SLEZSKÁ UNIVERZITA V OPAVĚ
FILOZOFICKO-PŘÍRODOVĚDECKÁ FAKULTA
ÚSTAV INFORMATIKY


# TEORIE JAZYKŮ A AUTOMATŮ II

**Studijní opora**

*Poslední změny: 26. února 2008*

RNDr. Šárka Vavrečková

sarka.vavreckova@fpf.slu.cz
http://fpf.slu.cz/~vav10ui

Opava 2008

# Obsah

# Seznam definic

# Úvod

Tento studijní text je určen pro studenty kurzu Teorie jazyků a automatů II a navazuje na obdobný studijní text pro předchozí kurz Teorie jazyků a automatů I.

Předpokládají se znalosti v rozsahu předchozího kurzu, především z oblasti bezkontextových a regulárních gramatik. Samotný výklad sestává zejména z definic, vět a důkazů, ovšem většinou nejde o důkazy v pravém smyslu slova, jsou hodně zjednodušené. „Opravdový“ důkaz není jen popis konstrukce, ale musí být dokázáno, že tato konstrukce je správná.

Text je hojně doprovázen příklady, které mají zvýšit názornost výkladu.

V textu jsou graficky vyznačeny některé části:

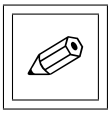
**Příklad 0.1**

Takto jsou vyznačeny příklady. Jsou číslovány v závislosti na čísle kapitoly a je na ně v textu odkazováno pomocí tohoto číslování.

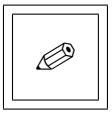
---

**Definice 0.1 (Definovaný pojem)** *Takto je vyznačena definice jednoho nebo více pojmů. Definice jsou číslovány v závislosti na čísle kapitoly.*

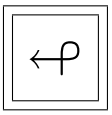
**Věta 0.1** *Takto jsou vyznačeny věty, opět jsou číslovány. Kapitola 1 ještě nebyla, proto zde máme jako číslo kapitoly 0.*



**Lemma 0.2** *Lemmata (pomocné věty) jsou vyznačena podobně jako věty, číslování je simultánní s větami. Lemma obvykle obsahuje tvrzení, které je pak použito v důkazu „větší“ věty.*



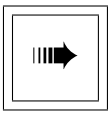
**Důkaz:** Takto vyznačujeme důkaz věty nebo lemmatu. Každá věta by měla být dokázána, zde však, stejně jako v předchozím studijním textu pro kurz Teorie jazyků a automatů I, jsou pod pojmem důkaz obvykle míněny spíše konstrukce naznačující důkaz nebo vztah.



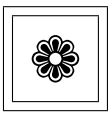
Důkazy končí symbolem prázdného čtverečku. □

**Myšlenka důkazu:** Myšlenka důkazu naznačuje, jak by mohl být vytvořen důkaz. Narozdíl od důkazu samotného nekončí symbolem čtverečku.

**Důsledek 0.3** *Takto je vyznačen důsledek předchozích vět. Obvykle je také uvedeno, ze kterých vět tento důsledek vyplývá, a nebo následuje důkaz stejně jako za kteroukoliv větou.*



**Poznámka:** Takto je vyznačena poznámka, ve které je obvykle okomentován důkaz, věta nebo definice.



Kapitola 1

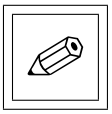
# Vlastnosti bezkontextových jazyků

*Tato kapitola doplňuje znalosti získané v kurzu Teorie jazyků a automatů I. Zaměříme se zde na dosud neprobrané vlastnosti bezkontextových jazyků, a to kritéria bezkontextovosti (určující, zda jazyk není bezkontextový) a uzávěrové vlastnosti.*

## 1.1 Kritéria bezkontextovosti

### 1.1.1 Pumping lemma pro bezkontextové jazyky

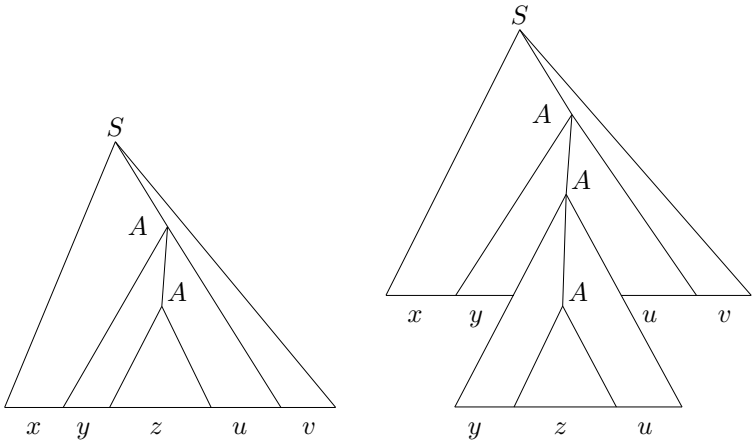
Pumping lemma pro bezkontextové jazyky bude podobné obdobnému lemmatu pro regulární jazyky probíranému v předchozím kurzu. Jde o to zachytit „nekonečné smyčky“, tentokrát v potenciální bezkontextové gramatice, a využít toho v důkazu sporem (dokazujeme obvykle, že když daný jazyk nemá tuto vlastnost, pak nemůže být bezkontextový).



**Věta 1.1 (Pumping lemma)** *Nechť $L$ je bezkontextový jazyk. Pak existují přirozená čísla $p$ a $q$ taková, že každé slovo $w \in L$, $|w| > p$, můžeme rozložit na pět částí $w = x \cdot y \cdot z \cdot u \cdot v$, přičemž*

- $|y \cdot u| > 0$ *(v alespoň jedné z těchto dvou částí musí být alespoň jeden symbol),*
- $|yzu| \leq q$ *(prostřední část má omezenou délku),*
- $x \cdot y^i \cdot z \cdot u^i \cdot v \in L$ *pro každé $i \geq 0$ (po iteraci těchto dvou částí slovo zůstává v jazyce).*

Nejdřív si větu rozebereme a potom teprve uvedeme důkaz.




(a) Odvození slova $xyzuv$ (b) Odvození slova $xy^2zu^2v$

Obrázek 1.1: Nákres použití Pumping lemma

**Myšlenka důkazu:** Podíváme se na obrázek 1.1. Věta se zakládá na této úvaze:

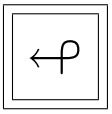
1. Když je jazyk bezkontextový, pak pro něj musí existovat bezkontextová gramatika.
2. V této gramatice musí pro každé slovo jazyka existovat derivace.
3. Označme $n = |N|$ počet neterminálů gramatiky a $d$ délku nejdelšího pravidla gramatiky, $d = max\left(|\alpha| \;\middle|\; (A \to \alpha) \in P\right)$ pro nějaké $A \in N\Big)$. Pak v každém kroku derivace se slovo prodlouží nejvýše o $d$ znaků, a tedy po $n$ krocích odvození $n \cdot d$ se prodlouží maximálně o $n \cdot d$ znaků.
4. Derivace slov delších než $n \cdot d$ znaků musí být proto delší než $n$.
5. Když je derivace delší než $n$, tak to znamená, že alespoň jeden neterminál musel být v průběhu derivace přepisován více než jednou (podle obrázku 1.1a je to neterminál $A$).
6. Když rozdělíme dostatečně dlouhé slovo $w \in L$ podle obrázku 1.1a na pět částí $w = x{\cdot}y{\cdot}z{\cdot}u{\cdot}v$, tak vlastně zkoumáme, zda v případné derivaci existuje některý opakovaně přepisovaný neterminál. Pokud jde o bezkontextový jazyk, tak takový neterminál (cyklus) musí existovat, ale není řečeno, že ho objevíme „náhodně“, okamžitě a napoprvé.
7. Když se nám takový cyklus podaří najít, pak (podle obrázku 1.1b) můžeme v této derivaci „pumpovat“ – při přepsání druhého zobrazeného výskytu

symbolu $A$ prostě použijeme to pravidlo, které jsme původně použili při přepsání prvního zobrazeného výskytu symbolu $A$ a tak pokračujeme v celém podstromě. Krajní řetězce $x$ a $v$ a nejvnitřnější řetězec $z$ zůstanou takové, jaké byly, jen řetězce $y$ a $u$ se zdvojí.

Cestu mezi dvěma výskyty symbolu $A$ jsme vlastně zdvojili, ale můžeme ji opakovat kolikrát chceme, a nebo dokonce oba výskyty symbolu $A$ ztotožnit (pro první výskyt symbolu $A$ použijeme to pravidlo, které jsme v původní derivaci použili pro druhý), to odpovídá hodnotě $i = 0$ ve větě 1.1.

Ještě zbývá číslo $q$. Jeho úkolem je zajistit, aby délka části derivace mezi dvěma výskyty $A$ nebyla nekonečná. Toto číslo může být stanoveno zcela náhodně (samozřejmě nikoliv nekonečno), například si můžeme stanovit, že na obrázku 1.1b v podstromě s kořenem ohodnoceným prvním výskytem $A$ je pouze to jediné další $A$, třetí se tam už nevyskytuje (ale v derivaci mezi $S$ a prvním $A$ klidně ještě nějaké být může).

**Důkaz:** Nechť $L$ je bezkontextový jazyk. Pak existuje některá bezkontextová gramatika $G = (N, T, P, S)$ taková, že $L = L(G)$. Pro zjednodušení důkazu předpokládejme, že tato gramatika je v Chomského normální formě (tj. na pravé straně pravidla jsou buď dva neterminály nebo jeden terminál). Derivační strom gramatiky v CNF je binární (kromě přímých cest k listům). Označme $n = |N|$.



Vezmeme si nyní některé slovo $w \in L$ takové, že v jeho derivaci je nejméně dvakrát přepisován tentýž neterminál (nazvěme ho třeba $A$), to znamená, že v derivačním stromě tohoto slova se na cestě od kořene k některému listu nachází dva uzly označené symbolem $A$, tyto uzly nazveme $u_1$ a $u_2$ ($u_1$ je blíž kořeni stromu). Pokud je na této cestě více uzlů označených symbolem $A$ než dva, zvolíme uzel $u_1$ tak, aby na cestě od $u_1$ k jakémukoliv listu v jeho podstromě byl jen jediný další uzel označený symbolem $A$, tj. $u_2$.

Při splnění podmínky z předchozího odstavce je cesta od $u_1$ ke kterémukoliv listu jeho podstromu dlouhá nejvýše $n + 1$ uzlů. Protože jde o binární strom, má podstrom uzlu $u_1$ nejvýše $2^n$ listů. Touto hodnotou je tedy omezena délka slova $y \cdot z \cdot u$, proto lze zvolit

$$q = 2^n.$$

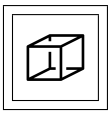
Hodnota čísla $p$ udává, jak dlouhé musí být slovo, aby v jeho derivačním stromě existovala cesta od kořene k některému listu taková, že některé dva uzly na této

cestě jsou ohodnoceny stejným neterminálem. Z předchozích odstavců důkazu je zřejmé, že všechna slova nevyhovující této podmínce mají derivační strom, v němž jsou všechny větve kratší než $n$ (tj. dlouhé nejvýše $n-1$). Takový derivační strom má tedy nejvýše $2^{n-1}$ listů (a tedy vygenerovaných terminálů). Proto položíme

$$p = 2^{n-1}.$$

□

**Příklad 1.1**



V tomto příkladu si ukážeme, že bezkontextový jazyk splňuje Pumping lemma.
$L_1 = \{a^n b^n \mid n \geq 1\}$

Protože předem neznáme přesné hodnoty $p$ a $q$ z věty 1.1, budeme používat symbolicky přímo index $p$ s předpokladem, že jde o „dostatečně velké“ číslo.

$w = a^p b^p$

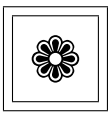
Zvolíme toto rozdělení:

| $x$ | $y$ | $z$ | $u$ | $v$ |
|---|---|---|---|---|
| $a^{p-1}$ | $a$ | $\varepsilon$ | $b$ | $b^{p-1}$ |

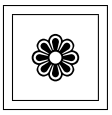
Pak dostáváme slova $a^{p-1}a^i b^i b^{p-1} = a^{p+i-1}b^{p+i-1}$, což jsou pro jakékoliv číslo $i \geq 0$ slova patřící do jazyka $L_1$. Číslo $p$ zde lze položit například $p = 2$ (nebo vyšší).

Tento jazyk generuje například gramatika s těmito pravidly $S \to aSb \mid ab$

---



**Poznámka:** Pumping lemma je pouze implikace (ve tvaru „jestliže je jazyk bezkontextový, pak splňuje tuto vlastnost“). Proto ji nelze použít tak, že po zjištění, že jazyk splňuje danou vlastnost, bychom tento jazyk prohlásili za bezkontextový. Například jazyk $\{a^n b^n \mid n \geq 0\} \cdot \{a^n b^n c^n \mid n \geq 0\}$ není bezkontextový, třebaže splňuje podmínky Pumping lemma. Obecně to lze říci o všech jazycích, které sice nejsou bezkontextové, ale lze je napsat jako zřetězení dvou jazyků, z nichž alespoň jeden je bezkontextový.



**Poznámka:** Pumping lemma se obvykle používá pro důkaz, že některý jazyk *není* bezkontextový, tedy důkaz sporem. Větu obrátíme:

$$A \Rightarrow B \quad \Longleftrightarrow \quad \neg B \Rightarrow \neg A \tag{1.1}$$

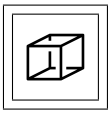
V přepisu:

*Jestliže jazyk je bezkontextový, pak má danou vlastnost.*

je totéž jako

*Jestliže jazyk nemá danou vlastnost, pak není bezkontextový.*

**Příklad 1.2**



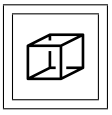
Dokážeme, že jazyk $L_2 = \{a^n b^n c^n \mid n \geq 0\}$ není bezkontextový.

Zvolíme slovo $w = a^p b^p c^p$ pro některé dostatečně velké číslo $p$. Možná rozdělení tohoto slova jsou v tabulce. Musíme brát v úvahu konečnost konstanty $q$, v části $yzu$ se proto nesmí vyskytovat „potenciálně nekonečný" index $p$.

| $x$ | $y$ | $z$ | $u$ | $v$ | $xy^izu^iv$ | pro $i = 0$ |
|---|---|---|---|---|---|---|
| $a^p b^p c^{p-1}$ | $c$ | $\varepsilon$ | $\varepsilon$ | $\varepsilon$ | $a^p b^p c^{p-1+i}$ | $a^p b^p c^{p-1} \notin L_2$ |
| $a^p b^{p-1}$ | $b$ | $c$ | $\varepsilon$ | $c^{p-1}$ | $a^p b^{p-1+i} c^p$ | $a^p b^{p-1} c^p \notin L_2$ |
| $a^p b^{p-1}$ | $b$ | $\varepsilon$ | $c$ | $c^{p-1}$ | $a^p b^{p-1+i} c^{p-1+i}$ | $a^p b^{p-1} c^{p-1} \notin L_2$ |
| $a^p b^{p-2}$ | $bb$ | $\varepsilon$ | $cc$ | $c^{p-2}$ | $a^p b^{p-2+2i} c^{p-2+2i}$ | $a^p b^{p-2} c^{p-2} \notin L_2$ |
| $\ldots$ | | | | | | |

Při splnění podmínek Pumping lemma ($|yu| > 0,\ |yzu| \leq q$) vidíme na tabulce, že „pumpující" část $yu$ může zachytit nejvýše dva druhy symbolů (buď jen symboly $a$, nebo jen $b$, $c$, a nebo jen $a, b$, $b, c$), tedy přibývat (nebo ubývat pro $i = 0$ nemohou zároveň symboly $a, b$ i $c$ a proto při jakémkoliv možném rozdělení vznikají iterací slova nepatřící do jazyka $L_2$. Proto $L_2 \notin \mathscr{L}(CF)$[1].

**Příklad 1.3**



Dokážeme, že jazyk $L_3 = \left\{a^{n^2} \;\middle|\; n \geq 0\right\}$ není bezkontextový.

Opět zvolíme nějaké slovo $w = a^{p^2} \in L_3$ s „dostatečně velkým" indexem $p$. Toto slovo rozdělíme následovně: $a^{p^2} = a^{x_1} a^{x_2} a^{x_3} a^{x_4} a^{x_5}$
a musí platit $x_2 + x_4 > 0,\ x_2 + x_3 + x_4 \leq q$

Před iterací:

$$p^2 = x_1 + x_2 + x_3 + x_4 + x_5$$

Po iteraci:

$$m^2 = x_1 + i \cdot x_2 + x_3 + i \cdot x_4 + x_5 = i \cdot (x_2 + x_4) + (x_1 + x_3 + x_5)$$

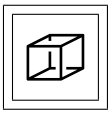
Získali jsme rovnici, v jejichž obou částech oddělených rovnítkem jsou funkce. Zatímco levá část rovnice roste exponenciálně, pravá lineárně (je to lineární funkce)

[1] $CF$ značí bezkontextové gramatiky, $\mathscr{L}(CF)$ znamená třídu (tj. množinu) jazyků generovaných bezkontextovými gramatikami, tj. bezkontextové jazyky.

a tedy mnohem pomaleji. Ať stanovíme indexy $x_2$ a $x_4$ jakkoliv, vždy se najde číslo $i$, pro které součet indexů není roven druhé mocnině žádného čísla. $L_3 \notin \mathscr{L}(CF)$.

---

**Příklad 1.4**



Pomocí Pumping lemma dokážeme, že jazyk $L_4 = \left\{ a^{2^n} \;\middle|\; n \geq 0 \right\}$ není bezkontextový. Důkaz bude podobný předchozímu.

Zvolíme nějaké slovo $w = a^{2^r} \in L_4$ s „dostatečně velkým" indexem $r$. Toto slovo rozdělíme na $a^{2^r} = a^{x_1}a^{x_2}a^{x_3}a^{x_4}a^{x_5}$ a musí platit $x_2 + x_4 > 0,\ x_2 + x_3 + x_4 \leq q$

Před iterací:

$$2^r = x_1 + x_2 + x_3 + x_4 + x_5$$

Po iteraci:

$$2^m = x_1 + i \cdot x_2 + x_3 + i \cdot x_4 + x_5 = i \cdot (x_2 + x_4) + (x_1 + x_3 + x_5)$$

Získali jsme rovnici, v jejichž obou částech oddělených rovnítkem jsou funkce. Zatímco levá část rovnice roste exponenciálně, pravá lineárně (je to lineární funkce) a tedy mnohem pomaleji. Ať stanovíme indexy $x_2$ a $x_4$ jakkoliv, vždy se najde číslo $i$, pro které součet indexů není roven žádné mocnině čísla 2. Proto $L_4 \notin \mathscr{L}(CF)$.

---

**Příklad 1.5**

Dokážeme, že jazyk $L_5 = \{ww \mid w \in \{a, b\}^*\}$ není bezkontextový. Tento jazyk se skládá ze slov, která mají obě poloviny stejné.

Pro důkaz si vybereme slovo $w = a^p b^p a^p b^p,\ p > 4, q = 4$ a dokážeme, že pro toto slovo neexistuje žádné rozdělení, které by umožňovalo „pumpování" dle Pumping lemma.

| $x$ | $y$ | $z$ | $u$ | $v$ | $xy^izu^iv$ | pro $i = 0$ |
|---|---|---|---|---|---|---|
| $a^p b^{p-1}$ | $b$ | $\varepsilon$ | $a$ | $a^{p-1}b^p$ | $a^p b^{p-1+i} a^{p-1+i} b^p$ | $a^p b^{p-1} a^{p-1} b^p \notin L_5$ |
| $a^p b^{p-1}$ | $\varepsilon$ | $b$ | $a$ | $a^{p-1}b^p$ | $a^p b^p a^{p-1+i} b^p$ | $a^p b^p a^{p-1} b^p \notin L_5$ |
| $a^p b^{p-1}$ | $\varepsilon$ | $\varepsilon$ | $ba$ | $a^{p-1}b^p$ | $a^p b^{p-1} (ba)^i a^{p-1} b^p$ | $a^p b^{p-1} a^{p-1} b^p \notin L_5$ |
| $\ldots$ | | | | | | |

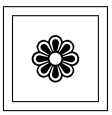
Jak vidíme, většinou nám úplně stačí pro každé rozdělení otestovat slovo pro $i = 0$. Můžeme pokračovat dalšími řádky tabulky, ale vždy dojdeme ke stejnému závěru.

Je to proto, že když je část $yzu$ omezena konstantou, může zabrat nejvýše dvě různé části ze čtyř částí vybraného slova $w$ (a přitom do $yu$ musí padnout alespoň

jeden symbol slova), tedy po iteraci pro žádné $i$ kromě $i = 1$ nedostaneme slovo, jehož obě poloviny jsou stejné. Proto $L_5 \notin \mathscr{L}(CF)$.

---

**Poznámka:** Podobně by mohl vypadat důkaz, že jazyk
$L_6 = \{a^n b^m a^n \mid 0 \leq m \leq n\}$
není bezkontextový (kdyby v definici tohoto jazyka nebyly nerovnosti, šlo by o bezkontextový jazyk). Tento důkaz ponecháváme na čtenáři.



### 1.1.2 Parikhův vektor

**Definice 1.1 (Parikhův vektor slova)** *Nechť $L$ je některý jazyk nad abecedou $\Sigma$, kde $\Sigma = \{a_1, a_2, \ldots, a_n\}$, $|\Sigma| = n$. Parikhův vektor slova $w \in L$ je*

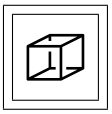
$$\psi(w) = (\#a_1(w), \#a_2(w), \ldots, \#a_n(w))$$

*kde $\#a_i(w)$ značí počet symbolů $a_i$ ve slově $w$.*

**Definice 1.2 (Parikhův vektor jazyka)** *Nechť $L$ je některý jazyk nad abecedou $\Sigma$, kde $\Sigma = \{a_1, a_2, \ldots, a_n\}$, $|\Sigma| = n$. Parikhův vektor jazyka $L$ je množina Parikhových vektorů všech slov tohoto jazyka, tedy*

$$\psi(L) = \{\psi(w) \mid w \in L\}$$

**Příklad 1.6**



Vyzkoušíme si vytvoření Parikhových vektorů pro slova i jazyky.

$L_1 = \{a^n b^n \mid n \geq 1\}$
$\psi(aaabbb) = (3, 3)$
$\psi(L_1) = \{i \cdot (1, 1) \mid i \geq 1\}$

$L_7 = \{a^{3n} b^{n+2} a^4 c^n \mid n \geq 1\}$
$\psi(a^6 b^4 a^4 c^2) = (10, 4, 2)$
$\psi(L_7) = \{(3n + 4, n + 2, n) \mid n \geq 1\} =$
$\quad = \{n \cdot (3, 1, 1) + (4, 2, 0) \mid n \geq 1\}$

$L_8 = \{a^{2n} b^{n-1} \mid n \geq 1\}$
zde je problém konstanta $(-1)$ v exponentu – položme $k = n - 1$, tedy $k + 1 = n$:
$\psi(L_8) = \{(2 \cdot (k + 1), k) \mid k \geq 0\} = \{k \cdot (2, 1) + (2, 0) \mid k \geq 0\}$

---

Z algebry si určitě pamatujeme operace s vektory – sčítání vektorů a násobení vektoru celým číslem:

$$(x_1, x_2, \ldots, x_n) + (y_1, y_2, \ldots, y_n) = (x_1 + y_1, x_2 + y_2, \ldots, x_n + y_n) \quad (1.2)$$

$$k \cdot (x_1, , x_2, \ldots, x_n) = (k \cdot x_1, k \cdot x_2, \ldots, k \cdot x_n) \quad (1.3)$$

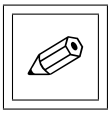
Dále budeme používat toto značení:
$\mathbb{N}$ je množina přirozených čísel (zde včetně 0)
$\mathbb{N}^n$ je množina všech n-prvkových vektorů nad množinou $\mathbb{N}$

**Definice 1.3 (Lineární podmnožina množiny $\mathbb{N}^n$)** *Lineární podmnožina množiny $\mathbb{N}^n$ pro nějaké $n \in \mathbb{N}$ je*

$$\{\bar{v}_0 + n_1 \cdot \bar{v}_1 + n_2 \cdot \bar{v}_2 + \ldots + n_k \cdot \bar{v}_k \mid n_i \in \mathbb{N}, 1 \leq i \leq k, \bar{v}_j \in \mathbb{N}^n, 0 \leq j \leq k\}$$

*($n_i$ jsou čísla, $\bar{v}_j$ jsou vektory čísel)*

**Definice 1.4 (Semilineární množina)** *Semilineární množina je konečné sjednocení lineárních podmnožin množiny $\mathbb{N}^n$.*



**Věta 1.2** *Pro každý bezkontextový jazyk $L$ je $\psi(L)$ semilineární množina.*

Důkaz této věty by byl složitý, proto ho zde nebudeme uvádět.

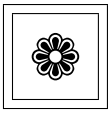
**Příklad 1.7**

$L_9 = \{a^i b^j c^k \mid i, j, k \geq 1,\ i = j \text{ nebo } j = k\}$

$\psi_1(L_9) = \{i \cdot (1, 1, 0) + k \cdot (0, 0, 1) \mid i, k \geq 1\}$
$\psi_2(L_9) = \{i \cdot (1, 0, 0) + j \cdot (0, 1, 1) \mid i, j \geq 1\}$
$\psi(L_9) = \psi_1(L_9) \cup \psi_2(L_9)$

Parikhův vektor jazyka $L_9$ je jednocení „dílčích“ lineárních množin, a tedy semilineární množina.

---



**Poznámka:** Stejně jako Pumping lemma, i zde se jedná pouze o implikaci. Opět budeme tuto větu používat v důkazu sporem – jestliže Parikhův vektor daného jazyka není semilineární množina, pak to není bezkontextový jazyk.

**Příklad 1.8**

$L_{10} = \{a^*bc\} \cup \{a^p b a^n c \mid n \geq 0, p \text{ je prvočíslo}\}$

$\psi_1(L_{10}) = \{i \cdot (1,0,0) + (0,1,1) \mid i \geq 0\}$
$\psi_2(L_{10}) = \{n \cdot (1,0,0) + p \cdot (1,0,0) + (0,1,1) \mid n \geq 0,\ p \text{ je prvočíslo}\}$
$\psi(L_{10}) = \psi_1(L_{10}) \cup \psi_2(L_{10})$

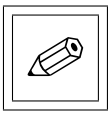
Množina $\psi_2(L_{10})$ není lineární, a proto množina $\psi(L_{10})$ není semilineární. Neexistuje *konečné* sjednocení lineárních množin popisujících množinu odvozenou z prvočísel, to bychom museli postupně vyjmenovat všechna prvočísla (a to by nebylo *konečné* sjednocení). Proto $L_{10} \notin \mathscr{L}(CF)$.

$L_{11} = \left\{a^{n^2} b^n \mid n \geq 1\right\}$
$\psi(L_{11}) = \{n^2 \cdot (1,0) + n \cdot (0,1) \mid n \geq 1\}$ není semilineární množina (je zde kvadratická funkce). Proto $L_{11} \notin \mathscr{L}(CF)$.

---

Ale pozor! $L_{12} = \left\{a^{n^2} b^n \mid 1 \leq n \leq 8\right\}$ ovšem je bezkontextový jazyk, protože je konečný ($L_{12} \in \mathscr{L}(FIN)$) – Parikhův vektor tohoto jazyka by mohl být složen z Parikhových vektorů jednotlivých (osmi) slov jazyka, což je sjednocení konečně mnoha jednoprvkových lineárních množin.

**Věta 1.3** *Ke každému bezkontextovému jazyku $L$ existuje regulární jazyk $R$ takový, že $\psi(L) = \psi(R)$.*



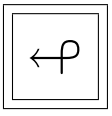
**Příklad 1.9**

$L_1 = \{a^n b^n \mid n \geq 1\}$
$\psi(L_1) = \{n \cdot (1,1) \mid n \geq 1\}$
$R_1 = ab(ab)^*, \quad \psi(R_1) = \psi(L_1)$

$L_7 = \{a^{3n} b^{n+2} a^4 c^n \mid n \geq 1\}$
$\psi(a^6 b^4 a^4 c^2) = (10,4,2)$
$\psi(L_7) = \{(3n+4, n+2, n) \mid n \geq 1\} = \{n \cdot (3,1,1) + (4,2,0) \mid n \geq 1\}$
$R_7 = (aaabc)^* aaabcaaaabb, \quad \psi(R_7) = \psi(L_7)$

---

**Důkaz:** Označme abecedu, nad kterou je definován jazyk, $\Sigma = \{a_1, \ldots, a_r\}$, tj. má celkem $r$ prvků. Věta vyplývá z věty 1.2 na straně 10 – když je Parikhův vektor jazyka semilineární množina a tedy sjednocení lineárních množin, tak postupně



všechny lineární množiny ve tvaru

$$\{\bar{v}_0 + n_1 \cdot \bar{v}_1 + n_2 \cdot \bar{v}_2 + \ldots + n_k \cdot \bar{v}_k \mid n_i \in \mathbb{N}, 1 \leq i \leq k, \bar{v}_j \in \mathbb{N}^n, 0 \leq j \leq k\}$$

rozložíme na jednotlivé vekory násobené číslem

$$n_i \cdot \bar{v}_i \quad = \quad n_i(v_{i,1}, v_{i,2}, \ldots, v_{i,r}),$$

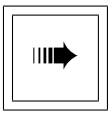
kde $n_i$ bývá buď konstanta nebo proměnná pro danou množinu nabývající hodnot $n_i \geq K_i$, $1 \leq n_i \leq k$.

Regulární jazyk pro $n_i \cdot \bar{v}_i$ vytvoříme takto:

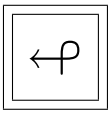
$$a_1^{v_{i,1} \cdot K_i} a_2^{v_{i,2} \cdot K_i} \ldots a_r^{v_{i,r} \cdot K_i} \cdot \left(a_1^{v_{i,1}} a_2^{v_{i,2}} \ldots a_r^{v_{i,r}}\right)^*$$

Tyto regulární jazyky vektorů lineární množiny spojíme operátorem $+$. Vzniknou dílčí regulární jazyky pro jednotlivé lineární množiny. Ty taktéž spojíme operátorem $+$ (pro připomenutí – tento operátor odpovídá sjednocení). □

**Důsledek 1.4** *Nad jednoprvkovou množinou bezkontextovost nepřidá na generativní síle, tj. každý bezkontextový jazyk nad jednoprvkovou abecedou je zároveň regulární. Když takový jazyk není regulární, tak není ani bezkontextový (je kontextový nebo typu 0).*



**Důkaz:** Důkaz je triviální – v důkazu věty 1.3 jsme vlastně „přemísťovali“ jednotlivé symboly v definici jazyka. Když však tímto způsobem proházíme symboly v definici jazyka nad jednoprvkovou abecedou, generovaný jazyk se nezmění (například $a^{1+2n}$ pro $n \geq 0$ je totéž jako $a(aa)^*$).



Zřejměji je možné si tento postup ukázat na pravidlech – když máme bezkontextové pravidlo $A \to a^i B a^j$ pro $A, B \in N$, $T = \{a\}$, indexy $i, j \geq 0$, pak ekvivalentní regulární pravidlo vytvoříme přesunem: $A \to a^i a^j B$, případně řetězec symbolů $a$ rozdělíme s použitím pomocných neterminálů.

Proto pro každou bezkontextovou gramatiku nad jednoprvkovou abecedou existuje regulární, která je s ní ekvivalentní. □

### 1.1.3 Dyckův jazyk

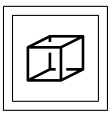
Dyckovy jazyky se také nazývají *dobře uzávorkované jazyky*. Jde vlastně o jazyky nad abecedou uspořádaných dvojic znaků odpovídajících levým a pravým závorkám různých typů.

**Definice 1.5 (Dyckův jazyk)** *Nechť* $\Sigma_n = \{a_1, a'_1, a_2, a'_2, \ldots, a_n, a'_n\},\ |\Sigma_n| = 2n$ *je abeceda,* $n \geq 1$.

*Dyckův jazyk nad touto abecedou je jazyk generovaný gramatikou s pravidly*

$$S \to a_1 S a'_1 \mid a_2 S a'_2 \mid \ldots \mid a_n S a'_n \mid SS \mid \varepsilon$$

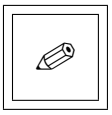
**Příklad 1.10**



$\Sigma_2 = \Big\{(,),[,]\Big\}$ $(n = 2, |\Sigma_2| = 4)$

$S \to (S) \;\Big|\; [S] \;\Big|\; SS \;\Big|\; \varepsilon$

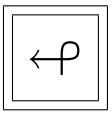
$S \Rightarrow SS \Rightarrow (S)S \Rightarrow (\ [S]\ )S \Rightarrow (\ [\ ]\ )S \Rightarrow (\ [\ ]\ )[S] \Rightarrow (\ [\ ]\ )\ [\ ]$



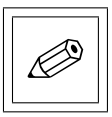
**Lemma 1.5 (Vlastnosti Dyckova jazyka)** *Nechť* $D_n$ *je Dyckův jazyk nad abecedou* $\Sigma_n$. *Pak pro jakákoliv dvě slova* $u, v \in \Sigma_n^*$ *platí*

1. *Jestliže* $u, v \in D_n$, *pak* $u \cdot v \in D_n$.
2. *Jestliže* $u \in D_n$, *pak* $a_i \cdot u \cdot a'_i \in D_n,\ 1 \leq i \leq n$.
3. *Každé slovo* $w \in D_n,\ w \neq \varepsilon$ *je ve tvaru* $a_i \cdot u \cdot a'_i v$ *pro nějaké* $1 \leq i \leq n,\ u, v \in D_n$.
4. *Jestliže* $a_i \cdot a'_i \cdot u \in D_n$, *pak* $u \in D_n$.

**Důkaz:** Důkazy jednotlivých tvrzení z lemmatu ponecháváme na čtenáři, jsou triviální a vyplývají přímo z definice Dyckova jazyka. □

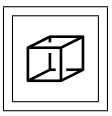


**Věta 1.6** *Jestliže* $L$ *je bezkontextový jazyk, pak existuje regulární jazyk* $R$ *a homomorfismus* $\varphi$ *takové, že* $L = \varphi(D \cap R)$ *pro nějaký Dyckův jazyk* $D$.



Důkaz této věty zde nebudeme uvádět.

**Příklad 1.11**



$L_1 = \{a^n b^n \mid n \geq 1\}$

Vybereme vhodný regulární jazyk $R$, homomorfismus $\varphi$ a Dyckův jazyk $D$:

$R = aa^*bb^*$ (dodá základní tvar slov – nejdřív $a$ a potom $b$)

$\varphi$ je identita,

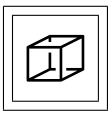
$D = (\{S\},\ \{a, b\},\ P,\ S)$, kde $P = \{S \to aSb \mid SS \mid \varepsilon\}$

Pak je zřejmé, že $L_1 = R \cap D$.

Jiný možný výběr:
$R = [^* \, ]^*$
$D = (\{S\},\ \{[,]\},\ P,\ S)$, kde $P = \left\{ S \to [\ S\ ] \;\middle|\; SS \;\middle|\; \varepsilon \right\}$
$\varphi([\,) = a, \quad \varphi(\,]\,) = b$

---

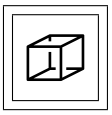
**Příklad 1.12**



$L_8 = \{a^{2n}b^{n-1} \mid n \geq 1\}$
$R = 230^*1^*$, jazyk $D$ má pravidla $P = \{S \to 0S1 \mid 2S3 \mid SS \mid \varepsilon\}$
$\varphi(0) = aa, \quad \varphi(1) = b, \quad \varphi(2) = a, \quad \varphi(3) = \varepsilon$

---

**Příklad 1.13**



Dokážeme, že jazyk $L_2 = \{a^n b^n c^n \mid n \geq 0\}$ není bezkontextový.

V předchozích příkladech jsme mohli pozorovat, že regulární jazyk zajišťuje správnou posloupnost symbolů a Dyckův jazyk synchronizuje počet symbolů v jednotlivých podskupinách.

Správnou posloupnost symbolů by mohl zajistit regulární jazyk $R = a^*b^*c^*$, ale nedokážeme najít Dyckův jazyk tak, aby dokázal synchronizovat tentýž počet symbolů na více než dvou místech. Proto $L_2 \notin \mathscr{L}(CF)$.

---

## 1.2 Jak poznat zda je jazyk bezkontextový

Nejlepším způsobem, jak určit, že je jazyk bezkontextový, a také prakticky jediným přímým důkazem, je sestrojení bezkontextové gramatiky generující tento jazyk.

V předchozích sekcích jsme si ukázali tři metody, které lze využít při důkazu sporem – Pumping lemma, Parikhův vektor jazyka a Dyckův jazyk. Ve všech třech případech jde o implikace, proto nejsou použitelné pro přímý důkaz.

Existuje však další možnost – využití uzávěrových vlastností bezkontextových jazyků. Bezkontextové jazyky jsou například uzavřeny vzhledem k operaci sjednocení, a tedy pokud lze některý jazyk napsat jako sjednocení dvou bezkontextových

jazyků, pak jde o bezkontextový jazyk. Uzávěrovým vlastnostem bezkontextových jazyků se budeme věnovat v sekci 1.3.

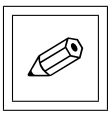
## 1.3 Uzávěrové vlastnosti bezkontextových jazyků

Ve větách a důkazech této sekce budeme používat následující značení (nesouvisející s průběžným číslováním jazyků v tomto dokumentu):
$L_1 = L(G_1),\ G_1 = (N_1, T_1, P_1, S_1)$
$L_2 = L(G_2),\ G_2 = (N_2, T_2, P_2, S_2),\ N_1 \cup N_2 = \emptyset$
vytváříme $L = L(G),\ G = (N, T, P, S)$

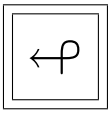
Narozdíl od regulárních jazyků, zde budou důkazy postaveny na konstrukci gramatik, ne automatů.

### 1.3.1 Regulární operace

**Věta 1.7** *Třída bezkontextových jazyků je uzavřena vzhledem k operaci sjednocení.*

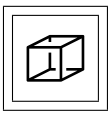


**Důkaz:** Vytvoříme gramatiku $G$ takovou, že $L(G) = L_1 \cup L_2$:
$G = (N, T, P, S)$, symbol $S \notin N_1 \cup N_2$ (nově přidaný), $T = T_1 \cup T_2$,
$N = N_1 \cup N_2 \cup \{S\}$,
$P = P_1 \cup P_2 \cup \{S \to S_1 \mid S_2\}$

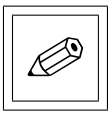


Přejímáme zde vše z původních gramatik, změna je jen na začátku derivace – prvním krokem derivace je rozhodnutí, zda bude generováno slovo z prvního nebo z druhého jazyka. Pak je provedena simulace činnosti některé z původních gramatik (resp. je předáno řízení některé z původních gramatik). □

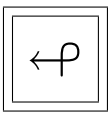
**Příklad 1.14**



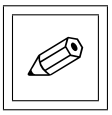
Jazyk $L_9 = \{a^i b^j c^k \mid i, j, k \geq 1,\ i = j \text{ nebo } j = k\}$ lze napsat jako sjednocení dvou bezkontextových jazyků $L_x \cup L_y$, kde
$L_x = \{a^i b^j c^k \mid i, j, k \geq 1,\ i = j\}$
$L_y = \{a^i b^j c^k \mid i, j, k \geq 1,\ j = k\}$ (gramatiky viz příklad 1.16 na straně 17)



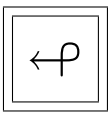
**Věta 1.8** *Třída bezkontextových jazyků je uzavřena vzhledem k operaci zřetězení.*



**Důkaz:** Vytvoříme gramatiku $G$ takovou, že $L(G) = L_1 \cdot L_2$:
$N = N_1 \cup N_2 \cup \{S\},\ S \notin N_1 \cup N_2$
$P = P_1 \cup P_2 \cup \{S \to S_1 \cdot S_2\}$
$T = T_1 \cup T_2\}$ □

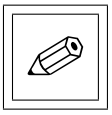


**Věta 1.9** *Třída bezkontextových jazyků je uzavřena vzhledem k operaci iterace (Kleeneho uzávěru, operace hvězdička).*

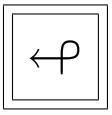


**Důkaz:** Vytvoříme gramatiku $G$ takovou, že $L(G) = L_1^*$:
$N = N_1 \cup \{S\},\ S \notin N_1,\ T = T_1,$
$P = P_1 \cup \{S \to S_1 S \mid \varepsilon\}$ □

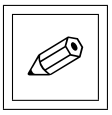
### 1.3.2 Operace, vzhledem k nimž je ještě třída $\mathscr{L}(CF)$ uzavřena



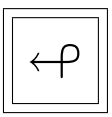
**Věta 1.10** *Třída bezkontextových jazyků je uzavřena vzhledem k operaci pozitivní iterace.*



**Důkaz:** Vytvoříme gramatiku $G$ takovou, že $L(G) = L_1^+$:
$N = N_1 \cup \{S\},\ S \notin N_1,\ T = T_1,$
$P = P_1 \cup \{S \to S_1 S \mid S_1\}$ □



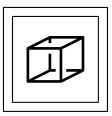
**Věta 1.11** *Třída bezkontextových jazyků je uzavřena vzhledem k operaci zrcadlení (reverze).*



**Důkaz:** Vytvoříme gramatiku $G$ takovou, že $L(G) = L_1^R$:
$N = N_1,\ T = T_1,\ S = S_1$

Každé pravidlo z původní množiny pravidel převrátíme – přeneseme reverzi na pravidla: $P = \{A \to \alpha^R \mid (A \to \alpha) \in P_1,\ A \in N_1,\ \alpha \in (N \cup T)^*\}$ □

**Příklad 1.15**



Pravidlo $A \to abbBcaCacb$ bude po reverzi $A \to bcaCacBbba$.

**Příklad 1.16**

Reverzi si ukážeme na tomto jazyce:
$L_9 = \{a^i b^j c^k \mid i, j, k \geq 1,\ i = j \text{ nebo } j = k\}$

Jazyk $L_9$ lze generovat bezkontextovou gramatikou s pravidly

$S \to S_1 \mid S_2$ $\quad A \to aAb \mid ab$ $\quad X \to cX \mid c$
$S_1 \to AX$ $\quad B \to bBc \mid bc$ $\quad Y \to aY \mid a$
$S_2 \to YB$

Ukázka derivace:
$S \Rightarrow S_1 \Rightarrow AX \Rightarrow aAbX \Rightarrow aabbX \Rightarrow aabbc$

Po reverzi:

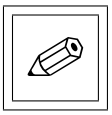
$S \to S_1 \mid S_2$ $\quad A \to bAa \mid ba$ $\quad X \to Xc \mid c$
$S_1 \to XA$ $\quad B \to cBb \mid cb$ $\quad Y \to Ya \mid a$
$S_2 \to BY$

Ukázka derivace:
$S \Rightarrow S_1 \Rightarrow XA \Rightarrow XbAa \Rightarrow Xbbaa \Rightarrow cbbaa$

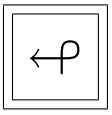
Generovaný jazyk je $L_9^R = \{c^k b^j a^i \mid i, j, k \geq 1,\ i = j \text{ nebo } j = k\}$ a stejně jako původní jazyk, i tento je bezkontextový (existuje bezkontextová gramatika, která ho generuje).

---

**Věta 1.12** *Třída bezkontextových jazyků je uzavřena vzhledem k operaci bezkontextové substituce.*



**Důkaz:** Bezkontextová substituce $s$ je takové zobrazení, které zobrazuje každý symbol původní abecedy na bezkontextový jazyk a přitom platí
$s(\varepsilon) = \varepsilon,\ s(a \cdot v) = s(a) \cdot s(v),\ a \in (N \cup T),\ v \in (N \cup T)^*$



Nechť $L_1$ je bezkontextový jazyk nad abecedou $\Sigma = \{a_1, a_2, \ldots, a_n\}$ a jsou dány bezkontextové jazyky $J_1, J_2, \ldots, J_n$ nad abecedami $\Sigma_1, \Sigma_2, \ldots, \Sigma_n$ tak, že $s(a_i) = J_i$, v každém jazyce $S_i$ je startovacím symbolem $a_i,\ 1 \leq i \leq n$. Pro všechny bezkontextové jazyky $J_i$ existují bezkontextové gramatiky $G_{J_i} = (N_{J_i}, \Sigma_i, P_{J_i}, a_i)$ a předpokládejme, že množiny jejich neterminálních symbolů jsou po dvou disjunktní a taktéž průnik kterékoliv z těchto množin neterminálů s množinou $N_1$ je prázdný.

Jazyk $L = s(L_1)$ sestrojíme takto:
$N = N_1 \cup \{a_1, a_2, \ldots, a_n\} \cup \bigcup_{i=1}^{n} N_{J_i}$

$T = \bigcup_{i=1}^{n} \Sigma_i$
$P = P_1 \cup \bigcup_{i=1}^{n} P_{J_i}$ □

**Příklad 1.17**

Postup si ukážeme na bezkontextovém jazyku
$L_9 = \{a^i b^j c^k \mid i, j, k \geq 1,\ i = j \text{ nebo } j = k\}$

$G_1 = (\{S, A, B, X, Y\},\ \{a, b, c\},\ P_1,\ S)$ a v množině $P_1$ jsou pravidla

$S \to AX \mid YB$ $\quad A \to aAb \mid ab$ $\quad X \to cX \mid c$
$\quad B \to bBc \mid bc$ $\quad Y \to aY \mid a$

Substituce:

$s(a) = \{1^n \mid n \geq 0\}, \quad G_{J_a} = (\{a\}, \{1\}, \{a \to 1a \mid \varepsilon\}, a)$
$s(b) = \{1^n 0^n \mid n \geq 1\}, \quad G_{J_b} = (\{b\}, \{1, 0\}, \{b \to 1b0 \mid 10\}, b)$
$s(c) = \{0^n \mid n \geq 0\}, \quad G_{J_c} = (\{c\}, \{0\}, \{c \to 0c \mid \varepsilon\}, c)$

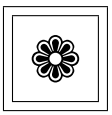
Po provedení substituce:

$G = (\{S, A, B, X, Y, a, b, c\},\ \{0, 1\},\ P,\ S)$ a v množině $P_1$ jsou pravidla

$S \to AX \mid YB$ $\quad X \to cX \mid c$ $\quad a \to 1a \mid \varepsilon$
$A \to aAb \mid ab$ $\quad Y \to aY \mid a$ $\quad b \to 1b0 \mid 10$
$B \to bBc \mid bc$ $\quad c \to 0c \mid \varepsilon$

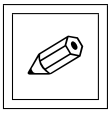
Generovaný jazyk je $L = s(L_9) = 1^* \cdot \{1^n 0^n \mid n \geq 1\}^* \cdot 0^*$

---

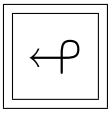


**Poznámka:** Protože homomorfismus je vlastně speciálním případem substituce (jde o jednoznačnou substituci, kdy jednomu symbolu přiřazujeme právě jedno slovo, tedy jednoprvkový jazyk), znamená to, že třída bezkontextových jazyků je uzavřena i vzhledem k operaci homomorfismu.

### 1.3.3 Operace, vzhledem k nimž třída $\mathscr{L}(CF)$ není uzavřena



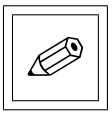
**Věta 1.13** *Třída bezkontextových jazyků* není *uzavřena vzhledem k operaci průniku.*



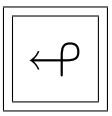
**Důkaz:** Najdeme dva bezkontextové jazyky, jejichž průnikem je jazyk, který není bezkontextový.
$L_x = \{a^i b^i c^j \mid i, j \geq 0\}$ (počet $a$ je stejný jako počet $b$)
$L_y = \{a^i b^k c^k \mid i, j \geq 0\}$ (počet $b$ je stejný jako počet $c$)

Průnikem těchto jazyků je $L = L_x \cap L_y = L_2 = \{a^n b^n c^n \mid n \geq 0\}$, o kterém jsme již dříve dokázali, že není bezkontextový (v příkladech 1.2 na straně 7 a 1.13 na straně 14).

Všimněme si, že sjednocením těchto jazyků je bezkontextový jazyk
$L_9 \cup \{\varepsilon\} = \{a^i b^j c^k \mid i, j, k \geq 0,\ i = j \text{ nebo } j = k\}$. □



**Věta 1.14** *Třída bezkontextových jazyků* není *uzavřena vzhledem k operaci doplňku (komplementu) nad danou abecedou.*



**Důkaz:** Využijeme již dokázaná tvrzení z předchozích vět, konkrétně to, že třída bezkontextových jazyků je uzavřena vzhledem ke sjednocení, ale není uzavřena vzhledem k průniku.

Podle DeMorganových pravidel, která si jistě pamatujeme z teorie množin nebo algebry (jazyky vlastně nejsou nic jiného než množiny slov), platí

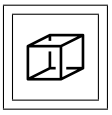
$$L_1 \cap L_2 = \overline{\overline{L_1} \cup \overline{L_2}} \tag{1.4}$$

Předpokládejme, že třída bezkontextových jazyků je uzavřena vzhledem k operaci doplňku (důkaz sporem). Pak na pravé straně rovnice (1.4) máme množinu bezkontextových jazyků. Jenže na levé straně rovnice je množina, ve které existují i jazyky, které nejsou bezkontextové (průnikem dvou bezkontextových jazyků může být i jazyk, který není bezkontextový), což je spor. □

## 1.4 Uzávěrové vlastnosti jako kritérium bezkontextovosti

Většina uzávěrových vlastností se dá použít v přímém důkazu:

**Příklad 1.18**

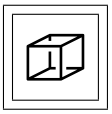


Zjistíme, zda je následující jazyk bezkontextový:
$L_{13} = \{a^{n_1} b^{n_1} a^{n_2} b^{n_2} \ldots a^{n_k} b^{n_k} \mid k \geq 0,\ n_i \geq 1,\ 1 \leq i \leq k\}$

Víme, že jazyk $L_1 = \{a^n b^n \mid n \geq 1\}$ je bezkontextový, a je zřejmé, že $L_{13} = L_1^*$ a proto i jazyk $L_{13}$ je bezkontextový.

Nesmíme zapomenout, že vlastně ani uzávěrové vlastnosti nejsou ekvivalence, ale pouze implikace (jestliže $L_1$ a $L_2$ jsou bezkontextové, pak ...). Důsledky můžeme vidět na příkladu

**Příklad 1.19**



Víme, že jazyk $L_3 = \left\{ a^{n^2} \;\middle|\; n \geq 0 \right\}$ není bezkontextový (to jsme dokázali v příkladu 1.3 na straně 7 pomocí Pumping lemma). Zvolíme následující bezkontextovou (dokonce regulární) substituci:

$s(a) = b^*$

Po uplatnění substituce dostaneme jazyk $\{b^i \mid i \geq 0\}$, což je bezkontextový (dokonce regulární) jazyk. Proto není pravda, že když je výsledkem operace bezkontextový jazyk, tak by operandy operace měly být také.

Kapitola 2

# Zásobníkový automat

*V této kapitole se podíváme na vlastnosti zásobníkového automatu, který jsme si stručně popsali již na začátku kurzu Teorie jazyků a automatů I. Tento typ automatu si definujeme, podíváme se na jeho typy a budeme se zabývat vztahem zásobníkových automatů k bezkontextovým jazykům a také deterministickou variantou.*

## 2.1 Definice zásobníkového automatu

Zásobníkový automat dostaneme tak, že konečný automat obohatíme o zásobníkovou pásku a zajistíme, aby výpočet byl řízen především podle této zásobníkové pásky.

Zásobníkový automat pracuje takto:

- vyjme symbol na vrcholu zásobníku,
- může nebo nemusí přečíst jeden symbol ze vstupní pásky, pokud přečte, posune se o políčko dál,
- dále se rozhoduje podle
  - svého vnitřního stavu,
  - symbolu, který vyndal ze zásobníku,
  - pokud četl ze vstupní pásky, pak i podle přečteného symbolu,
- akce automatu spočívá v přechodu do některého dalšího stavu a v uložení řetězce znaků do zásobníku.

**Definice 2.1 (Zásobníkový automat)** *Zásobníkový automat je definován jako* $\mathcal{A} = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$*, kde*

- $Q$ *je konečná neprázdná množina stavů,*
- $\Sigma$ *je konečná neprázdná abeceda,*
- $\Gamma$ *je konečná neprázdná zásobníková abeceda (symboly, které mohou být v zásobníku),*
- $\delta$ *je přechodová funkce definovaná níže,*
- $q_0$ *je počáteční stav automatu,* $q_0 \in Q$*,*
- $Z_0$ *je počáteční zásobníkový symbol,* $Z_0 \in \Gamma$*,*
- $F$ *je množina koncových stavů,* $F \subseteq Q$ *(může být i prázdná).*

*Přechodová funkce* $\delta$ *je zobrazení* $\delta : \ Q \times (\Sigma \cup \{\varepsilon\}) \times \Gamma \longrightarrow Q \times \Gamma^*$*, lze zapsat také jako* $\delta(q_i, a, Z) \ni (q_j, \gamma),\ q_i, q_j \in Q,\ a \in (\Sigma \cup \{\varepsilon\},\ Z \in \Gamma,\ \gamma \in \Gamma^*$*.*

*Zásobníkový automat je obecně nedeterministický.*

**Definice 2.2 (Konfigurace zásobníkového automatu)** *Konfigurace výše definovaného zásobníkového automatu* $\mathcal{A}$ *je* $Q \times \Sigma^* \times \Gamma^*$ *(také* $(q, \alpha, \gamma),\ q \in Q,\ \alpha \in \Sigma^*,\ \gamma \in \Gamma^*$*).*

*Počáteční konfigurace je* $(q_0, w, Z_0)$*, kde* $w$ *je slovo, které bylo dáno na vstup automatu. Koncová konfigurace závisí na typu zásobníkového automatu.*

První člen konfigurace je stav, ve kterém se automat právě nachází, druhý je nepřečtená část vstupní pásky a třetí momentální obsah zásobníku.

**Definice 2.3 (Přechod mezi konfiguracemi)** *Přechod mezi konfiguracemi zásobníkového automatu je relace*

$$(q_i, a\alpha, Z\beta) \vdash (q_j, \alpha, \gamma\beta) \quad \Longleftrightarrow \quad (q_j, \beta) \in \delta(q_i, a, Z)$$

*kde* $a \in \Sigma \cup \{\varepsilon\},\ Z \in \Gamma \cup \{\varepsilon\},\ \alpha \in \Sigma^*$

Symbol $\vdash^*$ značí reflexivní tranzitivní uzávěr relace $\vdash^*$, symbol $\vdash^+$ je tranzitivní uzávěr této relace (jakýkoliv počet přechodů, alespoň jeden), symbol $\vdash^i$ znamená přesně $i$ přechodů mezi konfiguracemi.

**Definice 2.4 (Základní typy zásobníkových automatů)** *Rozeznáváme tyto základní typy zásobníkových automatů:*

Zásobníkový automat končící přechodem do koncového stavu *je* $\mathcal{A}_F$ *s koncovou konfigurací*

$$(q_f, \varepsilon, \gamma),\ q_f \in F,\ \gamma \in \Gamma^*$$

*a rozpoznávaný jazyk je*

$$L(\mathcal{A}_F) = \{w \in \Sigma^* \mid\ (q_0, w, Z_0) \vdash^* (q_f, \varepsilon, \gamma),\ q_f \in F,\ \gamma \in \Gamma^*\}$$

Zásobníkový automat končící s prázdným zásobníkem *je* $\mathcal{A}_\emptyset$ *s koncovou konfigurací* $(q, \varepsilon, \varepsilon),\ q \in Q$ *a rozpoznávaný jazyk je*

$$L(\mathcal{A}_\emptyset) = \{w \in \Sigma^* \mid\ (q_0, w, Z_0) \vdash^* (q, \varepsilon, \varepsilon),\ q \in Q\}$$

Zásobníkový automat končící přechodem do koncového stavu a prázdným zásobníkem *je* $\mathcal{A}_{F,\emptyset}$ *s koncovou konfigurací* $(q_f, \varepsilon, \varepsilon),\ q_f \in F$ *a rozpoznávaný jazyk je*

$$L(\mathcal{A}_{F,\emptyset}) = \{w \in \Sigma^* \mid\ (q_0, w, Z_0) \vdash^* (q_f, \varepsilon, \varepsilon),\ q_f \in F\}$$

Jak vidíme, tyto tři základní typy se liší především svou koncovou konfigurací:

- zásobníkový automat končící v koncovém stavu ukončí výpočet ve chvíli, kdy má přečtenou celou vstupní pásku (prostřední člen konfigurace je $\varepsilon$) a nachází se v některém koncovém stavu,
- zásobníkový automat končící s prázdným zásobníkem končí výpočet, když má přečtenou celou vstupní pásku a zároveň je prázdný zásobník,
- třetí typ je kombinací (průnikem) obou předchozích – musí být splněny obě podmínky.

Všechny tři typy zásobníkových automatů končí samozřejmě výpočet i tehdy, když nejsou v žádné koncové konfiguraci, ale do žádné další se nemohou dostat (přechodová funkce nedává možnost reagovat v daném stavu s daným obsahem zásobníku a vstupní pásky).

**Příklad 2.1**

Sestrojte zásobníkový automat (dále ZA) končící s prázdným zásobníkem rozpoznávající jazyk
$L_{14} = \{wcw^R \mid w \in \{a,b\}^*\}$

Vytvoříme zásobníkový automat rozpoznávající prázdným zásobníkem:
$\mathcal{A}_\emptyset = (\{q_0, q_1\},\ \{a,b\},\ \{a,b,Z_0\},\ q_0,\ Z_0,\ \delta,\ \emptyset)$

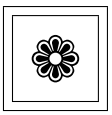
Automat bude pracovat takto:

- v první fázi bude číst obsah vstupu (první polovina slova) a ukládat do zásobníku (vždy co v každém kroku vyjmeme, vrátíme do zásobníku zároveň se symbolem ze vstupu, tedy ukládáme dva symboly), jsme ve stavu $q_0$,
- díky principu zásobníku (čteme v opačném pořadí, než jak byly symboly uloženy) je ukládaná první polovina slova zároveň zrcadlově převrácena,
- když na vstupu narazíme na $c$ (hranice, polovina slova), přejdeme do stavu $q_1$ a tím změníme způsob práce automatu,
- když jsme ve stavu $q_1$, nic do zásobníku neukládáme, symbol, který v každém kroku vyjmeme, porovnáme s tím, co je na vstupu – když souhlasí, můžeme pokračovat (tj. v každém kroku se posuneme na vstupu a zároveň ubereme symbol ze zásobníku).

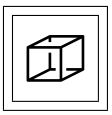
| | |
|---|---|
| $\delta(q_0, a, Z_0) = (q_0, aZ_0)$ | na začátku výpočtu, slovo začíná $a$ |
| $\delta(q_0, b, Z_0) = (q_0, bZ_0)$ | na začátku výpočtu, slovo začíná $b$ |
| $\delta(q_0, a, X) = (q_0, aX),\ X \in \{a,b\}$ | zatím jen načítáme a ukládáme do zásobníku |
| $\delta(q_0, c, X) = (q_1, X),\ X \in \{a,b\}$ | jsme na hranici |
| $\delta(q_1, a, a) = (q_1, \varepsilon)$ | shoda $a$ v obou polovinách slova |
| $\delta(q_1, b, b) = (q_1, \varepsilon)$ | shoda $b$ v obou polovinách slova |
| $\delta(q_1, \varepsilon, Z_0) = (q_1, \varepsilon)$ | skončili jsme na vstupu i v zásobníku |

Ukázka výpočtu automatu na slovo $abcba$:

| | |
|---|---|
| $(q_0, abcba, Z_0) \vdash (q_0, bcba, aZ_0) \vdash$ | $q_0$ : přenášíme do zásobníku obsah vstupu |
| $\vdash (q_0, cba, baZ_0) \vdash$ | hraniční bod, přejdeme do módu $q_1$ |
| $\vdash (q_1, ba, baZ_0) \vdash (q_1, a, aZ_0) \vdash$ | $q_1$ : jen vybíráme ze zásobníku a porovnáváme |
| $\vdash (q_1, \varepsilon, Z_0) \vdash (q_1, \varepsilon, \varepsilon)$ | konec |



**Poznámka:** Zásobníkový automat končící v koncovém stavu (a vlastně i „hybridní" typ) bychom z předešlého vytvořili jednoduše – stačí poslední část definice $\delta$ funkce přepsat takto:
$\delta(q_1, \varepsilon, Z_0) = (q_2, \varepsilon)$ s tím, že $Q = \{q_0, q_1, q_2\},\ F = \{q_2\}$.



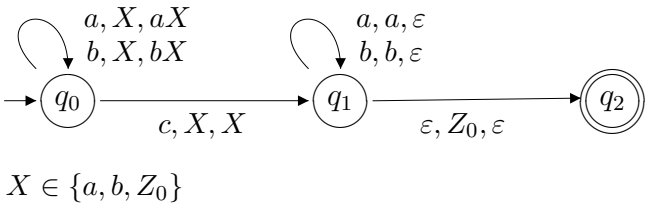
**Příklad 2.2**

Vytvořte zásobníkový automat končící v koncovém stavu reprezentovaný stavovým diagramem pro jazyk z příkladu 2.1
$L_{14} = \{wcw^R \mid w \in \{a, b\}^*\}$.

Narozdíl od konečných automatů, u zásobníkových nám nestačí ohodnotit šipky pouze symbolem načítaným ze vstupní pásky. Musíme vždy zadat tři údaje (ve správném pořadí!)

- symbol načtený ze vstupu (nebo $\varepsilon$, když ze vstupu nic nenačítáme),
- symbol, který vyjímáme ze zásobníku (nesmí zde být $\varepsilon$!, v každém kroku musíme nějaký symbol vyndat),
- řetězec, který ukládáme do zásobníku.

Diagram vytvoříme podle $\delta$ funkce v příkladu 2.1, jen navíc zakomponujeme změnu zahrnutou v poznámce nad tímto příkladem. Výsledný diagram vidíme na obrázku 2.1.

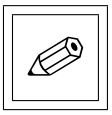



Obrázek 2.1: Diagram zásobníkového automatu

## 2.2 Vztah mezi různými typy zásobníkových automatů

Zde se podíváme na vztahy mezi třídami jazyků generovaných zásobníkovými automaty končícími v koncovém stavu, prázdným zásobníkem a nebo obojím.

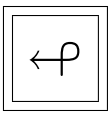
V dalším textu budeme používat toto označení:
$\mathscr{L}_F = \{L \mid L = L(\mathcal{A}_F)$ pro nějaký zásobníkový automat $\mathcal{A}_F\}$
. . . . . Třída jazyků generovaných ZA končícími v koncovém stavu
$\mathscr{L}_\emptyset = \{L \mid L = L(\mathcal{A}_\emptyset)$ pro nějaký zásobníkový automat $\mathcal{A}_\emptyset\}$
. . . . . Třída jazyků generovaných ZA končícími prázdným zásobníkem
$\mathscr{L}_{F,\emptyset} = \{L \mid L = L(\mathcal{A}_{F,\emptyset})$ pro nějaký zásobníkový automat $\mathcal{A}_F\}$
. . . . . Třída jazyků generovaných ZA končícími v koncovém stavu s prázdným zásobníkem



**Věta 2.1** *Nechť $\mathcal{A} = (Q, \Sigma, \Gamma, q_0, Z_0, \delta, \emptyset)$ je ZA končící výpočet prázdným zásobníkem. Potom existuje ZA $\mathcal{A}' = (Q', \Sigma, \Gamma', q_0', Z_0', \delta', F)$ končící výpočet v koncovém stavu takový, že $L(\mathcal{A}') = L(\mathcal{A})$. To znamená, že pro třídy jazyků generovaných jednotlivými typy ZA platí relace*

$$\mathscr{L}_\emptyset \subseteq \mathscr{L}_F \tag{2.1}$$

**Důkaz:** Budeme postupovat takto:



- do zásobníku dáme pomocnou „zarážku“ (počáteční zásobníkový symbol původního automatu $Z_0$) a dále budeme simulovat výpočet původního automatu,
- simulovaný výpočet probíhá nad vloženou „zarážkou“ a končí ve chvíli, kdy je tato zarážka vyjmuta (v té chvíli má simulovaný automat prázdný zásobník), v zásobníku je pouze $Z_0'$ (počáteční zásobníkový symbol nového automatu),
- pak jen přejdeme do koncového stavu; pokud je celá vstupní páska přečtená, pak končí výpočet s úspěchem, slovo je přijato.

$Q' = Q \cup \{q_0', q_f\},\ q_0' \notin Q, q_f \notin Q$
$F = \{q_f\}$
$\Gamma' = \Gamma \cup \{Z_0'\}$

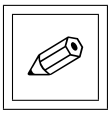
Funkci $\delta'$ definujeme takto:

- na začátku výpočtu přidáme do zásobníku počáteční zásobníkový symbol původního automatu, abychom mu pro simulaci připravili vhodné pracovní prostředí:

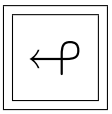
$$\delta'(q_0', \varepsilon, Z_0') = (q_0, Z_0 Z_0')$$

- potom simulujeme činnost původního automatu (tj. přejmeme chování jeho $\delta$ funkce):
$$\delta'(q,a,Z) = \delta(q,a,Z)\ \forall q \in Q, a \in \Sigma \cup \{\varepsilon\}, Z \in \Gamma$$
- po ukončení výpočtu původního automatu přejdeme do koncového stavu:
$$\delta'(q,\varepsilon,Z'_0) = (q_f,\varepsilon)$$

Přechod mezi konfiguracemi bude následující:
$(q'_0,w,Z'_0) \vdash (q_0,w,Z_0Z'_0) \vdash \ldots$ původní výpočet $\ldots \vdash (q,\varepsilon,Z'_0) \vdash (q_f,\varepsilon,\varepsilon)$ □



**Věta 2.2** *Nechť $\mathcal{A} = (Q,\Sigma,\Gamma,q_0,Z_0,\delta,F)$ je ZA končící výpočet v některém koncovém stavu. Potom existuje ZA $\mathcal{A}' = (Q',\Sigma,\Gamma',q'_0,Z'_0,\delta',\emptyset)$ končící výpočet prázdným zásobníkem takový, že $L(\mathcal{A}') = L(\mathcal{A})$. To znamená, že pro třídy jazyků generovaných jednotlivými typy ZA platí relace*
$$\mathscr{L}_F \subseteq \mathscr{L}_\emptyset \tag{2.2}$$



**Důkaz:** Abychom se v průběhu výpočtu „náhodou" nedostali na dno a tak předčasně neukončili výpočet, tak stejně jako v předchozím důkazu do zásobníku dáme pomocnou „zarážku" (počáteční zásobníkový symbol původního automatu $Z_0$) a dále budeme simulovat výpočet původního automatu.

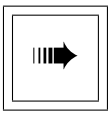
Nejdřív si definujeme funkci $\delta'$:

- na začátku výpočtu přidáme do zásobníku počáteční zásobníkový symbol původního automatu:
$$\delta'(q'_0,\varepsilon,Z'_0) = (q_0,Z_0Z'_0)$$
- potom simulujeme činnost původního automatu:
$$\delta'(q,a,Z) = \delta(q,a,Z)\ \forall q \in Q, a \in \Sigma \cup \{\varepsilon\}, Z \in \Gamma$$
- po ukončení výpočtu původního automatu vyprázdníme zásobník, ale nesmíme zapomenout na to, že simulovaný automat by v koncovém stavu ještě mohl chtít pracovat (stav, ve kterém bude mazán zásobník, je $d$ – delete):
$$\delta'(q_f,\varepsilon,Z) = \delta(q_f,\varepsilon,Z) \cup \{(d,\varepsilon)\}$$
$$\delta'(d,\varepsilon,Z) = (d,\varepsilon), \quad \forall Z \in \Gamma'$$

$Q' = Q \cup \{q'_0,d\},\ q'_0 \notin Q, d \notin Q,\ \Gamma' = \Gamma \cup \{Z'_0\}$

Přechod mezi konfiguracemi bude následující:
$(q'_0,w,Z'_0) \vdash (q_0,w,Z_0Z'_0) \vdash \ldots$ původní výpočet $\ldots \vdash (q_f,\varepsilon,Z\alpha Z'_0) \vdash$
$\vdash (d,\varepsilon,\alpha Z'_0) \vdash \ldots \vdash (d,\varepsilon,\varepsilon)$ □

**Důsledek 2.3** *Třídy jazyků generovaných zásobníkovými automaty končícími v koncovém stavu a zásobníkovými automaty končícími s prázdným zásobníkem jsou ekvivalentní:*



$$\mathscr{L}_F = \mathscr{L}_\emptyset \tag{2.3}$$

*Dále z vlastností zásobníkových automatů generujících jazyky z třídy $\mathscr{L}_{F,\emptyset}$ plyne, že*

$$\mathscr{L}_{F,\emptyset} = \mathscr{L}_\emptyset = \mathscr{L}_F \tag{2.4}$$

Množinu všech zásobníkových automatů budeme souhrnně označovat $ZA$, třídu (množinu) jazyků rozpoznávaných zásobníkovými automaty $\mathscr{L}(ZA)$.

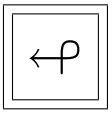
## 2.3 Vztah zásobníkových automatů a bezkontextových jazyků

Nadále budeme počítat s tím, že zásobníkové automaty všech tří základních typů jsou navzájem ekvivalentní, a tedy generují stejné třídy jazyků. Proto v důkazech budeme volně tyto typy zaměňovat.

**Věta 2.4** *Ke každé bezkontextové gramatice $G$ lze vytvořit zásobníkový automat $\mathcal{A}$ takový, že $L(G) = L(A)$, tedy*

$$\mathscr{L}(CF) \subseteq \mathscr{L}(ZA) \tag{2.5}$$

**Důkaz:** Máme bezkontextovou gramatiku $G = (N, T, P, S)$ a chceme sestrojit zásobníkový automat $\mathcal{A} = (Q, \Sigma, \Gamma, q_0, Z_0, \delta, \emptyset)$ (automat končí s prázdným zásobníkem). Budeme postupovat takto:



- v zásobníku mohou být neterminály a terminály, ze kterých se skládají pravidla, pravé strany pravidel ukládáme do zásobníku,
- pokud je na vrcholu zásobníku neterminál, vyjmeme ho a nahradíme pravou stranou některého pravidla přepisujícího tento neterminál,
- pokud ze zásobníku vyjme terminální symbol, načte symbol ze vstupu a porovná – pokud jsou stejné, může dál pokračovat (do zásobníku už vyjmutý symbol nevrací, a na vstupu se při přečtení vstupního symbolu posune na další).

$Q = \{q\},\ q_0 = q,\ \Sigma = T$ (terminální symboly použijeme pro abecedu),
$\Gamma = N \cup T,\ Z_0 = S$ (startovací symbol použijeme jako počáteční symbol zásobníku).

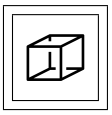
Pro každé pravidlo $A \to \alpha$ vytvoříme
$\delta(q, \varepsilon, A) \ni (q, \alpha)$ (ze zásobníku vyjmeme $A$ a nahradíme ho řetězcem $\alpha$)

Pro každý terminální symbol $a \in T$ vytvoříme
$\delta(q, a, a) = (q, \varepsilon, \varepsilon)$ (tentýž symbol vyndáme ze zásobníku i přečteme ze vstupu).

Shrňme si celou $\delta$ funkci:
$\delta(q, \varepsilon, A) = \{(q, \alpha) \mid (A \to \alpha) \in P\}$
$\delta(q, a, a) = \{(q, \varepsilon)\}\ ,\ \forall a \in T$

Vytvořený zásobníkový automat je nedeterministický, protože v gramatice je obvykle více pravidel pro stejný neterminální symbol. Automat jednoduše simuluje odvozování v gramatice – na zásobníku provádí „odvozování" stejně, jako bychom v gramatice přepisovali postupně neterminály v přepisovaném slově zleva. □

**Příklad 2.3**



$G = (\{S\},\ \{a, b, c\},\ P,\ S)$, kde $P = \{S \to aSa \mid bSb \mid c\}$

Tato gramatika generuje nám již známý jazyk $L_{14} = \left\{wcw^R \mid w \in \{a, b\}^*\right\}$

Vytvoříme zásobníkový automat $\mathcal{A} = (\{q\},\ \{a, b, c\},\ \{S, a, b, c\},\ q,\ S,\ \emptyset)$ generující tento jazyk.
$\delta(q, \varepsilon, S) = \{(q, aSa),\ (q, bSb),\ (q, c)\}$
$\delta(q, a, a) = \{(q, \varepsilon)\}$
$\delta(q, b, b) = \{(q, \varepsilon)\}$

Ukázka rozpoznání slova $abcba$:

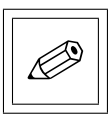
$(q, abcba, S) \vdash (q, abcba, aSa) \vdash (q, bcba, Sa) \vdash (q, bcba, bSba) \vdash (q, cba, Sba) \vdash$
$\vdash (q, cba, cba) \vdash (q, ba, ba) \vdash (q, a, a) \vdash (q, \varepsilon, \varepsilon)$

Ukázka neúspěšného výpočtu (slovo $acb$ bude odmítnuto):

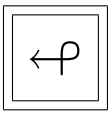
$(q, acb, S) \vdash (q, acb, aSa) \vdash (q, cb, Sa) \vdash (q, cb, ca) \vdash (q, b, a) \vdash \times$

Následující větu využijeme v důkazu věty 2.6 na straně 33.

**Věta 2.5** *Ke každému zásobníkovému automatu $\mathcal{A}$ existuje jednostavový zásobníkový automat $\mathcal{A}'$ takový, že $L(\mathcal{A}') = L(\mathcal{A})$.*



**Důkaz:** Původní a vytvářený automat jsou:
$\mathcal{A} = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, \emptyset)$
$\mathcal{A}' = (\{s\}, \Sigma, \Gamma', \delta', s, Z_0', \emptyset)$



Zásobníkový automat se v každém kroku obvykle rozhoduje podle tří kritérií – stavu vstupní pásky, stavu zásobníku a svého vnitřního stavu, ale když má k dispozici jen jediný vnitřní stav, musí se umístění této informace nahradit něčím jiným. Vstupní pásku nesmíme pozměnit, zbývá jen zásobník.

Tedy informaci původně uloženou ve vnitřním stavu přesuneme do zásobníku tak, že místo „jednoduchých" zásobníkových symbolů budeme používat uspořádané trojice, jejichž druhý prvek je některý symbol původní zásobníkové abecedy, první a třetí prvek jsou stavy:

$$\Gamma' = \Big\{ [q_i, Z, q_j] \;\Big|\; q_i, q_j \in Q,\ Z \in \Gamma \Big\}$$

- $q_i$ je stav, ve kterém je $Z$ na vrcholu zásobníku původního automatu (a tedy se v tom stavu vybírá ze zásobníku),
- $q_j$ je stav, do kterého přecházíme při vyjmutí všeho, co může být vygenerováno pomocí $Z$, ze zásobníku v původním automatu.

V zásobníku tedy kromě původních zásobníkových symbolů ukládáme také informaci o tom, v jakém stavu jsou tyto symboly zpracovávány a do jakého stavu přecházíme po jejich plném zpracování v původním automatu.

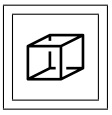
Nyní definujeme přechodovou funkci:

1. Na začátku výpočtu připravíme simulaci původního automatu:
$$\delta'(s, \varepsilon, Z_0') = \Big\{ \Big(s, [q_0, Z_0, p]\Big) \;\Big|\; p \in Q \Big\}$$
2. Pro všechny funkce automatu $\mathcal{A}$ ve tvaru $\delta(p, a, Z) \ni (q, \varepsilon)$ (do zásobníku nic nevkládají) vytvoříme
$$\delta'\Big(s, a, [p, Z, q]\Big) \ni (s, \varepsilon), \quad a \in \Sigma \cup \{\varepsilon\}$$
3. Pro všechny ostatní funkce ve tvaru $\delta(p, a, Z) \ni (q, B_1 B_2 \dots B_n)$:
$$\delta'\Big(s, a, [p, Z, u_n]\Big) \supseteq \Big\{ \Big(s, [q, B_1, u_1][u_1, B_2, u_2][u_2, B_3, u_3] \dots [u_{n-1} B_n u_n]\Big) \;\Big|\; \forall \text{ kombinace stavů } u_i \in Q, 1 \le i \le n,\ a \in \Sigma \cup \{\varepsilon\} \Big\}$$

Nejnáročnější je určitě poslední bod. Nové zásobníkové symboly zde určují všechny možné posloupnosti, jakými lze v původním automatu dojít ze stavu $q$, ve kterém vybíráme ze zásobníku symbol $Z$, do některého stavu $u_n$, do kterého přecházíme po zpracování posledního zde vkládaného symbolu, $B_n$. Musí zde být všechny možné kombinace $n$-tic původních stavů, protože nemůžeme předvídat, jaká posloupnost zpracovávaných stavů bude v těchto $n$ krocích použita.

Je zřejmé, že symbol $Z$ je vyjmut ve stavu $p$ a hned přecházíme do stavu $q$; zároveň vkládáme do zásobníku symboly $B_1, B_2, \ldots, B_n$, a to počínaje symbolem $B_n$ (symbol $B_1$ pak bude na vrcholu zásobníku), proto ze stavu $q$ po vyjmutí symbolu $B_1$ přecházíme do stavu $u_1$, atd. Až zpracujeme vše, co bylo vygenerováno ze symbolu $Z$ (tj. všechny symboly $B_1, \ldots, B_n$), dostaneme se do stavu $u_n$. □

**Příklad 2.4**



Postup ukážeme na jazyku
$L_{15} = \{a^n c\, b^n c^k \mid n \geq 0,\ k > 0\}$

Tento jazyk rozpoznává zásobníkový automat
$\mathcal{A} = (\{0,1\},\ \{a,b,c\},\ \{Z_0, a\},\ \delta,\ 0,\ Z_0,\ \emptyset)$

$\delta(0,a,X) = (0,aX), \quad X \in \{a, Z_0\}$
$\delta(0,c,X) = (1,X), \quad X \in \{a, Z_0\}$
$\delta(1,b,a) = (1,\varepsilon)$
$\delta(1,c,Z_0) = \{(1,Z_0),\ (1,\varepsilon)\}$

Vytvoříme automat $\mathcal{A}'$:

$\delta'\Big(s,\varepsilon,Z_0'\Big) = \Big\{\Big(s,[0,Z_0,0]\Big),\ \Big(s,[0,Z_0,1]\Big)\Big\}$
$\delta'\Big(s,a,[0,X,0]\Big) = \Big\{\Big(s,[0,a,0][0,X,0]\Big),\ \Big(s,[0,a,1][1,X,0]\Big)\Big\},\ X \in \{a,Z_0\}$
$\delta'\Big(s,a,[0,X,1]\Big) = \Big\{\Big(s,[0,a,0][0,X,1]\Big),\ \Big(s,[0,a,1][1,X,1]\Big)\Big\}$
$\delta'\Big(s,c,[0,X,0]\Big) = \Big\{\Big(s,[1,X,0]\Big)\Big\}$
$\delta'\Big(s,c,[0,X,1]\Big) = \Big\{\Big(s,[1,X,1]\Big)\Big\}$
$\delta'\Big(s,b,[1,a,1]\Big) = \{(s,\varepsilon)\}$
$\delta'\Big(s,c,[1,Z_0,0]\Big) = \Big\{\Big(s,[1,Z_0,0]\Big)\Big\}$
$\delta'\Big(s,c,[1,Z_0,1]\Big) = \Big\{\Big(s,[1,Z_0,1]\Big),\ (s,\varepsilon)\Big\}$

Nové zásobníkové symboly jsou možná přehledné z hlediska vytvoření, ale bude lepší nahradit je kratšími variantami podle následující tabulky.

| *Dlouhé označení* | $\longrightarrow$ | *Krátké označení* |
|---|---|---|
| $[0, Z_0, 0]$ | $\longrightarrow$ | $A$ |
| $[0, Z_0, 1]$ | $\longrightarrow$ | $B$ |
| $[1, Z_0, 0]$ | $\longrightarrow$ | $C$ |
| $[1, Z_0, 1]$ | $\longrightarrow$ | $D$ |
| $[0, a, 0]$ | $\longrightarrow$ | $E$ |
| $[0, a, 1]$ | $\longrightarrow$ | $F$ |
| $[1, a, 0]$ | $\longrightarrow$ | $G$ |
| $[1, a, 1]$ | $\longrightarrow$ | $H$ |

Upravíme $\delta'$ funkci:

$\delta'(s, \varepsilon, Z_0') = \{(s, A),\ (s, B)\}$

$\delta'(s, a, A) = \{(s, EA),\ (s, FC)\}$
$\delta'(s, a, E) = \{(s, EE),\ (s, FG)\}$
$\delta'(s, a, B) = \{(s, EB),\ (s, FD)\}$
$\delta'(s, a, F) = \{(s, EF),\ (s, FH)\}$

$\delta'(s, b, H) = \{(s, \varepsilon)\}$

$\delta'(s, c, A) = \{(s, C)\}$
$\delta'(s, c, E) = \{(s, G)\}$
$\delta'(s, c, B) = \{(s, D)\}$
$\delta'(s, c, F) = \{(s, H)\}$

$\delta'(s, c, C) = \{(s, C)\}$
$\delta'(s, c, D) = \{(s, D),\ (s, \varepsilon)\}$

Automat potom můžeme určit takto:

$\mathcal{A}' = (\{s\},\ \{a, b, c\},\ \{Z_0', A, B, C, D, E, F, G, H\},\ \delta',\ s,\ Z_0',\ \emptyset)$

Ukázka rozpoznání slova $aacbbc$ automatem $\mathcal{A}$:

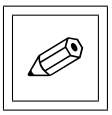
$(0, aacbbc, Z_0) \vdash (0, acbbc, aZ_0) \vdash (0, cbbc, aaZ_0) \vdash (1, bbc, aaZ_0) \vdash (1, bc, aZ_0) \vdash$
$\vdash (1, c, Z_0) \vdash (1, \varepsilon, \varepsilon)$

Ukázka rozpoznání slova $aacbbc$ automatem $\mathcal{A}'$:

$(s, aacbbc, Z_0') \vdash (s, aacbbc, B) \vdash (s, acbbc, FD) \vdash$
$\vdash (s, cbbc, FHD) \vdash (s, bbc, HHD) \vdash (s, bc, HD) \vdash (s, c, D) \vdash (s, \varepsilon, \varepsilon)$

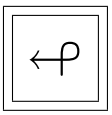
Totéž, ale bez nahrazení zásobníkových symbolů kratšími verzemi:

$(s, aacbbc, Z_0') \vdash \Big(s, aacbbc, [0, Z_0, 1]\Big) \vdash \Big(s, acbbc, [0, a, 1][1, Z_0, 1]\Big) \vdash$
$\vdash \Big(s, cbbc, [0, a, 1][1, a, 1][1, Z_0, 1]\Big) \vdash (s, bbc, [1, a, 1][1, a, 1][1, Z_0, 1]\Big) \vdash$
$\vdash \Big(s, bc, [1, a, 1][1, Z_0, 1]\Big) \vdash \Big(s, c, [1, Z_0, 1]\Big) \vdash (s, \varepsilon, \varepsilon)$



**Věta 2.6** *Ke každému zásobníkovému automatu $\mathcal{A}$ existuje bezkontextová gramatika $G$ taková, že $L(G) = L(\mathcal{A})$, tedy*

$$\mathscr{L}(ZA) \subseteq \mathscr{L}(CF) \tag{2.6}$$



**Důkaz:** Když jsme dokazovali, že ke každé bezkontextové gramatice lze sestrojit ekvivalentní zásobníkový automat (v důkazu věty 2.4 na straně 28), vytvořili jsme jednostavový zásobníkový automat. Zde využijeme přesně opačný postup – podle jednostavového automatu vytvoříme gramatiku.

Prvním krokem tedy bude vytvoření jednostavového zásobníkového automatu $\mathcal{A}'$ k automatu $\mathcal{A}$ podle postupu popsaného v důkazu věty 2.5. V druhém kroku vytvoříme gramatiku, jejíž neterminály vytvoříme ze zásobníkových symbolů.

Když oba kroky shrneme, postup je následující:

1. Inicializujeme výpočet:

$$\forall q \in Q: \quad S \to [q_0, Z_0, q]$$

2. Pro všechny funkce automatu $\mathcal{A}$ ve tvaru $\delta(p, a, Z) \ni (q, \varepsilon)$ (do zásobníku nic nevkládají) vytvoříme

$$[p, Z, q] \to a$$

3. Pro všechny ostatní funkce ve tvaru $\delta(p, a, Z) \ni (q, B_1 B_2 \dots B_n)$:

$$[p, Z, u_n] \to a[q, B_1, u_1][u_1, B_2, u_2] \dots [u_{n-1}, B_n, u_n]$$

pro každou kombinaci stavů $u_i \in Q,\ 1 \leq i \leq n$.

□

**Příklad 2.5**

Budeme pokračovat v příkladu 2.4 na straně 31. V zadání příkladu byl automat $\mathcal{A} = (\{0,1\},\ \{a,b,c\},\ \{Z_0, a\},\ \delta,\ 0,\ Z_0,\ \emptyset)$

$\delta(0, a, X) = (0, aX), \quad X \in \{a, Z_0\}$
$\delta(0, c, X) = (1, X), \quad X \in \{a, Z_0\}$
$\delta(1, b, a) = (1, \varepsilon)$
$\delta(1, c, Z_0) = \{(1, Z_0),\ (1, \varepsilon)\}$

Podle popsaného postupu vytvoříme gramatiku $G = (N, T, P, S),\ T = \Sigma$, s pravidly

$S \to [0, Z_0, 0] \mid [0, Z_0, 1]$

$[0, Z_0, 0] \to a[0, a, 0][0, Z_0, 0] \mid a[0, a, 1][1, Z_0, 0]$
$[0, a, 0] \to a[0, a, 0][0, a, 0] \mid a[0, a, 1][1, a, 0]$

$[0, Z_0, 1] \to a[0, a, 0][0, Z_0, 1] \mid a[0, a, 1][1, Z_0, 1]$
$[0, a, 1] \to a[0, a, 0][0, a, 1] \mid a[0, a, 1][1, a, 1]$

$[0, Z_0, 0] \to c[1, Z_0, 0]$
$[0, a, 0] \to c[1, a, 0]$
$[0, Z_0, 1] \to c[1, Z_0, 1]$
$[0, a, 1] \to c[1, a, 1]$

$[1, a, 1] \to b$

$[1, Z_0, 0] \to c[1, Z_0, 0]$
$[1, Z_0, 1] \to c[1, Z_0, 1] \mid c$

Zjednodušíme neterminály a shrneme pravidla přepisující stejný neterminál:

$S \to A \mid B$
$A \to aEA \mid aFC \mid cC$
$B \to aEB \mid aFD \mid cD$
$C \to cC$
$D \to cD \mid c$
$E \to aEE \mid aFG \mid cG$
$F \to aEF \mid aFH \mid cH$
$H \to b$

Odstraníme nadbytečné symboly:

$S \to B$
$B \to aFD \mid cD$
$C \to cC$
$D \to cD \mid c$
$F \to aFH \mid cH$
$H \to b$

Neterminály: $N = \{S, B, C, D, F, H\}$

Ukázka generování slova $aacbbc$:

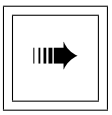
$S \Rightarrow B \Rightarrow aFD \Rightarrow aaFHD \Rightarrow aacHHD \Rightarrow aacbHD \Rightarrow aacbbD \Rightarrow$
$\Rightarrow aacbbcD \Rightarrow aacbbcc$

Dá se jednoduše dokázat, že generovaný jazyk je
$L_{15} = \left\{a^n c\, b^n c^k \mid n \geq 0,\ k > 0\right\}$

Důsledkem vět 2.4 na straně 28 a 2.6 na straně 33 je ekvivalence tříd jazyků:

**Důsledek 2.7**

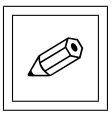
$$\mathscr{L}(ZA) \simeq \mathscr{L}(CF) \tag{2.7}$$

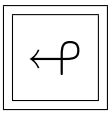


## 2.4 Zásobníkové automaty a uzávěrové vlastnosti bezkontextových jazyků

V sekci 1.3 na straně 15 jsme probrali téměř všechny operace, vzhledem k nimž je nebo není třída bezkontextových jazyků uzavřena, a dokázali jsme (věta 1.13 na straně 18), že třída bezkontextových jazyků není uzavřena vzhledem k operaci průniku. To však neplatí pro průnik s regulárním jazykem:

**Věta 2.8** *Třída bezkontextových jazyků je uzavřena vzhledem k průniku s regulárním jazykem.*





**Důkaz:** Narozdíl od jiných uzávěrových vlastností bezkontextových jazyků, zde konstrukci nebudeme provádět na gramatikách, ale na automatech. Postup bude podobný tomu, který jsme použili v kurzu Teorie jazyků a automatů I pro průnik dvou regulárních jazyků.

Vytváříme průnik bezkontextového jazyka reprezentovaného zásobníkovým automatem $\mathcal{A}_1$ a regulárního jazyka reprezentovaného konečným automatem $\mathcal{A}_2$:
$\mathcal{A}_1 = (Q_1, \Sigma_1, \Gamma_1, \delta_1, q_0^{(1)}, Z_0^{(1)}, F_1)$ (rozpoznává koncovým stavem)
$\mathcal{A}_2 = (Q_2, \Sigma_2, \delta_2, q_0^{(2)}, F_2)$

Sestrojíme $\mathcal{A} = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F), \quad L(\mathcal{A}) = L(\mathcal{A}_1) \cap L(\mathcal{A}_2)$.

Je zřejmé, že $\Sigma = \Sigma_1 \cup \Sigma_2,\ \Gamma = \Gamma_1,\ Z_0 = Z_0'$.

Výpočet v automatu $\mathcal{A}$ má být simultánní simulací obou původních automatů, slovo, které má být rozpoznáno, prostě zároveň dáme na vstup obou původních automatů a přijmeme ho, pokud v obou automatech bude existovat výpočet od počáteční k některé konečné konfiguraci.

Stavy automatu $\mathcal{A}$ budou uspořádané dvojice stavů původních automatů, první prvek je stav automatu $\mathcal{A}_1$ a druhý je stav automatu $\mathcal{A}_2$. Uspořádaná dvojice zachycuje, v jakém stavu v původních automatech je právě simulovaný výpočet.
$Q = \{[q_1, q_2] \mid q_1 \in Q_1,\ q_2 \in Q_2\}$

Množina koncových stavů: $F = \{[q_1, q_2] \mid q_1 \in F_1,\ q_2 \in F_2\}$

Definujeme $\delta$ funkci:

1. V každém kroku, ve kterém je čten symbol ze vstupu, se posouváme v obou simulovaných automatech, v tom zásobníkovém také pracujeme se zásobníkem – pro každé $a \in \Sigma,\ q_1, p_1 \in Q_1,\ q_2, p_2 \in Q_2,\ Z \in \Gamma,\ \gamma \in \Gamma_1^*$:

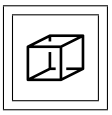
$$\delta([q_1, q_2], a, Z) \ni ([p_1, p_2], \gamma) \quad \Longleftrightarrow \quad \delta_1(q_1, a, Z) \ni (p_1, \gamma),\ \delta_2(q_2, a) \ni p_2$$

2. Vyřešíme odlišnost původních automatů při práci se vstupní abecedou. Zásobníkový automat nemusí v každém kroku číst ze vstupní pásky, kdežto konečný ano. Proto umožníme simulaci konečného automatu dělat $\varepsilon$-kroky, při kterých nebude číst ze vstupní pásky ani provádět změnu stavu – pro každé $q_1, p_1 \in Q_1,\ q_2 \in Q_2,\ Z \in \Gamma,\ \gamma \in \Gamma_1^*$:

$$\delta([q_1, q_2], \varepsilon, Z) \ni ([p_1, q_2], \gamma) \quad \Longleftrightarrow \quad \delta_1(q_1, \varepsilon, Z) \ni (p_1, \gamma)$$

□

**Příklad 2.6**



Vezmeme bezkontextový jazyk $L_{16} = \{ww^R \mid w \in \{a, b\}^*\}$ a regulární jazyk $a^*$. Jejich průnikem je jazyk $L_{17} = \{a^n a^n \mid n \geq 0\} = \{a^{2n} \mid n \geq 0\}$

Sestrojíme automat $\mathcal{A}_1,\ L(\mathcal{A}_1) = L_{16}$:

$\mathcal{A}_1 = (\{q_0, q_1, q_2\},\ \{a, b\},\ \{Z_0, a, b\},\ \delta_1,\ q_0,\ Z_0,\ \{q_2\})$, kde

$\delta_1(q_0, a, Z_0) = \{(q_0, aZ_0)\}$
$\delta_1(q_0, b, Z_0) = \{(q_0, bZ_0)\}$
$\delta_1(q_0, a, b) = \{(q_0, ab)\}$
$\delta_1(q_0, b, a) = \{(q_0, ba)\}$
$\delta_1(q_0, a, a) = \{(q_0, aa), (q_1, \varepsilon)\}$
$\delta_1(q_0, b, b) = \{(q_0, bb), (q_1, \varepsilon)\}$
$\delta_1(q_1, a, a) = \{(q_1, \varepsilon)\}$
$\delta_1(q_1, b, b) = \{(q_1, \varepsilon)\}$
$\delta_1(q_1, \varepsilon, Z_0) = \{(q_2, \varepsilon)\}$

Konečný automat pro jazyk $a^*$ je velmi jednoduchý:

$\mathcal{A}_2 = (\{r\}\ \{a\},\ \delta_2,\ r,\ \{r\})$, kde $\delta_2(r, a) = r$

Vytvoříme $\mathcal{A} = (Q,\ \{a, b\},\ \{Z_0, a, b\},\ \delta,\ [q_0, r],\ Z_0,\ F)$.

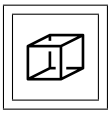
$\delta([q_0, r], a, Z_0) = \{[q_0, r], aZ_0])\}$
$\delta([q_0, r], a, a) = \{([q_0, r], aa), ([q_1, r], \varepsilon)\}$
$\delta([q_1, r], a, a) = \{[q_1, r], \varepsilon)\}$
$\delta([q_1, r], \varepsilon, Z_0) = \{([q_2, r], \varepsilon)\}$

Ještě doplníme:

$Q = \{[q_0, r],\ [q_1, r],\ [q_2, r]\}$
$F = \{[q_2, r]\}$

Jak vidíme, je jazyk $L_{17}$ průnikem bezkontextového a regulárního jazyka, proto (i bez konstrukce automatu rozpoznávajícího tento jazyk) můžeme říci, že je to bezkontextový jazyk.

**Příklad 2.7**



Pomocí uzávěrových vlastností dokážeme, že není bezkontextový jazyk
$L_{18} = \left\{ w \in \{a,b,c\}^* \;\middle|\; |w|_a = |w|_b = |w|_c \right\}$ (stejný počet $a$, $b$ a $c$)

Předpokládejme, že jazyk $L_{18}$ je bezkontextový. Pak by průnikem tohoto jazyka s jakýmkoliv regulárním jazykem byl také bezkontextový jazyk. Vezmeme regulární jazyk $R = a^*b^*c^*$. Jejich průnikem je

$L_{18} \cap R = L_2 = \{a^n b^n c^n \mid n \geq 0\}$

O tomto jazyce však víme, že není bezkontextový, proto $L_{18} \notin \mathscr{L}(CF)$.

## 2.5 Deterministické bezkontextové jazyky

### 2.5.1 Deterministický zásobníkový automat

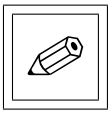
**Definice 2.5 (Deterministický ZA)** *Zásobníkový automat $\mathcal{A} = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$ je deterministický, jestliže pro každé $q \in Q,\ Z \in \Gamma$ platí zároveň*

- *$\delta(q, a, Z)$ má nejvýše jeden prvek pro každé $a \in \Sigma \cup \{\varepsilon\}$.*
- *je-li $\delta(q, \varepsilon, Z) \neq \emptyset$, pak $\delta(q, a, X) = \emptyset\ \forall a \in \Sigma$.*

To znamená, že v deterministickém zásobníkovém automatu máme v každém kroku právě jednu možnost, jak reagovat (i včetně rozhodování, zda máme číst ze vstupní pásky).

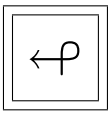
**Definice 2.6 (Deterministický bezkontextový jazyk)** *Jazyk $L$ se nazývá deterministický bezkontextový jazyk, jestliže existuje deterministický zásobníkový automat (DZA) $\mathcal{A}_D$ takový, že $L(\mathcal{A}_D) = L$.*

Z definice vyplývá, že pro každé slovo patřící do jazyka existuje právě jeden výpočet v $\mathcal{A}_D$. Deterministické bezkontextové jazyky budeme značit $DCF$ a třídu jazyků, které generují, $\mathscr{L}(DCF)$.



**Věta 2.9** *Třída jazyk§ generovaných deterministickými zásobníkovými automaty je vlastní podmnožinou třídy bezkontextových jazyků:*

$$\mathscr{L}(DCF) \subset \mathscr{L}(CF) \tag{2.8}$$



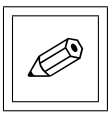
**Důkaz:** To, že platí $\mathscr{L}(DCF) \subseteq \mathscr{L}(CF)$, je zřejmé – vyplývá to z toho, že deterministický zásobníkový automat je vlastně speciálním případem (obobecného) zásobníkového automatu, a víme (viz důsledek 2.7), že třída jazyků generovaných bezkontextovými jazyky je ekvivalentní třídě jazyků rozpoznávaných zásobníkovými automaty.

Vlastní inkluzi (tj. podmnožinu, $\mathscr{L}(DZA) \subsetneq \mathscr{L}(CF)$) lze dokázat tak, že najdeme jazyk patřící do druhé, ale nepatřící do první třídy. Bezkontextovým jazykem, který není deterministickým bezkontextovým, je například
$L_{16} = \left\{ww^R \mid w \in \{a,b\}^*\right\}$
(zásobníkový automat pro tento jazyk je v příkladu 2.6 na straně 36).

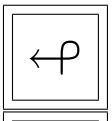
Tento jazyk je velmi podobný jazyku $L_{14}$, ale slova nejsou v polovině rozdělena znakem $c$.

Zásobníkový automat pro tento jazyk je možné sestrojit podobně jako pro jazyk $L_{14}$ v příkladu 2.1 na straně 24, ale bude to rozhodně automat nedeterministický – nevíme, ve kterém okamžiku vlastně přecházíme do druhé poloviny rozpoznávaného slova (nemáme možnost si předem tuto informaci zjistit a obě poloviny slova mají podobnou strukturu), a proto v každém kroku při načítání první poloviny slova potřebujeme možnost nedeterministicky zvolit buď pokračování v první polovině slova, a nebo přechod do druhé. □

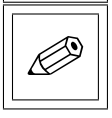
### 2.5.2 Uzávěrové vlastnosti deterministických bezkontextových jazyků



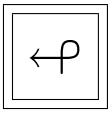
**Věta 2.10** *Třída jazyků $\mathscr{L}(DCF)$ je uzavřena vzhledem k operaci průniku s regulárním jazykem.*



**Důkaz:** Důkaz je stejný jako u (obecně) bezkontextových jazyků, viz věta 2.8 na straně 35. □



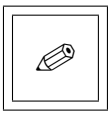
**Věta 2.11** *Třída jazyků $\mathscr{L}(DCF)$ není uzavřena vzhledem k operaci průniku.*



**Důkaz:** Důkaz je stejný jako u (obecně) bezkontextových jazyků, viz věta 1.13 na straně 18. □

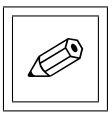
Dále budeme potřebovat tuto pomocnou větu (lemma):

**Lemma 2.12** *Ke každému DZA $\mathcal{A}$ lze zkonstruovat DZA $\mathcal{A}'$, který každý vstup dočte do konce.*

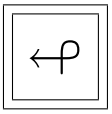


Důkaz je složitý, je třeba vyřešit problém zacyklení v epsilonových krocích (práce se zásobníkem).

**Věta 2.13** *Třída jazyků $\mathscr{L}(DCF)$ je uzavřena vzhledem k operaci doplňku.*



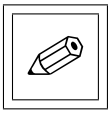
**Důkaz:** Každý deterministický zásobníkový automat, který vstup dočte do konce, dokáže v konečném počtu kroků rozhodnout, zda slovo patří nebo nepatří do jazyka rozpoznávaného tímto automatem (lemma 2.12). Proto je postup následující:



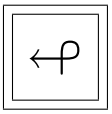
- sestrojíme k původnímu automatu $A$ zásobníkový automat $A'$, který čte každý vstup až do konce,
- zaměníme koncové a nekoncové stavy.

□

**Věta 2.14** *Třída jazyků $\mathscr{L}(DCF)$ není uzavřena vzhledem k operaci sjednocení.*



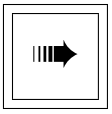
**Důkaz:** Vyplývá z De Morganových zákonů:



$$L_1 \cap L_2 = \overline{\overline{L_1} \cup \overline{L_2}} \tag{2.9}$$

Předpokládejme, že třída jazyků $\mathscr{L}(DCF)$ je uzavřena vzhledem k operaci sjednocení. Pak by na pravé straně vztahu (2.9) byla množina deterministických bezkontextových jazyků, jenže v množině na pravé straně rovnosti se mohou vyskytovat i jazyky, které nejsou deterministické bezkontextové (tato třída jazyků není uzavřena vzhledem k operaci průniku). Proto třída jazyků $\mathscr{L}(DCF)$ nemůže být uzavřena vzhledem k operaci sjednocení. □

**Důsledek 2.15**



$$\mathscr{L}(DCF) \subset \mathscr{L}(CF) \tag{2.10}$$

Kapitola 3

# Jazyky typu 0

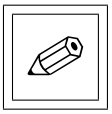
*V této kapitole se budeme zabývat nejvyšší třídou jazyků Chomského hierarchie s (téměř) obecným tvarem pravidel, jazyky typu 0. Stručně se podíváme na gramatiky typu 0 a pak se budeme věnovat především základní variantě Turingova stroje.*

## 3.1 Gramatiky typu 0

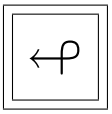
**Definice 3.1 (Gramatika typu 0)** *Gramatika typu 0 je taková gramatika, jejíž pravidla jsou ve tvaru*

$$\alpha \rightarrow \beta, \quad \alpha \in (N \cup T)^* N (N \cup T)^*, \ \beta \in (N \cup T)^*$$

**Definice 3.2 (Kurodova normální forma pro gramatiky typu 0)** *Gramatika typu 0 je v KNF, jestliže pro všechna její pravidla $\alpha \rightarrow \beta$ platí $|\alpha| \leq 2, \ |\beta| \leq 2$.*



**Věta 3.1** *Ke každé gramatice $G$ typu 0 lze sestrojit ekvivalentní gramatiku $G'$ v KNF.*



**Důkaz:**

- *pravidla v bezkontextovém tvaru:* použijeme algoritmus pro převod na Chomského NF.
- *pravidla, která nejsou bezkontextového typu:* upravíme pravidla podle následujícího algoritmu.

*Vstup:* gramatika bez jednoduchých pravidel typu $X \rightarrow Y, \quad X, Y \in N$

1. Pro všechna pravidla:
   - Všechny terminály (na levé i pravé straně pravidla) $a \in T$ nahradíme „pomocnými“ neterminály $N_a$.
   - Pro všechny neterminály vytvořené v předchozím bodu přidáme pravidlo $N_a \to a$.
2. Pravidla $A \to B_1B_2 \dots B_n,\ n > 2$ nahradíme pravidly
$$\begin{aligned} A &\to B_1X_1 \\ X_1 &\to B_2X_2 \\ &\vdots \\ X_{n-2} &\to B_{n-1}B_n \end{aligned}$$
3. Pravidla $A_1A_2 \dots A_m \to B_1B_2 \dots B_m,\ m > 2$ nahradíme pravidly
$$\begin{aligned} A_1A_2 &\to B_1X_1 \\ X_1A_3 &\to B_2X_2 \\ &\vdots \\ X_{m-2}A_m &\to B_{m-1}B_m \end{aligned}$$
4. Nezkracující pravidla $A_1A_2 \dots A_m \to B_1B_2 \dots B_n,\ 2 < m \leq n$ nahradíme pravidly
$$\begin{aligned} A_1A_2 &\to B_1X_1 \\ X_1A_3 &\to B_2X_2 \\ &\vdots \\ X_{m-2}A_m &\to B_{m-1}X_{m-1} \end{aligned} \qquad \begin{aligned} X_{m-1} &\to B_mX_m \\ X_m &\to B_{m+1}X_{m+1} \\ &\vdots \\ X_{n-2} &\to B_{n-1}B_n \end{aligned}$$
5. Zkracující pravidla $A_1A_2 \dots A_m \to B_1B_2 \dots B_n,\ n < m$ nahradíme pravidly
$$\begin{aligned} A_1A_2 &\to B_1X_1 \\ X_1A_3 &\to B_2X_2 \\ &\vdots \\ X_{n-1}A_{n+1} &\to B_nX_n \end{aligned} \qquad \begin{aligned} X_nA_{n+2} &\to X_{n+1} \\ &\vdots \\ X_{m-3}A_{m-1} &\to X_{m-2} \\ X_{m-2}A_m &\to \varepsilon \end{aligned}$$

□

**Příklad 3.1**

Postup si ukážeme na gramatice

$S \to AaBC$ ①
$C \to cBAa \mid bS$ ②,③
$AaBc \to d$ ④
$BA \to abcd$ ⑤

Provedeme náhradu terminálů $a \in T$ novými neterminály $N_a$. Týká se to pravidel ④ a ⑤. Pravidlo ③ nemusíme dále zpracovávat, už odpovídá Kurodově NF, po nahrazení terminálů je $C \to N_b S$.

Nejdřív zpracujeme pravidla bezkontextového typu ① a ②.

$S \to AX_1$ $C \to N_c X_3$
$X_1 \to N_a X_2$ $X_3 \to BX_4$
$X_2 \to BC$ $X_4 \to AN_a$

Zbývají pravidla ③ a ④ – jedno je zkracující a druhé nezkracující:

$BA \to N_a X_5$ $AN_a \to N_d X_7$
$N_a X_5 \to N_b X_6$ $X_7 B \to X_8$
$X_6 \to N_c N_d$ $X_8 N_c \to \varepsilon$

Celá gramatika po převodu do KNF:

| | | | |
|---|---|---|---|
| $S \to AX_1$ | $X_3 \to BX_4$ | $X_6 \to N_c N_d$ | $N_a \to a$ |
| $X_1 \to N_a X_2$ | $X_4 \to AN_a$ | $AN_a \to N_d X_7$ | $N_b \to b$ |
| $X_2 \to BC$ | $BA \to N_a X_5$ | $X_7 B \to X_8$ | $N_c \to c$ |
| $C \to N_c X_3$ | $N_a X_5 \to N_b X_6$ | $X_8 N_c \to \varepsilon$ | $N_d \to d$ |

## 3.2 Stroje rozpoznávající jazyky typu 0

Existují dva typy strojů (resp. matematických modelů), které rozpoznávají jazyky typu 0 – zásobníkový automat rozšířený o další zásobník a Turingův stroj.

### 3.2.1 Zásobníkový automat se dvěma zásobníky

Obecně můžeme mít jakýkoliv počet zásobníků, ale dá se dokázat, že k rozpoznávání jazyků typu 0 stačí dva zásobníky, resp. že ke každému zásobníkovému

automatu s jakýmkoliv počtem zásobníků je možno vytvořit ekvivalentní (rozpoznávající stejný jazyk) se dvěma zásobníky.

**Definice 3.3 (Zásobníkový automat se dvěma zásobníky)** *Zásobníkový automat se dvěma zásobníky je* $\mathcal{A}_2 = (Q, \Sigma, \Gamma_1, \Gamma_2, \delta, q_0, Z_1, Z_2, F)$*, kde*

- $Z_1 \in \Gamma_1$ *je počáteční zásobníkový symbol prvního zásobníku, abecedou prvního zásobníku je* $\Gamma_1$*,*
- $Z_2 \in \Gamma_2$ *je počáteční zásobníkový symbol druhého zásobníku, abecedou druhého zásobníku je* $\Gamma_2$*,*
- $\delta$ *funkce je definována takto:*
  $\delta : Q \times (\Sigma \cup \{\varepsilon\}) \times \Gamma_1 \times \Gamma_2 \rightarrow Q \times \Gamma_1^* \times \Gamma_2^*$
  $\delta(q_1, a, b_1, b_2) \in (q_2, \gamma_1, \gamma_2),\ a \in \Sigma \cup \{\varepsilon\},\ b_1 \in \Gamma_1,\ b_2 \in \Gamma_2,\ \gamma_1 \in \Gamma_1^*,\ \gamma_2 \in \Gamma_2^*$
- *vše ostatní se přejímá z definice zásobníkového automatu (s jedním zásobníkem).*

**Definice 3.4 (Konfigurace a přechod mezi konfiguracemi)** *Konfigurace zásobníkového automatu se dvěma zásobníky je*

$$(q, w, \gamma_1, \gamma_2) \in Q \times \Sigma^* \times \Gamma_1^* \times \Gamma_2^*$$

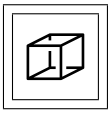
*(stav, nepřečtená část vstupu, obsah prvního zásobníku, obsah druhého zásobníku).*

*Přechod mezi konfiguracemi zásobníkového automatu se dvěma zásobníky je relace*

$$(q_i, a\alpha, b_1\gamma_1, b_2\gamma_2) \vdash (q_j, \beta_1\gamma_1, \beta_2\gamma_2) \quad \Longleftrightarrow \quad \delta(q_i, a, b_1, b_2) \ni (q_j, \beta_1, \beta_2)$$

Podobně jako u zásobníkových automatů s jedním zásobníkem bychom mohli definovat také reflexivní a tranzitivní uzávěr relace a také jazyk rozpoznávaný automatem, to necháváme na čtenáři, definice budou prakticky stejné.

**Příklad 3.2**



Vytvoříme zásobníkový automat se dvěma zásobníky pro jazyk, o kterém víme, že není bezkontextový:
$L_2 = \{a^n b^n c^n \mid n \geq 0\}$
$\mathcal{A}_2 = (\{q_0, q_1, q_2\},\ \{a, b, c\},\ \{a, Z_1\},\ \{b, Z_2\},\ \delta,\ Z_1,\ Z_2,\ \emptyset)$

$\delta$ funkce pracuje takto:

- v první fázi (stav $q_0$) načítáme symboly $a$ a ukládáme je do prvního zásobníku, druhý zásobník zatím není používán,

- v druhé fázi (stav $q_1$) načítáme symboly $b$, přitom vyjímáme z prvního zásobníku symboly $a$ (tak je zajištěn stejný počet $a$ a $b$) a zároveň ukládáme symboly $b$ do druhého zásobníku,
- v třetí fázi (stav $q_2$) musí již být první zásobník prázdný, načítáme ze vstupu symboly $c$ a zároveň vyjímáme z druhého zásobníku symboly $b$ (tak je zajištěn stejný počet $b$ a $c$).

$$\begin{array}{llll} \delta(q_0,\ a,\ Z_1,\ Z_2) = (q_0,\ aZ_1,\ Z_2) & & \delta(q_1,\ c,\ Z_1,\ b) = (q_2,\ Z_1,\ \varepsilon) \\ \delta(q_0,\ a,\ a,\ Z_2) = (q_0,\ aa,\ Z_2) & & \delta(q_2,\ c,\ Z_1,\ b) = (q_2,\ Z_1,\ \varepsilon) \\ \delta(q_0,\ b,\ a,\ Z_2) = (q_1,\ \varepsilon,\ bZ_2) & & \delta(q_0,\ \varepsilon,\ Z_1,\ Z_2) = (q_0,\ \varepsilon,\ \varepsilon) \\ \delta(q_1,\ b,\ a,\ b) = (q_1,\ \varepsilon,\ bb) & & \delta(q_2,\ \varepsilon,\ Z_1,\ Z_2) = (q_2,\ \varepsilon,\ \varepsilon) \end{array}$$

Ukázka rozpoznání slova $aabbcc$:

$(q_0, aabbcc, Z_1, Z_2) \vdash (q_0, abbcc, aZ_1, Z_2) \vdash (q_0, bbcc, aaZ_1, Z_2) \vdash (q_1, bcc, aZ_1, bZ_2) \vdash$
$\vdash (q_1, cc, Z_1, bbZ_2) \vdash (q_2, c, Z_1, bZ_2) \vdash (q_2, \varepsilon, Z_1, Z_2) \vdash (q_2, \varepsilon, \varepsilon, \varepsilon)$

---

### 3.2.2 Turingův stroj

Činnost Turingova stroje jsme si již trochu osvětlili v kurzu Teorie jazyků a automatů I. Shrneme si základní vlastnosti:

- konečná (konečněstavová) řídicí jednotka,
- nekonečná páska (obvykle doprava nekonečná),
- čtecí a zápisová hlava může číst symbol z pásky, přepsat ho jiným symbolem, pohybuje se o 1 políčko doleva nebo doprava,
- výpočet končí při přechodu do některého koncového stavu, nemusí být přečtená celá páska.

**Definice 3.5 (Turingův stroj)** *Turingův stroj je uspořádaná šestice* $\mathcal{M} = (Q, \Sigma, \Gamma, \delta, q_0, F)$, *kde*

- $Q$ *je konečná neprázdná množina stavů,*
- $\Sigma$ *je konečná neprázdná vstupní abeceda (symboly, ze kterých se může skládat vstupní slovo),* $\Sigma \subseteq \Gamma$,

- $\Gamma$ *je konečná neprázdná pásková abeceda (symboly, které se mohou vyskytovat na pásce),*
- $q_0 \in Q$ *je počáteční stav,*
- $F \subseteq Q$ *je množina koncových stavů,*
- $\delta$ *je přechodová funkce:*
  $\delta : \ Q \times \Gamma \ \rightarrow \ Q \times \Gamma \times \{-1, 0, 1\}$,
  *jinak* $\delta(q_i, a) \rightarrow (q_j, b, P),\ q_i, q_j \in Q,\ a, b \in \Gamma,\ P \in \{-1, 0, 1\}$

Podle definice $\delta$ funkce můžeme odvodit činnost Turingova stroje – v každém kroku, kdy je použito $\delta(q_i, a) \rightarrow (q_j, b, P),\ q_i, q_j \in Q,\ a, b \in \Gamma,\ P \in \{-1, 0, 1\}$, jsou provedeny následující akce:

- jsme ve stavu $q_i$ a na pásce čtecí a zápisová hlava právě ukazuje na políčko označené $a$,
- přejdeme do stavu $q_j$,
- symbol $a$ na pásce přepíšeme symbolem $b$,
- posuneme čtecí a zápisovou hlavu podle předpisu $P$.

Takto definovaný Turingův stroj je deterministický.

Prázdná políčka pásky se obvykle označují symbolem $\sqcup$ (nebo písmenem $B$, Blank), slovo je od prázdných políček odděleno (tedy obklopeno) symboly \$, tyto symboly v přechodové funkci pomáhají zjistit, zda jsme na začátku či konci slova. Je možné stanovit také dva různé symboly – jeden pro vymezení začátku a druhý pro vymezení konce slova (například \$ a #). Hraniční symbol (-y) je také součástí páskové abecedy.

Množina $F$ koncových stavů bývá často tvořena dvěma stavy, jedním pro přijetí a jedním pro odmítnutí slova: $F = \{q_{accept}, q_{reject}\}$, případně místo $q_{reject}$ je někdy použito $q_{error}$.

**Definice 3.6 (Konfigurace Turingova stroje)** *Konfigurace Turingova stroje* $\mathcal{M}$*, kde* $\mathcal{M} = (Q, \Sigma, \Gamma, \delta, q_0, F)$*, je* $(w_1, q, w_2)$*, kde* $w_1 \in \Gamma$ *je část pásky před čtecí a zápisovou hlavou,* $q \in Q$ *je stav, ve kterém se řídicí jednotka nachází a* $w_2 \in \Gamma$ *je část pásky za čtecí a zápisovou hlavou, čtecí a zápisová hlava ukazuje na první symbol řetězce* $w_2$*.*

Počáteční konfigurace *je* $(\varepsilon, q_0, w_0)$ *nebo* $(\$, q_0, w_0\$)$ *(pokud chceme zahrnout do konfigurace i hraniční symboly),* $w_0 \in \Sigma$ *je vstupní slovo, které má být zpracováno.*

Koncová konfigurace *je* $(w_1, q_f, w_2)$ *(resp.* $(\$w_1, q_f, w_2\$)$*),* $q_f \in F,\ w_1, w_2 \in \Gamma^*$.

**Příklad 3.3**

Například konfigurace $(abbca, q, daab)$ znamená, že se stroj nachází ve stavu $q$, na pásce je slovo $abbcadaab$ a čtecí a zápisová hlava ukazuje na šestý symbol slova – $d$.

**Definice 3.7 (Relace přechodu mezi konfiguracemi)** *Relace přechodu mezi konfiguracemi je určena takto:*

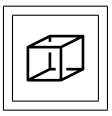
$$(\alpha, q_i, a\beta) \vdash (\alpha b, q_j, \beta) \Leftrightarrow \delta(q_i, a) = (q_j, b, 1) \tag{3.1}$$

$$(\alpha, q_i, a\beta) \vdash (\alpha, q_j, b\beta) \Leftrightarrow \delta(q_i, a) = (q_j, b, 0) \tag{3.2}$$

$$(\alpha c, q_i, a\beta) \vdash (\alpha, q_j, cb\beta) \Leftrightarrow \delta(q_i, a) = (q_j, b, -1) \tag{3.3}$$

*V prvním případě se čtecí a zápisová hlava posunuje doprava, v druhém zůstává na místě (tj. v dalším kroku bude číst totéž políčko jako v tomto) a v třetím případě se posunuje doleva.*

**Příklad 3.4**



Sestrojíme Turingův stroj rozpoznávající jazyk $L_2 = \{a^n b^n c^n \mid n \geq 0\}$

$\mathcal{M} = (\{q_0, q_P, q_A, q_B, q_C, q_f, q_{accept}\},\ \{a, b, c\},\ \{a, \bar{a}, b, \bar{b}, c, \bar{c}, \sqcup, \$\},\ \delta,\ \{q_{accept}\})$

Budeme postupovat takto: označíme první $a$ (tj. přepíšeme symbolem $\bar{a}$), najdeme první $b$, označíme ho, pak najdeme první $c$, taktéž označíme, potom přejdeme na začátek (postupujeme doleva, dokud nenajdeme nejbližší označené $\bar{a}$), posuneme se o políčko dále na první neoznačené $a$, označíme ho, atd.

Jednotlivé stavy znamenají:

- $q_A$ – označili jsme $a$, přeskakujeme symboly $a, \bar{b}$, hledáme první neoznačené $b$,
- $q_B$ – označili jsme $b$, přeskakujeme symboly $b, \bar{c}$, hledáme první neoznačené $c$,
- $q_C$ – označili jsme $c$, vracíme se na začátek k poslednímu označenému $\bar{a}$, při pohybu doleva přeskakujeme všechny symboly $\bar{c}, b, \bar{b}, a$.

Definujeme $\delta$ funkci:

$\delta(q_0, \$) = (q_{accept}, 0)$ (přijali jsme prázdné slovo)

| | | |
|---|---|---|
| $\delta(q_0, a) = (q_A, \bar{a}, 1)$ | $\delta(q_B, b) = (q_B, b, 1)$ | $\delta(q_C, \bar{a}) = (q_P, \bar{a}, 1)$ |
| $\delta(q_P, a) = (q_A, \bar{a}, 1)$ | $\delta(q_B, \bar{c}) = (q_B, \bar{c}, 1)$ | $\delta(q_0, \bar{b}) = (q_f, \bar{b}, 1)$ |
| $\delta(q_A, a) = (q_A, a, 1)$ | $\delta(q_B, c) = (q_C, \bar{c}, -1)$ | $\delta(q_f, \bar{b}) = (q_f, \bar{b}, 1)$ |
| $\delta(q_A, \bar{b}) = (q_A, \bar{b}, 1)$ | $\delta(q_C, X) = (q_C, X, -1),$ | $\delta(q_f, \bar{c}) = (q_f, \bar{c}, 1)$ |
| $\delta(q_A, b) = (q_B, \bar{b}, 1)$ | $X \in \{a, b, \bar{b}, \bar{c}\}$ | $\delta(q_f, \$) = (q_{accept}, \$, 0)$ |

Ukázka zpracování slova $abc$:

$(\varepsilon, q_0, abc) \vdash (\bar{a}, q_A, bc) \vdash (\bar{a}\bar{b}, q_B, c) \vdash (\bar{a}, q_C, \bar{b}\bar{c}) \vdash (\varepsilon, q_C, \bar{a}\bar{b}\bar{c}) \vdash (\bar{a}, q_0, \bar{b}\bar{c}) \vdash$
$\vdash (\bar{a}\bar{b}, q_f, \bar{c}) \vdash (\bar{a}\bar{b}\bar{c}, q_f, \varepsilon) \vdash (\bar{a}\bar{b}\bar{c}, q_{accept}, \varepsilon)$

Jestliže zaznamenáme i hraniční symboly, zpracování bude vypadat takto:

$(\$, q_0, abc\$) \vdash (\$\bar{a}, q_A, bc\$) \vdash (\$\bar{a}\bar{b}, q_B, c\$) \vdash (\$\bar{a}, q_C, \bar{b}\bar{c}\$) \vdash (\$, q_C, \bar{a}\bar{b}\bar{c}\$) \vdash$
$\vdash (\$\bar{a}, q_0, \bar{b}\bar{c}\$) \vdash (\$\bar{a}\bar{b}, q_f, \bar{c}\$) \vdash (\$\bar{a}\bar{b}\bar{c}, q_f, \$) \vdash (\$\bar{a}\bar{b}\bar{c}, q_{accept}, \$)$

Ukázka zpracování slova $aabbcc$:

$(\varepsilon, q_0, aabbcc) \vdash (\bar{a}, q_A, abbcc) \vdash (\bar{a}a, q_A, bbcc) \vdash (\bar{a}a\bar{b}, q_B, bcc) \vdash (\bar{a}a\bar{b}b, q_B, cc) \vdash$
$\vdash (\bar{a}a\bar{b}, q_C, b\bar{c}c) \vdash (\bar{a}a, q_C, \bar{b}\bar{c}c) \vdash (\bar{a}, q_C, a\bar{b}\bar{c}c) \vdash (\varepsilon, q_C, \bar{a}a\bar{b}\bar{c}c) \vdash (\bar{a}, q_0, a\bar{b}\bar{c}c) \vdash$
$\vdash (\bar{a}\bar{a}, q_A, \bar{b}\bar{c}c) \vdash (\bar{a}\bar{a}\bar{b}, q_A, b\bar{c}c) \vdash (\bar{a}\bar{a}\bar{b}\bar{b}, q_B, \bar{c}c) \vdash (\bar{a}\bar{a}\bar{b}\bar{b}\bar{c}, q_B, c) \vdash (\bar{a}\bar{a}\bar{b}\bar{b}, q_C, \bar{c}\bar{c}) \vdash$
$\vdash (\bar{a}\bar{a}\bar{b}, q_C, \bar{b}\bar{c}\bar{c}) \vdash (\bar{a}\bar{a}, q_C, \bar{b}\bar{b}\bar{c}\bar{c}) \vdash (\bar{a}, q_C, \bar{a}\bar{b}\bar{b}\bar{c}\bar{c}) \vdash (\bar{a}\bar{a}, q_0, \bar{b}\bar{b}\bar{c}\bar{c}) \vdash (\bar{a}\bar{a}\bar{b}, q_f, \bar{b}\bar{c}\bar{c}) \vdash$
$\vdash (\bar{a}\bar{a}\bar{b}\bar{b}, q_f, \bar{c}\bar{c}) \vdash (\bar{a}\bar{a}\bar{b}\bar{b}\bar{c}, q_f, \bar{c}) \vdash (\bar{a}\bar{a}\bar{b}\bar{b}\bar{c}\bar{c}, q_f, \varepsilon) \vdash (\bar{a}\bar{a}\bar{b}\bar{b}\bar{c}\bar{c}, q_{accept}, \varepsilon)$

Ukázka zpracování slova $ac$:

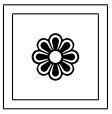
$(\varepsilon, q_0, ac) \vdash (\bar{a}, q_A, c)$ chyba $\Rightarrow$ slovo $ac$ není přijato.

Ukázka zpracování slova $\varepsilon$:

$(\varepsilon, q_0, \varepsilon) \vdash (\varepsilon, q_{accept}, \varepsilon)$

---

**Poznámka:** Všimněme si rozdílu mezi činností Turingova stroje a dříve definovaných automatů:



- Turingův stroj nejen čte vstupní pásku, může ji i zapisovat,
- čtecí a zápisová hlava se může (nemusí) pohybovat různými směry, nejen doprava,
- nepotřebujeme zásobník, ale přesto můžeme uchovávat i jinou informaci než označení stavu, ve kterém právě jsme – kamkoliv na pásku si můžeme cokoliv poznamenat a později tuto informaci využít,
- na Turingově stroji lze provádět i výpočty.

Turingovými stroji se budeme podrobněji zabývat v navazujícím kurzu Teorie vyčíslitelnosti.

### 3.2.3 Varianty Turingova stroje

Řekli jsme si, že Turingův stroj je v základní definici deterministický. Proto varianty můžeme především rozdělit podle tohoto kritéria:

- *deterministický* – základní varianta,
- *nedeterministický* – pro tentýž stav a obsah pásky lze definovat více různých akcí.

Podobně, jako zásobníkový automat může mít více zásobníků, Turingův stroj může mít více pásek:

- *jednopáskový* – základní varianta,
- *vícepáskový* – máme více pásek, každá má vlastní čtecí a zápisovou hlavu, tyto hlavy se mohou pohybovat navzájem nezávisle (například první doprava, druhá doleva a třetí třeba zůstane na místě v tomtéž kroku zpracování).

Na jedné pásce může být i více stop (podobně jako na paměťových médiích, třeba zálohovacích páskách):

- *jednostopý* – na (každé) pásce je jen jedna stopa,
- *vícestopý* – na pásce může být více stop, ale narozdíl od vícepáskového automatu zde všechny stopy téže pásky mají společnou čtecí a zápisovou hlavu.

Dále můžeme stanovit možnost pohybu čtecí a zápisové hlavy:

- *možnost pohybu* $\{-1, 0, 1\}$ – čtecí a zápisová hlava se může pohybovat doleva nebo doprava, a nebo zůstat na místě,
- *možnost pohybu* $\{-1, 1\}$ – čtecí a zápisová hlava se musí pohybovat v každém kroku, a to doleva nebo doprava, nesmí zůstat na místě.

Páska může být

- *jednostranně nekonečná* – vstupní slovo je na začátku výpočtu umístěno na začátek pásky, před ně již není možné nic napsat,
- *oboustranně nekonečná* – základní varianta.

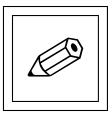
Lze dokázat, že všechny výše uvedené varianty jsou navzájem ekvivalentní, všechny varianty lze převést na základní variantu – deterministický jednopáskový jednostopý automat, jehož čtecí a zápisová hlava se může pohybovat oběma směry nebo zůstat na místě a lze zapisovat i před vstupní slovo.

## 3.3 Vztah Turingových strojů k jazykům typu 0

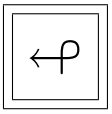
Zde ukážeme, že jazyky rozpoznávané Turingovým strojem jsou právě jazyky typu 0. Pro tuto vlastnost se také jazykům typu 0 říká *rekurzívně spočetné jazyky*, protože Turingův stroj vlastně pracuje na principu rekurze.

**Definice 3.8 (Rekurzívně spočetný jazyk)** *Jazyk nazveme rekurzívně spočetný (rekurzívně vyčíslitelný, částečně rekurzivní), pokud je přijímán nějakým Turingovým strojem (tento Turingův stroj se na slovo patřící do jazyka zastaví v akceptujícím stavu, na slovo nepatřící do jazyka se buď zastaví v odmítajícím stavu nebo se dostane do nekonečné smyčky).*

**Definice 3.9 (Rekurzívní jazyk)** *Jazyk nazveme rekurzívní, pokud je rozhodován nějakým Turingovým strojem (tento Turingův stroj se pro jakékoliv slovo zastaví, a to na slovo jazyka v akceptujícím stavu a na slovo nepatřící do jazyka v odmítajícím stavu, pro žádný vstup nepřejde do nekonečné smyčky).*



**Věta 3.2** *Ke každé gramatice $G$ typu 0 lze sestroji Turingův stroj $\mathcal{M}$ takový, že platí $L(\mathcal{M}) = L(G)$.*



**Důkaz:** Podle gramatiky $G = (N, T, P, S)$ sestrojíme nedeterministický dvoustopý Turingův stroj $\mathcal{M} = (Q, \Sigma, \Gamma, \delta, q_0, F)$. První stopa obsahuje vstupní slovo, nemění se, slouží pro kontrolu, druhá stopa simuluje derivaci v gramatice, obsahuje větnou formu, u které v derivaci právě jsme (na začátku to je $S$).

1. Nedeterministicky zvolíme některé pravidlo $\alpha \to \beta$ v gramatice.
2. Pokud se $\alpha$ nachází ve větné formě, zvolíme některý výskyt $\alpha$ a přepíšeme ho na $\beta$;
   pokud $|\beta| \neq |\alpha|$, nejdřív vhodně posuneme všechny symboly za řetězcem $\alpha$ doleva nebo doprava.
3. Porovnáme vstup na první stopě s obsahem druhé stopy – pokud je stejný, vstup přijmeme, jinak zpět k bodu 1.

Dále v důkazu nebudeme pokračovat, postup si ukážeme na příkladu. □

**Příklad 3.5**

Vytvoříme gramatiku typu 0 pro jazyk

$L_{19} = L_2 - \{\varepsilon\} = \{a^n b^n c^n \mid n \geq 1\}$

Gramatika: $G = (\{S, A, B, X\},\ \{a, b, c\},\ P,\ S)$, kde v $P$ jsou pravidla

$S \to aAbX$
$Ab \to bA$
$AX \to BXc$
$AX \to c$
$bB \to Bb$
$aB \to aaAb$

Ukázka odvození:

$S \Rightarrow aAbX \Rightarrow abAX \Rightarrow abBXc \Rightarrow aBbXc \Rightarrow aaAbbXc \Rightarrow aabAbXc \Rightarrow aabbAXc \Rightarrow$
$\Rightarrow aabbcc$

Podle této gramatiky sestrojíme Turingův stroj. Význam jednotlivých stavů:

- $q_0$... nedeterministicky vybereme pravidlo,
- $q_1, q_2, \ldots, q_6$... vybrali jsme 1., 2., ..., 6. pravidlo, teď nedeterministicky vybereme, kde ho chceme použít,
- $q_{11}$... už jsme si vybrali místo pro uplatnění prvního pravidla, zpracovali jsme první symbol pravidla (první znak $\alpha$ přepíšeme na první znak $\beta$),
- $q_{12}$... přepisujeme druhý symbol pravidla ...
  ⋮
- $q_{21}$... už jsme si vybrali místo pro uplatnění druhého pravidla, zpracovali jsme první symbol pravidla
  ⋮
- $q_Z$... přepsali jsme celé pravidlo, vracíme se na začátek větné formy, bude další pravidlo,
- $q_K$... kontrola, jestli máme skončit,
- $q_{JN}$... ještě ne (ještě neskončit, stopy mají různý obsah).

Postup si nejdřív ukážeme na průběhu výpočtu slova, které bylo v gramatice pro ukázku odvozeno. Na průběhu výpočtu vidíme, že horní stopa se nemění, zatímco na dolní probíhá simulace generování tohoto slova v gramatice.

$$\left(\begin{matrix}\$ \\ \$\end{matrix}, q_0, \begin{matrix}a & a & b & b & c & c & \$ \\ S & \$ & \sqcup & \sqcup & \sqcup & \sqcup & \sqcup\end{matrix}\right) \vdash \left(\begin{matrix}\$ \\ \$\end{matrix}, q_1, \begin{matrix}a & a & b & b & c & c & \$ \\ S & \$ & \sqcup & \sqcup & \sqcup & \sqcup & \sqcup\end{matrix}\right) \vdash$$

$$\vdash \begin{pmatrix} \$ & a \\ \$ & a \end{pmatrix}, q_{11}, \begin{matrix} a & b & b & c & c & \$ \\ \$ & \sqcup & \sqcup & \sqcup & \sqcup & \sqcup \end{matrix} \vdash \begin{pmatrix} \$ & a & a \\ \$ & a & A \end{pmatrix}, q_{12}, \begin{matrix} b & b & c & c & \$ \\ \sqcup & \sqcup & \sqcup & \sqcup & \sqcup \end{matrix} \vdash$$

$$\vdash \begin{pmatrix} \$ & a & a & b \\ \$ & a & A & b \end{pmatrix}, q_{13}, \begin{matrix} b & c & c & \$ \\ \sqcup & \sqcup & \sqcup & \sqcup \end{matrix} \vdash \begin{pmatrix} \$ & a & a & b & b \\ \$ & a & A & b & X \end{pmatrix}, q_{14}, \begin{matrix} c & c & \$ \\ \sqcup & \sqcup & \sqcup \end{matrix} \vdash$$

$$\vdash \begin{pmatrix} \$ & a & a & b \\ \$ & a & A & b \end{pmatrix}, q_Z, \begin{matrix} b & c & c & \$ \\ X & \$ & \sqcup & \sqcup \end{matrix} \vdash \begin{pmatrix} \$ & a & a \\ \$ & a & A \end{pmatrix}, q_Z, \begin{matrix} b & b & c & c & \$ \\ b & X & \$ & \sqcup & \sqcup \end{matrix} \vdash$$

$$\vdash \begin{pmatrix} \$ & a \\ \$ & a \end{pmatrix}, q_Z, \begin{matrix} a & b & b & c & c & \$ \\ A & b & X & \$ & \sqcup & \sqcup \end{matrix} \vdash \begin{pmatrix} \$ \\ \$ \end{pmatrix}, q_Z, \begin{matrix} a & a & b & b & c & c & \$ \\ a & A & b & X & \$ & \sqcup & \sqcup \end{matrix} \vdash$$

$$\vdash \begin{pmatrix} \$ \\ \$ \end{pmatrix}, q_K, \begin{matrix} a & a & b & b & c & c & \$ \\ a & A & b & X & \$ & \sqcup & \sqcup \end{matrix} \vdash \ldots \vdash \begin{pmatrix} \$ \\ \$ \end{pmatrix}, q_0, \begin{matrix} a & a & b & b & c & c & \$ \\ a & A & b & X & \$ & \sqcup & \sqcup \end{matrix} \quad \vdash \ldots$$

Definujeme $\delta$ funkci – pro každé $U \in \Sigma \cup \{\sqcup\},\ V \in \Gamma$:

Nedeterministicky vybereme pravidlo gramatiky, které budeme uplatňovat na obsah druhé stopy:

$$\delta \begin{pmatrix} q_0, & U \\ & V \end{pmatrix} = \left\{ \begin{pmatrix} q_1, & U \\ & V \end{pmatrix}, 0 \right), \begin{pmatrix} q_2, & U \\ & V \end{pmatrix}, 0 \right), \begin{pmatrix} q_3, & U \\ & V \end{pmatrix}, 0 \right), \ldots, \begin{pmatrix} q_6, & U \\ & V \end{pmatrix}, 0 \right) \right\}$$

Zpracování prvního pravidla – nejdřív nedeterministicky vybereme, na kterém místě řetězce pravidlo uplatníme, a pak pro $\alpha \to \beta$ začneme přepisovat $\alpha$ na $\beta$:

$$\delta \begin{pmatrix} q_1, & U \\ & S \end{pmatrix} = \left\{ \begin{pmatrix} q_1, & U \\ & S \end{pmatrix}, 1 \right), \begin{pmatrix} q_{11}, & U \\ & a \end{pmatrix}, 1 \right) \right\}$$

$$\delta \begin{pmatrix} q_1, & U \\ & M \end{pmatrix} = \begin{pmatrix} q_1, & U \\ & M \end{pmatrix}, 1 \Big), M \neq S$$

$$\delta \begin{pmatrix} q_{11}, & U \\ & V \end{pmatrix} = \begin{pmatrix} q_{12}, & U \\ & A \end{pmatrix}, 1 \Big) \qquad \delta \begin{pmatrix} q_{12}, & U \\ & V \end{pmatrix} = \begin{pmatrix} q_{13}, & U \\ & b \end{pmatrix}, 1 \Big)$$

$$\delta \begin{pmatrix} q_{13}, & U \\ & V \end{pmatrix} = \begin{pmatrix} q_{14}, & U \\ & X \end{pmatrix}, 1 \Big) \qquad \delta \begin{pmatrix} q_{14}, & U \\ & V \end{pmatrix} = \begin{pmatrix} q_Z, & U \\ & \$ \end{pmatrix}, -1 \Big)$$

$$\delta \begin{pmatrix} q_Z, & U \\ & V \end{pmatrix} = \begin{pmatrix} q_Z, & U \\ & V \end{pmatrix}, -1 \Big),\ V \neq \$ \text{ (jdeme na začátek větné formy)}$$

$$\delta \begin{pmatrix} q_Z, & \$ \\ & \$ \end{pmatrix} = \begin{pmatrix} q_K, & \$ \\ & \$ \end{pmatrix}, 1 \Big) \text{ (jsme na začátku)}$$

$\delta\left(q_K, \begin{matrix} U \\ U \end{matrix}\right) = \left(q_K, \begin{matrix} U \\ U \end{matrix}, 1\right)$ (kontrolujeme, jestli už jsme na konci odvození slova)

$\delta\left(q_K, \begin{matrix} \$ \\ \$ \end{matrix}\right) = \left(q_{acc}, \begin{matrix} \$ \\ \$ \end{matrix}, 0\right)$ (obě stopy mají stejný obsah $\Rightarrow$ konec odvození, konec práce)

$\delta\left(q_K, \begin{matrix} U \\ V \end{matrix}\right) = \left(q_{JN}, \begin{matrix} U \\ V \end{matrix}, -1\right),\ V \neq U$ (ještě není konec, přejdeme na začátek pásky a zvolíme další pravidlo)

$\delta\left(q_{JN}, \begin{matrix} U \\ V \end{matrix}\right) = \left(q_{JN}, \begin{matrix} U \\ V \end{matrix}, -1\right) \qquad \delta\left(q_{JN}, \begin{matrix} \$ \\ \$ \end{matrix}\right) = \left(q_0, \begin{matrix} \$ \\ \$ \end{matrix}, 1\right)$

$\delta\left(q_2, \begin{matrix} U \\ A \end{matrix}\right) = \left\{\left(q_2, \begin{matrix} U \\ A \end{matrix}, 1\right), \left(q_{21}, \begin{matrix} U \\ b \end{matrix}, 1\right)\right\}$ (zpracováváme druhé pravidlo)

$\delta\left(q_2, \begin{matrix} U \\ M \end{matrix}\right) = \left(q_2, \begin{matrix} U \\ M \end{matrix}, 1\right), M \neq A$

$\delta\left(q_{21}, \begin{matrix} U \\ b \end{matrix}\right) = \left(q_Z, \begin{matrix} U \\ A \end{matrix}, -1\right)$

Třetí pravidlo je typu $|\alpha| < |\beta|$, vše za $\alpha$ posuneme doprava, aby se $\beta$ vešla:

$\delta\left(q_3, \begin{matrix} U \\ A \end{matrix}\right) = \left\{\left(q_3, \begin{matrix} U \\ A \end{matrix}, 1\right), \left(q_P, \begin{matrix} U \\ 3 \end{matrix}, 1\right)\right\}$ (symbol 3 je zarážka, abychom po posouvání věděli, kde máme začít psát řetězec $\beta$)

$\delta\left(q_3, \begin{matrix} U \\ M \end{matrix}\right) = \left(q_3, \begin{matrix} U \\ M \end{matrix}, 1\right), M \neq A$

Funkce pro posun následujícího řetězce o 1 políčko doprava:

$\delta\left(q_P, \begin{matrix} U \\ V \end{matrix}\right) = \left(q_P, \begin{matrix} U \\ V \end{matrix}, 1\right),\ V \neq \$$ (nejdřív se přesuneme na konec řetězce)

$\delta\left(q_P, \begin{matrix} U \\ \$ \end{matrix}\right) = \left(q_{P\$}, \begin{matrix} U \\ \$ \end{matrix}, 1\right)$

$\delta\left(q_{PM}, \begin{matrix} U \\ V \end{matrix}\right) = \left(q_{PZ}, \begin{matrix} U \\ M \end{matrix}, -1\right),\ M \in \{a, b, c, S, A, B, X, \$\}$

$\delta\begin{pmatrix} q_{PZ}, & \begin{matrix}U\\M\end{matrix}\end{pmatrix} = \begin{pmatrix} q_{PM}, & \begin{matrix}U\\M\end{matrix}, 1\end{pmatrix}$ (zapamatujeme si $M$, předchozí částí $\delta$ fce ho pak zkopírujeme vpravo)

$\delta\begin{pmatrix} q_{PZ}, & \begin{matrix}U\\3\end{matrix}\end{pmatrix} = \begin{pmatrix} q_{31}, & \begin{matrix}U\\B\end{matrix}, 1\end{pmatrix}$ (zarážka 3 znamená, že jsme při uplatňování 3. pravidla gramatiky posunuli vše za tímto místem o 1 políčko doprava, teď můžeme zapsat pravou stranu pravidla, už se vejde)

$\delta\begin{pmatrix} q_{31}, & \begin{matrix}U\\V\end{matrix}\end{pmatrix} = \begin{pmatrix} q_{32}, & \begin{matrix}U\\X\end{matrix}, 1\end{pmatrix}$ $\quad$ $\delta\begin{pmatrix} q_{32}, & \begin{matrix}U\\V\end{matrix}\end{pmatrix} = \begin{pmatrix} q_{Z}, & \begin{matrix}U\\c\end{matrix}, -1\end{pmatrix}$

$\delta\begin{pmatrix} q_{PZ}, & \begin{matrix}U\\6\end{matrix}\end{pmatrix} = \begin{pmatrix} q_{P}, & \begin{matrix}U\\6'\end{matrix}, 1\end{pmatrix}$ (6. pravidlo gramatiky vyžaduje posun o 2 políčka, tedy musíme funkci pro posun volat 2x)

$\delta\begin{pmatrix} q_{PZ}, & \begin{matrix}U\\6'\end{matrix}\end{pmatrix} = \begin{pmatrix} q_{61}, & \begin{matrix}U\\a\end{matrix}, 1\end{pmatrix}$ $\quad$ $\delta\begin{pmatrix} q_{61}, & \begin{matrix}U\\B\end{matrix}\end{pmatrix} = \begin{pmatrix} q_{62}, & \begin{matrix}U\\a\end{matrix}, 1\end{pmatrix}$

$\delta\begin{pmatrix} q_{62}, & \begin{matrix}U\\V\end{matrix}\end{pmatrix} = \begin{pmatrix} q_{63}, & \begin{matrix}U\\A\end{matrix}, 1\end{pmatrix}$ $\quad$ $\delta\begin{pmatrix} q_{63}, & \begin{matrix}U\\V\end{matrix}\end{pmatrix} = \begin{pmatrix} q_{Z}, & \begin{matrix}U\\b\end{matrix}, -1\end{pmatrix}$

Čtvrté pravidlo gramatiky vyžaduje posun následujících symbolů doleva ($\beta$ je kratší než $\alpha$):

$\delta\begin{pmatrix} q_{4}, & \begin{matrix}U\\A\end{matrix}\end{pmatrix} = \begin{pmatrix} q_{41}, & \begin{matrix}U\\4\end{matrix}, 1\end{pmatrix}$

$\delta\begin{pmatrix} q_{41}, & \begin{matrix}U\\X\end{matrix}\end{pmatrix} = \begin{pmatrix} q_{R}, & \begin{matrix}U\\X\end{matrix}, 1\end{pmatrix}$ (kontrolujeme, zda je ve slově přítomna celá $\alpha$)

Funkce pro posun následujícího řetězce o 1 políčko doleva:

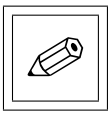
$\delta\begin{pmatrix} q_{R}, & \begin{matrix}U\\V\end{matrix}\end{pmatrix} = \begin{pmatrix} q_{RV}, & \begin{matrix}U\\V\end{matrix}, 1\end{pmatrix}$ $\quad$ $\delta\begin{pmatrix} q_{RV}, & \begin{matrix}U\\M\end{matrix}\end{pmatrix} = \begin{pmatrix} q_{RD}, & \begin{matrix}U\\V\end{matrix}, 1\end{pmatrix}$

$\delta\begin{pmatrix} q_{RD}, & \begin{matrix}U\\V\end{matrix}\end{pmatrix} = \begin{pmatrix} q_{R}, & \begin{matrix}U\\V\end{matrix}, 1\end{pmatrix}$ $\quad$ $\delta\begin{pmatrix} q_{RD}, & \begin{matrix}U\\\$\end{matrix}\end{pmatrix} = \begin{pmatrix} q_{RZ}, & \begin{matrix}U\\\sqcup\end{matrix}, -1\end{pmatrix}$

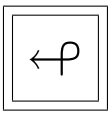
$\delta\begin{pmatrix} q_{RZ}, & \begin{matrix}U\\V\end{matrix}\end{pmatrix} = \begin{pmatrix} q_{RZ}, & \begin{matrix}U\\V\end{matrix}, -1\end{pmatrix}$ $\quad$ $\delta\begin{pmatrix} q_{RZ}, & \begin{matrix}U\\4\end{matrix}\end{pmatrix} = \begin{pmatrix} q_{Z}, & \begin{matrix}U\\c\end{matrix}, -1\end{pmatrix}$

Pokud by $\beta$ byla delší než 1 znak, museli bychom pro zpracování celého pravidla použít stavy $q_{42}, q_{43}, \ldots$

atd. pro další pravidla gramatiky.



**Věta 3.3** *Ke každému Turingovu stroji $\mathcal{M}$ lze sestrojit gramatiku $G$ typu 0 takovou, že $L(\mathcal{M}) = L(G)$.*



**Důkaz:** Budeme postupovat takto:

- vygenerujeme dvě kopie téhož slova,
- první (horní) na konci generování použijeme jako výstup gramatiky, na druhé (spodní) budeme simulovat činnost TS,
- výstup gramatiky bude složen pouze z terminálních symbolů jen tehdy, pokud simulace na druhé kopii skončí ve stavu $q_{accept}$.

Neterminály máme několika typů:

- neterminály ve tvaru sloupcového vektoru, jehož horní prvek $\in T$ nebo je $\varepsilon$,
- neterminály pro stav (zač. $q_0$) a začátek a konec řetězce \$,
- další neterminály bez přímého vztahu k TS ($S$, $S'$).

Pro lepší představu se podíváme na fragment ukázky odvození:

$$S \Rightarrow^* q_0 \begin{bmatrix} \varepsilon \\ \$ \end{bmatrix} \begin{bmatrix} a \\ a \end{bmatrix} \begin{bmatrix} a \\ a \end{bmatrix} \begin{bmatrix} b \\ b \end{bmatrix} \begin{bmatrix} b \\ b \end{bmatrix} \begin{bmatrix} c \\ c \end{bmatrix} \begin{bmatrix} c \\ c \end{bmatrix} \$ \Rightarrow^* aabbcc$$

Postupně vytvoříme pravidla gramatiky.

1. Vygenerujeme dvě kopie téhož slova:

$$S \rightarrow q_0 \begin{bmatrix} \varepsilon \\ \$ \end{bmatrix} S'$$
$$S' \rightarrow \begin{bmatrix} a \\ a \end{bmatrix} S' \quad \text{pro každé } a \in \Sigma$$
$$S' \rightarrow \$$$

2. Simulujeme činnost Turingova stroje:

   (a) pro $\delta(q_i, a) = (q_j, b, 1)$, $a, b \in \Gamma,\ q_i, q_j \in Q$ vytvoříme pravidlo
   $$q_i \begin{bmatrix} x \\ a \end{bmatrix} \rightarrow \begin{bmatrix} x \\ b \end{bmatrix} q_j, \text{ pro } x \in \Sigma \cup \{\varepsilon\}$$

(b) pro $\delta(q_i, a) = (q_j, b, 0)$, $a, b \in \Gamma,\ q_i, q_j \in Q$ vytvoříme pravidlo

$$q_i \begin{bmatrix} x \\ a \end{bmatrix} \rightarrow q_j \begin{bmatrix} x \\ b \end{bmatrix}, \text{ pro } x \in \Sigma \cup \{\varepsilon\}$$

(c) pro $\delta(q_i, a) = (q_j, b, -1)$, $a, b \in \Gamma,\ q_i, q_j \in Q$ vytvoříme pravidlo

$$\begin{bmatrix} y \\ c \end{bmatrix} q_i \begin{bmatrix} x \\ a \end{bmatrix} \rightarrow q_j \begin{bmatrix} y \\ c \end{bmatrix} \begin{bmatrix} x \\ b \end{bmatrix}, \text{ pro } x, y \in \Sigma \cup \{\varepsilon\}, c \in \Gamma$$

(d) pro $\delta(q_i, \sqcup) = (q_j, b, 1)$, příp. $\delta(q_i, \$) = (q_j, b, 1)$, $b \in \Gamma,\ q_i, q_j \in Q$ vytvoříme

$$q_i\$ \rightarrow \begin{bmatrix} \varepsilon \\ b \end{bmatrix} q_j\$$$

(e) pro $\delta(q_i, \sqcup) = (q_j, b, 0)$, příp. $\delta(q_i, \$) = (q_j, b, 0)$, $b \in \Gamma,\ q_i, q_j \in Q$ vytvoříme

$$q_i\$ \rightarrow q_j \begin{bmatrix} \varepsilon \\ b \end{bmatrix} \$$$

(f) pro $\delta(q_i, \sqcup) = (q_j, b, -1)$, příp. $\delta(q_i, \$) = (q_j, b, -1)$, $b \in \Gamma,\ q_i, q_j \in Q$ vytvoříme pravidlo

$$\begin{bmatrix} y \\ c \end{bmatrix} q_i\$ \rightarrow q_j \begin{bmatrix} y \\ c \end{bmatrix} \begin{bmatrix} \varepsilon \\ b \end{bmatrix} \$, \text{ pro } y \in \Sigma \cup \{\varepsilon\}, c \in \Gamma$$

3. Zakončíme derivaci (vytvoří se terminální slovo):

(a) Posuneme $q_{accept}$ na začátek větné formy:

$M q_{accept} \rightarrow q_{accept} M$ pro všechny neterminály $M \in N$

(b) Tvoříme terminály:

$$q_{accept} \begin{bmatrix} a \\ x \end{bmatrix} \rightarrow a\, q_{accept}$$

(c) Mažeme vše, co nevytvoří terminál:

$$q_{accept} \begin{bmatrix} \varepsilon \\ x \end{bmatrix} \rightarrow q_{accept}$$

(d) Konec derivace: $q_{accept}\$ \rightarrow \varepsilon$

□

Kapitola 4

# Jazyky typu 1

*V této kapitole se budeme zabývat třídou jazyků generovaných gramatikami typu 1 Chomského hierarchie. Nejdřív se podíváme na tvar gramatik, které mohou generovat tyto jazyky a vztah mezi nimi, dále se budeme zabývat modely – stroji rozpoznávajícími tyto gramatiky a také se budeme věnovat uzávěrovým vlastnostem jazyků typu 1 Chomského hierarchie.*

## 4.1 Gramatiky typu 1

V kurzu Teorie jazyků a automatů I jsme si vysvětlili, že třebaže gramatiky typu 1 jsou nezkracující, existuje jiný typ gramatik generujících tutéž třídu jazyků – gramatiky kontextové.

**Definice 4.1 (Nezkracující gramatika)** *Nezkracující gramatika je taková gramatika, jejíž pravidla jsou ve tvaru*

$$\alpha \to \beta, \text{ kde } |\alpha| \leq |\beta|, \ \alpha \in (N \cup T)^* N (N \cup T)^*, \ \beta \in (N \cup T)*$$

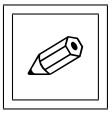
*Je přípustné také pravidlo $S \to \varepsilon$, pokud $\varepsilon$ je slovo jazyka, který gramatika generuje, $S$ se nesmí vyskytovat na pravé straně žádného pravidla.*

**Definice 4.2 (Kontextová gramatika)** *Kontextová gramatika je gramatika s pravidly ve tvaru*

$$\alpha A \beta \to \alpha \gamma \beta, \text{ kde } |\gamma| > 0, \ \alpha, \beta, \gamma \in (N \cup T)^*$$

*Je přípustné také pravidlo $S \to \varepsilon$, pokud $\varepsilon$ je slovo jazyka, který gramatika generuje, $S$ se nesmí vyskytovat na pravé straně žádného pravidla.*

Je zřejmé, že každá kontextová gramatika je zároveň nezkracující, vyplývá to přímo z tvaru pravidel. Platí však i opačná relace?



**Věta 4.1** *Ke každé nezkracující gramatice $G$ lze sestrojit kontextovou gramatiku $G'$ takovou, že $L(G') = L(G)$.*

**Důkaz:** Každé pravidlo typu

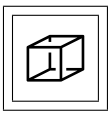
$$A_1 A_2 \dots A_m \to B_1 B_2 \dots B_n,$$

které nemá kontextový tvar, nahradíme množinou pravidel:

$\boxed{A_1}\ A_2\ A_3 \dots A_m \to \boxed{C_1}\ A_2\ A_3 \dots A_m$
$C_1\ \boxed{A_2}\ A_3 \dots A_m \to C_1\ \boxed{C_2}\ A_3 \dots A_m$
$C_1\ C_2\ \boxed{A_3} \dots A_m \to C_1\ C_2\ \boxed{C_3} \dots A_m$
$\vdots$
$C_1\ C_2 \dots C_{m-1}\ \boxed{A_m} \to C_1\ C_2 \dots C_{m-1}\ \boxed{C_m\ B_{m+1} \dots B_n}$
$C_1\ C_2 \dots C_{m-1} \boxed{C_m} B_{m+1} \dots B_n \to C_1 \dots C_{m-1} \boxed{B_m} \dots B_n$
$C_1\ C_2 \dots \boxed{C_{m-1}} B_m\ B_{m+1} \dots B_n \to C_1 \dots \boxed{B_{m-1}} B_m \dots B_n$
$\vdots$
$\boxed{C_1}\ B_2 \dots B_n \to \boxed{B_1} \dots B_n$

Kde $C_i$ jsou nově přidané neterminály, které se jinde nevyskytují. □

**Příklad 4.1**



Vytvoříme nezkracující gramatiku pro jazyk $L_5 = \{ww \mid w \in \{a,b\}^*\}$

$S \to XZ_aZ_a \mid XZ_bZ_b \mid aa \mid bb \mid \varepsilon$
pokud $w$ končí na $a$, začínáme prvním pravidlem – $\dots Z_a \dots Z_a$
pokud $w$ končí na $b$, začínáme druhým pravidlem – $\dots Z_b \dots Z_b$

$XZ_a \to aZ_aX_a \mid bZ_aX_b \mid aaX_a \mid baX_b$
v první polovině jsme vygenerovali $a$, pošleme info do druhé – $\dots aZ_aX_a \dots Z_a$
v první polovině jsme vygenerovali $b$, pošleme info do druhé – $\dots bZ_aX_b \dots Z_a$

$X_aa \to aX_a$ symbol $X_a$ nebo $X_b$ posíláme doprava
$X_ab \to bX_a$
$X_ba \to aX_b$
$X_bb \to bX_b$

$X_aZ_a \to YaZ_a \mid aa$ info doputovalo k zarážce na konci slova – $\ldots aZ_a \ldots YaZ_a$
$X_bZ_a \to YbZ_a \mid ba$ $\ldots bZ_a \ldots YbZ_a$

$aY \to Ya$ symbol $Y$ posíláme doleva – zpráva „pokračuj v první polovině“
$bY \to Yb$

$Z_aY \to XZ_a$ překročili jsme zarážku na konci první poloviny slova

Dále totéž pro $Z_b$ místo $Z_a$:
$XZ_b \to aZ_bX_a \mid bZ_bX_b \mid abX_a \mid bbX_b$
$X_aZ_b \to YaZ_b \mid ab$
$X_bZ_b \to YbZ_b \mid bb$
$Z_bY \to XZ_b$

Ukázka odvození:

$S \Rightarrow XZ_aZ_a \Rightarrow aZ_aX_aZ_a \Rightarrow aZ_aYaZ_a \Rightarrow aXZ_aaZ_a \Rightarrow abZ_aX_baZ_a \Rightarrow$
$\Rightarrow abZ_aaX_bZ_a \Rightarrow abZ_aaYbZ_a \Rightarrow abZ_aYabZ_a \Rightarrow abXZ_aabZ_a \Rightarrow abbX_babZ_a \Rightarrow$
$\Rightarrow abbaX_bbZ_a \Rightarrow abbabX_bZ_a \Rightarrow abbabba$

---

## 4.2 Kurodova normální forma pro gramatiky typu 1

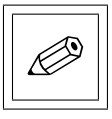
(Kuroda normal form, KNF)

**Definice 4.3 (Kurodova normální forma pro gramatiky typu 1)** *Gramatika je v KNF, jestliže všechna její pravidla jsou ve tvaru*

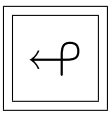
$$A \to BC$$
$$AB \to CD$$
$$A \to a$$

$A, B, C, D \in N$, $a \in T$, *případně* $S \to \varepsilon$

**Věta 4.2** *Ke každé nezkracující (i kontextové) gramatice $G$ lze sestrojit gramatiku $G'$ v Kurodově normální formě takovou, že $L(G') = L(G)$.*



**Důkaz:** Budeme postupovat takto:



- *pravidla v bezkontextovém tvaru:* použijeme algoritmus pro převod na Chomského NF.
- *pravidla, která nejsou bezkontextového typu:* upravíme pravidla podle následujícího algoritmu.

*Vstup:* nezkracující gramatika bez jednoduchých pravidel typu $X \to Y, \quad X, Y \in N$

1. Pro všechna pravidla:
   - všechny terminály (na levé i pravé straně pravidla) $a \in T$ nahradíme „pomocnými" neterminály $N_a$,
   - pro všechny neterminály vytvořené v předchozím bodu přidáme pravidlo $N_a \to a$.
2. Pravidla $A \to B_1 B_2 \dots B_n,\ n > 2$ nahradíme pravidly
$$\begin{aligned} A &\to B_1 X_1 \\ X_1 &\to B_2 X_2 \\ &\vdots \\ X_{n-2} &\to B_{n-1} B_n \end{aligned}$$
3. Pravidla $A_1 A_2 \dots A_m \to B_1 B_2 \dots B_m,\ m > 2$ nahradíme pravidly
$$\begin{aligned} A_1 A_2 &\to B_1 X_1 \\ X_1 A_3 &\to B_2 X_2 \\ &\vdots \\ X_{m-2} A_m &\to B_{m-1} B_m \end{aligned}$$
4. Nezkracující pravidla $A_1 A_2 \dots A_m \to B_1 B_2 \dots B_n,\ 2 < m \leq n$ nahradíme pravidly
$$\begin{aligned} A_1 A_2 &\to B_1 X_1 \\ X_1 A_3 &\to B_2 X_2 \\ &\vdots \\ X_{m-2} A_m &\to B_{m-1} X_{m-1} \end{aligned} \qquad \begin{aligned} X_{m-1} &\to B_m X_m \\ X_m &\to B_{m+1} X_{m+1} \\ &\vdots \\ X_{n-2} &\to B_{n-1} B_n \end{aligned}$$

□

## 4.3 Lineárně ohraničený automat

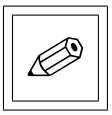
Tak jako jazykům typu 0 můžeme přiřadit Turingův stroj a například konečný automat rozpoznává právě regulární jazyky, k jazykům typu 1 můžeme přiřadit lineárně ohraničený automat (LOA, Lineary Bounded Automaton).

Definici LOA přejímáme z definice TS, jen přidáváme zákaz přepisu hraničních symbolů.

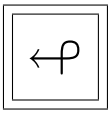
**Definice 4.4 (Lineárně ohraničený automat)** *Lineárně ohraničený automat je jednopáskový (nedeterministický) Turingův stroj $\mathcal{M} = (Q, \Sigma, \Gamma, \delta, q_0, F)$, ve kterém čtecí a zápisová hlava nesmí během výpočtu přepsat hraniční symboly \$ pásky (tj. slovo nesmí být během výpočtu prodlužováno).*

**Definice 4.5 (Konfigurace lineárně ohraničeného automatu)** *Konfigurace lineárně ohraničeného automatu je $(\alpha, q, \beta)$, kde $q$ je stav, na pásce je řetězec $\alpha\beta$, čtecí a zápisová hlava ukazuje na první symbol řetězce $\beta$.*

Přechod mezi konfiguracemi je definován stejně jako u Turingova stroje, jen je třeba zohlednit nemožnost přepisu hraničních symbolů.



**Věta 4.3** *Pro konkrétní lineárně ohraničený automat $\mathcal{M}$ a daný vstup $w_0$ existuje konečný počet různých konfigurací.*



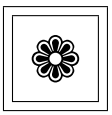
**Důkaz:** Protože máme konečný počet stavů, konečný počet páskových symbolů a omezenou část vstupní pásky, platí:
jestliže označíme

$d$ ... délka používané části pásky,

$s$ ... počet stavů v $Q$,

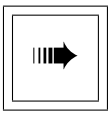
$g$ ... počet prvků páskové abecedy $\Gamma$,

pak počet všech možných různých konfigurací je $d \cdot s \cdot g^d$ (hlava může být na $d$ různých pozicích, nabývá hodnot $s$ různých stavů, počet všech možných řetězců nad abecedou $\Gamma$ o délce $d$ je $g^d$). □

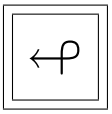


**Poznámka:** Rekurzívní jazyky jsme definovali v definici 3.9 na straně 49. Narozdíl od rekurzívně spočetných jazyků je lze zpracovat Turingovým strojem tak, že pro

jakýkoliv vstup výpočet skončí přechodem do některého koncového stavu, bez přechodu do nekonečné smyčky, a výpočet probíhá na principu rekurze (rekurzívně uplatňujeme $\delta$ funkci).



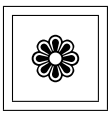
**Důsledek 4.4** *LOA lze vždy navrhnout tak, aby výpočet skončil nad jakýmkoliv vstupem. Proto jazyky, které jsou přijímány nějakým LOA, jsou právě rekurzívní jazyky.*



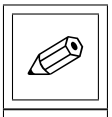
**Důkaz:** Stačí při každém kroku výpočtu LOA zkontrolovat, jestli se nenachází v konfiguraci, ve které už byl dříve. Pokud ano, výpočet v původním automatu se dostal do smyčky a my můžeme skončit v chybovém stavu $q_{reject}$ (vstup nepřijmeme).

Pokud pro každý vstup LOA najdeme počet políček pásky, se kterými během výpočtu bude pracovat čtecí a zápisová hlava (tj. délka části pásky použité při výpočtu, prostorová složitost výpočtu automatu nad daným vstupem), zjistíme, že tato hodnota je lineárně závislá na délce vstupu, tedy existuje přirozené číslo $k$ takové, že délka použité části pásky je menší než $k \cdot |w|$.

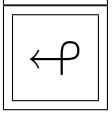
Odtud je odvozen také název tohoto automatu – automat s lineárně ohraničeným pracovním prostorem. □



**Poznámka:** V některých zdrojích je LOA definován přímo jako Turingův stroj s lineárně ohraničeným pracovním prostorem, a je tedy dovoleno používat pro výpočet každého slova $w$ nejvýše $k \cdot |w|$ políček pásky (tedy nejen pro $k = 1$).



**Věta 4.5** *Jazyky typu 1 jsou právě jazyky přijímané lineárně ohraničenými automaty.*



**Důkaz:** („⇒“)
Máme nezkracující gramatiku G, která generuje zadaný jazyk typu 1 $L$. Derivace v této gramatice je ve tvaru
$S \Rightarrow w_1 \Rightarrow w_2 \Rightarrow \ldots w_n$, kde $|w_i| \leq |w_j|$ pro $i < j$.

Sestrojíme LOA $\mathcal{M}$ takto:

1. Na vstupu je slovo, o kterém chceme zjistit, zda je generováno gramatikou G.
2. Na tento vstup budeme uplatňovat pravidla gramatiky takto:
   (a) nedeterministicky zvolíme pravidlo gramatiky ($\alpha \rightarrow \beta$),

(b) najdeme v řetězci na pásce některý výskyt pravé strany tohoto pravidla ($\beta$) – pokud je těchto výskytů více, nedeterministicky mezi nimi jeden zvolíme,

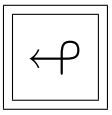
(c) výskyt $\beta$ přepíšeme řetězcem $\alpha$, pokud je $\alpha$ kratší, posuneme to, co následuje za $\beta$, doleva, aby zbytek pracovního slova navazoval na $\alpha$.

3. Výpočet ukončíme:

   - ve stavu akceptování ($q_{accept}$), pokud na pásce bude pouze startovací symbol gramatiky a nic jiného,
   - ve stavu odmítnutí slova ($q_{reject},\ q_{error}$), pokud je na pásce něco jiného a již nelze použít žádné pravidlo gramatiky.

Je zřejmé, že takto vytvořený LOA vytváří derivaci slova podle gramatiky G zprava doleva (a tedy konstruuje derivační strom pro toto slovo zdola nahoru ke kořeni stromu ohodnocenému startovacím symbolem) a ve stavu akceptování skončí právě na ta slova, která generuje gramatika G. □

**Důkaz:** („⇐“)
Je dán LOA $\mathcal{M}$. Vytvoříme nezkracující gramatiku $G$ podobně jako v obdobném důkazu pro Turingovy stroje a gramatiky typu 0, jen musíme pravidla upravit tak, aby byla nezkracující.



1. V gramatice G nejdřív vygenerujeme řetězec neterminálů, který představuje dvojici slov na vstupu simulovaného LOA. Oproti gramatice typu 0 musíme symboly pro začátek a konec pracovní části pásky a také symbol pro stav automatu umístit dovnitř neterminálů pro symboly slova, abychom nebyli nuceni na konci výpočtu použít epsilonová pravidla.

$$S \to \left[\begin{array}{c} a \\ \$, q_0, a \end{array}\right] S' \;\middle|\; \left[\begin{array}{c} \varepsilon \\ \$, q_0, \$ \end{array}\right]$$

$$S' \to \left[\begin{array}{c} a \\ a \end{array}\right] S' \;\middle|\; \left[\begin{array}{c} a \\ a, \$ \end{array}\right] \text{ pro každé } a \in \Sigma$$

Dostaneme řetězec neterminálů

$$\left[\begin{array}{c} a_1 \\ \$, q_0, a_1 \end{array}\right] \left[\begin{array}{c} a_2 \\ a_2 \end{array}\right] \left[\begin{array}{c} a_3 \\ a_3 \end{array}\right] \cdots \left[\begin{array}{c} a_k \\ a_k, \$ \end{array}\right]$$

2. Simulujeme průběh výpočtu LOA:

   Např. pro každou část $\delta$ funkce typu $\delta(q_i, a) \ni (q_j, b, 1)$

   $$\begin{bmatrix} x \\ q_i, a \end{bmatrix} \begin{bmatrix} y \\ c \end{bmatrix} \rightarrow \begin{bmatrix} x \\ b \end{bmatrix} \begin{bmatrix} y \\ q_j, c \end{bmatrix} \text{ pro každé } c \in \Gamma - \{\$\}$$

   Podobně pro ostatní směry pohybu čtecí a zápisové hlavy, navíc musíme ošetřit práci s neterminály, které obsahují hraniční znaky \$.

3. Ukončení výpočtu:

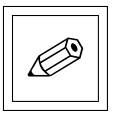
   $$\begin{bmatrix} x \\ a \end{bmatrix} \begin{bmatrix} y \\ q_{acc}, b \end{bmatrix} \rightarrow \begin{bmatrix} x \\ q_{acc}, a \end{bmatrix} \begin{bmatrix} y \\ b \end{bmatrix}$$
   $$\begin{bmatrix} x \\ \$, a \end{bmatrix} \begin{bmatrix} y \\ q_{acc}, b \end{bmatrix} \rightarrow x \begin{bmatrix} y \\ K, b \end{bmatrix}$$
   $$\begin{bmatrix} x \\ K, a \end{bmatrix} \begin{bmatrix} y \\ b \end{bmatrix} \rightarrow x \begin{bmatrix} y \\ K, b \end{bmatrix}$$
   $$\begin{bmatrix} x \\ K, a \end{bmatrix} \begin{bmatrix} y \\ b, \$ \end{bmatrix} \rightarrow xy$$

   Posuneme $q_{acc}$ dopředu na začátek řetězce, pak přejdeme do ukončujícího stavu $K$ a postupně všechny neterminály přepíšeme na terminály.

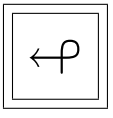
□

## 4.4 Uzávěrové vlastnosti jazyků typu 1

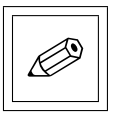
**Věta 4.6** *Třída jazyků typu 1 je uzavřena vzhledem k operacím sjednocení, zřetězení, iterace, pozitivní iterace, homomorfismu, substituce.*



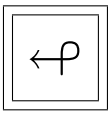
**Důkaz:** Stejně jako u bezkontextových jazyků, pomocí gramatik. Například pro sjednocení přidáme navíc pravidlo $S \rightarrow S_1 \mid S_2$, kde $S$ je nově přidaný symbol, $S_1$ je startovací symbol první gramatiky a $S_2$ je startovací symbol druhé gramatiky. □



**Věta 4.7** *Třída jazyků typu 1 je uzavřena vzhledem k operaci průniku.*



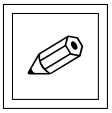
**Důkaz:** Máme dva LOA $\mathcal{M}_1$, $\mathcal{M}_2$. Sestrojíme LOA $\mathcal{M}$, který bude postupně simulovat výpočet obou těchto automatů, a pokud oba skončí s akceptováním vstupního slova, toto slovo také akceptuje.



1. Na vstupní pásce pro akceptování slova $w$ bude řetězec $\$w\#w\$$.
2. Nejdřív $\mathcal{M}$ simuluje na první kopii slova $w$ výpočet stroje $\mathcal{M}_1$ s tím, že hraniční symboly jsou $\$$ a $\#$.
3. Pokud simulovaný výpočet skončí ve stavu akceptování slova, posune čtecí a zápisovou hlavu na první symbol za znakem $\#$ (tj. na první symbol druhé kopie slova $w$) a simuluje výpočet stroje $\mathcal{M}_2$.
4. Pokud tento výpočet skončí ve stavu akceptování, M akceptuje slovo $w$.

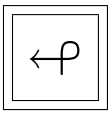
□

**Věta 4.8** *Třída jazyků typu 1 je uzavřena vzhledem k operaci doplňku.*



Důkaz lze provést pomocí DeMorganových pravidel nebo konstrukčně. Důkaz pomocí DeMorganových pravidel můžeme nechat na čtenáři, konstrukční důkaz je jednoduchý:

**Důkaz:** LOA lze vždy sestrojit tak, aby jeho výpočet byl konečný. Proto aby přijímal doplněk původního jazyka, pouze zaměníme akceptující a chybový stav. □



Kapitola 5

# L-systémy

*V této kapitole si představíme biologicky motivovaný typ systému, který nepatří do Chomského hierarchie, L-systémy. Seznámíme se zde s použitím paralelismu v tomto formálním systému, s jeho základními typy a také s možnostmi praktického využití.*

## 5.1 Paralelní odvozování

Gramatiky Chomského hierarchie pracují vždy sekvenčně. Obecně však můžeme používat i *paralelní mód odvození*, kdy je za určitých podmínek dovoleno zpracovávání více míst v přepisovaném slově.

Mezi prvními, kteří ve své práci používali paralelní mód odvození, byl maďarský biolog *Aristid Lindenmayer* (1925–1989). V roce 1968 publikoval článek, ve kterém představil možnost využití matematických modelů pro simulaci některých vývojových procesů v přírodě. Jeho model byl nazván Lindenmayerovy systémy (L-systémy) a jeho vlastnosti závisejí především na určení pro simulace dějů v přírodě – v přírodě mnoho dějů probíhá paralelně (dělení buněk, spolupráce mravenců, růst trávy na louce, apod.) a součástí přírody jsou i „meziprodukty“, nejen konečný výsledek odvození.

Účelem bylo shrnout do jednoho celku strukturu organismu a zároveň pravidla pro jeho vývoj tak, aby pravidla mohla působit obdobně jako je běžné v přírodě, paralelně (zároveň – buňky při svém dělení jedna na druhou nečekají). Jeho žáci o několik let později vyvinuli způsob vizualizace L-systémů především pro simulaci růstu rostlin.

Matematici později začali L-systémy používat pro studium soběpodobnosti (malé části objektu jsou podobné objektu celému, např. větvička na stromě vypadá jako zmenšený strom samotný).

L-systémy mají tyto vlastnosti:

- používá se pouze jediná *abeceda systému*, nerozlišujeme terminální a neterminální abecedu[1],
- vývoj probíhá v *prostředí*, v prostředí je přepisované slovo – posloupnost symbolů abecedy, kterou také nazýváme *stav prostředí*,
- pro každý symbol abecedy systému musí existovat alespoň jedno pravidlo (obvykle bezkontextové), pravidlům se říká *vývojová pravidla*,
- vývojová pravidla pracují paralelně,
- v každém kroku odvození jsou přepsány vždy všechny symboly, které se právě v prostředí nacházejí, a to kterýmkoliv (náhodně vybraným) pravidlem pro tento symbol,
- *axiom*, tedy počáteční slovo odvozování, může být oproti startovacímu symbolu Chomského gramatik jakkoliv dlouhé slovo,
- odvození je teoreticky nekonečné, ale můžeme si stanovit maximální délku odvozování.

Pokud do definice systému nezahrneme axiom, ale pouze abecedu a množinu vývojových pravidel, hovoříme o *L-schématu*, po zahrnutí axiomu do definice jde o *L-systém*.

## 5.2 0L-systémy

L-systémy mají hodně variant. Základní výše popsaná varianta s jedinou abecedou a bezkontextovými vývojovými pravidly se nazývá *0L-systémy*. *0L-schéma* pak má tvar $G = (V, P)$, kde $V$ je abeceda systému a $P$ je množina vývojových pravidel, 0L-systém má tvar $G = (V, P, \omega_0)$ – navíc je axiom $\omega_0 \in V^*$. Do *jazyka generovaného 0L-systémem* jsou řazeny všechny stavy prostředí, tedy vlastně všechny členy kterékoliv derivace z axiomu.

[1] Existují však odvozené modely, ve kterých je terminální abeceda odlišena.

**Definice 5.1 (0L-schéma)** *0L-schéma je definováno jako uspořádaná dvojice $\mathcal{S}_0 = (V, P)$, kde*

- $\Sigma$ *je neprázdná konečná abeceda systému,*
- $P$ *je množina pravidel bezkontextového typu $P \subseteq V \times V^*$ (např. $a \to bac$).*

**Definice 5.2 (0L-systém)** *0L-systém je definován jako $G_0 = (V, P, \omega_0)$, kde*

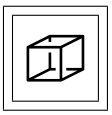
- $(V, P)$ *je 0L-schéma,*
- $\omega_0$ *je axiom (startovací slovo).*

Krok odvození zde nebudeme formálně definovat, nám bude stačit zatím jen informace, že se jedná o relaci $\Rightarrow_{G_0}$ takovou, že v každém kroku odvození přepisujeme všechny symboly přepisovaného slova, každý z těchto symbolů kterýmkoliv pravidlem z množiny $P$. Reflexivní a tranzitivní uzávěr relace $\Rightarrow_{G_0}$ budeme označovat $\Rightarrow^*_{G_0}$.

**Definice 5.3 (Jazyk 0L-systému)** *Jazyk generovaný 0L-systémem $G_0 = (V, P, \omega_0)$ je*

$$L(G_0) = \{w \in V^* \mid \omega_0 \Rightarrow^*_{G_0} w\}.$$

**Příklad 5.1**



Následující 0L-systém nad jednoprvkovou abecedou můžeme chápat jako jednoduchou simulaci dělení buněk.
$G_0 = (V, P, \omega_0)$, kde $V = \{a\},\ P = \{a \to aa\},\ \omega_0 = a$

Ukázka derivace:
$a \Rightarrow_{G_0} aa \Rightarrow_{G_0} aaaa \Rightarrow_{G_0} a^8 \Rightarrow_{G_0} a^{16} \Rightarrow_{G_0} a^{32} \Rightarrow_{G_0} \ldots$

Jazykem tohoto 0L-systému je
$L_4 = \left\{ a^{2^n} \;\middle|\; n \geq 0 \right\}$

Tento jazyk není bezkontextový, tedy nelze ho generovat žádnou bezkontextovou gramatikou.

**Příklad 5.2**

$G_0 = (\{a\},\ \{a \to aa\},\ aaa\}$

Ukázka odvození:

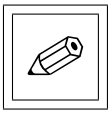
$aaa \Rightarrow_{G_0} aaaaaa \Rightarrow_{G_0} a^{12} \Rightarrow_{G_0} a^{24} \Rightarrow_{G_0} \ldots$

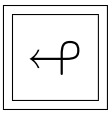
Tento 0L-systém generuje kontextový jazyk

$L(G_0) = L_{20} = \{a^{3 \cdot 2^n} \mid n \geq 1\}$

Množinu všech 0L-systémů budeme označovat zkratkou $0L$, tedy fakt, že $G_0$ je 0L-systém, můžeme napsat jako $G_0 \in 0L$. Třídu (množinu) jazyků generovaných 0L-systémy značíme $\mathscr{L}(0L)$, jak je v těchto skriptech pro třídy jazyků zvykem, a tedy fakt, že jazyk $L_{20}$ patří do této třídy, lze zapsat jako $L_{20} \in \mathscr{L}(0L)$.

**Věta 5.1** *Jazyk $L_{21} = \{a, aa\}$ nelze generovat žádným 0L-systémem, tedy $L_{21} \notin \mathscr{L}(0L)$.*





**Důkaz:** Předpokládejme, že existuje 0L-systém $G_0 = (\Sigma, P, \omega_0)$ generující jazyk $L_{21}$. Pak axiomem $\omega_0$ musí být některé ze slov jazyka (protože máme jen jednu abecedu $\Sigma$).

Když zvolíme jako axiom řetězec $a$, musíme mít možnost vygenerovat řetězec $aa$, tedy musí existovat derivace $a \Rightarrow^*_{G_0} aa$. To ale znamená, že lze symbol $a$ zdvojit, a proto by do jazyka musely patřit i řetězce $aaaa$, $aaaaaaaa$, $\ldots \Rightarrow$ axiomem nemůže být řetězec $a$.

Když zvolíme jako axiom řetězec $aa$, musíme mít možnost vygenerovat řetězec $a$, tedy musí existovat derivace $aa \Rightarrow^*_{G_0} a$. To ale znamená, že musí existovat možnosti derivace

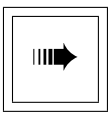
$a \Rightarrow^*_{G_0} a$

$a \Rightarrow^*_{G_0} \varepsilon$

a proto by do jazyka musel patřit i řetězec $\varepsilon$, tedy axiomem nemůže být řetězec $aa$.

Axiom musí vždy patřit do jazyka, tedy nenašli jsme 0L-systém, který by generoval tento jazyk, $L_{21} \notin \mathscr{L}(0L)$. □

**Důsledek 5.2** *Generativní síla 0L-systémů je neporovnatelná s generativní sílou bezkontextových gramatik, i když používá stejný typ pravidel – existují bezkontextové jazyky, které nepatří do třídy $\mathscr{L}(0L)$ a existují jazyky generované 0L-systémy, které nejsou bezkontextové. Proto paralelní způsob práce zásadně mění vlastnosti gramatik.*



*Zapisujeme*

$$\mathscr{L}(0L) \bigcirc\!\!\!\!\bigcirc \mathscr{L}(CF) \tag{5.1}$$

0L-systémy mají některé zajímavé vlastnosti týkající se výpočtů – v příkladu do těchto vlastností trochu nahlédneme:

**Příklad 5.3**

$G_F = (\{a, b\},\ \{a \to ab,\ b \to a\},\ b)$

Ukázka odvození:

$b \Rightarrow_{G_F} a \Rightarrow_{G_F} ab \Rightarrow_{G_F} aba \Rightarrow_{G_F} abaab \Rightarrow_{G_F} abaababa \Rightarrow_{G_F} abaababaabaab \Rightarrow_{G_F} \ldots$

Pro délky slov platí:

$|\omega_0| = 1$
$|\omega_1| = 1$
$|\omega_2| = 2$
$|\omega_3| = 3$
$|\omega_4| = 5$
$|\omega_5| = 8$
$|\omega_6| = 13 \ldots$

Prvky derivace jsou řetězce s délkou Fibbonacciho čísel, tedy
$|\omega_i| + |\omega_{i+1}| = |\omega_{i+2}|\ ,\ i \geq 0.$

## 5.3 E0L-systémy

E0L-systém má oproti 0L-systému navíc terminální abecedu $\Delta \subseteq V$. Do *jazyka generovaného E0L-systémem* řadíme pouze ty členy derivací, které se skládají jen z terminálních symbolů.

Terminální slovo se může dále vyvíjet, součástí jedné derivace bývá i více terminálních slov. Z biologického hlediska lze roli terminální abecedy chápat jako filtr vybírající pouze ty stavy prostředí, které nás z nějakého důvodu zajímají – členy derivace, které obsahují pouze prvky terminální abecedy, patří do jazyka generovaného systémem, kdežto ty členy derivace, které obsahují nejméně jeden symbol nepatřící do terminální abecedy, do jazyka generovaného systémem nepatří.

**Definice 5.4 (E0L-schéma)** *E0L-schéma je definováno jako $\mathcal{S}_E = (V, \Delta, P)$, kde*

- *$V$ je neprázdná konečná abeceda systému,*
- *$\Delta$ je neprázdná terminální abeceda systému, $\Delta \subseteq V$,*
- *$P$ je množina pravidel bezkontextového typu $P \subseteq V \times V^*$.*

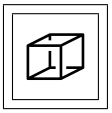
**Definice 5.5 (E0L-systém)** *E0L-systém je definován jako $G_E = (V, \Delta, P, \omega_0)$, kde*

- *$(V, \Delta, P)$ je E0L-schéma,*
- *$\omega_0$ je axiom (startovací slovo).*

Krok odvození je stejně určen jako u 0L-systémů.

**Definice 5.6 (Jazyk E0L-systému)** *Jazyk generovaný E0L-systémem $G_E$ určeným jako $G_E = (V, \Delta, P, \omega_0)$ je*

$$L(G_E) = \{w \in \Delta^* \mid \omega_0 \Rightarrow^* w\}.$$



**Příklad 5.4**

Následující E0L-systém $G_E$ generuje jazyk
$L_{19} = L_2 - \{\varepsilon\} = \{a^n b^n c^n \mid n \geq 1\}$

Stejně jako v příkladu 5.1 jde o jazyk, který není bezkontextový.
$G_E = (V, \Delta, P, \omega_0)$, kde $V = \{A, B, C, A', B', C', F, a, b, c\}$, $\Delta = \{a, b, c\}$,
$\omega_0 = ABC$, množina $P$ obsahuje tato vývojová pravidla:

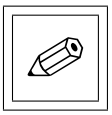
| | | | |
|---|---|---|---|
| $A \rightarrow AA' \mid a$ | $A' \rightarrow A' \mid a$ | $a \rightarrow F$ | $F \rightarrow F$ |
| $B \rightarrow BB' \mid b$ | $B' \rightarrow B' \mid b$ | $b \rightarrow F$ | |
| $C \rightarrow CC' \mid c$ | $C' \rightarrow C' \mid c$ | $c \rightarrow F$ | |

Ukázka dvou derivací, z nichž jen první vygeneruje terminální slovo:

$ABC \Rightarrow_{G_E} AA'BB'CC' \Rightarrow_{G_E} AA'A'BB'B'CC'C' \Rightarrow_{G_E} aaabbbccc \Rightarrow_{G_E} F^6 \Rightarrow_{G_E} \ldots$

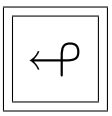
$ABC \Rightarrow_{G_E} AA'BB'CC' \Rightarrow_{G_E} AA'aBB'B'CC'C' \Rightarrow_{G_E} aaFbbbccc \Rightarrow_{G_E}$
$\Rightarrow_{G_E} FFFFFFFFF \Rightarrow_{G_E} \ldots$

V druhé derivaci se nikdy nedostaneme ke slovu, které se skládá pouze z terminálních symbolů, tedy ke slovům jazyka generovaného tímto systémem se dostaneme jen tak, že pravidla generující některý terminální symbol $(a, b, c)$ použijeme pro všechny symboly v přepisovaném slově zároveň. Symbol $F$ zde slouží pro synchronizaci generování terminálních symbolů $a, b, c$ tak, aby jich byl ve slově vždy stejný počet.



**Věta 5.3** *E0L-systémy jsou silnější než bezkontextové gramatiky, tj.*

$$\mathscr{L}(CF) \subset \mathscr{L}(E0L) \tag{5.2}$$



**Důkaz:** Algoritmus vytvoření E0L-systému ekvivalentního k zadané bezkontextové gramatice zde nebudeme přesně probírat, ale čtenář si ho jistě sám celý domyslí s malou nápovědou – každou sadu pravidel pro stejný neterminál $A \to \alpha_1 \mid \alpha_2 \mid \ldots$ v bezkontextové gramatice nahradíme sadou pravidel v E0L-systému $A \to A \mid \alpha_1 \mid \alpha_2 \mid \ldots$, čímž zajistíme možnost „sekvenčního“ odvození i při použití paralelismu.

Například jazyk generovaný E0L-systémem v příkladu 5.4 není bezkontextový, platí $L_{19} \in \mathscr{L}(E0L) - \mathscr{L}(CF)$. Proto platí, že E0L-systémy jsou silnější než bezkontextové gramatiky, třebaže se od nich liší vlastně jen postupem derivování (paralelním). □

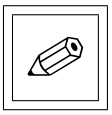
**Příklad 5.5**

$G_E = (\{A, a\},\ \{a\},\ \{A \to a,\ A \to aa,\ a \to a\},\ A)$

Ukázky odvození:

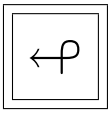
$A \Rightarrow_{G_E} a$
$A \Rightarrow_{G_E} aa$

Generovaný jazyk je
$L_{21} = \{a, aa\}$



**Věta 5.4** *E0L-systémy jsou silnější než 0L-systémy, tedy platí*

$$\mathscr{L}(0L) \subset \mathscr{L}(E0L) \tag{5.3}$$



**Důkaz:** 0L-systémy můžeme brát jako speciální případ E0L-systémů, kde $\Delta = V$. Tím je dána relace $\mathscr{L}(0L) \subseteq \mathscr{L}(E0L)$. Fakt, že jde o vlastní podmnožinu (tj. že existují jazyky generované E0L-systémy, které nelze generovat žádným 0L-systémem), dokazuje jazyk $L_{21} \in \mathscr{L}(E0L) - \mathscr{L}(0L)$. □

## 5.4 T0L-systémy

T0L-systém získáme z 0L-systému, když pravidla rozdělíme do tabulek a stanovíme, že v jednom kroku derivace se použijí pravidla z jediné vybrané tabulky.

T0L-systém je tedy určen abecedou, sadou tabulek $\mathcal{T}$ obsahujících pravidla (ta nahrazuje množinu pravidel $P$).

**Definice 5.7 (T0L-schéma)** *T0L-schéma je definováno jako $\mathcal{S}_T = (V, \mathcal{T})$, kde*

- *$V$ je neprázdná konečná abeceda systému,*
- *$\mathcal{T}$ je množina tabulek obsahujících pravidla bezkontextového typu $P \subseteq V \times V^*$.*

**Definice 5.8 (T0L-systém)** *T0L-systém je definován jako $G_T = (V, \mathcal{T}, \omega_0)$, kde*

- *$(V, \mathcal{T})$ je T0L-schéma,*
- *$\omega_0$ je axiom (startovací slovo).*

Krok odvození značený $\Rightarrow_{G_T}$ se provádí následovně:

- vybereme si některou tabulku z množiny tabulek $\mathcal{T}$,
- při odvození postupujeme stejně jako u 0L-systému (tj. všechny symboly přepisujeme), použijeme však pouze pravidla z jediné vybrané tabulky.

**Definice 5.9 (Jazyk T0L-systému)** *Jazyk generovaný T0L-systémem $G_E = (V, \mathcal{T}, \omega_0)$ je*

$$L(G_T) = \{w \in V^* \mid \omega_0 \Rightarrow^*_{G_T} w\}.$$

**Příklad 5.6**

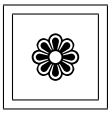
$G_T = (\{a\},\ \{T_1, T_2\},\ a)$, kde
$T_1 = \{a \to aa\}$,
$T_2 = \{a \to \varepsilon\}$

Ukázka odvození:

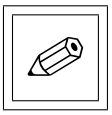
$a \Rightarrow_{G_T} aa \Rightarrow_{G_T} aaaa \Rightarrow_{G_T} \varepsilon$
(použili jsme dvakrát první tabulku, pak jednou druhou tabulku).

Generujeme tento jazyk:
$L_{22} = L_4 \cup \{\varepsilon\} = \left\{ a^{2^n} \;\middle|\; n \geq 0 \right\} \cup \{\varepsilon\}$

Kdybychom shrnuli obě tabulky T0L-systému z příkladu 5.6 do jedné a vytvořili tak pouhý 0L-systém, vygenerovaný jazyk by byl $a^*$.

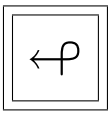


**Poznámka:** Dá se dokázat, že $L_{22} \notin \mathscr{L}(0L)$ (třebaže jazyk bez přidání $\varepsilon$ by pro 0L-systém nebyl problém), protože axiom musí obsahovat alespoň jeden znak a tedy přidání prázdného slova do abecedy by vyžadovalo „zkracující" pravidlo $a \to \varepsilon$, které by ovšem udělalo v odvození paseku – vznikl by jazyk $a^*$.



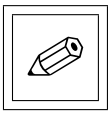
**Věta 5.5** *Ke každému 0L-systému lze sestrojit T0L-systém, který generuje tentýž jazyk, navíc, T0L-systémy jsou silnější než 0L-systémy:*

$$\mathscr{L}(0L) \subset \mathscr{L}(T0L) \tag{5.4}$$



**Důkaz:** Postup převodu 0L-systému na T0L-systém je jednoduchý – prostě všechna pravidla 0L-systému shrneme do jediné tabulky, ta se použije v každém kroku odvození.

Vlastní podmnožinu lze dokázat například pomocí jazyka $L_{22} \in \mathscr{L}(T0L) - \mathscr{L}(0L)$. □



**Věta 5.6** *Generativní síla T0L-systémů je neporovnatelná s generativní sílou bezkontextových gramatik:*

$$\mathscr{L}(T0L) \bigcirc\!\!\!\!\bigcirc \mathscr{L}(CF) \tag{5.5}$$

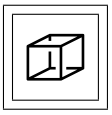
**Důkaz:** Do množiny $\mathscr{L}(T0L) - \mathscr{L}(CF)$ patří například jazyk
$L_4 = \left\{ a^{2^n} \;\middle|\; n \geq 0 \right\}$
(v příkladu 5.1 na straně 67 je generován 0L-systémem, a tedy je možno ho generovat i T0L-systémem).

Do množiny $\mathscr{L}(CF) - \mathscr{L}(T0L)$ zase můžeme zařadit jazyk
$L_1 = \{a^n b^n \mid n \geq 1\}$
(když počet znaků roste pomaleji než exponenciálně, potřebovali bychom $\varepsilon$-pravidlo $a \to \varepsilon$, a totéž i pro $b$, třeba i v jiné tabulce než exponenciální pravidlo $a \to aa$; jenže když existuje $\varepsilon$-pravidlo, pak se do jazyka dostane i prázdné slovo, což do $L_1$ nepatří). □

## 5.5 ET0L-systémy

ET0L-systémy jsou kombinace E0L a T0L-systémů, tedy máme terminální abecedu a pravidla rozdělená do tabulek. Definici si čtenář může vytvořit sám z definic E0L- a T0L-systémů.

**Příklad 5.7**



$G_{ET} = (\{A, B, C, a, b, c\},\ \{a, b, c\},\ \{T_1, T_2, T_3\},\ ABC)$, kde
$T_1 = \{A \to Aa,\ B \to Bb,\ C \to Cc,\ a \to a,\ b \to b,\ c \to c\}$,
$T_2 = \{A \to a,\ B \to b,\ C \to c,\ a \to a,\ b \to b,\ c \to c\}$
$T_3 = \{A \to \varepsilon,\ B \to \varepsilon,\ C \to \varepsilon,\ a \to \varepsilon,\ b \to \varepsilon,\ c \to \varepsilon\}$

Ukázka odvození:

$ABC \Rightarrow_{G_{ET}} AaBbCc \Rightarrow_{G_{ET}} AaaBbbCcc \Rightarrow_{G_{ET}} aaabbbccc$

Generovaný jazyk je

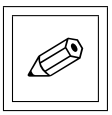
$L_2 = \{a^n b^n c^n \mid n \geq 0\}$

---

Při generování jazyka $L_2$ byly plně využity výhody ET0L-systému – oddělení terminální abecedy pro růst délky slova, který není exponenciální, a rozdělení do tabulek pro synchronizaci délky všech tří částí slova a navíc pro vygenerování prázdného slova. Tento jazyk nelze vygenerovat žádným 0L-systémem, E0L-systémem ani T0L-systémem.

Důkazy následujících vztahů jsou poměrně jednoduché, proto je necháme na čtenáři (vlastně vyplývají z vlastností dříve definovaných jednodušších L-systémů):

**Věta 5.7**



$$\mathscr{L}(0L) \subset \mathscr{L}(ET0L) \tag{5.6}$$
$$\mathscr{L}(E0L) \subset \mathscr{L}(ET0L) \tag{5.7}$$
$$\mathscr{L}(T0L) \subset \mathscr{L}(ET0L) \tag{5.8}$$
$$\mathscr{L}(CF) \subset \mathscr{L}(ET0L) \tag{5.9}$$

Existují i další varianty L-systémů včetně systémů pracujících s kontextem nebo stochastických L-systémů.

## 5.6 Možnosti využití L-systémů v grafice

L-systémy se v praxi využívají především v grafice, kde slouží k modelování a generování tzv. fraktálů[2]. Používají se například ve filmovém průmyslu pro generování realistických „hromadných" výjevů, jako je například les stromů nebo oblaka na nebi. Aby tyto výjevy vypadaly realisticky, lze do jejich generování zanést i prvek náhody – tak vznikly stochastické L-systémy, kde je každému pravidlu přiřazeno číslo z intervalu od 0 do 1 určující pravděpodobnost použití tohoto pravidla, součet hodnot pravděpodobností pravidel se stejnou levou stranou musí být 1.

Při grafické reprezentaci L-systémů symbol $F$ určuje čáru o zadané délce, která se má vykreslit a symboly $+$ a $-$ znamenají pootočení doprava nebo doleva o zadaný úhel. Podle axiomu a pravidel se provede určitý počet odvození a výsledný řetězec (stav prostředí) se pak graficky interpretuje. Na obrázku 5.1 na straně 76 je ukázka několika jednoduchých fraktálů.

**Příklad 5.8**

Fraktál na obrázku 5.1d se nazývá Kochova vločka a byl vytvořen L-systémem s axiomem $F++F++F$ a pravidlem $F \rightarrow F-F++F-F$ postupně čtyřmi průchody:

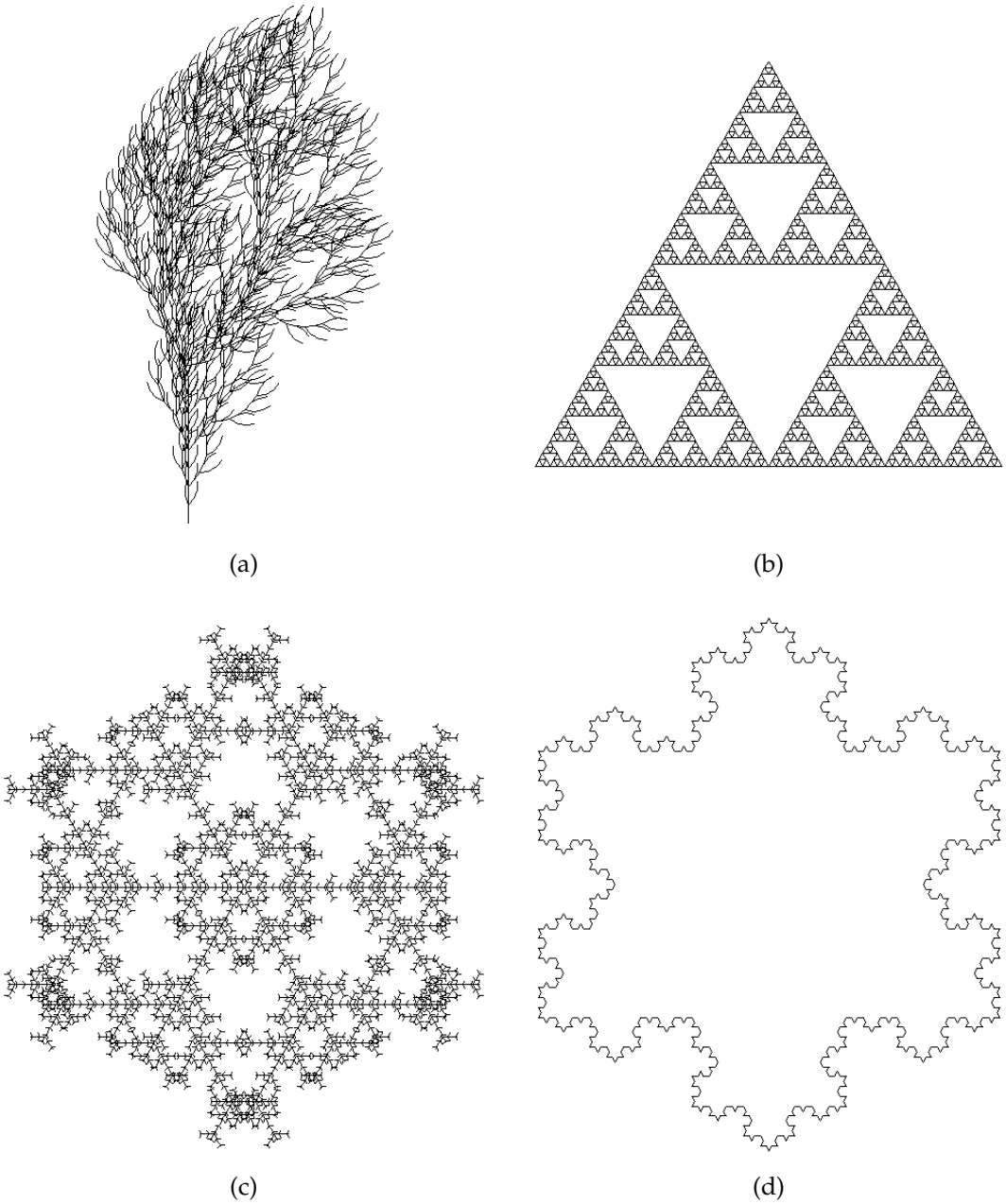
$$F++F++F \Rightarrow F-F++F-F++F-F++F-F++F-F++F-F++F-F \Rightarrow \ldots$$

(už třetí člen derivace se skládá ze 112 symbolů, tedy ho neuvádíme).

---

To jsou ovšem jen jednoduché příklady. Existuje více různých způsobů jak do obrázku zapracovat barvy, náhodnost, kontext apod. Některé programy pro vytváření velmi působivých fraktálů můžeme najít také na internetu, stačí do vyhledávače zadat například řetězec `L-systems` nebo `L-systémy`.

[2]Fraktál je soběpodobný útvar, který je velmi složitě tvarovaný a přesto je reprezentován velmi jednoduchou formou (třeba několika pravidly 0L-systému). To, že je útvar soběpodobný, znamená, že detaily útvaru v různých rozlišeních jsou podobné útvaru celému.



(a) (b)

(c) (d)

Obrázek 5.1: Několik fraktálů vygenerovaných pomocí L-Systémů

Kapitola 6

# Formální systémy

*V této kapitole se podíváme trochu dále za pouhé základy, kterými jsme se dosud zabývali. Nejdřív probereme různé typy derivací v gramatikách a potom si popíšeme základní vlastnosti gramatik a gramatických systémů, které nepatří do Chomského hierarchie.*

## 6.1 Derivace v gramatikách

Existuje mnoho různých typů derivací v gramatikách, nejběžnější jsou tyto:

- *Sekvenční postup* – sekvenční gramatiky (například gramatiky v Chomského hierarchii): v každém kroku derivace je vybráno a použito jedno pravidlo gramatiky na jeden neterminál.
- *Paralelní postup* – paralelní gramatiky (např. Lindenmayerovy systémy): v každém kroku derivace jsou najednou přepsány všechny neterminály ve slově, mohou být paralelně použita různá pravidla, jedno pravidlo i vícekrát na různých místech.
- *Sekvenčně-paralelní postup* – kombinace obou předchozích, např. v gramatice vybereme jedno pravidlo a to použijeme všude v generovaném slově, kde je to možné (tj. na více místech), tedy pravidla vybíráme sekvenčně a používáme je paralelně.
- *Distribuované výpočty* (více spolupracujících gramatik)
- *Další.*

## 6.2 Gramatiky používající řízenou sekvenční derivaci

Existuje několik typů gramatik, které sice používají sekvenční postup derivování, ale zároveň tento postup řídí (kdy, případně kde se které pravidlo použije). Jsou to například *maticové gramatiky*.

Maticová gramatika je rozšířením bezkontextové gramatiky z Chomského hierarchie. Množinu pravidel nahrazujeme množinou posloupností, kterou nazýváme *matice*. Jsou to posloupnosti pravidel s pevně daným pořadím (obdoba tabulek u T0L-systémů, až na pevné pořadí).

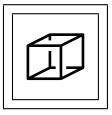
Derivace probíhá tak, že si v matici vybereme posloupnost, kterou v daném kroku použijeme, a pravidla v posloupnosti uplatňujeme v pořadí zleva doprava. Musíme použít všechna pravidla posloupnosti (a to právě jednou), v daném pořadí.

**Definice 6.1 (Maticová gramatika)** *Maticová gramatika je uspořádaná čtveřice* $G_M$, $G_M = (N, T, \mathcal{M}, S),\ \mathcal{M} = \{m_1, m_2, \ldots, m_k\}$, *kde* $\mathcal{M}$ *je matice* $k$ *posloupností pravidel.*

Krok odvození $\Rightarrow_M$ probíhá takto:

- vybereme si některou posloupnost pravidel $m_i \in \mathcal{M}$,
- na přepisovaný řetězec použijeme první pravidlo z $m_i$,
- použijeme druhé pravidlo z $m_i$ (to může zpracovat i symboly vygenerované prvním pravidlem z $m_i$),
- použijeme třetí pravidlo z $m_i$, atd.

**Příklad 6.1**



$G_M = (\{S, A, B, C, D\},\ \{a, b\},\ \{m_1, m_2, m_3, m_4\},\ S)$, kde

$m_1 = (S \to ABCD)$,
$m_2 = (A \to aA,\ B \to B,\ C \to aC,\ D \to D)$,
$m_3 = (A \to A,\ B \to bB,\ C \to C,\ D \to bD)$,
$m_4 = (A \to a,\ B \to b,\ C \to a,\ D \to b)$

Ukázka odvození:

$S \Rightarrow_M ABCD \Rightarrow_M aABaCD \Rightarrow_M aAbBaCbD \Rightarrow_M aAbbBaCbbD \Rightarrow_M aabbbaabbb$

V derivaci jsme použili nejdřív první matici, pak druhou, dvakrát za sebou třetí a nakonec čtvrtou, která ukončila derivaci. Generovaný jazyk není bezkontextový: $L(G_M) = L_{23} = \{a^i b^j a^i b^j \mid i, j \geq 1\}$

**Příklad 6.2**

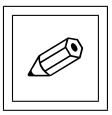
$G_M = (\{S, A, B, C\},\ \{a, b, c\},\ \{m_1, m_2, m_3\},\ S)$, kde
$m_1 = (S \to ABC)$,
$m_2 = (A \to aA,\ B \to bB,\ C \to cC)$,
$m_3 = (A \to a,\ B \to b,\ C \to c)$

Ukázka odvození:

$S \Rightarrow_M ABC \Rightarrow_M aAbBcC \Rightarrow_M aaAbbBccC \Rightarrow_M aaabbbccc$

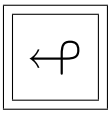
Generovaný jazyk je
$L_{19} = \{a^n b^n c^n \mid n \geq 1\}$

---



**Věta 6.1** *Maticové gramatiky jsou silnější než bezkontextové:*

$$\mathscr{L}(CF) \subset \mathscr{L}(MAT) \tag{6.1}$$



**Důkaz:** Pro kteroukoliv bezkontextovou gramatiku sestrojíme maticovou gramatiku, která by generovala tentýž jazyk, velmi jednoduše – pro každé pravidlo bezkontextové gramatiky vytvoříme samostatnou (tedy jednoprvkovou) posloupnost, matice tedy bude obsahovat tolik posloupností, kolik měla původní gramatika pravidel. Takto vytvořená maticová gramatika přesně kopíruje sekvenční postup odvozování původní bezkontextové gramatiky.

Vlastní podmnožina je zřejmá: $L_{19} = \{a^n b^n c^n \mid n \geq 1\} \in \mathscr{L}(MAT) - \mathscr{L}(CF)$ (je generován maticovou gramatikou v příkladu 6.2). □

## 6.3 Gramatické systémy

Dosud uvedené modely počítají s jedinou gramatikou vybavenou (vývojovými) pravidly, a s jediným slovem, které je zpracováváno pravidly gramatiky. Toto slovo můžeme považovat za *prostředí*, do kterého gramatika zasahuje svými pravidly.

Tento jednoduchý model lze rozšířit na *gramatický systém* buď tak, že použijeme několik gramatik, které mají každá vlastní prostředí a mezi nimi probíhá určitá komunikace, a nebo tyto gramatiky necháme spolupracovat na společném prostředí a případně přidáme některé další vlastnosti. Komunikace pak může probíhat

třeba přes prostředí. Činnost těchto gramatik bývá synchronizována *univerzálními hodinami*.

První modely gramatických systémů se objevily už v roce 1978 a byly motivovány využitím v distribuované umělé inteligenci. Šlo o tzv. spolupracující (cooperating) gramatiky. Větších možností využití se dočkaly tzv. CD (Cooperated Distributed) gramatické systémy, které pracovaly metodou dnes například v sociálních vědách označovanou jako *černá tabule* (blackboard architecture for problem solving) – společné prostředí je jakousi tabulí, na kterou se „dívají“ gramatiky a sledují, zda se na této tabuli neobjeví něco, co dokážou zpracovat svými pravidly.

Podobné modely mají své praktické využití například v robotice nebo při programování nezávislých softwarových agentů. Gramatickým systémem modelujeme chování *robotů* či *agentů* (představovaných gramatikami, pravidla jsou množina proveditelných akcí) pohybujících se ve společném prostředí, ať už softwarovém (operační systém či počítačová síť) nebo fyzicky reálném (místnost). V takovém případě někdy hovoříme o *agentových systémech*.

### 6.3.1 Kolonie

Kolonie jsou velmi jednoduchý gramatický systém, který můžeme chápat jako společenství jednoduchých gramatik – *komponent*, které spolupracují na sdíleném prostředí.

Všechny gramatiky sdružené v kolonii mají společnou abecedu a terminální abecedu. Abeceda kolonie je množina symbolů, které se mohou vyskytovat v prostředí a se kterými tedy lze pracovat, terminální abeceda (je podmnožinou abecedy kolonie) určuje symboly, ze kterých se mohou skládat terminální slova, která pak řadíme do jazyka kolonie.

Každá komponenta má svůj *startovací symbol* a *jazyk*. Startovací symbol je některý prvek abecedy kolonie (může patřit i do terminální abecedy systému), určuje „objekt“, se kterým komponenta dokáže pracovat. Jazyk komponenty je množina slov nad abecedou kolonie *neobsahujících startovací symbol komponenty*, je to jakási množina „akcí“, které komponenta může s některým výskytem svého startovacího symbolu v prostředí provést.

Činnost komponenty můžeme tedy chápat tak, že hledá v prostředí svůj startovací symbol, a pokud najde kterýkoliv jeho výskyt (případně nezabraný jinou

komponentou, pokud komponenty pracují paralelně), nahradí ho některým slovem svého jazyka.

Komponenta sama o sobě je vlastně jednoduchá gramatika generující konečný jazyk (konečný proto, že startovací symbol komponenty se nesmí vyskytovat v jejím jazyce), lze ji přepsat na regulární gramatiku tak, že na levou stranu pravidel dáme startovací symbol komponenty a na pravou postupně všechna slova jejího jazyka.

Prostředí je zpracováváno pouze komponentami. Může obsahovat jakýkoliv řetězec symbolů patřících do abecedy kolonie. Tento řetězec nazýváme *stavem prostředí*, počátečním stavem prostředí je *axiom* kolonie.

**Definice 6.2 (Kolonie)** *Kolonie je uspořádaná čtveřice* $\mathcal{C} = (V, T, \mathcal{R}, \omega_0)$, *kde*

- $V$ *je konečná neprázdná abeceda kolonie,*
- $T$ *neprázdná terminální abeceda kolonie,* $T \subseteq V$,
- $\mathcal{R}$ *je konečná* multimnožina[1] *komponent,*
  $\mathcal{R} = \{(S, F) \mid S \in V),\ F \subseteq (V - \{S\})^*,\ F$ *je konečná a neprázdná*$\}$, $S$ *je startovací symbol komponenty* $(S, F)$ *a* $F$ *je konečný jazyk této komponenty,*
- $\omega_0$ *je axiom.*

Pro kolonie existuje hodně typů odvození, k základním řadíme tyto čtyři:

- *sekvenční mód* ($b$, basic) – kolonie pracuje podobně jako gramatiky Chomského hierarchie, tedy v každém kroku odvození je náhodně vybrána jedna komponenta, která v prostředí zpracuje jediný výskyt svého startovacího symbolu (nahradí ho některým slovem svého jazyka),
- *sekvenčně-paralelní mód* ($t$, terminal) – v každém kroku odvození je sekvenčně vybrána jediná komponenta, ta pak zpracuje všechny výskyty svého startovacího symbolu v prostředí, každý z nich nahradí některým ze slov svého jazyka (obecně každý výskyt svého startovacího symbolu může nahradit jiným slovem),
- *slabý paralelismus* ($wp$, weakly parallel) – každá komponenta, která může pracovat, také musí pracovat, tedy když se v daném kroku odvození v prostředí

[1] Pro připomenutí: v multimnožině se narozdíl od množiny může vyskytovat i více stejných prvků, tedy v multimnožině komponent může existovat i více komponent stejně definovaných.

vyskytne startovací symbol komponenty a není zpracováván jinou komponentou, komponenta tento symbol musí zpracovat,

- *silný paralelismus* (*sp*, strongly parallel) – podobně jako slabý paralelismus, jen navíc v případě, že komponenta by měla pracovat (její startovací symbol se vyskytuje v prostředí), ale všechny výskyty jejího startovacího symbolu zabraly jiné komponenty (se stejným startovacím symbolem), je derivace zablokována, odvozování končí.

**Příklad 6.3**

$\mathcal{C} = (\{A, B, a\},\ \{a\},\ \mathcal{R},\ A)$, kde $\mathcal{R} = \{(A, \{BB\}),\ (A, \{a\}),\ (B, \{A\})\}$

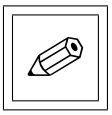
Ukázky derivací s použitím $b$ a $t$ módu odvození:

$A \Rightarrow_b BB \Rightarrow_b AB \Rightarrow_b aB \Rightarrow_b aA \Rightarrow_b aBB \Rightarrow_b aBA \Rightarrow_b \ldots$
$A \Rightarrow_t B^2 \Rightarrow_t A^2 \Rightarrow_t B^4 \Rightarrow_t A^4 \Rightarrow_t B^8 \Rightarrow_t A^8 \Rightarrow_t a^8$

S použitím módů odvození $b$ a $t$ jsou generovány tyto jazyky:

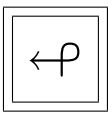
$L(\mathcal{C}, b) = \{a^n \mid n > 0\}$
$L(\mathcal{C}, t) = L_4 = \left\{a^{2^n} \;\middle|\; n \geq 0\right\}$ (také u 0L-systému v příkladu 5.1 na straně 67).

---



**Věta 6.2** *Sekvenční kolonie jsou ve své generativní síle stejně silné jako bezkontextové jazyky:*

$$\mathscr{L}(CF) \simeq \mathscr{L}(COL_b) \tag{6.2}$$



**Důkaz:** Podle bezkontextové gramatiky $G = (N, T, P, S)$ vytvoříme kolonii s $b$ módem odvození $\mathcal{C} = (V, T, \mathcal{R}, \omega_0)$ jednoduše takto:

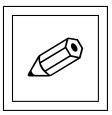
- odstraníme přímou rekurzi (pravidla, kde na pravé straně je neterminál z levé strany $A \to \alpha A \beta$, přepíšeme na dvojici pravidel $A \to A',\ A' \to \alpha A \beta$), symboly $A'$ jsou nově přidané,
- pravidla se stejnou levou stranou shrneme vždy do jedné komponenty (tj. budeme mít tolik komponent, kolik neterminálů),
- $V = N \cup T \cup \{A' \mid A \in N\}$,
- $\omega_0 = S$,
- $\mathcal{R} = \{(A, \{A'\}) \mid A \in N\} \cup \left\{(A', \{\alpha \mid (A \to \alpha) \in P\}) \;\middle|\; A \in N\right\}$.

Odvození bude probíhat prakticky stejně jako v bezkontextové gramatice, až na mezikroky provedené pro odstranění rekurze.

Opačný postup vyžaduje oddělení terminálů od symbolů, které lze přepisovat. Z kolonie $\mathcal{C} = (V, T, \mathcal{R}, \omega_0)$ vytvoříme bezkontextovou gramatiku $G = (N, T, P, S)$ takto:

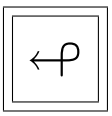
- pro každý symbol terminální abecedy $a \in T$ vytvoříme nový (nově přidaný) neterminál $N_a$,
- v gramatice budou pravidla $N_a \to a$ pro každý symbol $a \in T$,
- pravidla gramatiky vytvoříme z komponent tak, že v jazyce komponenty všechny terminály $a$ nahradíme novými neterminály $N_a$, označme $\overline{\alpha}$ řetězec, který vznikl z řetězce $\alpha$ tímto nahrazením,
- $N = V - T \cup \{N_a \mid a \in T\} \cup \{S\},\ S \notin V$,
- $P = \{N_a \to a \mid a \in T\} \cup \left\{ A \to \overline{f_1} \mid \overline{f_2} \mid \ldots \;\middle|\; (A, \{f_1, f_2, \ldots\}) \in \mathcal{R} \right\} \cup \{S \to \overline{\omega_0}\}$

□



**Věta 6.3** *Sekvenčně-paralelní kolonie jsou silnější než bezkontextové jazyky:*

$$\mathscr{L}(CF) \subset \mathscr{L}(COL_t) \tag{6.3}$$



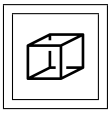
**Důkaz:** Podle bezkontextové gramatiky $G = (N, T, P, S)$ vytvoříme kolonii s $t$ módem odvození $\mathcal{C} = (V, T, \mathcal{R}, \omega_0)$ jednoduše takto:

- stejně jako v předchozím důkazu odstraníme přímou rekurzi (pravidla, kde na pravé straně je neterminál z levé strany $A \to \alpha A \beta$, přepíšeme na dvojici pravidel $A \to A',\ A' \to \alpha A \beta$), symboly $A'$ jsou nově přidané,
- pravidla se stejnou levou stranou shrneme vždy do jedné komponenty (tj. budeme mít tolik komponent, kolik neterminálů),
- $V = N \cup T \cup \{A' \mid A \in N\}$,
- $\omega_0 = S$,
- $\mathcal{R} = \{(A, \{A'\}) \mid A \in N\} \cup \left\{ (A', \{A\} \cup \{\alpha \mid (A \to \alpha) \in P\}) \;\middle|\; A \in N \right\}$.

Rozdíl oproti předchozímu důkazu je v definici množiny $\mathcal{R}$, kde musíme zajistit možnost simulace sekvenčního odvození. Zatímco při použití $t$ módu odvození v kolonii se v jednom kroku odvození přepisují všechny symboly, zde musí být možnost přepsat jakoby jen jeden výskyt symbolu $A$ a ostatní nechat bez přepsání (v simulaci na jeden výskyt symbolu $A$ použijeme dotyčné pravidlo, na ostatní použijeme přepis na $A$ přes $A'$).

Vlastní podmnožinu lze dokázat například jazykem $L_4 = \left\{ a^{2^n} \;\middle|\; n \geq 0 \right\}$ generovaným kolonií s $t$ módem odvození v příkladu 6.3 na straně 82. Tento jazyk není bezkontextový, což se dá dokázat například pomocí Pumping lemma. □

**Příklad 6.4**



$\mathcal{C} = (\{M, N, P, a, b\},\ \{a, b\},\ \mathcal{R},\ MM)$, kde množina $\mathcal{R}$ obsahuje komponenty

$(M, \{PPNP, PNPP, \varepsilon\})$, $(P, \{a\})$,
$(N, \{M\})$, $(P, \{b\})$

Ukázky derivace s použitím $wp$ a $sp$ módu odvození:

$MM \Rightarrow_{wp} PPNPM \Rightarrow_{wp} baMP \Rightarrow_{wp} baa$
$MM \Rightarrow_{wp} PPNPM \Rightarrow_{wp} aPMb \Rightarrow_{wp} aaPNPPb \Rightarrow_{wp} aaaMPbb \Rightarrow_{wp} aaabbb$
$MM \Rightarrow_{sp} PPNPM \Rightarrow_{sp} PaMb$ *odvození je zablokováno*
$MM \Rightarrow_{sp} PPNPM \Rightarrow_{sp} abMPPNPP \Rightarrow_{sp} abbaMPP \Rightarrow_{sp} abbaab$

S použitím módů odvození $wp$ a $sp$ jsou generovány tyto jazyky:

$L(\mathcal{C}, wp) = L_{24} = \left\{ w \in \{a, b\}^* \;\middle|\; \left| |w|_a - |w|_b \right| \leq 1, |w| = 3n, n \geq 0 \right\}$
$L(\mathcal{C}, sp) = L_{25} = \left\{ w \in \{a, b\}^* \;\middle|\; |w|_a = |w|_b,\ |w| = 6n, n \geq 0 \right\} - \{a^i b^i, b^i a^i \mid i \geq 1\}$

V příkladu 6.4 vidíme u $sp$ módu využití blokování odvození v případě, že v prostředí je právě jeden symbol $P$, přičemž máme dvě komponenty s tímto startovacím symbolem. Zde je blokování využito pro zajištění stejného počtu symbolů $a$ a $b$ v generovaném slově.

Kolonie s $wp$ a $sp$ módem odvození jsou silnější než bezkontextové jazyky, důkaz by byl stejný jako u důkazu věty 6.2 na straně 82.

Na příkladu 6.5 vidíme využití blokování odvození některých slov, zde použito k synchronizování generování obou stejných polovin slova.

**Příklad 6.5**

$\mathcal{C} = (\{P, A, B, a, b\},\ \{a, b\},\ \mathcal{R},\ PP)$, v množině $\mathcal{R}$ jsou komponenty
$(P, \{aA, bB, \varepsilon\})$, $(A, \{P\})$, $(B, \{P\})$,
$(P, \{aA, bB, \varepsilon\})$, $(A, \{P\})$, $(B, \{P\})$

Ukázka derivace s použitím $sp$ (silně paralelního) módu odvození:

$PP \Rightarrow_{sp} aAaA \Rightarrow_{sp} aPaP \Rightarrow_{sp} aaAaaA \Rightarrow_{sp} aaPaaP \Rightarrow_{sp} aabBaabB \Rightarrow_{sp}$
$\Rightarrow_{sp} aabPaabP \Rightarrow_{sp} aabaab$

Tato kolonie s $sp$ módem odvození generuje jazyk

$L(\mathcal{C}, wp) = L_5 = \{ww \mid w \in \{a, b\}^*\}$

S $wp$ módem odvození bychom vygenerovali jazyk $\{a, b\}^*$.

---

Jistou možnost blokování některých odvození však lze použít i u $wp$ módu odvození (nekonečným cyklem), jak vidíme na příkladu 6.6.

**Příklad 6.6**

$\mathcal{C} = (\{P, Q, R, X, Y, B, B', a, b\},\ \{a, b\},\ \mathcal{R},\ PP)$, v množině $\mathcal{R}$ jsou komponenty
$(P, \{aQX, bRX, Y\})$, $(Q, \{P\})$, $(R, \{P\})$, $(X, \{\varepsilon\})$, $(Y, \{\varepsilon\})$, $(B, \{B'\})$,
$(P, \{aRY, bQY, X\})$, $(Q, \{B\})$, $(R, \{B\})$, $(X, \{B\})$, $(Y, \{B\})$, $(B', \{B\})$

Ukázka derivací s použitím $wp$ (slabě paralelního) módu odvození:

$PP \Rightarrow_{wp} aQXaRY \Rightarrow_{wp} aPaP \Rightarrow_{wp} abRXabQY \Rightarrow_{wp} abPabP \Rightarrow_{wp} abXabY \Rightarrow_{wp}$
$\Rightarrow_{wp} abab$
$PP \Rightarrow_{wp} aQXbQY \Rightarrow_{wp} aPbB \Rightarrow_{wp} aYbB' \Rightarrow_{wp} abB \Rightarrow_{wp} abB' \Rightarrow_{wp} \ldots$

S použitím módu odvození $wp$ je generován jazyk $L_5$. Kdybychom zde použili při odvozování $sp$ mód, získali bychom prázdný jazyk.

---

# Seznam použitých jazyků

$L_1 = \{a^n b^n \mid n \geq 1\}$



$L_2 = \{a^n b^n c^n \mid n \geq 0\}$



$L_3 = \left\{a^{n^2} \;\middle|\; n \geq 0\right\}$



$L_4 = \left\{a^{2^n} \;\middle|\; n \geq 0\right\}$