

# Automaty a gramatiky

Jason

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

<b>1</b>	<b>Základní pojmy z teorie jazyků</b>	<b>3</b>
<b>2</b>	<b>Úvod do teorie automatů</b>	<b>3</b>
2.1	Historie . . . . .	3
2.2	Automaty a jazyky . . . . .	4
2.3	Klasifikace automatů . . . . .	4
2.4	Mooreův automat . . . . .	5
2.5	Mealyho automat . . . . .	6
2.6	Zadávání automatů v praxi . . . . .	6
2.7	Ekvivalence Mooreova a Mealyho automatu . . . . .	7
<b>3</b>	<b>Propojování a dekompozice automatů</b>	<b>10</b>
3.1	Automatová a stavová kongruence . . . . .	10
3.2	Podílový automat . . . . .	12
3.3	Sériová a paralelní dekompozice . . . . .	15
<b>4</b>	<b>Regulární jazyky</b>	<b>25</b>
4.1	Akceptory . . . . .	25
4.2	Syntaktické kongruence jazyků . . . . .	31
4.3	Nedeterministické akceptory . . . . .	38
4.4	Dvoucestné automaty . . . . .	48
4.5	Substituce . . . . .	54
4.6	Kvocienty jazyků . . . . .	59
<b>5</b>	<b>Gramatiky</b>	<b>61</b>
5.1	Chomského hierarchie . . . . .	62
5.2	Bezkontextové gramatiky . . . . .	70
5.2.1	Normální formy . . . . .	70
5.2.2	Derivační stromy . . . . .	72
5.2.3	Operace nad bezkontextovými gramatikami . . . . .	75
<b>6</b>	<b>Zásobníkové automaty</b>	<b>77</b>
<b>7</b>	<b>Turingovy stroje</b>	<b>88</b>
7.1	Odvozené Turingovy stroje . . . . .	89
7.2	Univerzální Turingův stroj . . . . .	99
	<b>Rejstřík</b>	<b>102</b>
	<b>Seznam vět</b>	<b>105</b>
	<b>Seznam příkladů</b>	<b>105</b>
	<b>Literatura</b>	<b>105</b>
<b>A</b>	<b>Přehled jazyků podle Chomského hierarchie</b>	<b>106</b>

## 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  $\Lambda$
- 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  $\text{conc}(\alpha, \beta) = \alpha\beta = a_1 \dots a_m b_1 \dots b_n$  (pro  $m, n \in \mathbb{N}$ ) (jedná se o složení řetězců, resp. slov) <sup>1</sup>

**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<sup>2</sup>. 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.

Naprostou stejnou tvrzení platí samozřejmě i pro  $\mathbb{N}_0$  a  $\mathbb{Z}$  (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 významnou odvětví moderní vědy. K dalšímu rozvoji přispěli v roce

<sup>1</sup> Lehce nahlédneme, že operace skládání řetězců je asociativní,  $\Lambda$  je její jednotkový prvek, a že tedy  $A^*(\text{conc}, \Lambda)$  je monoid (dokonce volný nad množinou  $A$ ).

<sup>2</sup> značíme  $N$ , resp.  $\mathbb{N}_0$  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<sup>1</sup> lze klasifikovat (třídít) podle nejrůznějších hledisek. Mezi nejpoužívanější patří

- použitá paměť<sup>2</sup>
- způsob určení následující konfigurace<sup>3</sup>
- přístup ke vstupním datům<sup>4</sup>

<sup>1</sup>nebo akceptory, protože to je pro naše potřeby jedno a to samé. Automaty je poněkud obecnější slovo, akceptory výstižnější

<sup>2</sup>žádná, zásobník, atd.

<sup>3</sup>jak je zadán způsob přechodu automatu z jednoho stavu do druhého (pevně daný, výběr z množiny možných přechodů)

<sup>4</sup>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 šesticí  $(Q, X, \delta, q_0, Y, \beta)$ , resp. pěticí  $(Q, X, \delta, Y, \beta)$ <sup>5</sup>, kde

- $X$  označuje vstupní abecedu
- $Q$  (konečnou) množinu stavů automatu
- $\delta : Q \times X \rightarrow Q$  přechodovou funkci
- $q_0 \in Q$  počáteční stav
- $Y$  výstupní abecedu
- $\beta : Q \rightarrow Y$  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  $\delta$  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  $\beta$ . Po zapsání výstupu se hlava na výstupní pásce přesune také na další políčko.

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

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

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

pro  $q \in Q$ ,  $a \in X$  a  $\alpha \in X^*$ .

Dále definujeme překladovou funkci  $\beta_p : Q \times X^* \rightarrow Y^*$

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

<sup>5</sup>někdy nás nezajímá počáteční stav, ale práce automatu bez závislosti na něm

pro  $q \in Q$ ,  $a \in X$  a  $\alpha \in X^*$ . Překládová funkce, jak je zřejmé z definice, vrátí pro dané vstupní slovo odpovídající slovo výstupní<sup>6</sup>.

**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) indukci dle délky  $\gamma$ . 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  $(Q, X, \delta, Y, \beta)$ . Jednotlivé prvky odpovídají popisu v kapitole 2.4 (Mooreův automat), až na  $\beta$ , které je zde definováno jako zobrazení  $\beta : Q \times X \rightarrow Y$ . Výstup Mealyho automatu tedy závisí na momentálním stavu a znaku na vstupu.*

I pro Mealyho automat definujeme zobrazení  $\delta^*$  a  $\beta^*$ , a to stejně jako pro Mooreův automat.<sup>7</sup>

## 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  $\beta$ . Ostatní sloupce definují funkci  $\delta$ , 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.

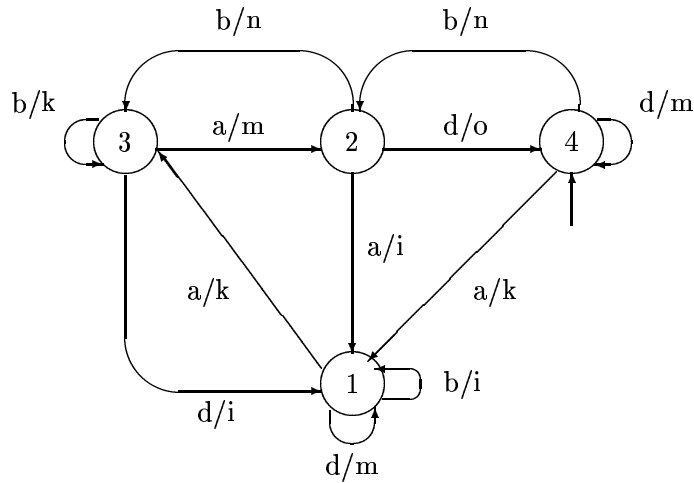
Mealyho automat se zadává podobnou tabulkou, jen poslední sloupec chybí a položky tabulky jsou tvořeny dvojicemi  $\delta/\beta$ , kde  $\delta$  je opět hodnota přechodové a  $\beta$  výstupní funkce.

Následující tabulky ukazují příklady zadání

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

<sup>6</sup>příčemž práce automatu nad slovem  $\alpha$  nemusí začínat v  $q_0$ ; překládová funkce je definována pro všechny stavy z  $Q$

<sup>7</sup>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  $o$ .<sup>8</sup>

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<sup>9</sup>  $(Q_i, X, \delta_i, Y, \beta_i)$ ,  $i = 1, 2$  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

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

<sup>8</sup>Jako cvičení by nebylo špatné zkusit si vypsát vstupní a výstupní abecedy a množiny stavů obou automatů

<sup>9</sup>ne nutně různé

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

Pro Mealyho automaty je  $\Lambda$ -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  $\beta$ , potřebná pro zavedení pojmu ekvivalence, však nad prázdným slovem v Mealyho automatu není definována. Pojem  $\Lambda$ -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 stejně chování (resp. stejně  $\Lambda$ -chování), pokud

$$\begin{aligned} (\forall q_1 \in Q_1)(\exists q_2 \in Q_2)(q_1 \sim q_2) \\ (\forall q_2 \in Q_2)(\exists q_1 \in Q_1)(q_1 \sim q_2), \end{aligned}$$

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

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

**Věta 2.6** (*Moore  $\rightarrow$  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 stejně  $\Lambda$ -chování.

Důkaz: Podle definice 2.5 je třeba najít ke každému stavu jednoho automatu  $\Lambda$ -ekvivalentní stav automatu druhého a naopak. Relace  $\Lambda$ -ekvivalence se nám zde přímo nabízí coby identita, tzn. stav  $q \in Q$  Mealyho automatu označíme za  $\Lambda$ -ekvivalentní s tímž stavem  $q$  v Mooreově a naopak. Musíme však ověřit, že identita  $id_Q$ <sup>11</sup> je zde skutečně  $\Lambda$ -ekvivalence (viz definice 2.4). Vezměme libovolný stav  $q \in Q$ , nějaké  $\alpha = \alpha'a$  pro  $\alpha' \in X^*$ ,  $a \in X$ . Z definice  $\beta^*$ , předpokladu (5), definice  $\delta^*$  a opět definice  $\beta^*$  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)  $\Lambda$ -ekvivalentní stav Mealyho automatu a naopak, tedy tyto automaty mají stejně  $\Lambda$ -chování. CBD

<sup>10</sup>Jinými slovy, dva stavy jsou ekvivalentní, pokud výpočet započatý v každém z nich nad stejným slovem dává stejný výstup

<sup>11</sup>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  $\Lambda$ -chování.*

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

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

Ověřit, že je to skutečně  $\Lambda$ -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  $\alpha$ : pro  $\alpha = a \in X$  je z definice  $\delta^*$ , předpokladu (6) a z definice  $\delta^*$  a  $\beta^*$

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

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

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

Tím jsme dokázali vztah (9) a můžeme přikročit k důkazu věty (tedy rovnice (8)). Postupným použitím definice  $\beta^*$ , 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  $\Lambda$ -ekvivalence. Podle definice  $\overset{\Lambda}{\sim}$  pro každý stav Mealyho automatu existuje alespoň jeden stav Mooreova automatu s ním  $\Lambda$ -ekvivalentní a naopak, tedy automaty popsané v předpokladech věty mají stejné  $\Lambda$ -chování. CBD

Uvedené věty nám říkají nejen, že lze Mooreův automat převést na Mealyho (a naopak) se stejným  $\Lambda$ -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<sup>12</sup>.

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

<sup>12</sup>až na to, že  $\Lambda$  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 považují za samozřejmost a poznámky pod čarou na účet  $\beta^*$  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  $\Lambda$ -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  $\beta$  předpisem (5). Výsledek je zřejmý z tabulky.

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

**FINE**<sup>13</sup>

**Příklad 2.9** (Převedení Mealy→Moore) Bud'  $M_2^e = (Q_2, X_2, \delta_2, Y_2, \beta_2)$  Mealyho automat zadaný tabulkou (viz níže). Sestrojte Mooreův automat se stejným  $\Lambda$ -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  $\delta_3$  bude definována vztahem (6) a výstupní funkce  $\beta_3$  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  $(Q, X, \delta, Y, \beta)$ , pokud platí

$$(10) \quad (\forall a \in X)(\forall q, q' \in Q)(q \approx q' \Rightarrow \delta(q, a) \approx \delta(q', a))$$

$$(11) \quad (\forall q, q' \in Q)(q \approx q' \Rightarrow \beta(q) = \beta(q'))$$

Pokud platí alespoň podmínka (10), hovoříme o stavové kongruenci<sup>1</sup>

<sup>13</sup>konec příkladu

<sup>1</sup>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  $(Q, X, \delta, Y, \beta)$ .

1. Hledání nejhrubší<sup>2</sup> automatové kongruence. Na automatu  $(Q, X, \delta, Y, \beta)$  definujeme ekvivalence  $\overset{i}{\sim}$  pro  $i \in N_0$ :

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

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

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

Jinak řečeno, stavy  $q$  a  $q'$  jsou ekvivalentní v  $\overset{i}{\sim}$ , pokud výpočty<sup>3</sup> 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<sup>4</sup> kongruence, pro kterou platí  $q \sim q'$ . Elementární stavové kongruence dostáváme opačným postupem než nejhrubší automatovou kongruencí. Vyjdeme od ekvivalence lepící právě dva prvky<sup>5</sup> a postupně vytváříme ekvivalence hrubší a hrubší. Definujme opět ekvivalenci

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

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

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

<sup>2</sup>tzn. má nejméně kongruenčních tříd

<sup>3</sup>výpočtem nad slovem  $a_1 \dots a_n$  započatém ve stavu  $q_1$  mám na mysli posloupnost stavů  $q_1, \dots, q_{n+1}$ , 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.

<sup>4</sup>tzn. má nejvíce kongruenčních tříd

<sup>5</sup>Definice: ekvivalence  $\rho$  lepí prvky  $x_1$  a  $x_2$ , právě když  $x_1 \rho x_2$

<sup>6</sup>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  $\delta$  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  $\pi, \tau \in K$ )

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

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, absorpci a idempotenci<sup>7</sup>. 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:

$$(12) \quad \delta'([q]_{\approx}, a) = [\delta(q, a)]_{\approx}$$

$$(13) \quad \beta'([q]_{\approx}) = \beta(q)$$

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  $\delta'$  a  $\beta'$  prokázána.

<sup>7</sup>Všechny čtyři vlastnosti lze ověřit přímo z definice.

**Tvrzení 3.4** *Ekvivalence z definice 2.4 na automatu  $(Q, X, \delta, Y, \beta)$  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<sup>8</sup>, 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  $\beta^*$  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<sup>9</sup>. 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 indukci podle délky  $\alpha$ . 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  $\delta^*, \delta'$ , indukčního předpokladu a definice  $\delta^*$

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

CBD

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

---

<sup>8</sup>mám na mysli řádek

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

resp. speciální případ

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

<sup>9</sup>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  $\beta^*, \beta'$  a  $\delta'$  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*

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

*se nazývá homomorfismus<sup>10</sup> 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  $M_2$ <sup>11</sup>*

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

*takový, že  $g$  je surjektivní (na).*

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

$$(\forall q \in Q_1)(g(q) \stackrel{df}{\sim} q) \quad \boxed{\text{CBD}}$$

**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  $M$ <sup>12</sup>. Pro automat  $M'$  dále platí, že je*

<sup>10</sup>Pro zájemce tu máme exkurzi do algebry: homomorfismus dvou algeber  $A(\alpha_1, \dots, \alpha_n)$ ,  $B(\beta_1, \dots, \beta_n)$  je jakékoliv zobrazení  $f : A \rightarrow B$ , které je slučitelné se všemi operacemi algeber  $A, B$ , tzn. platí

$$(\forall i \in \{1; n\})(\forall x_1, \dots, x_m \in A)(f(\alpha_i(x_1, \dots, x_m)) = \beta_i(f(x_1), \dots, f(x_m))),$$

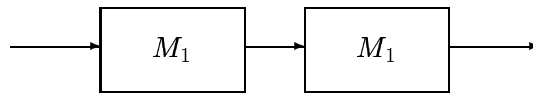
kde  $m$  je arita operací  $\alpha_i, \beta_i$ .

V našem případě jsou algebry 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  $\delta, \beta$  (viz rovnice (16), (17)).

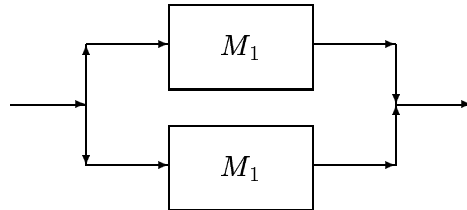
<sup>11</sup>zápis  $id_X$  značí identickou funkci na množině  $X$ , tzn.  $(\forall x \in X)(x = id_X(x))$ ; srovnejte s definicí identické relace

<sup>12</sup>což je poměrně pochopitelné

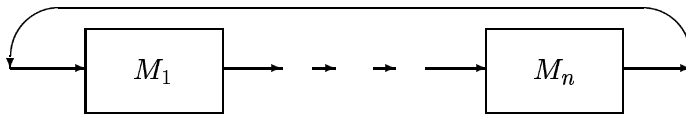
a)



b)



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<sup>13</sup>
- podílovým automatem automatu  $M$  podle kongruence  $\sim$ <sup>14</sup>

### 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).*

<sup>13</sup> 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)

<sup>14</sup>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

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

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  $f : X_2 \rightarrow X_1$  a  $g : Q_2 \rightarrow Q_1$  a zobrazení  $h : Y_1 \rightarrow Y_2$  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  $M_1(\implies)M_2$  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  $a \in X, q_1 \in Q_1$  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 ekvivalenci 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  $M' = M_1(\implies)M_2$ <sup>15</sup>, tedy existují  $f, g, h$  taková, že platí (20) a (21) (neboť  $M'$  realizuje  $M$ ), a dále platí (18), neboť  $M'$  je sériové spojení<sup>16</sup>. Definujme ekvivalenci  $\rho \subseteq Q \times Q$  následovně:

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

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

<sup>16</sup>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  $\rho$ , jakmile se první složky  $g(q_1)$  a  $g(q_2)$  rovnají). O relaci  $\rho$  ukážeme, že je netriviální stavovou kongruencí, tzn. ověříme podmínku (10) a potom netriviálnost  $\rho$ . 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

$$\begin{aligned} g(\delta(q_1, a)) &= \delta'(g(q_1), f(a)) = (\delta_1(r_1, f(a)), \delta_2(s_1, \beta_1(r_1, f(a)))) \\ &= (r'_1, s'_1) \\ g(\delta(q_2, a)) &= \delta'(g(q_2), f(a)) = (\delta_1(r_2, f(a)), \delta_2(s_2, \beta_1(r_2, f(a)))) \\ &= (r'_2, s'_2), \end{aligned}$$

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  $\rho$  je stavová kongruence. Protože dva stavy  $q_1, q_2 \in Q$  jsou v  $\rho$ , právě když se první složky funkce  $g : Q \rightarrow Q_1 \times Q_2$  na těchto stavech rovnají, vytváří  $\rho$  nejvýše tolik ekvivalenčních tříd, kolik je stavů v  $Q_1$ . Protože podle definice sériové dekompozice  $|Q_1| < |Q|$ , má  $\rho$  méně tříd než  $|Q|$ . Kdyby  $\rho$  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  $\rho$ .  $\rho$  je tedy netriviální.

Ale teď...<sup>17</sup>

Nechť nyní  $M$  má netriviální stavovou kongruenci  $\rho$ . Označíme-li  $k$  maximální velikost tříd kongruence  $\rho$ , je  $k < |Q|$ . Předpokládejme, že pro každou třídu  $V$  jsou její prvky seřazeny, tedy  $V = (q_{1,V}, q_{2,V}, \dots, q_{|V|,V})$ . Definujme automaty  $M_1$  a  $M_2$  následovně:<sup>18</sup>  $M_1 = (Q/\rho, X, \delta_1, Y_1, \beta_1)$ , nechť je „podílovým“ automatem<sup>19</sup> 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

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

<sup>17</sup>druhá implikace

<sup>18</sup>Idea definice  $M_1, M_2$  není nepochopitelná. Oba automaty kódují ve svých stavech stav z  $M$ , přičemž  $M_1$  uchovává třídu ekvivalence  $\rho$ , ve které se stav z  $M$  vyskytuje, a  $M_2$  kóduje jeho pořadí v této třídě.

<sup>19</sup>spíš je podílovému automatu představou blízký, protože  $\rho$  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  $M_1 (\implies) M_2 = (Q_1 \times Q_2, X, \delta', Y, \beta')$  je sériovou dekompozicí  $M$ , tedy v první řadě, že vůbec  $M$  realizuje (tzn. najít prostá zobrazení  $f : X \rightarrow X$  a  $g : Q \rightarrow Q_1 \times Q_2$  a zobrazení  $h : Y \rightarrow Y$  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  $\rho$ , obsahující stav  $q_{i,V}$  na pozici  $i$ . Důkaz vztahu (21) plyne z definice  $\beta_2, \beta_1$ , 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<sup>20</sup>

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

Z definice  $\delta_1$  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  $\delta_2$ , 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  $\rho$  je netriviální, plyne  $|Q_1| < |Q|$ . Protože  $k = |Q_2|$  a  $\rho$  je netriviální, víme  $|Q_2| < |Q|$ , tedy  $M_1 (\implies) M_2$  je sériovou dekompozicí  $M$ .<sup>21</sup>

**CBD**

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

Důkaz: Dokažme implikaci zleva doprava. Předpokládejme, že  $M' = M_1 \parallel M_2$  realizuje  $M$ <sup>23</sup> a zobrazení  $f : X \rightarrow X_1, g : Q \rightarrow Q_1 \times Q_2, h : Y_1 \times Y_2 \rightarrow Y$  ( $f, g$  prostá) tuto realizaci určují. Definujme ekvivalence  $\rho_1$  a  $\rho_2$ . 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

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

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

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

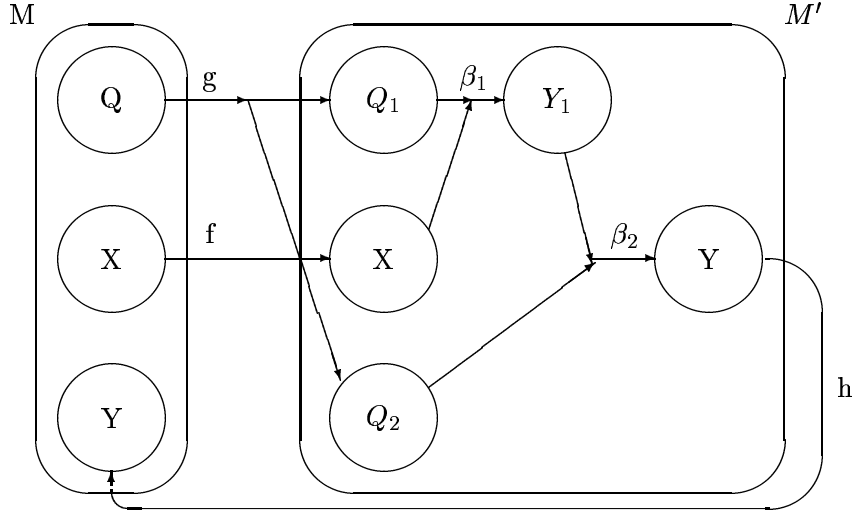
$$\begin{aligned} g(q_1) &= (r_1, s_1) \\ g(q_2) &= (r_2, s_2) \end{aligned}$$

<sup>20</sup>první výraz lze odvodit z definice  $g$  a definice sériového spojení, druhý je triviální

<sup>21</sup>zde je dobré si povšimnout, že  $M_1$  i  $M_2$  mají nejméně dva stavy

<sup>22</sup>nebo obojí

<sup>23</sup> $M' = (Q_1 \times Q_2, X, \delta', Y_1 \times Y_2, \beta')$ ,  $M_i = (Q_i, X, \delta_i, Y_i, \beta_i)$ ,  $M = (Q, X, \delta, Y, \beta)$



Obr. 4: Schéma realizujících funkcí sériové dekompozice

Z definice realizace a paralelního spojení platí

$$\begin{aligned} g(\delta(q_1, a)) &= \delta'(g(q_1), f(a)) = (\delta_1(r_1, f(a)), \delta_2(s_1, f(a))) = (r'_1, s'_1) \\ g(\delta(q_2, a)) &= \delta'(g(q_2), f(a)) = (\delta_1(r_2, f(a)), \delta_2(s_2, f(a))) = (r'_2, s'_2) \end{aligned}$$

Z uvedených rovností podle definice  $\rho_1$  a  $\rho_2$  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  $\rho_1$  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.  $\rho_2$  je stavová kongruence

Všimněme si nejprve, že  $\rho_1$  má nejvýše  $|Q_1|$  tříd a  $\rho_2$  má nejvýše  $|Q_2|$  tříd. Protože  $M'$  je paralelní dekompozice, platí  $|Q_1| < |Q|$  a  $|Q_2| < |Q|$ , tedy  $\rho_1$  a  $\rho_2$  nejsou identické kongruence. Dokážeme sporem, že  $\rho_1$  má více než jednu třídu. Kdyby  $\rho_1$  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, že  $\rho_2$  nelepí všechny prvky  $Q$ . Ekvivalence  $\rho_1$  a  $\rho_2$  jsou tedy netriviální stavové kongruence. Nyní je ještě třeba ověřit, že  $\rho_1 \cap \rho_2 = id_Q$ , tzn. dokázat implikaci

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

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

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

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

Dokázali jsme, že relace  $\rho_1$  a  $\rho_2$ , jak jsme je definovali na začátku této části důkazu, vyhovují podmínkám věty.

Nyní dokážeme druhou implikaci. Předpokládejme, že  $\rho_1$  a  $\rho_2$  jsou netriviální stavové kongruence automatu  $M$ , pro které navíc platí

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

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  $f, g, h$ <sup>24</sup> 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

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

Představou jsou  $M_1, M_2$  blízké podílovým automatům automatu  $M$  podle  $\rho_1$  a  $\rho_2$ . Dále definujme zobrazení  $h : Y_1 \times Y_2 \rightarrow Y$ ,  $f : X \rightarrow X$ ,  $g : Q \rightarrow Q_1 \times Q_2$  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

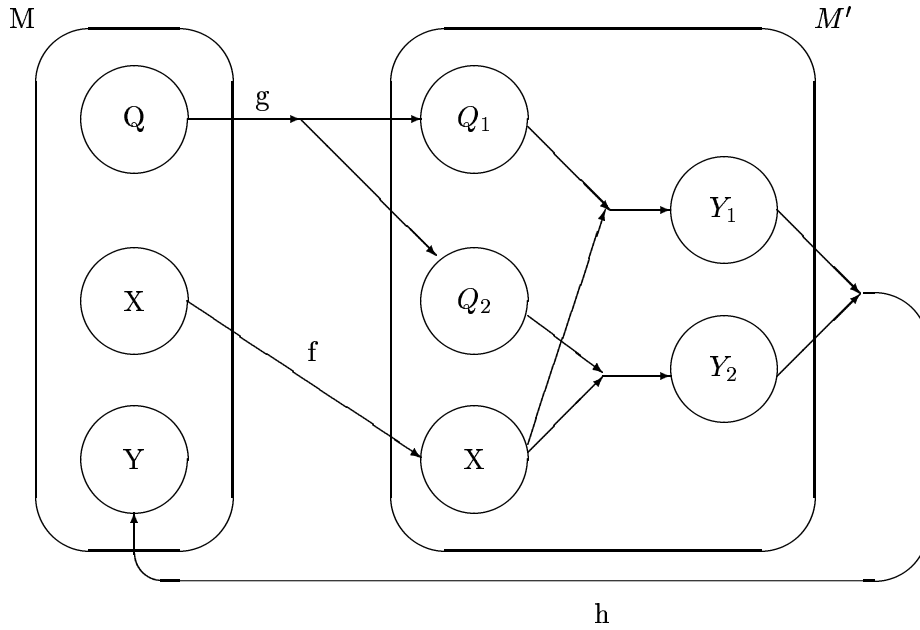
$$\begin{aligned} (q_1 \neq q_2) &\Rightarrow ((q_1, q_2) \notin \rho_1 \vee (q_1, q_2) \notin \rho_2) \Rightarrow \\ &\Rightarrow (([q_1]_{\rho_1} \neq [q_2]_{\rho_1}) \vee ([q_1]_{\rho_2} \neq [q_2]_{\rho_2})) \Rightarrow g(q_1) \neq g(q_2), \end{aligned}$$

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  $\rho_1$  a  $\rho_2$ . 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  $g, f, \delta', \delta_1, \delta_2$  a opět  $g$

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

<sup>24</sup>značení zobrazení zde odpovídá definici realizace, definice funkcí  $\delta'$  a  $\beta'$  viz (22) a (23)



Obr. 5: Přejechody realizujících funkcí paralelní dekompozice

Vztah (21) platí podobně z definic  $g, f, \beta', \beta_1, \beta_2$  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  $M$ .<sup>25</sup> 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

<sup>25</sup> 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  $\delta$  jsou ekvivalentní v  $\overset{0}{\sim}$

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

Postupně dostáváme

$$\overset{2}{\sim} = \{\{A, F\}, \{B, D\}, \{E\}, \{C\}\}$$

$$\overset{3}{\sim} = \{\{A, F\}, \{B\}, \{D\}, \{E\}, \{C\}\}$$

$$\overset{4}{\sim} = \{\{A, F\}, \{B\}, \{D\}, \{E\}, \{C\}\}$$

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 přidělovat práci a dělat víc, než se v zadání požaduje, v rámci lepšího zažití konstrukce stavových kongruencí uvedu některé další:

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

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

$$\overset{2}{\sim} = \{\{A, B, C, D, E\}, \{F\}\}$$

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

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

$$\overset{2}{\sim} = \{\{A, B, C, F\}, \{D\}, \{E\}\}$$

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  $T_1, \dots, T_5$ , 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  $X = \{0, 1\}$ ,  $Y = \{0, 1\}$ . Přejchodovou a výstupní funkci  $\delta'$  a  $\beta'$  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

$$Q = \{1, 2, 3, 4, 5, 6\}$$

$$X = \{a, b, c\}$$

$$Y = \{u, y, z\}$$

Sestrojíme nejprve sériovou dekompozici podle postupu, který nám nabízí důkaz věty 3.16. Nejprve budeme potřebovat jednu stavovou kongruenci (označme ji  $\rho_1$  automatu  $M$ ). Podle algoritmu 1 volbou  $1 \stackrel{0}{\sim} 2$  dostáváme

$$\stackrel{0}{\sim} = \{\{1, 2\}, \{3\}, \{4\}, \{5\}, \{6\}\}$$

$$\stackrel{1}{\sim} = \{\{1, 2\}, \{3, 4\}, \{5, 6\}\}$$

$$\stackrel{2}{\sim} = \{\{1, 2\}, \{3, 4\}, \{5, 6\}\}$$

Algoritmus se zastavil a my jsme obdrželi stavovou kongruenci

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

Kongruence  $\rho_1$  a volba pořadí prvků v jednotlivých třídách kongruence  $\rho_1$ <sup>26</sup> 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  $M' = M_1(\implies)M_2$  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  $\rho_1$  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  $\delta$  zobrazuje  $i$ -tý prvek třídy  $T$  na  $j$ -tý prvek třídy  $S = \delta_1(T, x)$ <sup>27</sup>, pro  $i, j \in Q_2, T, S \in Q_1, x \in X$
- $\beta_1$  je identická funkce
- $\beta_2(i, (T, x)) = y$ , když výstup funkce  $\beta$  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 získáme 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  $\rho_2$  takovou, aby bylo splněno  $\rho_1 \cap \rho_2 = id_Q$ . Vezměme například<sup>28</sup>

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

<sup>26</sup>v našem případě je pořadí dáno zápisem (29)

<sup>27</sup>třída  $S$  nás sama o sobě nezajímá, pouze pořadí prvků v ní

<sup>28</sup>algoritmus pro získání  $\rho_2$  nastartujeme volbou  $1 \stackrel{0}{\sim} 3$



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)

- $X_3 = X$
- $Q_3 = Q/\rho_2 = \{S_1, S_2\}$
- $Y_3 = Q_3 \times X_3$
- $\delta_3$  a  $\beta_3$  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
- $g' : Q \rightarrow Q_1 \times Q_3$ ;  $g'(q) = (T, S)$ ,  
kde  $q \in T \in Q_1, q \in S \in Q_2, q \in T \cap S$
- $h' : Y_1 \times Y_3 \rightarrow Y$ ;  $h'((T, x), (S, x)) = \beta(q, x)$ ,  
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  $h((T, x), (S, x')) = \text{lib. prvek } Y$  a do tabulky nezapíšeme):<sup>29</sup>

$q$	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  $\alpha$  rozhodne o tom, zda  $\alpha$  patří do  $L$  či nikoliv, nazveme *akceptorem*. Jedná se

<sup>29</sup>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
- $\delta : Q \times X \rightarrow Q$  je přechodová funkce
- $q_0 \in Q$  je iniciální (nebo také počáteční) stav
- $F \subseteq Q$  je množina koncových (nebo také přijímacích) stavů<sup>1</sup>

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

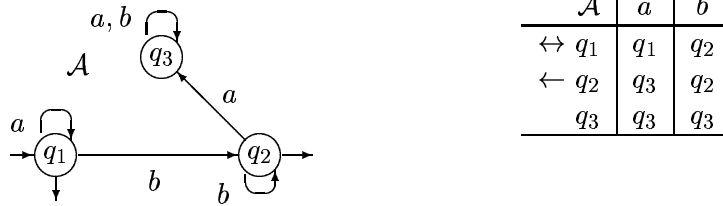
**Poznámka:** Podobně jako Mooreovy a Mealyho automaty, i akceptory mohou být zadány graficky, tabulkou funkce  $\delta$ , případně slovně nebo jiným vhodným způsobem. Vyhnou se složitým formálním popisům zadání akceptoru. Raději uvedu příklad, ze kterého by mělo být všechno jasné.

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

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

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.

<sup>1</sup>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  $(Q, X, \delta, q_0, F)$  pouze formálním zápisem, pokud platí  $F = \{q; \beta(q) = true\}$



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

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

**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 přijímací, neboť funkce  $\delta$  musí být definovaná pro každou dvojici z  $Q \times X$ , tzn. akceptor musí v každém stavu znát pokračování přes každé písmeno vstupní abecedy<sup>3</sup>.

**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.  $Q_2 = \{q \in Q_1; q \text{ je dosažitelný v } \mathcal{A}_1\}$
2.  $\delta_2 : Q_2 \times X \rightarrow Q_2$  je restrikce (zúžení)  $\delta_1$  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  $\alpha$  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  $\mathcal{A}$  jsou ekvivalentní (píšeme  $q_1 \sim q_2$ ), když pro všechna  $\alpha \in X^*$  platí*

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

<sup>2</sup>vstupní abeceda, množina stavů, iniciální stav (označen šipkou dovnitř), přijímací stavy (šipka ven), přechodová funkce

<sup>3</sup>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  $\delta^*$  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<sup>4</sup>:

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

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  $\delta^*$  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  $F$ <sup>5</sup>.

**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

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

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

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

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

<sup>4</sup>že tato formule je ekvivalentní s (11), ponechávám čtenáři na rozvážení

<sup>5</sup>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 redukováný, jakmile  $\sim$  je automatová kongruence z definice 4.6<sup>6</sup>.

**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. Bijekcí<sup>7</sup>  $g : Q_1 \rightarrow Q_2$  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 izomorfni“ 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 redukováný 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 je-li  $Q' = Q \setminus M$ ,  $F' = F \cap Q'$  a  $\delta' : Q' \times X \rightarrow Q'$  zúžení  $\delta$  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  $\mathcal{A}'/\sim$ <sup>8</sup> je ekvivalentní s  $\mathcal{A}'$ , tedy i s  $\mathcal{A}$ . Navíc  $\mathcal{A}'/\sim$  je redukováný<sup>9</sup> a jednoduše dostaneme, že  $\mathcal{A}'/\sim$  je dosahující, protože  $\mathcal{A}'$  je dosahující, tudíž jsme k  $\mathcal{A}$  našli požadovaný redukováný dosahující akceptor.

CBD

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

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 nenáročni, požadujeme pouze, aby byly ekvivalentní, redukované a dosahující. Dokážeme, že jsou izomorfni, tzn. najdeme bijekci  $g : Q_1 \rightarrow Q_2$  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:

<sup>6</sup>viz věta 3.6

<sup>7</sup>vzájemně jednoznačné zobrazení, tzn. zobrazení *prosté a na*

<sup>8</sup> $\sim$  viz definice 4.6

<sup>9</sup>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  $\gamma$ , 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  $\alpha$  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  $\delta^*$  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 injektivit  $g^{10}$ . 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  $\delta^*$ ,  $q, q'$  pro libovolné  $\gamma \in X^*$  platí

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

Dále z  $g(q) = g(q')$  platí

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

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.*<sup>11</sup>

## 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

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

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

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

<sup>10</sup>tzn. budeme dokazovat, že  $g$  je prosté

<sup>11</sup>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) kongruence můžeme samozřejmě definovat i levé (syntaktické) kongruence. V dalším textu s nimi však pracovat nebudeme, neobtěžují 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  $\gamma_1, \gamma_2, \delta_1, \delta_2 \in X^*$

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

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

$$(\forall \alpha, \beta \in X^*)(\alpha \sim \beta \implies (\alpha \in L \iff \beta \in L)).$$

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:

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

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  $\delta$ , tzn. dokázat implikaci

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



kteřá 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 redukováný a podle důsledku 4.13 má ze všech akceptorů<sup>12</sup>, 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<sup>13</sup>. Pak pro všechna  $\gamma \in X^*$  platí

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

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 redukováný.

1  $\Rightarrow$  3. Máme konečný redukováný 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  $|Q|^{|Q|}$ <sup>14</sup>, tedy konečné číslo. Zbývá dokázat, že  $\approx$  je syntaktickou kongruencí jazyka  $L(\mathcal{A})$ . Vezměme

<sup>12</sup>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!!!

<sup>13</sup>ve smyslu definice 4.6

<sup>14</sup>Označme  $n = |Q|$ . Když pro slovo  $\alpha \in X^*$  definujeme zobrazení  $f_\alpha : Q \rightarrow Q$  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  $\delta^*$  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  $\alpha, \beta \in X^*$

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

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  $q_1 \sim q_2$ <sup>15</sup>, neboli problém jsme opět převedli, tentokrát na

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

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

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

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

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

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

$$\begin{aligned} L_1 &= \{a^i b^j; i, j = 0, 1, \dots\} \\ L_2 &= \{a^i b^i; i = 0, 1, \dots\} \end{aligned}$$

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  $\sigma_{ij} = a^i a b b^j$  jsou mezi sebou kongruentní, protože vezmeme-li  $\sigma_{ij}$  a  $\sigma_{kl}$ , potom

<sup>15</sup>ve smyslu definice 4.6

- přidáme-li oběma slovům suffix<sup>16</sup> 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  $\sigma_{ij}$  pro  $i, j = 0, 1, \dots$  jsou tedy mezi sebou navzájem kongruentní v  $L_1$  a mohou tvořit třídu  $\sigma$  hledané syntaktické kongruence  $\sim_{L_1}$ . Obdobně se ukáže kongruence všech slov typu

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

pro  $i, k_a, k_b, k, \ell_a, \ell_b, \ell \geq 0, j > 0$ <sup>17</sup> (tedy  $\alpha, \beta, \gamma$  jsou další množiny, které budou kandidovat na třídy v  $\sim_{L_1}$ ). Zbývá dokázat, že  $\alpha, \beta, \gamma, \sigma$  jsou po dvou různé. Třída  $\gamma$  se vyznačuje tím, že přidáním jakéhokoliv prefixu nebo suffixu k jejím slovům vytvoříme slova, která nepatří do  $L_1$ . Slova  $\alpha_i$  nejsou kongruentní se slovy  $\beta_j$  (stačí volit suffix  $a$ )<sup>18</sup>, 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  $j, k \leq i, j \neq k, i$  pevně<sup>19</sup>, 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<sup>20</sup>.

Z Myhill-Nerodovy věty (4.17) vyplývá, že jazyk  $L_1$  je regulární, kdežto  $L_2$  regulární není<sup>21</sup>. FINE

Nyní podáme charakterizaci syntaktické kongruence pomocí jazyka.

**Tvrzení 4.19** *Bud'  $L \subseteq X^*$  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*

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

<sup>16</sup>suffixem délky  $n$  slova  $\alpha$  se rozumí posledních  $n$  znaků slova  $\alpha$ ; podobně prefix délky  $n$  je prvních  $n$  znaků; vlastní prefix (suffix) slova  $\alpha$  je prefix (suffix) délky menší než  $|\alpha|$  a různý od  $\Lambda$ .

<sup>17</sup>povšimněte si, že prázdné slovo  $\Lambda$  patří právě do  $\alpha_i$

<sup>18</sup> $\alpha_i a \in L_1, \beta_j a \notin L_1$

<sup>19</sup>leč libovolné

<sup>20</sup>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  $L_2$ .

<sup>21</sup>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  $\sim$ <sup>22</sup>. Tvrzení je pro syntaktické kongruence dokázané.

Důkaz pro pravé syntaktické kongruence je analogický. CBD

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

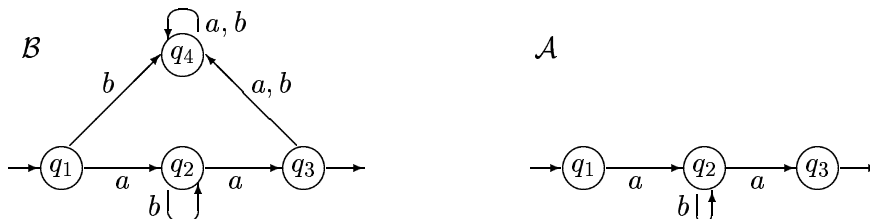
**Lemma 4.21** *Regulární jazyky jsou uzavřeny na doplňky.*

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

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

Odlíšně 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  $\delta$  v  $\mathcal{A}$ .

**Poučení:** Používejte garbage stavy a čtěte řádně definice<sup>23</sup>!

<sup>22</sup>tím jsme se vyhnuli důkazu druhé implikace

<sup>23</sup>aby jste 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  $\alpha$  nechť skončí ve stavu  $(q, q')$ . Lze snadno nahlédnout, že  $\alpha$  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  $\alpha$  jen tehdy, pokud  $\alpha$  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  $\alpha$ , pokud přijme  $\alpha$  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 indukci 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  $\alpha$  musí projít některými stavy vícekrát. Nechť

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

je přijímacím výpočtem nad slovem  $\alpha$ , 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

$$\begin{aligned} \delta^*(q_0, \alpha_1) &= q_i = q_j = \delta^*(q_i, \alpha_2) \\ \delta^*(q_j, \alpha_3) &= q_h \in F. \end{aligned}$$

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  $n \in \mathbb{N}$  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!!!<sup>24</sup>

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

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

*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  $\alpha$  délky větší než  $n$  lze rozložit na  $\alpha = \alpha_1 \alpha_2 \alpha_3$  ( $0 < |\alpha_2| \leq n$ ) tak, že  $(\forall i \in \mathbb{N}_0)(\alpha_1 \alpha_2^i \alpha_3 \in L)$ . Proč si nevzít třeba  $\alpha_0 = a^{n+1} b^{n+1}$ . Jak může vypadat rozklad slova  $\alpha_0$ ? 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  $\alpha_0$  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  $(Q, X, \delta, I, F)$ , kde*

- $X$  vstupní abeceda
- $Q$  je konečná množina stavů
- $\delta : Q \times X \rightarrow \mathcal{P}(Q)$  přechodová funkce<sup>25</sup>
- $I \subseteq Q$  množina iniciálních stavů

<sup>24</sup>Takovým neregulárním jazykem je například

$$\{a_1 \dots a_n \in \{0, 1\}^*; a_1 \dots a_n = a_n \dots a_1; n \in \mathbb{N}_0\},$$

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.

<sup>25</sup>někdy je  $\delta$  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<sup>26</sup>.

Dále si povšimněme, že obrazem funkce  $\delta$  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  $\alpha$ . Pokud posloupnost  $r_0, r_1, \dots, r_n$  splňuje alespoň první dvě podmínky, mluvíme o ní jako o výpočtu akceptoru  $\mathcal{A}$  nad  $\alpha$ .

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

**Tvrzení 4.28** *Nedeterministické akceptory přijímají právě regulární jazyky.*

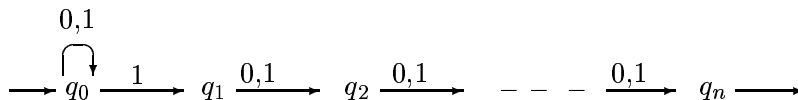
Důkaz: Každý regulární jazyk je přijímán deterministickým akceptorem (viz definice 4.2) a každý D-akceptor je zároveň nedeterministický. Zbývá dokázat implikace zleva doprava, totiž že pro libovolný N-akceptor  $\mathcal{A} = (Q, X, \delta, I, F)$  je  $L(\mathcal{A})$  regulární jazyk. Jinými slovy, 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

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

<sup>26</sup>vzhledem k neskladnosti slov (ne)deterministický budeme napříště používat zkratk N-akceptor=nedeterministický akceptor, D-akceptor=deterministický akceptor

Obr. 6: N-akceptor přijímající  $L_n$ 

Akceptor  $\mathcal{A}'$  je nesporně deterministický<sup>27</sup> a konečný a přijímá stejný jazyk jako  $\mathcal{A}$ <sup>28</sup>, 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<sup>29</sup>. 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  $\alpha = a_1, \dots, a_k, \beta = b_1, \dots, b_\ell \in \{0, 1\}^*$  a jejich suffixy  $\gamma_\alpha, \gamma_\beta$  délky  $n$ <sup>30</sup>. Dokážeme ekvivalenci

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

To je snadné. Je-li  $\gamma_\alpha = \gamma_\beta$ , pak se posledních  $n$  písmen slov  $\alpha, \beta$  shoduje. Přidáme-li za obě slova  $\alpha, \beta$  suffix  $\sigma$  libovolné délky,  $n$ -tá písmena slov  $\alpha, \beta$  od konce se zřejmě shodují, neboli

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

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  $\sigma$  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  $L_3$ <sup>31</sup>.

**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.*

<sup>27</sup>pro každou množinu  $A \subseteq Q$  je jednoznačně definována přechodová funkce

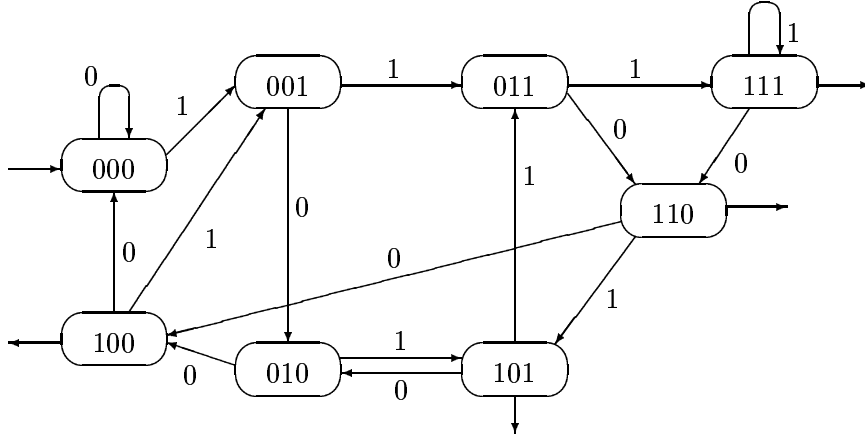
<sup>28</sup>přijímací výpočet v  $\mathcal{A}$  je vlastně transversálou přijímacího výpočtu v  $\mathcal{A}'$

<sup>29</sup>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.

<sup>30</sup>Jsou-li  $\alpha$  nebo  $\beta$  kratší než  $n$ , doplníme je zpredu písmenem 0.

<sup>31</sup>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  $\delta$  je pro všechna  $x \in X$  definována následovně:

$$\begin{aligned} (\forall q \in Q_2)(\delta(q, x) &= \delta_2(q, x)) \\ (\forall q \in Q_1)(\delta_1(q, x) \cap F_1 &= \emptyset \Rightarrow \delta(q, x) = \delta_1(q, x)) \\ (\forall q \in Q_1)(\delta_1(q, x) \cap F_1 &\neq \emptyset \Rightarrow \delta(q, x) = \delta_1(q, x) \cup I_2) \end{aligned}$$

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

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

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  $R = Q_2$ <sup>32</sup>,  $q_0 \in I_2$  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  $\delta$  pouze v části  $Q_1$ , druhá pouze v  $Q_2$ . Přejít 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  $\gamma$ , tzn.  $\beta = a_{i+1}, \dots, a_k$  je přijímáno v  $\mathcal{A}_2$ , neboť výpočet nad  $\beta$  začal v  $r_i \in I_2$ , procházel pouze stavy  $Q_2$  a skončil v množině  $F_2$ <sup>33</sup>. Pro  $\gamma$  jsme našli rozklad  $\alpha\beta$  takový, že  $\alpha \in L_1, \beta \in L_2$ ,

<sup>32</sup>tzn. do výpočtu se žádný stav z  $Q_1$  nezapojí.

<sup>33</sup>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  $L(\mathcal{A}) \subseteq L_1L_2$ .

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  $\beta$  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  $\alpha$  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  $\beta$  v  $\mathcal{A}_2$ . Potom  $r_0, \dots, r_{k-1}, r_k, \dots, r_{k+\ell}$  je z definice  $\delta$  přijímacím výpočtem slova  $\alpha\beta$  v  $\mathcal{A}$ <sup>34</sup>. Dokázali jsme  $L_1L_2 \subseteq L(\mathcal{A})$ , což s opačnou inkluzí z první části důkazu dává kýženou rovnost  $L(\mathcal{A}) = L_1L_2$ . Neboť  $\mathcal{A}$  je nesporně konečný N-akceptor, je  $L_1L_2$  regulární jazyk. CBD

Když teď máme operaci konkatence, 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  $L^* = \bigcup_{i \in \mathbb{N}_0} L^i$ . 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<sup>35</sup>.

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  $\alpha$  je přijímáno v  $\mathcal{A}_1$  stavem  $s$ <sup>36</sup>. Je-li  $\alpha = \alpha_1\alpha_2 \dots \alpha_k$ ,  $\alpha_i \in L$ , pak pro každé  $\alpha_i$  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  $\delta_1$  je  $r_{11}, r_{12}, \dots, r_{1(n_1-1)}, r_{21}, \dots, r_{kn_k}$  přijímacím výpočtem  $\alpha$ , 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  $r_{j+1} \in \delta_1(r_j, a_{j+1})$ ,  $r_0 \in I$ ,

<sup>34</sup>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  $\delta$  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  $\alpha\beta$

<sup>35</sup>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.

<sup>36</sup>od toho tam ten stav taky máme, že ...

$r_n \in F$ . Vezměme všechny indexy  $i_1 < \dots < i_k$ , pro něž  $r_{i_j+1} \notin \delta(r_{i_j}, a_{i_j+1})$ , dodefinujeme  $i_0 = 0, i_{k+1} = n$  a rozdělme příjmací výpočet slova  $\alpha$  na  $k+1$  částí ( $j$ -tá část obsahuje stavy  $r_{i_j+1}, r_{i_j+2}, \dots, r_{i_{j+1}-1}; j \in \langle 0, k \rangle$ ). Přejechy v rámci jedné části vždy patří do funkce  $\delta$ , přechody mezi jednotlivými částmi nikoli. Musely tedy z definice  $\delta_1$  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  $\delta_1$ . Potom ale posloupnosti  $r_{i_j}, r_{i_j+1}, \dots, r_{i_{j+1}-1}, r'_{i_{j+1}}$  jsou<sup>37</sup> pro  $j \in \langle 0, k \rangle$  příjmacími výpočty akceptoru  $\mathcal{A}$  nad slovy

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

Slovo  $\alpha$  jsme rozložili na  $\alpha_1 \dots \alpha_{k+1}$ ,  $\alpha_i \in L(\mathcal{A}) = L$ , 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  $\alpha = a_1 \dots a_n$  rozumíme slovo

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

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<sup>38</sup> jazyků, která*

1. *obsahuje všechny konečné jazyky<sup>39</sup>*
2. *je uzavřena na konečnou sjednocení, konkatenaci a iteraci*

<sup>37</sup>při definici  $r'_n = r_n$

<sup>38</sup>ve smyslu teorie množin

<sup>39</sup>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 lemat 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  $0, \dots, n-1$ <sup>40</sup>. 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

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

Řečeno lidsky, v  $L_{ij}^k$  jsou všechna slova, která převedou  $q_i$  na  $q_j$  přes stavy  $q_\ell$  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. ( $k \rightarrow k+1$ ): 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  $\alpha'$  libovolné vlastní prefixy  $\alpha$  platí  $\delta^*(q_i, \alpha') = q_\ell$ , kde  $\ell < k < k+1$ , tedy  $\alpha \in L_{ij}^{k+1}$ ,
- ii. nebo  $\alpha = \beta\alpha_1 \dots \alpha_m\gamma$ , kde  $\beta \in L_{ik}^k$ ,  $\alpha_i \in L_{kk}^k$ ,  $\gamma \in L_{kj}^k$ . Je-li  $\alpha'$  vlastní prefix  $\alpha$ , může nastat právě jeden z případů
  - A.  $\alpha'$  je vlastní prefix  $\beta$ , 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.  $\alpha' = \beta\alpha_1 \dots \alpha_j\alpha''$ , kde  $\alpha''$  je vlastní prefix  $\alpha_{j+1}$ . Potom
$$\delta^*(q_i, \alpha') = \delta^*(\delta^*(q_i, \beta), \alpha_1 \dots \alpha_j\alpha'') =$$

$$\delta^*(q_k, \alpha_1 \dots \alpha_j\alpha'') = \dots =$$

$$\delta^*(q_k, \alpha'') = q_\ell, \ell < k < k+1.$$
  - D.  $\alpha' = \beta\alpha_1 \dots \alpha_j$ , pak  $\delta^*(q_i, \alpha') = \delta^*(q_k, \alpha_j) = q_k$ ,  $k < k+1$ .

<sup>40</sup>tedy  $Q = \{q_0, q_1, \dots, q_{n-1}\}$

E.  $\alpha' = \beta\alpha_1 \dots \alpha_m\gamma'$ ,  $\gamma'$  vlastní prefix  $\gamma$ . Potom platí

$$\delta^*(q_i, \alpha') = \delta^*(q_k, \gamma') = q_\ell,$$

kde  $\ell < k < k + 1$ .

Slovo  $\alpha$  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  $\alpha_s$  prefix  $\alpha$  délky  $s$ . Zřejmě můžeme najít posloupnost všech indexů  $s_1 < s_2 < \dots < s_t$  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  $\alpha = \beta\sigma_1 \dots \sigma_{t-1}\gamma$ , 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 indukci 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  $L$  sestojitelný zmíněnými operacemi z elementárních jazyků. Protože kromě regularity na jazyk  $L$  požadavků nižších kladeno nebylo, lze říci, že každý regulární jazyk lze složit z elementárních jazyků operacemi konečného sjednocení, iterací a konkatencí. CBD

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

**Definice 4.36** Regulární výraz je definován indukci

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

Každému regulárnímu výrazu  $\alpha$  odpovídá jazyk nad abecedou proměnných, který označíme  $L(\alpha)$ . Operaci  $+$  odpovídá sjednocení,  $\cdot$  konkatence 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  $\mathcal{A}$  sestrojít regulární výraz popisující jazyk  $L(\mathcal{A})$ . Stačí pro všechna  $q_j \in F$  sestrojít indukci 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*

$$(34) \quad v = a^* \cdot (b \cdot c)^* + b \cdot (a \cdot c^*)^*$$

Výrazu  $v$  odpovídá regulární jazyk  $L(v)$  nad abecedou  $X = \{a, b, c\}$ . K popisu  $L(v)$  použijeme formuli (34), kde  $+$  nahradíme sjednocením a  $\cdot$  konkatencí. Sekvence  $x^*$  se nahradí sekvencemi  $x^i, i = 0, 1, \dots$ <sup>41</sup>. 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 konkatence se zpravidla  $\cdot$  nahrazuje operací konkatence 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 sestrojít akceptor, který by přijímal jemu odpovídající regulární jazyk.

**Algoritmus 2** (*Konstrukce automatů pro regulární výrazy*)

Řekněme, že máme dány regulární výrazy  $\alpha_1, \dots, \alpha_n$  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 popsáný výrazem  $\alpha_i$ . Obecný postup posléze ozřejmíme na příkladu:

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

2. Vytvoříme množiny<sup>42</sup>

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

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

<sup>41</sup>iterace jednoprvkového jazyka  $\{x\}$  jsou všechny posloupnosti  $x^i$  pro  $i \in N_0$ ;  $x$  značí libovolné písmeno z  $X$

<sup>42</sup>pro  $y, z \in X'$

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

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

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

- 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  $\alpha, \beta$ , 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<sup>43</sup>.

<sup>43</sup>Nehledě na to, že zmíněný akceptor byl již sestaven 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 řídicí jednotky (počtu stavů), ale neumožní zpracovávat více informací<sup>44</sup>. 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 definujeme:

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

*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  $(Q, X, \delta, q_0, F)$ , kde  $Q, X, q_0$  a  $F$  má stejný význam jako pro deterministický akceptor a  $\delta : Q \times X \rightarrow Q \times \{-1, 0, 1\}$  je přechodová funkce.*

*Na vstupní pásce je každá pozice mimo vstupní slovo prázdná<sup>45</sup>.*

*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  $\alpha$  je přijímáno dvoucestným automatem  $\mathcal{A}$ , když existuje přijímající výpočet automatu  $\mathcal{A}$  nad slovem  $\alpha$ . 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  $\alpha$  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.*

<sup>44</sup>podobně jako N-akceptory byly efektivnější než D-akceptory

<sup>45</sup>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 necht'  $(r_0, i_0), \dots, (r_k, i_k)$  je přijímající výpočet  $\mathcal{A}$  nad slovem  $\alpha$ . Pozorujme, jak se chová výpočet na přechodu mezi  $p$ -tou buňkou a  $(p+1)$ -ní buňkou pro nějaké  $p = 0, \dots, n$ . Necht'  $j_1, \dots, j_m$  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  $\alpha$  mezi  $p$ -tou a  $(p+1)$ -ní pozicí (viz obr. 8a). Všimněme si, že platí:

- $m$  je liché
- $(\forall t, s)(1 \leq 2t+1, 2s+1 \leq m, t \neq s \Rightarrow (q_{2t+1} \neq q_{2s+1}) \wedge (q_{2t} \neq q_{2s}))$ <sup>46</sup>

Tedy  $m \leq 2|Q|$ . Každou posloupnost stavů  $q_1, \dots, q_m$ , splňující předchozí dvě podmínky<sup>47</sup>, budeme nazývat platnou posloupností  $\mathcal{A}$ . Dále definujeme proces  $\check{S}krt(A, B, x)$ , 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 zaškrťávat 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<sup>48</sup>,
- 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<sup>49</sup>.
2. Dokud lze zaškrťávat<sup>50</sup>, provádíme

<sup>46</sup> pokud by se např.  $q_\ell$  vyskytl dvakrát v přechodové posloupnosti ve stejném směru,  $\mathcal{A}$  by se zacyklil. Jestliže totiž před druhým průchodem stavem  $q_\ell$  automat nedosáhl konce vstupního slova, po druhém průchodu  $q_\ell$  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  $q_\ell$  do druhého, tedy na konec vstupního slova se nikdy nedostane. Pokud před druhým průchodem  $q_\ell$  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  $q_\ell$  se již podruhé nejde).

<sup>47</sup> nyní bez ohledu na vstupní slovo

<sup>48</sup> neboli součet všech zaškrtnutých prvků v  $A$  i  $B$  je sudý

<sup>49</sup> zatím je to snadné, že ... ☺

<sup>50</sup> pochopitelně nelze zaškrťávat (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 zaškrťávání 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 zaškrťávání skončí neúspěšně.
3. pokud jsou obě posloupnosti  $A$  i  $B$  celé zaškrtnuté, algoritmus úspěšně skončil<sup>52</sup>.

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  $\alpha$  mezi  $p$ -tou a  $(p + 1)$ -ní pozicí,  $B$  je přechodová posloupnost nad  $\alpha$  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  $\alpha$  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  $\alpha$ . 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  $\alpha$  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  $p = 0, \dots, n - 1$ .*

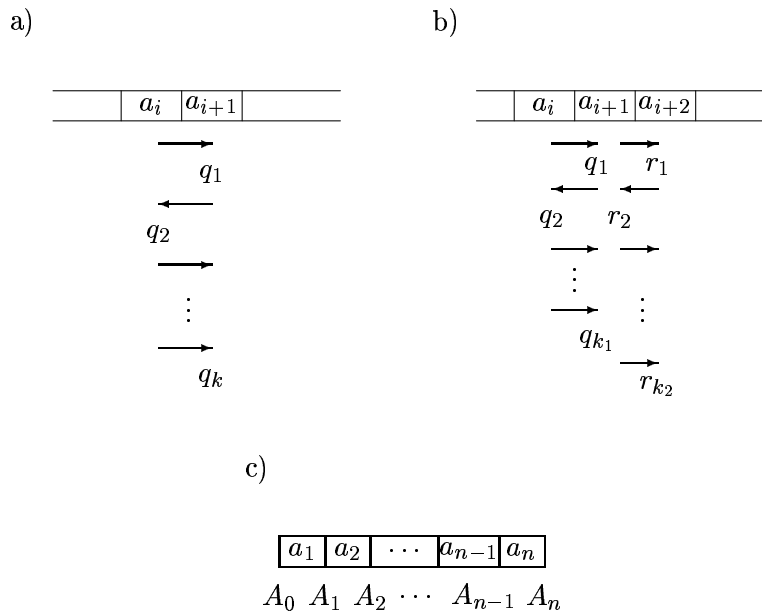
*Naopak, když  $\alpha = a_1 \dots a_n \in X^*$  a  $A_0, \dots, A_n$  jsou platné posloupnosti nad  $\mathcal{A}$  takové, že  $A_0 = \{q_0\}$ ,  $A_n = \{q\}$  pro nějaké  $q \in F$  a  $\check{S}krt(A_p, A_{p+1}, a_{p+1})$  skončí úspěšně pro každé  $p = 0, \dots, n - 1$ , pak existuje přijímací výpočet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

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

<sup>51</sup>to jsou přechody z buňky  $x$  zpět do  $x$ , což nás teď nezajímá

<sup>52</sup>Pokud skončil, aniž by byly  $A$  i  $B$  celé zaškrtnuté, jde pochopitelně o neúspěch.



Obr. 8: Přechodové posloupnosti

$A_0, \dots, A_n$  tvoří přechodové posloupnosti doposud zkonstruovaného výpočtu. Dále bude dán aktivní prvek, bude to poslední stav ve výpočtu a zároveň poslední zaškrtnutý prvek v posloupnostech  $A_0, \dots, A_n$ . 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. necht'  $q \in A_p$  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  $\alpha$ , pak po přidání opět dostaneme výpočet nad  $\alpha$ , 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  $\alpha$ , protože před přidáním jsme měli výpočet nad  $\alpha$  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  $\alpha$ , pak po přidání opět dostaneme výpočet nad  $\alpha$ , 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  $\alpha$ , 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 přijímací výpočet nad  $\alpha$  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
- $\delta'(A, x) = \{B \in Q'; \check{S}krt(A, B, x) \text{ úspěšně skončí}\}$  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  $(r_0, i_0), \dots, (r_k, i_k)$  je přijímající výpočet  $\mathcal{A}$  nad slovem  $\alpha$  a  $(r_t, i_t) = (r_s, i_s)$  pro  $t < s$ , pak

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

je opět přijímající výpočet  $\mathcal{A}$ . Proto pro každé slovo  $\alpha \in L(\mathcal{A})$  existuje prostý přijímající výpočet<sup>53</sup>. 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 sestavit si nedeterministický akceptor přijímající  $L'_n$ . My teď máme jiný úkol: sestavit pro  $L'_n$  deterministický dvoucestný automat<sup>54</sup>. Je to snadné<sup>55</sup>...

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.  $q_0, \dots, q_{n+2}$  jsou stavy  $\mathcal{A}$ , ohodnocení hran kóduje přechodovou funkci a posun čtecí hlavy na vstupní pásce<sup>56</sup>. Není příliš obtížné přesvědčit se, že  $\mathcal{A}$  přijímá právě  $L'_n$ . **FINE**

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

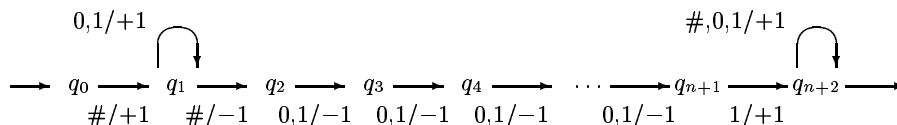
<sup>53</sup> prostý přijímací výpočet je takový, ve kterém se žádná konfigurace neopakuje

<sup>54</sup> pochopitelně s co nejméně stavy, jinak to asi nemá valný význam

<sup>55</sup> jako vždycky

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

<sup>57</sup> 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<sup>58</sup>, ale jsou efektivnější<sup>59</sup>“. Uvedená hypotéza je samozřejmě pravdivá, nicméně její ověření není jednoduché a přesahuje rámec těchto skript. Je možné ukázat (viz [5],[2]), že existuje jazyk

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

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 prokázána, 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.<sup>60</sup>

## 4.5 Substituce

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

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

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

<sup>58</sup>v tom smyslu, že přijímají stejnou třídu jazyků

<sup>59</sup>tedy pro daný jazyk mají buď řádově méně stavů, nebo alespoň odstaňují nedeterminismus

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

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

Zobrazení  $\sigma^{-1}$  nazýváme inverzní substitucí.

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

Abychom zápis v definici trochu polidštili. Substituce  $\sigma$  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  $\sigma$  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  $\sigma : A \rightarrow \mathcal{P}(B^*)$  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  $\sigma(x)$ <sup>61</sup> 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<sup>62</sup>.

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 inkluze  $\sigma^{-1}(N) \subseteq L$ , 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  $a$ .

<sup>61</sup> $x \in \{a, b, c\}$

<sup>62</sup>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  $\sigma$  je substituční zobrazení  $\sigma'$

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

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<sup>63</sup>:

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

Někomu by se možná zdálo, že

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

(neboli  $\sigma^{-1}(N')$  je konkatenační obraz množin  $O, P, R$  v  $\sigma^{-1}$ ) 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<sup>64</sup>:

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

<sup>63</sup>opět ztotožňujeme množinu a reg. výraz, který ji popisuje

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



což po rozepsání  $\sigma^{-1}$  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  $\lambda$  z  $\{a, b, c\}^*$ , pro která existuje nějaké slovo  $\alpha \in OPR$  s rozkladem  $\alpha = \alpha_1 \dots \alpha_n$  (kde  $\alpha_i \in \sigma(a) \cup \sigma(b) \cup \sigma(c)$ ) takovým, že pro nějaké  $i_0$  ( $1 \leq i_0 \leq n$ )

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

a zároveň

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

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  $M$ <sup>65</sup>. 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  $\alpha$  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  $\sigma$  i  $\sigma'$  jsou regulární substituce. Uznávám, že popsany 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  $\sigma$  a libovolné regulární jazyky  $L \subseteq A^*, M \subseteq B^*$  jsou  $\sigma(L), \sigma^{-1}(M)$  také regulární jazyky.*

Důkaz:

---

<sup>65</sup>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  $\sigma$  regulární substituce, pak  $\forall a \in A$  je  $\sigma(a)$  regulární jazyk. Podle Kleeneho věty (4.35) existuje regulární výraz  $\beta_a$  popisující jazyk  $\sigma(a)$ . Podobně existuje regulární výraz  $\beta$  popisující  $L$ . Potom ale můžeme každou proměnnou  $a \in \beta$  nahradit výrazem  $(\beta_a)$  a dostaneme regulární výraz  $\gamma$  (regularita  $\gamma$  plyne z definice). Jazyk  $\sigma(L)$  je zřejmě popisován regulárním výrazem  $\gamma$ , neboli je regulární.
2. Je-li  $\sigma$  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 buno<sup>66</sup> 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<sup>67</sup>  $\mathcal{A} = (Q, B, \delta, q_0, F)$ . Definujme dvoucestný nedeterministický automat  $\mathcal{A}' = (Q', A, \delta', I', F')$ , kde

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

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  $a_1 \dots a_n$  se pokouší najít ke každému  $a_i$  odpovídající slovo  $\alpha_i \in \sigma(a_i)$  takové, aby  $\alpha_1 \dots \alpha_n \in M$ . 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  $a_1 \dots a_n$ , právě když  $\mathcal{A}$  akceptuje slovo  $\alpha_1 \dots \alpha_n$ .

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  $\delta'$  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  $\delta'$ ) 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  $\alpha_1$ . Tak postupujeme až k  $a_n$ , kde  $\mathcal{A}_{a_n}$  „navrhne“ slovo  $\alpha_n$  a skončí v některém ze svých přijímacích stavů<sup>68</sup>. Pokud zároveň i momentální stav v  $\mathcal{A}$  je přijímací,  $\mathcal{A}'$  se podle třetího

<sup>66</sup>bez újmy na obecnosti

<sup>67</sup>determinismus akceptorů  $\mathcal{A}$  a  $\mathcal{A}_a$  není až tak důležitý, jen se snažím buno zjednodušit definici  $\mathcal{A}'$

<sup>68</sup>když ne, vezmeme si jiný výpočet — od toho je nedeterministický

řádku definice  $\delta'$  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  $\delta'$  dále snadno plyne, že  $\mathcal{A}'$  může přijmout jen slova, která jsou v  $\sigma^{-1}(M)$ , což ponechávám čtenáři na rozmyšlení<sup>69</sup>.

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

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

- 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:

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

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

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

<sup>69</sup>Prakticky to znamená ověřit, že se do  $q_f$  můžeme dostat jen způsobem popsáním 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  $u \in L' \setminus L$ . Naopak je-li  $vu \in L$  pro nějaké  $v \in L'$  (tzn.  $u \in L' \setminus L$ ), 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  $L(\mathcal{A}_1) = L' \setminus L$ .

Pro  $\mathcal{A}_2$  vezmeme všechna slova  $v \in L'$  a najdeme k nim stavy  $q$  takové, že  $\delta^*(q, v) \in F$ <sup>70</sup>. 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  $\mathcal{A}$ <sup>71</sup>.

**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<sup>72</sup>. Na druhou stranu je-li  $T(\mathcal{A})$  jazyk přijímaný dvoucestným automatem  $\mathcal{A}$  s koncovými znaky, pak zřejmě

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

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...

<sup>70</sup>nemusí to jít pro všechna  $v$

<sup>71</sup>viz definice 4.39

<sup>72</sup>ale je vhodné jako cvičení si to rozmyslet

## 5 Gramatiky

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

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

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

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

**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

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

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

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

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

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

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

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

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

3. Bezkontextové gramatiky (gramatiky typu 2) splňují

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

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

4. Regulární gramatiky (gramatiky typu 3) splňují

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

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

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

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

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  $(X \rightarrow Y\alpha)$ , kde  $X \in V_N$  a  $\alpha \in V_T^*$ ,  $Y \in V_N \cup \{\Lambda\}$ . Nikoho asi nepřekvapí, že generují právě regulární jazyky.<sup>1</sup>
- Lineární gramatiky jsou obecnější. Obsahují pravidla  $(X \rightarrow \alpha Y \beta)$ , případně  $(X \rightarrow \alpha)$ , kde  $\alpha, \beta \in V_T^*$ ,  $X, Y \in V_N$ . Generují regulární jazyky a některé další (např.  $\{a^n b^n; n = 0, 1, \dots\}$  — 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

<sup>1</sup>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.  $(\alpha \rightarrow \beta) \in P \Rightarrow ((\alpha = S \wedge \beta = \Lambda) \vee (|\alpha| \leq |\beta|))$

**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 chvíli 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

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

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  $D \rightarrow c$  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

$$\begin{aligned} S &\xrightarrow{G} aBD \xrightarrow{G} aBaBDc \xrightarrow{G} aaBBDC \xrightarrow{G} \\ &\xrightarrow{G} aaBBcc \xrightarrow{G} aaBbcc \xrightarrow{G} aabbcc. \end{aligned}$$

**FINE**

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

1.  $S$  není na pravé straně žádného pravidla

---

<sup>2</sup>prázdné slovo nás teď nezajímá

2.  $(X \rightarrow \Lambda) \in P \Rightarrow X = S$
3.  $(X \rightarrow \alpha) \in P \Rightarrow \alpha \notin V_N$  (neboli  $\alpha$  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 standardní formě. Je-li navíc  $G$  regulární, je i  $\tilde{G}$  regulární.*

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

1. Za každé pravidlo  $(S \rightarrow \alpha) \in P$  přidáme<sup>3</sup> do  $\tilde{P}$  pravidlo  $(\tilde{S} \rightarrow \alpha)$ . 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  $\Lambda$ <sup>4</sup>. Pokud  $S \in A$ , přidáme do  $\tilde{P}$  pravidlo  $\tilde{S} \rightarrow \Lambda$ . Dále pro všechna  $X \in V_N \setminus \{\tilde{S}\}$ 
  - vyřadíme z  $\tilde{P}$  pravidlo  $(X \rightarrow \Lambda)$  (pokud tam vůbec bylo)
  - a pro každé pravidlo  $(X \rightarrow \alpha)$ , 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  $(X \rightarrow \alpha_1 \alpha_2 \dots \alpha_n)$ <sup>5</sup>.

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

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

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

<sup>3</sup>tzn.  $(S \rightarrow \alpha)$  v  $\tilde{P}$  zůstane

<sup>4</sup>množinu  $A = \{X \in V_N; X \xrightarrow[G]{*} \Lambda\}$  je možné získat například rekurzivně — od množiny

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

postupně přecházet k delším odvozením, která přepisují neterminály na  $\Lambda$ , tzn.

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

Zřejmě platí  $A_{i-1} \subseteq A_i$ .  $A = \bigcup_{i \in \mathbb{N}_0} A_i$  a neboť  $V_N$  je konečná množina, je i  $A$  konečná, tudíž existuje  $k$  takové, že  $A_k = A$ .

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

<sup>6</sup>první pravidlo z  $\tilde{P}$ , které není v  $P$  je  $(\tilde{S} \rightarrow \beta)$ , které nás nyní nezajímá (bylo náplní první části důkazu)



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

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

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  $\Lambda$  (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  $(S \rightarrow \beta)$ , kde  $\beta \in A^*$  a tedy pravidlo  $(\tilde{S} \rightarrow \Lambda) \in \tilde{P}$  a proto  $\alpha \in L(\tilde{G})$ .

Předpokládejme  $\alpha \neq \Lambda$ . Vezměme derivaci

$$S \Rightarrow \beta_1 \Rightarrow \beta_2 \dots \Rightarrow \beta_n = \alpha$$

v gramatice  $G$  a budeme postupně zatrhávat písmena ve slovech  $\beta_i$  ( $i = n, \dots, 1$ ). 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  $(X \rightarrow \Lambda) \in P$ , pak zatrhneme písmeno  $X$  v  $\beta_i$ .
- Dále když  $\beta_i = \beta_{i,0} X \beta_{i,1}$ ,  $\beta_{i+1} = \beta_{i,0} \beta_{i,2} \beta_{i,1}$  a  $(X \rightarrow \beta_{i,2}) \in P$  a celé  $\beta_{i,2}$  je ve slově  $\beta_{i+1}$  zatrženo, pak zatrhneme  $X$  ve slově  $\beta_i$ .

Všimněte si, že všechna zatržená písmena jsou z  $A$  a v derivaci  $\alpha$  se přepíší na  $\Lambda$ .

Nyní budeme indukcí konstruovat derivaci

$$\tilde{S} \Rightarrow \gamma_1 \Rightarrow \gamma_2 \dots \Rightarrow \gamma_m = \alpha$$

v gramatice  $\tilde{G}$ . Ke každému slovu  $\beta_i$  bude přiřazeno slovo  $\gamma_j$ , které vznikne ze slova  $\beta_i$  vynecháním všech zatržených písmen. Řekneme, že takové  $\gamma_j$  odpovídá slovu  $\beta_i$ . Protože  $\alpha \neq \Lambda$ , není  $S$  zatržené a tedy ani nejsou všechna písmena v  $\beta_1$  zatržena.

Slovo vzniklé z  $\beta_1$  vynecháním všech zatržených písmen označme  $\gamma_1$ . Zřejmě  $\gamma_1$  odpovídá  $\beta_1$  a z definice  $A$  platí  $(\tilde{S} \rightarrow \gamma_1) \in \tilde{P}$ . Mějme  $\beta_i$  a nechť mu odpovídá  $\gamma_j$ . 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  $(X \rightarrow \beta_{i,2}) \in P$ . Když  $X$  je zatržené v  $\beta_i$ , pak každé písmeno  $\beta_{i,2}$  je zatrženo v  $\beta_{i+1}$  a  $\gamma_j$  odpovídá  $\beta_{i+1}$ . Uvažujme, že  $X$  není v  $\beta_i$  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  $\beta_{i+1}$  nejsou zatržená, dostáváme  $\gamma_{j,2} \neq \Lambda$  a tedy  $(X \rightarrow \gamma_{j,2})$  je pravidlo z  $\tilde{P}$ . Pak  $\gamma_{j+1} = \gamma_{j,0} \gamma_{j,2} \gamma_{j,1}$  je přímo odvozeno z  $\gamma_j$  a  $\gamma_{j+1}$  odpovídá slovu  $\beta_{i+1}$ . Protože  $\alpha$  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 přechodí části, která byla naopak poněkud neformální. Definujme relaci  $\rho$  tak, že

$$(\forall X, Y \in V_N)[(X, Y) \in \rho \stackrel{df}{=} (X \xrightarrow[*]{\tilde{G}} Y)]$$

Relace  $\rho$  je zřejmě tranzitivní a reflexivní<sup>7</sup>. Z  $\tilde{P}$  vyřadíme všechna pravidla  $(X \rightarrow Y)$ , kde  $X, Y \in V_N$ , a za každé pravidlo

$$(X \rightarrow \alpha_1 Z_1 \alpha_2 Z_2 \dots Z_{n-1} \alpha_n)$$

přidáme všechna pravidla

$$(X \rightarrow \alpha_1 Y_1 \alpha_2 Y_2 \dots Y_{n-1} \alpha_n)$$

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<sup>8</sup>. Sestrojíme nejprve gramatiku  $\tilde{G} = (\tilde{V}_N, V_T, S, \tilde{P})$ , která bude také regulární ve standartní formě a navíc bude splňovat

$$\begin{aligned} (\forall X, Y \in V_N \setminus \{S\})(\forall \alpha \in V_T^*) \quad & [(X \rightarrow \alpha Y) \in \tilde{P} \Rightarrow |\alpha| = 1] \\ (\forall X \in V_N \setminus \{S\})(\forall \alpha \in V_T^*) \quad & [(X \rightarrow \alpha) \in \tilde{P} \Rightarrow |\alpha| = 1] \\ (\forall \alpha \in V_T^*) \quad & [(S \rightarrow \alpha) \in \tilde{P} \Rightarrow |\alpha| \leq 1] \end{aligned}$$

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  $\tilde{p} = (X \rightarrow a_1 \dots a_n Y) \in \tilde{P}^9$  z  $\tilde{P}$  vyhodíme a nahradíme sérií pravidel

$$\begin{aligned} & (X \rightarrow a_1 X_1), \\ (\forall i = 1, \dots, n-2) \quad & [(X_i \rightarrow a_{i+1} X_{i+1})], \\ & (X_{n-1} \rightarrow a_n Y), \end{aligned}$$

kde  $X_i$  jsou nové neterminály, dosud se ve  $\tilde{V}_N$  nevyskytující<sup>10</sup>. Tato pravidla odpovídají požadavkům na  $\tilde{P}$ , zachovávají standartní formu  $\tilde{G}$  a

<sup>7</sup>ne však nutně symetrická, nemusí tedy být ekvivalencí

<sup>8</sup>tímto požadavkem neztrácí důkaz díky tvrzení 5.7 na obecnosti

<sup>9</sup> $Y \in V_N \cup \{\Lambda\}$ ,  $X \in V_N$ , může být tedy  $X = S$

<sup>10</sup>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

$$\begin{aligned} \bullet F &= \begin{cases} \{f\}, & \text{pokud } (S \rightarrow \Lambda) \notin \tilde{P} \\ \{f, S\}, & \text{pokud } (S \rightarrow \Lambda) \in \tilde{P} \end{cases} \\ \bullet \delta(X, a) &= \begin{cases} \{Y \in \tilde{V}_N; X \rightarrow aY \in \tilde{P}\} \cup \{f\}, & \text{když } (\exists a \in V_T)((X \rightarrow a) \in \tilde{P}) \\ \{Y \in \tilde{V}_N; X \rightarrow aY \in \tilde{P}\}, & \text{když } (\forall a \in V_T)((X \rightarrow a) \notin \tilde{P}) \end{cases} \end{aligned}$$

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  $\alpha$  z  $S$  v  $\tilde{G}$ , tzn.

$$S \xrightarrow{\tilde{G}} a_1 X_1 \xrightarrow{\tilde{G}} \dots \xrightarrow{\tilde{G}} a_1 \dots a_{n-1} X_{n-1} \xrightarrow{\tilde{G}} a_1 \dots a_n,$$

což je podle definice  $\delta$  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  $\Lambda$  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

$$\begin{aligned} (\forall q \in I) \quad & [(S \rightarrow q) \in P] \\ (\forall q, r \in Q)(\forall a \in X) \quad & [q \in \delta(r, a) \Rightarrow (r \rightarrow aq) \in P] \\ (\forall r \in Q)(\forall a \in X) \quad & [F \cap \delta(r, a) \neq \emptyset \Rightarrow (r \rightarrow a) \in P] \end{aligned}$$

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

$$(37) \quad S \Rightarrow q_0 \Rightarrow a_1 q_1 \Rightarrow \dots \Rightarrow a_1 \dots a_{n-1} q_{n-1} \Rightarrow a_1 \dots a_n$$

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  $S \xrightarrow[*]{G} a_1 \dots a_n$ . Vzhledem k pravidlům v  $P$  jediný možný tvar derivace  $\alpha$  z  $S$  je (37), což nastane pouze pro

$$\begin{aligned} q_0 &\in I, \\ (\forall i = 1, \dots, n-2) \quad &[q_{i+1} \in \delta(q_i, a_{i+1})], \\ q_n &\in F \cap \delta(q_{n-1}, a_n) \neq \emptyset. \end{aligned}$$

Pak ovšem  $q_0, \dots, q_n$  je přijímací výpočet  $\mathcal{A}$  nad  $\alpha$ , 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  $\mathcal{L}_3 \subseteq \mathcal{L}_2$ . 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

$$(\{S\}, \{a, b\}, S, \{S \rightarrow aSb \mid \Lambda\})$$

**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  $(S \rightarrow \alpha) \in P$  přidáme  $(S_1 \rightarrow \alpha)$  do  $P$ <sup>11</sup>.  $G_L$  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  $(S \rightarrow \Lambda)$  nám nevádí, neboť z definice nezkracující gramatiky (str. 62) není  $S$  na pravé straně žádného pravidla. Dále jsou „v pořádku“ pravidla  $(\alpha \rightarrow \beta)$ , 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<sup>13</sup>, leč my se s nimi vypořádáme a upravíme  $G$  tak, aby byla kontextová. Nechť

$$p = (A_1 \dots A_n \rightarrow B_1 \dots B_m) \in P,$$

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

<sup>11</sup>podrobněji viz také první část důkazu lemmatu 5.7

<sup>12</sup>vyplývá triviálně z definice kontextové gramatiky

<sup>13</sup>např.  $(XY \rightarrow YX)$  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  $(A_t \rightarrow t)$  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

$$p = (A_1 \dots A_n \rightarrow B_1 \dots B_m) \in M$$

přidáme do  $V_N$  nové neterminály  $C_1, \dots, C_{n-1}$  a do  $P$  pravidla

$$\begin{aligned} A_1 A_2 \dots A_n &\rightarrow C_1 A_2 \dots A_n \\ C_1 A_2 \dots A_n &\rightarrow C_1 C_2 A_3 \dots A_n \\ &\vdots \\ C_1 \dots C_{n-2} A_{n-1} A_n &\rightarrow C_1 \dots C_{n-1} A_n \\ C_1 \dots C_{n-1} A_n &\rightarrow C_1 \dots C_{n-1} B_n \dots B_m \\ C_1 \dots C_{n-1} B_n \dots B_m &\rightarrow C_1 \dots C_{n-2} B_{n-1} B_n \dots B_m \\ &\vdots \\ C_1 B_2 \dots B_m &\rightarrow B_1 B_2 \dots B_m \end{aligned}$$

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  $\rightarrow$  Kontextová*) *Nezkracující gramatiku*

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

kde  $P$  obsahuje pravidla

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

převeďte na ekvivalentní kontextovou gramatiku  $G'$ .<sup>14</sup>

Jak snadno zjistíme z definice Chomského hierarchie, jediné pravidlo, které dělá z gramatiky  $G$  nekontextovou, je  $p = (Ba \rightarrow aB)$ . 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<sup>15</sup> lze  $P$  modifikovat

<sup>14</sup>pokud si vzpomínáte, byla gramatika  $G$  již jednou vzpomenua v příkladu 5.5

<sup>15</sup>a dokonce jak

na ekvivalentní systém  $P'$ , obsahující jen kontextová pravidla<sup>16</sup>. 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  $p_a = (A_a \rightarrow a)$ . Dále obohatíme množinu neterminálů v  $G'$  o nový neterminál  $C_1$  a přepisovací systém  $P'^{17}$  o pravidla

$$\begin{aligned} BA_a &\rightarrow BC_1 \\ BC_1 &\rightarrow A_aC_1 \\ A_aC_1 &\rightarrow A_aB, \end{aligned}$$

která jsou již kontextová a podle důkazu věty 5.9 nahrazují spolu s pravidlem  $p_a$  jedno-jednoznačně pravidlo  $p = (BA_a \rightarrow A_aB)$ , 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

$$\begin{aligned} S &\rightarrow A_aBD \mid \Lambda \\ D &\rightarrow c \mid A_aBDC \\ Bc &\rightarrow bc \\ Bb &\rightarrow bb \\ BA_a &\rightarrow BC_1 \\ BC_1 &\rightarrow A_aC_1 \\ A_aC_1 &\rightarrow A_aB \\ A_a &\rightarrow a. \end{aligned}$$

**FINE**

## 5.2 Bezkontextové gramatiky

### 5.2.1 Normální formy

**Definice 5.11** *Bezkontextová gramatika  $G = (V_N, V_T, S, P)$  je v Chomského normální formě, pokud každé pravidlo z  $P$  má jeden z následujících tvarů:*

1.  $X \rightarrow YZ$ , kde  $X, Y, Z \in V_N$
2.  $X \rightarrow a$ , kde  $X \in V_N, a \in V_T$
3.  $S \rightarrow \Lambda$

Gramatiku v Chomského normální formě lze triviálně pozměnit tak, aby byla ve standardní formě (viz definice 5.6)<sup>18</sup>.

<sup>16</sup>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  $G'$ .

<sup>17</sup> $P'$  je vytvářený přepisovací systém gramatiky  $G'$ , pokud to není jasné...

<sup>18</sup>stačí přidat nový počáteční neterminál  $S_1$  a pokud  $S$  byl původní počáteční neterminál, pak za každé pravidlo  $(S \rightarrow \alpha)$  přidáme  $(S_1 \rightarrow \alpha)$ , 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<sup>19</sup> můžeme předpokládat, že  $G$  je ve standartní formě, čímž se zbavíme povinnosti upravovat pravidla  $(X \rightarrow \Lambda)$  a  $(X \rightarrow Y)$  pro  $(X, Y \in V_N)$ . Jako inicializaci položíme

$$G' = (V'_N, V_T, S, P') = (V_N, V_T, S, P)$$

(neboli  $L(G) = L(G')$ ). Vezměme pravidlo

$$p = (X \rightarrow A_1 \dots A_n) \in P', A_i \in V_N \cup V_T.$$

Je-li  $n = 0$ , musí být  $X = S$ . Je-li  $n = 1$ , pak  $A_1 \in V_T$ <sup>20</sup>. Necht' tedy  $n \geq 2$ . Potom každý terminál  $A_j, j \in J \subseteq \langle 1, n \rangle$  nahradíme v  $p$  novým neterminálem  $A'_j \in V'_N$  a do  $P'$  přidáme pravidlo  $(A'_j \rightarrow A_j)$ . 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

$$\begin{aligned} X &\rightarrow A_1 B_1 \\ B_1 &\rightarrow A_2 B_2 \\ &\vdots \\ B_{n-2} &\rightarrow A_{n-1} A_n \end{aligned}$$

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  $(X \rightarrow a\alpha)$ , kde  $a \in V_T, \alpha \in V_N^*, S \notin \alpha$ . Navíc může  $G$  obsahovat pravidlo  $(S \rightarrow \Lambda)$ .*

*Bezkontextovou gramatiku  $G$  nazveme v Hopfově normální formě, pokud její pravidla mají tvar  $(X \rightarrow a\alpha b)$ , kde  $a, b \in V_T, \alpha \in V_N^*, S \notin \alpha$ . Navíc může  $G$  obsahovat pravidlo  $(S \rightarrow \Lambda)$  a pravidla  $(S \rightarrow a)$  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.

<sup>19</sup>za použití lemmatu 5.7

<sup>20</sup>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  $B_1, \dots, B_k$  uspořádanými v rovinném nakreslení  $T$  zleva doprava, pak  $(A \rightarrow B_1 \dots B_k) \in P$ .*

*Derivačnímu stromu  $T$  přiřadíme slovo  $\alpha = A_1 \dots A_n$ , 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  $\alpha$ .

**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*

$$G = (\{S, A\}, \{a, b, c\}, S, \{S \rightarrow aSb \mid A, A \rightarrow aA \mid Ab \mid c\})$$

*derivační strom slova  $\alpha = aaacbb$ .*

Nejprve si z cvičných důvodů uvědomíme, jaký jazyk gramatika  $G$  generuje. Asi každý po chvíli uvažování uzná, že  $L(G) = \{a^i cb^j; i, j = 1, 2, \dots\}$ . Jeden z derivačních stromů slova  $\alpha$  je na obrázku 10. FINE

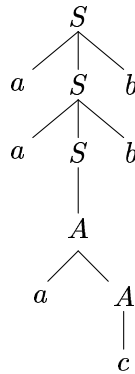
**Věta 5.17** *(Pumping lemma) Když  $G = (V_N, V_T, S, P)$  je bezkontextová gramatika, pak existují čísla  $m, n \in N_0$  taková, že každé slovo  $\alpha \in L(G)$ ,  $|\alpha| \geq n$  lze napsat jako  $\alpha = uvwx^{21}$  tak, že platí*

1.  $|vwx| < m$
2.  $v \neq \Lambda \vee x \neq \Lambda$
3.  $(\forall i \in N_0)(uv^iwx^i \in L(G))$

---

<sup>21</sup>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<sup>22</sup>.

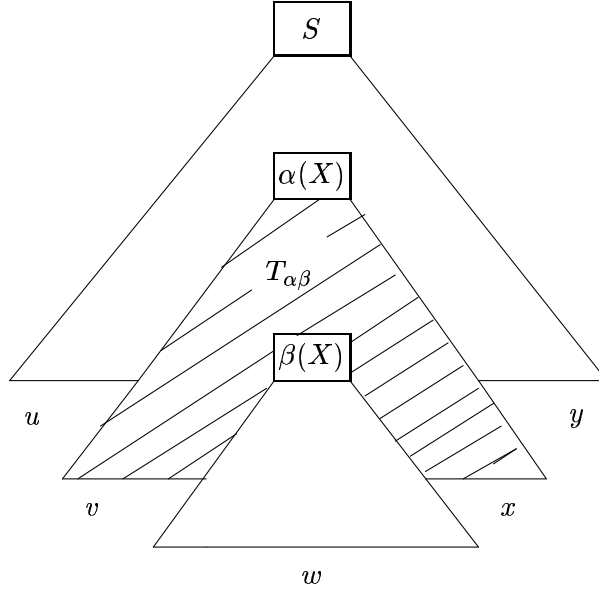
Označme  $\mathcal{T}$  množinu všech derivačních stromů  $T$  gramatiky  $G$ , které mají za kořen  $S$ , listy ohodnocené terminály a na žádné cestě z kořene do listu neexistují dva různé uzly ohodnocené stejným neterminálem. Protože  $V_N$  je konečná, je délka každé cesty kořen–list v  $T \in \mathcal{T}$  menší nebo rovna  $|V_N|$ , neboli stromů  $T \in \mathcal{T}$  je konečně mnoho<sup>23</sup>, 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  $\mathcal{T}$ <sup>24</sup>. Protože  $T \notin \mathcal{T}$ , existují v  $T$  na jedné cestě kořen–list dva různé uzly (označme je  $\alpha$ ,  $\beta$ , necht'  $\alpha$  je výš než  $\beta$ ) ohodnocené stejným neterminálem  $X$  — viz obrázek 11. Potom ale část  $T$  mezi  $\alpha$ ,  $\beta$  (označme ji

<sup>22</sup>nač bychom je také jinak měli. . .

<sup>23</sup>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 konečkonců konečné číslo. Jak jsme dospěli k tomuto číslu? Rozšířme 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  $\mathcal{T}$

<sup>24</sup>neboť všechny derivační stromy pro  $\alpha$  mají více než  $n$  listů



Obr. 11: Rozklad slova z bezkontextového jazyka podle Pumping lematu

$T_{\alpha\beta}$ ) můžeme vynechat, nebo i libovolněkrát zopakovat<sup>25</sup> a dostaneme derivační stromy  $T_0$  (vynecháním  $T_{\alpha\beta}$ ),  $T_1 = T$ ,  $T_2$  dvojnásobným zopakováním  $T_{\alpha\beta}, \dots, T_i$  jsou derivační stromy v  $G$  a generují slova  $uvw, uvwxy, uvvwxy, \dots$ <sup>26</sup>, neboli  $(\forall i \in \mathbb{N}_0)(uv^iwx^i y \in L(G))$ , 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<sup>27</sup>. 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 standardní formě<sup>28</sup>. Pak je-li  $X$  neterminál ohodnocující kořen a zároveň některý další uzel nějakého stromu  $T \in \mathcal{T}_1$ , nemůže se přepsat na  $\Lambda$  ani na samotné  $X$ , neboli v odvození  $X \xrightarrow{*} vXx$

<sup>25</sup>Strom na obrázku 11 odpovídá zřejmě odvození  $S \xrightarrow{*} uXy \xrightarrow{T_{\alpha\beta}} uvXxy \xrightarrow{*} uvwx$ , neboli existují derivace  $X \xrightarrow{*} vXx$ ,  $X \xrightarrow{*} w$  (protože  $G$  je bezkontextová). Tím pádem jsou

$$\begin{aligned} S &\xrightarrow{*} uXy \xrightarrow{*} uvw \\ S &\xrightarrow{*} uXy \xrightarrow{T_{\alpha\beta}} uvXxy \xrightarrow{T_{\alpha\beta}} uvvXxxy \xrightarrow{*} \dots \xrightarrow{*} uv^iXx^i y \xrightarrow{*} uv^iwx^i y, \end{aligned}$$

derivace v  $G$ , které se liší pouze počtem zopakování části  $T_{\alpha\beta}$ .

<sup>26</sup>derivační strom  $T_i$  reprezentuje odvození

$$S \xrightarrow{*} uXy \xrightarrow{*} uvXxy \xrightarrow{*} uv^2Xx^2y \xrightarrow{*} \dots \xrightarrow{*} uv^iXx^i y \xrightarrow{*} uv^iwx^i y$$

<sup>27</sup>tato definice znamená, že stromy z  $\mathcal{T}_1$  reprezentují odvození  $X \xrightarrow{*} vXx \xrightarrow{*} vwx$

<sup>28</sup>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

$$L = \{a^i b^i c^i; i = 0, 1, \dots\}$$

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  $(\forall i \in \mathbb{N}_0)(uv^i wx^i y \in L)$ , 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

$$\{a^i b^j c^i d^j; i, j = 0, 1, \dots\},$$

který není bezkontextový. Naproti tomu velmi podobné jazyky

$$\begin{aligned} &\{a^i b^i c^j d^j; i, j = 0, 1, \dots\} \\ &\{a^i b^j c^j d^i; i, j = 0, 1, \dots\} \end{aligned}$$

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<sup>30</sup>.

### 5.2.3 Operace nad bezkontextovými gramatikami

<sup>29</sup>  $\#x$  značí počet výskytů písmene  $x$  v nějakém slově, v našem případě  $\alpha_0$

<sup>30</sup> Leč nezufojte! 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

$$\begin{aligned} G_1 &= (V_{N_1}, X, S_1, P_1), \\ G_2 &= (V_{N_2}, X, S_2, P_2) \end{aligned}$$

generují jazyky  $L_1$  a  $L_2$ . Předpokládejme, že  $V_{N_1} \cap V_{N_2} = \emptyset$ . Gramatika, generující  $L$ , je

$$G = (V_{N_1} \cup V_{N_2} \cup \{S\}, X, S, P_1 \cup P_2 \cup \{S \rightarrow S_1, S \rightarrow S_2\})$$

pro  $S \notin V_{N_1} \cup V_{N_2}$ . 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

$$\begin{aligned} L_1 &= \{a^i b^j c^j; i, j = 0, 1, \dots\} \\ L_2 &= \{a^i b^j c^j; i, j = 0, 1, \dots\} \\ L &= L_1 \cap L_2 = \{a^i b^i c^i; i = 0, 1, \dots\} \end{aligned}$$

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

$$\begin{aligned} X &= \{a, b, c\} \\ P_1 &= \{S_1 \rightarrow aS'_1bC \mid C \mid \Lambda, C \rightarrow cC \mid \Lambda, S'_1 \rightarrow aS'_1b \mid \Lambda\} \\ P_2 &= \{S_2 \rightarrow AbS'_2c \mid A \mid \Lambda, A \rightarrow aA \mid \Lambda, S'_2 \rightarrow bS'_2c \mid \Lambda\} \end{aligned}$$

$L$  ale bezkontextový není, což je ukázáno v příkladu 5.18.

**Doplňěk** Protože z *de Morganových pravidel* pro jakékoli množiny, tedy i pro bezkontextové jazyky platí<sup>31</sup>

$$L_1 \cap L_2 = \overline{\overline{L_1} \cup \overline{L_2}}$$

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  $G_1 = (V_{N_1}, X, S_1, P_1)$ ,  $G_2 = (V_{N_2}, X, S_2, P_2)$ , pak  $L = L_1L_2$  je také bezkontextový. Opět předpokládáme  $V_{N_1} \cap V_{N_2} = \emptyset$ . Gramatika, generující  $L$ , je

$$G = (V_{N_1} \cup V_{N_2} \cup \{S\}, X, S, P_1 \cup P_2 \cup \{S \rightarrow S_1S_2\})$$

pro  $S \notin V_{N_1} \cup V_{N_2}$ . Opět je triviální, že  $G$  je bezkontextová a  $L(G) = L_1L_2$ . Bezkontextové jazyky jsou tedy uzavřené na operaci konkatenace.

<sup>31</sup>připomeňme si, že  $\bar{L}$  značí doplňěk jazyka  $L$

**Iterace (operace \*)** Necht'  $L$  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

$$P' = P \cup \{S_1 \rightarrow \Lambda, S_1 \rightarrow SS_1\}$$

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

$$P' = \{X \rightarrow \alpha^R; (X \rightarrow \alpha) \in P\}.$$

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<sup>1</sup>. 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
- $\Sigma$  zásobníková abeceda
- $\delta : Q \times X_\Lambda \times \Sigma \rightarrow \mathcal{P}(Q \times \Sigma^*)$  přechodová funkce
- $q_0 \in Q$  iniciální stav
- $r_0 \in \Sigma$  iniciální zásobníkový symbol
- $F \subseteq Q$  množina přijímacích stavů

<sup>1</sup>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  $\delta$  definována tak, že posun hlavy není zřejmý z oboru hodnot. Zato je v definičním oboru  $X_\Lambda$  místo  $X$ , neboli existují přechody funkce  $\delta$  přes prázdné slovo (tzv.  $\Lambda$ -přechody), při kterých se hlava na vstupní pásce nepohne (čte  $\Lambda$ ).

Také je asi moudré zdůraznit, že ZA v každém kroku přečte právě jedno písmeno ze zásobníku a na zásobník zapíše slovo z  $\Sigma^*$ .

**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  $(q, \alpha, \beta)$ , kde  $q \in Q$ ,  $\alpha \in X^*$ ,  $\beta \in \Sigma^*$ .  $\mathcal{A}$  se nachází v konfiguraci  $(q, \alpha, \beta)$ , pokud je ve stavu  $q$ , na vstupní pásce zbývá přečíst slovo  $\alpha$  a na zásobníku je uloženo  $\beta$  (tzn. na vrcholu zásobníku je první písmeno slova  $\beta$ ).
- $\mathcal{A}$  přímo přejde z konfigurace  $(q, \alpha, \beta)$  do konfigurace  $(q', \alpha', \beta')$  (zapišeme  $(q, \alpha, \beta) \vdash (q', \alpha', \beta')$ ), jakmile platí

$$\begin{aligned} \alpha &= a\alpha' \\ \beta &= b\beta'' \\ \beta' &= \gamma\beta'' & \beta, \beta', \beta'', \gamma \in \Sigma^*, b \in \Sigma, \alpha, \alpha' \in X^*, a \in X_\Lambda \\ (q', \gamma) &\in \delta(q, a, b) \end{aligned}$$

(zdůrazňuji možnost  $a = \Lambda$ )

- $\mathcal{A}$  přejde z  $(q_0, \alpha_0, \beta_0)$  do  $(q_n, \alpha_n, \beta_n)$  (píšeme  $(q_0, \alpha_0, \beta_0) \vdash_*^{\mathcal{A}} (q_n, \alpha_n, \beta_n)$ ), 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, zapišeme  $(q_0, \alpha_0, \beta_0) \vdash_*^{\mathcal{A}} (q_n, \alpha_n, \beta_n)$  (přechody probíhají v automatu  $\mathcal{A}$ ).

- Posloupnost konfigurací nazveme výpočtem automatu na vstupním slovem  $\alpha$ , pokud první konfigurace je  $(q_0, \alpha, r_0)$  a poslední má tvar  $(q, \Lambda, \beta)$ , 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  $\alpha$  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  $\alpha$  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

- $(\forall a \in X_\Lambda)(\forall q \in Q)(\forall r \in \Sigma) \quad (|\delta(q, a, r)| \leq 1)$
- $(\forall q \in Q)(\forall r \in \Sigma) \quad [(\delta(q, \Lambda, r) \neq \emptyset) \Rightarrow (\forall a \in X)[\delta(q, a, r) = \emptyset]]$

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  $r$   $\Lambda$ -přechod, pak neexistuje přechod přes písmeno z  $X$ . Naopak existuje-li alespoň jeden přechod přes písmeno z  $X$ , neexistuje  $\Lambda$ -přechod. Tím pádem pro daná  $q$  a  $r$  hlava *bud'* zůstane stát ( $\Lambda$ -přechod), *nebo* se pohne o políčko vpravo (čte písmeno z  $X$ ).<sup>2</sup>

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

$$\begin{aligned} \Sigma_1 &= \Sigma \cup \{r_1\} & r_1 &\notin \Sigma \\ Q_1 &= Q \cup \{q_1, q_2\} & q_1, q_2 &\notin Q \\ \delta_1(q, a, r) &= \begin{cases} \delta(q, a, r) & \forall q \in Q, r \in \Sigma, a \in X_\Lambda \\ \{(q_0, r_0 r_1)\} & \text{pro } q = q_1, a = \Lambda, r = r_1 \\ \{(q_2, \Lambda)\} & \text{pro } a = \Lambda, r = r_1, \forall q \in Q \\ \emptyset & \text{jinak} \end{cases} \end{aligned}$$

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  $\delta_1$ ). 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  $\alpha$

$$(38) \quad (q_0, \alpha, r_0) \stackrel{\mathcal{A}}{\vdash}_* (q, \Lambda, \Lambda)$$

pro nějaké  $q \in Q$ . Pak ale

$$(q_1, \alpha, r_1) \stackrel{\mathcal{B}}{\vdash} (q_0, \alpha, r_0 r_1) \stackrel{\mathcal{B}}{\vdash}_* (q, \Lambda, r_1) \stackrel{\mathcal{B}}{\vdash} (q_2, \Lambda, \Lambda)$$

je nutně výpočet, kterým  $\mathcal{B}$  přijímá  $\alpha$  přijímacím stavem<sup>3</sup>, tudíž  $\alpha \in L(\mathcal{B})$  a  $N(\mathcal{A}) \subseteq L(\mathcal{B})$ .

<sup>2</sup>poslední souvětí je v ostře vylučovacím poměru (XOR)

<sup>3</sup>První a poslední přímý přechod vyplývá rovnou z definice  $\delta_1$ , prostřední přechod z (38), neboť  $r_1$  zůstává stále na dně zásobníku  $\mathcal{B}$ .

2. Máme-li  $\alpha \in L(\mathcal{B})$ , pak

$$(q_1, \alpha, r_1) \stackrel{\mathcal{B}}{\vdash} (q_0, \alpha, r_0 r_1) \stackrel{\mathcal{B}}{\vdash}_* (q, \alpha', \beta') \stackrel{\mathcal{B}}{\vdash} (q_2, \Lambda, \beta)$$

je výpočet  $\mathcal{B}$  nad  $\alpha$ . Z definice  $\delta_1$  (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  $\Lambda$  — neboli  $\beta = \Lambda$ . Přitom celý druhý přechod probíhal z definice  $\delta_1$  (první řádek) podle automatu  $\mathcal{A}$ , až na  $r_1$  na dně zásobníku, ale na ten se v druhém přechodu stejně nesáhlo. Tudíž

$$(q_0, \alpha, r_0) \stackrel{\mathcal{A}}{\vdash}_* (q, \Lambda, \Lambda)$$

je výpočet nad  $\alpha$  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  $\Lambda$ -přechody ve stavu  $q_1$ . Automat  $\mathcal{B}$  nepotřebuje žádné přijímací stavy (neboť přijímá prázdným zásobníkem), nuže definujeme

$$\mathcal{B} = (Q \cup \{q_1, q_2\}, X, \Sigma \cup \{r_1\}, \delta_1, q_2, r_1, \emptyset)$$

pro  $q_1, q_2 \notin Q$ ,  $r_1 \notin \Sigma$ , kde

$$\begin{aligned} \delta_1(q, a, r) &= \begin{cases} \delta(q, a, r) & \forall q \in Q \setminus F, a \in X_\Lambda, r \in \Sigma \\ \text{nebo } q \in F, a \in X, r \in \Sigma \\ \delta(q, a, r) \cup \{(q_1, \Lambda)\} & \text{pro } q \in F, a = \Lambda, r \in \Sigma \end{cases} \\ \delta_1(q, \Lambda, r_1) &= \{(q_1, r_1)\} \text{ pro } q \in F \\ \delta_1(q_1, \Lambda, r) &= \{(q_1, \Lambda)\} \text{ pro } r \in \Sigma \cup \{r_1\} \\ \delta_1(q_2, \Lambda, r_1) &= \{(q_0, r_0 r_1)\} \\ \delta_1(q, a, r) &= \emptyset \text{ v ostatních případech} \end{aligned}$$

Funkce  $\delta_1$  je definována tak, aby pro stavy z  $Q \setminus F$  kopírovala výpočet v  $\mathcal{A}$ , pro stavy z  $F$  měla možnost provést  $\Lambda$ -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  $(q_0, \alpha, r_0) \stackrel{\mathcal{A}}{\vdash}_* (q, \Lambda, \beta)$  přijímací výpočet v  $\mathcal{A}$ , tedy

$$(39) \quad (q_2, \alpha, r_1) \stackrel{\mathcal{B}}{\vdash} (q_0, \alpha, r_0 r_1) \stackrel{\mathcal{B}}{\vdash}_* (q, \Lambda, \beta r_1) \stackrel{\mathcal{B}}{\vdash} (q_1, \Lambda, \beta_1 r_1) \stackrel{\mathcal{B}}{\vdash}_* (q_1, \Lambda, \Lambda)$$

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  $\delta_1$ , kde se na  $r_1$  nesahá, a stav  $q_1$  se v těchto přechodech nevyskytl, tedy zde  $\delta_1 = \delta$  a tudíž  $(q_0, \alpha, r_0) \stackrel{\mathcal{A}}{\vdash}_* (q, \Lambda, \beta)$  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  $\mathcal{A}$ .

$$V_N = \{S\} \cup \{(q, r, q'); q, q' \in Q, r \in \Sigma\}$$

$$P \ni \begin{cases} S \rightarrow (q_0, r_0, q) & \forall q \in Q \\ (q, r, q') \rightarrow a \text{ pokud } (q', \Lambda) \in \delta(q, a, r) \\ (q, r, q') \rightarrow a(q_1, r_1, q_2)(q_2, r_2, q_3) \dots (q_k, r_k, q') \\ \text{pro } k > 0 \text{ a } \forall q_1, \dots, q_k \in Q \\ \text{pokud } (q_1, r_1 \dots r_k) \in \delta(q, a, r) \end{cases}$$

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  $P$  přidávají jen tehdy, když existují slova  $\alpha_1, \dots, \alpha_k \in X^*$  taková, že

$$(q_i, \alpha_i \dots \alpha_k \beta, r_i \dots r_k \sigma) \stackrel{\mathcal{A}}{\vdash}_* (q_{i+1}, \alpha_{i+1} \dots \alpha_k \beta, r_{i+1} \dots r_k \sigma)$$

$$(q_k, \alpha_k \beta, r_k \sigma) \stackrel{\mathcal{A}}{\vdash}_* (q', \beta, \sigma)$$

jsou pro  $i = 1, \dots, (k-1)$  výpočty  $\mathcal{A}$  pro libovolná  $\beta \in X^*$ ,  $\sigma \in \Sigma^*$ . Ne-terminál  $(q, r, q')$  explicitně řečeno znamená, že  $\mathcal{A}$  je ve stavu  $q$ , na vrcholu zásobníku má  $r$  a výpočet odstraňující  $r$  z vrcholu zásobníku<sup>5</sup> skončí v  $q'$ <sup>6</sup>.  $G$  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  $r \in \Sigma$*

$$(q, r, q') \stackrel{G}{\underset{*}{\Rightarrow}} \alpha \iff (q, \alpha, r) \stackrel{\mathcal{A}}{\vdash}_* (q', \Lambda, \Lambda).$$

<sup>4</sup> $q$  musel být nutně z  $F$ , neboť to je jediná možnost, jak se dostat do stavu  $q_1$  v následujícím přechodu v (39)

<sup>5</sup>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  $\beta$

<sup>6</sup>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

$$(q, r, q') \xrightarrow{G} \alpha \Leftrightarrow \alpha \in X_\Lambda \wedge (q', \Lambda) \in \delta(q, \alpha, r) \Leftrightarrow (q, \alpha, r) \vdash^A (q', \Lambda, \Lambda)$$

Nyní druhý krok důkazu implikace zleva doprava. Mějme

$$(q, r, q') \Longrightarrow a(q_1, r_1, q_2) \dots (q_k, r_k, q') \xrightarrow{*} \alpha.$$

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  $(q_i, r_i, q_{i+1}) \xrightarrow{*} \alpha_i^7$  a  $\alpha = a\alpha_1\alpha_2 \dots \alpha_k$ . Derivace  $\alpha_i$  jsou však nejméně o jeden krok kratší než derivace  $\alpha$ , můžeme tedy použít indukční předpoklad

$$(40) \quad (q_i, r_i, q_{i+1}) \xrightarrow{*} \alpha_i \Leftrightarrow (q_i, \alpha_i, r_i) \vdash^* (q_{i+1}, \Lambda, \Lambda)$$

pro všechna  $i = 1, \dots, k$ , z čehož

$$(q, \alpha, r) \vdash (q_1, \alpha_1 \dots \alpha_k, r_1 \dots r_k) \vdash^* (q_2, \alpha_2 \dots \alpha_k, r_2 \dots r_k) \vdash^* \dots \\ \dots \vdash^* (q_k, \alpha_k, r_k) \vdash^* (q', \Lambda, \Lambda).$$

Tím je dokázána první implikace.

Druhý krok druhé implikace (zprava doleva) bude velmi podobný. Víme, že

$$(q, \alpha, r) \vdash (q_1, \alpha', r_1 \dots r_k) \vdash^* (q', \Lambda, \Lambda)$$

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

$$(q, \alpha, r) \vdash (q_1, \beta_1, r_1 \dots r_k) \vdash^* (q_2, \beta_2, r_2 \dots r_k) \vdash^* \dots \\ \dots \vdash^* (q_k, \beta_k, r_k) \vdash^* (q', \Lambda, \Lambda).$$

Z toho vyplývá, že  $G$  obsahuje pravidlo

$$(q, r, q') \rightarrow a(q_1, r_1, q_2) \dots (q_k, r_k, q').$$

Definujme  $\alpha_i$  pro  $i = 1, \dots, (k-1)$  tak, že  $\beta_i = \alpha_i\beta_{i+1}$  a  $\alpha_k = \beta_k$ . Potom zřejmě  $(q_i, \alpha_i, r_i) \vdash^* (q_{i+1}, \Lambda, \Lambda)$  pro  $i \leq k$ . Výpočty nad  $\alpha_i$  jsou zřejmě nejméně o jeden krok kratší, než výpočty nad  $\alpha$ , z indukčního předpokladu tedy víme (40), z čehož

$$(q, r, q') \Longrightarrow a(q_1, r_1, q_2) \dots (q_k, r_k, q') \xrightarrow{*} a\alpha_1\alpha_2 \dots \alpha_k = a\alpha' = \alpha.$$

<sup>7</sup>s tím, že  $q' = q_{k+1}$

<sup>8</sup> $\beta_{i+1}$  je vždy suffix  $\beta_i$

Tím je pomocné lemma dokázáno. CBD

Pro každé  $\alpha \in X^*$  z definice  $P$  a lemmatu 6.8 plyne

$$\begin{aligned} \alpha \in L(G) &\Leftrightarrow (S \xrightarrow{G} (q_0, r_0, q) \xrightarrow{*} \alpha) \Leftrightarrow ((q_0, \alpha, r_0) \vdash_*^{\mathcal{A}} (q, \Lambda, \Lambda)) \Leftrightarrow \\ &\Leftrightarrow \alpha \in N(\mathcal{A}), \end{aligned}$$

pro nějaké  $q \in Q$ , neboli  $L(G) = N(\mathcal{A})$ .

2. Mějme bezkontextovou gramatiku  $G = (V_N, V_T, S, P)$ . Definujeme jednotavový nedeterministický zásobníkový automat

$$\mathcal{A} = (\{q_0\}, V_T, V_T \cup V_N, \delta, q_0, S, \emptyset),$$

kde

$$\begin{aligned} (q_0, X_1 \dots X_k) &\in \delta(q_0, \Lambda, X) \text{ pokud } (X \rightarrow X_1 \dots X_k) \in P \\ &\text{pro } X \in V_N, X_1, \dots, X_k \in V_T \cup V_N, k \geq 0 \\ (q_0, \Lambda) &\in \delta(q_0, a, a) \text{ pro } a \in V_T \end{aligned}$$

Automat  $\mathcal{A}$  generuje na zásobníku postupně derivaci vstupního slova (označme ho  $\alpha$ ) 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  $\alpha$ , neboli výpočet skončí neúspěchem<sup>9</sup>. 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**  $(q_0, \alpha, \beta) \vdash_*^{\mathcal{A}} (q_0, \Lambda, \Lambda) \iff \beta \xrightarrow{*} \alpha$

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  $\delta$  právě tehdy, když buď  $\alpha = \beta \in V_T$  (potom zřejmě  $\beta \xrightarrow{*} \alpha$ ), nebo  $\alpha = \Lambda, \beta \in V_N$  a  $(\beta \rightarrow \alpha) \in P$ , čili také  $\beta \xrightarrow{*} \alpha$ , čímž máme první krok. Ve výpočtu

$\overbrace{(q_0, \alpha, \beta) \vdash_*^{\mathcal{A}} (q_0, \Lambda, \Lambda)}^V$  delším než jeden krok mohou nastat z definice  $\delta$  dva případy:

- $(q_0, \alpha, \beta) \vdash \overbrace{(q_0, \alpha, \gamma\beta') \vdash_*^{\mathcal{A}} (q_0, \Lambda, \Lambda)}^{V_1}$ , kde  $\beta = X\beta'$ ,  $(X \rightarrow \gamma) \in P$ . Výpočet  $V_1$  je ovšem kratší než  $V$ , neboli z indukčního předpokladu  $\beta = X\beta' \xrightarrow{*} \gamma\beta' \xrightarrow{*} \alpha$ .

<sup>9</sup> $\mathcal{A}$  je však nedeterministický, proto není třeba zoufat. Může uspět jiný výpočet.

- $(q_0, \alpha, \beta) \vdash \overbrace{(q_0, \alpha', \beta')}^{V2} \vdash_* (q_0, \Lambda, \Lambda)$ , kde  $\beta = a\beta'$ ,  $\alpha = a\alpha'$ ,  $a \in V_T$ . Výpočet  $V2$  je ovšem kratší než  $V$ , neboli z indukčního předpokladu  $\beta = a\beta' \xRightarrow{*} a\alpha' = \alpha$ .

Zprava doleva. Je-li  $\beta = \alpha \in V_T^*$ , potom opakovaným použitím druhého řádku definice  $\delta$  dostaneme  $(q_0, \alpha, \beta) \vdash_* (q_0, \Lambda, \Lambda)$ . Je-li  $\beta \xRightarrow{*} \alpha$  odvození délky alespoň 1, pak rozdělíme  $\beta$  na  $\beta = \beta' X \beta''^{10}$ . Zřejmě platí  $\alpha = \beta' \alpha'$ ,

můžeme tedy přejít k derivaci  $\overbrace{X \beta'' \xRightarrow{*} \alpha'}^O$  místo  $\beta \xRightarrow{*} \alpha$ . Protože  $G$  je bezkontextová, jistě pro nějaké pravidlo  $(X \rightarrow \gamma) \in P$  platí

$$X \beta'' \xRightarrow{*} \overbrace{\gamma \beta'' \xRightarrow{*} \alpha'}^{O1}.$$

Derivace  $O1$  je kratší než  $O$ , použijeme tudíž indukční předpoklad, který nám říká, že  $(q_0, \alpha', \gamma \beta'') \vdash_* (q_0, \Lambda, \Lambda)$ , a dostáváme

$$(q_0, \alpha, \beta) = (q_0, \beta' \alpha', \beta' X \beta'') \vdash_* (q_0, \alpha', X \beta'') \vdash_* (q_0, \alpha', \gamma \beta'') \vdash_* (q_0, \Lambda, \Lambda)$$

(Na odstranění  $\beta'$  jsme použili druhý řádek definice  $\delta$ .) Tím je lemma 6.9 dokázáno. CBD

Pro libovolné  $\alpha \in V_T^*$  z lemmatu 6.9 vyplývá

$$\alpha \in L(G) \Leftrightarrow S \xRightarrow{*} \alpha \Leftrightarrow \left( (q_0, \alpha, S) \vdash_*^A (q_0, \Lambda, \Lambda) \right) \Leftrightarrow \alpha \in N(\mathcal{A}),$$

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.*

<sup>10</sup>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<sup>11</sup>. Je-li  $\alpha \in N(\mathcal{A})$ , pak musí existovat výpočet  $\mathcal{A}$  nad  $\alpha$ , který skončí s prázdným zásobníkem po přečtení celého  $\alpha$ . Během tohoto výpočtu musí být na zásobníku v každém kroku alespoň jeden znak. Pokud  $\mathcal{A}$  je deterministický, pak je výpočet nad  $\alpha$  právě jeden a zásobník se vyprázdní až po přečtení  $\alpha$ . 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** *Doplňěk 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 doplňěk. Není to bohužel tak jednoduché. Jedním z důvodů je například fakt, že výpočet  $\mathcal{A}$  se nad nějakým slovem  $\alpha$  může dostat do cyklu  $\Lambda$ -kroků a nikdy neskončí. Takové  $\alpha$  by jistě nepatřilo do  $L$  ani do  $\bar{L}$ , protože výpočet neskončil v žádném ze 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.

$$\begin{aligned} L_1 &= \{a^i b^j c^k; i, j = 1, \dots\} \\ L_2 &= \{a^i b^j c^k; i, j = 1, \dots\} \\ L &= L_1 \cap L_2 = \{a^i b^i c^i; i = 1, \dots\} \end{aligned}$$

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.<sup>12</sup>

<sup>11</sup>neboť ZA mají z definice povinnost číst ze zásobníku

<sup>12</sup>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  $\mathcal{A}_1$ .

$$\begin{aligned} & ((q', p'), \beta) \in \delta_2((q, p), a, r), \text{ právě když} \\ & \left( (p' \in \delta_1(p, a) \wedge a \neq \Lambda) \vee (p' = p \wedge a = \Lambda) \right) \wedge \left( (q', \beta) \in \delta(q, a, r) \right) \end{aligned}$$

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  $\delta_1$  a v případě, že  $\mathcal{A}_2$  čte  $\Lambda$ , zůstává stát. Výpočet  $\mathcal{A}_2$  je přijímací, pokud simulace v obou složkách skončí v přijímacích stavech. Odtud  $L \cap R = L(\mathcal{A}_2)$ .<sup>13</sup>

Dále dokážeme, že  $R \setminus L$  je bezkontextový jazyk. Idea bude podobná jako pro průnik, jen neznámou část slova z  $L$  musíme uhodnout, ale od toho máme nedeterminismus. Definujme  $\mathcal{Z}\mathcal{A}$

$$\mathcal{A}_3 = (Q \times Q_1 \cup Q, X, \delta_3, (q_0, q_1), r_0, F),$$

kde  $\delta_3$  je dána předpisem

- $\delta_3((q, p), \Lambda, r) = A \cup \{((q', p), \beta); (q', \beta) \in \delta(q, \Lambda, r)\} \cup \{((q', p'), \beta); (\exists y \in X)[(q', \beta) \in \delta(q, y, r) \wedge p' \in \delta_1(p, y)]\},$

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  $\mathcal{A}$ <sup>14</sup> a

<sup>13</sup>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  $\mathcal{A}_2$ ?

<sup>14</sup>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  $L(\mathcal{A}_3) = R \setminus L$ .

Podobně dokážeme, že  $L/R$  je bezkontextový jazyk. Definujme  $\mathcal{Z}\mathcal{A}$

$$\mathcal{A}_4 = (Q \times Q_1 \cup Q, X, \delta_4, (q_0, q_1), r_0, F \times F_1),$$

kde  $\delta_3$  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  $\delta_4$ . 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  $\delta_4$ . 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  $a_1, \dots, a_k \in V$  a  $V_1, \dots, V_k \subseteq V$  vektorové podprostory  $V$  takové, že  $\vec{a} \in A$  právě když každá souřadnice vektoru  $\vec{a}$  je nezáporná a  $\vec{a} \in \bigcup_{i=1}^k (a_i + V_i)$ .*

Je-li  $A = \{a_1, \dots, a_n\}$  a  $\alpha \in A^*$ , označme  $p_{a_i}(\alpha)$  počet výskytů prvku  $a_i$  v  $\alpha$  a  $v(\alpha) = (p_{a_1}(\alpha), \dots, p_{a_n}(\alpha))$  vektor výskytů jednotlivých písmen z  $A$  v  $\alpha$ . 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
- $\delta : Q \times Y \rightarrow Q \times Y \times \{-1, 0, 1\}$  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  $\delta$  v deterministickém Turingově stroji definujeme často jako  $\delta : (Q \setminus F) \times Y \rightarrow Q \times Y \times \{-1, 0, 1\}$ . 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  $(\alpha, q, \beta)$ , kde  $\alpha, \beta \in Y^*, q \in Q$ . Řekneme, že *T se nachází v konfiguraci*  $(\alpha, q, \beta)$ , 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  $\beta$ , jsou ekvivalentní. Proto vezte, 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*  $(\alpha, q, \beta)$  *do konfigurace*  $(\gamma, r, \sigma)$  *(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  $(\alpha, q, \beta)$  do  $(\gamma, r, \sigma)$  (píšeme  $(\alpha, q, \beta) \vdash_*(\gamma, r, \sigma)$ ), pokud existuje posloupnost přímých přechodů začínající v  $(\alpha, q, \beta)$  a končící v  $(\gamma, r, \sigma)$ .*

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  $(a, q_0, \alpha) \vdash_*(B, q, \Lambda)$  pro nějaké  $q \in F$ .*

*$T$  přijímá  $\Lambda$ , když  $(B, q_0, \Lambda) \vdash_*(B, q, \Lambda)$  pro nějaké  $q \in F$ .<sup>2</sup>*

Slovo  $\alpha$  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<sup>3</sup>.

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<sup>4</sup>.

## 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  $\Lambda$ -přechody<sup>5</sup> 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  $(a, q_0, \alpha) \vdash_*(\beta, q, \gamma)$ ,  $q \in F$ , 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

<sup>1</sup>pro nějaké  $a \in X$ ,  $\alpha \in X^*$

<sup>2</sup>Místo  $\Lambda$  zde může být libovolné slovo z Blanků, záleží na kandidátovi vybraném ze třídy ekvivalentních konfigurací.

<sup>3</sup>Definice akceptující konfigurace se často v různých učených knihách liší, nicméně na třídy přijímaných jazyků to nemá vliv.

<sup>4</sup>takové důkazy budeme označovat CBD/2, abychom je odlišili od těch „korektnějších“

<sup>5</sup> $\Lambda$ -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 ( $k < +\infty$ )<sup>6</sup>. 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

$$\delta : (Q \setminus F) \times Y^k \rightarrow Q \times Y^k \times \{-1, 0, 1\}^k,$$

resp.

$$\delta : (Q \setminus F) \times Y^k \rightarrow Q \times (Y \times \{-1, 0, 1\})^k.$$

Konfigurace  $k$ -páskového stroje je  $(\alpha_i, q, \beta_i)_{i=1}^k$ <sup>7</sup>. 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 pásece 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  $({}_i y_j, {}_i u_j)_{j=1}^k$ , kde<sup>8</sup>  ${}_i y_j$  je  $i$ -té písmeno slova  $\alpha_j \beta_j B^*$  a  ${}_i u_j = 1$ , pokud hlava na  $j$ -té pásece je na  $i$ -té pozici (jinak  ${}_i u_j = 0$ )<sup>9</sup>. Podrobnosti o  $Q'$ ,  $F'$  a  $\delta'$  si odpustím.  $T.s.$   $T$  pracuje podle následujícího algoritmu:

1.  $T$  přepíše svoji pásku tak, aby kódovala pásky<sup>10</sup>  $T_k$ , 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.

<sup>6</sup>Tzn. pásek je konečně mnoho.

<sup>7</sup>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

<sup>8</sup>zápis  ${}_i u_j$  není nějaký standartní, jen jsem nevěděl, kam umístit index  $i$ , aby nevypadal jako mocnina nebo index dvourozměrného pole

<sup>9</sup>je vhodné si uvědomit, že pro pevné  $j$  existuje právě jedno  $i$  takové, že  ${}_i u_j = 1$

<sup>10</sup> $T$  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  $\delta'$  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<sup>11</sup> je definován jako deterministický. Rozdíl je opět pouze v

$$\delta : (Q \setminus F) \times Y \rightarrow \mathcal{P}(Q \times Y \times \{-1, 0, 1\}).$$

Slovo  $\alpha$  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  $\Lambda$  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  $((r, a, i) \text{ je členem } O_{q,y}) \Leftrightarrow ((r, a, i) \in \delta(q, y))$ <sup>12</sup>.

Na třetí pásce je slovo z  $\{1, \dots, k\}$  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.<sup>13</sup>

$T'$  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 pásy, nastaví hlavy na začátek a přejde do přijímacího stavu.

$T'$  je deterministický přijímá stejný jazyk jako  $T_N$ . CBD/2

<sup>11</sup>Pro zkrácení budeme často psát  $N$ -T.s. a  $D$ -T.s. místo nedeterministický a deterministický Turingův stroj.

<sup>12</sup> $O_{q,y}$  se liší od  $\delta(q, y)$  tím, že je uspořádaná

<sup>13</sup>Pro odteknění. Na třetí pásce je slovo  $l_1 \dots l_i$ , kde  $l_j \in \{1, \dots, k\}$ . 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  $\delta(q, B) = (q', B, -1)$ <sup>14</sup>.*

**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  $(a, q_0, \alpha) \vdash_*(B, q_f, \Lambda)$  generovat postupně odzadu jednotlivé konfigurace  $T$ , až se dostane k výchozí konfiguraci, ze které nakonec vypudí  $q_0$  a získá vstupní slovo<sup>15</sup>.  $B_1$  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.  $(S \rightarrow B_0 B q_f B_1)$
2.  $(rxy \rightarrow zqy)$ , pokud  $\delta(q, z) = (r, x, -1)$
3.  $(xry \rightarrow zqy)$ , pokud  $\delta(q, z) = (r, x, 0)$
4.  $(xyr \rightarrow zqy)$ , pokud  $\delta(q, z) = (r, x, +1)$
5.  $(rxB_1 \rightarrow zqB_1)$ , pokud  $\delta(q, z) = (r, x, -1)$
6.  $(rB_1 \rightarrow zqB_1)$ , pokud  $\delta(q, z) = (r, B, -1)$
7.  $(xrrB_1 \rightarrow zqB_1)$ , pokud  $\delta(q, z) = (r, x, 0)$
8.  $(xBrB_1 \rightarrow zqB_1)$ , pokud  $\delta(q, z) = (r, x, +1)$
9.  $(B_0 x q_0 \rightarrow xM)$ , pokud  $x \in X$
10.  $(B_0 B q_0 \rightarrow M)$
11.  $(Mx \rightarrow xM)$
12.  $(MB_1 \rightarrow \Lambda)$
13.  $(MB \rightarrow M)$

Pokud  $a\alpha \in L(T)$ , existuje přijímací výpočet  $(a, q_0, \alpha B_1) \vdash_*(B, q_f, B_1)$ , což odpovídá v  $G$  derivaci

$$S \Longrightarrow B_0 B q_f B_1 \xRightarrow{*} B_0 a q_0 \alpha B_1 \Longrightarrow a M \alpha B_1 \xRightarrow{*} a \alpha M B_1 \Longrightarrow a \alpha \in L(G)$$

Pokud  $\Lambda \in L(T)$ , existuje přijímací výpočet  $(B, q_0, \Lambda) \vdash_*(B, q_f, \Lambda)$ , což v  $G$  odpovídá derivaci

$$S \Longrightarrow B_0 B q_f B_1 \xRightarrow{*} B_0 B q_0 B_1 \Longrightarrow M B_1 \Longrightarrow \Lambda \in L(G)$$

<sup>14</sup>Tato druhá možnost je nutná, aby T.s. měl vůbec šanci poznat konec vstupního slova

<sup>15</sup>velmi názorně je to vidět v příkladu 7.18

Je-li  $\alpha \in L(G)$ , existuje odvození  $S \Rightarrow \gamma_1 \Rightarrow \dots \Rightarrow \gamma_k = \alpha$ , kde pro nějaké  $i$  slovo  $\gamma_i$  obsahuje nějaký stav a  $\gamma_{i+1}$  již ne. Stavem obsaženým v  $\gamma_i$  musel být z definice  $G$  stav  $q_0$  bezprostředně následující po prvním písmenu  $\gamma_i$ <sup>16</sup>, neboli  $\gamma_i \stackrel{T}{\vdash} \gamma_1$  je přijímací výpočet  $T$  nad  $\alpha$ , 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  $\alpha = a_1 \dots a_n \in (V_N \cup X)^*$ <sup>17</sup> vygenerované gramatikou  $G$  bez použití pravidla 9, resp. 10 transformujeme na slovo  $\alpha^T = b_1 \dots b_{n-3} \in V'_N$  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<sup>18</sup>.

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  $\alpha$ , pak se v pozměněné gramatice  $G^T$  vygeneruje slovo  $\alpha^T$ <sup>19</sup>. Zbylá pravidla z gramatiky  $G$  vyřadíme a do  $G^T$  přidáme pravidla

- $(a, q_0, 0, 1)(b, 0, 0, 0) \rightarrow ab$  pro  $a, b \in X$

<sup>16</sup>nepočítaje  $B_0$

<sup>17</sup> $V'_N = \{B_0, B_1, S, M\} \cup Q \cup Y$

<sup>18</sup>komu se správně zdá, že ve slově  $\alpha^T$  zůstanou mezery po  $B_0, B_1$  a stavu z  $Q$ , může být klidný — po skončení nahrazování tyto mezery vyhodíme a získáme slovo  $\alpha^T$  délky  $n - 3$  bez mezer.

<sup>19</sup>například pravidla typu 5 přejdou na pravidla typu

$$\{(u, r, 0, i)(x, 0, 1, 0) \rightarrow (u, 0, 0, i)(z, q, 1, 0); \\ i \in \{0, 1\}, u, x, z \in Y, r, q \in Q, \delta(q, z) = (r, x, -1)\}$$

- $(a, q_0, 0, 1)(b, 0, 1, 0) \rightarrow ab$  pro  $a, b \in X$
- $(a, q_0, 1, 1) \rightarrow a$  pro  $a \in X$
- $a(b, 0, 0, 0) \rightarrow ab$  pro  $a, b \in X$
- $a(b, 0, 1, 0) \rightarrow ab$  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  $\alpha_i$  (neboli  $S \Rightarrow \alpha_i$ ) a  $\alpha_i$  obsahuje neterminál, pak  $T$  „uhodne“ pravidlo  $p$  použité k dalšímu generování a místo použití v  $\alpha_i$  a přepíše část druhé pásky podle  $p$ , čímž dostane další slovo  $\alpha_{i+1}$ .
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<sup>21</sup> 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<sup>22</sup>. 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*<sup>23</sup>:

<sup>20</sup> $T$  je nedeterministický proto, že  $G$  může obsahovat více pravidel s jednou levou stranou.

<sup>21</sup>Nezkracující gramatiky generují právě kontextové jazyky.

<sup>22</sup>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.

<sup>23</sup>Linear Bounding Automaton — 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  $\alpha$  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  $\bar{L}$  (doplňěk) 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<sup>24</sup> 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<sup>25</sup>, tedy by  $L$  i  $\bar{L}$  byly rekursivně spočetné<sup>26</sup>, 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“.

<sup>24</sup>zastaví, ne pozastaví

<sup>25</sup>protože kontextové jazyky jsou uzavřeny na doplňky (viz tvrzení 7.14)

<sup>26</sup>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  $\delta : (Q \setminus F) \times Y \rightarrow Q \times Y \times \{-1, 0, 1\}$  je popsána tabulkou

$X \setminus Q$	$q_0$	$\bar{q}_0$	$q_1$	$q_c$
$c$	$q_c, \#, +1$	$q_c, B, +1$	$q_1, c, -1$	$q_c, c, +1$
$d$	$q_d, \#, +1$	$q_d, B, +1$	$q_1, d, -1$	$q_c, d, +1$
$B$	$q_f, B, 0$	$q_2, B, -1$	$\bar{q}_0, B, +1$	$q_1^c, B, -1$
$\#$	—	—	$\bar{q}_0, \#, +1$	—
	$q_d$	$q_1^c$	$q_1^d$	$q_2$
$c$	$q_d, c, +1$	$q_1, B, -1$	—	—
$d$	$q_d, d, +1$	—	$q_1, B, -1$	—
$B$	$q_1^d, B, -1$	$q_2, B, -1$	$q_2, B, -1$	$q_2, B, -1$
$\#$	—	$q_f, B, 0$	$q_f, B, 0$	$q_f, B, 0$

$T$  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 změnil 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  $q_1$ )<sup>27</sup>. 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  $\{B, \#, c, d\}$ ,  $x \in \{c, d\}$

$$\begin{aligned} {}^1) S &\rightarrow B_0 B q_f B_1 \\ {}^2) q_1^c B y &\rightarrow B q_c y \\ q_1^d B y &\rightarrow B q_d y \\ q_1 B y &\rightarrow c q_1^c y \mid d q_1^d y \end{aligned}$$

<sup>27</sup>Není-li poslední písmeno shodné s prvním, pak se výpočet zastaví ( $\delta$  není definována).



$$\begin{aligned}
& q_1cy \rightarrow cq_1y \\
& q_1dy \rightarrow dq_1y \\
& q_2By \rightarrow B\bar{q}_0y \mid Bq_1^c y \mid Bq_1^d y \mid Bq_2y \\
& {}^3) Bq_f y \rightarrow Bq_0y \mid \#q_1^c y \mid \#q_1^d y \mid \#q_2y \\
& {}^4) Byq_c \rightarrow c\bar{q}_0y \\
& Byq_d \rightarrow d\bar{q}_0y \\
& cyq_c \rightarrow cq_cy \\
& dyq_c \rightarrow dq_cy \\
& cyq_d \rightarrow cq_dy \\
& dyq_d \rightarrow dq_dy \\
& By\bar{q}_0 \rightarrow Bq_1y \\
& \#yq_c \rightarrow cq_0y \\
& \#yq_d \rightarrow dq_0y \\
& \#y\bar{q}_0 \rightarrow \#q_1y \\
& {}^5) q_1^c BB_1 \rightarrow Bq_c B_1 \\
& q_1^d BB_1 \rightarrow Bq_d B_1 \\
& q_1 BB_1 \rightarrow cq_1^c B_1 \mid dq_1^d B_1 \\
& q_2 BB_1 \rightarrow B\bar{q}_0 B_1 \mid Bq_1^c B_1 \mid Bq_1^d B_1 \mid Bq_2 B_1 \\
& q_1 c B_1 \rightarrow cq_1 B_1 \\
& q_1 d B_1 \rightarrow dq_1 B_1 \\
& {}^6) q_1^c B_1 \rightarrow Bq_c B_1 \\
& q_1^d B_1 \rightarrow Bq_d B_1 \\
& q_1 B_1 \rightarrow cq_1^c B_1 \mid dq_1^d B_1 \\
& q_2 B_1 \rightarrow B\bar{q}_0 B_1 \mid Bq_1^c B_1 \mid Bq_1^d B_1 \mid Bq_2 B_1 \\
& {}^7) Bq_f B_1 \rightarrow Bq_0 B_1 \mid \#q_1^c B_1 \mid \#q_1^d B_1 \mid \#q_2 B_1 \\
& {}^8) BBq_c B_1 \rightarrow c\bar{q}_0 B_1 \\
& BBq_d B_1 \rightarrow d\bar{q}_0 B_1 \\
& cBq_c B_1 \rightarrow cq_c B_1 \\
& cBq_d B_1 \rightarrow cq_d B_1 \\
& dBq_d B_1 \rightarrow dq_d B_1 \\
& dBq_c B_1 \rightarrow dq_c B_1 \\
& BB\bar{q}_0 B_1 \rightarrow Bq_1 B_1 \\
& \#Bq_c B_1 \rightarrow cq_0 B_1 \\
& \#Bq_d B_1 \rightarrow dq_0 B_1 \\
& \#B\bar{q}_0 B_1 \rightarrow \#q_1 B_1 \\
& {}^9) B_0 x q_0 \rightarrow xM \\
& {}^{10}) B_0 B q_0 \rightarrow M \\
& {}^{11}) M c \rightarrow cM
\end{aligned}$$

$$\begin{aligned}
Md &\rightarrow dM \\
MB &\rightarrow BM \\
M\# &\rightarrow \#M \\
^{12)} MB_1 &\rightarrow \Lambda \\
^{13)} MB &\rightarrow M
\end{aligned}$$

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

$$(c, q_0, B\Lambda) \stackrel{T}{\vdash} (\#B, q_c, \Lambda) \stackrel{T}{\vdash} (\#, q_1^c, B\Lambda) \stackrel{T}{\vdash} (B, q_f, B\Lambda) \sim (B, q_f, \Lambda)$$

Teď stačí v podstatě vzít posloupnost konfigurací, obrátit ji, místo  $\vdash$  dát  $\xrightarrow{G}$  a dostáváme<sup>28</sup>

$$\begin{aligned}
S &\xrightarrow{1} B_0 B q_f B_1 \xrightarrow{7} B_0 \# q_1^c B_1 \xrightarrow{6} \\
&\xrightarrow{6} B_0 \# B q_c B_1 \xrightarrow{8} B_0 c q_0 B_1 \xrightarrow{9} c M B_1 \xrightarrow{12} c
\end{aligned}$$

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} &\xrightarrow{1} B_0 \underline{B q_f B_1} \xrightarrow{7} B_0 \# \underline{q_2 B_1} \xrightarrow{6} B_0 \# \underline{B q_2 B_1} \xrightarrow{6} B_0 \# \underline{B B q_2 B_1} \xrightarrow{6} \\
&\xrightarrow{6} B_0 \# \underline{B B B \bar{q}_0 B_1} \xrightarrow{8} B_0 \# \underline{B B q_1 B_1} \xrightarrow{6} B_0 \# \underline{B B c q_1^c B_1} \xrightarrow{6} \\
&\xrightarrow{6} B_0 \# \underline{B B c B q_c B_1} \xrightarrow{4} B_0 \# \underline{B B c q_c B B_1} \xrightarrow{4} \# \underline{B c \bar{q}_0 c B B_1} \xrightarrow{4} \\
&\xrightarrow{4} B_0 \# \underline{B q_1 c c B B_1} \xrightarrow{2} B_0 \# \underline{B c q_1 c B B_1} \xrightarrow{2} B_0 \# \underline{B c c q_1 B B_1} \xrightarrow{5} \\
&\xrightarrow{5} B_0 \# \underline{B c c d q_1^d B_1} \xrightarrow{6} B_0 \# \underline{B c c d B q_d B_1} \xrightarrow{4} B_0 \# \underline{B c c d q_d B B_1} \xrightarrow{4} \\
&\xrightarrow{4} B_0 \# \underline{B c c q_d d B B_1} \xrightarrow{4} B_0 \# \underline{B c q_d c d B B_1} \xrightarrow{4} B_0 \# \underline{d \bar{q}_0 c c d B B_1} \xrightarrow{4} \\
&\xrightarrow{4} B_0 \# \underline{q_1 d c c d B B_1} \xrightarrow{2} B_0 \# \underline{d q_1 c c d B B_1} \xrightarrow{2} B_0 \# \underline{d c q_1 c d B B_1} \xrightarrow{2} \\
&\xrightarrow{2} B_0 \# \underline{d c c q_1 d B B_1} \xrightarrow{2} B_0 \# \underline{d c c d q_1 B B_1} \xrightarrow{5} B_0 \# \underline{d c c d c q_1^c B_1} \xrightarrow{6} \\
&\xrightarrow{6} B_0 \# \underline{d c c d c B q_c B_1} \xrightarrow{4} B_0 \# \underline{d c c d c q_c B B_1} \xrightarrow{4} B_0 \# \underline{d c c d q_c c B B_1} \xrightarrow{4} \\
&\xrightarrow{4} B_0 \# \underline{d c c q_c d c B B_1} \xrightarrow{4} B_0 \# \underline{d c q_c c d c B B_1} \xrightarrow{4} B_0 \# \underline{d q_c c c d c B B_1} \xrightarrow{4} \\
&\xrightarrow{4} \underline{B_0 c q_0 d c c d c B B_1} \xrightarrow{9} \underline{c M d c c d c B B_1} \xrightarrow{11} \underline{c d M c c d c B B_1} \xrightarrow{11} \\
&\xrightarrow{11} \underline{c d c M c d c B B_1} \xrightarrow{11} \underline{c d c c M d c B B_1} \xrightarrow{11} \underline{c d c c d M c B B_1} \xrightarrow{11} \\
&\xrightarrow{11} \underline{c d c c d c M B B_1} \xrightarrow{13} \underline{c d c c d c M B_1} \xrightarrow{12} \underline{c d c c d c}
\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

<sup>28</sup>nad operátor  $\implies$  píše místo jména gramatiky číslo skupiny použitého pravidla

## 7.2 Univerzální Turingův stroj

Každý  $T.s.$ <sup>29</sup> 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  $\{0, 1\}$ <sup>30</sup>. Řekněme, že

$$\begin{aligned} q_0 &= 010 \\ q_1 &= 0110 \\ a &= 010 \\ b &= 0110 \\ B &= 01110 \\ \# &= 011110 \\ +1 &= 010 \\ 0 &= 0110 \\ -1 &= 01110 \end{aligned}$$

$$(\delta(q, x) = (q', y, j)) = \text{kód}(q)\text{kód}(x)\text{kód}(q')\text{kód}(y)\text{kód}(j)$$

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  $\delta$ , 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  $(\text{kód}(T); \alpha)$ <sup>31</sup> takové že  $T.s.$   $T$  přijímá slovo  $\alpha$ .*

<sup>29</sup>obecně jakýkoliv automat

<sup>30</sup>jak jinak, vždyť jsme informatici

<sup>31</sup>kde znak „,“ (středník) je nějaký jednoznačný oddělovač

---

$ \begin{array}{ccccccc} \overbrace{000\ 0100110000}^Q & \overbrace{0100110000}^X & \overbrace{01110011110}^{Y\setminus X} & \overbrace{000\ 0100100110011110010}^{\delta_{11}} \\ \delta_{12} & & \delta_{13} & \delta_{14} \\ \overbrace{01001100110011110010}^{\delta_{21}} & \overbrace{010011100100111001110}^{\delta_{21}} & \overbrace{010011110010011100110}^{\delta_{22}} \\ \overbrace{0110010011001110010}^{\delta_{23}} & \overbrace{0110010011001110010}^{q_0} & \overbrace{0110011001100100110}^F \\ \overbrace{0110011100100111001110000}^{\delta_{23}} & 010 & 000 & 010 & 000 \end{array} $
--

---

Tabulka 3: Kód stroje  $T$ . ( $\delta_{ij}$  znamená, že kódujeme  $i$ -tý řádek a  $j$ -tý sloupec tabulky zadání funkce  $\delta$ ).

Univerzální  $T$ .s existuje a jazyk, který přijímá, je

$$L_u = \{(\text{kód}(T); \alpha); T \text{ přijímá } \alpha\}$$

a je nesporně rekurzivně spočetný. Naznačíme způsob práce univerzálního Turingova stroje. Bude mít dvě pásy. Na první pásce má napsané vstupní slovo  $(\text{kód}(T); \alpha)$ . Nejprve ověříme, jestli je vstupní slovo formálně v pořádku (t.j. jestli kód má všechny komponenty dané předepsaným způsobem, jestli kód( $T$ ) a  $\alpha$  jsou odděleny předepsaným oddělovačem, atd. Pak přepíšeme  $\alpha$  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 pásy  $T$ . 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  $F$ )<sup>32</sup>. Dá se ukázat, že  $\bar{L}_u$  není rekurzivně spočetný, tzn. podle věty 7.16  $L_u$  není rekurzivní 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 rekurzivně spočetný, čímž pádem by tento odstavec přišel vniveč, uvedu jiný jazyk, o kterém dokážu, že skutečně není rekurzivní. Uvedený důkaz lze zmodifikovat tak, abychom dostali, že  $\bar{L}_u$  není rekurzivně spočetný.

Uvažujme jazyk

$$L'_u = \{\alpha; \text{stroj s kódem } \alpha \text{ přijímá slovo } \alpha\}$$

**Lemma 7.21**  $\bar{L}'_u$  není rekurzivně spočetný.

Důkaz. Provedeme sporem. Předpokládejme, že  $\bar{L}'_u$  je rekurzivně spočetný, tzn. existuje  $T$ .s  $T'$ , který ho přijímá ( $L(T') = \bar{L}'_u$ ). Pak  $\alpha \in \bar{L}'_u$ , právě když buď  $\alpha$  není kódem žádného stroje, nebo stroj s kódem  $\alpha$  slovo  $\alpha$  nepřijímá. Potom ale

<sup>32</sup> 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<sup>33</sup>.

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

$$\begin{aligned} L_{\text{all}}^G &= \{\text{kód}(G); L(G) = V_T^*\} \notin \mathcal{L}_0 \\ L_{\text{exists}}^G &= \{\text{kód}(G); L(G) \neq \emptyset\} \in \mathcal{L}_0 \\ L_{\text{reg}}^G &= \{\text{kód}(G); L(G) \in \mathcal{L}_3\} \notin \mathcal{L}_0 \\ L_{\text{fin}}^G &= \{\text{kód}(G); L(G) \text{ konečný}\} \notin \mathcal{L}_0 \end{aligned}$$

**Postův problém:** Vstupem pro Postův problém jsou posloupnosti slov  $\alpha_1, \dots, \alpha_n$  a  $\beta_1, \dots, \beta_n$ , kde  $\alpha_i, \beta_i \in X^*$  pro nějakou abecedu  $X$ . Postův problém má řešení, jakmile existuje posloupnost indexů  $i_1, \dots, i_r$  taková, že

$$\alpha_{i_1} \alpha_{i_2} \dots \alpha_{i_r} = \beta_{i_1} \beta_{i_2} \dots \beta_{i_r},$$

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: Vezmeme 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  $\Lambda$ -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

<sup>33</sup>Intuitivně — vezmeme modifikovaný univerzální T.s., který si vstupní slovo  $\alpha$  nejprve zduplikuje na  $(\alpha; \alpha)$  a pak teprve provede výpočet.

## Rejstřík

- abeceda, 3, 57
    - neterminální, 60
    - pracovní, 87
    - terminální, 60
    - vstupní, 5, 26, 38, 77, 87
    - výstupní, 5
    - zásobníková, 77
  - akceptor, 4, 25, 26, 46
    - D-, 38
    - deterministický, 33, 38
    - dosahující, 28, 29, 33
    - N-, 38
    - nedeterministický, 33, 38
    - podílový, 28
    - redukováný, 28, 29, 33
  - automat, 4, 65
    - dvoucestný, 47, 48, 53, 59, 60
      - deterministický, 48
      - nedeterministický, 48
    - dvousměrný, 47
    - konečný, 26
    - Mealy, 6, 8, 9
    - Moore, 5, 8, 9, 14, 26
    - podílový, 12, 13, 15, 21
    - redukováný, 10, 13, 14
    - zásobníkový, 76, 78–80, 84
      - deterministický, 78
      - nedeterministický, 77
  - blank, 87
  - dekompozice
    - paralelní, 16, 18, 23
    - sériová, 16, 23
  - délka
    - slova, 3
  - derivace, 60
    - levá, 72
    - minimální, 60
    - pravá, 72
  - doplněk
    - jazyka, 36, 76, 84, 94, 95
  - ekvivalence
    - $\Lambda$
  - stavů, 8
  - akceptorů, 26, 27, 29
  - gramatik, 61
  - lepící dva prvky, 11
  - přepisovacích systémů, 69
  - stavů, 7, 13, 15, 27
  - triviální, 16
- forma
- normální
    - Greibachova, 71
    - Hopfova, 71
    - Chomského, 70
  - standartní, 63
- funkce
- identická, 14
  - přechodová, 5, 26, 38, 77, 87, 88
  - překladová, 5
  - výstupní, 5
- gramatika, 4, 60, 62, 91, 93, 94, 96
- bezkontextová, 61, 63, 70–72
  - kontextová, 61, 69
  - nezkracující, 62, 68, 69, 91, 93
  - obecná, 61
  - regulární, 61, 63
    - levá, 62
  - typu 0, 61
- hierarchie
- Chomského, 61, 67, 74, 75, 100
- homomorfismus
- automatů, 14
- chování
- $\Lambda$ 
    - stejně, 8, 9
  - stejně, 8, 13, 14
- iterace
- jazyka, 41, 43, 76, 94
- izomorfismus
- akceptorů, 29
  - automatů, 15
- jazyk, 3, 95

- bezkontextový, 61, 80, 84, 85, 87
    - deterministický, 84, 85
  - bezprefixový, 84
  - elementární, 43
  - kontextový, 61, 68, 94, 95
  - přijímaný, 26, 38, 48, 89
    - přijímacím stavem, 78
    - s koncovými znaky, 59
    - s prázdným zásobníkem, 78
  - regulární, 26, 32, 36–38, 40, 41, 43, 48, 54, 57, 59–61, 65, 85, 87
  - rekursivně spočetný, 89, 90, 94, 95, 100, 101
  - rekursivní, 95, 100
  - rozhodovaný, 95
  - typu 0, 61
- konfigurace, 77, 88, 89
- kongruence, 31, 32, 35
  - automatová, 10, 11, 13, 21, 27
  - levá, 31
  - nejhrubší, 11
  - nejjemnější, 11
  - netriviální, 16
  - pravá, 31, 32, 35
  - stavová, 10, 11, 16, 18, 21, 23
  - syntaktická, 31, 32, 34, 35
    - levá, 31
    - pravá, 31, 32, 35
- konkatenace, 40
  - jazyků, 40, 43, 76, 94
  - slov, 3
- kvocient, 58, 59, 85
  - levý, 58
  - pravý, 58
- množina
  - potenční, 38
  - semilineární, 87
  - slov, 3
  - stavů, 5, 26, 38, 77, 87
- mocnina
  - jazyka, 41
- neterminál, 61, 71
- odvození, 60
  - přímé, 60
- operace
  - \*, 41
  - $R$ , 42
- písmeno, 3
- pravidlo, 60
- prefix, 34
  - vlastní, 34
- problém
  - Postův, 101
- průnik
  - jazyků, 36, 75, 85, 94, 101
- přechod, 77, 88
  - $\Lambda$ , 77, 89
  - přímý, 77, 88
- přepisovací systém, 60
- realizace, 16, 24
- relace
  - identická, 8
  - přechodová, 38
- restrikce, 27
- sjednocení
  - jazyků, 36, 43, 75, 85, 94
- slovo, 3
  - prázdné, 3
  - přijímané, 26, 38, 48, 88, 90
    - přijímacím stavem, 78
    - s prázdným zásobníkem, 78
- spojení
  - kaskádní, 15
  - paralelní, 16
  - sériové, 16
- stav
  - dosažitelný, 26
  - garbage, 27
  - iniciální, 26, 38, 77, 87
  - koncový, 26, 38
  - odpadový, 27
  - počáteční, 5, 26
  - přijímací, 26, 38, 77, 88
- strom
  - derivační, 71, 72
- substituce, 54
  - inverzní, 54
  - regulární, 54, 57

- suffix, 34
  - vlastní, 34
- symbol, 3
  - iniciální, 60
  - prázdný, 87
  - zásobníkový
    - iniciální, 77
- T.s., 88
- terminál, 61, 71
- these
  - Churchova, 95
- Turingův stroj, 3, 87, 91, 93
  - deterministický, 87
  - jednopáskový, 87, 90
  - lineární, 91, 93, 94
  - nedeterministický, 90
  - univerzální, 99
  - vícepáskový, 89
- věta
  - Kleene, 43, 45
- výpočet, 11, 26, 38, 48, 78
  - přijímací, 26, 38, 48
- výraz
  - konečný, 95
  - regulární, 45, 46, 55
- ZA, 76
- zrcadlový obraz
  - jazyka, 42, 43, 76, 94
  - slova, 42



## Seznam vět

2.6	Věta (Moore→Mealy)	8
2.7	Věta (Mealy→Moore)	8
3.6	Věta (O podílovém automatu)	13
3.16	Věta (Sériová dekompozice)	16
3.17	Věta (Paralelní dekompozice)	18
4.17	Věta (Myhill-Nerode)	32
4.24	Věta (Nutná podmínka regularity jazyka)	37
4.35	Věta (Kleene)	43
5.17	Věta (Pumping lemma)	72
6.16	Věta (Parikh)	87

## Seznam příkladů

2.8	Příklad (Převedení Moore→Mealy)	9
2.9	Příklad (Převedení Mealy→Moore)	10
3.18	Příklad (Hledání kongruencí automatu)	21
3.19	Příklad (Sestrojení dekompozic)	23
4.4	Příklad (Akceptor)	26
4.18	Příklad (Syntaktické kongruence)	34
4.25	Příklad (Důkaz neregularity jazyka)	38
4.37	Příklad (Regulární výraz)	46
4.38	Příklad (Regulární výrazy)	47
4.42	Příklad (Dvoucestný automat)	53
4.44	Příklad (Substituce)	55
4.47	Příklad (Kvocienty)	59
5.5	Příklad (Gramatika)	63
5.10	Příklad (Nezkracující→Kontextová)	69
5.16	Příklad (Derivační strom)	72
5.18	Příklad (Pumping lemma)	75
7.18	Příklad (Turingův stroj)	95
7.19	Příklad (Kódování)	99

## 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)
Doplňek	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 <i>T.s.</i> (7.12)	kontextové nezkracující (5.9)
$\mathcal{L}_0$	rekursivně spočetné (7.12)	<i>T.s.</i>	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 sestavení 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) věta 4.24	Myhill-Nerode 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