

## Foja

1. **Regulárne jazyky.** [Deterministické a nedeterministické konečné automaty, regulárne gramatiky, regulárne výrazy, ekvivalencia popisov regulárnych jazykov, pumpovacia lema, uzáverové vlastnosti.]

**Definícia 1.** *Deterministický konečný automat* je pätnica  $A = (K, \Sigma, \delta, q_0, F)$ , kde  $K$  je konečná množina stavov,  $\Sigma$  je vstupná abeceda,  $\delta: K \times \Sigma \rightarrow K$  je prechodová funkcia,  $q_0 \in K$  je počiatok stavu a  $F \subseteq K$  je množina koncových (alebo akceptačných) stavov.

Všimnime si, že dôsledkom existencie počiatok stavu  $q_0$  je neprázdnosť množiny stavov  $K$ .

**Definícia 2.** *Konfigurácia* deterministického konečného automatu  $A = (K, \Sigma, \delta, q_0, F)$  je dvojica  $(q, w)$ , kde  $q \in K$  je stav a  $w \in \Sigma^*$  je slovo (reprezentujúce nedočítanú časť vstupu).

**Definícia 3.** *Krok výpočtu* deterministického konečného automatu  $A = (K, \Sigma, \delta, q_0, F)$  je binárna relácia  $\vdash_A$  na množine konfigurácií automatu  $A$  taká, že pre  $p, q \in K$  a  $u, v \in \Sigma^*$  je  $(p, u) \vdash_A (q, v)$  práve tedy, keď existuje  $c \in \Sigma$  také, že  $u = cv$  a  $\delta(p, c) = q$ .

V prípade, že je uvažovaný automat  $A$  zrejmý z kontextu, píšeme namiesto  $\vdash_A$  často iba  $\vdash$ . Ide pritom o reláciu; pre každé  $k \in \mathbb{N}$  sú teda v relácii  $\vdash^k$  tie dvojice konfigurácií, medzi ktorými možno „prejsť“ na  $k$  krokov výpočtu. V relácii  $\vdash^*$ , reflexívno-tranzitívnom uzávere relácie  $\vdash$ , sú tie dvojice konfigurácií, medzi ktorými možno „prejsť“ na nejaký počet krokov a v tranzitívnom uzávere, relácii  $\vdash^+$ , sú tie dvojice konfigurácií, medzi ktorými možno „prejsť“ na nejaký nenulový počet krokov. Relácia  $\vdash^0$  je identita.

**Definícia 4.** Nech  $A = (K, \Sigma, \delta, q_0, F)$  je deterministický konečný automat. *Jazyk akceptovaný* (alebo rozoznávaný) automatom  $A$  je daný ako  $L(A) = \{w \in \Sigma^* \mid \exists q \in F : (q_0, w) \vdash^* (q, \varepsilon)\}$ .

**Poznámka 1.** V literatúre sa deterministické konečné automaty niekedy definujú aj s čiastočnou prechodovou funkciou, čo znamená, že hodnota  $\delta(q, c)$  nemusí byť definovaná pre všetky stavy  $q$  a symboly  $c$ .<sup>1</sup> Na tomto predmete však pracujeme s definíciou s úplnou prechodovou funkciou – pri zadávaní deterministického konečného automatu je teda potrebné špecifikovať výstupy jeho prechodovej funkcie pre všetky stavy a všetky písmená.

Nedeterministický konečný automat sa oproti deterministickému líší hlavne tým, že v ňom z jedného stavu môže viest' aj viacero prechodov na ten istý symbol (prípadne nemusí existovať žiadny takýto prechod). To bolo na prednáške sformalizované pomocou prechodovej funkcie, ktorá pre každý stav  $q$  a symbol  $c$  vráti *množinu* stavov  $\delta(q, c)$ . Výpočet automatu tak môže zo stavu  $q$  na písmeno  $c$  pokračovať do všetkých stavov z množiny  $\delta(q, c)$ , v dôsledku čoho môže na vstupnom slove  $w$  existovať viacero rôznych výpočtov. Automat pritom slovo  $w$  akceptuje práve tedy, keď *existuje aspoň jeden* jeho výpočet na slove  $w$ , ktorý sa skončí v akceptačnom stave. Pri definícii z prednášky majú navyše nedeterministické konečné automaty možnosť vykonávať aj *prechody na prázdne slovo*, pri ktorých nič neprečítajú zo vstupu.

**Definícia 5.** Nedeterministický konečný automat je päťica  $A = (K, \Sigma, \delta, q_0, F)$ , kde  $K$  je konečná množina stavov,  $\Sigma$  je vstupná abeceda,  $\delta: K \times (\Sigma \cup \{\varepsilon\}) \rightarrow 2^K$  je prechodová funkcia,  $q_0 \in K$  je počiatočný stav a  $F \subseteq K$  je množina koncových (alebo akceptačných) stavov.

**Definícia 6.** Konfigurácia nedeterministického konečného automatu  $A = (K, \Sigma, \delta, q_0, F)$  je dvojica  $(q, w)$ , kde  $q \in K$  je stav a  $w \in \Sigma^*$  je slovo (reprezentujúce nedočítanú časť vstupu).

**Definícia 7.** Krok výpočtu nedeterministického konečného automatu  $A = (K, \Sigma, \delta, q_0, F)$  je binárna relácia  $\vdash_A$  na množine konfigurácií automatu  $A$  taká, že pre  $p, q \in K$  a  $u, v \in \Sigma^*$  je  $(p, u) \vdash_A (q, v)$  práve vtedy, keď existuje  $z \in \Sigma \cup \{\varepsilon\}$  také, že  $u = zv$  a  $q \in \delta(p, z)$ .

Podobne ako pri deterministických konečných automatoch píšeme v prípadoch, keď je uvažovaný automat  $A$  zrejmý z kontextu, namiesto  $\vdash_A$  často iba  $\vdash$ .

**Definícia 8.** Nech  $A = (K, \Sigma, \delta, q_0, F)$  je nedeterministický konečný automat. Jazyk akceptovaný (alebo rozoznávaný) automatom  $A$  je daný ako  $L(A) = \{w \in \Sigma^* \mid \exists q \in F : (q_0, w) \vdash^* (q, \varepsilon)\}$ .

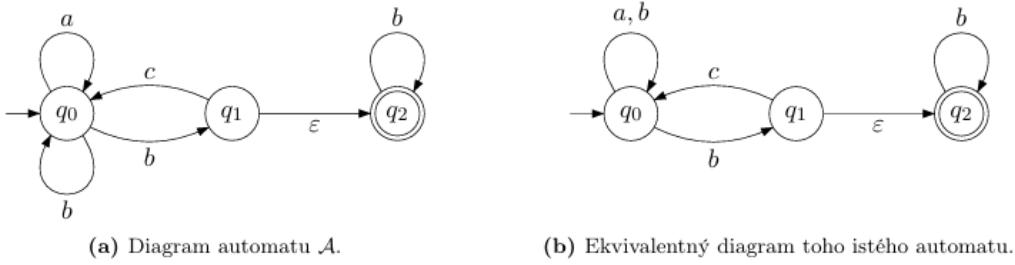
Častým spôsobom ako zadať konečný automat je tzv. *prechodový diagram*, v ktorom sú stavy znázornené „kolečkami“, prechodová funkcia šípkami medzi stavmi, počiatočný stav krátkou šípkou do zodpovedajúceho „kolečka“ a akceptačné stavы „zdvojenými kolečkami“.

*Priklad 1.* Uvažujme nedeterministický konečný automat  $A = (K, \Sigma, \delta, q_0, F)$  s  $K = \{q_0, q_1, q_2\}$ ,  $\Sigma = \{a, b, c\}$ ,  $F = \{q_2\}$  a prechodovou funkciou danou nasledovne:

$$\begin{aligned} \delta(q_0, a) &= \{q_0\}, & \delta(q_0, b) &= \{q_0, q_1\}, & \delta(q_0, c) &= \emptyset, & \delta(q_0, \varepsilon) &= \emptyset, \\ \delta(q_1, a) &= \emptyset, & \delta(q_1, b) &= \emptyset, & \delta(q_1, c) &= \{q_0\}, & \delta(q_1, \varepsilon) &= \{q_2\}, \\ \delta(q_2, a) &= \emptyset, & \delta(q_2, b) &= \{q_2\}, & \delta(q_2, c) &= \emptyset & \delta(q_2, \varepsilon) &= \emptyset. \end{aligned}$$

Lahko vidieť – a pri troche úsilia aj dokázať – že tento automat rozoznáva jazyk  $L(A) = \{a, b, bc\}^*b^+$ .

Prechodový diagram automatu  $A$  je na obrázku 1a. Násobné prechody medzi rovnakými stavmi niekedy pre prehľadnosť kreslíme ako jedinú šípkou s viacerými ohodnoteniami, tak ako v prípade slučky v diagrame na obrázku 1b.



**Obr. 1:** Prechodový diagram nedeterministického konečného automatu  $A$ .

Nasledujúce jednoduché tvrdenie, ktorého dôkaz prenehávame ako jednu z úloh na nasledujúce cvičenie, budeme často používať bez toho, aby sme to explicitne uvádzali (platí pre deterministické aj pre nedeterministické konečné automaty):

**Tvrdenie 1.** Nech  $A = (K, \Sigma, \delta, q_0, F)$  je konečný automat. Potom:

- a) Pre všetky  $p, q \in K$  a  $u, v \in \Sigma^*$  je  $(p, uv) \vdash^* (q, v)$  práve vtedy, keď  $(p, u) \vdash^* (q, \varepsilon)$ ,
- b) Pre všetky  $p, q, r \in K$  a  $u, v \in \Sigma^*$  také, že  $(p, u) \vdash^* (q, \varepsilon)$  a  $(q, v) \vdash^* (r, \varepsilon)$  je aj  $(p, uv) \vdash^* (r, \varepsilon)$ .

## 2 Dokazovanie správnosti konštrukcie automatu

Dokazovanie tvrdenia  $L(A) = L$  pre konečný automat  $A$  a jazyk  $L$  je v mnohom podobné dokazovaniu takýchto tvrdení pre bezkontextové gramatiky. Hlavným spoločným znakom je matematická indukcia, ktorou sa obyčajne dokáže o niečo silnejšie tvrdenie, než to pôvodne zamýšľané. Pre konečné automaty sú takýmto vhodným silnejším tvrdením väčšinou *invarianty* pre jednotlivé stavy: pre každý stav  $q$  sa charakterizujú slová  $w$  dočítané automatom v stave  $q$  – čiže slová  $w$ , pre ktoré je  $(q_0, w) \vdash^* (q, \varepsilon)$ .<sup>2</sup> Tvrdenie  $L(A) = L$  potom obyčajne vyplýva z invariantov pre akceptačné stavy.

### 1 Determinizácia konečných automatov

Je jedným z kľúčových výsledkov teórie konečných automatov, že deterministické a nedeterministické konečné automaty sú rovnako silné. Jeden smer tohto tvrdenia je zrejmý: aj keď deterministický konečný automat v zmysle jeho formálnej definície z prednášky *nie je* súčasne aj nedeterministickým konečným automatom, po drobnej úprave prechodovej funkcie sa ním stane. Zvyšná časť tvrdenia bola dokázaná na prednáške a možno ju sformulovať v podobe nasledujúcej vety.

**Veta 1.** *Nech  $A = (K, \Sigma, \delta, q_0, F)$  je nedeterministický konečný automat. Potom existuje deterministický konečný automat  $A' = (K', \Sigma', \delta', q'_0, F')$  taký, že  $L(A') = L(A)$ .*

#### 1.1 Podmnožinová konštrukcia

Algoritmus determinizácie nedeterministického konečného automatu predpokladá ako svoj vstup už „odepsilonovaný“ nedeterministický konečný automat  $A = (K, \Sigma, \delta, q_0, F)$  a spočíva v takzvanej podmnožinovej konštrukcii:

1. Nech  $K' = 2^K$ .
2. Pre každé  $Q \in K'$  a každé  $c \in \Sigma$  polož  $\delta'(Q, c) = \{p \in K \mid \exists q \in Q : p \in \delta(q, c)\}$ .
3. Za počiatočný stav  $q'_0$  vezmi množinu  $\{q_0\}$ .
4. Za  $F'$  vezmi množinu všetkých stavov  $Q \in K'$  takých, že  $Q \cap F \neq \emptyset$ .

Výstupom je deterministický konečný automat  $A' = (K', \Sigma, \delta', q'_0, F')$  taký, že  $L(A') = L(A)$ .

#### 1.2 Zefektívnenie podmnožinovej konštrukcie

Výsledkom podmnožinovej konštrukcie v podobe, v akej bola opísaná vyššie, môže vo všeobecnosti byť aj automat s veľmi veľkým množstvom nedosiahnutelných – a teda zbytočných – stavov. Tomuto nedostatku možno predísť *postupným konštruuovaním* ekvivalentného deterministického automatu  $A'$  tak, aby tento obsahoval *iba dosiahnutelné* stavy. Vstupom nasledujúcej schémy determinizačného algoritmu<sup>1</sup> je opäť už „odepsilonovaný“ nedeterministický konečný automat  $A = (K, \Sigma, \delta, q_0, F)$  a jej výstupom bude deterministický konečný automat  $A' = (K', \Sigma, \delta', q'_0, F')$  taký, že  $L(A') = L(A)$ . V priebehu vykonávania bude množina  $K'_?$  pozostávať z tých stavov konštruuovaného automatu, ktoré už boli pridané do množiny  $K'$ , ale ešte pre ne nie sú definované výstupy prechodovej funkcie  $\delta'$ .

1. Inicializuj  $K' := \{\{q_0\}\}$  a  $K'_? := \{\{q_0\}\}$ .
2. Vyber ľubovoľné  $Q \in K'_?$  a polož  $K'_? := K'_? - \{Q\}$ .
3. Opakuj pre všetky  $c \in \Sigma$ :
  - 3.1. Nech  $P := \{p \in K \mid \exists q \in Q : p \in \delta(q, c)\}$ .
  - 3.2. Ak  $P \notin K'$ , polož  $K' := K' \cup \{P\}$  a  $K'_? := K'_? \cup \{P\}$ .
  - 3.3. Polož  $\delta'(Q, c) := P$ .
4. Ak  $K'_? \neq \emptyset$ , pokračuj krokom 2; inak pokračuj krokom 5.
5. Polož  $q'_0 := \{q_0\}$  a  $F' := \{Q \in K' \mid Q \cap F \neq \emptyset\}$ .

## 2 Konštrukcia regulárnej gramatiky ku konečnému automatu

Na prednáške bola dokázaná ekvivalencia (deterministických alebo nedeterministických) konečných automatov a regulárnych gramatík.

Zamerajme sa teraz na konštrukciu regulárnej gramatiky  $G = (N, T, P, \sigma)$  ekvivalentnej danému nedeterministickému konečnému automatu  $A = (K, \Sigma, \delta, q_0, F)$ , ktorú možno zhrnúť nasledovne.

1. Abeceda neterminálov gramatiky  $G$  je množina stavov automatu  $A$ , t. j.  $N = K$ .
2. Abeceda terminálov gramatiky  $G$  je identická so vstupnou abecedou automatu  $A$ , t. j.  $T = \Sigma$ .
3. Množina prepisovacích pravidiel  $P$  gramatiky  $G$  je daná takto:
  - (i) Pre každé  $p, q \in K$  a  $z \in \Sigma \cup \{\varepsilon\}$  také, že  $q \in \delta(p, z)$ , je v množine  $P$  pravidlo  $p \rightarrow zq$ .
  - (ii) Pre každé  $q \in F$  je v množine prepisovacích pravidiel  $P$  pravidlo  $q \rightarrow \varepsilon$ .
  - (iii) Množina  $P$  neobsahuje žiadne iné prepisovacie pravidlá.
4. Počiatočný neterminál gramatiky  $G$  je počiatočný stav automatu  $A$ , t. j.  $\sigma = q_0$ .

## 3 Konštrukcia konečného automatu k regulárnej gramatike

Pripomeňme si ešte opačný problém konštrukcie nedeterministického konečného automatu ekvivalentného danej regulárnej gramatike. Konštrukciu automatu  $A = (K, \Sigma, \delta, q_0, F)$  ku gramatike  $G = (N, T, P, \sigma)$  možno zhrnúť nasledovne.

1. Zostroj regulárnu gramatiku  $G' = (N', T, P', \sigma)$  takú, že  $P' \subseteq N' \times (TN' \cup N' \cup T \cup \{\varepsilon\})$  a  $L(G') = L(G)$ . Pre každé pravidlo  $\pi = (\alpha \rightarrow a_1 \dots a_k s) \in P$ , kde  $k \geq 2$ ,  $a_1, \dots, a_k \in T$  a  $s \in N \cup \{\varepsilon\}$  teda:
  - 1.1. Zaved nové neterminály  $\psi_{\pi,1}, \dots, \psi_{\pi,k-1}$ .
  - 1.2. Odober pôvodné pravidlo  $\alpha \rightarrow a_1 \dots a_k s$ .
  - 1.3. Pridaj nové pravidlá  $\alpha \rightarrow a_1 \psi_{\pi,1}, \psi_{\pi,1} \rightarrow a_2 \psi_{\pi,2}, \dots, \psi_{\pi,k-1} \rightarrow a_k s$ .
2. Polož  $K = N' \cup \{q_{\text{fin}}\}$ , kde  $q_{\text{fin}} \notin N'$ .
3. Polož  $\Sigma = T$ .
4. Prechodovú funkciu  $\delta$  automatu  $A$  definuj pre každé  $\xi \in N'$  a  $z \in T \cup \{\varepsilon\}$  nasledovne:
  - (i) Ak pre  $\eta \in N'$  je  $\xi \rightarrow z\eta \in P'$ , tak  $\eta \in \delta(\xi, z)$ .
  - (ii) Ak  $\xi \rightarrow z \in P'$ , tak  $q_{\text{fin}} \in \delta(\xi, z)$ .
  - (iii) Množina  $\delta(\xi, z)$  neobsahuje žiadne iné stavy.
- Navyše  $\delta(q_{\text{fin}}, z) = \emptyset$  pre všetky  $z \in T \cup \{\varepsilon\}$ .
5. Polož  $q_0 = \sigma$ .
6. Polož  $F = \{q_{\text{fin}}\}$ .

# Regulárne jazyky

Peter Kostolányi

2. novembra 2022

Deterministické konečné automaty, nedeterministické konečné automaty a regulárne gramatiky sú – ako už dobre vieme – rovnako silné modely opisujúce rovnakú triedu jazykov. Jazyky z tejto triedy nazývame *regulárnymi*. Na definíciu regulárnych jazykov tak môžeme použiť ľubovoľný z uvedených troch ekvivalentných modelov.

**Definícia 1.** *Regulárny jazyk* je jazyk  $L$ , pre ktorý existuje deterministický konečný automat  $A$  taký, že  $L(A) = L$ . Triedu všetkých regulárnych jazykov označujeme  $\mathcal{R}$ .

**Veta 1.** Nech  $L$  je jazyk. Nasledujúce tvrdenia sú ekvivalentné:

- (i) Jazyk  $L$  je regulárny.
- (ii) Existuje nedeterministický konečný automat  $A$  taký, že  $L(A) = L$ .
- (iii) Existuje regulárna gramatika  $G$  taká, že  $L(G) = L$ .

## 1 Pumpovacia lema pre regulárne jazyky

Nech  $L \subseteq \Sigma^*$  je ľubovoľný regulárny jazyk. Určite potom *existuje* deterministický konečný automat  $A = (K, \Sigma, \delta, q_0, F)$  taký, že  $L(A) = L$ . Ten má nejaký počet stavov  $|K| =: p$ . Nech  $w \in L$  je ľubovoľné slovo z jazyka  $L$  také, že  $|w| \geq p$ . Z Dirichletovho príncipa vyplýva, že počas výpočtu automatu  $A$  na tomto slove sa niektorý jeho stav  $q_{op}$  musí „vyskytnúť“ aspoň dvakrát. Existujú teda slová  $u, v \in \Sigma^*$  a  $x \in \Sigma^+$  také, že  $w = uxv$  a

$$(q_0, uxv) \vdash^* (q_{op}, xv) \vdash^+ (q_{op}, v) \vdash^* (q, \varepsilon),$$

kde  $q \in F$  je akceptačný stav. Nech teraz  $i \in \mathbb{N}$  je ľubovoľné prirodzené číslo. Ľahko vidieť, že existuje akceptačný výpočet automatu  $A$  na slove  $ux^i v$ : stačí namiesto jedného „cyklu“  $(q_{op}, x) \vdash^+ (q_{op}, \varepsilon)$  vykonať takýchto „cyklov“ niekoľko. Inými slovami:

$$(q_0, ux^i v) \vdash^* (q_{op}, x^i v) \vdash^+ (q_{op}, x^{i-1} v) \vdash^+ \dots \vdash^+ (q_{op}, v) \vdash^* (q, \varepsilon).$$

Pre  $i = 0$  zodpovedá uvedený zápis „vyniechaniu cyklu“. Je navyše zrejmé, že prvý opakovany výskyt nejakého stavu musí prísť po najviac  $p$  krokoch výpočtu, takže možno pridať obmedzenie  $|ux| \leq p$ .

Pomocou takýchto jednoduchých úvah možno odvodiť pumpovaciu lemu pre regulárne jazyky, pričom správne poradie kvantifikátorov je dané poradím slov, ktoré sme zvýrazňovali italicou.

**Veta 2.** Nech  $\Sigma$  je abeceda a  $L \subseteq \Sigma^*$  regulárny jazyk. Potom existuje  $p \in \mathbb{N}$  také, že pre všetky  $w \in L$  s  $|w| \geq p$  existujú slová  $u, x, v \in \Sigma^*$  také, že:

- (i)  $w = uxv$ ,
- (ii)  $|ux| \leq p$ ,
- (iii)  $|x| \geq 1$ ,
- (iv)  $\forall i \in \mathbb{N} : ux^i v \in L$ .

## 2 Uzáverové vlastnosti triedy regulárnych jazykov

Trieda jazykov  $\mathcal{L}$  je uzavretá na nejakú  $k$ -árnu operáciu  $\Phi$ , ak  $\Phi(L_1, \dots, L_k) \in \mathcal{L}$  kedykoľvek  $L_1, \dots, L_k \in \mathcal{L}$ . Napríklad trieda regulárnych jazykov  $\mathcal{R}$  je teda uzavretá na zjednotenie, pretože pre všetky  $L_1, L_2 \in \mathcal{R}$  je  $L_1 \cup L_2 \in \mathcal{R}$ .

Dokázať uzavretosť triedy regulárnych jazykov na  $k$ -árnu operáciu  $\Phi$  teda znamená pre všetky regulárne jazyky  $L_1, \dots, L_k$  ukázať, že aj jazyk  $\Phi(L_1, \dots, L_k)$  musí byť regulárny. To možno urobiť napríklad konštrukciou konečného automatu alebo regulárnej gramatiky pre jazyk  $\Phi(L_1, \dots, L_k)$ , pričom pri konštrukcii vychádzame z konečných automatov alebo regulárnych gramatík, ktorých existenciu predpokladáme pre jazyky  $L_1, \dots, L_k$ .

Vyvrátiť uzavretosť triedy  $\mathcal{R}$  na operáciu  $\Phi$  naopak znamená nájsť konkrétny príklad jazykov  $L_1, \dots, L_k \in \mathcal{R}$  takých, že  $\Phi(L_1, \dots, L_k) \notin \mathcal{R}$ .

**Definícia 2.** Nech  $\Sigma$  je abeceda a  $L_1, L_2 \subseteq \Sigma^*$  sú jazyky. Jazyk  $L_1 \sqcup L_2$  potom definujeme ako

$$L_1 \sqcup L_2 = \{u_1v_1u_2v_2 \dots u_nv_n \mid n \in \mathbb{N}; u_1, \dots, u_n, v_1, \dots, v_n \in \Sigma^*; u_1 \dots u_n \in L_1; v_1 \dots v_n \in L_2\}.$$

Do jazyka  $L_1 \sqcup L_2$  teda patria všetky slová, ktoré vzniknú z nejakého slova  $u \in L_1$  a slova  $v \in L_2$  ich „premiešaním“ zachovávajúcim relatívne poradie písmen v oboch slovách. Nejakým ľubovoľným spôsobom tieto dve slová vyjadríme ako zreťazenia  $n \in \mathbb{N}$  faktorov – položíme teda  $u = u_1u_2 \dots u_n$  a  $v = v_1v_2 \dots v_n$ , kde  $u_1, \dots, u_n \in \Sigma^*$  a  $v_1, \dots, v_n \in \Sigma^*$  sú slová. Do jazyka  $L_1 \sqcup L_2$  potom bude patriť slovo  $u_1v_1u_2v_2 \dots u_nv_n$ . Faktory  $u_1, \dots, u_n$  a  $v_1, \dots, v_n$  tu môžu byť aj prázdne, čo okrem iného znamená, že slovo z jazyka  $L_1 \sqcup L_2$  sa môže začínať aj faktorom slova z  $L_2$ .

*Priklad 1.* Nech  $L_1 = \{aa, \textcolor{blue}{ba}\}$  a  $L_2 = \{\textcolor{red}{bb}\}$ . Potom

$$L_1 = \{aab\textcolor{red}{b}, \textcolor{blue}{ab}\textcolor{red}{b}, abba, \textcolor{red}{baab}, \textcolor{blue}{bab}, \textcolor{red}{bbaa}, \textcolor{blue}{bab}, \textcolor{red}{bbab}, \textcolor{blue}{bbba}\}.$$

**Definícia 3.** Nech  $\Sigma$  je abeceda a  $L \subseteq \Sigma^*$  je jazyk. *Rotáciou* jazyka  $L$  nazveme jazyk

$$\text{rot}(L) = \{vu \mid u, v \in \Sigma^*; uv \in L\}.$$

## 2.10 Regulárne výrazy

Ukázali sme si zatiaľ dve možnosti, ako zadať ľubovoľný regulárny jazyk – konečné automaty a regulárne gramatiky.<sup>11</sup> Teraz si ukážeme tretiu možnosť – regulárne výrazy. Ich výhodou bude jednoduchší zápis v porovnaní s automatmi aj gramatikami.

**Definícia 2.10.1** *Regulárny výraz môžeme rekurzívne definovať nasledovne:*

- $\emptyset$  je regulárny výraz, predstavujúci jazyk  $\emptyset$ .
- Nech  $x \in \Sigma$ , potom  $\bar{x}$  je regulárny výraz, predstavujúci jazyk  $\{x\}$ .
- Nech  $\bar{R}_1, \bar{R}_2$  sú regulárne výrazy, predstavujúce jazyky  $L_1, L_2$ , potom  $\overline{(R_1 + R_2)}$  je regulárny výraz, predstavujúci jazyk  $L_1 \cup L_2$ .
- Nech  $\bar{R}_1, \bar{R}_2$  sú regulárne výrazy, predstavujúce jazyky  $L_1, L_2$ , potom  $\overline{R_1 R_2}$  je regulárny výraz, predstavujúci jazyk  $L_1 \cdot L_2$ .
- Nech  $\bar{R}$  je regulárny výraz, predstavujúci jazyk  $L$ , potom  $\overline{(R)^*}$  je regulárny výraz, predstavujúci jazyk  $L_1^*$ .
- Nič iné nie je regulárny výraz a žiadnen regulárny výraz nepredstavuje žiadnen iný jazyk.

**Príklad 2.10.1** Reg. výraz  $\overline{((a+b)) * baa((a+b)) *}$  predstavuje jazyk  $L = \{ubaav \mid u, v \in \{a, b\}^*\}$ .

**Veta 2.10.1** Každý regulárny výraz predstavuje regulárny jazyk.

**Dôkaz.** Zjavné – jazyky  $\emptyset$  a  $\{x\}$  (pre  $x \in \Sigma$ ) sú regulárne, regulárne jazyky sú uzavreté na zjednotenie, zreťazenie a iteráciu.

**Veta 2.10.2** Ku každému regulárnemu jazyku existuje regulárny výraz, ktorý ho predstavuje.

**Dôkaz.** Ku každému regulárnemu jazyku existuje DKA, ktorý ho akceptuje. Lahko upravíme postup z dôkazu Kleeneho vety (veta 2.7.2) tak, aby nám postupne zstrojil regulárne výrazy pre všetky jazyky  $R_{i,j}^k$ .

**Poznámka 2.10.1** Sila regulárnych výrazov spočíva v tom, že nimi vieme jednoducho popísat niektoré regulárne jazyky. Mnohé programy (obzvlášť na Unix-like platformách) preto používajú upravenú verziu regulárnych výrazov pri vyhľadávaní či nahradzovaní vzoriek. Za všetky spomeňme *grep*, *sed*, *vim* a *perl*. Popíšeme teraz, ako vyzerajú tieto upravené regulárne výrazy (regexp). Naša definícia bude jemne zjednodušená, jej cieľom nie je uvádzat všetky technické detaily ale predviesť približný vzhľad regexpov.

Ak slovo  $w$  bude patriť do jazyka predstavovaného regexpom  $R$ , budeme hovoriť, že  $R$  **matchuje** slovo  $w$ .

**Regexp** je refazec, zložený z niekoľkých **vetiev**, oddelených znakom  $|$ . **Regexp** matchuje tie slová, ktoré matchuje aspoň jedna **vetva**. (Znak  $|$  teda plní funkciu logického or, zjednotenia jazykov, resp.  $+$  z našej formálnej definície.)

**Vetva** je refazec, zložený z niekoľkých **častí**. **Vetva** matchuje práve všetky slová, ktoré sú zreťazením slov matchujúcich jednotlivé časti.

**Časť** je **atóm**, ktorý môže byť nasledovaný jedným znakom  $*$ ,  $+$  alebo  $?$ . Nech **atóm** matchuje slová z  $L$ . Potom: **Atóm** nasledovaný znakom  $*$  matchuje slová z  $L^*$ . **Atóm** nasledovaný znakom  $+$  matchuje slová z  $L^+$ . **Atóm** nasledovaný znakom  $?$  matchuje slová z  $L \cup \{\epsilon\}$ .

**Atóm** je buď  $(R)$  (kde  $R$  je opäť **regexp**), alebo znak  $x$  (ktorý matchuje refazec  $x$ ), alebo jeden zo špeciálnych znakov  $., ^, \$$ . Znak  $.$  matchuje jeden ľubovoľný znak. Znak  $^$  matchuje začiatok slova, znak  $\$$  matchuje koniec slova.

2. **Bezkontextové jazyky.** [gramatiky, normálne tvary, nedeterministické zásobníkové automaty, ekvivalencia ZA a BG, pumpovacia lema, uzáverové vlastnosti.]

## Bezkontextové gramatiky (1. časť)

Peter Kostolányi

5. októbra 2022

### 1 Bezkontextové a regulárne gramatiky

*Bezkontextové gramatiky* sú frázové gramatiky, v ktorých ľavá strana každého prepisovacieho pravidla pozostáva z jediného neterminálu. Definície najdôležitejších pojmov súvisiacich s frázovými gramatikami boli zavedené na prednáške. V rámci ich rekapitulácie sa obmedzíme výhradne na špeciálny prípad bezkontextových gramatík.

**Definícia 1.** *Bezkontextová gramatika* je štvorica  $G = (N, T, P, \sigma)$ , kde  $N$  je abeceda<sup>1</sup> neterminálnych symbolov,  $T$  je abeceda terminálnych symbolov,  $N \cap T = \emptyset$ ,  $P \subseteq N \times (N \cup T)^*$  je konečná množina prepisovacích pravidiel a  $\sigma \in N$  je počiatočný neterminál.

Prepisovacie pravidlá  $(\xi, x) \in P$  obyčajne zapisujeme ako  $\xi \rightarrow x$ . Ide iba o špeciálny spôsob zápisu usporiadanej dvojice používanej v tomto kontexte a symbol  $\rightarrow$  sám o sebe nemá definovaný žiadny význam. Zápis  $\xi \rightarrow x_1, \xi \rightarrow x_2, \dots, \xi \rightarrow x_k$  skracujeme aj ako  $\xi \rightarrow x_1 | x_2 | \dots | x_k$ .

**Definícia 2.** Nech  $G = (N, T, P, \sigma)$  je bezkontextová gramatika. *Krok odvodenia* v gramatike  $G$  je binárna relácia  $\Rightarrow_G$  na  $(N \cup T)^*$  taká, že

$$\begin{aligned} \forall u, v \in (N \cup T)^* : u \Rightarrow_G v \text{ práve vtedy, keď} \\ \exists u_1, u_2, x \in (N \cup T)^* \exists \xi \in N : u = u_1 \xi u_2 \wedge v = u_1 x u_2 \wedge \xi \rightarrow x \in P. \end{aligned}$$

V prípade, že je gramatika  $G$  zrejmá z kontextu, píšeme namiesto  $\Rightarrow_G$  iba  $\Rightarrow$ .

Kedže je krok odvodenia  $\Rightarrow$  definovaný ako binárna relácia na  $(N \cup T)^*$ , dajú sa naň aplikovať všetky bežné operácie na takýchto reláciách. Pre  $k \in \mathbb{N}$  tak napríklad môžeme hovoriť o  $k$ -tej mocnine relácie  $\Rightarrow$ , označovanej symbolom  $\Rightarrow^k$ . V relácii  $\Rightarrow^k$  sú potom všetky dvojice slov  $u, v$  také, že slovo  $v$  sa dá odvodiť zo slova  $u$  na práve  $k$  krokov odvodenia. Podobne možno definovať reflexívno-tranzitívny uzáver  $\Rightarrow^*$  a tranzitívny uzáver  $\Rightarrow^+$  relácie  $\Rightarrow$ . V týchto reláciach sú všetky dvojice slov  $u, v$  také, že slovo  $v$  sa dá odvodiť zo slova  $u$  na ľubovoľný resp. Ľubovoľný nenulový počet krokov. Relácia  $\Rightarrow^0$  je identita, podobne ako nultá mocnina ľubovoľnej inej relácie.

Zneužívajúc notáciu často hovoríme o „odvodení  $\sigma \Rightarrow w_1 \Rightarrow \dots \Rightarrow w_n$ “, o „odvodení  $\sigma \Rightarrow^n w_n$  (dlžky  $n$ )“, alebo o „odvodení  $\sigma \Rightarrow^* w_n$ “, aj keď každý z týchto zápisov vyjadruje iba reláciu medzi slovami a ľažko tak hovorí o „entite odvodenia“. Žiadnen z uvedených zápisov navyše neposkytuje kompletnejšiu informáciu, ktorú od konceptu odvodenia intuitívne očakávame. Problém formálneho uchopenia pojmu odvodenia sa však ukazuje ako pomerne delikátny a v literatúre sa často možno stretnúť s chybňami definíciami neskôr kompenzovanými intuíciou. My sa k týmto záležitostiam vrátíme na nasledujúcom cvičení, kde sa pokúsime aj o formálne presnú definíciu. Väčšinou sa ale budeme pridŕžať zaužívaných zvyklostí z úvodu tohto odstavca.

**Definícia 3.** Nech  $G = (N, T, P, \sigma)$  je bezkontextová gramatika. *Jazyk* generovaný gramatikou  $G$  je daný ako  $L(G) = \{w \in T^* \mid \sigma \Rightarrow^* w\}$ .

O slovách z jazyka  $L(G)$  tiež hovoríme, že sú *generované* gramatikou  $G$ . Z definície vyplýva, že takéto slová pozostávajú výhradne z terminálnych symbolov. Slovo nad abecedou  $N \cup T$ , pre ktoré existuje v gramatike  $G$  odvodenie, nazývame *vetnou formou* v gramatike  $G$ . Pre jazyk všetkých vetných foriem v gramatike  $G$  budeme *pre účely týchto cvičení* používať označenie  $F(G)$ .

**Definícia 4.** Nech  $G = (N, T, P, \sigma)$  je bezkontextová gramatika. *Vetná forma* v gramatike  $G$  je slovo  $x \in (N \cup T)^*$  také, že  $\sigma \Rightarrow^* x$ . Pre jazyk všetkých vetných foriem v gramatike  $G$  píšeme  $F(G) = \{x \in (N \cup T)^* \mid \sigma \Rightarrow^* x\}$ .

Jazyk nazveme *bezkontextovým*, ak existuje bezkontextová gramatika, ktorá ho generuje.

**Definícia 5.** *Bezkontextový jazyk* je jazyk  $L$  taký, že  $L = L(G)$  pre nejakú bezkontextovú gramatiku  $G$ . Triedu všetkých bezkontextových jazykov označujeme  $\mathcal{L}_{CF}$ .

Skutočnosť, že hovoríme o *triede* bezkontextových jazykov a nie o *množine* bezkontextových jazykov, nie je náhodná. Pokiaľ sa totiž neobmedzíme na nejakú konkrétnu množinu symbolov, ktoré môžu byť prvkami abecied, množina všetkých bezkontextových jazykov neexistuje. Trieda sa od množiny lisi predovšetkým tým, že sama nie je množinou, a teda existuje napríklad aj trieda všetkých množín a podobne.

*Regulárne gramatiky* sú bezkontextové gramatiky, v ktorých sa môže každý neterminál prepísat iba na niekoľko (aj nula) terminálov nasledovaných najviac jedným neterminálom.

**Definícia 6.** *Regulárna gramatika* je bezkontextová gramatika  $G = (N, T, P, \sigma)$ , pre ktorú platí  $P \subseteq N \times (T^* \cup T^* N)$ .<sup>2</sup>

Kedže sme regulárne gramatiky definovali ako špeciálne bezkontextové gramatiky, vzťahuje sa väčšina definícii a označení zavedených vyššie aj na regulárne gramatiky. V nasledujúcom sa budeme zaoberať metódami dokazovania, že daná gramatika skutočne generuje zamýšľaný jazyk; aj keď budeme hovoriť o bezkontextových gramatikách vo všeobecnosti, rovnaké metódy bude možné použiť aj pre regulárne gramatiky, ktoré sú ich špeciálnym prípadom.

**Poznámka 1.** *Lineárna gramatika* je bezkontextová gramatika  $G = (N, T, P, \sigma)$  taká, že množina pravidiel  $P$  je konečnou podmnožinou množiny  $N \times (T^* \cup T^* NT^*)$  – na pravej strane každého pravidla sa teda môže vyskytovať najviac jeden neterminál. Regulárne gramatiky sú očividne špeciálnym prípadom lineárnych gramatík a niekedy sa preto nazývajú aj *sprava lineárne gramatiky*.

**Poznámka 2.** Regulárne gramatiky sú ďalším z radu objektov, ktorých definícia nie je úplne ustálená. Kým niektorí autori napríklad týmto pojmom označujú sprava lineárne gramatiky, ktorých pravidlá nemôžu na pravej strane obsahovať viac ako jeden terminál, inde sa zas pod regulárnu gramatikou rozumie bezkontextová gramatika, ktorá je lineárna sprava *alebo* zlava – množina prepisovacích pravidiel zlava lineárnej gramatiky  $G = (N, T, P, \sigma)$  je konečnou podmnožinou množiny  $N \times (T^* \cup NT^*)$ . Neskôr pochopíme, že tieto drobné variácie v skutočnosti nie sú nijak podstatné.

## 2 Dokazovanie správnosti konštrukcie gramatiky

V úlohách nasledujúcich za týmto oddielom je väčšinou cieľom *zostrojiť* bezkontextovú gramatiku  $G$  generujúcu daný jazyk  $L$  a *dokázať správnosť* konštrukcie, čo možno vyjadriť rovnosťou  $L(G) = L$ .

Po nájdení samotnej gramatiky, ktoré zvyčajne býva jedinou skutočne tvorivou časťou celej úlohy, je teda ešte potrebné dokázať rovnosť jazyka  $L(G)$  generovaného zostrojenou gramatikou  $G$  a jazyka  $L$  zo zadania úlohy.

Dokázať rovnosť  $L(G) = L$  znamená overiť platnosť obidvoch inkluzií  $L(G) \subseteq L$  a  $L(G) \supseteq L$ .

Pri dôkaze inkluzie  $L(G) \subseteq L$  pre bezkontextovú gramatiku  $G = (N, T, P, \sigma)$  uvažujeme ľuboľné slovo  $w \in L(G)$  a ukazujeme, že toto slovo patrí do jazyka  $L$ . Predpoklad  $w \in L(G)$  znamená, že  $w \in T^*$  a súčasne  $\sigma \Rightarrow^* w$ , v dôsledku čoho  $\sigma \Rightarrow^n w$  pre nejaké  $n \in \mathbb{N}$ . O slove  $w$  v tomto momente nevieme nič ďalšie a potrebujeme ukázať, že  $w \in L$ . Ako dôkazová metóda sa teda ponúka matematická indukcia vzhľadom na dĺžku  $n$  uvažovaného odvodenia slova  $w$ .

V rámci indukčného kroku takého dôkazu predpokladáme platnosť tvrdenia pre  $n = k$  a dokazujeme jeho platnosť pre  $n = k + 1$ . Ak teda  $\sigma \Rightarrow^{k+1} w$ , môžeme sa pokúsiť prepísať toto odvodenie napríklad ako

$$\sigma \Rightarrow^k y \Rightarrow w$$

a aplikovať indukčný predpoklad na odvodenie  $\sigma \Rightarrow^k y$ . Tu však narázame: indukčný predpoklad nám o slove  $y$  nehovorí nič, pretože slovo  $y$  nepatrí do jazyka  $L(G)$ ; ide iba o *vetnú formu* v gramatike  $G$ , ktorá nutne obsahuje aspoň jeden neterminál. Oplatí sa teda dokazované tvrdenie *zosilniť* tak, aby sme do úvah zahrnuli všetky vetné formy. Napríklad môžeme nájsť jazyk  $F$  všetkých vetných foriem v gramatike  $G$  a namiesto inkluzie  $L(G) \subseteq L$  dokazovať inkluziu  $F(G) \subseteq F$ .

*Pri dôkaze inkluzie  $L(G) \supseteq L$  naopak uvažujeme ľubovoľné slovo  $w \in L$  a ukazujeme, že pre toto slovo v gramatike  $G$  existuje odvodenie. Obvykle tak robíme indukciou vzhľadom na nejakú štrukturálnu vlastnosť slova  $w$ , ktorú volíme na základe jazyka  $L$  a skonštruovanej gramatiky  $G$  pre tento jazyk. Ak napríklad  $L = \{a, b\}^*$ , je prirodzené tvrdenie dokazovať indukciou na dĺžku slova; pri jazyku  $L = \{a^n b^n \mid n \in \mathbb{N}\}$  sa ponúka dôkaz indukciou vzhľadom na  $n$  a napríklad pri jazyku  $L = \{abu \mid u \in \{a, b\}^*\}$  je prirodzeným prístupom indukcia vzhľadom na dĺžku slova  $u$ . Často je potrebných aj viacero indukcií, ako čoskoro uvidíme na príklade.*

Nech ale túto štrukturálnu vlastnosť zvolíme akokoľvek, je často žiaduce vedieť na odvodeľnosť slova v gramatike usúdiť z indukčného predpokladu o odvodeľnosti nejakého iného slova. Preto môže byť aj tu užitočné dokazované tvrdenie *zosilniť*. Opäť môžeme napríklad identifikovať jazyk  $F$  všetkých vetných foriem v  $G$  a namiesto inkluzie  $L(G) \supseteq L$  dokázať inkluziu  $F(G) \supseteq F$ .

Rovnosť  $L(G) = L$  pre bezkontextovú gramatiku  $G = (N, T, P, \sigma)$  tak typicky môžeme dokázať nasledovne.

1. Identifikujeme jazyk  $F$  všetkých vetných foriem v gramatike  $G$ ; preň by, samozrejme, vždy malo platit  $F \cap T^* = L$ .
2. Namiesto inkluzie  $L(G) \subseteq L$  dokážeme silnejšie tvrdenie  $F(G) \subseteq F$  – čiže „ak  $x \in (N \cup T)^*$  a  $\sigma \Rightarrow^* x$ , tak  $x \in F$ “. Obyčajne tak robíme indukciou vzhľadom na dĺžku uvažovaného odvodenia slova  $x$ .
3. Namiesto inkluzie  $L(G) \supseteq L$  dokážeme silnejšie tvrdenie  $F(G) \supseteq F$  – čiže „ak  $x \in F$ , tak  $\sigma \Rightarrow^* x$ “. Obyčajne tak robíme indukciou vzhľadom na určité štrukturálne vlastnosti slova  $x$ .

Uvedený postup je pri konštrukcii gramatík pre „učebnicové“ bezkontextové jazyky použiteľný skoro univerzálne, ale občas môže byť zbytočne prácny. V rámci nasledujúcich riešených úloh si preto okrem príkladov použitia tohto postupu ukážeme aj ďalšie možnosti dokazovania správnosti konštrukcie bezkontextových gramatík.

**Tvrdenie 1.** Nech  $G = (N, T, P, \sigma)$  je bezkontextová gramatika,  $\xi_1, \dots, \xi_k \in N$ ,  $u_0, \dots, u_k \in T^*$  a  $n \in \mathbb{N}$ . Pre slovo  $w \in (N \cup T)^*$  je potom  $u_0 \xi_1 u_1 \xi_2 u_2 \dots u_{k-1} \xi_k u_k \Rightarrow^n w$  práve vtedy, keď existujú  $x_1, \dots, x_k \in (N \cup T)^*$  a  $n_1, \dots, n_k \in \mathbb{N}$  také, že  $w = u_0 x_1 u_1 x_2 u_2 \dots u_{k-1} x_k u_k$ ,  $n_1 + \dots + n_k = n$  a pre  $i = 1, \dots, k$  je  $\xi_i \Rightarrow^{n_i} x_i$ .

V dôsledku toho  $u_0 \xi_1 u_1 \xi_2 u_2 \dots u_{k-1} \xi_k u_k \Rightarrow^* w$  práve vtedy, keď existujú  $x_1, \dots, x_k \in (N \cup T)^*$  také, že  $w = u_0 x_1 u_1 x_2 u_2 \dots u_{k-1} x_k u_k$  a pre  $i = 1, \dots, k$  je  $\xi_i \Rightarrow^* x_i$ .

Opísaný prístup teraz v rámci riešenia nasledujúcej úlohy demonštrujeme na ďalšej gramatike  $G$  generujúcej jazyk  $L = \{w \in \{a, b\}^* \mid \#_a(w) = \#_b(w)\}$ . Inkluziu  $L(G) \subseteq L$  by sme aj pre túto gramatiku mohli dokázať prakticky rovnako ako v úlohe 3; výhodnosť alternatívneho prístupu sa však naplno ukáže pri dôkaze opačnej inkluzie.

## 1 Normálne tvary bezkontextových gramatík

Vetou o normálnom tvari bezkontextových gramatík možno nazvať prakticky ľubovoľné tvrdenie, podľa ktorého ku každej alebo „skoro každej“ bezkontextovej gramatike  $G$  existuje ekvivalentná alebo „takmer ekvivalentná“ gramatika  $G'$  s nejakou špeciálnou vlastnosťou.

**Poznámka 1.** Normálne tvary možno, samozrejme, študovať aj pre iné objekty ako gramatiky. Jediným predpokladom je, aby na danej triede objektov  $X$  bola definovaná relácia ekvivalencie  $\sim$ . Veta o normálnom tvari potom vždy pre nejakú triedu  $Y \subseteq X$  hovorí, že pre každé  $x \in X$  existuje  $y \in Y$  také, že  $y \sim x$ .

V našom prípade je touto triedou  $X$  trieda všetkých bezkontextových gramatík, pričom pre dvojicu bezkontextových gramatík  $G, G'$  je  $G \sim G'$  práve vtedy, keď  $L(G) = L(G')$ .

Normálne tvary bezkontextových gramatík je užitočné skúmať predovšetkým z nasledujúcich dvoch dôvodov.

- Predpoklad normálneho tvaru môže často uľahčiť dôkazy tvrdení o bezkontextových jazykoch. Ak totiž namiesto všeobecných bezkontextových gramatík uvažujeme iba gramatiky v normálnom tvari, môže sa podstatne zjednodušiť argumentácia. Dôležitá je pritom predovšetkým skutočnosť, že ekvivalentná gramatika v normálnom tvari vždy *existuje*; algoritmus na prevod do normálneho tvaru je tu menej podstatný, avšak zaručuje konštruktívnosť dôkazu.
- Viaceré užitočné algoritmy pracujúce s bezkontextovými gramatikami<sup>1</sup> predpokladajú ako svoj vstup gramatiku v nejakom normálnom tvari. Pri implementácii takýchto algoritmov sa preto môže zísť aj procedúra na prevod do požadovaného normálneho tvaru.

V nasledujúcom sa zameriame na praktickú stránku prevodu gramatík do jednotlivých normálnych tvarov. Matematické zápisť týchto konštrukcií a dôkazy ich správnosti odzneli na prednáške.

## 2 Redukovaný normálny tvar

Bezkontextová gramatika je *redukovaná* alebo v *redukovanom normálnom tvari*, ak neobsahuje žiadne pravidlo typu  $\xi \rightarrow \xi$  a každý jej neterminál  $\xi$  je súčasne „terminujúci“ (teda existuje aspoň jedno terminálne slovo odvodeľné zo  $\xi$ ) a dosiahnuteľný (v danej gramatike teda existuje vettá forma obsahujúca  $\xi$  – tá musí byť z definície odvodeľná z počiatočného neterminálu).

**Definícia 1.** Bezkontextová gramatika  $G = (N, T, P, \sigma)$  je v *redukovanom normálnom tvari*, ak sú splnené nasledujúce tri podmienky:

- (i) Množina pravidiel  $P$  neobsahuje pre žiadne  $\xi \in N$  pravidlo  $\xi \rightarrow \xi$ .
- (ii) Pre každé  $\xi \in N$  existuje terminálne slovo  $w \in T^*$  také, že  $\xi \Rightarrow^* w$ .
- (iii) Pre každé  $\xi \in N$  existujú slová  $u, v \in (N \cup T)^*$  také, že  $\sigma \Rightarrow^* u\xi v$ .

O *normálnom tvari* tu môžeme hovoriť vďaka nasledujúcej vete z prednášky.

**Veta 1.** Nech  $G$  je bezkontextová gramatika taká, že  $L(G) \neq \emptyset$ . Potom existuje bezkontextová gramatika  $G'$  v redukovanom normálnom tvari taká, že  $L(G') = L(G)$ .

Predpoklad  $L(G) \neq \emptyset$  je v predchádzajúcej vete naozaj nutný. Každá bezkontextová gramatika  $G$  totiž musí mať počiatočný neterminál  $\sigma$ . Pre redukovanú gramatiku  $G$  tak podľa podmienky (ii) definície 1 existuje terminálne slovo  $w$  také, že  $\sigma \Rightarrow^* w$ . Potom ale nutne  $w \in L(G)$ .

Každá z podmienok (i) až (iii) definície 1 sama o sebe určuje normálny tvar bezkontextových gramatík. V nasledujúcom najprv zosumarizujeme algoritmy na prevod gramatík do týchto „pomocných“ normálnych tvarov a následne ukážeme, že použitie všetkých troch týchto algoritmov vo vhodnom poradí garantuje ako výstup gramatiku v redukovanom normálnom tvare.

## 2.1 Odstránenie pravidiel typu $\xi \rightarrow \xi$

Prevod bezkontextovej gramatiky do normálneho tvaru daného podmienkou (i) definície 1 spočíva jednoducho vo „vypustení“ všetkých pravidiel typu  $\xi \rightarrow \xi$  z množiny prepisovacích pravidiel  $P$ .

## 2.2 Odstránenie „neterminujúcich“ neterminálov

Prevod bezkontextovej gramatiky do normálneho tvaru daného podmienkou (ii) definície 1 spočíva v nájdení množiny  $S$  všetkých „terminujúcich“ neterminálov gramatiky a v následnom odstránení zvyšných – čiže „neterminujúcich“ – neterminálov, ako aj všetkých pravidiel, v ktorých sa takéto neterminály vyskytujú. Množinu  $S$  možno pre bezkontextovú gramatiku  $G = (N, T, P, \sigma)$  formálne zapísť ako  $S = \{\xi \in N \mid \exists w \in T^*: \xi \Rightarrow^* w\}$  a dá sa nájsť nasledujúcim algoritmom.

1. Nech  $S_0 = \{\xi \in N \mid \exists u \in T^*: \xi \rightarrow u \in P\}$ .
2. Pre  $i = 1, 2, \dots$  iteruj predpis  $S_i = S_{i-1} \cup \{\xi \in N \mid \exists u \in (T \cup S_{i-1})^*: \xi \rightarrow u \in P\}$ , až kým pre nejaké  $k$  nenastane  $S_k = S_{k-1}$ .
3. Polož  $S = S_k$ .

Množina  $S_0$  teda pozostáva z práve tých neterminálov  $\xi$ , ktoré možno pomocou nejakého pravidla prepísať na rýdzo terminálne slovo. Pre  $i = 1, \dots, k$  následne množina  $S_i$  obsahuje všetky neterminály z množiny  $S_{i-1}$ , ako aj tie neterminály  $\xi$ , ktoré možno pomocou nejakého pravidla prepísať na slovo obsahujúce iba terminály a neterminály z  $S_{i-1}$ . Akonáhle týmto spôsobom nepribudnú žiadne nové neterminály, možno celú iteratívnu konštrukciu zastaviť.

**Poznámka 2.** Uvedený algoritmus sa na každom vstupe zastaví, pretože v kroku 2 sa do množiny  $S_i$  v každej iterácii okrem poslednej pridá oproti množine  $S_{i-1}$  aspoň jeden neterminál. Celkový počet neterminálov je ale konečný.

Skutočnosť, že množina  $S_k$  je naozaj hľadaná množina „terminujúcich“ neterminálov, bola dokázaná na prednáške.

**Poznámka 3.** V kroku 2 horeuvedeného algoritmu by v skutočnosti bolo možné iterovať iba predpis  $S_i = \{\xi \in N \mid \exists u \in (T \cup S_{i-1})^*: \xi \rightarrow u \in P\}$ . Lahko totiž vidieť, že táto množina musí obsahovať okrem iného aj všetky neterminály z  $S_{i-1}$ .

Pre gramatiku  $G = (N, T, P, \sigma)$  takú, že  $L(G) \neq \emptyset$ , je nájdená množina  $S$  vždy neprázdna. V takom prípade môžeme z gramatiky  $G$  odstrániť všetky neterminály z  $N - S$ , ako aj všetky pravidlá z  $P$ , ktorých ľavá alebo pravá strana obsahuje neterminál z  $N - S$ . Vo výsledku dostaneme gramatiku spĺňajúcu podmienku (ii) definície 1.

## 2.3 Odstránenie nedosiahnutelných neterminálov

Prevod bezkontextovej gramatiky do normálneho tvaru daného podmienkou (iii) definície 1 spočíva v nájdení množiny  $H$  všetkých dosiahnutelných neterminálov a v následnom odstránení zvyšných – čiže nedosiahnutelných – neterminálov a prepisovacích pravidiel obsahujúcich nedosiahnutelné neterminály. Množinu  $H$  možno pre bezkontextovú gramatiku  $G = (N, T, P, \sigma)$  formálne zapísť ako  $H = \{\xi \in N \mid \exists u, v \in (N \cup T)^*: \sigma \Rightarrow^* u\xi v\}$  a dá sa nájsť nasledujúcim algoritmom.

1. Nech  $H_0 = \{\sigma\}$ .
2. Pre  $i = 1, 2, \dots$  iteruj  $H_i = H_{i-1} \cup \{\xi \in N \mid \exists \eta \in H_{i-1} \exists u, v \in (N \cup T)^*: \eta \rightarrow u\xi v \in P\}$ , až kým pre nejaké  $k$  nenastane  $H_k = H_{k-1}$ .
3. Polož  $H = H_k$ .

Množina  $H_0$  teda obsahuje iba počiatočný neterminál, ktorý je ako jediný dosiahnutelný na nula krokov. Pre  $i = 1, \dots, k$  následne množina  $H_i$  obsahuje všetky neterminály z  $H_{i-1}$ , ako aj neterminály, ktoré sa vyskytujú na pravej strane niektorého pravidla s ľavou stranou v  $H_{i-1}$ ; ide tak o neterminály dosiahnutelné na najviac  $i$  krokov. Akonáhle takto nepribudnú žiadne nové neterminály, možno iteratívnu konštrukciu ukončiť.

**Poznámka 4.** Zastavenie algoritmu na každom vstupe je zrejmé z rovnakých dôvodov, ako pri hľadaní „terminujúcich“ neterminálov. Správnosť algoritmu bola dokázaná na prednáške.

**Poznámka 5.** V kroku 2 je tentokrát zjednotenie s množinou  $H_{i-1}$  naozaj podstatné.

Množina  $H$  nájdená pre gramatiku  $G = (N, T, P, \sigma)$  je vždy neprázdna. Možno teda z gramatiky  $G$  odstrániť všetky neterminály z  $N - H$  a podobne aj všetky pravidlá z  $P$ , ktorých ľavá alebo pravá strana obsahuje neterminál z  $N - H$ . Výsledkom tohto procesu je gramatika splňajúca podmienku (ii) definície 1.

## 2.4 Redukovaný normálny tvar: algoritmus

Ľubovoľnú bezkontextovú gramatiku možno previesť do redukovaného normálneho tvaru použitím nasledujúceho algoritmu.

1. Odstráň pravidlá typu  $\xi \rightarrow \xi$ .
2. Odstráň „neterminujúce“ neterminály (a pravidlá, ktoré ich obsahujú).
3. Odstráň nedosiahnutelné neterminály (a pravidlá, ktoré ich obsahujú).

Správnosť uvedeného algoritmu je založená na nasledujúcich dvoch jednoduchých pozorovaniach.

- a) Po odstránení „neterminujúcich“ a nedosiahnutelných neterminálov algoritmami z predchádzajúcich pododdielov nemôže vzniknúť žiadne nové pravidlo typu  $\xi \rightarrow \xi$ .
- b) Po odstránení nedosiahnutelných neterminálov algoritmom z pododdielu 2.3 nemôže vzniknúť žiadnenový „neterminujúci“ neterminál.

Dôkaz obidvoch týchto tvrdení je priamočiary a prenechávame ho čitateľovi.

**Poznámka 6.** Poradie odstraňovania „neterminujúcich“ a nedosiahnutelných neterminálov vo všeobecnosti nemožno vymeniť. Uvažujme napríklad gramatiku  $G = (N, T, P, \sigma)$  s  $N = \{\sigma, \alpha, \beta\}$ ,  $T = \{a\}$  a  $P = \{\sigma \rightarrow \alpha \mid a, \alpha \rightarrow \alpha\beta, \beta \rightarrow a\}$ . Táto neobsahuje žiadne pravidlo typu  $\xi \rightarrow \xi$  – odstraňovaním takýchto pravidiel sa teda zapodievať nemusíme.

Použitím horeuvedeného algoritmu dostávame redukovanú gramatiku  $G' = (N', T', P', \sigma')$ , kde  $N' = \{\sigma\}$ ,  $T' = T$ ,  $P' = \{\sigma \rightarrow a\}$  a  $\sigma' = \sigma$ .

Skúsme teraz aplikovať opačný postup a uvažujme najprv dosiahnutenosť neterminálov. Dostávame  $H_0 = \{\sigma\}$ ,  $H_1 = \{\sigma, \alpha\}$  a  $H_2 = \{\sigma, \alpha, \beta\} = H_3 = H$  – všetky neterminály sú teda dosiahnutelné. Zamerajme sa teraz na „terminujúce“ neterminály. Dostávame  $S_0 = \{\sigma, \beta\} = S_1 = S$ . Na konci celého procesu teda prichádzame ku gramatike s množinou neterminálov  $N' = \{\sigma, \beta\}$  a množinou pravidiel  $P' = \{\sigma \rightarrow a, \beta \rightarrow a\}$ . Tá celkom očividne nie je v redukovanom normálnom tvere. Po odstránení „neterminujúcich“ neterminálov sa totiž môžu niektoré pôvodne dosiahnutelné neterminály stať nedosiahnutelnými.

Odstraňovanie pravidiel typu  $\xi \rightarrow \xi$  naopak možno robiť aj uprostred alebo na konci celého algoritmu, ako by čitateľ istotne ľahko dokázal.

### 3 Chomského normálny tvar

*Chomského normálny tvar* umožňuje bez ujmy na všeobecnosti predpokladať, že sa v každom kroku odvodenia bezkontextovej gramatiky prepíše niektorý neterminál na dva neterminály, na jeden terminál, alebo na prázdne slovo.

**Definícia 2.** Nech  $G = (N, T, P, \sigma)$  je bezkontextová gramatika. Gramatika  $G$  je v *Chomského normálnom tvere*, ak  $P \subseteq N \times (N^2 \cup T \cup \{\varepsilon\})$ .

**Veta 2.** Nech  $G$  je bezkontextová gramatika. Potom existuje bezkontextová gramatika  $G'$  v Chomského normálnom tvere taká, že  $L(G') = L(G)$ .

Vstupom nasledujúceho algoritmu na prevod do Chomského normálneho tvaru je ľubovoľná bezkontextová gramatika  $G = (N, T, P, \sigma)$ .

1. Pre každý terminál  $c \in T$  zaved nový<sup>2</sup> neterminál  $\xi_c$  a pravidlo  $\xi_c \rightarrow c$ .
2. Nahrad vo všetkých pravidlach  $\alpha \rightarrow x \in P$  – teda v tých pravidlach, ktoré sa vyskytovali aj v pôvodnej gramatike  $G$  – všetky terminály  $c \in T$  neterminálom  $\xi_c$ . Nech  $G_1 = (N_1, T, P_1, \sigma)$  je výsledná gramatika.  
Pravá strana každého pravidla gramatiky  $G_1$  pozostáva z jedného terminálu, z prázdnego slova, alebo z neprázdnego slova zloženého iba z neterminálov. V nasledujúcich krokoch už iba zabezpečíme, aby v prípade neprázdnego slova zloženého z neterminálov bola jeho dĺžka rovná dvom.
3. Zaved nový neterminál  $\xi_\varepsilon$  a pravidlo  $\xi_\varepsilon \rightarrow \varepsilon$ . Všetky pravidlá  $\alpha \rightarrow \beta \in P_1$ , kde  $\alpha, \beta \in N_1$ , nahrad pravidlom  $\alpha \rightarrow \beta \xi_\varepsilon$ . Nech  $G_2 = (N_2, T, P_2, \sigma)$  je výsledná gramatika.

Gramatika  $G_2$  obsahuje iba prepisovacie pravidlá, ktoré majú na pravej strane jeden terminál, prázdroj slovo, alebo slovo dĺžky aspoň dva pozostávajúce iba z neterminálov. V záverečnom kroku teda stačí ošetriť „priľiš dlhé“ slová na pravej strane.

4. Pre každé pravidlo  $\pi = (\alpha \rightarrow \beta_1 \beta_2 \dots \beta_k) \in P_2$  s  $k > 2$  a  $\alpha, \beta_1, \dots, \beta_k \in N_2$ :
  - 4.1 Zaved nové neterminály  $\psi_{\pi,1}, \dots, \psi_{\pi,k-2}$ .
  - 4.2 Odober pôvodné pravidlo  $\alpha \rightarrow \beta_1 \beta_2 \dots \beta_k$ .
  - 4.3 Pridaj nové pravidlá  $\alpha \rightarrow \beta_1 \psi_{\pi,1}, \psi_{\pi,1} \rightarrow \beta_2 \psi_{\pi,2}, \dots, \psi_{\pi,k-2} \rightarrow \beta_{k-1} \beta_{k-2}$ .

Nech  $G' = (N', T, P', \sigma)$  je výsledná gramatika.

Výsledná gramatika  $G'$  je evidentne vždy v Chomského normálnom tvere a na prednáške bolo dokázané, že  $L(G') = L(G)$ .

**Poznámka 7.** Uvedený algoritmus nie je optimálny: napríklad všetky pravidlá  $\alpha \rightarrow c$ ,  $c \in T$ , pôvodnej gramatiky (ktoré Chomského normálny tvar neporušujú) sa nahradia trojicou pravidiel  $\alpha \rightarrow \xi_c \xi_\varepsilon$ ,  $\xi_c \rightarrow c$  a  $\xi_\varepsilon \rightarrow \varepsilon$ . Poznamenajme však, že toto neplatí pre pravidlá  $\alpha \rightarrow \varepsilon$  pôvodnej gramatiky, ktoré sú vo výsledku nezmenené.

## 4 „Bezepsilonový“ normálny tvar

Bezkontextová gramatika je v „bezepsilonovom“ normálnom tvere, ak neobsahuje vymazávajúce pravidlá typu  $\xi \rightarrow \varepsilon$ . V tomto prípade ide o normálny tvar len pre gramatiky generujúce neprázdné slová. Gramatika bez pravidiel typu  $\xi \rightarrow \varepsilon$  totiž nikdy nemôže vygenerovať prázdne slovo  $\varepsilon$ , a teda nie každá bezkontextová gramatika sa dá do „bezepsilonového“ tvaru v pravom slova zmysle previesť. Ako však bolo dokázané na prednáške, ku každej bezkontextovej gramatike  $G$  existuje bezkontextová gramatika  $G'$  v „bezepsilonovom“ normálnom tvere taká, že  $L(G') = L(G) - \{\varepsilon\}$ .

**Definícia 3.** Nech  $G = (N, T, P, \sigma)$  je bezkontextová gramatika. Hovoríme, že gramatika  $G$  je v „bezepsilonovom“ normálnom tvere, ak  $P \subseteq N \times (N \cup T)^+$ .

**Veta 3.** Nech  $G = (N, T, P, \sigma)$  je bezkontextová gramatika. Potom existuje bezkontextová gramatika  $G' = (N', T', P', \sigma')$  v „bezepsilonovom“ normálnom tvere taká, že  $L(G') = L(G) - \{\varepsilon\}$ .

Správnosť nasledujúceho algoritmu na prevod gramatiky do „bezepsilonového“ normálneho tvaru bola dokázaná na prednáške. Jeho vstupom je bezkontextová gramatika  $G = (N, T, P, \sigma)$  a výstupom je bezkontextová gramatika  $G' = (N', T', P', \sigma')$  v „bezepsilonovom“ normálnom tvere taká, že  $L(G') = L(G) - \{\varepsilon\}$ .

1. Nájdi množinu  $E = \{\xi \in N \mid \xi \Rightarrow^* \varepsilon\}$  vymazávajúcich neterminálov v gramatike  $G$ .
  - 1.1 Nech  $E_0 = \{\xi \in N \mid \xi \rightarrow \varepsilon \in P\}$  (ide o neterminály „vymazávajúce na jeden krok“).
  - 1.2 Pre  $i = 1, 2, \dots$  iteruj  $E_i = E_{i-1} \cup \{\xi \in N \mid \exists u \in E_{i-1}^* : \xi \rightarrow u \in P\}$ , až kým pre nejaké  $k$  nenastane  $E_k = E_{k-1}$ .
  - 1.3 Polož  $E = E_k$ .
2. Pre každé pôvodné pravidlo  $\xi \rightarrow u \in P$  s  $\xi \in N$  a  $u \in (N \cup T)^+$  pridaj do množiny prepisovacích pravidiel všetky pravidlá  $\xi \rightarrow v_0 v_1 \dots v_j$  s  $j \in \mathbb{N}$  a  $v_0, \dots, v_j \in (N \cup T)^*$  také, že pre nejaké  $\alpha_1, \dots, \alpha_j \in E$  je  $u = v_0 \alpha_1 v_1 \alpha_2 \dots v_{j-1} \alpha_j v_j$ . (Čiže pridáme všetky pravidlá, ktoré vzniknú z pravidla  $\xi \rightarrow u$  vypustením niektorých výskytov vymazávajúcich neterminálov v slove  $u$ .)
3. Odober z množiny prepisovacích pravidiel všetky pravidlá  $\xi \rightarrow \varepsilon$ , kde  $\xi \in N$ . Vráť výslednú gramatiku  $G' = (N', T', P', \sigma')$  ako výstup.

V kroku 2 by, samozrejme, bolo možné pridať iba také pravidlá  $\xi \rightarrow v_0 v_1 \dots v_j$ , pre ktoré  $v_0 v_1 \dots v_j \neq \varepsilon$ . V opačnom prípade sa totiž pravidlo hneď v nasledujúcom kroku z gramatiky odstráni.

# 1 Zásobníkové automaty a ich akceptačné módy

Pod *zásobníkovým automatom* – bez ďalšieho prílastku – sa obyčajne rozumie *nedeterministický zásobníkový automat*. Deterministický variant zásobníkových automatov sa zvykne preberať v rámci pokračovania tohto predmetu v letnom semestri.

**Definícia 1.** *Zásobníkový automat* je sedmica  $A = (K, \Sigma, \Gamma, \delta, q_0, Z_0, F)$ , kde  $K$  je konečná množina stavov,  $\Sigma$  je vstupná abeceda,  $\Gamma$  je abeceda zásobníkových symbolov,  $\delta: K \times (\Sigma \cup \{\varepsilon\}) \times \Gamma \rightarrow 2_{\text{kon}}^{K \times \Gamma^*}$  je prechodová funkcia,  $q_0 \in K$  je počiatočný stav,  $Z_0 \in \Gamma$  je počiatočný zásobníkový symbol a  $F \subseteq K$  je množina koncových (alebo akceptačných) stavov.

Ako  $2_{\text{kon}}^{K \times \Gamma^*}$  označujeme množinu všetkých *konečných* podmnožín množiny  $K \times \Gamma^*$ . Obmedzenie sa na konečné podmnožiny je nutné v záujme zachovania konečnosti opisu zásobníkových automatov. Čitateľ sa tiež ľahko presvedčí o tom, že tolerovanie nekonečných podmnožín by malo za následok neúmerný nárast výpočtovej sily zásobníkových automatov, ktoré by takto boli schopné jednoduchým spôsobom akceptovať ľubovoľný jazyk.

**Definícia 2.** Nech  $A = (K, \Sigma, \Gamma, \delta, q_0, Z_0, F)$  je zásobníkový automat. *Konfigurácia* zásobníkového automatu  $A$  je trojica  $(q, w, \gamma)$ , kde  $q \in K$  je stav,  $w \in \Sigma^*$  je slovo nad abecedou vstupných symbolov (reprezentujúce nedočítanú časť vstupného slova) a  $\gamma \in \Gamma^*$  je slovo nad abecedou zásobníkových symbolov (reprezentujúce obsah zásobníka s dnom naľavo a vrchom napravo).

**Poznámka 1.** V literatúre sa obsah zásobníka občas zapisuje aj opačne – t. j. s dnom napravo a vrchom naľavo.

**Definícia 3.** Nech  $A = (K, \Sigma, \Gamma, \delta, q_0, Z_0, F)$  je zásobníkový automat. *Krok výpočtu* automatu  $A$  je binárna relácia  $\vdash_A$  na množine konfigurácií automatu  $A$  taká, že pre všetky  $p, q \in K$ ,  $u, v \in \Sigma^*$  a  $\gamma_1, \gamma_2 \in \Gamma^*$  je  $(p, u, \gamma_1) \vdash_A (q, v, \gamma_2)$  práve vtedy, keď existujú  $z \in \Sigma \cup \{\varepsilon\}$ ,  $\gamma, \beta \in \Gamma^*$  a  $Z \in \Gamma$  také, že  $u = zv$ ,  $\gamma_1 = \gamma Z$ ,  $\gamma_2 = \gamma \beta$  a  $(q, \beta) \in \delta(p, z, Z)$ .

V prípade, že je uvažovaný automat  $A$  zrejmý z kontextu, často pre krok jeho výpočtu píšeme namiesto  $\vdash_A$  len  $\vdash$ .

**Definícia 4.** *Jazyk* akceptovaný zásobníkovým automatom  $A = (K, \Sigma, \Gamma, \delta, q_0, Z_0, F)$  *stavom* je daný ako

$$L(A) = \{w \in \Sigma^* \mid \exists q \in F \ \exists \gamma \in \Gamma^* : (q_0, w, Z_0) \vdash^* (q, \varepsilon, \gamma)\}.$$

**Definícia 5.** *Jazyk* akceptovaný zásobníkovým automatom  $A = (K, \Sigma, \Gamma, \delta, q_0, Z_0, F)$  *prázdnym zásobníkom* je daný ako

$$N(A) = \{w \in \Sigma^* \mid \exists q \in K : (q_0, w, Z_0) \vdash^* (q, \varepsilon, \varepsilon)\}.$$

Všimnime si, že definícia jazyka  $N(A)$  nezávisí od množiny koncových stavov  $F$ . Pri automatoch konštruovaných na akceptáciu prázdnym zásobníkom tak často kladieme  $F = \emptyset$ .

### 3 Konštrukcia zásobníkového automatu k bezkontextovej gramatike

Na prednáške bola dokázaná ekvivalencia zásobníkových automatov a bezkontextových gramatík. Trieda všetkých jazykov akceptovaných zásobníkovými automatmi je teda rovná triede  $\mathcal{L}_{CF}$  všetkých bezkontextových jazykov.

V nasledujúcom si zopakujeme štandardnú konštrukciu zásobníkového automatu ekvivalentného danej bezkontextovej gramatike.

**Tvrdenie 3.** *Nech  $G = (N, T, P, \sigma)$  je bezkontextová gramatika. Potom existuje zásobníkový automat  $A = (K, \Sigma, \Gamma, \delta, q_0, Z_0, F)$  taký, že  $N(A) = L(G)$ .*

Automat  $A$  ekvivalentný gramatike  $G$  na zásobníku simuluje ľavé krajné odvodenie gramatiky  $G$ , príčom obsah zásobníka čítaný „zhora nadol“ – teda reverz obsahu zásobníka pri zvyčajnom zápise v konfiguráciach – reprezentuje vždy určitý sufix vetnej formy.

Na začiatku výpočtu je na zásobníku iba počiatočný neterminál  $\sigma$ . Ak sa niekedy v priebehu výpočtu ocitne na vrchu zásobníka neterminál, prepíše sa tento pomocou niektorého z pravidiel a zo vstupu sa pritom neprečíta nič – to zodpovedá simulácii jedného kroku odvodenia gramatiky  $G$ . Ak je naopak na vrchu zásobníka terminál, automat nemôže priamo simulovať krok odvodenia gramatiky  $G$ . Daný terminálny symbol sa už však počas odvodenia meniť nebude, a preto ho môže automat konfrontovať so svojím vstupom. Ak je symbol na vstupe rovnaký, automat ho prečíta a zmaže vrch zásobníka. V opačnom prípade sa automat zasekne. Vyprázdenie zásobníka zodpovedá vygenerovaniu terminálneho slova gramatikou  $G$ . V prípade, že toto slovo zodpovedá kompletnému vstupu, automat tento vstup akceptuje prázdnnym zásobníkom. Na konštrukciu takéhoto automatu zjavne stačí jediný stav.

Formálne teda môžeme konštrukciu automatu  $A = (K, \Sigma, \Gamma, \delta, q_0, Z_0, F)$  ekvivalentného gramatike  $G = (N, T, P, \sigma)$  zapísť nasledovne:  $K = \{q_0\}$ ,  $\Sigma = T$ ,  $\Gamma = N \cup T$ ,  $Z_0 = \sigma$ ,  $F = \emptyset$  a

$$\begin{aligned}\delta(q_0, \varepsilon, \xi) &= \{(q_0, x^R) \mid \xi \rightarrow x \in P\} & \forall \xi \in N, \\ \delta(q_0, c, c) &= \{(q_0, \varepsilon)\} & \forall c \in T,\end{aligned}$$

pričom pre všetky ostatné trojice  $(q, z, Z) \in K \times (\Sigma \cup \{\varepsilon\}) \times \Gamma$  je  $\delta(q, z, Z) = \emptyset$ .

### 4 Konštrukcia bezkontextovej gramatiky k zásobníkovému automatu

Vyššie sme načali problematiku ekvivalence zásobníkových automatov s bezkontextovými gramatikami a rozobili sme konštrukciu zásobníkového automatu ekvivalentného danej bezkontextovej gramatike. V nasledujúcom sa zameriame na opačnú konštrukciu, pomocou ktorej možno k danému zásobníkovému automatu  $A$  zostrojiť bezkontextovú gramatiku  $G$  takú, že  $L(G) = N(A)$ .

**Tvrdenie 4.** *Nech  $A = (K, \Sigma, \Gamma, \delta, q_0, Z_0, F)$  je zásobníkový automat. Potom existuje bezkontextová gramatika  $G = (N, T, P, \sigma)$  taká, že  $L(G) = N(A)$ .*

Gramatika  $G$ , ekvivalentná automatu  $A$ , obsahuje pre všetky dvojice stavov  $p, q \in K$  a všetky zásobníkové symboly  $Z \in \Gamma$  neterminál  $[p, Z, q]$ . Množinu pravidiel gramatiky  $G$  skonštruujeme tak, aby pre slovo  $w \in T^*$  bolo  $[p, Z, q] \Rightarrow^* w$  práve vtedy, keď  $(p, w, Z) \vdash^* (q, \varepsilon, \varepsilon)$ . Tento zápis vyjadruje, že z neterminálu  $[p, Z, q]$  sa dajú vygenerovať práve všetky terminálne slová  $w$ , pre ktoré existuje výpočet automatu  $A$  na slove  $w$  s nasledujúcimi vlastnosťami:

- (i) výpočet sa začína v konfigurácii so stavom  $p$  a so symbolom  $Z$  na vrchu zásobníka,
- (ii) končí sa v konfigurácii so stavom  $q$  a
- (iii) na konci tohto výpočtu sa výška zásobníka automatu  $A$  po prvý raz dostane o úroveň nižšie v porovnaní s jeho začiatkom.

Kedže automat  $A$  akceptuje prázdnym zásobníkom, mala by gramatika  $G$  generovať práve všetky slová odvoditeľné z neterminálov  $[q_0, Z_0, q]$  pre  $q \in K$ . Pre každé  $q \in K$  teda v množine  $P$  bude pravidlo  $\sigma \rightarrow [q_0, Z_0, q]$ . Zostáva definovať pravidlá zabezpečujúce „správanie“ neterminálov  $[p, Z, q]$  opísané v predchádzajúcim odstavci. Ak pre nejaké  $p, q \in K, z \in \Sigma \cup \{\varepsilon\}$  a  $Z \in \Gamma$  je  $(q, \varepsilon) \in \delta(p, z, Z)$ , malo by byť aj  $[p, Z, q] \Rightarrow^* z$ . Preto bude množina  $P$  obsahovať pravidlo  $[p, Z, q] \rightarrow z$ . Ak ďalej pre nejaké  $p, r \in K, z \in \Sigma \cup \{\varepsilon\}, k \in \mathbb{N} - \{0\}$  a  $Z, Z_1, \dots, Z_k \in \Gamma$  je  $(r, Z_1 \dots Z_k) \in \delta(p, z, Z)$ , dôjde k zníženiu výšky zásobníka pod pôvodnú úroveň až potom, čo postupne príde k zníženiu pod úroveň novo pridaného  $Z_k$ , neskôr pod úroveň novo pridaného  $Z_{k-1}$ , atď. – až napokon zásobník klesne pod úroveň novo pridaného  $Z_1$ . Stavy, v ktorých k týmto zníženiam úrovne zásobníka dôjde, môžu byť vo všeobecnosti ľubovoľné. Pre všetky stavy  $q, q_1, \dots, q_{k-1} \in K$  tak bude množina  $P$  obsahovať pravidlo  $[p, Z, q] \rightarrow z[r, Z_k, q_1][q_1, Z_{k-1}, q_2] \dots [q_{k-1}, Z_1, q]$ .

Gramatika  $G = (N, T, P, \sigma)$  ekvivalentná automatu  $A = (K, \Sigma, \Gamma, \delta, q_0, Z_0, F)$  je teda daná nasledovne:  $N = \{[p, Z, q] \mid p, q \in K; Z \in \Gamma\} \cup \{\sigma\}$ ,  $T = \Sigma$  a

- (i) Pre každé  $q \in K$  obsahuje  $P$  pravidlo  $\sigma \rightarrow [q_0, Z_0, q]$ .
- (ii) Pre všetky  $p, q \in K, z \in \Sigma \cup \{\varepsilon\}$  a  $Z \in \Gamma$  s  $(q, \varepsilon) \in \delta(p, z, Z)$  obsahuje  $P$  pravidlo  $[p, Z, q] \rightarrow z$ .
- (iii) Pre všetky  $p, r \in K, z \in \Sigma \cup \{\varepsilon\}, k \in \mathbb{N} - \{0\}$  a  $Z, Z_1, \dots, Z_k \in \Gamma$  s  $(r, Z_1 \dots Z_k) \in \delta(p, z, Z)$  obsahuje  $P$  pre všetky  $q, q_1, \dots, q_{k-1} \in K$  pravidlo  $[p, Z, q] \rightarrow z[r, Z_k, q_1] \dots [q_{k-1}, Z_1, q]$ .
- (iv) Množina  $P$  neobsahuje žiadne iné pravidlá.

# Bezkontextové jazyky

Peter Kostolányi

16. novembra 2022

Vieme už, že zásobníkové automaty sú – bez ohľadu na mód akceptácie – z hľadiska opisnej sily ekvivalentné bezkontextovým gramatikám. Obidva modely teda realizujú rovnakú triedu jazykov, ktoré nazývame *bezkontextovými*.

**Definícia 1.** *Bezkontextový jazyk* je jazyk  $L$ , pre ktorý existuje bezkontextová gramatika  $G$  taká, že  $L(G) = L$ . Triedu všetkých bezkontextových jazykov označujeme  $\mathcal{L}_{CF}$ .

**Veta 1.** Nech  $L$  je jazyk. Nasledujúce tvrdenia sú ekvivalentné:

- (i) Jazyk  $L$  je bezkontextový.
- (ii) Existuje zásobníkový automat  $A$  taký, že  $L(A) = L$ .
- (iii) Existuje zásobníkový automat  $A$  taký, že  $N(A) = L$ .

## 1 Pumpovacia lema pre bezkontextové jazyky

Nech  $L$  je *ľuboľný* bezkontextový jazyk. Potom *existuje* bezkontextová gramatika  $G = (N, T, P, \sigma)$  taká, že  $L(G) = L$ . Nech má najdlhšia pravá strana pravidla z  $P$  dĺžku  $k$ ; bez ujmy na všeobecnosti predpokladajme, že  $k \geq 2$ . V každom kroku odvodenia potom možno jeden neterminál prepísať na najviac  $k$  symbolov. Strom odvodenia s hĺbkou menšou alebo rovnou  $n$  tak môže zodpovedať iba slovu dĺžky najviac  $k^n$  a každý strom odvodenia slova dĺžky aspoň  $k^n$  preto musí mať hĺbku najmenej  $n$ . Položme  $p := k^{|N|+1}$ .

Nech  $w \in L$  je *ľuboľné* slovo také, že  $|w| \geq p$ . Vezmime strom najkratšieho<sup>1</sup> odvodenia slova  $w$  v gramatike  $G$ . Ten má hĺbkou aspoň  $|N| + 1$ . Najdlhšia cesta z koreňa do niektorého z listov preto obsahuje aspoň  $|N| + 2$  uzlov, pričom v liste je terminál alebo prázdne slovo a vo vnútorných uzloch, ktorých je aspoň  $|N| + 1$ , sú neterminály. Na tejto ceste sa teda musí zopakovať niektorý neterminál  $\xi$ . Ak je cesta dlhšia ako  $|N| + 1$ , môžeme predpokladať, že k obom výskytom neterminálu  $\xi$  dôjde v rámci najnižších  $|N| + 2$  úrovni. Situácia je znázornená na obrázku 1.

Odvodenie slova  $w$  v gramatike  $G$  tak možno s odkazom na obrázok 1 zapísat nasledovne:

$$\sigma \Rightarrow^* x\xi z \Rightarrow^+ xu\xi v z \Rightarrow^* xuyvz = w.$$

Pre slovo  $w$  teda *existujú* slová  $x, u, y, v, z$  také, že  $w = xuyvz$ . Keďže k podľa  $uyv$  prislúcha podstrom hĺbky najviac  $|N| + 1$ , nutne  $|uyv| \leq k^{|N|+1} = p$ . Keby boli obidve slová  $u, v$  prázdne, bolo by možné časť odvodenia – konkrétnie  $x\xi z \Rightarrow^+ xu\xi v z$  – vynechať bezo zmeny vygenerovaného slova; to by ale odporovalo nášmu predpokladu, podľa ktorého ide o najkratšie odvodenie slova  $w$ . Preto  $|uv| \geq 1$ . Keďže napokon možno uvedenú časť odvodenia –  $x\xi z \Rightarrow^+ xu\xi v z$  – vynechať alebo niekoľkokrát zopakovať, pre každé  $i \in \mathbb{N}$  patrí aj slovo  $xu^i yv^i z$  do jazyka generovaného gramatikou  $G$ . Dokázali sme teda nasledujúce tvrdenie – *pumpovaciu lemu pre bezkontextové jazyky*.

**Veta 2.** Nech  $\Sigma$  je abeceda a  $L \subseteq \Sigma^*$  bezkontextový jazyk. Potom existuje  $p \in \mathbb{N}$  také, že pre všetky  $w \in L$  s  $|w| \geq p$  existujú slová  $x, u, y, v, z \in \Sigma^*$  také, že:

- (i)  $w = xuyvz$ ,
- (ii)  $|uyv| \leq p$ ,
- (iii)  $|uv| \geq 1$ ,
- (iv)  $\forall i \in \mathbb{N} : xu^i yv^i z \in L$ .

3. **Rekurzívne vyčísliteľné a rekurzívne jazyky.** [Turingove stroje, frázové gramatiky, ich ekvivalencia, uzáverové vlastnosti, univerzálny Turingov stroj, Turingova hypotéza.]

# 1 Deterministické Turingove stroje

**Definícia 1.** *Deterministický Turingov stroj* je šestica  $A = (K, \Sigma, \Gamma, \delta, q_0, F)$ , kde  $K$  je konečná množina stavov,  $\Sigma$  je vstupná abeceda,  $\Gamma \supseteq \Sigma$  je pracovná abeceda neobsahujúca špeciálny symbol  $\mathbf{B} \notin \Gamma$  („blank“),  $\delta: K \times (\Gamma \cup \{\mathbf{B}\}) \rightarrow K \times \Gamma \times \{-1, 0, 1\}$  je čiastočná prechodová funkcia,  $q_0 \in K$  je počiatočný stav a  $F \subseteq K$  je množina akceptačných stavov.

Čiastočnosť prechodovej funkcie  $\delta$  znamená, že pre niektoré dvojice  $(q, c) \in K \times (\Gamma \cup \{\mathbf{B}\})$  môže byť jej výstup aj nedefinovaný – to sa niekedy označuje ako  $\delta(q, c) = \perp$ . Symbol  $\rightarrow$ , použitý v uvedenej definícii, sa občas používa namiesto klasickej šípky na zdôraznenie čiastočnosti funkcie.

Deterministický Turingov stroj sa niekedy definuje aj ako špeciálny prípad nedeterministického Turingovho stroja – pre každé  $q \in K$  a  $c \in \Gamma$  je potom  $\delta(q, c)$  najviac jednoprvková podmnožina množiny  $K \times \Gamma \times \{-1, 0, 1\}$ . Aj keď ide o formálne odlišnú definíciu, korešpondencia s tou našou by mala byť očividná. Možno sa tiež stretnúť s definíciou, pri ktorej  $\mathbf{B} \in \Gamma$ . V takom prípade je prechodovou funkciou zobrazenie  $\delta: K \times \Gamma \rightarrow K \times (\Gamma - \{\mathbf{B}\}) \times \{-1, 0, 1\}$ .

Podobne ako pri konečných a zásobníkových automatoch, rozumieme aj pri Turingových strojoch pod konfiguráciou objekt nesúci informáciu o dosiaľ vykonanej časti výpočtu postačujúcu na to, aby bolo možné iba na základe tej vo výpočte pokračovať. Z toho je zrejmé, že v konfigurácii musí byť nejakým spôsobom zakódovaný stav, obsah pásky, ako aj pozícia čítacej hlavy. Objekt nesúci takúto informáciu možno definovať množstvom spôsobov, pričom v rôznych situáciách môžu byť výhodné rôzne formalizácie. Z tohto dôvodu konfiguráciu deterministického Turingovho stroja definujeme hneď v troch variantoch, pričom neskôr budeme vždy pracovať s variantom, ktorý sa bude javiť ako momentálne najvhodnejší.

**Definícia 2.** Nech  $A = (K, \Sigma, \Gamma, \delta, q_0, F)$  je deterministický Turingov stroj.

- (i) Konfiguráciou stroja  $A$  možno nazvať trojicu  $(q, w, n) \in K \times \mathbf{B}\Gamma^*\mathbf{B} \times \mathbb{N}$ , kde  $q$  je stav,  $w$  je slovo reprezentujúce obsah zapísanej časti pásky s dvoma susednými symbolmi „blank“ a číslo  $n \in \{0, \dots, |w| - 1\}$  udáva pozíciu čítacej hlavy vzhľadom na začiatok zapísanej časti pásky.<sup>1</sup>
- (ii) Nech  $\uparrow \notin \Gamma$ . Konfiguráciou Turingovho stroja  $A$  možno v takom prípade rozumieť aj dvojicu  $(q, w) \in K \times (\uparrow\mathbf{B}\Gamma^*\mathbf{B} \cup \mathbf{B}\Gamma^*\uparrow\Gamma^*\mathbf{B})$ , kde  $q$  je stav a  $w$  je slovo reprezentujúce obsah zapísanej časti pásky s dvoma susednými symbolmi „blank“ a pozíciou hlavy vyznačenou špeciálnym symbolom  $\uparrow$  – hlava v takom prípade číta písmeno nasledujúce bezprostredne za  $\uparrow$ .
- (iii) Nech  $K \cap \Gamma = \emptyset$ . Konfiguráciou stroja  $A$  možno nazvať aj slovo  $w \in K\mathbf{B}\Gamma^*\mathbf{B} \cup \mathbf{B}\Gamma^*K\Gamma^*\mathbf{B}$ . Špeciálny symbol  $\uparrow$  z variantu (ii) je tu nahradený priamo stavom stroja  $A$ , ktorý v takom prípade nie je nutné udržiavať ako samostatnú zložku konfigurácie.

**Definícia 3.** Nech  $A = (K, \Sigma, \Gamma, \delta, q_0, F)$  je deterministický Turingov stroj. Krok výpočtu stroja  $A$  je binárna relácia  $\vdash_A$  na konfiguráciách tohto stroja definovaná nasledovne:

- (i) Pre všetky  $p, q \in K$ ,  $n \in \mathbb{N}$ ,  $a_1, \dots, a_n, c \in \Gamma$ ,  $k \in [n]$  a  $d \in \{-1, 0, 1\}$  je

$$(p, \mathbf{B}a_1 \dots a_{k-1}a_k a_{k+1} \dots a_n \mathbf{B}, k) \vdash_A (q, \mathbf{B}a_1 \dots a_{k-1}c a_{k+1} \dots a_n \mathbf{B}, k + d)$$

práve vtedy, keď  $\delta(p, a_k) = (q, c, d)$ .

- (ii) Pre všetky  $p, q \in K$ ,  $n \in \mathbb{N}$ ,  $a_1, \dots, a_n, c \in \Gamma$  a  $d \in \{-1, 0, 1\}$  je

$$(p, \mathbf{B}a_1 \dots a_n \mathbf{B}, 0) \vdash_A (q, \mathbf{B}c a_1 \dots a_n \mathbf{B}, d + 1)$$

práve vtedy, keď  $\delta(p, \mathbf{B}) = (q, c, d)$ .

- (iii) Pre všetky  $p, q \in K$ ,  $n \in \mathbb{N}$ ,  $a_1, \dots, a_n, c \in \Gamma$  a  $d \in \{-1, 0, 1\}$  je

$$(p, \mathbf{B}a_1 \dots a_n \mathbf{B}, n + 1) \vdash_A (q, \mathbf{B}a_1 \dots a_n c \mathbf{B}, n + 1 + d)$$

práve vtedy, keď  $\delta(p, \mathbf{B}) = (q, c, d)$ .

- (iv) V relácii  $\vdash_A$  nie sú žiadne ďalšie dvojice konfigurácií.

V prípade, že je uvažovaný stroj  $A$  zrejmý z kontextu, píšeme namiesto  $\vdash_A$  iba  $\vdash$ .

Jazyk akceptovaný Turingovým strojom sa v literatúre definuje množstvom rôznych – ale vzájomne ekvivalentných – spôsobov. Podľa našej definície bude stroj akceptovať svoj vstup, kedykoľvek sa na ňom dokáže dostať do akceptačného stavu – bez ohľadu na to, či sa výpočet v tomto stave zastaví a či bol vstup dočítaný až do konca.

**Definícia 4.** Nech  $A = (K, \Sigma, \Gamma, \delta, q_0, F)$  je deterministický Turingov stroj. Jazyk akceptovaný strojom  $A$  je daný ako

$$L(A) = \{w \in \Sigma^* \mid \exists q \in F \ \exists u \in \Gamma^* \ \exists k \in \mathbb{N} : (q_0, \mathbf{B}w\mathbf{B}, 1) \vdash^* (q, \mathbf{Bu}\mathbf{B}, k)\}.$$

Podľa predchádzajúcej definície teda Turingov stroj akceptuje svoj vstup aj v prípade, že výpočet cez akceptačnú konfiguráciu iba „prejde“, pričom skončí v neakceptačnej konfigurácii alebo neskončí vôbec (pod koncom výpočtu tu máme na mysli situáciu, keď sa stroj „zasekne“, pretože na daný symbol nie je z daného stavu definovaný žiadny prechod).

Lahko však vidieť, že naša definícia akceptovaného jazyka je ekvivalentná zdanlivo prísnejšej definícii požadujúcej *zastavenie* výpočtu v akceptačnom stave. Ak totiž výpočet vyhovuje takejto silnejšej podmienke akceptácie, triviálne vyhovuje aj podmienku z definície 7. Pre dôkaz opačným smerom si stačí uvedomiť, že prechody vedúce z akceptačných stavov sú zbytočné: ak sa stroj raz dostane do niektorého akceptačného stavu, nemusí už vo výpočte ďalej pokračovať, pretože o akceptovaní vstupu „už je rozhodnuté“. Po odobratí takýchto zbytočných prechodov teda zjavne dostávame Turingov stroj akceptujúci rovnaký jazyk podľa pozmenenej definície.

Definíciu akceptovaného jazyka je možné ekvivalentne upraviť aj tak, aby bol stroj donútený pred akceptovaním prečítať celý vstup. Detaily konštrukcie tu prenechávame čitateľovi.

## 2 Nedeterministické Turingove stroje

Princíp nedeterminizmu je pri Turingových strojoch rovnaký ako pri konečných alebo zásobníkových automatoch: nedeterministický Turingov stroj môže mať pre daný stav a symbol definovaných vo všeobecnosti aj viacero rôznych prechodov, pričom v rámci kroku výpočtu z „kompatibilnej“ konfigurácie sa môže použiť ktorýkoľvek z nich. Na jednom vstupe teda môže existovať aj viacero výpočtov, pričom nedeterministický Turingov stroj svoj vstup akceptuje práve vtedy, keď *existuje aspoň jeden akceptačný výpočet*.

**Definícia 5.** *Nedeterministický Turingov stroj* je šestica  $A = (K, \Sigma, \Gamma, \delta, q_0, F)$ , kde  $K$  je konečná množina stavov,  $\Sigma$  je vstupná abeceda,  $\Gamma \supseteq \Sigma$  je pracovná abeceda neobsahujúca špeciálny symbol  $\mathbf{B} \notin \Gamma$  („blank“),  $\delta: K \times (\Gamma \cup \{\mathbf{B}\}) \rightarrow 2^{K \times \Gamma \times \{-1, 0, 1\}}$  je prechodová funkcia,  $q_0 \in K$  je počiatok stav a  $F \subseteq K$  je množina akceptačných stavov.

*Konfigurácie* nedeterministického Turingovho stroja definujeme navlas rovnako ako pri strojoch deterministických – možno použiť ľubovoľný z variantov definície 2. Málo prekvapivou úpravou definície pre deterministické Turingove stroje získame aj nasledujúcu definíciu *kroku výpočtu*.

**Definícia 6.** Nech  $A = (K, \Sigma, \Gamma, \delta, q_0, F)$  je nedeterministický Turingov stroj. *Krok výpočtu* stroja  $A$  je binárna relácia  $\vdash_A$  na konfiguráciách tohto stroja definovaná nasledovne:

(i) Pre všetky  $p, q \in K$ ,  $n \in \mathbb{N}$ ,  $a_1, \dots, a_n, c \in \Gamma$ ,  $k \in [n]$  a  $d \in \{-1, 0, 1\}$  je

$$(p, \mathbf{B}a_1 \dots a_{k-1}a_k a_{k+1} \dots a_n \mathbf{B}, k) \vdash_A (q, \mathbf{B}a_1 \dots a_{k-1}c a_{k+1} \dots a_n \mathbf{B}, k+d)$$

práve vtedy, keď  $(q, c, d) \in \delta(p, a_k)$ .

(ii) Pre všetky  $p, q \in K$ ,  $n \in \mathbb{N}$ ,  $a_1, \dots, a_n, c \in \Gamma$  a  $d \in \{-1, 0, 1\}$  je

$$(p, \mathbf{B}a_1 \dots a_n \mathbf{B}, 0) \vdash_A (q, \mathbf{B}c a_1 \dots a_n \mathbf{B}, d+1)$$

práve vtedy, keď  $(q, c, d) \in \delta(p, \mathbf{B})$ .

(iii) Pre všetky  $p, q \in K$ ,  $n \in \mathbb{N}$ ,  $a_1, \dots, a_n, c \in \Gamma$  a  $d \in \{-1, 0, 1\}$  je

$$(p, \mathbf{B}a_1 \dots a_n \mathbf{B}, n+1) \vdash_A (q, \mathbf{B}a_1 \dots a_n c \mathbf{B}, n+1+d)$$

práve vtedy, keď  $(q, c, d) \in \delta(p, \mathbf{B})$ .

(iv) V relácii  $\vdash_A$  nie sú žiadne ďalšie dvojice konfigurácií.

V prípade, že je uvažovaný stroj  $A$  zrejmý z kontextu, píšeme namiesto  $\vdash_A$  iba  $\vdash$ .

Nasledujúca definícia *akceptovaného jazyka* je opäť rovnaká ako pri deterministických Turingových strojoch.

**Definícia 7.** Nech  $A = (K, \Sigma, \Gamma, \delta, q_0, F)$  je nedeterministický Turingov stroj. *Jazyk akceptovaný* strojom  $A$  je daný ako

$$L(A) = \{w \in \Sigma^* \mid \exists q \in F \ \exists u \in \Gamma^* \ \exists k \in \mathbb{N}: (q_0, \mathbf{B}w\mathbf{B}, 1) \vdash^* (q, \mathbf{B}u\mathbf{B}, k)\}.$$

**Definícia 1.4.1** *Frázová gramatika* je štvorica  $G = (N, T, P, \sigma)$ , kde  $N$  a  $T$  sú abecedy neterminálnych a terminálnych<sup>2</sup> symbolov ( $N \cap T = \emptyset$ ),  $\sigma \in N$  je začiatočný neterminál a  $P \subseteq_{kon} (N \cup T)^* N (N \cup T)^* \times (N \cup T)^*$  je konečná množina prepisovacích pravidiel.

## 6.4 Ekvivalence Turingovych strojov a frázových gramatík

**Veta 6.4.1** Ku každej frázovej gramatike  $G$  existuje NTS  $A$  taký, že  $L(G) = L(A)$ .

**Dôkaz.** Analogicky ako pri LBA, simuláciou odvodenia (buď od začiatku alebo od konca). Pri LBA sme uviedli konštrukciu, kedy od konca hámame odvodenie a príslušne upravujeme vettú formu až kým nedostaneme iba začatočný neterminál. Teraz uvedieme druhú možnú konštrukciu.

Vstupné slovo si odložíme na jednu stopu pásky. Na druhej stope nedeterministicky hámame a simulujeme jeho odvodenie v  $G$ . Keďže máme k dispozícii nekonečne dlhú pásku, už nás netrápi, že vettú forma počas výpočtu môže byť dlhšia ako vstupné slovo. Ak sa nám vstupné slovo podarí odvodiť, akceptujeme.

**Veta 6.4.2** Ku každému NTS  $A$  existuje frázová gramatika  $G$  taká, že  $L(A) = L(G)$ .

**Dôkaz.** Opäť analogicky ako pri LBA: Gramatika bude mať dvojposchodové neterminály. Najskôr na hornom poschode vygeneruje ľubovoľné slovo. To skopíruje na dolné poschodie, upraví ho na začatočnú konfiguráciu  $A$  (najlepšie v nami definovanom tvare, kde je pozícia hlavy vyznačená symbolom stavu) a prepisovaním vettnej formy simuluje výpočet  $A$ . Ak sa vo vettnej forme vyskytne akceptačný stav,  $G$  prepíše horné poschodie neterminálov na terminály a ostatné nadbytočné neterminály zmaže.

### Kódovanie TS

V tejto časti si poviem o zakódovaní daného Turingovo stroja do postupnosti z  $\{0, 1\}^*$ . Je zjavne nemožné zakódovať konkrétnie vstupné symboly, ktoré používa, ale lema 6.6.1 hovorí, že to ani nie je nutné. Navyše vieme, že výpočtová sila DTS a NTS je rovnaká. Preto sa (v tejto aj ďalších častiach) obmedzíme na deterministické Turingove stroje so vstupnou abecedou  $\{0, 1\}$ . Keď budeme mať daný kód TS (kód TS  $A$  označujme  $\langle A \rangle$ ), budeme môcť okrem iného neskôr zstrojiť tzv. univerzálny TS, ktorý dostane na vstup kód nejakého TS  $A$  a vstupné slovo  $w$  a bude simulovať stroj  $A$  na slove  $w$ .

## 2 Turingova téza

Dospeli sme teda k formálnej definícii rozhodovacieho problému – ide o jazyk nad nejakou abecedou  $\Sigma$ . V podobnom duchu ešte potrebujeme formalizovať aj pojem algoritmu riešiaceho daný rozhodovací problém. Tu sa vhodnou formalizáciou java byť *deterministické Turingove stroje, ktoré sa zastavia na každom vstupe* a ktoré akceptujú jazyk zodpovedajúci uvažovanému rozhodovaciemu problému.

Alana Turinga viedla k definícii matematickej abstrakcie počítacieho zariadenia dnes známej ako Turingov stroj práve potreba formalizácie pojmu algoritmus.<sup>1</sup> *Turingova téza* je – principiálne neoveriteľné – tvrdenie, podľa ktorého každý (v neformálnom zmysle) algoritmický výpočet možno realizovať na Turingovom stroji. Turingovu tézu nemožno ani dokázať, ani vyvrátiť – ide totiž o tvrdenie, ktoré hovorí, že Turingove stroje sú vhodnou formalizáciou inak čisto intuitívne chápaného pojmu algoritmu. Turingov stroj teda možno chápať ako formálnu *definíciu* algoritmu, pričom Turingova téza vyjadruje presvedčenie, že táto definícia je odzrkadlením skutočnosti.

Napriek principiálnej neoveriteľnosti Turingovej tézy existujú viaceré argumenty na jej podporu. Tým najdôležitejším je skutočnosť, že prakticky všetky pokusy Turingových súčasníkov a následníkov o alternatívnu definíciu algoritmu vyústili v model, ktorý je z hľadiska sily s Turingovými strojmi ekvivalentný. Jednou z takýchto alternatívnych formalizácií je napríklad  $\lambda$ -kalkul Alonza Churcha, vďaka čomu je Turingova téza v literatúre známa aj ako Churchova-Turingova téza. Ďalšími formalizáciami sú napríklad rekurzívne funkcie, Minského registrové stroje, Markovove algoritmy, atď. V nasledujúcom prijmete Turingovu tézu a Turingove stroje využijeme ako prostriedok na vybudovanie základov teórie algoritmickej vypočítateľnosti.

**Formalizácia 2.** Pod *algoritmom* rozhodujúcim daný rozhodovací problém budeme rozumieť *deterministický Turingov stroj, ktorý sa na každom vstupe zastaví a ktorý akceptuje jazyk zodpovedajúci uvažovanému rozhodovaciemu problému*.

4. **Nerozhodnuteľné problémy.** [Diagonalizácia, problém zastavenia, metódy dokazovania nerozhodnuteľnosti.]

## 7 Diagonálny problém a jeho nerozhodnuteľnosť

Diagonálny problém je rozhodovací problém daný nasledovne:

**Vstup:** Kód  $\langle A \rangle$  deterministického Turingovho stroja<sup>2</sup>  $A$  nad vstupnou abecedou  $\{0, 1\}$ .

**Výstup:** „Áno“ práve vtedy, keď  $\langle A \rangle \in L(A)$ .

Diagonálnemu problému zodpovedá *diagonálny jazyk*:

$$L_D = \{\langle A \rangle \mid A \text{ je det. TS nad vstupnou abecedou } \{0, 1\}; \langle A \rangle \in L(A)\}.$$

Podobne možno uvažovať aj *komplementárny diagonálny problém*, ktorý je daný takto:

**Vstup:** Kód  $\langle A \rangle$  deterministického Turingovho stroja  $A$  nad vstupnou abecedou  $\{0, 1\}$ .

**Výstup:** „Áno“ práve vtedy, keď  $\langle A \rangle \notin L(A)$ .

Komplementárному diagonálnemu problému zodpovedá *komplement diagonálneho jazyka*:

$$L_D^C = \{\langle A \rangle \mid A \text{ je det. TS nad vstupnou abecedou } \{0, 1\}; \langle A \rangle \notin L(A)\}.$$

**Poznámka 1.** Skutočnosť, že komplementárному diagonálnemu problému zodpovedá jazyk  $L_D^C$  je dôsledkom toho, že každý binárny reťazec je kódom nejakého Turingovho stroja – „nezmyselné“ reťazce zodpovedajú stroju akceptujúcemu prázdny jazyk.

**Veta 1.** *Komplementárny diagonálny problém nie je rekurzívne vyčísliteľný:  $L_D^C \notin \mathcal{L}_{RE}$ .*

*Dôkaz.* Sporom – nech  $L_D^C \in \mathcal{L}_{RE}$  a  $M$  je deterministický Turingov stroj taký, že  $L(M) = L_D^C$ . Ak  $\langle M \rangle \in L(M)$ , z definície diagonálneho jazyka je  $\langle M \rangle \in L_D$ . To je spor, pretože  $L(M) = L_D^C$ , a teda  $L(M)$  nemôže obsahovať slovo  $\langle M \rangle \in L_D$ . Preto  $\langle M \rangle \notin L(M)$ . V takom prípade ale z definície diagonálneho jazyka vyplýva  $\langle M \rangle \in L_D^C$  a z definície stroja  $M$  je  $\langle M \rangle \in L(M)$ : spor.  $\square$

**Veta 2.** *Diagonálny problém je rekurzívne vyčísliteľný, ale nie je rozhodnuteľný:  $L_D \in \mathcal{L}_{RE} - \mathcal{L}_{rec}$ .*

*Dôkaz.* Zjavne  $L_D \in \mathcal{L}_{RE}$  – stroj akceptujúci jazyk  $L_D$  môže pracovať tak, že pre každý vstup  $\langle A \rangle$  odsimuluje výpočet univerzálneho Turingovho stroja na vstupe  $\langle A \rangle \# \langle A \rangle$ .

Ďalej sporom, nech  $L_D \in \mathcal{L}_{rec}$ . Potom existuje deterministický Turingov stroj  $M$  zastavujúci na každom vstupe taký, že  $L(M) = L_D$ . Nech  $M'$  pracuje ako  $M$  s rozdielom, že  $M'$  akceptuje práve vtedy, keď  $M$  zamieta. Zjavne  $L(M') = L_D^C$ , z čoho  $L_D^C \in \mathcal{L}_{rec} \subseteq \mathcal{L}_{RE}$  – spor.  $\square$

### 3 Stroje zastavujúce na každom vstupe a rekurzívne jazyky

Požiadavka zastavenia Turingovho stroja na každom vstupe odzrkadľuje vlastnosť konečnosti výpočtov, ktorá je späť s intuitívnou predstavou o algoritme. Zamerajme sa na jej význam a dôsledky.

Uvažujme deterministický Turingov stroj  $A$  pracujúci na vstupe  $w$ . Bez ujmy na všeobecnosti môžeme predpokladať, že stroj  $A$  neobsahuje žiadne prechody vedúce z akceptačných stavov. Stroj  $A$  sa teda v akceptačnej konfigurácii vždy „zasekne“. Výpočet stroja  $A$  na slove  $w$  tak môže prebiehať nasledujúcimi troma spôsobmi.

1. Po nejakom počte krokov príde do akceptačnej konfigurácie a zastaví sa. Potom  $w \in L(A)$ .
2. Po nejakom počte krokov sa zastaví v neakceptačnej konfigurácii. Potom  $w \notin L(A)$ .
3. Nikdy sa nezastaví a všetky konfigurácie, cez ktoré prejde, sú neakceptačné. Potom  $w \notin L(A)$ .

Uvažujme teraz pozorovateľa výpočtu stroja  $A$  na slove  $w$ , ktorý sa iba na základe neho snaží zistiť, či  $w \in L(A)$ . Jedinou informáciou, ktorú má pozorovateľ k dispozícii je, či výpočet skončil a ak áno, tak s akým výsledkom. Ľahko vidieť, že pozorovateľ nedokáže v žiadnom bode výpočtu odlišiť prípad, keď je výpočet nekonečný – a teda  $w \notin L(A)$  – od prípadu, keď je nutné odsimulovať ešte niekoľko krokov a výpočet sa zastaví – a teda môže platiť  $w \in L(A)$  aj  $w \notin L(A)$ .

Požiadavka zastavenia Turingovho stroja  $A$  na každom vstupe zabezpečí, že výpočet stroja  $A$  na slove  $w$  môže prebiehať iba nasledujúcimi dvoma spôsobmi.

1. Po nejakom počte krokov príde do akceptačnej konfigurácie a zastaví sa. Potom  $w \in L(A)$ .
2. Po nejakom počte krokov sa zastaví v neakceptačnej konfigurácii. Potom  $w \notin L(A)$ .

Jazyky akceptované takýmito strojmi nazveme *rekurzívnymi*.

**Definícia 1.** Jazyk  $L$  je *rekurzívny*, ak existuje deterministický Turingov stroj  $A$ , ktorý sa na každom vstupe zastaví a pre ktorý platí  $L(A) = L$ . Triedu všetkých rekurzívnych jazykov označujeme  $\mathcal{L}_{rec}$ .

Neskôr ukážeme, že táto definícia má skutočne svoj význam – existuje Turingov stroj, ku ktorému neexistuje žiadny ekvivalentný Turingov stroj zastavujúci na každom vstupe. Trieda  $\mathcal{L}_{rec}$  je teda *vlastnou* podriedou  $\mathcal{L}_{RE}$ .

## 4 Rozhodnuteľné a nerozhodnuteľné problémy

Na základe Turingovej tézy sme pojem algoritmu rozhodujúceho daný rozhodovací problém sformalizovali ako deterministický Turingov stroj, ktorý sa na každom vstupe zastaví a ktorý akceptuje jazyk zodpovedajúci tomuto rozhodovaciemu problému.

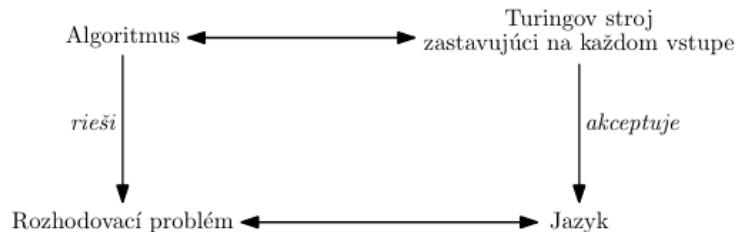
Rozhodovací problém nazveme *rozhodnuteľným*, ak existuje algoritmus, ktorý ho rozhoduje – čiže ak existuje Turingov stroj, ktorý sa na každom vstupe zastaví a ktorý akceptuje jazyk k uvažovanému rozhodovaciemu problému prislúchajúci. Táto situácia nastane práve vtedy, keď je jazyk prislúchajúci k danému rozhodovaciemu problému rekurzívny.

Rozhodovací problém nazveme *nerozhodnuteľným*, ak nie je rozhodnuteľný. Existenciu nerozhodnuteľných problémov dokážeme neskôr.

Rozhodovací problém nazveme *rekurzívne vyčísliteľným* (niekde tiež *čiastočne rozhodnuteľným*), ak existuje Turingov stroj, ktorý akceptuje jemu zodpovedajúci jazyk – pričom tento sa už nemusí zastaviť na každom vstupe. To nastane práve vtedy, keď je jazyk zodpovedajúci danému rozhodovaciemu problému rekurzívne vyčísliteľný. Rekurzívne vyčísliteľný problém môže byť rozhodnuteľný (ak mu zodpovedá rekurzívny jazyk), alebo nerozhodnuteľný (v opačnom prípade).

## 5 Rozhodovacie problémy, algoritmy a ich formalizácie

Vzťahy intuitívne chápanych pojmov rozhodovacieho problému a algoritmu k ich formalizáciám sú znázornené na obrázku 1.



Obr. 1: Rozhodovacie problémy, algoritmy a ich formalizácie.

Ďalšie pojmy zavedené vyššie sú spoločne s ich formalizáciami zhrnuté v tabuľke 1.

Pojem	Formalizácia
Rozhodovací problém	Jazyk (nad nejakou abecedou $\Sigma$ )
Algoritmus	Turingov stroj zastavujúci na každom vstupe
Rozhodnuteľný problém	Jazyk z triedy $\mathcal{L}_{rec}$
Nerozhodnuteľný problém	Jazyk mimo triedy $\mathcal{L}_{rec}$
Rekurzívne vyčísliteľný problém	Jazyk z triedy $\mathcal{L}_{RE}$

Tabuľka 1: Zhrnutie kľúčových pojmov teórie rozhodnuteľnosti a ich formalizácií.

## 6 Existencia jazykov mimo $\mathcal{L}_{RE}$

Rekurzívne vyčísliteľných jazykov nad abecedou  $\Sigma = \{0, 1\}$  existuje najviac toľko, čo všetkých kódov Turingových strojov s binárnou vstupnou abecedou – každé zobrazenie, ktoré rekurzívne vyčísliteľnému jazyku  $L$  priradí kód  $\langle A \rangle$  niektorého deterministického Turingovho stroja  $A$  akceptujúceho  $L$ , je totiž evidentne injektívne. Kód Turingovho stroja je (konečný) binárny reťazec a množina všetkých takýchto reťazcov je spočítateľne nekonečná. Všetkých binárnych rekurzívne vyčísliteľných jazykov je teda tiež len spočítateľne veľa. Všetkých jazykov nad abecedou  $\Sigma = \{0, 1\}$  je naopak nespočítateľne veľa, lebo ide o podmnožiny spočítateľne nekonečnej množiny  $\Sigma^*$ . Musí preto existovať jazyk nad abecedou  $\Sigma = \{0, 1\}$ , ktorý nie je rekurzívne vyčísliteľný.

Rekurzívne vyčísliteľné jazyky zodpovedajú rekurzívne vyčísliteľným problémom. Z uvedeného teda vyplýva, že existuje rozhodovací problém, ktorý nie je rekurzívne vyčísliteľný. Každý rozhodnutieľný problém je nutne aj rekurzívne vyčísliteľný. Z dokázaného tvrdenia teda tiež vyplýva, že existuje rozhodovací problém, ktorý nie je rozhodnuteľný.

Uvedený dôkaz je čisto existenčný. Nasledujúcim ale nájdeme aj *konkrétny* jazyk (rozhodovací problém), o ktorom dokážeme, že nie je rekurzívne vyčísliteľný.

5. **Miery zložitosti pre Turingove stroje** [triedy zložitosti, kompresia pásky, zrýchľovanie výpočtov, vplyv redukcie počtu pások na zložitosť]

### 7.3 Triedy zložitosti

Na základe definícií časovo a priestorovo ohraničeného TS rozčleníme jazyky do tried zložitosti – podľa najmenšieho času, resp. pamäte, ktoré stačia na jeho rozpoznanie Turingovym strojom.

**Definícia 7.3.1** Každá funkcia nám určuje triedu jazykov, ku ktorým existuje ľahké ohraničenie TS. Musíme rozlísiť, či tento TS je deterministický a či ide o časové alebo pamäťové ohraničenie.

Definujeme triedy

$$\begin{aligned} \text{DSPACE}(S(n)) &= \{L \mid \exists \text{ DTS } A, \text{ kt. je } S(n) \text{ priestorovo ohraničený a } L = L(A)\} \\ \text{NSPACE}(S(n)) &= \{L \mid \exists \text{ NTS } A, \text{ kt. je } S(n) \text{ priestorovo ohraničený a } L = L(A)\} \\ \text{DTIME}(T(n)) &= \{L \mid \exists \text{ DTS } A, \text{ kt. je } T(n) \text{ časovo ohraničený a } L = L(A)\} \\ \text{NTIME}(T(n)) &= \{L \mid \exists \text{ NTS } A, \text{ kt. je } T(n) \text{ časovo ohraničený a } L = L(A)\} \end{aligned}$$

**Poznámka 7.3.1** Pripomíname, že nadálej hovoríme o modeli TS so vstupnou páskou len na čítanie a  $k$  pracovnými páskami.

**Poznámka 7.3.2** Definujú sa tiež triedy  $(N/D)\text{SPACETIME}(S(n), T(n))$ , kde obmedzíme TS ako priestor, tak aj čas.

**Poznámka 7.3.3** Ako ohraničenie môžeme použiť aj asymptotickú notáciu, teda namiesto  $f(n)$  uvedieme  $O(f(n))$ . Z viede 7.2.1 o kompresii pásky a 7.2.2 o lineárnom zrýchlení vyplýva, že pre rozumné funkcie  $f(n)$  je jedno, či zoberieme ako ohraničenie  $f(n)$  alebo  $cf(n)$ , kde  $c$  je kladná konštantá. V takomto prípade je teda jedno, či použijeme ohraničenie  $f(n)$  alebo  $O(f(n))$ .

Teda napr.  $\text{DSPACE}(O(n))$  je trieda jazykov, pre ktoré existuje  $O(n)$  priestorovo ohraničený DTS. Z vety o kompresii pásky vieme, že  $\text{DSPACE}(O(n)) = \text{DSPACE}(n)$ . Formálne môžeme definovať  $\text{DSPACE}(O(f(n))) = \bigcup_{g(n) \in O(f(n))} \text{DSPACE}(g(n))$ , pre ostatné triedy analogicky.

**Definícia 7.3.2** Špeciálne nás budú neskôr zaujímať triedy, kde je čas alebo priestor obmedzený nejakým polynomom, prípadne inou jednoduchou funkciou, pričom nás bude zaujímať len typ tejto funkcie, nie konkrétné koeficienty v nej. Najznámejšie takéto triedy jazykov sú:

$$\begin{aligned} P &= \bigcup_{f \text{ je polynom}} \text{DTIME}(f(n)) = \bigcup_{k>0} \text{DTIME}(O(n^k)) = \text{DTIME}(n^{O(1)}) \\ NP &= \bigcup_{f \text{ je polynom}} \text{NTIME}(f(n)) = \bigcup_{k>0} \text{NTIME}(O(n^k)) = \text{NTIME}(n^{O(1)}) \\ \text{PSPACE} &= \bigcup_{k>0} \text{DSPACE}(n^k) = \text{DSPACE}(n^{O(1)}) \\ \text{NPSPACE} &= \bigcup_{k>0} \text{NSPACE}(n^k) = \text{NSPACE}(n^{O(1)}) \\ L &= \text{DSPACE}(\log n) \\ NL &= \text{NSPACE}(\log n) \end{aligned}$$

Teraz ukážeme, že niektoré z takto definovaných tried jazykov už poznáme pod inými menami.

**Veta 7.3.1**  $\mathcal{L}_{ECS} = NSPACE(n) = NSPACE(O(n))$

**Dôkaz.** Prvá rovnosť vyplýva z definície LBA, druhá z vety 7.2.1 o kompresii pásky. Táto veta poukazuje na skutočnosť, že LBA je skutočne Turingov stroj s lineárne obmedzeným priestorom.

**Veta 7.3.2**  $\mathcal{R} = NSPACE(1) = NSPACE(O(1)) = DSPACE(1) = DSPACE(O(1))$

**Dôkaz.** Podľa lemy 7.5.1 a definícií uvedených tried zjavne  $NSPACE(O(1)) \supseteq DSPACE(O(1))$ ,  $DSPACE(O(1)) \supseteq DSPACE(1)$  a  $NSPACE(O(1)) \supseteq NSPACE(1)$ . Platí  $DSPACE(1) \supseteq \mathcal{R}$ , lebo pre každý regulárny jazyk existuje DTS (simulujúci príslušný DKA), ktorý ho akceptuje dokonca bez používania pomocnej pamäte. A platí aj  $\mathcal{R} \supseteq NSPACE(O(1))$ . Keď totiž máme NTS, ktorý na každej z  $k$  pracovných pások použije najviac  $c$  poličok, vieme si týchto  $kc$  poličok pamätať v stave dvojsmerného nedeterministického konečného automatu, ktorým príslušný TS ľahko odsimulujeme. Z ukázaných inkluzií už vyplýva platnosť všetkých rovností.

**Poznámka 7.3.4** A zjavne platí aj  $\mathcal{R} = DSPACETIME(1, n)$ .

**Poznámka 7.3.5** Z triviálnej platnej inkluzie  $\mathcal{L}_{CF} \subseteq \mathcal{L}_{ECS}$  a rovnosti  $\mathcal{L}_{ECS} = NSPACE(n)$  dostávame  $\mathcal{L}_{CF} \subseteq NSPACE(n)$ . Rozmyslite si, ako by ste túto inkluziu dokázali priamo.

## 7.2 Vety o kompresii, zrýchlení a redukcii počtu pások

Citateľ by si už mal uvedomovať, že medzi silou TS, ktorý je priestorovo obmedzený funkciou  $n^7$  a TS, ktorý je obmedzený funkciou  $2n^7$  nebude prílišný (vlastne žiadny) rozdiel. Trochu prekvapivejšie už bude tvrdenie, že každý TS vieme zrýchliť tak, že jeho čas výpočtu bude  $k$ -krát menší (kde  $k$  je vopred zvolená konštantá). Dôsledkom týchto viet bude, že môžeme pri obmedzovaní na čas a pamäť ignorovať konštanty a dôležitý bude len asymptotický rast funkcie, ktorá bude udávať obmedzenie.

**Veta 7.2.1** (O kompresii pásky) *K ľubovoľnému deterministickému Turingovmu stroju  $A$ , ktorý je  $S(n)$  priestorovo ohraničený a konštante  $k > 1$  existuje deterministický Turingov stroj  $A'$ , ktorý je  $\lceil S(n)/k \rceil$  priestorovo ohraničený a platí  $L(A) = L(A')$ .*

**Dôkaz.** Stačí použiť novú pracovnú abecedu  $\Gamma' = \Gamma^k$ . Každé poličko pásky  $A'$  bude predstavovať  $k$  poličok TS  $A$ .

Podobná veta platí aj o zrýchlení. Budeme chcieť simulovať naraz  $k$  krokov pôvodného TS. Na to najskôr použijeme vhodnú kompresiu pásky, aby sme v dotyčnom jednom kroku zvládli prečítať všetky potrebné symboly. Musíme si však uvedomiť, že vo všeobecnosti potrebujeme prečítať celý vstup, na čo potrebujeme  $n$  krokov.<sup>11</sup> Preto budeme musieť predpokladať, že toto je zanedbateľne malý čas oproti času výpočtu pôvodného TS.

Ak chceme simulať v jednom kroku  $k$  krokov pôvodného TS  $A$ , musíme mať informáciu o tom, čo je na páske o  $k$  poličok vľavo i vpravo od polička, na ktorom je umiestnená hlava. Nech používame ľubovoľne veľkú kompresiu pásky, môže sa stať, že hlava simulovaného TS sa bude nachádzať pri okraji úseku pásky, ktorý je zapísaný na príslušnom poličku. V takomto prípade sa budeme musieť pozrieť aj na susedné poličko.

**Veta 7.2.2 (O lineárnom zrýchlení)** Nech  $A$  je ľubovoľný deterministický Turingov stroj, ktorý je  $T(n)$  časovo ohraničený a nech

$$\lim_{n \rightarrow \infty} \frac{T(n)}{n} = \infty$$

Nech  $k > 1$  je ľubovoľná konštantá. Potom existuje deterministický Turingov stroj  $A'$  taký, že  $L(A) = L(A')$ ,  $A'$  je  $T'(n)$  časovo ohraničený a od nejakého  $n_0$  je  $T'(n') \leq T(n)/k$ .

**Dôkaz.** Uvedomme si najskôr, že krátke slová nám robia pri urýchlení problém. Napr. TS, ktorý akceptuje slovo *aaaaaa* na 7 krokov (ale neakceptuje *aaaab*) by ho po 10-násobnom urýchlení mal akceptovať na 1 krok, čo ale nie je možné. Zostrojíme preto TS, ktorý bude  $k$ -krát rýchlejší len na dostatočne dlhých slovách. Uvedomte si, že nie je problém tento TS na záver upraviť tak, že všetky slová z  $L(A)$  kratšie ako  $n_0$  do neho „zadrôtujeme“, takže pre  $n < n_0$  bude  $T'(n) = n + 1$ . Kedže vo všeobecnosti TS musí vstup dočítať, nič lepšie sa dosiahnuť nedá. Ukážeme teraz, ako zostrojiť hľadaný  $A'$ .

Nech platia uvedené predpoklady. Turingov stroj  $A'$  bude pracovať nasledovne: Najprv reorganizuje pásku do blokov po  $m$  poličkach. (Číslo  $m$  je vhodná konštantá. Neskôr ukážeme, aký je vzťah medzi  $k$  a  $m$ .) To znamená, že každých  $m$  znakov vstupu prepíše na jeden znak na jednej pracovnej páske. Pracovné pásky budú mať abecedu  $\Gamma_A^m$ .

Dalej  $A'$  simuluje prácu pôvodného automatu  $A$  s tým, že sa rozšíri množina stavov na  $K \times \{1, 2, \dots, m\}$ , aby sme si pamätali číslo skutočného polička, na ktorom by bol TS  $A$  v rámci aktuálneho bloku. TS  $A'$  spraví 4 kroky: Prečíta poličko, resp. blok, na ktorom sa nachádza, posunie sa doľava a tiež toto poličko prečíta, potom urobí dva kroky doprava, aby aj tento blok prečítal a vráti sa späť. Určíte si teraz v stave pamäťa aspoň  $m$  poličok pôvodnej pásky naľavo aj napravo od pozície hlavy, preto vie odsimulovať nasledujúcich  $m$  krokov TS  $A$ . Ten na  $m$  krokov určíte nezapisoval do blokov vzdialenosť od aktuálneho viac ako o 1. Na tento zápis potrebujeme opäť 4 kroky. To znamená, že pôvodných  $m$  krokov TS  $A$  vieme simulať ôsmimi krokmi TS  $A'$ . Ak má  $A'$  byť od  $A$   $k$ -krát rýchlejší, stačí zvoliť  $m = 8k + h$ , kde  $h$  je konštantá „na pokrytie režijných nákladov na začiatku“.

Aby bol dôkaz kompletný, formálne dokážeme, že  $m = 8k + 47$  naozaj stačí. Čas behu TS  $A'$  na vstupe veľkosti  $n$  bude

$$T'(n) = n + \left\lceil \frac{n}{m} \right\rceil + \left\lceil \frac{8T(n)}{m} \right\rceil$$

(Skomprimujeme vstup, vrátime sa hlavou na jeho začiatok a simulujeme  $A$ .) Je

$$T'(n) < n + \frac{n+m}{m} + \frac{8T(n)+m}{m} = \frac{8T(n)+2m+(m+1)n}{m} < \frac{8T(n)+(m+1)(n+2)}{m}$$

Treba ukázať, že od nejakého  $n_0$  je  $T'(n) \leq T(n)/k$ . Na to stačí, aby

$$\frac{8T(n)+(m+1)(n+2)}{m} < \frac{T(n)}{k}$$

odkiaľ po prepísaní dostávame, že má platiť

$$\frac{k(8k+48)}{47}(n+2) < T(n)$$

Kedže však  $\lim_{n \rightarrow \infty} (T(n)/n) = \infty$ , také  $n_0$  zjavne existuje.

V nasledujúcom texte ukážeme, že i počet použitých pásov možno redukovať. Bohužiaľ, v niektorých prípadoch tým čosi stratíme.

**Veta 7.2.3** (*O redukcii počtu pások z k na jednu pre priestor*) K ľubovoľnému deterministickému TS A, ktorý má k pracovných pások a je  $S(n)$  priestorovo ohraničený existuje ekvivalentný TS A', ktorý je tiež  $S(n)$  priestorovo ohraničený a má len jednu pracovnú pásku.

**Dôkaz.** Stačí použiť viacstopú pracovnú pásku. (Jedna možnosť je mať na každej stopre zaznačenú aj pozíciu hlavy, druhá možnosť je posúvať jednotlivé stopy tak, aby boli všetky hlavy stále nad tým istým poličkom.) Uvedomte si, že simulácia jedného kroku pôvodného TS nám môže trvať až  $S(n)$  krokov, keďže musíme prejsť celú pásku a nájsť všetky simulované hlavy. (Resp.  $kS(n)$  krokov, lebo musíme poposúvať jednotlivé pásky.)

**Dôsledok 7.2.4** (*O redukcii počtu pások z k na jednu pre čas*) K ľubovoľnému deterministickému TS A, ktorý má vstupnú a k pracovných pások a je  $T(n)$  časovo ohraničený existuje ekvivalentný klasický TS A' (s jedinou páskou, na ktorú môže aj zapisovať), ktorý je  $T^2(n)$  časovo ohraničený.

**Dôkaz.** Keďže A je  $T(n)$  časovo ohraničený, je aj  $T(n)$  priestorovo ohraničený. Zostrojíme ekvivalentný TS s jednou páskou podľa dôkazu predchádzajúcej vety. Ľahko nahliadneme, že je  $T^2(n)$  časovo ohraničený.

Naša redukcia počtu pások na jednu sice spôsobí značné spomalenie, redukcia na dve pásky však na tom bude oveľa lepšie.

**Veta 7.2.5** (*O redukcii počtu pások z k na dve pre čas*) K ľubovoľnému deterministickému k-páskovemu Turingovmu stroju A pracujúcemu v čase  $T(n)$  existuje dvojpáskový deterministický Turingov stroj A' pracujúci v čase  $T(n) \log(T(n))$  taký, že  $L(A) = L(A')$ .

**Dôkaz.** Použijeme druhú pásku ako akúsi cache pamäť. Použijeme metódu posunu stopy pásky namiesto posunu hlavy. No nebudeme posúvať celú stopu, ale väčšinou len jej malý kúsok. Podstatá použitej myšlienky spočíva v tom, že si pásku rozdelíme na myslené bloky o veľkostiach  $1, 2, 4, 8, \dots, 2^k$  poličok. Fungovanie automatu A' vysvetlíme na príklade.

Príklad sem časom pribudne. Možno :-)

**Poznámka 7.2.1** Mohlo by nás zaujímať, či sa podobný výsledok nedá dosiahnuť aj pri redukcii z k pások na jednu. (Čo ak nami ukázaný postup nebol optimálny?) Ukážeme ale príklad jazyka, ktorý vieme na dvojpáskovom TS akceptovať v lineárnom čase, zatiaľ čo na klasickom jednopáskovom TS potrebujeme kvadratický čas. Takýmto jazykom je  $L = \{wcw^R \mid w \in \{a, b\}^*\}$ .

Dôkaz, že klasický TS potrebuje na jeho akceptovanie kvadratický čas, presahuje rámec tohto textu. Jeho myšlienka je v zostrojení prechodových postupností (podobne ako sme urobili v dôkaze vety 2.11.4 o ekvivalencii NKA a 2NKA) a dokázanie, že pre vhodné rôzne slová musia byť prechodové postupnosti navzájom rôzne. Z toho vyplýva, že niektoré z vybraných slov majú veľký súčet dĺžok prechodových postupností, lebo krátkych prechodových postupností je málo. Ale časová zložitosť výpočtu TS na slove je práve súčet dĺžok prechodových postupností pre všetky poličky pásky.

**Dôsledok 7.2.6** Vo všeobecnosti nevieme spraviť redukciu z k pások na jednu lepšie ako za cenu kvadratickeho spomalenia.

## 6. Triedy P a NP [polynomiálna redukovateľnosť. Cook-Levinova veta a ďalšie NP-úplné problémy]

### 7.6 Redukcie, ťažké a úplné problémy

Túto časť začneme príkladom, na ktorom ukážeme koncept, ktorý neskôr vo všeobecnejšej podobe formálne definujeme.

**Príklad 7.6.1** Uvažujme nasledovný problém, známy pod názvom *SAT*: Daná je boolovská formula s  $n$  premennými, rozhodnite, či existuje ohodnotenie premenných, pri ktorom je splnená.<sup>3</sup>

Zjavne  $SAT \in NP$ , stačí nedeterministicky uhádnuť jedno správne ohodnotenie premenných a overiť, že pre uhádnuté ohodnotenie je naozaj daná formula splnená.

V rokoch 1971 až 1973 Cook a Levin publikovali nasledujúci výsledok: Majme ľubovoľný problém, ktorý patrí do  $NP$ . Potom k tomuto problému existuje nedeterministický TS, ktorý ho v polynomiálnom čase rozhoduje. K tomuto stroju a ľubovoľnému danému slovu vieme v polynomiálnom čase (od súčtu veľkosti popisu stroja a dĺžky slova) zestrojiť boolovskú formulu, ktorá je splnitelná práve vtedy, ak daný stroj dané slovo akceptuje.

Čo nám tento výsledok hovorí?

Ak by problém *SAT* bol v  $P$ , tak by sme každý problém z  $NP$  vedeli riešiť v deterministickom polynomiálnom čase – zoberieme nedeterministický stroj pre daný problém, k tomu v polynomiálnom čase zstrojíme ekvivalentnú boolovskú formulu, a následne v polynomiálnom čase rozhodneme, či je splnitelná.<sup>4</sup>

Ak by teda *SAT* patril do  $P$ , tak každý problém z  $NP$  patrí do  $P$ , a teda  $P = NP$ . Inými slovami, v  $NP$  neexistuje žiadny problém „ťažší“ ako *SAT*.

Ukážeme teraz všeobecnejšiu definíciu, ktorá zachytáva túto myšlienku.

**Definícia 7.6.1** Nech  $T$  a  $R$  sú triedy zložitosti a  $X$  problém. Hovoríme, že problém  $X$  je  $T$ -ťažký (pri  $R$ -obmedzenej redukcii), ak pre každý problém  $Y \in T$  existuje redukcia z  $Y$  na  $X$ , ktorá je v  $R$ .

Hovoríme, že problém  $X$  je  $T$ -úplný (pri  $R$ -obmedzenej redukcii), ak je  $T$ -ťažký a patrí do  $T$ .

Ak je z kontextu jasné, ako obmedzenú redukciu používame, môžeme túto časť vyniechať a hovoriť jednoducho o  $T$ -ťažkých a  $T$ -úplných problémoch.

**Príklad 7.6.2** Vyššie spomínaný jazyk *SAT* je  $NP$ -úplný (pri redukcii z  $P$ , teda deterministickej a polynomiálne časovo obmedzenej).

**Poznámka 7.6.1** Sama o sebe nám definícia  $T$ -ťažkých a  $T$ -úplných problémov veľa nehovorí. Na to, aby bola užitočná, potrebujeme vhodne zvoliť triedu  $R$ , teda to, aké redukcie povoľujeme.

Redukcie by vo všeobecnosti mali byť v porovnaní s triedou  $T$  ľahké. Pri voľbe, aké redukcie povoliť a aké nie, chceme dosiahnuť, aby platilo: „Ak vieme efektívne riešiť nejaký  $T$ -úplný problém  $X$ , tak potom vieme efektívne riešiť každý problém z  $T$  tak, že ho zredukujeme na  $X$  a následne vyriešime  $X$ .“

**Poznámka 7.6.2** Majme triedu  $T$  a podriedu  $S$ , o ktorej nás zaujíma, či  $S = T$ . Zvoľme teda obmedzenie na redukciu tak, aby platilo, že každý algoritmus tvaru „zredukuj problém  $Y$  na  $X$ , a následne vyrieš  $X$  algoritmom z  $S$ “ patril do  $S$ .

Ak sa nám toto podarí, dosiahneme, že  $T$ -úplné problémy budú najvhodnejšími kandidátkami na zistenie, či  $S = T$ : Ak rovnosť neplatí, všetky  $T$ -úplné problémy sú nutne protipríkladmi. Na druhej strane, akonáhle by sme o nejakom  $T$ -úplnom probléme ukázali, že patrí do  $S$ , vyplynulo by z toho automaticky, že  $S = T$ .

V každej z tried  $NL$ ,  $P$ ,  $NP$  a  $PSPACE$  poznáme problémy, ktoré sú pre ňu pri vhodnej redukcii úplné, a teda v určitom zmysle najťažšie v nej. Ukažeme si príklady takýchto problémov:

- **PATH:** Daný je orientovaný graf  $G$  a dva jeho vrcholy  $s$  a  $t$ , rozhodnite, či v  $G$  existuje cesta z  $s$  do  $t$ .  
*PATH* patrí do  $NL$ , a ak by patril do  $L$ , tak  $L = NL$ .
- **HALTN:** Daný je deterministický Turingov stroj  $A$ , vstup  $w$  a unárne zapísané číslo  $n$ , rozhodnite, či  $A$  na  $w$  po nanajvýš  $n$  krokoch zastane.  
*HALTN* patrí do  $P$ , a ak by patril do  $NL$ , tak  $NL = P$ .
- **SAT:** Daná je boolovská formula s  $n$  premennými, rozhodnite, či existuje ohodnenie premenných, pri ktorom je splnená.  
*SAT* patrí do  $NP$ , a ak by patril do  $P$ , tak  $P = NP$ .
- **TQBF:** Daná je úplne kvantifikovaná boolovská formula s  $n$  premennými, rozhodnite, či je pravdivá.  
*TQBF* patrí do  $PSPACE$ , a ak by patril do  $NP$ , tak  $NP = PSPACE$ .  
 Navyše vieme, že *TQBF* nepatrí do  $NL$ .

**Definícia 7.3.2** Špeciálne nás budú neskôr zaujímať triedy, kde je čas alebo priestor obmedzený nejakým polynómom, prípadne inou jednoduchou funkciou, pričom nás bude zaujímať len typ tejto funkcie, nie konkrétnie koeficienty v nej. Najznámejšie takéto triedy jazykov sú:

$$\begin{aligned}
 P &= \bigcup_{f \text{ je polynóm}} DTIME(f(n)) = \bigcup_{k>0} DTIME(O(n^k)) = DTIME(n^{O(1)}) \\
 NP &= \bigcup_{f \text{ je polynóm}} NTIME(f(n)) = \bigcup_{k>0} NTIME(O(n^k)) = NTIME(n^{O(1)}) \\
 PSPACE &= \bigcup_{k>0} DSPACE(n^k) = DSPACE(n^{O(1)}) \\
 NPSPACE &= \bigcup_{k>0} NSPACE(n^k) = NSPACE(n^{O(1)}) \\
 L &= DSPACE(\log n) \\
 NL &= NSPACE(\log n)
 \end{aligned}$$

Teraz ukážeme, že niektoré z takto definovaných tried jazykov už poznáme pod inými menami.

**Veta 7.3.1**  $\mathcal{L}_{ECS} = NSPACE(n) = NSPACE(O(n))$

**Dôkaz.** Prvá rovnosť vyplýva z definície LBA, druhá z vety 7.2.1 o kompresii pásky. Táto veta poukazuje na skutočnosť, že LBA je skutočne Turingov stroj s lineárne obmedzeným priestorom.

**Veta 7.3.2**  $\mathcal{R} = NSPACE(1) = NSPACE(O(1)) = DSPACE(1) = DSPACE(O(1))$

**Dôkaz.** Podľa lemy 7.5.1 a definícií uvedených tried zjavne  $NSPACE(O(1)) \supseteq DSPACE(O(1))$ ,  $DSPACE(O(1)) \supseteq DSPACE(1)$  a  $NSPACE(O(1)) \supseteq NSPACE(1)$ . Platí  $DSPACE(1) \supseteq \mathcal{R}$ , lebo pre každý regulárny jazyk existuje DTS (simulujúci príslušný DKA), ktorý ho akceptuje dokonca bez používania pomocnej pamäte. A platí aj  $\mathcal{R} \supseteq NSPACE(O(1))$ . Keď totiž máme NTS, ktorý na každej z  $k$  pracovných pások použije najviac  $c$  poličok, vieme si týchto  $kc$  poličok pamätať v stave dvojsmerného nedeterministického konečného automatu, ktorým príslušný TS ľahko odsimulujeme. Z ukázaných inkluzií už vyplýva platnosť všetkých rovností.

**Poznámka 7.3.4** A zjavne platí aj  $\mathcal{R} = DSPACETIME(1, n)$ .

**Poznámka 7.3.5** Z triviálne platnej inkluzie  $\mathcal{L}_{CF} \subseteq \mathcal{L}_{ECS}$  a rovnosti  $\mathcal{L}_{ECS} = NSPACE(n)$  dostávame  $\mathcal{L}_{CF} \subseteq NSPACE(n)$ . Rozmyslite si, ako by ste túto inkluziu dokázali priamo.