, ale lze je rozložit ještě jiným způsobem, než jaký naznačuje množina M[65]. Takové slovo je např. $\alpha = 1001010010 \in OPR$ s rozkladem např.

$$\begin{aligned}\alpha_1 &= 1001\\ \alpha_2 &= 0\\ \alpha_3 &= 1001\\ \alpha_4 &= 0\end{aligned}$$

a $i_0 = 2$. Rozborem případů lze zjistit, že slovo α je jediné slovo, které přispívá do množiny Q. Vynecháme-li z Q slova zahrnutá již jednou v M, máme $Q = baba + bacac + cacba + caccca + ccaba + ccaca$. Po zjednodušení

$$\begin{aligned}\sigma^{-1}(N') &= (b+cc)^* \cdot (c+cb+acc+(ba+cac)(ba+cac)+\\ &\quad cca(ba+ca)+caccca)\cdot(a+aba+ac+ca)^*\end{aligned}$$

V uvedeném příkladu je možné si povšimnout, že σ i σ' jsou regulární substituce. Uznávám, že popsaný postup je velmi málo podložen fakty, na čemž budu dále pracovat. Snad alespoň nastiňuje problémy, které mohou řešení podobných příkladů komplikovat, a možnosti jejich překonání. FINE

Věta 4.45 *Mějme dvě abecedy A, B. Pro každou regulární substituci σ a libovolné regulární jazyky $L \subseteq A^*, M \subseteq B^*$ jsou $\sigma(L), \sigma^{-1}(M)$ také regulární jazyky.*

Důkaz:

[65] neboli zvolený rozklad těchto slov je možné zobrazit inverzní substitucí, ale není možné jej rozdělit na část v OP a část v R (resp. O a PR)

1. Je-li σ regulární substituce, pak $\forall a \in A$ je $\sigma(a)$ regulární jazyk. Podle Kleeneho věty (4.35) existuje regulární výraz β_a popisující jazyk $\sigma(a)$. Podobně existuje regulární výraz β popisující L. Potom ale můžeme každou proměnnou $a \in \beta$ nahradit výrazem (β_a) a dostaneme regulární výraz γ (regularita γ plyne z definice). Jazyk $\sigma(L)$ je zřejmě popisován regulárním výrazem γ, neboli je regulární.

2. Je-li σ regulární substituce, pak pro každé $a \in A$ existuje D-akceptor

   $$\mathcal{A}_a = (Q_a, B, \delta_a, q_{a0}, F_a)$$

   přijímající jazyk $\sigma(a)$. Od akceptorů $\mathcal{A}_a$ můžeme búno[66] požadovat

   $$(\forall a_1, a_2 \in A)(a_1 \neq a_2 \Rightarrow Q_{a_1} \cap Q_{a_2} = \emptyset).$$

   Nechť jazyk M je přijímán D-akceptorem[67] $\mathcal{A} = (Q, B, \delta, q_0, F)$. Definujme dvoucestný nedeterministický automat $\mathcal{A}' = (Q', A, \delta', I', F')$, kde

   $$\begin{aligned}
   Q' &= (Q \times \bigcup_{a \in A} Q_a) \cup \{q_f\} \qquad \text{pro } q_f \notin Q \times \bigcup_{a \in A} Q_a \\
   I' &= \{(q_0, q_{a0}); a \in A\} \\
   F' &= \{q_f\} \\
   \delta'((q,r),a) &\ni \begin{cases} ((\delta(q,b), \delta_a(r,b)), 0) & \forall b \in B, \text{ je-li } r \in Q_a \\ ((q, q_{b0}), 1) & \forall b \in A, \text{ pokud } r \in F_a \\ (q_f, 1) & \text{pokud } r \in F_a, q \in F \end{cases} \\
   \delta'(q_f, a) &= \emptyset
   \end{aligned}$$

   Pozor!!! Čtecí hlava automatu $\mathcal{A}'$ se může (ale nemusí) pohnout pouze tehdy, je-li $r \in F_a$. $\mathcal{A}'$ je nedeterministický, tzn. přijme vstupní slovo, pokud *existuje* přijímací posloupnost. Funguje vlastně velmi jednoduše. Pro vstupní slovo $a_1 \ldots a_n$ se pokouší najít ke každému a_i odpovídající slovo $\alpha_i \in \sigma(a_i)$ takové, aby $\alpha_1 \ldots \alpha_n \in M$. Z definice $\mathcal{A}'$ lze „vykoukat“, že paralelně probíhají výpočty v akceptorech $\mathcal{A}_{a_i}$ a $\mathcal{A}$. Nakonec $\mathcal{A}'$ akceptuje vstupní slovo $a_1 \ldots a_n$, právě když $\mathcal{A}$ akceptuje slovo $\alpha_1 \ldots \alpha_n$.

   Podrobněji: $\mathcal{A}'$ začne výpočet a přečte a_1. Od té chvíle se poloha čtecí hlavy nemění a výpočet se pohybuje paralelně v akceptorech $\mathcal{A}$ (první složka) a $\mathcal{A}_{a_1}$ (druhá složka stavů $\mathcal{A}'$) podle prvního řádku definice δ' tak dlouho, dokud akceptor $\mathcal{A}'$ nenajde vhodné slovo $\alpha_1 \in \sigma(a_1)$. Pak se může čtecí hlava posunout o pozici vpravo (druhý řádek definice δ') a výpočet nyní probíhá v $\mathcal{A}$ a $\mathcal{A}_{a_2}$, přičemž v $\mathcal{A}$ se nevrací do počátečního stavu, ale pokračuje tam, kde přestal po přečtení α_1. Tak postupujeme až k a_n, kde $\mathcal{A}_{a_n}$ „navrhne“ slovo α_n a skončí v některém ze svých přijímacích stavů[68]. Pokud zároveň i momentální stav v $\mathcal{A}$ je přijímací, $\mathcal{A}'$ se podle třetího

[66] bez újmy na obecnosti

[67] determinismus akceptorů $\mathcal{A}$ a $\mathcal{A}_a$ není až tak důležitý, jen se snažím búno zjednodušit definici $\mathcal{A}'$

[68] když ne, vezmeme si jiný výpočet — od toho je nedeterministický

řádku definice δ' může přesunout za vstupní slovo a přejít do přijímacího stavu q_f, takže $a_1 \dots a_n \in L(\mathcal{A}')$ kdykoliv $\alpha_1 \dots \alpha_n \in M$. Z definice δ' dále snadno plyne, že $\mathcal{A}'$ může přijmout jen slova, která jsou v $\sigma^{-1}(M)$, což ponechávám čtenáři na rozmyšlení[69].

Ukázali jsme, že $\mathcal{A}'$ nejen přijímá jazyk $\sigma^{-1}(M)$, ale dokonce hledá odpovídající substituovaná slova z M. $\mathcal{A}'$ je vzhledem ke své definici jistě konečný, tedy $\sigma^{-1}(M)$ je regulární. CBD

4.6 Kvocienty jazyků

Definice 4.46 *Jsou-li L_1, L_2 jazyky nad abecedou X, pak*

- *levým kvocientem L_2 podle L_1 nazveme jazyk*

$$L_1 \backslash L_2 = \{u \in X^*; (\exists\, v \in L_1)(vu \in L_2)\}$$

- *pravým kvocientem L_2 podle L_1 nazveme jazyk*

$$L_2 / L_1 = \{u \in X^*; (\exists\, v \in L_1)(uv \in L_2)\}$$

Příklad 4.47 *(Kvocienty) K jazykům $L_1 = 0^*10^*, L_2 = 10^*1$ sestrojte pravé a levé kvocienty L_1 podle L_2 i naopak.*

Řešení:

$$\begin{aligned} L_1/L_2 &= \emptyset \\ L_2\backslash L_1 &= \emptyset \\ L_2/L_1 &= 10^* \\ L_1\backslash L_2 &= 0^*1 \end{aligned}$$

Tvrzení 4.48 *Je-li L regulární jazyk nad X, pak pro každý (i neregulární) jazyk L' nad X jsou L/L' a $L'\backslash L$ regulární jazyky.*

Důkaz: L je regulární, tedy existuje dosahující akceptor $\mathcal{A} = (Q, X, \delta, q_0, F)$ takový, že $L(\mathcal{A}) = L$. Definujeme akceptory

$$\mathcal{A}_1 = (Q, X, \delta, I, F), \quad \mathcal{A}_2 = (Q, X, \delta, q_0, F'),$$

kde

$$\begin{aligned} I &= \{q \in Q; (\exists\, v \in L')(\delta^*(q_0, v) = q)\} \\ F' &= \{q \in Q; (\exists\, v \in L')(\delta^*(q, v) \in F)\} \end{aligned}$$

[69] Prakticky to znamená ověřit, že se do q_f můžeme dostat jen způsobem popsaným v tomto odstavci.

Idea této definice je zhruba následující:

Vezmeme všechna $v \in L'$ a stavy, do kterých se při výpočtu $\mathcal{A}$ nad slovy v dostaneme, označíme za iniciální v $\mathcal{A}_1$. Je-li nějaké slovo u přijímáno v $\mathcal{A}_1$ s výpočtem začínajícím v $q \in I$, muselo nutně existovat slovo $v \in L'$ takové, že $\delta(q_0, v) = q$, tzn. $vu \in L$ a $u \in L' \backslash L$. Naopak je-li $vu \in L$ pro nějaké $v \in L'$ (tzn. $u \in L' \backslash L$), pak výpočet nad u v $\mathcal{A}$ musel začít ve stavu, do kterého jsme se dostali po výpočtu nad v. Tyto stavy jsme však označili za iniciální v $\mathcal{A}_1$, neboli $u \in L(\mathcal{A}_1)$ a z celého odstavce $L(\mathcal{A}_1) = L' \backslash L$.

Pro $\mathcal{A}_2$ vezmeme všechna slova $v \in L'$ a najdeme k nim stavy q takové, že $\delta^*(q, v) \in F$[70]. Ty prohlásíme za přijímací v $\mathcal{A}_2$. Je-li nějaké slovo u přijímáno v $\mathcal{A}_2$, pak výpočet nad u končí v některém přijímacím stavu q v $\mathcal{A}_2$ a lze tedy najít $v \in L'$, které prodlužuje výpočet z q do přijímacího stavu v $\mathcal{A}$, neboli $uv \in L$ a $u \in L/L'$. Je-li naopak $uv \in L$ pro nějaké $v \in L'$ (tzn. $u \in L/L'$), pak jistě existuje stav q takový, že v něm končí výpočet nad u v $\mathcal{A}$. Potom ale $\delta^*(q, v) \in F$, neboli q byl přijímací stav v $\mathcal{A}_2$, což dává $u \in L(\mathcal{A}_2)$ a z celého odstavce $L(\mathcal{A}_2) = L/L'$.

Akceptory $\mathcal{A}_1, \mathcal{A}_2$ jsou stejně jako $\mathcal{A}$ konečné, tedy oba kvocienty jsou regulární. CBD

V kapitole o dvoucestných automatech jsem slíbil, že zde ukážeme, že dvoucestné automaty se „zarážkami“ přijímají právě regulární jazyky. Tak do toho.

Definice 4.49 *Dvoucestný automat $\mathcal{A} = (Q, X \cup \{\#\}, \delta, I, F)$ $(\# \notin X)$ přijímá jazyk $T(\mathcal{A})$ s koncovými znaky (se zarážkami), jestliže*

$$T(\mathcal{A}) = \{\alpha \in X^*; \#\alpha\# \in L(\mathcal{A})\},$$

kde $L(\mathcal{A})$ je jazyk přijímaný automatem $\mathcal{A}$[71].

Tvrzení 4.50 *Dvoucestné automaty přijímají s koncovými znaky právě regulární jazyky.*

Důkaz: Že každý regulární jazyk je přijímaný nějakým dvoucestným automatem s koncovými znaky, je myslím triviální[72]. Na druhou stranu je-li $T(\mathcal{A})$ jazyk přijímaný dvoucestným automatem $\mathcal{A}$ s koncovými znaky, pak zřejmě

$$T(\mathcal{A}) = \{\#\} \backslash [(L(\mathcal{A}) \cap L^{\#}_{X^*})/\{\#\}],$$

kde

$$L^{\#}_{X^*} = \{\#\alpha\#; \alpha \in X^*\}.$$

Jazyk $L^{\#}_{X^*}$ je regulární (trivialita — 1DFA s třemi stavy). $L(\mathcal{A})$ je regulární podle 4.40, průnik podle 4.23 a kvocienty podle 4.48, tzn. $T(\mathcal{A})$ je regulární. CBD

Kapitolou o kvocientech jazyků jsme ukončili velkou pasáž, týkající se akceptorů neboli zařízení na rozpoznávání regulárních jazyků. Vrhněme se nyní na mechanismy generující jazyky...

[70] nemusí to jít pro všechna v

[71] viz definice 4.39

[72] ale je vhodné jako cvičení si to rozmyslet

5 Gramatiky

Definice 5.1 *Dvojici (X, P) nazveme přepisovací systém, pokud X je abeceda a $P \subseteq \{u \to v; u, v \in X^*\}$ konečná. Dvojici $(u \to v)$ nazýváme pravidlo. Kvůli zkrácení používáme často pro pravidla $(u \to v_1), \ldots, (u \to v_n)$ se stejnou levou stranou zápisu $u \to v_1 \mid v_2 \mid \ldots \mid v_n$.*

Řekneme, že přepisovací systém (X, P) přímo odvozuje ze slova α slovo β, když $\alpha = \alpha_1 u \alpha_2$, $\beta = \alpha_1 v \alpha_2$ a $(u \to v) \in P$. Píšeme $\alpha \overset{(X,P)}{\Longrightarrow} \beta$.

Posloupnost přímých odvození $\alpha_1 \overset{(X,P)}{\Longrightarrow} \alpha_2 \overset{(X,P)}{\Longrightarrow} \alpha_3 \overset{(X,P)}{\Longrightarrow} \ldots \overset{(X,P)}{\Longrightarrow} \alpha_n$ nazveme derivací α_n z α_1 v přepisovacím systému (X, P). Derivace je minimální, pokud má ze všech derivací pro daná α_1, α_n minimální počet přímých odvození.

Přepisovací systém (X, P) odvozuje ze slova α slovo β, pokud existuje derivace β z α v (X, P). Píšeme $\alpha \overset{(X,P)}{\underset{*}{\Longrightarrow}} \beta$.

Definice 5.2 *Gramatikou nazveme každou čtveřici (V_N, V_T, S, P), kde*

- *V_N je neterminální abeceda*
- *V_T je terminální abeceda*
- *$S \in V_N$ je iniciální symbol*
- *$(V_N \cup V_T, P)$ je přepisovací systém, pro který*

$$(u \to v) \in P \Rightarrow (\exists\, A \in V_N)(A \in u).$$

Prvky v neterminální abecedě jsou zvány neterminály, prvky terminální abecedy překvapivě terminály.

Gramatika $G = (V_N, V_T, S, P)$ generuje jazyk

$$L(G) = \{\alpha \in V_T^*; S \overset{(V_N \cup V_T, P)}{\underset{*}{\Longrightarrow}} \alpha\}$$

Máme-li gramatiku $G = (V_N, V_T, S, P)$, pak kvůli zkrácení a zpřehlednění píšeme $\alpha \overset{G}{\underset{*}{\Longrightarrow}} \beta$ místo $\alpha \overset{(V_N \cup V_T, P)}{\underset{*}{\Longrightarrow}} \beta$. Oba zápisy považujeme za ekvivalentní a znamenají, že slovo β je gramatikou G odvoditelné ze slova α.

Z důvodu úspory prostoru a zvýšení přehlednosti v následujícím textu zavedeme zcela přirozeně ekvivalenci gramatik.

Definice 5.3 *O dvou gramatikách G, G' řekneme, že jsou ekvivalentní, kdykoli generují stejný jazyk $(L(G) = L(G'))$.*

Jazykozpytec Chomski [čomsky] rozdělil gramatiky do čtyř tříd. Na jeho památku je následující rozdělení zváno *Chomského hierarchie*.

5.1 Chomského hierarchie

Definice 5.4 *1. O libovolné (obecné) gramatice řekneme, že je typu 0. Třída jazyků generovaných libovolnou gramatikou se označuje $\mathcal{L}_0$ (mluvíme pak o jazycích typu 0).*

2. Kontextové gramatiky (neboli gramatiky typu 1) jsou gramatiky splňující

$$\begin{aligned}&(\forall\,(u \to v) \in P)(\exists\, X \in V_N)\\ &(\exists\, y \in (V_N \cup V_T)^+)(\exists\, \alpha_1, \alpha_2 \in (V_N \cup V_T)^*)\\ &(u = \alpha_1 X \alpha_2 \wedge v = \alpha_1 y \alpha_2)\end{aligned}$$

V případě, že S není na pravé straně žádného pravidla, může navíc kontextová gramatika obsahovat pravidlo $S \to \Lambda$. Generují třídu $\mathcal{L}_1$ kontextových jazyků.

3. Bezkontextové gramatiky (gramatiky typu 2) splňují

$$(\forall\,(u \to v) \in P)(u \in V_N, v \in (V_N \cup V_T)^*)$$

Generují třídu $\mathcal{L}_2$ bezkontextových jazyků.

4. Regulární gramatiky (gramatiky typu 3) splňují

$$(\forall\,(u \to v) \in P)\,[(u \in V_N) \wedge (v = \alpha X, \alpha \in V_T^*, X \in V_N \cup \{\Lambda\})]$$

Generují třídu $\mathcal{L}_3$ regulárních jazyků.

Postupně ukážeme, že třídy jazyků z Chomského hierarchie jsou uspořádány inkluzí

(36) $\mathcal{L}_3 \subset \mathcal{L}_2 \subset \mathcal{L}_1 \subset \mathcal{L}_0$

a že $\mathcal{L}_3$ jsou vskutku ty regulární jazyky, jak je známe z kapitoly o akceptorech.

Chomského hierarchie pochopitelně nepopisuje všechny typy gramatik, to by asi ani nešlo. Jako příklad uvádím alespoň tři další typy gramatik:

- Levé regulární gramatiky obsahují pouze pravidla typu $(X \to Y\alpha)$, kde $X \in V_N$ a $\alpha \in V_T^*$, $Y \in V_N \cup \{\Lambda\}$. Nikoho asi nepřekvapí, že generují právě regulární jazyky.[1]

- Lineární gramatiky jsou obecnější. Obsahují pravidla $(X \to \alpha Y \beta)$, případně $(X \to \alpha)$, kde $\alpha, \beta \in V_T^*$, $X, Y \in V_N$. Generují regulární jazyky a některé další (např. $\{a^n b^n; n = 0, 1, \ldots\}$ — tento jazyk není regulární, viz příklad 4.25) a jsou podmnožinou bezkontextových gramatik.

- O gramatice $G = (V_N, V_T, S, P)$ řekneme, že je nezkracující, pokud

[1] Tento fakt lze nahlédnout například z důkazu tvrzení 5.8 (viz níže), kde místo regulární gramatiky G můžeme pracovat bez podstatnějších změn s levou regulární gramatikou, ale jistě by šel najít i jednodušší důkaz.

1. S není na pravé straně žádného pravidla
2. $(\alpha \to \beta) \in P \Rightarrow ((\alpha = S \wedge \beta = \Lambda) \vee (|\alpha| \leq |\beta|))$

Příklad 5.5 *(Gramatika) Sestrojte nezkracující gramatiku G, generující jazyk*

$$L = \{a^i b^i c^i; i \in N_0\}.$$

Zapište odvození slova aabbcc v gramatice G.

Jak asi každého po chvilce zkoušení možných gramatik napadne, hledaná gramatika by měla pracovat tak, že v první fázi vygeneruje slovo $a^i c^i$, proložené nějakými neterminály, kterých by mělo být také i. Tyto neterminály se v druhé fázi budou přepisovat na b. Vzhledem k tomu, že gramatika má být nezkracující, nesmí být v žádném kroku vygenerováno celkem více než $3i$ znaků. Jedna z možných gramatik je

$$G = (\{S, B, D\}, \{a, b, c\}, S, P),$$

kde P obsahuje pravidla

$$\begin{aligned} S &\to aBD \mid \Lambda \\ D &\to c \mid aBDc \\ Ba &\to aB \\ Bc &\to bc \\ Bb &\to bb \end{aligned}$$

Generování v G probíhá tak, že nejprve vygeneruje slovo aBD[2]. Neterminálu D se může zbavit pouze přepisem na c. Než k tomu dojde, mohla být použita pouze pravidla druhého a třetího řádku. Pravidlo ve třetím řádku nám však nemění počty terminálů a neterminálů ve slově a pravidlo druhého řádku splňuje invariant $\#a = \#B = \#c + 1$, kde $\#x$ značí počet výskytů znaku x v dosud vygenerovaném slově. Použitím $D \to c$ vyrovnáme počty a, B, c a odstraníme D. Teď již přicházejí v úvahu pouze pravidla ve třetím, čtvrtém a pátém řádku definice P, jejichž použitím vygenerujeme slovo z L.

Zbývá zapsat posloupnost derivací v G, kterými lze odvodit slovo *aabbcc*. Jednou z možných posloupností je

$$S \overset{G}{\Longrightarrow} aBD \overset{G}{\Longrightarrow} aBaBDc \overset{G}{\Longrightarrow} aaBBDc \overset{G}{\Longrightarrow}$$
$$\overset{G}{\Longrightarrow} aaBBcc \overset{G}{\Longrightarrow} aaBbcc \overset{G}{\Longrightarrow} aabbcc.$$

FINE

Definice 5.6 *O bezkontextové gramatice $G = (V_N, V_T, S, P)$ řekneme, že je ve standartní formě, pokud splňuje následující podmínky:*

1. *S není na pravé straně žádného pravidla*

[2] prázdné slovo nás teď nezajímá

2. $(X \to \Lambda) \in P \Rightarrow X = S$
3. $(X \to \alpha) \in P \Rightarrow \alpha \notin V_N$ *(neboli α není právě jeden neterminál)*

Každému je jistě zřejmé, že v derivacích v bezkontextových gramatikách se v jednom kroku (přímém odvození) přepisuje právě jeden neterminál, nicméně myslím, že je vhodné tuto trivialitu explicitně vyjádřit.

Lemma 5.7 *Pro každou bezkontextovou gramatiku $G = (V_N, V_T, S, P)$ existuje s ní ekvivalentní bezkontextová gramatika $\tilde{G}$ ve standartní formě. Je-li navíc G regulární, je i $\tilde{G}$ regulární.*

Důkaz: Nechť $\tilde{S} \notin V_N$. Definujme $\tilde{G} = (V_N \cup \{\tilde{S}\}, V_T, \tilde{S}, \tilde{P})$. Množinu $\tilde{P}$ zkonstruujeme z P ve třech fázích odpovídajících definici gramatiky ve standartní formě. Jako inicializační krok zkopírujeme celé P do $\tilde{P}$.

1. Za každé pravidlo $(S \to \alpha) \in P$ přidáme[3] do $\tilde{P}$ pravidlo $(\tilde{S} \to \alpha)$. Počáteční symbol $\tilde{S}$ se tudíž na pravé straně žádného pravidla neobjeví a zřejmě $L(G) = L(\tilde{G})$.
2. Označme $A \subseteq V_N$ množinu všech neterminálů, které lze nějakým odvozením přepsat na Λ[4]. Pokud $S \in A$, přidáme do $\tilde{P}$ pravidlo $\tilde{S} \to \Lambda$. Dále pro všechna $X \in V_N \setminus \{\tilde{S}\}$
   - vyřadíme z $\tilde{P}$ pravidlo $(X \to \Lambda)$ (pokud tam vůbec bylo)
   - a pro každé pravidlo $(X \to \alpha)$, které zůstalo v $\tilde{P}$ a pro každé vyjádření $\alpha = \alpha_1 Z_1 \alpha_2 Z_2 \dots Z_{n-1} \alpha_n$, kde $\alpha_i \in (V_N \cup V_T)^*$, $\alpha_1 \dots \alpha_n \neq \Lambda$ a $Z_i \in A$, přidáme do $\tilde{P}$ pravidlo $(X \to \alpha_1 \alpha_2 \dots \alpha_n)$[5].

   Teď je třeba dokázat, že $L(G) = L(\tilde{G})$.

   - Mějme $\alpha \in L(\tilde{G})$. Slovo α je postupně odvozováno pravidly $\tilde{P}$, přičemž některá jsou i v P. V derivaci slova α vezměme druhé pravidlo $(X \to \alpha') \in \tilde{P}$, které není v P[6]. Takové pravidlo se mohlo do $\tilde{P}$ dostat pouze tak, že bylo odvozeno z některého pravidla

$$p = (X \to \alpha_1 Z_1 \alpha_2 Z_2 \dots Z_{n-1} \alpha_n) \in P$$

[3] tzn. $(S \to \alpha)$ v $\tilde{P}$ zůstane

[4] množinu $A = \{X \in V_N; X \overset{G}{\underset{*}{\Longrightarrow}} \Lambda\}$ je možné získat například rekursivně — od množiny

$$A_0 = \{X \in V_N; (X \to \Lambda) \in P\}$$

postupně přecházet k delším odvozením, která přepisují neterminály na Λ, tzn.

$$A_i = \{X \in V_N; (X \to \alpha) \in P, \alpha \in A_{i-1}^*\}.$$

Zřejmě platí $A_{i-1} \subseteq A_i$. $A = \bigcup_{i \in N_0} A_i$ a neboť V_N je konečná množina, je i A konečná, tudíž existuje k takové, že $A_k = A$.

[5] Všimněte si, že prvky z A mohou být obsaženy i v α_i, tzn. $Z_1, \dots, Z_{n-1}$ nemusí být všechny prvky z A obsažené v α; výsledek je ten, že na pravých stranách přidaných pravidel získáme všechny možné kombinace vynechání některých písmen z A v α.

[6] první pravidlo z $\tilde{P}$, které není v P je $(\tilde{S} \to \beta)$, které nás nyní nezajímá (bylo náplní první části důkazu)

vynecháním všech $Z_i \in A$, tedy mělo tvar například

$$(X \to \alpha_1\alpha_2 \ldots \alpha_n) = \tilde{p} \in \tilde{P}.$$

Pravidlo $\tilde{p}$ můžeme však v P simulovat jako odvození, které začíná v X a pravou stranu p převádí na $\tilde{p}$ postupnými přepisy Z_i na Λ (takové přepisy existují z definice množiny A), tzn. $\alpha \in L(G)$.

- Mějme naopak slovo $\alpha \in L(G)$. Když $\alpha = \Lambda$, pak $S \in A$ a tedy P obsahuje pravidlo $(S \to \beta)$, kde $\beta \in A^*$ a tedy pravidlo $(\tilde{S} \to \Lambda) \in \tilde{P}$ a proto $\alpha \in L(\tilde{G})$.

  Předpokládejme $\alpha \neq \Lambda$. Vezměme derivaci

  $$S \Rightarrow \beta_1 \Rightarrow \beta_2 \ldots \Rightarrow \beta_n = \alpha$$

  v gramatice G a budeme postupně zatrhávat písmena ve slovech β_i $(i = n, \ldots, 1)$. Ve slově $\beta_n = \alpha$ není zaškrtnuto žádné písmeno.
  - Když $\beta_i = \beta_{i,0}X\beta_{i,1}$, $\beta_{i+1} = \beta_{i,0}\beta_{i,1}$ a $(X \to \Lambda) \in P$, pak zatrhneme písmeno X v β_i.
  - Dále když $\beta_i = \beta_{i,0}X\beta_{i,1}$, $\beta_{i+1} = \beta_{i,0}\beta_{i,2}\beta_{i,1}$ a $(X \to \beta_{i,2}) \in P$ a celé $\beta_{i,2}$ je ve slově β_{i+1} zatrženo, pak zatrhneme X ve slově β_i.

  Všimněte si, že všechna zatržená písmena jsou z A a v derivaci α se přepíšou na Λ.

  Nyní budeme indukcí konstruovat derivaci

  $$\tilde{S} \Rightarrow \gamma_1 \Rightarrow \gamma_2 \ldots \Rightarrow \gamma_m = \alpha$$

  v gramatice $\tilde{G}$. Ke každému slovu β_i bude přiřazeno slovo γ_j, které vznikne ze slova β_i vynecháním všech zatržených písmen. Řekneme, že takové γ_j odpovídá slovu β_i. Protože $\alpha \neq \Lambda$, není S zatržené a tedy ani nejsou všechna písmena v β_1 zatržena.

  Slovo vzniklé z β_1 vynecháním všech zatržených písmen označme γ_1. Zřejmě γ_1 odpovídá β_1 a z definice A platí $(\tilde{S} \to \gamma_1) \in \tilde{P}$. Mějme β_i a nechť mu odpovídá γ_j. Předpokládejme, že $\beta_i = \beta_{i,0}X\beta_{i,1}$, $\beta_{i+1} = \beta_{i,0}\beta_{i,2}\beta_{i,1}$ a nechť $(X \to \beta_{i,2}) \in P$. Když X je zatržené v β_i, pak každé písmeno $\beta_{i,2}$ je zatrženo v β_{i+1} a γ_j odpovídá β_{i+1}. Uvažujme, že X není v β_i zatrženo. Označme postupně $\gamma_{j,0}$, $\gamma_{j,1}$, $\gamma_{j,2}$ slova, která vzniknou ze slov $\beta_{i,0}$, $\beta_{i,1}$, $\beta_{i,2}$ vynecháním zatržených písmen. Pak $\gamma_j = \gamma_{j,0}X\gamma_{j,1}$ a protože některá písmena z $\beta_{i,2}$ ve slově β_{i+1} nejsou zatržená, dostáváme $\gamma_{j,2} \neq \Lambda$ a tedy $(X \to \gamma_{j,2})$ je pravidlo z $\tilde{P}$. Pak $\gamma_{j+1} = \gamma_{j,0}\gamma_{j,2}\gamma_{j,1}$ je přímo odvozeno z γ_j a γ_{j+1} odpovídá slovu β_{i+1}. Protože α neobsahuje zatržené symboly, dostáváme pro nějaké m, že $\gamma_m = \alpha$ odpovídá $\beta_n = \alpha$ a proto $\alpha \in L(\tilde{G})$.

Z obou inkluzí platí $L(G) = L(\tilde{G})$.

3. Máme již bezkontextovou gramatiku $\tilde{G}$ splňující první dvě podmínky z definice 5.6. Zbývá třetí podmínka. Dovolím si tuto část konstrukce $\tilde{G}$ poněkud zatemnit formalitami, neboť je analogická přechozí části, která byla naopak poněkud neformální. Definujme relaci ρ tak, že

$$(\forall\, X, Y \in V_N)[(X, Y) \in \rho \overset{df}{\equiv} (X \overset{\tilde{G}}{\underset{*}{\Longrightarrow}} Y)]$$

Relace ρ je zřejmě tranzitivní a reflexivní[7]. Z $\tilde{P}$ vyřadíme všechna pravidla $(X \to Y)$, kde $X, Y \in V_N$, a za každé pravidlo

$$(X \to \alpha_1 Z_1 \alpha_2 Z_2 \dots Z_{n-1} \alpha_n)$$

přidáme všechna pravidla

$$(X \to \alpha_1 Y_1 \alpha_2 Y_2 \dots Y_{n-1} \alpha_n)$$

taková, že $(\forall\, i = 1, \dots n-1)\,((Z_i, Y_i) \in \rho)$. Důkaz rovnosti $L(G) = L(\tilde{G})$ by byl velmi podobný jako v bodu 2 a opět by se opíral o fakt, že odvození v G lze simulovat odvozením v $\tilde{G}$ a naopak, což si formálně může každý provést sám. Velmi názorný důkaz této části lze provést přes derivační stromy, o kterých se zmíním později.

Z konstrukce $\tilde{G}$ jednoznačně vyplývá, že $\tilde{G}$ bude regulární, kdykoli G je regulární. CBD

Tvrzení 5.8 *Jazyky třídy $\mathcal{L}_3$ jsou regulární ve smyslu definice regulárních jazyků přes automaty.*

Důkaz: Spočívá v tom, že k libovolnému jazyku $L \in \mathcal{L}_3$ najdeme akceptor $\mathcal{A}$ tak, že $L = L(\mathcal{A})$, a naopak k akceptoru $\mathcal{A}$ najdeme regulární gramatiku G takovou, že $L(G) = L(\mathcal{A})$

1. Mějme regulární gramatiku G ve standartní formě[8]. Sestrojíme nejprve gramatiku $\tilde{G} = (\tilde{V}_N, V_T, S, \tilde{P})$, která bude také regulární ve standartní formě a navíc bude splňovat

$$\begin{aligned}(\forall\, X, Y \in V_N \setminus \{S\})(\forall\, \alpha \in V_T^*) \quad & [(X \to \alpha Y) \in \tilde{P} \Rightarrow |\alpha| = 1] \\ (\forall\, X \in V_N \setminus \{S\})(\forall\, \alpha \in V_T^*) \quad & [(X \to \alpha) \in \tilde{P} \Rightarrow |\alpha| = 1] \\ (\forall\, \alpha \in V_T^*) \quad & [(S \to \alpha) \in \tilde{P} \Rightarrow |\alpha| \leq 1]\end{aligned}$$

   Je to jednoduché. Jako inicializaci všechna pravidla z P zkopírujeme do $\tilde{P}$. Teď je $\tilde{G}$ jistě ve standartní formě. Budeme vyřazovat (přidávat) pravidla z (do) $\tilde{P}$ tak, aby se zachovala standartní forma $\tilde{G}$ a jazyk generovaný G. Každé pravidlo $\tilde{p} = (X \to a_1 \dots a_n Y) \in \tilde{P}$[9] z $\tilde{P}$ vyhodíme a nahradíme sérií pravidel

$$\begin{aligned}& (X \to a_1 X_1), \\ (\forall\, i = 1, \dots n-2) \quad & [(X_i \to a_{i+1} X_{i+1})], \\ & (X_{n-1} \to a_n Y),\end{aligned}$$

   kde X_i jsou nové neterminály, dosud se ve $\tilde{V}_N$ nevyskytující[10]. Tato pravidla odpovídají požadavkům na $\tilde{P}$, zachovávají standartní formu $\tilde{G}$ a

[7] ne však nutně symetrická, nemusí tedy být ekvivalencí

[8] tímto požadavkem neztrácí důkaz díky tvrzení 5.7 na obecnosti

[9] $Y \in V_N \cup \{\Lambda\}$, $X \in V_N$, může být tedy $X = S$

[10] s přidáním nových pravidel je samozřejmě nutné množinu $\tilde{V}_N$ patřičně rozšířit

jimi vytvořená derivace zřejmě odpovídá původnímu pravidlu $\tilde{p}$, tudíž $L(G) = L(\tilde{G})$. Pro gramatiku $\tilde{G}$ sestrojíme N-akceptor $\mathcal{A}$, pro který bude platit $L(\mathcal{A}) = L(G)$. Nechť $f \notin \tilde{V}_N \cup V_T$. Pak definujeme

$$\mathcal{A} = (\tilde{V}_N \cup \{f\}, V_T, \delta, \{S\}, F),$$

kde

- $F = \begin{cases} \{f\}, \text{ pokud } (S \to \Lambda) \notin \tilde{P} \\ \{f, S\}, \text{ pokud } (S \to \Lambda) \in \tilde{P} \end{cases}$
- $\delta(X, a) = \begin{cases} \{Y \in \tilde{V}_N; X \to aY \in \tilde{P}\} \cup \{f\}, \\ \qquad \text{když } (\exists\, a \in V_T)((X \to a) \in \tilde{P}) \\ \{Y \in \tilde{V}_N; X \to aY \in \tilde{P}\}, \\ \qquad \text{když } (\forall\, a \in V_T)((X \to a) \notin \tilde{P}) \end{cases}$

Dokážeme $L(G) = L(\mathcal{A})$. Nechť $\alpha = a_1 \dots a_n$, $\Lambda \neq \alpha$. Pak $\alpha \in L(\tilde{G})$, právě když existuje derivace α z S v $\tilde{G}$, tzn.

$$S \overset{\tilde{G}}{\Longrightarrow} a_1 X_1 \overset{\tilde{G}}{\Longrightarrow} \dots \overset{\tilde{G}}{\Longrightarrow} a_1 \dots a_{n-1} X_{n-1} \overset{\tilde{G}}{\Longrightarrow} a_1 \dots a_n,$$

což je podle definice δ nastane právě tehdy, je-li

$$\begin{aligned} & X_1 \in \delta(S, a_1), \\ (\forall\, i = 1, \dots, n-2) \quad & [X_{i+1} \in \delta(X_i, a_{i+1})], \\ & f \in \delta(X_{n-1}, a_n), \end{aligned}$$

čili $\alpha \in L(\mathcal{A})$. Protože všechny provedené kroky byly ekvivalence a prázdné slovo Λ je zřejmě přijímáno v $\mathcal{A}$ právě když patří do $L(\tilde{G})$, můžeme psát $L(\tilde{G}) = L(\mathcal{A})$.

2. Nyní mějme $\mathcal{A} = (Q, X, \delta, I, F)$ nedeterministický akceptor. Nechť S je nějaký symbol mimo $Q \cup X$. Definujme gramatiku $G = (Q \cup \{S\}, X, S, P)$, kde pro P platí

$$\begin{aligned} (\forall\, q \in I) \quad & [(S \to q) \in P] \\ (\forall\, q, r \in Q)(\forall\, a \in X) \quad & [q \in \delta(r, a) \Rightarrow (r \to aq) \in P] \\ (\forall\, r \in Q)(\forall\, a \in X) \quad & [F \cap \delta(r, a) \neq \emptyset \Rightarrow (r \to a) \in P] \end{aligned}$$

Dokážeme, že G generuje $L(\mathcal{A})$.

(a) Nechť $\alpha = a_1 \dots a_n$ je slovo z $L(\mathcal{A})$. Pak existuje přijímací výpočet $q_0, \dots, q_n$, kde

$$\begin{aligned} & q_0 \in I \\ & q_n \in F \\ (\forall\, i = 1, \dots, n-1) \quad & [q_{i+1} \in \delta(q_i, a_{i+1})]. \end{aligned}$$

Pak ovšem posloupnost

(37) $S \Rightarrow q_0 \Rightarrow a_1 q_1 \Rightarrow \dots \Rightarrow a_1 \dots a_{n-1} q_{n-1} \Rightarrow a_1 \dots a_n$

je z definice P derivací v G, neboli $\alpha \in L(G)$ a $L(\mathcal{A}) \subseteq L(G)$.

(b) Je-li $\alpha = a_1 \dots a_n \in L(G)$, pak $S \overset{G}{\underset{*}{\Longrightarrow}} a_1 \dots a_n$. Vzhledem k pravidlům v P jediný možný tvar derivace α z S je (37), což nastane pouze pro

$$\begin{aligned} & q_0 \in I, \\ (\forall\, i = 1, \dots, n-2) \qquad & [q_{i+1} \in \delta(q_i, a_{i+1})], \\ & q_n \in F \cap \delta(q_{n-1}, a_n) \neq \emptyset. \end{aligned}$$

Pak ovšem $q_0, \dots, q_n$ je přijímací výpočet $\mathcal{A}$ nad α, tedy $\alpha \in L(\mathcal{A})$ a $L(G) \subseteq L(\mathcal{A})$.

Z obou inkluzí dostáváme rovnost $L(G) = L(\mathcal{A})$. CBD

Když už víme, že definice regulárních jazyků si neodporují, ale jsou ve shodě, můžeme si dokázat jednu ze snadnějších inkluzí Chomského hierarchie (36), konkrétně $\mathcal{L}_3 \subset \mathcal{L}_2$. Každý regulární jazyk je bezkontextový (tedy $\mathcal{L}_3 \subseteq \mathcal{L}_2$), neboť pravidla regulárních gramatik splňují požadavky kladené na pravidla gramatik bezkontextových. Naproti tomu existuje jazyk $L = \{a^n b^n; n = 0, 1, \dots\}$, který podle příkladu 4.25 není regulární, ale je generován bezkontextovou gramatikou

$$(\{S\}, \{a, b\}, S, \{S \to aSb \mid \Lambda\})$$

Věta 5.9 *Nezkracující gramatiky generují právě kontextové jazyky.*

Důkaz: Mějme kontextovou gramatiku $G_L = (V_N, V_T, S, P)$, generující jazyk L. G_L zřejmě splňuje druhou podmínku z definice nezkracující gramatiky. Použijeme otřepaný trik a upravíme G_L tak, aby splňovala obě podmínky a aby se jazyk $L(G_L)$ nezměnil. Zavedeme nový počáteční neterminál S_1 a za každé pravidlo $(S \to \alpha) \in P$ přidáme $(S_1 \to \alpha)$ do P[11]. G_L je nyní jistě nezkracující, tedy každou kontextovou gramatiku lze převést na nezkracující.

Naopak. Nechť G je nezkracující gramatika. Podívejme se, jaká pravidla mohou překážet tomu, aby G byla kontextová. Pravidlo $(S \to \Lambda)$ nám nevadí, neboť z definice nezkracující gramatiky (str. 62) není S na pravé straně žádného pravidla. Dále jsou „v pořádku“ pravidla $(\alpha \to \beta)$, ve kterých existuje $X \in V_N$, $X \in \alpha$ takové, že $\alpha = \alpha_1 X \alpha_2$, $\beta = \alpha_1 \gamma \alpha_2$, kde $\gamma \in (V_N \cup V_T)^+$[12]. Na obtíž tedy mohou být buď pravidla, kde $\gamma = \Lambda$ (což ale nemůže nastat, neboť by bylo $|\alpha| > |\beta|$, neboli G by nebyla nezkracující), nebo kde neexistuje X splňující uvedené požadavky. Taková pravidla mohou existovat[13], leč my se s nimi vypořádáme a upravíme G tak, aby byla kontextová. Nechť

$$p = (A_1 \dots A_n \to B_1 \dots B_m) \in P,$$

kde $A_i, B_i \in V_N \cup V_T$, není kontextové. Neboť G je nezkracující, platí zřejmě $m \geq n$. Dále nechť M je množina všech nekontextových pravidel p z P. Tato pravidla z G vyhodíme a každé nahradíme ekvivalentní posloupností kontextových pravidel. Označme T množinu všech terminálů vyskytujících se v množině

[11] podrobněji viz také první část důkazu lemmatu 5.7

[12] vyplývá triviálně z definice kontextové gramatiky

[13] např. $(XY \to YX)$ je pravidlo nezkracující, ale ne kontextové

M. Za každý terminál $t \in T$ přidáme do V_N nový neterminál A_t a do P pravidlo $(A_t \to t)$ a ve všech pravidlech z P nahradíme t neterminálem A_t. Jazyk $L(G)$ se zatím jistě nezměnil, neboť jediná možnost, jak se zbavit terminálů A_t, je přepsat je na t. Nyní máme v M pouze neterminály. Je-li

$$p = (A_1 \dots A_n \to B_1 \dots B_m) \in M$$

přidáme do V_N nové neterminály $C_1, \dots, C_{n-1}$ a do P pravidla

$$\begin{aligned}
A_1A_2 \dots A_n &\to C_1A_2 \dots A_n \\
C_1A_2 \dots A_n &\to C_1C_2A_3 \dots A_n \\
&\vdots \\
C_1 \dots C_{n-2}A_{n-1}A_n &\to C_1 \dots C_{n-1}A_n \\
C_1 \dots C_{n-1}A_n &\to C_1 \dots C_{n-1}B_n \dots B_m \\
C_1 \dots C_{n-1}B_n \dots B_m &\to C_1 \dots C_{n-2}B_{n-1}B_n \dots B_m \\
&\vdots \\
C_1B_2 \dots B_m &\to B_1B_2 \dots B_m
\end{aligned}$$

Výše uvedená posloupnost je nepochybně tvořena kontextovými pravidly a můžeme jí pravidlo p nahradit. Naopak jakmile jednou použijeme prvního pravidla posloupnosti, pak se neterminálů C_i zbavíme až posledním pravidlem, tedy přidaná posloupnost negeneruje víc než p, neboli jazyk $L(G)$ nedoznal změny a G je nyní kontextová.

Každou nezkracující gramatiku lze převést na kontextovou a naopak, čímž je důkaz proveden. CBD

Příklad 5.10 *(Nezkracující→Kontextová) Nezkracující gramatiku*

$$G = (\{S, B, D\}, \{a, b, c\}, S, P),$$

kde P obsahuje pravidla

$$\begin{aligned}
S &\to aBD \mid \Lambda \\
D &\to c \mid aBDc \\
Ba &\to aB \\
Bc &\to bc \\
Bb &\to bb,
\end{aligned}$$

převeďte na ekvivalentní kontextovou gramatiku G'.[14]

Jak snadno zjistíme z definice Chomského hierarchie, jediné pravidlo, které dělá z gramatiky G nekontextovou, je $p = (Ba \to aB)$. Problém vyřešíme podle návodu poskytnutého předchozí větou.

Jako inicializační krok položíme

$$G' = G = (\{S, B, D\}, \{a, b, c\}, S, P).$$

G' je teď jistě ekvivalentní s G, ale jako na potvoru není kontextová, což asi nikoho nepřekvapí.

Věta 5.9 říká, že nic není ztraceno. Naopak tvrdí, že[15] lze P modifikovat

[14] pokud si vzpomínáte, byla gramatika G již jednou vzpomenuta v příkladu 5.5

[15] a dokonce jak

na ekvivalentní systém P', obsahující jen kontextová pravidla[16]. Stačí, abychom pravidlo p zaměnili za sekvenci kontextových pravidel, která ho budou jedno-jednoznačně nahrazovat. Zavedeme nový neterminál A_a, kterým ve všech pravidlech z P nahradíme písmeno a, a přidáme pravidlo $p_a = (A_a \to a)$. Dále obohatíme množinu neterminálů v G' o nový neterminál C_1 a přepisovací systém P'[17] o pravidla

$$\begin{aligned} BA_a &\to BC_1 \\ BC_1 &\to A_aC_1 \\ A_aC_1 &\to A_aB, \end{aligned}$$

která jsou již kontextová a podle důkazu věty 5.9 nahrazují spolu s pravidlem p_a jedno-jednoznačně pravidlo $p = (BA_a \to A_aB)$, které z P' samozřejmě vyloučíme.

Zkonstruovali jsme gramatiku $G' = (\{S, B, D, A_a, C_1\}, \{a, b, c\}, S, P')$, která je kontextová a ekvivalentní s G. Systém P' obsahuje pravidla

$$\begin{aligned} S &\to A_aBD \mid \Lambda \\ D &\to c \mid A_aBDc \\ Bc &\to bc \\ Bb &\to bb \\ BA_a &\to BC_1 \\ BC_1 &\to A_aC_1 \\ A_aC_1 &\to A_aB \\ A_a &\to a. \end{aligned}$$

FINE

5.2 Bezkontextové gramatiky

5.2.1 Normální formy

Definice 5.11 *Bezkontextová gramatika $G = (V_N, V_T, S, P)$ je v v Chomského normální formě, pokud každé pravidlo z P má jeden z následujících tvarů:*

1. $X \to YZ$, *kde* $X, Y, Z \in V_N$
2. $X \to a$, *kde* $X \in V_N, a \in V_T$
3. $S \to \Lambda$

Gramatiku v Chomského normální formě lze triviálně pozměnit tak, aby byla ve standartní formě (viz definice 5.6)[18].

[16] Ekvivalenci na přepisovacích systémech jsme sice nedefinovali, ale intuice jistě napovídá, že přepisovací systémy dvou gramatik G, G' jsou ekvivalentní, právě když je G ekvivalentní s G'.

[17] P' je vytvářený přepisovací systém gramatiky G', pokud to není jasné...

[18] stačí přidat nový počáteční neterminál S_1 a pokud S byl původní počáteční neterminál, pak za každé pravidlo $(S \to \alpha)$ přidáme $(S_1 \to \alpha)$, čímž počáteční neterminál nebude na pravé straně žádného pravidla a splníme první podmínku z definice 5.6. Ostatní podmínky jsou z definice Chomského normální formy okamžitě splněny

Tvrzení 5.12 *Pro každou bezkontextovou gramatiku G existuje s ní ekvivalentní bezkontextová gramatika G' v Chomského normální formě.*

Důkaz: Tradičně budeme nahrazovat pravidla z G tak dlouho, dokud nebudou v Chomského normální formě. Búno[19] můžeme předpokládat, že G je ve standartní formě, čímž se zbavíme povinnosti upravovat pravidla $(X \to \Lambda)$ a $(X \to Y)$ pro $(X, Y \in V_N)$. Jako inicializaci položme

$$G' = (V'_N, V_T, S, P') = (V_N, V_T, S, P)$$

(neboli $L(G) = L(G')$). Vezměme pravidlo

$$p = (X \to A_1 \dots A_n) \in P', A_i \in V_N \cup V_T.$$

Je-li $n = 0$, musí být $X = S$. Je-li $n = 1$, pak $A_1 \in V_T$[20]. Nechť tedy $n \geq 2$. Potom každý terminál $A_j, j \in J \subseteq \langle 1, n \rangle$ nahradíme v p novým neterminálem $A_j^N \in V'_N$ a do P' přidáme pravidlo $(A_j^N \to A_j)$. Jazyk $L(G')$ se zřejmě nezměnil a v pravidle p jsou nyní samé neterminály. Do V'_N teď přidáme nové neterminály $B_1, \dots B_{n-2}$ a p z P vyhodíme a nahradíme posloupností pravidel

$$\begin{aligned} X &\to A_1 B_1 \\ B_1 &\to A_2 B_2 \\ &\vdots \\ B_{n-2} &\to A_{n-1} A_n \end{aligned}$$

Tak nahradíme všechna původní pravidla z P'. Platnost $L(G) = L(G')$ je poměrně zřejmá a připadá mi zbytečné ji dokazovat, neboť v podobných případech tak již bylo několikrát učiněno např. v důkazech lemmatu 5.7 a tvrzení 5.8. CBD

Definice 5.13 *Bezkontextovou gramatiku G nazveme v Greibachově normální formě, pokud její pravidla mají tvar $(X \to a\alpha)$, kde $a \in V_T, \alpha \in V_N^*$, $S \notin \alpha$. Navíc může G obsahovat pravidlo $(S \to \Lambda)$.*

Bezkontextovou gramatiku G nazveme v Hopfově normální formě, pokud její pravidla mají tvar $(X \to a\alpha b)$, kde $a, b \in V_T$, $\alpha \in V_N^*$, $S \notin \alpha$. Navíc může G obsahovat pravidlo $(S \to \Lambda)$ a pravidla $(S \to a)$ pro $a \in V_T$.

K předcházejícím definicím vedlo časté praktické používání gramatik s uvedenými vlastnostmi, podobně jako když jsme definovali standartní formu. Lze dokázat, že každou bezkontextovou gramatiku můžeme převést na gramatiku ve kterékoliv z uvedených forem. Tento problém však ponechávám jako cvičení. Návod na řešení poskytuje např. důkaz tvrzení 5.12 nebo lemmatu 5.7.

[19] za použití lemmatu 5.7

[20] Obojí vyplývá z toho, že $G' = G$ je ve standartní formě.

5.2.2 Derivační stromy

Intuitivnímu pochopení odvozování v bezkontextových gramatikách velmi pomáhají struktury, zvané výstižně *derivační stromy*. Přesná matematická definice by byla velmi složitá a nepřehledná, uvedu raději názornější definici a příklad.

Definice 5.14 *Strom T je derivačním stromem bezkontextové gramatiky G, pokud*

- *vnitřní vrcholy jsou ohodnocené neterminály*
- *listy jsou ohodnocené terminály nebo neterminály*
- *je-li A vnitřní neterminál se syny $B_1, \ldots B_k$ uspořádanými v rovinném nakreslení T zleva doprava, pak $(A \to B_1 \ldots B_k) \in P$.*

Derivačnímu stromu T přiřadíme slovo $\alpha = A_1 \ldots A_n$, kde A_i jsou ohodnocení všech listů stromu T přečtená zleva doprava.

Je zřejmé, že $\alpha \in L(G)$, právě když existuje derivační strom T gramatiky G, jehož kořen (nejvyšší vrchol) je ohodnocen počátečním neterminálem a slovo přiřazené stromu T je α.

Definice 5.15 *Levou (pravou) derivaci odpovídající derivačnímu stromu T získáme tak, že začneme písmenem ohodnocujícím kořen a použijeme rekursivně pravidla gramatiky G na nejlevější (nejpravější) list–neterminál, který je třeba přepsat.*

Příklad 5.16 *(Derivační strom) Sestrojte v gramatice*

$$G = (\{S, A\}, \{a, b, c\}, S, \{S \to aSb \mid A, A \to aA \mid Ab \mid c\})$$

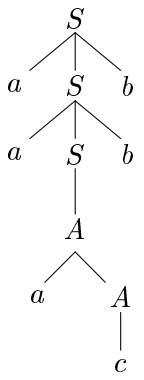
derivační strom slova $\alpha = aaacbb$.

Nejprve si z cvičných důvodů uvědomíme, jaký jazyk gramatika G generuje. Asi každý po chvilce uvažování uzná, že $L(G) = \{a^i c b^j; i, j = 1, 2, \ldots\}$. Jeden z derivačních stromů slova α je na obrázku 10. FINE

Věta 5.17 *(Pumping lemma) Když $G = (V_N, V_T, S, P)$ je bezkontextová gramatika, pak existují čísla $m, n \in N_0$ taková, že každé slovo $\alpha \in L(G)$, $|\alpha| \geq n$ lze napsat jako $\alpha = uvwxy$[21] tak, že platí*

1. $|vwx| < m$
2. $v \neq \Lambda \vee x \neq \Lambda$
3. $(\forall\, i \in N_0)(uv^i wx^i y \in L(G))$

[21] samozřejmě $u, v, w, x, y \in V_T^*$



Obr. 10: Derivační strom slova *aaacbb*

Tato věta má *velmi podstatný* význam, chceme-li dokázat, že nějaký jazyk není bezkontextový. Aplikaci na příkladě ukážeme později. Nejprve je třeba větu dokázat.

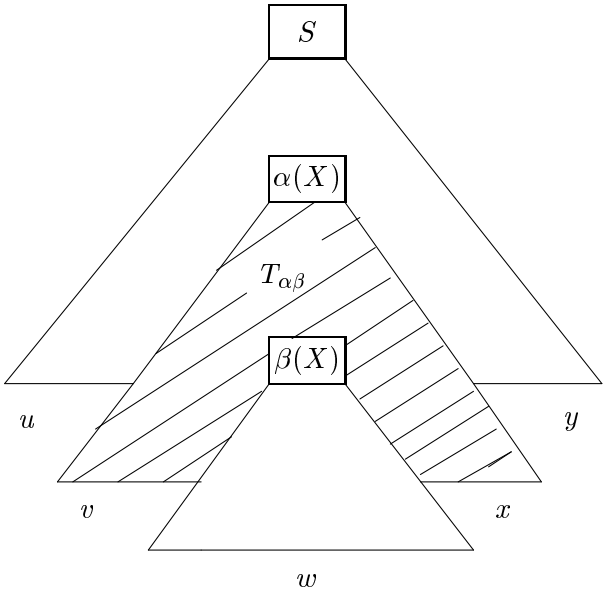
Důkaz: Budeme se snažit pro dané slovo $\alpha \in L(G)$ najít požadovaný rozklad $uvwxy$. K tomu použijeme derivačních stromů[22].

Označme $\mathcal{T}$ množinu všech derivačních stromů T gramatiky G, které mají za kořen S, listy ohodnocené terminály a na žádné cestě z kořene do listu neexistují dva různé uzly ohodnocené stejným neterminálem. Protože V_N je konečná, je délka každé cesty kořen–list v $T \in \mathcal{T}$ menší nebo rovna $|V_N|$, neboli stromů $T \in \mathcal{T}$ je konečně mnoho[23], neboli můžeme najít n takové, že každý strom z $\mathcal{T}$ má méně než n listů. Vezmeme-li slovo $\alpha \in L(G)$, $|\alpha| \geq n$, žádný jeho derivační strom není v $\mathcal{T}$[24]. Protože $T \notin \mathcal{T}$, existují v T na jedné cestě kořen–list dva různé uzly (označme je α, β, nechť α je výš než β) ohodnocené stejným neterminálem X — viz obrázek 11. Potom ale část T mezi α, β (označme ji

[22] nač bychom je také jinak měli...

[23] intuitivně to asi každý chápe, ale zkuste si provést korektní důkaz, neboli odhadnout počet možných permutací konečně mnoha terminálů a neterminálů v konečně mnoha hladinách. Můj poměrně velmi hrubý horní odhad je $\prod_{i=1}^{|V_N|} (|V_N \cup V_T| + 1)^{g^i}$, kde g je maximální délka pravé strany libovolného pravidla v G. Toto je koneckonců konečné číslo. Jak jsme dospěli k tomuto číslu? Rozšiřme každý derivační strom z $\mathcal{T}$ na strom, kde všechny listy jsou ve $|V_N|$-té hladině a každý vnitřní vrchol má g synů. Přidané vrcholy ohodnotíme novým symbolem, který nepatří do množiny $V_N \cup V_T$. Pak v i-té hladině je g^i vrcholů a proto existuje $(|V_N \cup V_T| + 1)^{g^i}$ ohodnocení vrcholů v i-té hladině. Tedy celkový počet rozšířených stromů je menší než $\prod_{i=1}^{|V_N|} (|V_N \cup V_T| + 1)^{g^i}$ stromů a to dává také horní odhad počtu derivačních stromů v $\mathcal{T}$

[24] neboť všechny derivační stromy pro α mají více než n listů



Obr. 11: Rozklad slova z bezkontextového jazyka podle Pumping lemmatu

$T_{\alpha\beta}$) můžeme vynechat, nebo i libovolněkrát zopakovat[25] a dostaneme derivační stromy T_0 (vynecháním $T_{\alpha\beta}$), $T_1 = T$, T_2 dvojím zopakováním $T_{\alpha\beta}$,... T_i jsou derivační stromy v G a generují slova $uwy, uvwxy, uvvwxxy, \ldots$[26], neboli $(\forall\, i \in N_0)(uv^iwx^iy \in L(G))$, čímž je dokázána třetí část věty.

Označme $\mathcal{T}_1$ množinu všech derivačních stromů T, které mají za listy terminály a jsou-li dva různé uzly na cestě kořen-list ohodnoceny stejným neterminálem, pak jeden z těchto prvků je kořen[27]. Podobně jako $\mathcal{T}$ je i $\mathcal{T}_1$ konečná, přestože větší než $\mathcal{T}$, a existuje tedy m takové, že každý strom z $\mathcal{T}_1$ má méně než m listů, neboli $|vwx| \leq m$. Tím máme první část věty.

Búno můžeme předpokládat, že G je ve standartní formě[28]. Pak je-li X neterminál ohodnocující kořen a zároveň některý další uzel nějakého stromu $T \in \mathcal{T}_1$, nemůže se přepsat na Λ ani na samotné X, neboli v odvození $X \overset{G}{\underset{*}{\Longrightarrow}} vXx$

---

[25] Strom na obrázku 11 odpovídá zřejmě odvození $S \underset{*}{\Longrightarrow} uXy \overset{T_{\alpha\beta}}{\underset{*}{\Longrightarrow}} uvXxy \underset{*}{\Longrightarrow} uvwxy$, neboli existují derivace $X \overset{G}{\underset{*}{\Longrightarrow}} vXx$, $X \overset{G}{\underset{*}{\Longrightarrow}} w$ (protože G je bezkontextová). Tím pádem jsou

$$S \underset{*}{\Longrightarrow} uXy \underset{*}{\Longrightarrow} uwy$$
$$S \underset{*}{\Longrightarrow} uXy \overset{T_{\alpha\beta}}{\underset{*}{\Longrightarrow}} uvXxy \overset{T_{\alpha\beta}}{\underset{*}{\Longrightarrow}} uvvXxxy \underset{*}{\Longrightarrow} \cdots \underset{*}{\Longrightarrow} uv^iXx^iy \underset{*}{\Longrightarrow} uv^iwx^iy,$$

derivace v G, které se liší pouze počtem zopakování části $T_{\alpha\beta}$.

[26] derivační strom T_i reprezentuje odvození

$$S \overset{G}{\underset{*}{\Longrightarrow}} uXy \overset{G}{\underset{*}{\Longrightarrow}} uvXxy \overset{G}{\underset{*}{\Longrightarrow}} uv^2Xx^2y \overset{G}{\underset{*}{\Longrightarrow}} \ldots \overset{G}{\underset{*}{\Longrightarrow}} uv^iXx^iy \overset{G}{\underset{*}{\Longrightarrow}} uv^iwx^iy$$

[27] tato definice znamená, že stromy z $\mathcal{T}_1$ reprezentují odvození $X \overset{G}{\underset{*}{\Longrightarrow}} vXx \overset{G}{\underset{*}{\Longrightarrow}} vwx$

[28] máme na to přeci lemma 5.7

je buď $v \neq \Lambda$ nebo $x \neq \Lambda$, což druhé tvrzení věty. CBD

Příklad 5.18 *(Pumping lemma) Ukažte, že jazyk*

$$L = \{a^i b^i c^i; i = 0, 1, \ldots\}$$

není bezkontextový.

Důkaz provedeme sporem pomocí Pumping lemmatu (věta 5.17). Předpokládejme $L \in \mathcal{L}_2$. Pak existují čísla m, n pro něž platí tvrzení Pumping lemmatu. Zvolme $i_0 > \max\{m, n\}$, slovo $\alpha_0 = a^{i_0} b^{i_0} c^{i_0}$ nechť má rozklad $uvwxy$. Neboť $|vwx| < m$, musí mít vwx jeden z tvarů $a^i, a^i b^j, b^j, b^j c^k, c^k$ pro $0 < i, j, k < m$. Slovo uwy je kratší než $uvwxy$, protože $v \neq \Lambda \vee x \neq \Lambda$. Vypuštěním podslov v, x z $uvwxy$ se zmenšil počet jednoho nebo dvou písmen z a, b, c, ale nikdy všech tří najednou. Porušila se tedy rovnost $\#a = \#b = \#c$[29], neboli $uwy \notin L$. Ukázali jsme, že slovo $\alpha_0 \in L$ nemá rozklad takový, aby $(\forall\, i \in N_0)(uv^i wx^i y \in L)$, tedy L nesplňuje Pumping lemma a není bezkontextový. FINE

Nyní si můžeme dokázat další inkluzi v Chomského hierarchii (36). Půjde nám o $\mathcal{L}_2 \subset \mathcal{L}_1$. Vezměme si nejprve $L \in \mathcal{L}_2$, tzn. L bezkontextový. Podle lemmatu 5.7 existuje jistě bezkontextová gramatika G ve standartní formě, která jej generuje. Z definice standartní formy a nezkracující gramatiky vyplývá, že každá gramatika ve standartní formě (tedy i G) je nezkracující. Podle věty 5.9 přijímají nezkracující gramatiky právě kontextové jazyky, neboli L je kontextový ($L \in \mathcal{L}_1$). Zatím víme $\mathcal{L}_2 \subseteq \mathcal{L}_1$. Na druhou stranu jazyk L z příkladu 5.5 je podle věty 5.9 kontextový, ale není bezkontextový (viz příklad 5.18), tzn. $\mathcal{L}_2 \subset \mathcal{L}_1$.

Poznámka. Pro zajímavost a bez důkazu uvádím ještě kromě právě použitého jazyka L další jazyk

$$\{a^i b^j c^i d^j; i, j = 0, 1, \ldots\},$$

který není bezkontextový. Naproti tomu velmi podobné jazyky

$$\{a^i b^i c^j d^j; i, j = 0, 1, \ldots\}$$
$$\{a^i b^j c^j d^i; i, j = 0, 1, \ldots\}$$

bezkontextové jsou.

Aby byla posloupnost inkluzí (36) dokázána úplně, zbývá $\mathcal{L}_1 \subset \mathcal{L}_0$. Každý jistě uzná, že $\mathcal{L}_1 \subseteq \mathcal{L}_0$, neboť kontextové gramatiky jsou speciálním případem obecných gramatik. Najít jazyk, který by nebyl kontextový, ovšem již tak triviální není a zatím na to ani nemáme dostatečné prostředky. Budeme se proto muset spokojit pouze s neostrou inkluzí[30].

5.2.3 Operace nad bezkontextovými gramatikami

[29] $\#x$ značí počet výskytů písmene x v nějakém slově, v našem případě α_0

[30] Leč nezoufejte! Na straně 99 je nekontextový jazyk patřící do $\mathcal{L}_0$ definován! :)

Sjednocení Jsou-li L_1, L_2 bezkontextové jazyky, pak $L = L_1 \cup L_2$ je také bezkontextový. Skutečně, nechť bezkontextové gramatiky

$$\begin{aligned} G_1 &= (V_{N1}, X, S_1, P_1), \\ G_2 &= (V_{N2}, X, S_2, P_2) \end{aligned}$$

generují jazyky L_1 a L_2. Předpokládejme, že $V_{N_1} \cap V_{N_2} = \emptyset$. Gramatika, generující L, je

$$G = (V_{N1} \cup V_{N2} \cup \{S\}, X, S, P_1 \cup P_2 \cup \{S \to S_1, S \to S_2\})$$

pro $S \notin V_{N1} \cup V_{N2}$. Zřejmě je G bezkontextová a generuje L, to je trivialita a nebudeme s tím mařit čas. Indukcí okamžitě dostáváme, že jsou-li $L_i, i \in I$, I konečná, bezkontextové jazyky, je i $L = \bigcup_{i \in I} L_i$ bezkontextový jazyk. Třída $\mathcal{L}_2$ je tedy uzavřena na konečná sjednocení.

Průniky Bezkontextové jazyky nejsou uzavřeny na průniky. Stačí vzít

$$\begin{aligned} L_1 &= \{a^i b^i c^j; i, j = 0, 1, \ldots\} \\ L_2 &= \{a^i b^j c^j; i, j = 0, 1, \ldots\} \\ L = L_1 \cap L_2 &= \{a^i b^i c^i; i = 0, 1, \ldots\} \end{aligned}$$

Jazyky L_1 a L_2 jsou zřejmě generované bezkontextovými gramatikami $G_1 = (\{S_1, S_1', C\}, X, S_1, P_1)$, $G_2 = (\{S_2, S_2', A\}, X, S_2, P_2)$, kde

$$\begin{aligned} X &= \{a, b, c\} \\ P_1 &= \{S_1 \to aS_1'bC \mid C \mid \Lambda, C \to cC \mid \Lambda, S_1' \to aS_1'b \mid \Lambda\} \\ P_2 &= \{S_2 \to AbS_2'c \mid A \mid \Lambda, A \to aA \mid \Lambda, S_2' \to bS_2'c \mid \Lambda\} \end{aligned}$$

L ale bezkontextový není, což je ukázáno v příkladu 5.18.

Doplněk Protože z *de Morganových pravidel* pro jakékoli množiny, tedy i pro bezkontextové jazyky platí[31]

$$L_1 \cap L_2 = \overline{\bar{L}_1 \cup \bar{L}_2}$$

a bezkontextové jazyky jsou uzavřeny na sjednocení, nemohou být uzavřeny na doplňky, neboť kdyby byly, musely by být uzavřeny i na průniky, což nejsou.

Konkatenace Jsou-li L_1, L_2 bezkontextové jazyky generované gramatikami $G_1 = (V_{N1}, X, S_1, P_1)$, $G_2 = (V_{N2}, X, S_2, P_2)$, pak $L = L_1 L_2$ je také bezkontextový. Opět předpokládáme $V_{N_1} \cap V_{N_2} = \emptyset$. Gramatika, generující L, je

$$G = (V_{N1} \cup V_{N2} \cup \{S\}, X, S, P_1 \cup P_2 \cup \{S \to S_1 S_2\})$$

pro $S \notin V_{N1} \cup V_{N2}$. Opět je triviální, že G je bezkontextová a $L(G) = L_1 L_2$. Bezkontextové jazyky jsou tedy uzavřené na operaci konkatenace.

[31] připomeňme si, že $\bar{L}$ značí doplněk jazyka L

Iterace (operace *) Nechť L je bezkontextový jazyk a $G = (V_N, X, S, P)$ gramatika, která ho generuje. Potom definujme $G' = (V_N \cup \{S_1\}, X, S_1, P')$, kde

$$P' = P \cup \{S_1 \to \Lambda, S_1 \to SS_1\}$$

pro $S_1 \notin V_N$. Zřejmě G' je bezkontextová a $L^* = L(G')$, bezkontextové jazyky jsou tudíž uzavřené na iteraci.

Zrcadlový obraz (operace ^{R}) Je-li $G = (V_N, X, S, P)$ bezkontextová gramatika, pak gramatika, generující $L(G)^R$ (zrcadlový obraz jazyka $L(G)$), je zřejmě $G^R = (V_N, X, S, P')$, kde

$$P' = \{X \to \alpha^R; (X \to \alpha) \in P\}.$$

Protože G^R je bezkontextová, jsou tedy bezkontextové jazyky uzavřeny na zrcadlový obraz.

6 Zásobníkové automaty

Zásobníkové automaty (dále často jen *ZA*) jsou rozšířením dříve definovaných automatů. Mají navíc k dispozici zvláštní druh *nekonečné* paměti – *zásobník*. V každém kroku musejí přečíst a smazat symbol na vrcholu zásobníku a na vrchol zásobníku zapsat nějaké slovo (může být prázdné). Symbol, vložený na zásobník jako poslední, musí být přečten a smazán jako první[1]. Druhým rozšířením oproti nám již známým automatům je možnost na nějakou dobu zastavit přijímání dalších symbolů ze vstupní pásky, tzn. zůstat libovolně dlouho na jedné pozici vstupní pásky a číst pouze ze zásobníku.

Pro potřeby této kapitoly označme $X_\Lambda = X \cup \{\Lambda\}$, kdykoli X je konečná abeceda.

Definice 6.1 *Nedeterministický zásobníkový automat je sedmice*

$$(Q, X, \Sigma, \delta, q_0, r_0, F),$$

kde

- Q *je konečná množina stavů*
- X *vstupní abeceda*
- Σ *zásobníková abeceda*
- $\delta : Q \times X_\Lambda \times \Sigma \to \mathcal{P}(Q \times \Sigma^*)$ *přechodová funkce*
- $q_0 \in Q$ *iniciální stav*
- $r_0 \in \Sigma$ *iniciální zásobníkový symbol*
- $F \subseteq Q$ *množina přijímacích stavů*

[1] Neboli aby se *ZA* dostal k n-tému symbolu pod vrcholem zásobníku, musí napřed přečíst všech vyšších $n-1$ symbolů

Stojí za povšimnutí, jak je vyřešena volba, zda posunout čtecí hlavu či nikoliv. Na rozdíl od definice dvoucestných automatů je zde funkce δ definována tak, že posun hlavy není zřejmý z oboru hodnot. Zato je v definičním oboru X_Λ místo X, neboli existují přechody funkce δ přes prázdné slovo (tzv. Λ–přechody), při kterých se hlava na vstupní pásce nepohne (čte Λ).

Také je asi moudré zdůraznit, že *ZA* v každém kroku přečte právě jedno písmeno ze zásobníku a na zásobník zapíše slovo z Σ^*.

Definice 6.2 *Nechť $\mathcal{A} = (Q, X, \Sigma, \delta, q_0, r_0, F)$ je zásobníkový automat. Potom*

- *Konfigurace zásobníkového automatu $\mathcal{A}$ je trojice (q, α, β), kde $q \in Q$, $\alpha \in X^*$, $\beta \in \Sigma^*$. $\mathcal{A}$ se nachází v konfiguraci (q, α, β), pokud je ve stavu q, na vstupní pásce zbývá přečíst slovo α a na zásobníku je uloženo β (tzn. na vrcholu zásobníku je první písmeno slova β).*

- *$\mathcal{A}$ přímo přejde z konfigurace (q, α, β) do konfigurace (q', α', β') (zapíšeme $(q, \alpha, \beta) \vdash (q', \alpha', \beta')$), jakmile platí*

$$\begin{array}{l} \alpha = a\alpha' \\ \beta = b\beta'' \\ \beta' = \gamma\beta'' \\ (q', \gamma) \in \delta(q, a, b) \end{array} \qquad \beta, \beta', \beta'', \gamma \in \Sigma^*, b \in \Sigma, \alpha, \alpha' \in X^*, a \in X_\Lambda$$

  (zdůrazňuji možnost $a = \Lambda$)

- *$\mathcal{A}$ přejde z (q_0, α_0, β_0) do (q_n, α_n, β_n) (píšeme $(q_0, \alpha_0, \beta_0) \vdash_*^{} (q_n, \alpha_n, \beta_n)$), pokud existuje posloupnost*

$$(q_0, \alpha_0, \beta_0) \vdash (q_1, \alpha_1, \beta_1) \vdash \cdots \vdash (q_n, \alpha_n, \beta_n)$$

  Pokud budeme potřebovat rozlišit, který automat přechází z jedné konfigurace do druhé, zapíšeme $(q_0, \alpha_0, \beta_0) \vdash_*^{\mathcal{A}} (q_n, \alpha_n, \beta_n)$ (přechody probíhají v automatu $\mathcal{A}$).

- *Posloupnost konfigurací nazveme výpočtem automatu na vstupním slovem α, pokud první konfigurace je (q_0, α, r_0) a poslední má tvar (q, Λ, β), kde $q \in Q, \beta \in \Sigma^*$.*

Definice 6.3 *Máme-li $\alpha \in X^*$, pak řekneme, že ZA $\mathcal{A} = (Q, X, \Sigma, \delta, q_0, r_0, F)$ přijímá α s prázdným zásobníkem, když*

$$(q_0, \alpha, r_0) \vdash_*^{} (q, \Lambda, \Lambda)$$

pro nějaké $q \in Q$. Jazyk všech slov přijímaných s prázdným zásobníkem označíme $N(\mathcal{A})$. Zásobníkový automat $\mathcal{A}$ přijímá α přijímacím stavem, pokud

$$(q_0, \alpha, r_0) \vdash_*^{} (q, \Lambda, \beta)$$

pro nějaké $q \in F$, $\beta \in \Sigma^*$. Jazyk všech slov přijímaných přijímacím stavem označíme $L(\mathcal{A})$.

Definice 6.4 *O ZA* $\mathcal{A} = (Q, X, \Sigma, \delta, q_0, r_0, F)$ *řekneme, že je deterministický, kdykoli splňuje podmínky*

- $(\forall\, a \in X_\Lambda)(\forall\, q \in Q)(\forall\, r \in \Sigma) \qquad (|\delta(q, a, r)| \leq 1)$
- $(\forall\, q \in Q)(\forall\, r \in \Sigma) \qquad [(\delta(q, \Lambda, r) \neq \emptyset) \Rightarrow (\forall\, a \in X)[\delta(q, a, r) = \emptyset]]$

Praktický význam první podmínky je podobný jako u „normálních" deterministických automatů, tzn. aby jednomu stavu byla pro jeden vstup a jeden znak na zásobníku přiřazena nejvýše jedna dvojice *(stav; slovo na zásobníku)*. Druhá podmínka nám zaručuje jednoznačnost posunů hlavy po vstupní pásce. Existuje-li pro daná q a r Λ–přechod, pak neexistuje přechod přes písmeno z X. Naopak existuje-li alespoň jeden přechod přes písmeno z X, neexistuje Λ–přechod. Tím pádem pro daná q a r hlava *buď* zůstane stát (Λ–přechod), *nebo* se pohne o políčko vpravo (čte písmeno z X).[2]

V následujících tvrzeních si shrneme některé poznatky o zásobníkových automatech.

Tvrzení 6.5 *Pro každý zásobníkový automat* $\mathcal{A} = (Q, X, \Sigma, \delta, q_0, r_0, \emptyset)$ *existuje zásobníkový automat* $\mathcal{B}$ *takový, že* $N(\mathcal{A}) = L(\mathcal{B})$.

Důkaz: Definujme *ZA* $\mathcal{B} = (Q_1, X, \Sigma_1, \delta_1, q_1, r_1, \{q_2\})$, kde

$$\begin{aligned}
\Sigma_1 &= \Sigma \cup \{r_1\} \qquad r_1 \notin \Sigma \\
Q_1 &= Q \cup \{q_1, q_2\} \qquad q_1, q_2 \notin Q \\
\delta_1(q, a, r) &= \begin{cases} \delta(q, a, r) & \forall\, q \in Q, r \in \Sigma, a \in X_\Lambda \\ \{(q_0, r_0 r_1)\} & \text{pro } q = q_1, a = \Lambda, r = r_1 \\ \{(q_2, \Lambda)\} & \text{pro } a = \Lambda, r = r_1, \forall\, q \in Q \\ \emptyset & \text{jinak} \end{cases}
\end{aligned}$$

Kouzlo této definice není nijak převratné. $\mathcal{B}$ pracuje skoro stejně jako $\mathcal{A}$. Na začátku při přechodu z q_1 nechá na dně zásobníku r_1 (druhý řádek definice δ_1). Pak probíhá výpočet jako v $\mathcal{A}$ (první řádek) a když skončí výpočet $\mathcal{A}$ s prázdným zásobníkem (tzn. na zásobníku $\mathcal{B}$ zůstane r_1), přejde do přijímacího stavu q_2, neboli přijímá vstupní slovo přijímacím stavem. Toto byla hrubě nastíněná idea definice $\mathcal{B}$. Formální důkaz rovnosti $L(\mathcal{B}) = N(\mathcal{A})$ rozdělíme, jak je zvykem, na dvě opačné inkluze.

1. Nechť $\alpha \in N(\mathcal{A})$. Pak existuje výpočet $\mathcal{A}$ nad α

   (38) $(q_0, \alpha, r_0) \overset{\mathcal{A}}{\underset{*}{\vdash}} (q, \Lambda, \Lambda)$

   pro nějaké $q \in Q$. Pak ale

   $$(q_1, \alpha, r_1) \overset{\mathcal{B}}{\vdash} (q_0, \alpha, r_0 r_1) \overset{\mathcal{B}}{\underset{*}{\vdash}} (q, \Lambda, r_1) \overset{\mathcal{B}}{\vdash} (q_2, \Lambda, \Lambda)$$

   je nutně výpočet, kterým $\mathcal{B}$ přijímá α přijímacím stavem[3], tudíž $\alpha \in L(\mathcal{B})$ a $N(\mathcal{A}) \subseteq L(\mathcal{B})$.

[2] poslední souvětí je v ostře vylučovacím poměru (XOR)

[3] První a poslední přímý přechod vyplývá rovnou z definice δ_1, prostřední přechod z (38), neboť r_1 zůstává stále na dně zásobníku $\mathcal{B}$.

2. Máme-li $\alpha \in L(\mathcal{B})$, pak

$$(q_1, \alpha, r_1) \overset{\mathcal{B}}{\vdash} (q_0, \alpha, r_0 r_1) \overset{\mathcal{B}}{\vdash}_{*} (q, \alpha', \beta') \overset{\mathcal{B}}{\vdash} (q_2, \Lambda, \beta)$$

je výpočet $\mathcal{B}$ nad α. Z definice δ_1 (třetí řádek) vyplývá, že před přechodem do q_2 muselo být na zásobníku právě r_1 — neboli $\beta' = r_1$ a $\alpha' = \Lambda$ — a po přechodu do q_2 je na zásobníku vždy Λ — neboli $\beta = \Lambda$. Přitom celý druhý přechod probíhal z definice δ_1 (první řádek) podle automatu $\mathcal{A}$, až na r_1 na dně zásobníku, ale na ten se v druhém přechodu stejně nesáhlo. Tudíž

$$(q_0, \alpha, r_0) \overset{\mathcal{A}}{\vdash}_{*} (q, \Lambda, \Lambda)$$

je výpočet nad α v $\mathcal{A}$ s prázdným zásobníkem, neboli $\alpha \in N(\mathcal{A})$ a spolu inkluzí z první části dostáváme $L(\mathcal{B}) = N(\mathcal{A})$. CBD

Tvrzení 6.6 *Pro každý zásobníkový automat $\mathcal{A} = (Q, X, \Sigma, \delta, q_0, r_0, F)$ existuje zásobníkový automat $\mathcal{B}$ takový, že $L(\mathcal{A}) = N(\mathcal{B})$.*

Důkaz: Sestrojíme automat $\mathcal{B}$. Podobně jako v důkazu předchozího tvrzení $\mathcal{B}$ bude nepatrně upravený automat $\mathcal{A}$. Potřebujeme pouze, pokud $\mathcal{A}$ přijme nějaké slovo přijímacím stavem, aby $\mathcal{B}$ vyprázdnil zásobník. To nám zaručí Λ–přechody ve stavu q_1. Automat $\mathcal{B}$ nepotřebuje žádné přijímací stavy (neboť přijímá prázdným zásobníkem), nuže definujme

$$\mathcal{B} = (Q \cup \{q_1, q_2\}, X, \Sigma \cup \{r_1\}, \delta_1, q_2, r_1, \emptyset)$$

pro $q_1, q_2 \notin Q$, $r_1 \notin \Sigma$, kde

$$\begin{aligned}
\delta_1(q,a,r) &= \begin{cases} \delta(q,a,r) \quad \forall\, q \in Q \setminus F, a \in X_\Lambda, r \in \Sigma \\ \qquad \text{nebo } q \in F, a \in X, r \in \Sigma \\ \delta(q,a,r) \cup \{(q_1, \Lambda)\} \text{ pro } q \in F, a = \Lambda, r \in \Sigma \end{cases} \\
\delta_1(q, \Lambda, r_1) &= \{(q_1, r_1)\} \text{ pro } q \in F \\
\delta_1(q_1, \Lambda, r) &= \{(q_1, \Lambda)\} \text{ pro } r \in \Sigma \cup \{r_1\} \\
\delta_1(q_2, \Lambda, r_1) &= \{(q_0, r_0 r_1)\} \\
\delta_1(q,a,r) &= \emptyset \text{ v ostatních případech}
\end{aligned}$$

Funkce δ_1 je definována tak, aby pro stavy z $Q \setminus F$ kopírovala výpočet v $\mathcal{A}$, pro stavy z F měla možnost provést Λ–přechod do q_1 a pokud se někdy $\mathcal{B}$ dostane do q_1, aby v něm již zůstal a smazal zásobník. Přidané stavy a zásobníkový symbol se uplatní pouze na začátku a v poslední části výpočtu $\mathcal{B}$, výpočet okopírovaný z $\mathcal{A}$ neovlivní. Dokážeme $L(\mathcal{A}) = N(\mathcal{B})$.

1. Nechť $\alpha \in L(\mathcal{A})$. Potom pro nějaké $q \in F$, $\beta \in \Sigma^*$ je $(q_0, \alpha, r_0) \overset{\mathcal{A}}{\vdash}_{*} (q, \Lambda, \beta)$ přijímací výpočet v $\mathcal{A}$, tedy

$$(39) \quad (q_2, \alpha, r_1) \overset{\mathcal{B}}{\vdash} (q_0, \alpha, r_0 r_1) \overset{\mathcal{B}}{\vdash}_{*} (q, \Lambda, \beta r_1) \overset{\mathcal{B}}{\vdash} (q_1, \Lambda, \beta_1 r_1) \overset{\mathcal{B}}{\vdash}_{*} (q_1, \Lambda, \Lambda)$$

pro $\beta, \beta_1 \in \Sigma^*$ je přijímací výpočet v $\mathcal{B}$, neboli $\alpha \in N(\mathcal{B})$.

2. Mějme $\alpha \in N(\mathcal{B})$. Pak (39) je přijímací výpočet v $\mathcal{B}$, přičemž prostřední přechody musely nutně probíhat podle prvních dvou řádků definice δ_1, kde se na r_1 nesahá, a stav q_1 se v těchto přechodech nevyskytl, tedy zde $\delta_1 = \delta$ a tudíž $(q_0, \alpha, r_0) \overset{\mathcal{A}}{\underset{*}{\vdash}} (q, \Lambda, \beta)$ je přijímací výpočet v $\mathcal{A}$[4], z čehož $\alpha \in L(\mathcal{A})$ a s první částí důkazu $L(\mathcal{A}) = N(\mathcal{B})$. CBD

Tvrzení 6.7 *Zásobníkové automaty přijímají s prázdným zásobníkem právě bezkontextové jazyky.*

Důkaz. Jinými slovy, ke každému zásobníkovému automatu $\mathcal{A}$ existuje bezkontextová gramatika G taková, že $N(\mathcal{A}) = L(G)$ a naopak. Důkaz bude spočívat v tom, že k libovolnému *ZA* $\mathcal{A}$ sestrojíme bezkontextovou gramatiku G a k libovolné gramatice $G \in \mathcal{L}_2$ sestrojíme *ZA* $\mathcal{A}$.

1. Buď nejprve $\mathcal{A} = (Q, X, \Sigma, \delta, q_0, r_0, \emptyset)$. Gramatiku $G = (V_N, X, S, P)$ definujeme tak, aby simulovala výpočet v $\mathcal{A}$.

$$
\begin{aligned}
V_N &= \{S\} \cup \{(q, r, q'); q, q' \in Q, r \in \Sigma\} \\
P &\ni \begin{cases}
S \to (q_0, r_0, q) \quad \forall\, q \in Q \\
(q, r, q') \to a \text{ pokud } (q', \Lambda) \in \delta(q, a, r) \\
(q, r, q') \to a(q_1, r_1, q_2)(q_2, r_2, q_3) \dots (q_k, r_k, q') \\
\qquad \text{pro } k > 0 \text{ a } \forall\, q_1, \dots, q_k \in Q \\
\qquad \text{pokud } (q_1, r_1 \dots r_k) \in \delta(q, a, r)
\end{cases}
\end{aligned}
$$

pro všechna $a \in X_\Lambda,\ q, q' \in Q,\ r, r_1, \dots, r_k \in \Sigma$, přičemž pravidla podle třetího řádku definice P se do P přidají jen tehdy, když existují slova $\alpha_1, \dots, \alpha_k \in X^*$ taková, že

$$
\begin{aligned}
(q_i, \alpha_i \dots \alpha_k \beta, r_i \dots r_k \sigma) &\overset{\mathcal{A}}{\underset{*}{\vdash}} (q_{i+1}, \alpha_{i+1} \dots \alpha_k \beta, r_{i+1} \dots r_k \sigma) \\
(q_k, \alpha_k \beta, r_k \sigma) &\overset{\mathcal{A}}{\underset{*}{\vdash}} (q', \beta, \sigma)
\end{aligned}
$$

jsou pro $i = 1, \dots, (k-1)$ výpočty $\mathcal{A}$ pro libovolná $\beta \in X^*$, $\sigma \in \Sigma^*$. Neterminál (q, r, q') explicitně řečeno znamená, že $\mathcal{A}$ je ve stavu q, na vrcholu zásobníku má r a výpočet odstraňující r z vrcholu zásobníku[5] skončí v q'[6]. G je nesporně bezkontextová. Abychom mohli ukázat $L(G) = N(\mathcal{A})$, vyslovíme nejprve pomocné lemma.

Lemma 6.8 *Pro výše definovanou gramatiku G a automat $\mathcal{A}$ platí pro libovolné slovo $\alpha \in X^*$, stavy q, q' a $r \in \Sigma$*

$$
(q, r, q') \overset{G}{\underset{*}{\Longrightarrow}} \alpha \iff (q, \alpha, r) \overset{\mathcal{A}}{\underset{*}{\vdash}} (q', \Lambda, \Lambda).
$$

[4] q musel být nutně z F, neboť to je jediná možnost, jak se dostat do stavu q_1 v následujícím přechodu v (39)

[5] máme na mysli výpočet, který odstraní r z vrcholu zásobníku, ale nic na něj nezapíše, tzn. bylo-li před začátkem výpočtu na zásobníku slovo $r\beta$, po skončení výpočtu bude na zásobníku β

[6] Přechod z q do q' probíhá přes stavy $q_1, \dots, q_k$ (viz třetí řádek definice P), případně rovnou, pokud se nezapisuje na zásobník.

Důkaz pomocného lemmatu rozdělíme na dvě implikace a každou dokážeme indukcí. Zleva doprava podle délky derivace, zprava doleva podle délky výpočtu. První krok je pro obě implikace společný, neboť z definice P platí

$$(q, r, q') \overset{G}{\Longrightarrow} \alpha \Leftrightarrow \alpha \in X_\Lambda \wedge (q', \Lambda) \in \delta(q, \alpha, r) \Leftrightarrow (q, \alpha, r) \overset{\mathcal{A}}{\vdash} (q', \Lambda, \Lambda)$$

Nyní druhý krok důkazu implikace zleva doprava. Mějme

$$(q, r, q') \Longrightarrow a(q_1, r_1, q_2) \dots (q_k, r_k, q') \underset{*}{\Longrightarrow} \alpha.$$

Potom $\alpha = a\alpha'$ pro nějaké $a \in X_\Lambda, \alpha' \in X^*$ a $(q_1, r_1 \dots r_k) \in \delta(q, a, r)$. Protože gramatika G je bezkontextová, pro každé $i = 1, 2, \dots, k$ platí $(q_i, r_i, q_{i+1}) \underset{*}{\Longrightarrow} \alpha_i$[7] a $\alpha = a\alpha_1\alpha_2 \dots \alpha_k$. Derivace α_i jsou však nejméně o jeden krok kratší než derivace α, můžeme tedy použít indukční předpoklad

$$(40) \quad (q_i, r_i, q_{i+1}) \underset{*}{\Longrightarrow} \alpha_i \Leftrightarrow (q_i, \alpha_i, r_i) \underset{*}{\vdash} (q_{i+1}, \Lambda, \Lambda)$$

pro všechna $i = 1, \dots, k$, z čehož

$$\begin{aligned} (q, \alpha, r) \vdash (q_1, \alpha_1 \dots \alpha_k, r_1 \dots r_k) &\underset{*}{\vdash} (q_2, \alpha_2 \dots \alpha_k, r_2 \dots r_k) \underset{*}{\vdash} \cdots \\ \cdots &\underset{*}{\vdash} (q_k, \alpha_k, r_k) \underset{*}{\vdash} (q', \Lambda, \Lambda). \end{aligned}$$

Tím je dokázána první implikace.

Druhý krok druhé implikace (zprava doleva) bude velmi podobný. Víme, že

$$(q, \alpha, r) \vdash (q_1, \alpha', r_1 \dots r_k) \underset{*}{\vdash} (q', \Lambda, \Lambda)$$

pro nějaké $a\alpha' = \alpha, a \in X_\Lambda, \alpha' \in X^*, (q_1, r_1 \dots r_k) \in \delta(q, a, r)$. Potom ale jistě existují slova $\beta_1, \dots, \beta_k \in X^*$[8] a stavy $q_2, \dots, q_k$ takové, že $\beta_1 = \alpha'$ a

$$\begin{aligned} (q, \alpha, r) &\vdash (q_1, \beta_1, r_1 \dots r_k) \underset{*}{\vdash} (q_2, \beta_2, r_2 \dots r_k) \underset{*}{\vdash} \cdots \\ \cdots &\underset{*}{\vdash} (q_k, \beta_k, r_k) \underset{*}{\vdash} (q', \Lambda, \Lambda). \end{aligned}$$

Z toho vyplývá, že G obsahuje pravidlo

$$(q, r, q') \to a(q_1, r_1, q_2) \dots (q_k, r_k, q').$$

Definujme α_i pro $i = 1, \dots, (k-1)$ tak, že $\beta_i = \alpha_i\beta_{i+1}$ a $\alpha_k = \beta_k$. Potom zřejmě $(q_i, \alpha_i, r_i) \underset{*}{\vdash} (q_{i+1}, \Lambda, \Lambda)$ pro $i \leq k$. Výpočty nad α_i jsou zřejmě nejméně o jeden krok kratší, než výpočty nad α, z indukčního předpokladu tedy víme (40), z čehož

$$(q, r, q') \Longrightarrow a(q_1, r_1, q_2) \dots (q_k, r_k, q') \underset{*}{\Longrightarrow} a\alpha_1\alpha_2 \dots \alpha_k = a\alpha' = \alpha.$$

[7] s tím, že $q' = q_{k+1}$

[8] β_{i+1} je vždy suffix β_i

Tím je pomocné lemma dokázáno. CBD

Pro každé $\alpha \in X^*$ z definice P a lemmatu 6.8 plyne

$$\begin{aligned} \alpha \in L(G) &\Leftrightarrow (S \overset{G}{\Longrightarrow} (q_0, r_0, q) \overset{G}{\underset{*}{\Longrightarrow}} \alpha) \Leftrightarrow \left((q_0, \alpha, r_0) \overset{\mathcal{A}}{\underset{*}{\vdash}} (q, \Lambda, \Lambda)\right) \Leftrightarrow \\ &\Leftrightarrow \alpha \in N(\mathcal{A}), \end{aligned}$$

pro nějaké $q \in Q$, neboli $L(G) = N(\mathcal{A})$.

2. Mějme bezkontextovou gramatiku $G = (V_N, V_T, S, P)$. Definujeme jednostavový nedeterministický zásobníkový automat

$$\mathcal{A} = (\{q_0\}, V_T, V_T \cup V_N, \delta, q_0, S, \emptyset),$$

kde

$$\begin{aligned} (q_0, X_1 \dots X_k) &\in \delta(q_0, \Lambda, X) \text{ pokud } (X \to X_1 \dots X_k) \in P \\ &\quad \text{pro } X \in V_N, X_1, \dots, X_k \in V_T \cup V_N, k \geq 0 \\ (q_0, \Lambda) &\in \delta(q_0, a, a) \text{ pro } a \in V_T \end{aligned}$$

Automat $\mathcal{A}$ generuje na zásobníku postupně derivaci vstupního slova (označme ho α) v gramatice G, přičemž čtecí hlava na vstupní pásce se pohne pouze v případě, že na vrcholu zásobníku je terminál, který se shoduje s písmenem na pozici čtecí hlavy. Neterminál na vrcholu zásobníku přepíše podle nějakého pravidla z P. Pokud je na vrcholu zásobníku terminál, který se neshoduje s písmenem na vstupní pásce, je zřejmé, že generované slovo již nemůže být α, neboli výpočet skončí neúspěchem[9]. Stejně jako v první části důkazu, i teď vyslovíme pomocné lemma, které z rovnosti $L(G) = N(\mathcal{A})$, o kterou nám jde, udělá trivialitu.

Lemma 6.9 $(q_0, \alpha, \beta) \overset{\mathcal{A}}{\underset{*}{\vdash}} (q_0, \Lambda, \Lambda) \iff \beta \overset{G}{\underset{*}{\Longrightarrow}} \alpha$

Obě implikace dokážeme indukcí, zleva doprava dle délky výpočtu, zprava doleva dle délky odvození.

Zleva doprava. $(q_0, \alpha, \beta) \vdash (q_0, \Lambda, \Lambda)$ nastane podle definice δ právě tehdy, když buď $\alpha = \beta \in V_T$ (potom zřejmě $\beta \overset{}{\underset{*}{\Longrightarrow}} \alpha$), nebo $\alpha = \Lambda, \beta \in V_N$ a $(\beta \to \alpha) \in P$, čili také $\beta \overset{}{\underset{*}{\Longrightarrow}} \alpha$, čímž máme první krok. Ve výpočtu $\overbrace{(q_0, \alpha, \beta) \overset{}{\underset{*}{\vdash}} (q_0, \Lambda, \Lambda)}^{V}$ delším než jeden krok mohou nastat z definice δ dva případy:

- $(q_0, \alpha, \beta) \vdash \overbrace{(q_0, \alpha, \gamma\beta') \overset{}{\underset{*}{\vdash}} (q_0, \Lambda, \Lambda)}^{V1}$, kde $\beta = X\beta', (X \to \gamma) \in P$. Výpočet $V1$ je ovšem kratší než V, neboli z indukčního předpokladu $\beta = X\beta' \Longrightarrow \gamma\beta' \overset{}{\underset{*}{\Longrightarrow}} \alpha$.

[9] $\mathcal{A}$ je však nedeterministický, proto není třeba zoufat. Může uspět jiný výpočet.

- $(q_0,\alpha,\beta)\vdash\overbrace{(q_0,\alpha',\beta')\underset{*}{\vdash}(q_0,\Lambda,\Lambda)}^{V2}$, kde $\beta=a\beta', \alpha=a\alpha', a\in V_T$. Výpočet $V2$ je ovšem kratší než V, neboli z indukčního předpokladu $\beta=a\beta'\underset{*}{\Longrightarrow}a\alpha'=\alpha$.

Zprava doleva. Je-li $\beta=\alpha\in V_T^*$, potom opakovaným použitím druhého řádku definice δ dostaneme $(q_0,\alpha,\beta)\underset{*}{\vdash}(q_0,\Lambda,\Lambda)$. Je-li $\beta\underset{*}{\Longrightarrow}\alpha$ odvození délky alespoň 1, pak rozdělíme β na $\beta=\beta' X\beta''$[10]. Zřejmě platí $\alpha=\beta'\alpha'$, můžeme tedy přejít k derivaci $\overbrace{X\beta''\underset{*}{\Longrightarrow}\alpha'}^{O}$ místo $\beta\underset{*}{\Longrightarrow}\alpha$. Protože G je bezkontextová, jistě pro nějaké pravidlo $(X\to\gamma)\in P$ platí

$$X\beta''\Longrightarrow\overbrace{\gamma\beta''\underset{*}{\Longrightarrow}\alpha'}^{O1}.$$

Derivace $O1$ je kratší než O, použijeme tudíž indukční předpoklad, který nám říká, že $(q_0,\alpha',\gamma\beta'')\underset{*}{\vdash}(q_0,\Lambda,\Lambda)$, a dostáváme

$$(q_0,\alpha,\beta)=(q_0,\beta'\alpha',\beta' X\beta'')\underset{*}{\vdash}(q_0,\alpha',X\beta'')\vdash(q_0,\alpha',\gamma\beta'')\underset{*}{\vdash}(q_0,\Lambda,\Lambda)$$

(Na odstranění β' jsme použili druhý řádek definice δ.) Tím je lemma 6.9 dokázáno. CBD

Pro libovolné $\alpha\in V_T^*$ z lemmatu 6.9 vyplývá

$$\alpha\in L(G)\Leftrightarrow S\underset{*}{\overset{G}{\Longrightarrow}}\alpha\Leftrightarrow\left((q_0,\alpha,S)\underset{*}{\overset{\mathcal{A}}{\vdash}}(q_0,\Lambda,\Lambda)\right)\Leftrightarrow\alpha\in N(\mathcal{A}),$$

neboli $L(G)=N(\mathcal{A})$.

K libovolné bezkontextové gramatice jsme sestrojili zásobníkový automat přijímající s prázdným zásobníkem stejný jazyk a naopak, tvrzení 6.7 je tedy dokázáno. CBD

Důsledek 6.10 *Následující výroky o jazyku L jsou ekvivalentní:*

1. *L je bezkontextový jazyk.*
2. *Existuje zásobníkový automat $\mathcal{A}$ takový, že $L=N(\mathcal{A})$*
3. *Existuje zásobníkový automat $\mathcal{B}$ takový, že $L=L(\mathcal{B})$*

Důkaz: vyplývá přímo z tvrzení 6.5,6.6,6.7. CBD

Definice 6.11 *Jazyk L nazýváme bezkontextovým deterministickým jazykem, když existuje deterministický zásobníkový automat, který ho přijímá.*

[10] kde $\beta'\in V_T^*, X\in V_N, \beta''\in(V_T\cup V_N)^*$

Je zřejmé, že bezkontextové deterministické jazyky jsou podmnožinou bezkontextových jazyků.

Poznámka. Máme-li zásobníkový automat $\mathcal{A}$, výpočet nad vstupním slovem se zastaví, kdykoliv se vyprázdní zásobník[11]. Je-li $\alpha \in N(\mathcal{A})$, pak musí existovat výpočet $\mathcal{A}$ nad α, který skončí s prázdným zásobníkem po přečtení celého α. Během tohoto výpočtu musí být na zásobníku v každém kroku alespoň jeden znak. Pokud $\mathcal{A}$ je deterministický, pak je výpočet nad α právě jeden a zásobník se vyprázdní až po přečtení α. To ale znamená, že žádný vlastní prefix není v $N(\mathcal{A})$! Jazykům, které neobsahují žádný vlastní prefix žádného svého slova, říkáme bezprefixové. Z předchozího vyplývá, že bezprefixové deterministické bezkontextové jazyky jsou přijímány právě deterministickými ZA s prázdným zásobníkem.

Tvrzení 6.12 *Doplněk bezkontextového deterministického jazyka L je bezkontextový deterministický jazyk.*

Každého asi napadne, že stačí v automatu $\mathcal{A}$ ($L(\mathcal{A}) = L$) zaměnit F za $Q \setminus F$ a bude přijímat doplněk. Není to bohužel tak jednoduché. Jedním z důvodů je například fakt, že výpočet $\mathcal{A}$ se nad nějakým slovem α může dostat do cyklu Λ–kroků a nikdy neskončí. Takové α by jistě nepatřilo do L ani do $\bar{L}$, protože výpočet neskončil v žádném za stavů z F ani $Q \setminus F$. Spokojme se tedy s tím, že tvrzení platí a že víme, proč není jednoduché ho dokázat (uvedený důvod není jediný, který brání prosté záměně F za $Q \setminus F$).

Tvrzení 6.13 *Bezkontextové deterministické jazyky nejsou uzavřené na průnik ani sjednocení.*

Důkaz: Pro průnik stačí vzít nepatrně pozměněné jazyky L_1, L_2 z kapitoly 5.2.3, na kterých jsme ukazovali, že bezkontextové jazyky nejsou uzavřené na průnik.

$$\begin{aligned} L_1 &= \{a^i b^i c^j; i,j = 1,\ldots\} \\ L_2 &= \{a^i b^j c^j; i,j = 1,\ldots\} \\ L = L_1 \cap L_2 &= \{a^i b^i c^i; i = 1,\ldots\} \end{aligned}$$

Nechávám na čtenáři, aby přijal zřejmý fakt $L_1 = L(\mathcal{A}_1)$ a $L_2 = N(\mathcal{A}_2)$, kde pro $X = \{a,b,c\}$, $\Sigma = \{a,b,c,r_0,r_1\}$ je

$$\begin{array}{ll} \mathcal{A}_1 = (\{f_1,q_1\}, X, \Sigma, \delta_1, q_1, r_0, \{f_1\}) & \mathcal{A}_2 = (\{q_2\}, X, \Sigma, \delta_2, q_2, r_0, \emptyset\}) \\ \delta_1(q_1,a,r_0) = (q_1, ar_1) & \delta_2(q_2,a,r_0) = (q_2,r_1) \\ \delta_1(q_1,a,a) = (q_1,aa) & \delta_2(q_2,a,r_1) = (q_2,r_1) \\ \delta_1(q_1,b,a) = (q_1,\Lambda) & \delta_2(q_2,b,r_1) = (q_2,b) \\ \delta_1(q_1,c,r_1) = (f_1,r_1) & \delta_2(q_2,b,b) = (q_2,bb) \\ \delta_1(f_1,c,r_1) = (f_1,r_1) & \delta_2(q_2,c,b) = (q_2,\Lambda) \end{array}$$

Protože $L_1 \cap L_2 = L$ a jak již víme z příkladu 5.18, L není bezkontextový, tím méně bezkontextový deterministický, nejsou bezkontextové deterministické jazyky uzavřeny na průnik.[12]

[11] neboť ZA mají z definice povinnost číst ze zásobníku

[12] Komu by se nezdálo korektní, že zde bereme L bez prázdného slova, může být klidný, neboť důkaz z příkladu 5.18 na prázdném slově nestojí, jak je možno se okamžitě přesvědčit.

Podobně jako v kapitole 5.2.3 z de Morganových pravidel dostáváme, že bezkontextové deterministické jazyky nejsou uzavřeny na sjednocení, protože nejsou uzavřeny na průniky. CBD

Tvrzení 6.14 *Bezkontextové jazyky jsou uzavřeny na průniky s regulárními jazyky a na levé i pravé kvocienty podle regulárních jazyků.*

Důkaz: Mějme bezkontextový jazyk L přijímaný nedeterministickým zásobníkovým automatem $\mathcal{A} = (Q, X, \Sigma, \delta, q_0, r_0, F)$ koncovým stavem a regulární jazyk R přijímaný N-akceptorem $\mathcal{A}_1 = (Q_1, X, \delta_1, q_1, F_1)$.

Nejprve dokážeme, že $L \cap R$ je bezkontextový jazyk. Definujeme zásobníkový automat $\mathcal{A}_2 = (Q \times Q_1, X, \Sigma, \delta_2, (q_0, q_1), r_0, F \times F_1)$, který paralelně simuluje výpočty $\mathcal{A}$ a $\mathcal{A}_1$.

$$((q', p'), \beta) \in \delta_2((q, p), a, r), \text{ právě když}$$
$$\Big((p' \in \delta_1(p, a) \wedge a \neq \Lambda) \vee (p' = p \wedge a = \Lambda)\Big) \wedge \Big((q', \beta) \in \delta(q, a, r)\Big)$$

pro všechna $a \in X_\Lambda, r \in \Sigma, \beta \in \Sigma^*, q, q' \in Q, p, p' \in Q_1$. $\mathcal{A}_2$ zřejmě v první složce svých stavů simuluje výpočet $\mathcal{A}$. Výpočet v druhé složce probíhá stavy z Q_1 podle funkce δ_1 a v případě, že $\mathcal{A}_2$ čte Λ, zůstává stát. Výpočet $\mathcal{A}_2$ je přijímací, pokud simulace v obou složkách skončí v přijímacích stavech. Odtud $L \cap R = L(\mathcal{A}_2)$.[13]

Dále dokážeme, že $R \backslash L$ je bezkontextový jazyk. Idea bude podobná jako pro průnik, jen neznámou část slova z L musíme uhodnout, ale od toho máme nedeterminismus. Definujme *ZA*

$$\mathcal{A}_3 = (Q \times Q_1 \cup Q, X, \delta_3, (q_0, q_1), r_0, F),$$

kde δ_3 je dána předpisem

- $\delta_3((q, p), \Lambda, r) = A \cup \{((q', p), \beta); (q', \beta) \in \delta(q, \Lambda, r)\} \cup$
  $\cup \{((q', p'), \beta); (\exists\, y \in X)[(q', \beta) \in \delta(q, y, r) \wedge p' \in \delta_1(p, y)]\}$,

  kde

  $$A = \begin{cases} \{(q, r)\} & \text{když } p \in F_1 \\ \emptyset & \text{když } p \notin F_1 \end{cases}$$

  pro každé $q \in Q, p \in Q_1, r \in \Sigma$

- $\delta_3(q, x, r) = \delta(q, x, r)$ pro všechna $q \in Q, x \in X_\Lambda, r \in \Sigma$

Zásobníkový automat $\mathcal{A}_3$ nejprve hádá slovo $v \in R$ a zároveň paralelně simuluje výpočty $\mathcal{A}$ a $\mathcal{A}_1$ nad v (tím ověřuje, že $v \in R$). Hlava na vstupní pásce přitom zůstává na prvním políčku. Kdykoliv se simulace $\mathcal{A}_1$ dostane do koncového stavu (tzn. $v \in R$), může $\mathcal{A}_3$ přejít do „normálních“ stavů automatu $\mathcal{A}$[14] a

[13] Místo R není možné dát jednoduše bezkontextový jazyk, protože pak by $\mathcal{A}_2$ musel zapisovat na zásobník slova z $\Sigma \times \Sigma_1$ a nemáme zaručeno, že automaty $\mathcal{A}$ a $\mathcal{A}_1$ zapisují na zásobník slova stejné délky. Kdyby například $\mathcal{A}$ zapsal na zásobník v jednom kroku delší slovo než $\mathcal{A}_1$, kde by byl vrchol zásobníku $\mathcal{A}_2$?

[14] myslíme stavy z Q

pokračovat v simulaci výpočtu $\mathcal{A}$ nad vstupním slovem. Z toho vidíme, že $\mathcal{A}_3$ přijímá vstupní slovo u právě tehdy, když existuje slovo $v \in R$ takové, že $vu \in L$, neboli $L(\mathcal{A}_3) = R \backslash L$.

Podobně dokážeme, že L/R je bezkontextový jazyk. Definujme ZA

$$\mathcal{A}_4 = (Q \times Q_1 \cup Q, X, \delta_4, (q_0, q_1), r_0, F \times F_1),$$

kde δ_3 je dána předpisem

- $\delta_4(q, x, r) = \delta(q, x, r) \cup A$, kde

$$A = \begin{cases} \{(q, q_1), r\} & \text{když } x = \Lambda \\ \emptyset & \text{když } x \neq \Lambda \end{cases}$$

  pro každé $q \in Q, x \in X_\Lambda, r \in \Sigma$

- $\delta_4((q,p), \Lambda, r) = \{((q', p), \beta); (q', \beta) \in \delta(q, \Lambda, r)\} \cup$
  $\cup \{((q', p'), \beta); (\exists\, y \in X)[(q', \beta) \in \delta(q, y, r) \wedge p' \in \delta_1(p, y)]\}$

  pro každé $q \in Q, p \in Q_1, r \in \Sigma$

Automat $\mathcal{A}_4$ přijme vstupní slovo u jen tehdy, pokud přečte celou vstupní pásku, což se mu pro neprázdné vstupní slovo může podařit pouze v prvním bodu definice δ_4. Další nutnou podmínkou pro to, aby výpočet $\mathcal{A}_4$ byl přijímací, je nalezení slova $v \in R$ takového, že $uv \in L$. To je vynuceno druhým bodem definice δ_4. Obě nutné podmínky nám zaručují $L(\mathcal{A}_4) = L/R$. CBD

Poznámka. Je-li L deterministický bezkontextový jazyk, pak konstrukce první části předchozího důkazu (ovšem pro deterministické automaty) ukazuje, že $L \cap R$ je deterministický bezkontextový jazyk. O kvocientech to však tvrdit nemůžeme, neboť nedeterminismus byl pro nás v druhé a třetí části důkazu velmi podstatný.

Ponořme se nyní nepatrně do vektorových prostorů, kde překvapivě zjistíme, že každý bezkontextový jazyk nad jednoprvkovou množinou je regulární.

Definice 6.15 *Množina A je semilineární ve vektorovém prostoru V, když existují $a_1, \ldots, a_k \in V$ a $V_1, \ldots, V_k \subseteq V$ vektorové podprostory V takové, že $\vec{\alpha} \in A$ právě když každá souřadnice vektoru $\vec{\alpha}$ je nezáporná a $\vec{\alpha} \in \bigcup_{i=1}^{k} (a_i + V_i)$.*

Je-li $A = \{a_1, \ldots, a_n\}$ a $\alpha \in A^*$, označme $p_{a_i}(\alpha)$ počet výskytů prvku a_i v α a $v(\alpha) = (p_{a_1}(\alpha), \ldots, p_{a_2}(\alpha))$ vektor výskytů jednotlivých písmen z A v α. Potom platí následující věta, kterou uvedeme bez důkazu.

Věta 6.16 *(Parikh) Pro každý bezkontextový jazyk L je $V(L) = \{v(\alpha); \alpha \in L\}$ semilineární množina ve vektorovém prostoru nad celými čísly a existuje regulární jazyk R takový, že $V(L) = V(R)$.*

Že $V(L)$ je semilineární, nám teď zrovna k ničemu není, ale pokud je L jazyk nad jednoprvkovou abecedou $\{a\}$, tak podle druhé části věty okamžitě vidíme, že existuje R regulární jazyk, který má stejné počty výskytů a jako jazyk L, neboli $L = R$. Z toho

Důsledek 6.17 *Každý bezkontextový jazyk nad jednoprvkovou abecedou je regulární.*

Ukončeme nyní tuto ostudnou kapitolu a vrhněme se na vyvrcholení přednášky, na *Turingovy stroje* (čti [tjúringovy], resp. [turingovy]).

7 Turingovy stroje

Definice 7.1 *Deterministický jednopáskový Turingův stroj je sedmice*

$$(Q, X, Y, \delta, q_0, B, F),$$

kde

- Q *je konečná množina stavů*
- X *vstupní abeceda*
- $Y \supset X$ *pracovní abeceda*
- $\delta : Q \times Y \to Q \times Y \times \{-1, 0, 1\}$ *přechodová funkce*
- q_0 *iniciální stav*
- $B \in Y, B \notin X$ *prázdný symbol (Blank)*
- $F \subseteq Q$ *přijímací stavy*

Turingův stroj může zapisovat na vstupní pásku a libovolně se po ní pohybovat.

Místo *Turingův(ovy) stroj(e)* kvůli zkrácení zápisu občas použijeme zkratky *T.s.*.

Přechodovou funkci δ v deterministickém Turingově stroji definujeme často jako $\delta : (Q \setminus F) \times Y \to Q \times Y \times \{-1, 0, 1\}$. To proto, aby *T.s.* z přijímacích stavů již nepokračoval ve výpočtu.

Definice 7.2 *Konfigurace T.s.* $T = (Q, X, Y, \delta, q_0, B, F)$ *je trojice* (α, q, β), *kde* $\alpha, \beta \in Y^*$, $q \in Q$. *Řekneme, že T se nachází v konfiguraci* (α, q, β), *právě když*

- *na i-tém políčku vstupní pásky je i-té písmeno slova* $\alpha\beta$, *pokud* $i \leq |\alpha\beta|$, *nebo B, pokud* $i > |\alpha\beta|$.
- T *je ve stavu* q *a hlava na vstupní pásce čte pozici* $|\alpha|$.

Možná někoho napadne, že konfigurace, které se liší pouze počtem B na konci β, jsou ekvivalentní. Proto vězte, že řekneme-li konfigurace, myslíme tím zástupce celé jedné třídy ekvivalence (většinou není podstatné, kterou konkrétní konfiguraci máme na mysli).

Definice 7.3 *T.s. přímo (v jednom kroku) přejde z konfigurace* (α, q, β) *do konfigurace* (γ, r, σ) *(píšeme* $(\alpha, q, \beta) \vdash (\gamma, r, \sigma)$*), pokud nastane právě jedna z následujících podmínek:*

1. $\alpha = \gamma a$, $c\beta = \sigma$, $\delta(q,a) = (r,c,-1)$ *pro nějaké* $a,c \in Y$
2. $\alpha = \alpha' a$, $\gamma = \alpha' c$, $\beta = \sigma$, $\delta(q,a) = (r,c,0)$ *pro* $a,c \in Y$
3. $\alpha = \alpha' a$, $\gamma = \alpha' cd$, $\beta = d\sigma$, $\delta(q,a) = (r,c,+1)$ *pro* $a,c,d \in Y$

T.s. přejde z konfigurace (α,q,β) *do* (γ,r,σ) *(píšeme* $(\alpha,q,\beta) \underset{*}{\vdash} (\gamma,r,\sigma)$*), pokud existuje posloupnost přímých přechodů začínající v* (α,q,β) *a končící v* (γ,r,σ).

Budeme-li potřebovat rozlišit, v kterém *T.s.* přechody probíhají, uvedeme podobně jako u zásobníkových automatů nad operátor přechodu jméno příslušného Turingova stroje.

Definice 7.4 *Turingův stroj* $T = (Q,X,Y,\delta,q_0,B,F)$ *přijímá slovo* $a\alpha \in X^{+}$[1], *pokud platí* $(a,q_0,\alpha) \underset{*}{\vdash} (B,q,\Lambda)$ *pro nějaké* $q \in F$.

T *přijímá* Λ, *když* $(B,q_0,\Lambda) \underset{*}{\vdash} (B,q,\Lambda)$ *pro nějaké* $q \in F$.[2]

Slovo α je tedy přijímáno *T.s.* T, pokud po skončení výpočtu je vstupní páska celá smazaná (přepsaná na B), čtecí hlava je na prvním políčku pásky a T se nachází v přijímacím stavu[3].

Podobně jako již několikrát, definujeme $L(T) = \{\alpha \in X^*; T \text{ přijímá } \alpha\}$ jazyk všech slov přijímaných *T.s.* T.

Definice 7.5 *Jazyk* L *nazveme rekursivně spočetným, jakmile je přijímán nějakým Turingovým strojem* T $(L = L(T))$.

Poznámka. Důkazy všech tvrzení, které si budeme o Turingových strojích uvádět, by při formálním provedení zabraly velmi mnoho prostoročasu a byly by značně nepřehledné. Omezíme se proto vždy na vysvětlení hlavní myšlenky, případně použitých triků, což považuji za dostačující[4].

7.1 Odvozené Turingovy stroje

Pro libovolný *T.s* T můžeme předpokládat $|F| = 1$, protože kdyby $|F| > 1$, lze T rozšířit o jeden stav $q_{f'}$, do kterého povedou Λ–přechody[5] ze všech stavů z F a předefinovat $F = \{q_{f'}\}$. Jazyk $L(T)$ se zřejmě nezmění a $|F| = 1$.

Podobně můžeme říci, že T přijímá $a\alpha$, pokud $(a,q_0,\alpha) \underset{*}{\vdash} (\beta,q,\gamma), q \in F$, neboť T lze rozšířit tak, aby po dosažení stavu $q \in F$ smazal obsah pásky a přešel do nového přijímacího stavu $q_{f'}$, čímž dostáváme *T.s.* přijímající $a\alpha$ podle původní definice (7.4). Pochopitelně při použití této definice přijímacího výpočtu

[1] pro nějaké $a \in X$, $\alpha \in X^*$

[2] Místo Λ zde může být libovolné slovo z Blanků, záleží na kandidátovi vybraném ze třídy ekvivalentních konfigurací.

[3] Definice akceptující konfigurace se často v různých učených knihách liší, nicméně na třídy přijímaných jazyků to nemá vliv.

[4] takové důkazy budeme označovat CBD/2, abychom je odlišili od těch „korektnějších"

[5] Λ–přechody jsou zde realizovány možností nepohybovat se po vstupní pásce a zapisovat přečtené písmeno, tzn. $\delta(q,x) = (q',x,0)$

je třeba zajistit, aby se *T.s.* nedostal do stavů v F před přečtením vstupního slova.

Sedmice $T_k = (Q, X, Y, \delta, q_0, B, F)$ je k–páskový Turingův stroj s k páskami ($k < +\infty$)[6]. Ze všech pásek T_k najednou čte i na všechny najednou může zapisovat. Čtecí hlavy nemusí být na páskách všechny na stejné pozici. Definice k–páskového stroje je vesměs stejná jako u jednopáskového, až na

$$\delta : (Q \setminus F) \times Y^k \to Q \times Y^k \times \{-1, 0, 1\}^k,$$

resp.

$$\delta : (Q \setminus F) \times Y^k \to Q \times (Y \times \{-1, 0, 1\})^k.$$

Konfigurace k–páskového stroje je $(\alpha_i, q, \beta_i)_{i=1}^k$[7]. Před začátkem výpočtu obsahuje první páska vstupní slovo, ostatní jsou prázdné.

Tvrzení 7.6 *k–páskové Turingovy stroje přijímají právě rekursivně spočetné jazyky.*

Důkaz: K danému k–páskovému $T_k = (Q, X, Y, \delta, q_0, B, F)$ sestrojíme jednopáskový *T.s* T, přijímající stejný jazyk jako T_k. *T.s* $T = (Q', X, Y', \delta', q_0, B, F')$ si bude na své jediné pásce pamatovat obsahy pásek a polohy čtecích hlav na jednotlivých páskách stroje T_k. Proto definujeme $Y' = X \cup (Y \times \{0, 1\})^k \cup \{B\}$. Je-li T_k v konfiguraci $(\alpha_1 q \beta_1) \dots (\alpha_k q \beta_k)$, pak na i–tém místě pásky T je symbol $({}_iy_j, {}_iu_j)_{j=1}^k$, kde[8] ${}_iy_j$ je i–té písmeno slova $\alpha_j \beta_j B^*$ a ${}_iu_j = 1$, pokud hlava na j–té pásce je na i–té pozici (jinak ${}_iu_j = 0$)[9]. Podrobnosti o Q', F' a δ' si odpustím. *T.s* T pracuje podle následujícího algoritmu:

1. T přepíše svoji pásku tak, aby kódovala pásky[10] T_k, hlava T bude na první pozici a q_0 bude kódován ve stavu T.

2. T přejde celou popsanou pásku, zjistí, které písmeno čte j–tá hlava T_k pro všechna $j = 1, \dots, k$, a zapamatuje si to ve stavech.

3. Simuluje krok T_k a výsledek si zapamatuje ve stavech.

4. Hlava se pohybuje nazpátek a aktualizuje obsah pásky. Pokračuje krokem 2.

5. Když se simulace T_k dostane do přijímacího stavu, T smaže obsah pásky, přesune hlavu na první pozici a přejde do přijímacího stavu.

[6] Tzn. pásek je konečně mnoho.

[7] tzn. konfigurace jsou z množiny $(Y^* \times Q \times Y^*)^k$; povšimněte si prosím, že pro každou k-tici je stav z množiny Q stejný pro všechny složky

[8] zápis ${}_iu_j$ není nějaký standartní, jen jsem nevěděl, kam umístit index i, aby nevypadal jako mocnina nebo index dvourozměrného pole

[9] je vhodné si uvědomit, že pro pevné j existuje právě jedno i takové, že ${}_iu_j = 1$

[10] T sice kóduje všechny pásky, ale na začátku výpočtu T_k jsou všechny pásky kromě první prázdné

Je myslím možné intuitivně nahlédnout, že $L(T) = L(T_k)$. Pochopitelně T bude mít daleko více stavů, neboť v nich musí kódovat písmena přečtená na páskách T_k. Funkce δ' asi také není zrovna „čtení ke kafi“, nicméně tyto problémy nechám vážným zájemcům o teorii automatů. CBD/2

Definice 7.7 *Nedeterministický jednopáskový Turingův stroj*[11] *je definován jako deterministický. Rozdíl je opět pouze v*

$$\delta : (Q \setminus F) \times Y \to \mathcal{P}(Q \times Y \times \{-1, 0, 1\}).$$

Slovo $a\alpha$ je přijímáno N-Turingovým strojem $(Q, X, Y, \delta, q_0, B, F)$, když existuje přijímací výpočet $(a, q_0, \alpha) \vdash_*^{} (B, q, \Lambda)$ pro $q \in F$. Slovo Λ je přijímáno, když existuje přijímací výpočet $(B, q_0, \Lambda) \vdash_*^{} (B, q, \Lambda)$ pro nějaké $q \in F$.

Tvrzení 7.8 *N-Turingovy stroje přijímají právě rekursivně spočetné jazyky.*

Důkaz: Pro daný *N-T.s.* $T_N = (Q, X, Y, \delta, q_0, B, F)$ budeme definovat deterministický třípáskový stroj $T' = (Q', X, Y', \delta', q_0', B, F')$, jehož první páska obsahuje vstupní slovo, druhá simuluje výpočet T_N a třetí určuje, který výpočet se provádí.

Je-li k maximální velikost množiny $\delta(q, y)$, pak pro všechna $q \in Q, y \in Y$ taková, že $\delta(q, y) \neq \emptyset$, zvolíme posloupnost $O_{q,y}$ délky nejvýše k z prvků množiny $Q \times Y \times \{-1, 0, 1\}$ takovou, že $((r, a, i) je\ členem\ O_{q,y}) \Leftrightarrow ((r, a, i) \in \delta(q, y))$[12].

Na třetí pásce je slovo z $\{1, \ldots, k\}$ délky i. Tím je určeno, že na druhé pásce se provede výpočet T_N mající i kroků a pro j–tý krok výpočtu je vybrán l_j–tý člen posloupnosti $O_{q,y}$ (pro q, y určené v kroku $j-1$), kde l_j je j–té písmeno na třetí pásce. Výsledný efekt je ten, že existuje jednoznačná korespondence mezi všemi výpočty T_N a obsahem třetí pásky.[13]

T' pracuje podle schématu

1. Na třetí pásce napíše na první pozici 1 a zbytek nechá B. Na první pásce je vstupní slovo.

2. Simuluje prvních i kroků výpočtu T, kde i je délka slova na třetí pásce. Jestliže výpočet skončí předčasně v nepřijímacím stavu nebo proběhne všech i kroků a T nepřijme, pak T' vygeneruje na třetí pásce další posloupnost a pokračuje krokem 2.

3. Pokud najde přijímací výpočet, smaže všechny pásky, nastaví hlavy na začátek a přejde do přijímacího stavu.

T' je deterministický přijímá stejný jazyk jako T_N. CBD/2

[11] Pro zkrácení budeme často psát *N-T.s.* a *D-T.s.* místo nedeterministický a deterministický Turingův stroj.

[12] $O_{q,y}$ se liší od $\delta(q, y)$ tím, že je uspořádaná

[13] Pro odtemnění. Na třetí pásce je slovo $l_1 \ldots l_i$, kde $l_j \in \{1, \ldots, k\}$. Když T' v j-tém kroku čte a_j (nemusí být j-té písmeno na první pásce) a je ve stavu q, pak pokračujeme l_j-tým členem posloupnosti O_{q,a_j} (pokud takový člen neexistuje, T' generuje další posloupnost — viz níže)

Definice 7.9 *T.s. je lineární, pokud nemůže překročit délku vstupního slova. To znamená, že pro všechna $q \in Q$ buď $\delta(q, B)$ není definováno (prázdná množina), nebo $\delta(q, B) = (q', B, -1)$*[14].

Tvrzení 7.10 *Pro každý T.s. T existuje gramatika G taková, že $L(G) = L(T)$. Je-li T lineární, pak existuje nezkracující gramatika s touto vlastností.*

Důkaz: Pro Turingův stroj $T = (Q, X, Y, \delta, q_0, B, \{q_f\})$ vybudujeme gramatiku $G = (\{B_1, S, M\} \cup Q \cup Y, X, S, P)$, která bude pro každý přijímací výpočet $(a, q_0, \alpha) \overset{T}{\underset{*}{\vdash}} (B, q_f, \Lambda)$ generovat postupně odzadu jednotlivé konfigurace T, až se dostane k výchozí konfiguraci, ze které nakonec vypudí q_0 a získá vstupní slovo[15]. B_1 je pouze formální symbol, který říká, že za ním stojí již jen samá B. Podobně B_0 formálně určuje začátek pásky. P obsahuje následující pravidla (pro všechna $x, y, z \in Y, q, r \in Q$):

1. $(S \to B_0 B q_f B_1)$
2. $(rxy \to zqy)$, pokud $\delta(q, z) = (r, x, -1)$
3. $(xry \to zqy)$, pokud $\delta(q, z) = (r, x, 0)$
4. $(xyr \to zqy)$, pokud $\delta(q, z) = (r, x, +1)$
5. $(rxB_1 \to zqB_1)$, pokud $\delta(q, z) = (r, x, -1)$
6. $(rB_1 \to zqB_1)$, pokud $\delta(q, z) = (r, B, -1)$
7. $(xrB_1 \to zqB_1)$, pokud $\delta(q, z) = (r, x, 0)$
8. $(xBrB_1 \to zqB_1)$, pokud $\delta(q, z) = (r, x, +1)$
9. $(B_0 x q_0 \to xM)$, pokud $x \in X$
10. $(B_0 B q_0 \to M)$
11. $(Mx \to xM)$
12. $(MB_1 \to \Lambda)$
13. $(MB \to M)$

Pokud $a\alpha \in L(T)$, existuje přijímací výpočet $(a, q_0, \alpha B_1) \overset{}{\underset{*}{\vdash}} (B, q_f, B_1)$, což odpovídá v G derivaci

$$S \Longrightarrow B_0 B q_f B_1 \underset{*}{\Longrightarrow} B_0 a q_0 \alpha B_1 \Longrightarrow aM\alpha B_1 \underset{*}{\Longrightarrow} a\alpha M B_1 \Longrightarrow a\alpha \in L(G)$$

Pokud $\Lambda \in L(T)$, existuje přijímací výpočet $(B, q_0, \Lambda) \overset{}{\underset{*}{\vdash}} (B, q_f, \Lambda)$, což v G odpovídá derivaci

$$S \Longrightarrow B_0 B q_f B_1 \underset{*}{\Longrightarrow} B_0 B q_0 B_1 \Longrightarrow M B_1 \Longrightarrow \Lambda \in L(G)$$

[14] Tato druhá možnost je nutná, aby *T.s.* měl vůbec šanci poznat konec vstupního slova

[15] velmi názorně je to vidět v příkladu 7.18

Je-li $\alpha \in L(G)$, existuje odvození $S \Longrightarrow \gamma_1 \Longrightarrow \cdots \Longrightarrow \gamma_k = \alpha$, kde pro nějaké i slovo γ_i obsahuje nějaký stav a γ_{i+1} již ne. Stavem obsaženým v γ_i musel být z definice G stav q_0 bezprostředně následující po prvním písmenu γ_i[16], neboli $\gamma_i \overset{T}{\underset{*}{\vdash}} \gamma_1$ je přijímací výpočet T nad α, tzn. $L(G) = L(T)$.

Je-li T lineární, pak nesmí překročit délku vstupního slova, neboli pravidlo 8 se nikdy nepoužije. Problém jsou pravidla 9, 10, 12 a 13. Pro jejich odstranění použijeme jednoduchý trik. Všimněme si, že každé vygenerované slovo obsahuje před použitím pravidla 9, resp. 10 právě jedno písmeno B_0, které je vždy na začátku slova, a právě jedno písmeno B_1, které je vždy na konci slova. Stejně tak obsahuje právě jedno písmeno z množiny Q. Vytvořme novou množinu neterminálů $V'_N = Y \times Q' \times \{0,1\} \times \{0,1\}$, kde Q' vznikne z Q přidáním nového písmene 0. Nyní libovolné slovo $\alpha = a_1 \ldots a_n \in (V_N \cup X)^*$[17] vygenerované gramatikou G bez použití pravidla 9, resp. 10 transformujeme na slovo $\alpha^T = b_1 \ldots b_{n-3} \in V'_N$ tak, že

- $b_i = (a_i, 0, 0, 0)$, když $a_{i+1} \notin Q \cup \{B_1\}$ a $a_{i-1} \neq B_0$
- $b_i = (a_i, q, 0, 0)$, když $a_{i+1} = q \in Q$, $a_{i+2} \neq B_1$ a $a_{i-1} \neq B_0$
- $b_i = (a_i, 0, 1, 0)$, když $a_{i+1} = B_1$ a $a_{i-1} \neq B_0$
- $b_i = (a_i, q, 1, 0)$, když $a_{i+1} = q \in Q$, $a_{i+2} = B_1$ a $a_{i-1} \neq B_0$
- $b_i = (a_i, 0, 0, 1)$, když $a_{i+1} \notin Q \cup \{B_1\}$ a $a_{i-1} = B_0$
- $b_i = (a_i, q, 0, 1)$, když $a_{i+1} = q \in Q$, $a_{i+2} \neq B_1$ a $a_{i-1} = B_0$
- $b_i = (a_i, 0, 1, 1)$, když $a_{i+1} = B_1$ a $a_{i-1} = B_0$
- $b_i = (a_i, q, 1, 1)$, když $a_{i+1} = q \in Q$, $a_{i+2} = B_1$ a $a_{i-1} = B_0$

pro všechna $a_i \in \{S, M\} \cup Y$. Písmena $a_i \in Q \cup \{B_0, B_1\}$ nenahrazujeme[18].

Nyní zkonstruujeme gramatiku $G^T = (V'_N, X, (B, q_f, 1), P')$, která bude nezkracující a bude generovat stejný jazyk jako G. Prvních sedm typů pravidel gramatiky G zkopírujeme do G^T a upravíme je tak, aby platilo, že když se v původní gramatice vygeneruje slovo α, pak se v pozměněné gramatice G^T vygeneruje slovo α^T[19]. Zbylá pravidla z gramatiky G vyřadíme a do G^T přidáme pravidla

- $(a, q_0, 0, 1)(b, 0, 0, 0) \to ab$ pro $a, b \in X$

[16] nepočítaje B_0

[17] $V_N = \{B_0, B_1, S, M\} \cup Q \cup Y$

[18] komu se správně zdá, že ve slově α^T zůstanou mezery po B_0, B_1 a stavu z Q, může být klidný — po skončení nahrazování tyto mezery vyhodíme a získáme slovo α^T délky $n-3$ bez mezer.

[19] například pravidla typu 5 přejdou na pravidla typu

$$\{(u, r, 0, i)(x, 0, 1, 0) \to (u, 0, 0, i)(z, q, 1, 0);$$
$$i \in \{0,1\}, u, x, z \in Y, r, q \in Q, \delta(q, z) = (r, x, -1)\}$$

- $(a, q_0, 0, 1)(b, 0, 1, 0) \to ab$ pro $a, b \in X$
- $(a, q_0, 1, 1) \to a$ pro $a \in X$
- $a(b, 0, 0, 0) \to ab$ pro $a, b \in X$
- $a(b, 0, 1, 0) \to ab$ pro $a, b \in X$

Lehce se přesvědčíme, že nová gramatika je nezkracující a generuje jazyk $L(T)$. CBD/2

Tvrzení 7.11 *Pro každou gramatiku G existuje T.s. T tak, že $L(G) = L(T)$. Je-li G nezkracující, pak T je lineární.*

Důkaz: Mějme gramatiku $G = (V_N, V_T, S, P)$. Definujeme dvoupáskový nedeterministický *T.s.* $T = (Q, V_T, V_N \cup V_T, \delta, q_0, r_0, F)$. Na první pásce bude vstupní slovo, na druhé bude probíhat simulace generování G[20]. T pracuje podle algoritmu:

1. Na druhé pásce vygeneruje S. Na první pásce je vstupní slovo.
2. Je-li na druhé pásce slovo α_i (neboli $S \overset{*}{\Longrightarrow} \alpha_i$) a α_i obsahuje neterminál, pak T „uhodne“ pravidlo p použité k dalšímu generování a místo použití v α_i a přepíše část druhé pásky podle p, čímž dostane další slovo α_{i+1}.
3. Jakmile slovo na druhé pásce neobsahuje neterminál, porovná jej s první páskou. Pokud se slova shodují, máme vyhráno. Pokud ne, nemusíme zoufat, neboť T je nedeterministický.

Je-li G nezkracující a T by vygeneroval na druhé pásce slovo delší než vstupní, každá derivace na druhé pásce již bude delší než vstupní slovo, protože T má k dispozici pouze nezkracující pravidla. CBD/2

Z tvrzení 7.10,7.11 a 5.9[21] vyplývá

Důsledek 7.12 *Gramatiky generují právě rekursivně spočetné jazyky. Kontextové jazyky jsou přijímány právě lineárními T.s.*

Tvrzení 7.13 *Rekursivně spočetné i kontextové jazyky jsou uzavřeny na konečná sjednocení, konkatenaci, iteraci a zrcadlový obraz.*

Důkaz: Konstrukce gramatik generujících výsledky zmíněných operací je stejná jako v kapitole 5.2.3[22]. Důkaz lze provést i přes konstrukci *T.s.* přijímajících uvedené jazyky. Většinou vystačíme s dvoupáskovými stroji, simulujícími na každé pásce výpočet jednoho stroje přijímajícího vstupní jazyk.

Tvrzení 7.14 *Kontextové jazyky jsou uzavřeny na doplňky, tím pádem i na průniky.*

První část tvrzení patří mezi tzv. *LBA-problémy*[23]:

[20] T je nedeterministický proto, že G může obsahovat více pravidel s jednou levou stranou.
[21] Nezkracující gramatiky generují právě kontextové jazyky.
[22] Při konkatenaci musíme oddělit generování obou gramatik od sebe, což uděláme tak, že gramatiky budou mít disjunktní množiny neterminálů a žádný terminál se nevyskytne na levé straně žádného pravidla. Nechávám čtenáři na rozvážení, že toto omezení neujímá důkazu na obecnosti.
[23] Linear Bounding Automatons — lineárně omezené automaty

1. Jsou kontextové jazyky přijímány deterministickými lineárními *T.s.*? — dosud nevyřešeno!!!
2. Jsou kontextové jazyky uzavřeny na doplňky? — Ano.

Poznámka. Je-li T Turingův stroj pracující nad vstupní abecedou X, pak pro $\alpha \in X^*$ může zřejmě nastat právě jeden z následujících případů:

- Výpočet T nad α skončí.
- Výpočet se zacyklí (vrátí se do stejné konfigurace).
- Výpočet pokračuje do nekonečna (nikdy neprojde dvakrát stejnou konfigurací).

Definice 7.15 *T.s. T rozhoduje jazyk L nad X, když se výpočet pro každé vstupní slovo zastaví a $L(T) = L$. Jazyk L je rekursivní, pokud existuje T.s., který ho rozhoduje.*

Tato definice tvoří model pojmu algoritmus.

Věta 7.16 *L je rekursivní, právě když L a $\bar{L}$ (doplněk) jsou rekursivně spočetné.*

Důkaz: Je-li L rekursivní, pak existuje *T.s.* $T = (Q, X, Y, \delta, q_0, B, F)$, který ho rozhoduje. Předpokládejme, že T má právě dva stavy q_e, q_f, ve kterých se zastaví. Nechť $F = \{q_f\}$. Pak T přijímá L a $(Q, X, Y, \delta, q_0, \{q_e\})$ zjevně přijímá $\bar{L}$, tzn. L, $\bar{L}$ jsou rekursivně spočetné.

Naopak, jsou-li L, $\bar{L}$ rekursivně spočetné, pak existují *T.s.* T_1, T_2 takové, že $L = L(T_1)$, $\bar{L} = L(T_2)$. Definujme dvoupáskový *T.s.* T', který na první pásce simuluje výpočet T_1, na druhé T_2. Nechť T' se zastaví, jakmile se zastaví T_1 nebo T_2, tedy se zastaví[24] nad každým vstupním slovem, a přijme, pokud přijímá T_1, tzn. pokud simulace na první pásce skončí v přijímacím stavu. CBD/2

Z věty 7.16 vyplývá, že jazyk L, který není rekursivní, není ani kontextový. Kdyby totiž byl kontextový, byl by i jeho doplněk kontextový[25], tedy by L i $\bar{L}$ byly rekursivně spočetné[26], neboli L by byl rekursivní, což je spor. Tento fakt později využijeme, proto ho zformalizujeme do lemmatu.

Lemma 7.17 *Každý kontextový jazyk je rekursivní.*

Churchova these: *Každý jazyk, který lze nějakým způsobem popsat konečným výrazem, je rekursivně spočetný.*

Churchova these je metavěta, nelze ji tedy v jazyce automatů dokázat, jen vyvrátit. Problémy s dokazatelností plynou z nemožnosti formalizovat pojem „konečný výraz“.

[24] zastaví, ne pozastaví

[25] protože kontextové jazyky jsou uzavřeny na doplňky (viz tvrzení 7.14)

[26] protože rekursivně spočetné jazyky jsou generovány obecnými gramatikami a jsou tedy nadmnožinou kontextových jazyků

Příklad 7.18 *(Turingův stroj) Sestrojíme T.s.* $T = (Q, X, Y, \delta, q_0, B, F)$ *přijímající palindromy nad abecedou* $X = \{c, d\}$*, tzn.* $L(T) = \{\alpha; \alpha = \alpha^R\}$.

Řešení: Slovní vysvětlení připojím později. Definujme Turingův stroj T:

$$\begin{aligned} Q &= \{q_0, \bar{q}_0, q_1, q_c, q_d, q_1^c, q_1^d, q_2, q_f\} \\ F &= \{q_f\} \\ Y &= \{c, d, B, \#\} \end{aligned}$$

Funkce $\delta : (Q \setminus F) \times Y \to Q \times Y \times \{-1, 0, 1\}$ je popsána tabulkou

$X \backslash Q$	q_0	$\bar{q}_0$	q_1	q_c
c	$q_c, \#, +1$	$q_c, B, +1$	$q_1, c, -1$	$q_c, c, +1$
d	$q_d, \#, +1$	$q_d, B, +1$	$q_1, d, -1$	$q_c, d, +1$
B	$q_f, B, 0$	$q_2, B, -1$	$\bar{q}_0, B, +1$	$q_1^c, B, -1$
$\#$	—	—	$\bar{q}_0, \#, +1$	—

	q_d	q_1^c	q_1^d	q_2
c	$q_d, c, +1$	$q_1, B, -1$	—	—
d	$q_d, d, +1$	—	$q_1, B, -1$	—
B	$q_1^d, B, -1$	$q_2, B, -1$	$q_2, B, -1$	$q_2, B, -1$
$\#$	—	$q_f, B, 0$	$q_f, B, 0$	$q_f, B, 0$

T je konstruován tak, že vždy přečte písmeno na začátku vstupního slova, zapamatuje si ho ve stavu (q_c nebo q_d) a smaže ho. V q_c nebo q_d přejde na konec slova (aniž by měnil údaje na vstupní pásce) a pokud je poslední písmeno shodné s prvním (viz indexy stavů), smaže ho a vrací se na začátek slova, dokud nenarazí na B nebo # (návrat probíhá v q_1)[27]. Pak se zase obrátí a s informací o prvním písmenu jde na konec, což se opakuje tak dlouho, dokud se první a poslední písmena shodují a není celá páska smazána. Po smazání celé pásky T přejde do q_2, dojde na začátek pásky (stále v q_2), kde leží symbol #, přes který přejde do přijímacího stavu q_f a # smaže — slovo je přijato. Je vhodné si rozmyslet, jak T přijímá prázdné slovo a jak se vypořádá s palindromy liché délky, což nechávám jako cvičení.

Když už máme *T.s.* T, přijímající palindromy, sestrojíme podle návodu z důkazu tvrzení 7.10 gramatiku $G = (V_N, V_T, S, P)$, která palindromy generuje.

$$\begin{aligned} V_N &= \{B_1, S, M, q_0, \bar{q}_0, q_1, q_c, q_d, q_1^c, q_1^d, q_2, q_f, B, \#, c, d\} \\ V_T &= \{c, d\} \end{aligned}$$

P bude obsahovat následující pravidla. Uvádím je označená vždy podle příslušného bodu v důkazu 7.10. y značí libovolný symbol z $\{B, \#, c, d\}$, $x \in \{c, d\}$

$$\begin{aligned} {}^{1)}S &\to B_0 B q_f B_1 \\ {}^{2)}q_1^c B y &\to B q_c y \\ q_1^d B y &\to B q_d y \\ q_1 B y &\to c q_1^c y \mid d q_1^d y \end{aligned}$$

[27] Není-li poslední písmeno shodné s prvním, pak se výpočet zastaví (δ není definována).

$$
\begin{array}{rcl}
q_1cy & \to & cq_1y \\
q_1dy & \to & dq_1y \\
q_2By & \to & B\bar{q}_0y \mid Bq_1^cy \mid Bq_1^dy \mid Bq_2y \\
^{3)}Bq_fy & \to & Bq_0y \mid \#q_1^cy \mid \#q_1^dy \mid \#q_2y \\
^{4)}Byq_c & \to & c\bar{q}_0y \\
Byq_d & \to & d\bar{q}_0y \\
cyq_c & \to & cq_cy \\
dyq_c & \to & dq_cy \\
cyq_d & \to & cq_dy \\
dyq_d & \to & dq_dy \\
By\bar{q}_0 & \to & Bq_1y \\
\#yq_c & \to & cq_0y \\
\#yq_d & \to & dq_0y \\
\#y\bar{q}_0 & \to & \#q_1y \\
^{5)}q_1^cBB_1 & \to & Bq_cB_1 \\
q_1^dBB_1 & \to & Bq_dB_1 \\
q_1BB_1 & \to & cq_1^cB_1 \mid dq_1^dB_1 \\
q_2BB_1 & \to & B\bar{q}_0B_1 \mid Bq_1^cB_1 \mid Bq_1^dB_1 \mid Bq_2B_1 \\
q_1cB_1 & \to & cq_1B_1 \\
q_1dB_1 & \to & dq_1B_1 \\
^{6)}q_1^cB_1 & \to & Bq_cB_1 \\
q_1^dB_1 & \to & Bq_dB_1 \\
q_1B_1 & \to & cq_1^cB_1 \mid dq_1^dB_1 \\
q_2B_1 & \to & B\bar{q}_0B_1 \mid Bq_1^cB_1 \mid Bq_1^dB_1 \mid Bq_2B_1 \\
^{7)}Bq_fB_1 & \to & Bq_0B_1 \mid \#q_1^cB_1 \mid \#q_1^dB_1 \mid \#q_2B_1 \\
^{8)}BBq_cB_1 & \to & c\bar{q}_0B_1 \\
BBq_dB_1 & \to & d\bar{q}_0B_1 \\
cBq_cB_1 & \to & cq_cB_1 \\
cBq_dB_1 & \to & cq_dB_1 \\
dBq_dB_1 & \to & dq_dB_1 \\
dBq_cB_1 & \to & dq_cB_1 \\
BB\bar{q}_0B_1 & \to & Bq_1B_1 \\
\#Bq_cB_1 & \to & cq_0B_1 \\
\#Bq_dB_1 & \to & dq_0B_1 \\
\#B\bar{q}_0B_1 & \to & \#q_1B_1 \\
^{9)}B_0xq_0 & \to & xM \\
^{10)}B_0Bq_0 & \to & M \\
^{11)}Mc & \to & cM
\end{array}
$$

$$\begin{aligned} Md &\to dM \\ MB &\to BM \\ M\# &\to \#M \\ {}^{12)}MB_1 &\to \Lambda \\ {}^{13)}MB &\to M \end{aligned}$$

Pokusme se nyní vygenerovat nějaký jednoduchý palindrom, např. c. Která pravidla G a v jakém pořadí použijeme? Experimentálně jsem ověřil, že nejlepší je zjistit posloupnost stavů (resp. konfigurací) při výpočtu T a zpětně určit pravidla použitá pro generování. Výpočet nad slovem c probíhá konfigurace

$$(c, q_0, B\Lambda) \overset{T}{\vdash} (\#B, q_c, \Lambda) \overset{T}{\vdash} (\#, q_1^c, B\Lambda) \overset{T}{\vdash} (B, q_f, B\Lambda) \sim (B, q_f, \Lambda)$$

Teď stačí v podstatě vzít posloupnost konfigurací, obrátit ji, místo $\overset{T}{\vdash}$ dát $\overset{G}{\Longrightarrow}$ a dostáváme[28]

$$\begin{aligned} S &\overset{1}{\Longrightarrow} B_0Bq_fB_1 \overset{7}{\Longrightarrow} B_0\#q_1^cB_1 \overset{6}{\Longrightarrow} \\ &\overset{6}{\Longrightarrow} B_0\#Bq_cB_1 \overset{8}{\Longrightarrow} B_0cq_0B_1 \overset{9}{\Longrightarrow} cMB_1 \overset{12}{\Longrightarrow} c \end{aligned}$$

Pro zajímavost uvádím ještě odvození palindromu *cdccdc*, již bez návodné posloupnosti konfigurací stroje T. Nad operátorem odvození je opět číslo použitého pravidla, nahrazujeme podtrženou část slova.

$$\begin{aligned} &\underline{S} \overset{1}{\Longrightarrow} B_0\underline{Bq_fB_1} \overset{7}{\Longrightarrow} B_0\#\underline{q_2B_1} \overset{6}{\Longrightarrow} B_0\#B\underline{q_2B_1} \overset{6}{\Longrightarrow} B_0\#BB\underline{q_2B_1} \overset{6}{\Longrightarrow} \\ &\overset{6}{\Longrightarrow} B_0\#B\underline{BB\bar{q}_0B_1} \overset{8}{\Longrightarrow} B_0\#BB\underline{q_1B_1} \overset{6}{\Longrightarrow} B_0\#BBc\underline{q_1^cB_1} \overset{6}{\Longrightarrow} \\ &\overset{6}{\Longrightarrow} B_0\#BB\underline{cBq_c}B_1 \overset{4}{\Longrightarrow} B_0\#B\underline{Bcq_c}BB_1 \overset{4}{\Longrightarrow} \#\underline{Bc\bar{q}_0}cBB_1 \overset{4}{\Longrightarrow} \\ &\overset{4}{\Longrightarrow} B_0\#B\underline{q_1cc}BB_1 \overset{2}{\Longrightarrow} B_0\#Bc\underline{q_1cB}B_1 \overset{2}{\Longrightarrow} B_0\#Bcc\underline{q_1BB_1} \overset{5}{\Longrightarrow} \\ &\overset{5}{\Longrightarrow} B_0\#Bccd\underline{q_1^dB_1} \overset{6}{\Longrightarrow} B_0\#Bccd\underline{Bq_d}B_1 \overset{4}{\Longrightarrow} B_0\#Bc\underline{cdq_d}BB_1 \overset{4}{\Longrightarrow} \\ &\overset{4}{\Longrightarrow} B_0\#B\underline{ccq_d}dBB_1 \overset{4}{\Longrightarrow} B_0\#\underline{Bcq_d}cdBB_1 \overset{4}{\Longrightarrow} B_0\underline{\#d\bar{q}_0}ccdBB_1 \overset{4}{\Longrightarrow} \\ &\overset{4}{\Longrightarrow} B_0\#\underline{q_1dcc}dBB_1 \overset{2}{\Longrightarrow} B_0\#d\underline{q_1cc}dBB_1 \overset{2}{\Longrightarrow} B_0\#dc\underline{q_1cd}BB_1 \overset{2}{\Longrightarrow} \\ &\overset{2}{\Longrightarrow} B_0\#dcc\underline{q_1dB}B_1 \overset{2}{\Longrightarrow} B_0\#dccd\underline{q_1BB_1} \overset{5}{\Longrightarrow} B_0\#dccdc\underline{q_1^cB_1} \overset{6}{\Longrightarrow} \\ &\overset{6}{\Longrightarrow} B_0\#dccd\underline{cBq_c}B_1 \overset{4}{\Longrightarrow} B_0\#dccd\underline{cq_c}BB_1 \overset{4}{\Longrightarrow} B_0\#dc\underline{cdq_c}cBB_1 \overset{4}{\Longrightarrow} \\ &\overset{4}{\Longrightarrow} B_0\#d\underline{ccq_c}dcBB_1 \overset{4}{\Longrightarrow} B_0\#\underline{dcq_c}cdcBB_1 \overset{4}{\Longrightarrow} B_0\underline{\#dq_c}ccdcBB_1 \overset{4}{\Longrightarrow} \\ &\overset{4}{\Longrightarrow} \underline{B_0cq_0}dccdcBB_1 \overset{9}{\Longrightarrow} c\underline{Md}ccdcBB_1 \overset{11}{\Longrightarrow} cd\underline{Mc}cdcBB_1 \overset{11}{\Longrightarrow} \\ &\overset{11}{\Longrightarrow} cdc\underline{Mc}dcBB_1 \overset{11}{\Longrightarrow} cdcc\underline{Md}cBB_1 \overset{11}{\Longrightarrow} cdccd\underline{Mc}BB_1 \overset{11}{\Longrightarrow} \\ &\overset{11}{\Longrightarrow} cdccdc\underline{MB}B_1 \overset{13}{\Longrightarrow} cdccdc\underline{MB_1} \overset{12}{\Longrightarrow} cdccdc \end{aligned}$$

Z příkladu je velmi dobře vidět, jak gramatika G přesně kopíruje obrácený výpočet T, jak stavy q_1 a q_2 putují vstupním slovem doleva a stavy q_c a q_d doprava a jak fungují zarážky q_1^c, q_1^d, $\bar{q}_0$ a q_0. FINE

[28] nad operátor $\Longrightarrow$ píši místo jména gramatiky číslo skupiny použitého pravidla

7.2 Univerzální Turingův stroj

Každý *T.s.*[29] lze jedno-jednoznačně zakódovat do řetězce znaků. Na jednoduchém příkladu ukážeme jednu z mnoha možností kódování.

Příklad 7.19 *(Kódování) Mějme T.s.* $T = (Q, X, Y, \delta, q_0, B, F)$, *kde*

$$\begin{aligned} Q &= \{q_0, q_1\} \\ X &= \{a, b\} \\ Y &= \{a, b, B, \#\} \\ F &= \{q_0\} \end{aligned}$$

$Q \backslash X$	a	b	B	$\#$
q_0	$q_1, \#, +1$	$q_1, \#, +1$	$q_0, B, -1$	$q_0, B, 0$
q_1	$q_1, B, +1$	$q_1, a, 0$	$q_0, B, -1$	—

Zakódujte T jedno-jednoznačně do řetězce $\{0,1\}^*$.

Nehloubejte příliš nad tím, co T dělá (podle mého názoru přijímá celou množinu X^*), důležité je, že ho zakódujeme do řetězce z $\{0,1\}^*$[30]. Řekněme, že

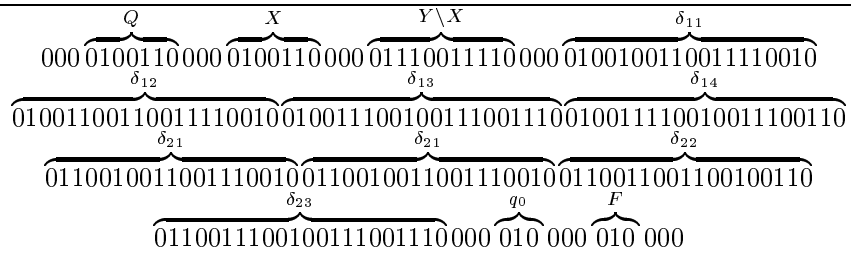
$$\begin{aligned} q_0 &= 010 \\ q_1 &= 0110 \\ a &= 010 \\ b &= 0110 \\ B &= 01110 \\ \# &= 011110 \\ +1 &= 010 \\ 0 &= 0110 \\ -1 &= 01110 \\ (\delta(q,x) = (q', y, j)) &= \text{kód}(q)\text{kód}(x)\text{kód}(q')\text{kód}(y)\text{kód}(j) \end{aligned}$$

Všimněte si, že všechny kódy začínají a končí nulou a uvnitř žádnou nulu neobsahují. Kódu 000 lze užít jako oddělovače. Zakódujeme postupně množiny Q, X, $Y \setminus X$, funkci δ, počáteční stav, Blank a F a vytvoříme z nich jeden řetězec, kde jednotlivé komponenty oddělíme třemi nulami (tabulka 3). Tento řetězec jedno-jednoznačně kóduje T. FINE

Samozřejmě je nepřeberné množství jiných způsobů kódování. Pro nás však není důležité, *jak* stroje kódovat, ale *že* jedno-jednoznačně kódovat jdou.

Definice 7.20 *Turingův stroj nazveme univerzální, právě když přijímá všechny dvojice* $(kód(T); \alpha)$[31] *takové že T.s. T přijímá slovo* α.

[29] obecně jakýkoliv automat
[30] jak jinak, vždyť jsme informatici
[31] kde znak „;“ (středník) je nějaký jednoznačný oddělovač



Tabulka 3: Kód stroje T. (δ_{ij} znamená, že kódujeme i-tý řádek a j-tý sloupec tabulky zadání funkce δ).

Univerzální $T.s$ existuje a jazyk, který přijímá, je

$$L_u = \{(\text{kód}(T); \alpha);\ T \text{ přijímá } \alpha\}$$

a je nesporně rekursivně spočetný. Naznačím způsob práce univerzálního Turingova stroje. Bude mít dvě pásky. Na první pásce má napsané vstupní slovo (kód$(T);\alpha$). Nejprve ověříme, jestli je vstupní slovo formálně v pořádku (t.j. jestli kód má všechny komponenty dané předepsaným způsobem, jestli kód(T) a α jsou odděleny předepsaným oddělovačem, atd. Pak přepíšeme α na druhou pásku, kde budeme simulovat práci $T.s\ T$. Okamžitý stav a pozice hlavy T bude zaznamenána na první pásce za vstupním slovem. Univerzální Turingův stroj na základě znalosti pozice hlavy, zjistí písmeno, které přečetl T. Na základě znalosti stavu a přečteného písmena, najde v kód(T) hodnotu přechodové funkce, změní okamžitý stav, pozici hlavy a na druhé pásce zaznamená odpovídající změnu obsahu pásky T. Tento proces pořád opakuje. Zastaví se, až když se zastaví T, a přijme, když přijme T (toto může provést, protože v kód(T) je zaznamenána i množina F)[32]. Dá se ukázat, že $\bar{L}_u$ není rekursivně spočetný, tzn. podle věty 7.16 L_u není rekursivní a podle tvrzení 7.17 není ani kontextový. Máme $L_u \in \mathcal{L}_0$, ale $L_u \notin \mathcal{L}_1$, a protože již víme $\mathcal{L}_1 \subseteq \mathcal{L}_0$ (viz strana 75), dokázali jsme konečně i poslední ostrou inkluzi $\mathcal{L}_1 \subset \mathcal{L}_0$ z Chomského hierarchie (36). Protože ne každý mi musí věřit, že $\bar{L}_u$ není rekursivně spočetný, čímž pádem by tento odstavec přišel vniveč, uvedu jiný jazyk, o kterém dokáži, že skutečně není rekursivní. Uvedený důkaz lze z modifikovat tak, abychom dostali, že $\bar{L}_u$ není rekursivně spočetný.

Uvažujme jazyk

$$L'_u = \{\alpha; \text{stroj s kódem } \alpha \text{ přijímá slovo } \alpha\}$$

Lemma 7.21 *$\bar{L}'_u$ není rekursivně spočetný.*

Důkaz. Provedeme sporem. Předpokládejme, že $\bar{L}'_u$ je rekursivně spočetný, tzn. existuje $T.s.\ T'$, který ho přijímá ($L(T') = \bar{L}'_u$). Pak $\alpha \in \bar{L}'_u$, právě když buď α není kódem žádného stroje, nebo stroj s kódem α slovo α nepřijímá. Potom ale

[32] uvedený popis předpokládá, že T je deterministický, pro nedeterministické se musí použít metoda simulace nedeterministického Turingova stroje deterministickým, viz tvrzení 7.8

T' přijímá kód(T'), právě když kód(T') $\in \bar{L}'_u$, právě když T' nepřijímá kód(T'), což je spor. CBD

Jazyk $\bar{L}'_u$ není rekursivně spočetný, neboli podle věty 7.16 L'_u není rekursivní. Lze lehce ukázat, že L'_u je rekursivně spočetný[33].

Bez důkazu si uvedeme několik jazyků a jejich zařazení:

$$\begin{aligned}
L_{\text{halt}} &= \{(\text{kód}(T);\alpha);\ \text{výpočet } T \text{ nad } \alpha \text{ se zastaví}\} \in \mathcal{L}_0 \\
&\quad \text{— není rekursivní} \\
L_{\text{h}} &= \{\text{kód}(T);\ \text{výpočet } T \text{ se zastaví pro každé vst. slovo}\} \notin \mathcal{L}_0 \\
L_{\text{reg}} &= \{\text{kód}(T); T \text{ přijímá regulární jazyk}\} \notin \mathcal{L}_0 \\
L_{\emptyset} &= \{\text{kód}(T); T \text{ přijímá prázdný jazyk}\} \notin \mathcal{L}_0 \\
L_{\text{fin}} &= \{\text{kód}(T); T \text{ přijímá konečný jazyk}\} \notin \mathcal{L}_0
\end{aligned}$$

Je-li G bezkontextová gramatika, pak

$$\begin{aligned}
L^G_{\text{all}} &= \{\text{kód}(G); L(G) = V_T^*\} \notin \mathcal{L}_0 \\
L^G_{\text{exists}} &= \{\text{kód}(G); L(G) \neq \emptyset\} \in \mathcal{L}_0 \\
L^G_{\text{reg}} &= \{\text{kód}(G); L(G) \in \mathcal{L}_3\} \notin \mathcal{L}_0 \\
L^G_{\text{fin}} &= \{\text{kód}(G); L(G) \text{ konečný}\} \notin \mathcal{L}_0
\end{aligned}$$

Postův problém: Vstupem pro Postův problém jsou posloupnosti slov $\alpha_1, \ldots, \alpha_n$ a $\beta_1, \ldots, \beta_n$, kde $\alpha_i, \beta_i \in X^*$ pro nějakou abecedu X. Postův problém má řešení, jakmile existuje posloupnost indexů $i_1, \ldots, i_r$ taková, že

$$\alpha_{i_1}\alpha_{i_2}\ldots\alpha_{i_r} = \beta_{i_1}\beta_{i_2}\ldots\beta_{i_r},$$

kde $1 \leq i_j \leq n$ pro všechna j. Jazyk

$$L_{\text{Post}} = \{\text{vstup pro Postův problém, který má řešení}\}$$

je rekursivně spočetný, ale není rekursivní. Postův problém se často využívá při zjišťování, že nějaký jazyk není rekursivní.

Tvrzení 7.22 *Rekursivně spočetné jazyky jsou uzavřeny na průnik.*

Důkaz: Vezměme dva jazyky L_1, L_2 přijímané jednopáskovými deterministickými Turingovými stroji T_1, T_2. Definujme nový dvoupáskový *T.s.* T, který na první pásce bude simulovat výpočet T_1 a na druhé T_2. Pokud jeden z výpočtů (např. T_1) skončí dříve než druhý, na odpovídající (první) pásce budou probíhat Λ-přechody ve stavu, ve kterém výpočet skončil, dokud neskončí výpočet i na druhé pásce. Když obě simulace skončí v přijímacích stavech, T vstupní slovo přijme. Myslím, že je intuitivně jasné, že $L(T) = L_1 \cap L_2$. CBD/2

[33] Intuitivně — vezmeme modifikovaný univerzální *T.s.*, který si vstupní slovo α nejprve zduplikuje na $(\alpha;\alpha)$ a pak teprve provede výpočet.

Rejstřík

Seznam vět

Seznam příkladů

Literatura

[1] M. Chytil: *Teorie automatů a formálních jazyků*

[2] M. Chytil: *Automaty a gramatiky*, SNTL, Praha 1984

[3] M. Demlová - V. Koubek: *Algebraická teorie automatů*, SNTL, Praha 1990

[4] Poznámky studentů z přednášky *Automaty a gramatiky* v letním semestru 1994/1995 na MFF UK

[5] David Karkoška: Diplomová práce *Simulace zařízení rozpoznávající regulární jazyky* (1995)

Třída	$\mathcal{L}_3$	$\mathcal{L}_2$	$\mathcal{L}_1$	$\mathcal{L}_0$
Sjednocení	ano (4.22)	ano (kap. 5.2.3)	ano (7.13)	ano (7.13)
Průnik	ano (4.22)	ne (kap. 5.2.3)	ano (7.14)	ano (7.22)
Iterace	ano (4.32)	ano (kap. 5.2.3)	ano (7.13)	ano (7.13)
Zrc. obraz	ano (4.34)	ano (kap. 5.2.3)	ano (7.13)	ano (7.13)
Doplněk	ano (4.21)	ne (kap. 5.2.3)	ano (7.14)	ne (7.21)
Konkatenace	ano (4.30)	ano (kap. 5.2.3)	ano (7.13)	ano (7.13)

Tabulka 4: Uzavřenost tříd Chomského hierarchie na operace

Třída	Název	Automaty	Gramatiky
$\mathcal{L}_3$	regulární	Akceptory dvouc. automaty (4.40)	regulární
$\mathcal{L}_2$	bezkontextové	Zásobníkové automaty (6.10)	bezkontextové
$\mathcal{L}_1$	kontextové	Lineární *T.s.* (7.12)	kontextové nezkracující (5.9)
$\mathcal{L}_0$	rekursivně spočetné (7.12)	*T.s.*	všechny

Tabulka 5: Kritéria pro zařazení jazyka do Chomského hierarchie

A Přehled jazyků podle Chomského hierarchie

Tabulka 5 ukazuje, které gramatiky generují a které automatu přijímají kterou třídu jazyků.

V tabulce 4 je uvedena uzavřenost tříd Chomského hierarchie na operace.

Tabulka 6 říká, jak určit, že nějaký jazyk (ne)patří do některé z tříd. Pokud jazyk nesplňuje uvedenou(é) nutnou(é) podmínku(y), zaručeně do dané třídy nepatří. Pokud splňuje postačující, jistě do dané třídy patří. Nutnými i postačujícími podmínkami pro všechny třídy je samozřejmě sestrojení automatu (gramatiky) z tabulky 5.

V závorkách jsou čísla lemmat, tvrzení, vět nebo důsledků, ze kterých uvedený údaj vyplývá.

Třída	Podmínka	
	nutná	postačující
$\mathcal{L}_3$	Myhill-Nerode (4.17)	Myhill-Nerode
	věta 4.24	sestrojení regulárního výrazu
$\mathcal{L}_2$	Pumping lemma (5.17)	—
$\mathcal{L}_1$	Doplněk je v $\mathcal{L}_0$ (7.16,7.17)	—
$\mathcal{L}_0$	—	—

Tabulka 6: Nutné a postačující podmínky pro zařazení jazyka