

# FIRST, FOLLOW, Zásobníkové automaty

Ing. Viliam Hromada, PhD.

C-510

Ústav informatiky a matematiky  
FEI STU

`viliam.hromada@stuba.sk`



## FIRST príklad č. 1

Nech je daná redukovaná gramatika  $G = (\{S, A, B, C\}, \{a, b, c\}, P, S)$  a pravidlá  $P$ :

$$S \rightarrow ABC \mid aSb$$

$$A \rightarrow aAb \mid c \mid C$$

$$B \rightarrow BabB \mid AA$$

$$C \rightarrow \varepsilon \mid baCab$$

Aká je množina  $FIRST(X)$  pre symboly gramatiky?



## FIRST príklad č. 1 - množina $N_\epsilon$

Najprv vyšetříme množinu  $N_\epsilon$ . Na začiatku  $N_\epsilon = \emptyset$ .

1. Ako prvý do  $N_\epsilon$  patrí neterminál  $C$ , pretože priamo na základe pravidla  $C \rightarrow \epsilon$  vidíme, že  $C \Rightarrow \epsilon$ , teda určite  $N_\epsilon = \{C\}$
2. Ďalej, keď vieme, že  $C \in N_\epsilon$ , tak na základe pravidla  $A \rightarrow C$  vidíme, že  $A \Rightarrow C \Rightarrow \epsilon$ , teda  $A \Rightarrow^* \epsilon$ , a teda  $N_\epsilon = \{A, C\}$ .
3. Ďalej, keď vieme, že  $A, C \in N_\epsilon$ , tak na základe pravidla  $B \rightarrow AA$  vidíme, že  $B \Rightarrow AA \Rightarrow A \Rightarrow \epsilon$ , teda  $B \Rightarrow^* \epsilon$ , a teda  $N_\epsilon = \{B, A, C\}$ .
4. Ďalej, keď vieme, že  $B, A, C \in N_\epsilon$ , tak na základe pravidla  $S \rightarrow ABC$  vidíme, že  $S \Rightarrow ABC \Rightarrow BC \Rightarrow C \Rightarrow \epsilon$ , teda  $S \Rightarrow^* \epsilon$ , a teda  $N_\epsilon = \{S, B, A, C\}$ .

Výsledná množina  $N_\epsilon = \{S, A, B, C\}$ .



# Množina *FIRST* - príklad

*FIRST*( $X$ ):

<i>FIRST</i>	<i>S</i>	<i>A</i>	<i>B</i>	<i>C</i>	<i>a</i>	<i>b</i>	<i>c</i>
Terminály:					<i>a</i>	<i>b</i>	<i>c</i>
Pravidlá $A \rightarrow a\alpha$	<i>a</i>	<i>a, c</i>		<i>b</i>			
Neterminály $A \in N_\epsilon$	$\epsilon$	$\epsilon$	$\epsilon$	$\epsilon$			
1. iterácia:	<i>c, b</i>	<i>b</i>	<i>a, b, c</i>				
2. iterácia:							
Celkom:	<i>a, b, c, <math>\epsilon</math></i>	<i>a, b, c, <math>\epsilon</math></i>	<i>a, b, c, <math>\epsilon</math></i>	<i>b, <math>\epsilon</math></i>			



## FOLLOW príklad č. 1

Nech je daná redukovaná gramatika  $G = (\{S, A, B, C\}, \{a, b, c\}, P, S)$  a pravidlá  $P$ :

$$S \rightarrow ABC \mid aSb$$

$$A \rightarrow aAb \mid c \mid C$$

$$B \rightarrow BabB \mid AA$$

$$C \rightarrow \varepsilon \mid baCab$$

Už vieme, že:

$$FIRST(S) = \{\varepsilon, a, b, c\}$$

$$FIRST(A) = \{\varepsilon, a, b, c\}$$

$$FIRST(B) = \{\varepsilon, a, b, c\}$$

$$FIRST(C) = \{\varepsilon, b\}$$

Ako vyzerajú množiny *FOLLOW* pre neterminály gramatiky?



# Množina FOLLOW

(podčiarknutý je neterminál, na základe ktorého je pravidlo aktuálne vyšetrované a zelenou farbou časť za príslušným neterminálom, t.j. na prednáške označená ako  $\beta$ )  
 (červenou sú označené terminály, ktoré sa už v danej množine nachádzajú)

FOLLOW	S	A	B	C	Dôvod
Počiatkový neterminál	$\epsilon$				
1. iterácia:					
$S \rightarrow \underline{A}BC$ $S \rightarrow \underline{A}BC$		$a, b, c$ $\epsilon$			$a, b, c \in FIRST(BC)$ $\epsilon \in FIRST(BC)$ a teda $FOLLOW(S) \subseteq FOLLOW(A)$
$S \rightarrow \underline{A}BC$ $S \rightarrow \underline{A}BC$			$b$ $\epsilon$		$b \in FIRST(C)$ $\epsilon \in FIRST(C)$ a teda $FOLLOW(S) \subseteq FOLLOW(B)$
$S \rightarrow \underline{ABC}\epsilon$				$\epsilon$	$FIRST(\epsilon) = \{\epsilon\}$ a teda $FOLLOW(S) \subseteq FOLLOW(C)$
$S \rightarrow a\underline{S}b$	$b$				$FIRST(b) = \{b\}$
$A \rightarrow a\underline{A}b$		$b$			$FIRST(b) = \{b\}$
$A \rightarrow \underline{C}\epsilon$				$a, b, c, \epsilon$	$FIRST(\epsilon) = \{\epsilon\}$ a teda $FOLLOW(A) \subseteq FOLLOW(C)$
$B \rightarrow \underline{B}abB$			$a$		$FIRST(abB) = \{a\}$
$B \rightarrow \underline{B}abB\epsilon$			$a, b, \epsilon$		$FIRST(\epsilon) = \{\epsilon\}$ a teda $FOLLOW(B) \subseteq FOLLOW(B)$
$B \rightarrow \underline{A}A$ $B \rightarrow \underline{A}A$		$a, b, c$ $a, b, \epsilon$			$a, b, c \in FIRST(A)$ $\epsilon \in FIRST(A)$ a teda $FOLLOW(B) \subseteq FOLLOW(A)$
$B \rightarrow \underline{AA}\epsilon$		$a, b, \epsilon$			$FIRST(\epsilon) = \{\epsilon\}$ a teda $FOLLOW(B) \subseteq FOLLOW(A)$
$C \rightarrow ba\underline{C}ab$				$a$	$FIRST(ab) = \{a\}$

# Množina FOLLOW

(podčiarknutý je neterminál, na základe ktorého je pravidlo aktuálne vyšetrované a zelenou farbou časť za príslušným neterminálom)

(červenou sú označené terminály, ktoré sa už v danej množine nachádzajú)

FOLLOW	S	A	B	C	Dôvod
Po 1. iterácii	$\epsilon, b$	$a, b, c, \epsilon$	$a, b, \epsilon$	$a, b, c, \epsilon$	
2. iterácia:					
$S \rightarrow \underline{A}BC$ $S \rightarrow \underline{A}BC$		$a, b, c$ $\epsilon, b$			$a, b, c \in \text{FIRST}(BC)$ $\epsilon \in \text{FIRST}(BC)$ a teda $\text{FOLLOW}(S) \subseteq \text{FOLLOW}(A)$
$S \rightarrow \underline{A}BC$ $S \rightarrow \underline{A}BC$			$b$ $\epsilon, b$		$b \in \text{FIRST}(C)$ $\epsilon \in \text{FIRST}(C)$ a teda $\text{FOLLOW}(S) \subseteq \text{FOLLOW}(B)$
$S \rightarrow \underline{ABC}\epsilon$				$\epsilon, b$	$\text{FIRST}(\epsilon) = \{\epsilon\}$ a teda $\text{FOLLOW}(S) \subseteq \text{FOLLOW}(C)$
$S \rightarrow a\underline{S}b$	$b$				$\text{FIRST}(b) = \{b\}$
$A \rightarrow a\underline{A}b$		$b$			$\text{FIRST}(b) = \{b\}$
$A \rightarrow \underline{C}\epsilon$				$a, b, c, \epsilon$	$\text{FIRST}(\epsilon) = \{\epsilon\}$ a teda $\text{FOLLOW}(A) \subseteq \text{FOLLOW}(C)$
$B \rightarrow \underline{B}abB$			$a$		$\text{FIRST}(abB) = \{a\}$
$B \rightarrow \underline{B}abB\epsilon$			$a, b, \epsilon$		$\text{FIRST}(\epsilon) = \{\epsilon\}$ a teda $\text{FOLLOW}(B) \subseteq \text{FOLLOW}(B)$
$B \rightarrow \underline{A}A$ $B \rightarrow \underline{A}A$		$a, b, c$ $a, b, \epsilon$			$a, b, c \in \text{FIRST}(A)$ $\epsilon \in \text{FIRST}(A)$ a teda $\text{FOLLOW}(B) \subseteq \text{FOLLOW}(A)$
$B \rightarrow \underline{AA}\epsilon$		$a, b, \epsilon$			$\text{FIRST}(\epsilon) = \{\epsilon\}$ a teda $\text{FOLLOW}(B) \subseteq \text{FOLLOW}(A)$
$C \rightarrow ba\underline{C}ab$				$a$	$\text{FIRST}(ab) = \{a\}$



Vidíme, že v druhej iterácii už nedošlo k nájdeniu nových symbolov, teda algoritmus končí a výsledok:

$$FOLLOW(S) = \{\varepsilon, b\}$$

$$FOLLOW(A) = \{\varepsilon, a, b, c\}$$

$$FOLLOW(B) = \{\varepsilon, a, b\}$$

$$FOLLOW(C) = \{\varepsilon, a, b, c\}$$





## FIRST, FOLLOW příklad č. 2

### Príklad:

Gramatika  $G = (\{ \langle \text{program} \rangle, \langle \text{příkaz} \rangle, \langle \text{příkazy} \rangle \}, \{ \mathbf{begin}, \mathbf{end}, \mathbf{p}, ; \}, P, \langle \text{program} \rangle)$ , pravidlá  $P$ :

$\langle \text{program} \rangle \rightarrow \mathbf{begin} \langle \text{příkazy} \rangle \mathbf{end}$

$\langle \text{příkazy} \rangle \rightarrow \langle \text{příkaz} \rangle \langle \text{příkazy} \rangle$

$\langle \text{příkazy} \rangle \rightarrow \varepsilon$

$\langle \text{příkaz} \rangle \rightarrow \mathbf{p};$

$\langle \text{příkaz} \rangle \rightarrow \mathbf{begin} \langle \text{příkazy} \rangle \mathbf{end}$

Určte množiny *FIRST* a *FOLLOW* pre symboly gramatiky.



## FIRST príklad č. 2 - množina $N_\epsilon$

Najprv vyšetříme množinu  $N_\epsilon$ . Na začiatku  $N_\epsilon = \emptyset$ .

1. Ako prvý do  $N_\epsilon$  patrí neterminál  $\langle \text{príkazy} \rangle$ , pretože priamo na základe pravidla  $\langle \text{príkazy} \rangle \rightarrow \epsilon$  vidíme, že  $\langle \text{príkazy} \rangle \Rightarrow \epsilon$ , teda určite  $N_\epsilon = \{ \langle \text{príkazy} \rangle \}$
2. Ďalej, keď vieme, že  $\langle \text{príkazy} \rangle \in N_\epsilon$ , tak by sme hľadali v pravidlách pravidlo, ktoré má na pravej strane len reťazec zložený z neterminálu  $\langle \text{príkazy} \rangle$ . Keďže také pravidlo v gramatike nemáme, tak množina  $N_\epsilon$  už nebude obsahovať ďalšie neterminály.

Výsledná množina  $N_\epsilon = \{ \langle \text{príkazy} \rangle \}$ .



<i>FIRST</i>	<program>	<príkazy>	<príkaz>	<b>begin</b>	<b>end</b>	<b>p</b>	<b>;</b>
Terminály:				<b>begin</b>	<b>end</b>	<b>p</b>	<b>;</b>
Pravidlá $A \rightarrow a\alpha$ :	<b>begin</b>		<b>p, begin</b>				
$A \in N_\epsilon$		$\epsilon$					
1. iterácia		<b>p, begin</b>					
2. iterácia							
Celkom:	<b>begin</b>	<b>p, begin, <math>\epsilon</math></b>	<b>p, begin</b>	<b>begin</b>	<b>end</b>	<b>p</b>	<b>;</b>

<i>FOLLOW</i>	<program>	<príkazy>	<príkaz>
Začiatkový symbol	$\epsilon$		
1. iterácia <program> → <b>begin</b> <príkazy> <b>end</b>		<b>end</b>	
<príkazy> → <príkaz><príkazy>			<b>p, begin, end</b>
<príkazy> → <príkaz><príkazy>			
<príkaz> → <b>begin</b> <príkazy> <b>end</b>			
2. iterácia <program> → <b>begin</b> <príkazy> <b>end</b>			
<príkazy> → <príkaz><príkazy>			
<príkazy> → <príkaz><príkazy>			
<príkaz> → <b>begin</b> <príkazy> <b>end</b>			
<b>Celkom:</b>	$\epsilon$	<b>end</b>	<b>p, begin, end</b>

## FIRST, FOLLOW č. 3

Je daná redukovaná gramatika  $G = (N, T, P, S)$ . Zistite, ako vyzerá množina *FIRST* symbolov gramatiky, resp. *FOLLOW* neterminálov gramatiky. V gramatike  $N = \{S, A, B, C\}$ ,  $T = \{a, b, c\}$ ,  $S$  je počiatkový neterminál, pravidlá  $P$ :

- $S \rightarrow AbBB$
- $A \rightarrow CC \mid cSA$
- $B \rightarrow aCa \mid bb$
- $C \rightarrow \varepsilon \mid BC$



## FIRST príklad č. 3 - množina $N_\epsilon$

Najprv vyšetříme množinu  $N_\epsilon$ . Na začiatku  $N_\epsilon = \emptyset$ .

1. Ako prvý do  $N_\epsilon$  patrí neterminál  $C$ , pretože priamo na základe pravidla  $C \rightarrow \epsilon$  vidíme, že  $C \Rightarrow \epsilon$ , teda určite  $N_\epsilon = \{C\}$
2. Ďalej, keď vieme, že  $C \in N_\epsilon$ , tak na základe pravidla  $A \rightarrow CC$  vidíme, že  $A \Rightarrow CC \Rightarrow C \Rightarrow \epsilon$ , teda  $A \Rightarrow^* \epsilon$ , a teda  $N_\epsilon = \{A, C\}$ .
3. Ďalej by sme hľadali v pravidlách pravidlo, ktoré ma na pravej strane reťazec zložený len z neterminálov  $A, C$ . Keďže také pravidlo tam nemáme, hľadanie množiny  $N_\epsilon$  končí.

Výsledná množina  $N_\epsilon = \{A, C\}$ .



(červenou sú označené terminály, ktoré by mali byť pridané do príslušnej množiny *FIRST* na základe aktuálne vyšetrovaného pravidla, ale už sa v danej množine nachádzajú)

<i>FIRST</i>	<i>S</i>	<i>A</i>	<i>B</i>	<i>C</i>	<i>a</i>	<i>b</i>	<i>c</i>
Terminály:					<i>a</i>	<i>b</i>	<i>c</i>
Pravidlá $A \rightarrow a\alpha$ :		<i>c</i>	<i>a, b</i>				
$A \in N_\epsilon$		$\epsilon$		$\epsilon$			
1. iterácia							
$S \rightarrow AbBB$	<i>c, b</i>						
$A \rightarrow CC$		$\epsilon$					
$C \rightarrow BC$				<i>a, b</i>			

(červenou sú označené terminály, ktoré by mali byť pridané do príslušnej množiny *FIRST* na základe aktuálne vyšetřovaného pravidla, ale už sa v danej množine nachádzajú)

<i>FIRST</i>	<i>S</i>	<i>A</i>	<i>B</i>	<i>C</i>	<i>a</i>	<i>b</i>	<i>c</i>
Po 1. iterácii	<i>b, c</i>	<i>c, ε</i>	<i>a, b</i>	<i>a, b, ε</i>	<i>a</i>	<i>b</i>	<i>c</i>
2. iterácia							
$S \rightarrow AbBB$	<i>c, b</i>						
$A \rightarrow CC$		<i>a, b, ε</i>					
$C \rightarrow BC$				<i>a, b</i>			



(červenou sú označené terminály, ktoré by mali byť pridané do príslušnej množiny *FIRST* na základe aktuálne vyšetřovaného pravidla, ale už sa v danej množine nachádzajú)

<i>FIRST</i>	<i>S</i>	<i>A</i>	<i>B</i>	<i>C</i>	<i>a</i>	<i>b</i>	<i>c</i>
Po 2. iterácii	<i>b, c</i>	<i>a, b, c, ε</i>	<i>a, b</i>	<i>a, b, ε</i>	<i>a</i>	<i>b</i>	<i>c</i>
3. iterácia							
$S \rightarrow AbBB$	<i>a, c, b</i>						
$A \rightarrow CC$		<i>a, b, ε</i>					
$C \rightarrow BC$				<i>a, b</i>			

(červenou sú označené terminály, ktoré by mali byť pridané do príslušnej množiny *FIRST* na základe aktuálne vyšetřovaného pravidla, ale už sa v danej množine nachádzajú)

<i>FIRST</i>	<i>S</i>	<i>A</i>	<i>B</i>	<i>C</i>	<i>a</i>	<i>b</i>	<i>c</i>
Po 3. iterácii	<i>a, b, c</i>	<i>a, b, c, ε</i>	<i>a, b</i>	<i>a, b, ε</i>	<i>a</i>	<i>b</i>	<i>c</i>
4. iterácia							
$S \rightarrow AbBB$	<i>a, c, b</i>						
$A \rightarrow CC$		<i>a, b, ε</i>					
$C \rightarrow BC$				<i>a, b</i>			

Vidíme, že v štvrtej iterácii sme nedostali žiaden nový symbol do žiadnej množiny *FIRST*, algoritmus teda končí.

Výsledné množiny *FIRST*:

1.  $FIRST(S) = \{a, b, c\}$
2.  $FIRST(A) = \{a, b, c, \varepsilon\}$
3.  $FIRST(B) = \{a, b\}$
4.  $FIRST(C) = \{a, b, \varepsilon\}$



# Množina FOLLOW

(podčiarknutý je neterminál, na základe ktorého je pravidlo aktuálne vyšetované a zelenou farbou časť za príslušným neterminálom)

(červenou sú označené terminály, ktoré by mali byť pridané do príslušnej množiny FOLLOW na základe aktuálne vyšetovaného pravidla, ale už sa v danej množine nachádzajú)

FOLLOW	S	A	B	C	Dôvod
Počiatkový neterminál	$\epsilon$				
1. iterácia:					
$S \rightarrow \underline{A}bBB$		$b$			$FIRST(bBB) = \{b\}$
$S \rightarrow Ab\underline{B}B$			$a, b$		$FIRST(B) = \{a, b\}$
$S \rightarrow AbBB\underline{\epsilon}$			$\epsilon$		$FIRST(\epsilon) = \{\epsilon\}$ a teda $FOLLOW(S) \subseteq FOLLOW(B)$
$A \rightarrow \underline{C}C$				$a, b$	$a, b \in FIRST(C)$
$A \rightarrow CC\underline{\epsilon}$				$b$	$\epsilon \in FIRST(C)$ a teda $FOLLOW(A) \subseteq FOLLOW(C)$
$A \rightarrow c\underline{S}A$	$a, b, c$				$a, b, c \in FIRST(A)$
$A \rightarrow aSA\underline{\epsilon}$		$b$			$\epsilon \in FIRST(A)$ a teda $FOLLOW(A) \subseteq FOLLOW(S)$
$B \rightarrow a\underline{C}a$			$a$		$FIRST(a) = \{a\}$
$C \rightarrow \underline{B}C$			$a, b$		$a, b \in FIRST(C)$
$C \rightarrow BC\underline{\epsilon}$			$a, b$		$\epsilon \in FIRST(C)$ a teda $FOLLOW(C) \subseteq FOLLOW(B)$
$C \rightarrow BC\underline{\epsilon}$				$a, b$	$FIRST(\epsilon) = \{\epsilon\}$ a teda $FOLLOW(C) \subseteq FOLLOW(C)$

# Množina FOLLOW

(podčiarknutý je neterminál, na základe ktorého je pravidlo aktuálne vyšetrované a zelenou farbou časť za príslušným neterminálom)

(červenou sú označené terminály, ktoré by mali byť pridané do príslušnej množiny FOLLOW na základe aktuálne vyšetrovaného pravidla, ale už sa v danej množine nachádzajú)

FOLLOW	S	A	B	C	Dôvod
Po 1. iterácii	$\epsilon, a, b, c$	$b$	$a, b, \epsilon$	$a, b$	
2. iterácia:					
$S \rightarrow \underline{A}bBB$		$b$			$FIRST(bBB) = \{b\}$
$S \rightarrow Ab\underline{B}B$			$a, b$		$FIRST(B) = \{a, b\}$
$S \rightarrow AbBB\underline{\epsilon}$			$c, \epsilon, a, b$		$FIRST(\epsilon) = \{\epsilon\}$ a teda $FOLLOW(S) \subseteq FOLLOW(B)$
$A \rightarrow \underline{C}C$				$a, b$ $b$	$a, b \in FIRST(C)$ $\epsilon \in FIRST(C)$ a teda $FOLLOW(A) \subseteq FOLLOW(C)$
$A \rightarrow CC\underline{\epsilon}$				$b$	$FIRST(\epsilon) = \{\epsilon\}$ a teda $FOLLOW(A) \subseteq FOLLOW(C)$
$A \rightarrow c\underline{S}A$	$a, b, c$ $b$				$a, b, c \in FIRST(A)$ $\epsilon \in FIRST(A)$ a teda $FOLLOW(A) \subseteq FOLLOW(S)$
$A \rightarrow aS\underline{A}\epsilon$		$b$			$FIRST(\epsilon) = \{\epsilon\}$ a teda $FOLLOW(A) \subseteq FOLLOW(A)$
$B \rightarrow a\underline{C}a$			$a$		$FIRST(a) = \{a\}$
$C \rightarrow \underline{B}C$			$a, b$ $a, b$		$a, b \in FIRST(C)$ $\epsilon \in FIRST(C)$ a teda $FOLLOW(C) \subseteq FOLLOW(B)$
$C \rightarrow BC\underline{\epsilon}$				$a, b$	$FIRST(\epsilon) = \{\epsilon\}$ a teda $FOLLOW(C) \subseteq FOLLOW(C)$

# Množina FOLLOW

(podčiarknutý je neterminál, na základe ktorého je pravidlo aktuálne vyšetrované a zelenou farbou časť za príslušným neterminálom)

(červenou sú označené terminály, ktoré by mali byť pridané do príslušnej množiny FOLLOW na základe aktuálne vyšetrovaného pravidla, ale už sa v danej množine nachádzajú)

FOLLOW	S	A	B	C	Dôvod
Po 2. iterácii	$\epsilon, a, b, c$	$b$	$a, b, c, \epsilon$	$a, b$	
3. iterácia:					
$S \rightarrow \underline{A}bBB$		$b$			$FIRST(bBB) = \{b\}$
$S \rightarrow Ab\underline{B}B$			$a, b$		$FIRST(B) = \{a, b\}$
$S \rightarrow AbBB\underline{\epsilon}$			$\epsilon, a, b, c$		$FIRST(\epsilon) = \{\epsilon\}$ a teda $FOLLOW(S) \subseteq FOLLOW(B)$
$A \rightarrow \underline{C}C$				$a, b$ $b$	$a, b \in FIRST(C)$ $\epsilon \in FIRST(C)$ a teda $FOLLOW(A) \subseteq FOLLOW(C)$
$A \rightarrow CC\underline{\epsilon}$				$b$	$FIRST(\epsilon) = \{\epsilon\}$ a teda $FOLLOW(A) \subseteq FOLLOW(C)$
$A \rightarrow c\underline{S}A$	$a, b, c$ $b$				$a, b, c \in FIRST(A)$ $\epsilon \in FIRST(A)$ a teda $FOLLOW(A) \subseteq FOLLOW(S)$
$A \rightarrow aS\underline{A}\epsilon$		$b$			$FIRST(\epsilon) = \{\epsilon\}$ a teda $FOLLOW(A) \subseteq FOLLOW(A)$
$B \rightarrow a\underline{C}a$			$a$		$FIRST(a) = \{a\}$
$C \rightarrow \underline{B}C$			$a, b$ $a, b$		$a, b \in FIRST(C)$ $\epsilon \in FIRST(C)$ a teda $FOLLOW(C) \subseteq FOLLOW(B)$
$C \rightarrow BC\underline{\epsilon}$				$a, b$	$FIRST(\epsilon) = \{\epsilon\}$ a teda $FOLLOW(C) \subseteq FOLLOW(C)$

Vidíme, že v tretej iterácii sme nedostali žiaden nový symbol do žiadnej množiny FOLLOW, algoritmus teda končí.



Výsledné množiny *FOLLOW*:

1.  $FOLLOW(S) = \{\varepsilon, a, b, c\}$
2.  $FOLLOW(A) = \{b\}$
3.  $FOLLOW(B) = \{\varepsilon, a, b, c\}$
4.  $FOLLOW(C) = \{a, b\}$



## Zásobníkový automat - příklad č. 1

**Příklad:** ZA akceptující jazyk  $L = \{ww^R \mid w \in \{0, 1\}^*\}$ .

**Idea:**

- Slova z jazyka sa dajú rozdeliť na 2 časti -  $w$  a jeho zrkadlový obraz  $w^R$ . V prvej fáze bude automat čítať prvú časť -  $w$  - a každý prečítaný symbol uloží do zásobníka. Po prečítaní  $w$  bude teda zásobník obsahovať zvrchu práve zrkadlový obraz slova  $w$ .
- V druhej fáze bude potom ZA kontrolovať, či zvyšok slova na vstupe, t.j.  $w^R$ , je zrkadlový obraz  $w$ , t.j. či sa vždy navrchu zásobníka nachádza rovnaký symbol, ako je aktuálny symbol na vstupe. Ak áno, tak po spracovaní celého vstupu ostane v zásobníku len počiatočný zásobníkový symbol  $Z_0$  a slovo je akceptované.
- Prechod z prvej fázy do druhej fázy sa vykoná **nedeterministicky**.





## Zásobníkový automat - příklad (pokr.)

Formálně:

- $Q = \{q_0, q_1, q_f\}$
- $T = \{0, 1\}$
- $\Gamma = \{0, 1, Z_0\}$
- $\delta$  :
  - $\delta(q_0, 0, \varepsilon) = \{(q_0, 0)\}$ ,  $\delta(q_0, 1, \varepsilon) = \{(q_0, 1)\}$  - prvá fáza
  - $\delta(q_0, \varepsilon, \varepsilon) = \{(q_1, \varepsilon)\}$  - prechod do druhej fázy
  - $\delta(q_1, 0, 0) = \{(q_1, \varepsilon)\}$ ,  $\delta(q_1, 1, 1) = \{(q_1, \varepsilon)\}$  - druhá fáza
  - $\delta(q_1, \varepsilon, Z_0) = \{(q_f, \varepsilon)\}$  - prechod do akceptujúceho stavu
- $F = \{q_f\}$ .



## Zásobníkový automat - příklad (pokr.)

Vstupné slovo: 010010 (existuje akceptačná konfigurácia)

$$(q_0, 010010, Z_0) \vdash (q_0, 10010, 0Z_0) \vdash (q_0, 0010, 10Z_0) \vdash (q_0, 010, 010Z_0) \vdash (q_1, 010, 010Z_0) \vdash (q_1, 10, 10Z_0) \vdash (q_1, 0, 0Z_0) \vdash (q_1, \varepsilon, Z_0) \vdash (q_f, \varepsilon, \varepsilon)$$

Vstupné slovo: 000 (neexistuje akceptačná konfigurácia) - jedna zaujímavá:

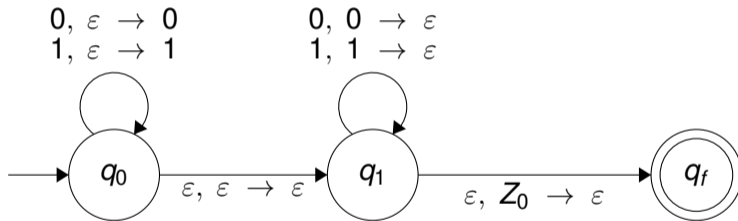
$$(q_0, 000, Z_0) \vdash (q_0, 00, 0Z_0) \vdash (q_1, 00, 0Z_0) \vdash (q_1, 0, Z_0) \vdash (q_f, 0, \varepsilon)$$

Horeuvedený výpočet **nie je** akceptačný, pretože na vstupe zostala neprečítaná časť vstupného slova.



## Zásobníkový automat - příklad (pokr.)

Grafická reprezentácia ZA by bola nasledovná:



## Nájdite zásobníkové automaty

Nájdite zásobníkové automaty k jazykom:

1.  $L_1 = \{w \in \{a, b\}^* \mid \#_a(w) = \#_b(w)\}$ , t.j. reťazce z písmen  $\{a, b\}$ , v ktorých je počet symbolov  $a, b$  rovnaký.
2.  $L_2 = \{w \in \{a, b\}^* \mid \#_a(w) > \#_b(w)\}$ , t.j. reťazce z písmen  $\{a, b\}$ , v ktorých je počet symbolov  $a$  väčší než počet symbolov  $b$ .
3.  $L_3 = \{w \in \{a, b\}^* \mid \#_a(w) < \#_b(w)\}$ , t.j. reťazce z písmen  $\{a, b\}$ , v ktorých je počet symbolov  $a$  menší než počet symbolov  $b$ .
4.  $L_4 = \{wcw^R \mid w \in \{a, b\}^*\}$ , t.j. palindrómy v strede ktorých je symbol  $c$ .
5.  $L_5 = \{xaby \mid x \in \{a, b\}^*, y \in \{a, b\}^*\}$ , t.j. reťazce z písmen  $\{a, b\}$  obsahujúce  $ab$  ako podreťazec
6.  $L_6 = \{a^n b^n c^m \mid n \geq 0, m \geq 0\}$



# $L_1$

Nájdite zásobníkový automat, ktorý akceptuje jazyk

$L_1 = \{w \in \{a, b\}^* \mid \#_a(w) = \#_b(w)\}$ , t.j. reťazce z písmen  $\{a, b\}$ , v ktorých je počet symbolov  $a, b$  rovnaký.

- Je zrejmé, že automat bude akceptovať len také reťazce, ktoré obsahujú rovnaký počet symbolov  $a$  a  $b$ .
- Keďže ZA číta vstup symbol po symbole, po prvom symbole automat vie, či prečítal  $a$  a teda niekde ďalej musí nasledovať  $b$ , alebo naopak.
- Preto zásobník vieme využiť ako pamäťové médium, pomocou ktorého si budeme pamätať, **koľko** "opačných" symbolov (k  $a$  je opačný  $b$  a naopak, k  $b$  je opačný  $a$ ) ešte musíme prečítať, aby boli počty vyrovnané.



Pri spracúvaní reťazca by bolo vhodné rozlišovať 3 situácie vzhľadom na **doteraz prečítanú časť slova**:

- Počet doteraz prečítaných  $a$  a  $b$  bol rovnaký - taká je situácia na **začiatku** a taktiež je to situácia, v ktorej ak skončíme  $a$  vstup bol celý prečítaný, vstup **budeme akceptovať**. Teda ju bude reprezentovať **počiatočný a zároveň akceptačný** stav  $q_0$ .
- Počet doteraz prečítaných  $a$  bol **väčší** než počet doteraz prečítaných  $b$ . To môže reprezentovať stav  $q_{a>b}$ . Zároveň to znamená, že aby sme slovo akceptovali, **musíme** prečítať ešte nejaké  $b$ -čka.
- Počet doteraz prečítaných  $a$  bol **menší** než počet doteraz prečítaných  $b$ . To môže reprezentovať stav  $q_{a<b}$ . Zároveň to znamená, že aby sme slovo akceptovali, **musíme** prečítať ešte nejaké  $a$ -čka.



- Ak sme v stave  $q_0$ , na vrchu zásobníka je  $Z_0$  a počet doteraz prečítaných symbolov je rovnaký. Ak teraz čítame na vstupe  $a$ , prejdeme do stavu  $q_{a>b}$ . Zároveň si do zásobníka vložíme symbol  $b$ , ktorý bude reprezentovať to, že musíme ešte prečítať jedno  $b$ -čko.
- Ak sme v stave  $q_0$ , na vrchu zásobníka je  $Z_0$  a počet doteraz prečítaných symbolov je rovnaký. Ak teraz čítame na vstupe  $b$ , prejdeme do stavu  $q_{a<b}$ . Zároveň si do zásobníka vložíme symbol  $a$ , ktorý bude reprezentovať to, že musíme ešte prečítať jedno  $a$ -čko.
- Ak sme v stave  $q_{a>b}$ , znamená to, že na vrchu zásobníka je symbol  $b$ , t.j. čakáme *minimálne* na jedno  $b$ -čko, aby sa počty vyrovnali. Ak v tomto stave príde na vstupe  $b$ , tak toto  $b$  zo zásobníka môžeme **odstrániť**. Naopak, ak v tomto stave zo vstupu čítame  $a$ , tak nám chýba ďalšie  $b$  na vstupe, čo znázorníme tak, že znovu do zásobníka vložíme ďalšie  $b$ . Čiže v tomto stave platí, že koľkokrát je symbol  $b$  v zásobníku, tak na toľko  $b$  na vstupe čakáme, aby sa počty vyrovnali.

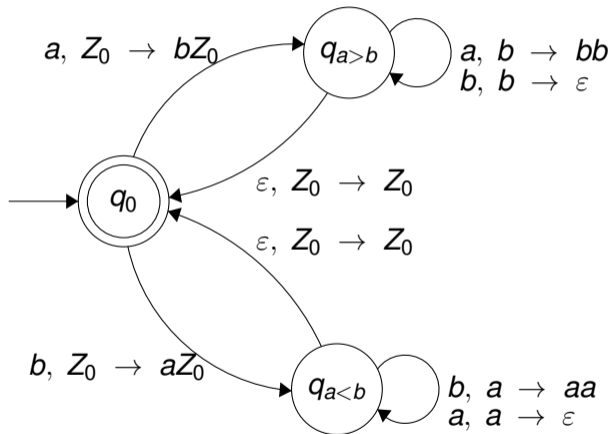


- Analogicky, ak sme v stave  $q_{a<b}$ , znamená to, že na vrchu zásobníka je symbol  $a$ , t.j. čakáme *minimálne* na jedno  $a$ -čko, aby sa počty vyrovnali. Ak v tomto stave príde na vstupe  $a$ , tak toto  $a$  zo zásobníka môžeme **odstrániť**. Naopak, ak v tomto stave zo vstupu čítame  $b$ , tak nám chýba ďalšie  $a$  na vstupe, čo znázorníme tak, že znovu do zásobníka vložíme ďalšie  $a$ . Čiže v tomto stave platí, že koľkokrát je symbol  $a$  v zásobníku, tak na toľko  $a$  na vstupe čakáme, aby sa počty vyrovnali.
- Ak sa v stavoch  $q_{a>b}$  alebo  $q_{a<b}$  stane, že sa na vrchu zásobníka ocitne symbol  $Z_0$ , znamená to, že sa nám doteraz prečítané vstupné symboly vyrovnali a môžeme sa vrátiť do stavu  $q_0$ .





Keď si uvedené pravidlá znázorníme obrázkom, dostaneme nasledovný obrázok zásobníkového automatu:



A formálne dostávame zásobníkový automat  $P = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$ , kde:

- $Q = \{q_0, q_{a>b}, q_{a<b}\}$
- $\Sigma = \{a, b\}$
- $\Gamma = \{Z_0, a, b\}$
- $q_0$  je počiatkový stav
- $Z_0$  je počiatkový zásobníkový symbol
- $F = \{q_0\}$
- a prechodová funkcia  $\delta$  je daná obrázkom, resp. aj predpisom (vid' ďalší slajd)

Prechodová funkcia  $\delta$ :

- $\delta(q_0, a, Z_0) = \{(q_{a>b}, bZ_0)\}$
- $\delta(q_0, a, Z_0) = \{(q_{a<b}, aZ_0)\}$
- $\delta(q_{a>b}, a, b) = \{(q_{a>b}, bb)\}$
- $\delta(q_{a>b}, b, b) = \{(q_{a>b}, \varepsilon)\}$
- $\delta(q_{a>b}, \varepsilon, Z_0) = \{(q_0, Z_0)\}$
- $\delta(q_{a<b}, a, a) = \{(q_{a>b}, \varepsilon)\}$
- $\delta(q_{a<b}, b, a) = \{(q_{a>b}, aa)\}$
- $\delta(q_{a<b}, \varepsilon, Z_0) = \{(q_0, Z_0)\}$



Ak je zostrojený zásobníkový automat korektný, musia platiť 2 veci:

- Každý reťazec z jazyka  $L_1$  (t.j. taký, ktorý obsahuje rovnaký počet  $a$  a  $b$ ) musí byť v automate akceptovaný, t.j.  $L_1 \subseteq L(P)$ .
- Každý reťazec, ktorý automat akceptuje, musí patriť do jazyka  $L_1$ , t.j.  $L(P) \subseteq L_1$ .

Neformálny dôkaz toho, že každý reťazec, ktorý obsahuje rovnaký počet  $a$  a  $b$ , má v automate akceptačný výpočet, t.j. že  $L_1 \subseteq L(P)$ :

- Každý reťazec  $w$ , ktorý obsahuje rovnaký počet  $a$  a  $b$  sa dá rozdeliť na menšie segmenty  $w_1, w_2, \dots, w_n$ , t.j.  $w = w_1 w_2 \dots w_n$ , v ktorých tiež platí, že počet  $a$  a  $b$  je rovnaký.
- Toto delenie je robené tak, aby segment začínal napr. symbolom  $a$  a končil symbolom  $b$ , ktorý je "párový" k prvému symbolu  $a$ , t.j. uzatvára segment, aby bol v ňom rovnaký počet symbolov  $a$  a  $b$ . Analogicky naopak, ak segment začína symbolom  $b$ , tak končí párovým  $a$ -čkom.
- Napríklad v reťazci  $w = abbbbaa$  by bolo delenie  $w = w_1 w_2$ , kde  $w_1 = ab$  a  $w_2 = bbaa$ .
- Automat je skonštruovaný tak, aby každý segment svojim prvým symbolom prešiel do stavu  $q_{a>b}$  alebo  $q_{a<b}$  a zo stavu sa vrátil do stavu  $q_0$  po prečítaní svojho posledného symbolu (t.j. po vyrovnaní počtu symbolov).



Napríklad pre  $w = abbbbaa$ , kde sme delenie určili ako  $w = w_1 w_2$ ,  $w_1 = ab$ ,  $w_2 = bbaa$  by bol výpočet pre prvú časť  $w_1 = ab$ :

$$(q_0, abbbbaa, Z_0) \vdash (q_{a>b}, bbbbaa, bZ_0) \vdash (q_{a>b}, bbaa, Z_0) \vdash (q_0, bbaa, Z_0)$$

a následne pre druhú časť  $w_2 = bbaa$ :

$$(q_0, bbaa, Z_0) \vdash (q_{a<b}, baa, aZ_0) \vdash (q_{a<b}, aa, aaZ_0) \vdash (q_{a<b}, a, aZ_0) \vdash \\ \vdash (q_{a<b}, \varepsilon, Z_0) \vdash (q_0, \varepsilon, Z_0)$$

Teda spolu:

$$(q_0, abbbbaa, Z_0) \vdash^* (q_0, \varepsilon, Z_0)$$

a teda reťazec  $abbbbaa$  by bol akceptovaný.



Neformálny dôkaz toho, že každý akceptovaný reťazec obsahuje rovnaký počet  $a$  a  $b$ , t.j.  $L(P) \subseteq L_1$ .

1. Akceptovaný reťazec je napríklad  $\varepsilon$ . Tento uvedenú vlastnosť spĺňa.
2. Ak akceptovaný reťazec obsahuje aspoň 1 symbol, tak po prečítaní prvého symbolu skončí v stave  $q_{a>b}$  ak bol prvý symbol  $a$ , alebo v stave  $q_{a<b}$ , ak bol prvý symbol  $b$ .
3. Z tohto stavu **neodíde dovedy**, kým sa nepodarí vyrovnať počty doteraz prečítaných  $a$  a  $b$ . Pretože až ich vyrovnaním sa na vrch zásobníka dostane symbol  $Z_0$ , vďaka ktorému sa dá vrátiť do stavu  $q_0$ .
4. Teda každý akceptovaný reťazec sa dá rozdeliť na segmenty  $w_1, w_2, \dots, w_n$ , pre ktoré platí, že ich čítaním sa automat dostane do stavu  $q_{a>b}$  alebo  $q_{a<b}$  a odchádza z neho až vtedy, keď je počet symbolov  $a$  a  $b$  rovnaký.
5. Teda segmenty obsahujú všetky rovnaký počet symbolov. A ich zreťazenie  $w = w_1 w_2 \dots w_n$  rovnako musí obsahovať **rovnaký počet symbolov**.



## $L_2$

Nájdite zásobníkový automat, ktorý akceptuje jazyk

$L_2 = \{w \in \{a, b\}^* \mid \#_a(w) > \#_b(w)\}$ , t.j. reťazce z písmen  $\{a, b\}$ , v ktorých je počet symbolov  $a$  väčší než počet symbolov  $b$ .

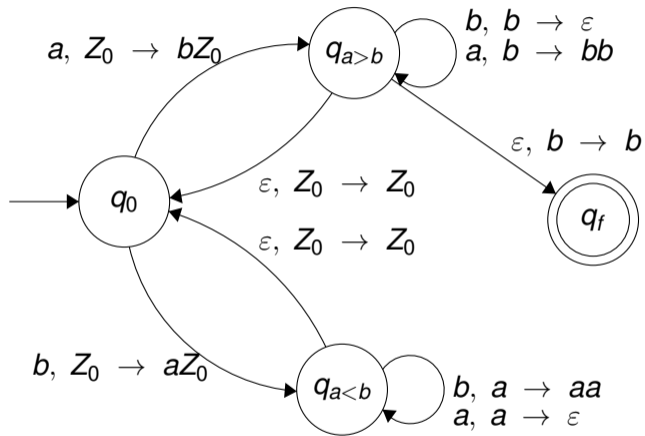
- Je zrejmé, že automat bude akceptovať len také reťazce, ktoré obsahujú väčší počet symbolov  $a$  než  $b$ .
- Keby sme vychádzali z automatu, ktorý sme skonštruovali v predchádzajúcej úlohe, tak pre reťazce, ktoré majú väčší počet  $a$  než  $b$  platí, že po ich dočítaní viem skončiť v stave  $q_{a>b}$  a zároveň v zásobníku **musí byť** aspoň jedno  $b$ -čko (lebo signalizuje, že ešte potrebujem  $b$  prečítať, aby bol počet rovnaký).
- Preto malou úpravou automatu vieme dostať zásobníkový automat, ktorý bude akceptovať jazyk  $L_2$ .





- Vieme teda, že ak je na vstupe slovo, ktoré má viac  $a$  než  $b$ , tak po jeho prečítaní výpočet skončí v stave  $q_{a>b}$  a v zásobníku bude aspoň jeden symbol  $b$ .
- Niekomu by mohlo napadnúť, že riešením je jednoducho **urobiť** z  $q_{a>b}$  **akceptačný stav**.
- Problém s týmto riešením je, že v stave  $q_{a>b}$  vieme skončiť aj pre slová, v ktorých je počet  $a$  a  $b$  rovnaký (pretože v pôvodnom automate sme potom prechodom, ktorý nečítal vstup, len potreboval, aby bol na vrchu zásobníka  $Z_0$ , prešli do akceptačného stavu  $q_0$ ).
- Preto teraz potrebujeme pri akceptácii testovať, či je na vrchu zásobníka symbol  $b$ .
- Najjednoduchšie je pridať do automatu nový stav, ktorý bude jediný **akceptačný stav**.
- Do tohto stavu prejdeme zo stavu  $q_{a>b}$  bez čítania vstupného symbolu za podmienky, že na vrchu zásobníka je  $b$ .





A formálne dostávame zásobníkový automat  $P = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$ , kde:

- $Q = \{q_0, q_{a>b}, q_{a<b}, q_f\}$
- $\Sigma = \{a, b\}$
- $\Gamma = \{Z_0, a, b\}$
- $q_0$  je počiatkový stav
- $Z_0$  je počiatkový zásobníkový symbol
- $F = \{q_f\}$
- a prechodová funkcia  $\delta$  je daná obrázkom, resp. aj predpisom (vid' ďalší slajd)



Prechodová funkcia  $\delta$ :

- $\delta(q_0, a, Z_0) = \{(q_{a>b}, bZ_0)\}$
- $\delta(q_0, a, Z_0) = \{(q_{a<b}, aZ_0)\}$
- $\delta(q_{a>b}, a, b) = \{(q_{a>b}, bb)\}$
- $\delta(q_{a>b}, b, b) = \{(q_{a>b}, \varepsilon)\}$
- $\delta(q_{a>b}, \varepsilon, Z_0) = \{(q_0, Z_0)\}$
- $\delta(q_{a<b}, a, a) = \{(q_{a>b}, \varepsilon)\}$
- $\delta(q_{a<b}, b, a) = \{(q_{a>b}, aa)\}$
- $\delta(q_{a<b}, \varepsilon, Z_0) = \{(q_0, Z_0)\}$
- $\delta(q_{a>b}, \varepsilon, b) = \{(q_f, b)\}$



Ak je zostrojený zásobníkový automat korektný, musia platiť 2 veci:

- Každý reťazec z jazyka  $L_2$  (t.j. taký, ktorý obsahuje rovnaký počet  $a$  a  $b$  ) musí byť v automate akceptovaný, t.j.  $L_2 \subseteq L(P)$ .
- Každý reťazec, ktorý automat akceptuje, musí patriť do jazyka  $L_2$ , t.j.  $L(P) \subseteq L_2$ .



Neformálny dôkaz toho, že každý reťazec, ktorý obsahuje väčší počet  $a$  než  $b$ , má v automate akceptačný výpočet, t.j. že  $L_2 \subseteq L(P)$ :

- Každý reťazec  $w$ , ktorý obsahuje väčší počet  $a$  a  $b$  sa dá rozdeliť na menšie segmenty  $w_1, w_2, \dots, w_n$ , t.j.  $w = w_1 w_2 \dots w_n$ , v ktorých tiež platí, že počet  $a$  a  $b$  je rovnaký v segmentoch  $w_1, \dots, w_{n-1}$  a v poslednom segmente je aspoň o 1  $a$  viac než  $b$ .
- Toto delenie je robené tak, aby segmenty  $w_1$  až  $w_{n-1}$  začínali napr. symbolom  $a$  a končili symbolom  $b$ , ktorý je "párový" k prvému symbolu  $a$ , t.j. uzatvára segment, aby bol v ňom rovnaký počet symbolov  $a$  a  $b$ . Analogicky naopak, ak segment začína symbolom  $b$ , tak končí párovým  $a$ -čkom.
- Napríklad v reťazci  $w = abaabab$  by bolo delenie  $w = w_1 w_2$ , kde  $w_1 = ab$  a  $w_2 = aabab$ .
- Automat je skonštruovaný tak, aby každý segment  $w_1$  až  $w_{n-1}$  svojim prvým symbolom prešiel do stavu  $q_{a>b}$  alebo  $q_{a<b}$  a zo stavu sa vrátil do stavu  $q_0$  po prečítaní svojho posledného symbolu (t.j. po vyrovnaní počtu symbolov).
- A posledný segment sa dostane do  $q_{a>b}$  v ktorom sa dočíta do konca, pričom v zásobníku bude po jeho dočítaní aspoň jedno  $b$  na vrchu.



Napríklad pre  $w = abaabab$ , kde sme delenie určili ako  $w = w_1 w_2$ ,  $w_1 = ab$ ,  $w_2 = aabab$  by bol výpočet pre prvú časť  $w_1 = ab$ :

$$(q_0, abaabab, Z_0) \vdash (q_{a>b}, baabab, bZ_0) \vdash (q_{a>b}, aabab, Z_0) \vdash (q_0, aabab, Z_0)$$

a následne pre druhú časť  $w_2 = aabab$ :

$$(q_0, aabab, Z_0) \vdash (q_{a>b}, abab, bZ_0) \vdash (q_{a>b}, bab, bbZ_0) \vdash (q_{a>b}, ab, bZ_0) \vdash \\ \vdash (q_{a>b}, b, bbZ_0) \vdash (q_{a>b}, \varepsilon, bZ_0) \vdash (q_f, \varepsilon, bZ_0)$$

Teda spolu:

$$(q_0, abaabab, Z_0) \vdash^* (q_f, \varepsilon, bZ_0)$$

a teda reťazec  $abaabab$  by bol akceptovaný.



Neformálny dôkaz toho, že každý akceptovaný reťazec obsahuje väčší počet  $a$  než  $b$ , t.j.  $L(P) \subseteq L_2$ .

1. Akceptovaný reťazec má tú vlastnosť, že po jeho dočítaní musí automat skončiť v stave  $q_{a>b}$  so symbolom  $b$  na vrchu zásobníka, aby sa v ďalšom kroku vedel presunúť do  $q_f$ .
2. Aby sa dostal do stavu  $q_{a>b}$ , musí v stave  $q_0$  prečítať aspoň jedno  $a$  a následne, aby mu v stave  $q_{a>b}$  zostal aspoň 1 symbol  $b$  v zásobníku, tak nesmie prečítať toľko  $b$ , aby dorovnal prečítaný počet  $a$ .
3. Avšak do stavu  $q_0$  sa už predtým mohol dostať tým, že by prečítal segmenty vstupného slova  $w_i$ , v ktorých bol rovnaký počet symbolov  $a$  a  $b$ .
4. Teda každý akceptovaný reťazec sa dá rozdeliť na segmenty  $w_1, w_2, \dots, w_n$ , pre ktoré platí, že segmenty  $w_1, \dots, w_{n-1}$  obsahujú rovnaký počet  $a$  a  $b$  a ich čítaním sa vždy vie ZA vrátiť do  $q_0$  a následne segment  $w_n$  začína symbolom  $a$ , ktorým sa presunie do stavu  $q_{a>b}$  a neobsahuje toľko symbolov  $b$ , aby sa počty dorovnali a v zásobníku zostalo  $Z_0$  na vrchu.
5. Teda v zret'azení  $w_1 w_2 \dots w_{n-1} w_n$  bude aspoň o 1 **a viac** než počet  $b$ .



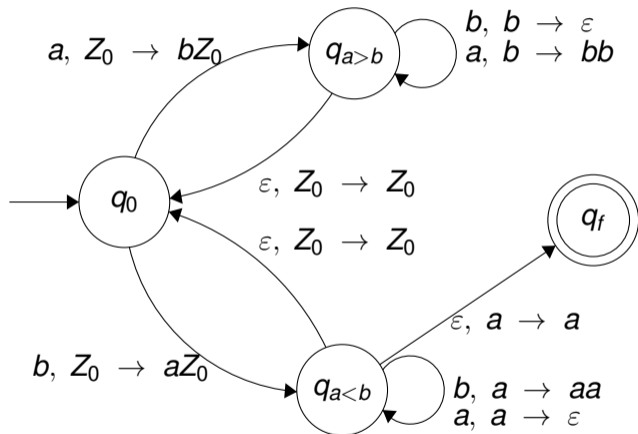
## $L_3$

Nájdite zásobníkový automat, ktorý akceptuje jazyk

$L_3 = \{w \in \{a, b\}^* \mid \#_a(w) < \#_b(w)\}$ , t.j. reťazce z písmen  $\{a, b\}$ , v ktorých je počet symbolov  $a$  menší než počet symbolov  $b$ .

- Tento zásobníkový automat bude **analógiou** automatu z predchádzajúcej úlohy, akurát že do akceptačného stavu nepôjde zo stavu  $q_{a>b}$ , ale zo stavu  $q_{a<b}$  za situácie, že na vrchu zásobníka bude symbol  $a$  signalizujúci, že bol prečítaný väčší počet symbolov  $b$  než  $a$ .

Tu si vystačíme len s obrázkom:



$L_4$

Nájdite zásobníkový automat, ktorý akceptuje jazyk  $L_4 = \{w c w^R \mid w \in \{a, b\}^*\}$ , t.j. palindrómy v strede ktorých je symbol  $c$ .

Automat by mal akceptovať reťazce:

- $c$
- $aca, bcb$
- $aacaa, abcba, bacab, bbcbb$
- ...



Vidíme z popisu, že reťazce pozostávajú z 3 častí:

1. Reťazca  $w$ , ktorý tvoria všetky možné reťazce zo symbolov  $\{a, b\}$
  2. V strede je symbol  $c$
  3. reťazca  $w^R$ , ktorý predstavuje zrkadlový obraz (t.j. napísaný odzadu) reťazca  $w$ .
- Zásobníkový automat teda musí kontrolovať, či sa **za symbolom**  $c$  nachádza zrkadlový obraz reťazca, ktorý je **pred symbolom**  $c$ .
  - Keďže pred symbolom  $c$  môže byť reťazec ľubovoľnej dĺžky, je potrebné si ho niekde zapamätať - ideálne v zásobníku.
  - Navyše má zásobník výhodnú vlastnosť, že posledný vložený symbol je zároveň prvý vyberaný symbol, t.j. ak doň postupne ukladáme symboly reťazca  $w$ , tak ak zásobník čítame odvrchu, tak vlastne čítame priamo reťazec  $w^R$ .



To znamená, že automat bude fungovať nasledovným spôsobom:

1. Najprv číta časť vstupu predstavujúcu reťazec  $w$  pred symbolom  $c$  - každý prečítaný symbol vloží do zásobníka. Toto bude predstavovať stav ZA  $q_0$ .
2. Následne, keď narazí na symbol  $c$ , tak sa prepne do režimu, v ktorom bude kontrolovať, či je na vstupe zrkadlový obraz reťazca  $w$  tým, že bude porovnávať vstup s obsahom zásobníka - keďže na vstupe by **malo byť**  $w^R$  a v zásobníku sa momentálne nachádza odvrchu **práve**  $w^R$ . Túto kontrolu, či sa obsah zásobníka zhoduje so vstupom, bude predstavovať stav  $q_1$
3. Ak sa podarí *napárovať* každý vstupný symbol so zásobníkom, tak sa v momente, keď sa celý vstup prečíta v zásobníku na vrchu nachádza symbol  $Z_0$ , čo signalizuje akceptáciu slova. Tú bude signalizovať akceptačný stav  $q_2$ .



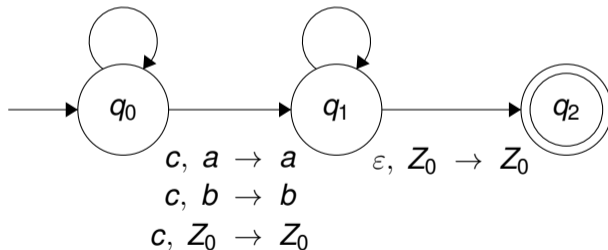
## Pre zhrnutie

- ZA bude mať stav  $q_0$  v ktorom bude zo vstupu čítať symboly  $a, b$  z časti  $w$  vstupu a ukladať ich do zásobníka.
- V momente, keď prečíta zo vstupu  $c$ , bude vedieť, že ďalej by mala nasledovať časť  $w^R$  vstupu a prepne sa do stavu  $q_1$ . Pri tomto prepínaní sa do zásobníka nič nové nevkladá.
- V stave  $q_1$  číta vstupné symboly a porovnáva ich s obsahom zásobníka. Ak sa zhodujú, odstráni symbol zo zásobníka a pokračuje s ďalším vstupným symbolom.
- Keď v stave  $q_1$  dočíta celý vstup a skončí so zásobníkom na vrchu ktorého je len symbol  $Z_0$ , tak to signalizuje, že vstup je v požadovanom tvare. V takom prípade sa prepne do akceptačného stavu  $q_2$ .



Grafická reprezentácia takého ZA by bola:

$b, b \rightarrow bb$   
 $a, b \rightarrow ab$   
 $b, a \rightarrow ba$   
 $a, a \rightarrow aa$   
 $b, Z_0 \rightarrow bZ_0$        $a, a \rightarrow \varepsilon$   
 $a, Z_0 \rightarrow aZ_0$        $b, b \rightarrow \varepsilon$



A formálne dostávame zásobníkový automat  $P = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$ , kde:

- $Q = \{q_0, q_1, q_2\}$
- $\Sigma = \{a, b, c\}$
- $\Gamma = \{Z_0, a, b\}$
- $q_0$  je počiatkový stav
- $Z_0$  je počiatkový zásobníkový symbol
- $F = \{q_2\}$
- a prechodová funkcia  $\delta$  je daná obrázkom, resp. aj predpisom (vid' ďalší slajd)





## Prechodová funkcia $\delta$ :

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



Ak je zostrojený zásobníkový automat korektný, musia platiť 2 veci:

- Každý reťazec z jazyka  $L_4$  (t.j. palindrómy v strede ktorých je  $c$ ) musí byť v automate akceptovaný, t.j.  $L_4 \subseteq L(P)$ .
- Každý reťazec, ktorý automat akceptuje, musí patriť do jazyka  $L_4$ , t.j.  $L(P) \subseteq L_4$ .



Neformálny dôkaz toho, že každý palindróm, ktorý obsahuje v strede  $c$ , má v automate akceptačný výpočet, t.j. že  $L_4 \subseteq L(P)$ :

- Každý palindróm, ktorý má v strede  $c$ , je v tvare  $wcw^R$ .
- V automate je možnosť pre časť reťazca  $w$  túto časť v stave  $q_0$  čítať a vkladať do zásobníka. Tým sa v zásobníku postupne od vrchu po spodok vytvorí reťazec  $w^R$ .
- Následne sa prečítaním symbolu  $c$  automat prepne do stavu  $q_1$ .
- V tomto stave sa čítaním  $w^R$  zo vstupu a porovnávaním  $w^R$  v zásobníku postupne číta vstup a vyprázdňuje zásobník.
- Následne po prečítaní posledného symbolu a vyprázdnení posledného  $\{a, b\}$  zo zásobníka na vrchu zásobníka ocitne len  $Z_0$ , čím je umožnený prechod do akceptačného stavu  $q_2$ .
- T.j. ak je na vstupe  $wcw^R$ , teda slovo z jazyka  $L_4$ , tak v automate existuje akceptačný výpočet:

$$(q_0, (wcw^R), Z_0) \vdash^* (q_0, cw^R, w^R Z_0) \vdash (q_1, w^R, w^R Z_0) \vdash^* (q_1, \varepsilon, Z_0) \vdash (q_2, \varepsilon, Z_0)$$



Neformálny dôkaz toho, že každý akceptovaný reťazec je zároveň palindrómom  $wcw^R$ .

1. Predpokladajme, že automat akceptuje nejaký reťazec  $x$ . To znamená, že  $(q_0, x, Z_0) \vdash^* (q_2, \varepsilon, Z_0)$
2. To zároveň znamená, že aj  $(q_0, x, Z_0) \vdash^* (q_1, \varepsilon, Z_0)$ , teda že pri spracovaní celého reťazca  $x$  sa vie automat dostať do  $q_1$  a v zásobníku je len symbol  $Z_0$ .
3. Zamyslime sa, čo musel byť posledný čítaný symbol: ak ním bolo  $c$ , tak to znamená, že bol vykonaný prechod do stavu  $q_1$ . Keďže v zásobníku je po tomto prechode  $Z_0$ , muselo tam byť aj **predtým**.
4. A teda pred symbolom  $c$  **nebolo nič iné čítané zo vstupu** - v opačnom prípade by to bolo totiž v zásobníku.
5. Teda vstup musel byť len reťazec  $c$  - a ten je palindrómom.
6. Čo ak v kroku 3 nebol posledný čítaný symbol  $c$  - muselo to byť  $a$  alebo  $b$ .



(pokračovanie)

7. V takom prípade však musel byť prechod z predchádzajúceho stavu  $q_1$  a teda prechod, ktorý je možný, len ak je v zásobníku rovnaký symbol ako na vstupe - t.j. prechod, ktorý odstraňuje zo zásobníka vrchný symbol.
8. Túto úvahu vieme opakovať, kým nedospejeme do situácie, že sme do stavu  $q_1$  prešli čítaním symbolu  $c$  zo stavu  $q_0$ . Čiže prechody zo stavu  $q_1$  do stavu  $q_1$  boli umožnené len ak bol na vstupe taký reťazec, ktorý bol zároveň v zásobníku - nech je tento reťazec  $w$ .
9. Pred týmto reťazcom sa na vstupe musel teda nachádzať symbol  $c$ , ktorý spôsobil prechod do stavu  $q_1$  zo stavu  $q_0$ .
10. Aby bol reťazec akceptovaný, tak počas jeho čítania v stave  $q_0$  sa každý symbol vkladal do zásobníka, t.j. ak je v zásobníku pri prechode do  $q_1$  reťazec  $w$ , tak sa tam musel dostať čítaním v stave  $q_0$  - z dôvodu toho, ako funguje zásobník sa musel čítať "odzadu", t.j. ako  $w^R$ .
11. To znamená, že ak je nejaký reťazec akceptovaný, tak musí byť tvaru  $w^Rcw = cw^R$ , t.j. **práve** palindróm so stredným symbolom  $c$ .



## $L_5$

Nájdite zásobníkový automat, ktorý akceptuje jazyk

$L_5 = \{xaby \mid x \in \{a, b\}^*, y \in \{a, b\}^*\}$ , t.j. reťazce z písmen  $\{a, b\}$  obsahujúce  $ab$  ako podreťazec.

Automat by mal akceptovať reťazce pozostávajúce z 3 častí:

- Ľubovoľného prefixu zo symbolov  $a, b$
- Podreťazca  $ab$
- Ľubovoľného sufixu zo symbolov  $a, b$

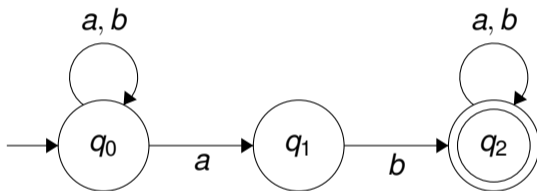


Trikom pri konštrukcii tohto zásobníkového automatu je uvedomenie si nasledovnej skutočnosti:

**Daný jazyk je regulárny! To znamená, že stačí zostrojiť DKA / NKA /  $\epsilon$ -NKA, ktorý daný jazyk akceptuje a pridať k nemu prácu so zásobníkom, pri ktorej sa zásobník úplne ignoruje**



NKA akceptujúci uvedený jazyk je napríklad:



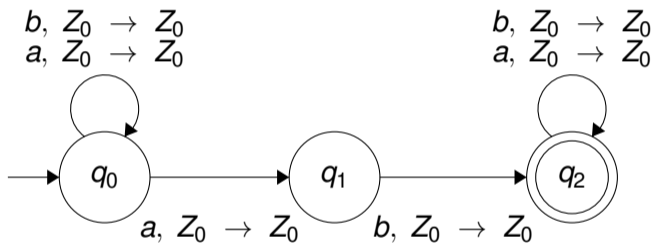


Keďže my chceme zásobníkový automat, ku každému prechodu pridáme prácu so zásobníkom, ktorá vezme z vrchu zásobníka symbol  $Z_0$  a vzápätí ho vráti naspäť.

Tým v podstate modelujeme situáciu, že sa zásobník ignoruje.



Grafická reprezentácia výsledného ZA by bola:



A formálne dostávame zásobníkový automat  $P = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$ , kde:

- $Q = \{q_0, q_1, q_2\}$
- $\Sigma = \{a, b\}$
- $\Gamma = \{Z_0\}$
- $q_0$  je počiatkový stav
- $Z_0$  je počiatkový zásobníkový symbol
- $F = \{q_2\}$
- a prechodová funkcia  $\delta$  je daná obrázkom, resp. aj predpisom (vid' ďalší slajd)



Prechodová funkcia  $\delta$ :

- $\delta(q_0, a, Z_0) = \{(q_0, Z_0), (q_1, Z_0)\}$
- $\delta(q_0, b, Z_0) = \{(q_0, Z_0)\}$
- $\delta(q_1, b, Z_0) = \{(q_2, Z_0)\}$
- $\delta(q_2, a, Z_0) = \{(q_2, Z_0)\}$
- $\delta(q_2, b, Z_0) = \{(q_2, Z_0)\}$



Ak je zostrojený zásobníkový automat korektný, musia platiť 2 veci:

- Každý reťazec z jazyka  $L_5$  (t.j. reťazce v tvare  $xaby$ , kde  $x, y$  sú reťazce symbolov  $\{a, b\}$ ) musí byť v automate akceptovaný, t.j.  $L_5 \subseteq L(P)$ .
- Každý reťazec, ktorý automat akceptuje, musí patriť do jazyka  $L_5$ , t.j.  $L(P) \subseteq L_5$ .

Neformálny dôkaz toho, že každý reťazec v tvare  $xaby$  má v automate akceptačný výpočet, t.j. že  $L_5 \subseteq L(P)$ :

- Pre prvú časť vstupného reťazca označenú  $x$  platí, že pri jej čítaní ostáva automat v stave  $q_0$ .
- Následne sa podreťazcom  $ab$  postupne prepne zo stavu  $q_0$  do stavu  $q_1$  prečítaním  $a$ , a prečítaním  $b$  sa prepne z  $q_1$  do  $q_2$ .
- Pre zvyšnú časť vstupu označenú  $y$  platí, že sa dá prečítať a stále zotrvať v akceptačnom stave  $q_2$ .
- Preto platí, že sa dá taký reťazec celý prečítať a skončiť v akceptačnom stave  $q_2$ , t.j. existuje výpočet:

$$(q_0, xaby, Z_0) \vdash^* (q_0, aby, Z_0) \vdash (q_1, by, Z_0) \vdash (q_2, y, Z_0) \vdash^* (q_2, \varepsilon, Z_0)$$



Neformálny dôkaz toho, že každý akceptovaný reťazec je zároveň reťazec v tvare  $xaby$ , kde  $x, y$  sú ľubovoľné reťazce zo symbolov  $\{a, b\}$ .

1. Predpokladajme, že automat akceptuje nejaký reťazec  $w$ . To znamená, že  $(q_0, w, Z_0) \vdash^* (q_2, \varepsilon, Z_0)$
2. Z pohľadu na automat vidíme, že každý akceptovaný reťazec musí mať tú vlastnosť, že počas jeho spracovania sa prejde zo stavu  $q_0$  cez  $q_1$  do stavu  $q_2$ .
3. To zároveň znamená, že reťazec **musí** niekde obsahovať postupnosť  $ab$ .
4. Navyše vidíme, že ak sa už dostane do stavu  $q_2$ , tak potom čokoľvek, čo obsahuje, je možné bez problémov prečítať, t.j. za reťazcom  $ab$  môže nasledovať čokoľvek.
5. Taktiež vidíme, že ak pred  $ab$  obsahoval čokoľvek, tak sme to v podstate mohli prečítať a stále zotrvať v stave  $q_0$ .
6. Preto každý akceptovaný reťazec spĺňa tvar  $xaby$ , kde  $x, y$  sú ľubovoľné (aj prázdne) reťazce zo symbolov  $\{a, b\}$ .



## $L_6$

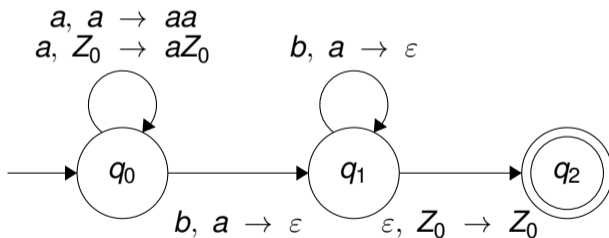
Nájdite zásobníkový automat, ktorý akceptuje jazyk  $L_6 = \{a^n b^n c^m \mid n \geq 1, m \geq 0\}$ . Automat by mal akceptovať reťazce pozostávajúce z 2 hlavných častí:

- Reťazca zo symbolov  $a, b$ , ktorých je rovnako veľa a navyše sú v tvare, že sú najprv symboly  $a$  a potom symboly  $b$ . Navyše je tam aspoň 1  $a$  a 1  $b$ .
- Reťazca zo symbolov  $c$ , ktorých je nejaký počet.
- Navyše platí, že počet symbolov  $c$  nijako nezávisí na počte  $a$ , resp.  $b$ .



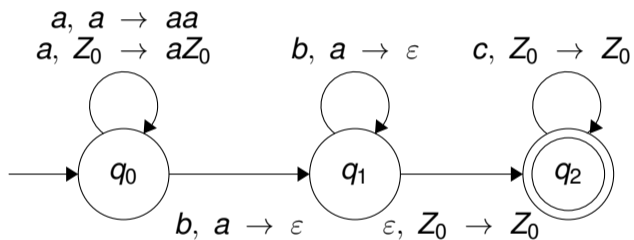


Keďže reťazce, ktoré má automat akceptovať, by mali začínať časťou, ktorá je v tvare  $a^n b^n$ , môžeme si pri konštrukcii ZA pomôcť zásobníkovým automatom pre tento jazyk (riešili sme ho v prednáške na slajde č. 35). V tomto automate platí, že ak má na vstupe reťazec  $a^n b^n$ , tak existuje výpočet, ktorý skončí v stave  $q_2$  a na vrchu zásobníka je symbol  $Z_0$ :



- V uvedenom automate teda platí, že do akceptačného stavu  $q_2$  sa vieme dostať a prečítať pri tom reťazec  $a^n b^n$ .
- Keďže my chceme akceptovať reťazce tvaru  $a^n b^n c^m$ , tak vlastne potrebujeme zo stavu  $q_2$  ešte prečítať zvyšok reťazca, t.j.  $c^m$ .
- Keďže týchto  $c$ -čok je ľubovoľný počet a navyše je tento počet nezávislý od iných hodnôt, v podstate ani zásobník nepotrebujeme, t.j. môžeme ignorovať jeho obsah - keďže v stave  $q_2$  je na jeho vrchu  $Z_0$ , tak ho tam necháme a len v slučke dočítame symboly  $c$  do konca.
- Preto stačí len pridať do stavu  $q_2$  slučku, v ktorej budeme čítať symboly  $c$ , pričom z vrchu zásobníka vezmeme  $Z_0$  a zase ho naspäť vložíme.

Grafická reprezentácia výsledného ZA by bola:



A formálne dostávame zásobníkový automat  $P = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$ , kde:

- $Q = \{q_0, q_1, q_2\}$
- $\Sigma = \{a, b, c\}$
- $\Gamma = \{Z_0, a\}$
- $q_0$  je počiatkový stav
- $Z_0$  je počiatkový zásobníkový symbol
- $F = \{q_2\}$
- a prechodová funkcia  $\delta$  je daná obrázkom, resp. aj predpisom (vid' ďalší slajd)



Prechodová funkcia  $\delta$ :

- $\delta(q_0, a, Z_0) = \{(q_0, aZ_0)\}$
- $\delta(q_0, a, a) = \{(q_0, aa)\}$
- $\delta(q_0, b, a) = \{(q_1, \varepsilon)\}$
- $\delta(q_1, b, a) = \{(q_1, \varepsilon)\}$
- $\delta(q_1, \varepsilon, Z_0) = \{(q_2, Z_0)\}$
- $\delta(q_2, c, Z_0) = \{(q_2, Z_0)\}$



Ak je zostrojený zásobníkový automat korektný, musia platiť 2 veci:

- Každý reťazec z jazyka  $L_6$  (t.j. reťazce v tvare  $a^n b^n c^m$ ) musí byť v automate akceptovaný, t.j.  $L_6 \subseteq L(P)$ .
- Každý reťazec, ktorý automat akceptuje, musí patriť do jazyka  $L_6$ , t.j.  $L(P) \subseteq L_6$ .

Neformálny dôkaz toho, že každý reťazec v tvare  $a^n b^n c^m$  má v automate akceptačný výpočet, t.j. že  $L_6 \subseteq L(P)$ :

- Pre prvú časť vstupného reťazca, t.j.  $a^n$  existuje výpočet, ktorý každé prečítané  $a$  v stave  $q_0$  vloží do zásobníka.
- Následne sa čítaním prvého  $b$  na vstupe a odstránením  $a$  zo zásobníka automat prepne do stavu  $q_1$ , v ktorom sa každé ďalšie  $b$  porovná so symbolom v zásobníku a ak je na vstupe  $b$  a v zásobníku  $a$ , tak sa  $a$  odstráni zo zásobníka a na vstupe sa prejde na ďalšie  $b$ . Ak je ich rovnaký počet, t.j. na vstupe je  $b^n$ , tak na vrchu zásobníka sa ocitne  $Z_0$ .
- Následne je možné prejsť do stavu  $q_2$  vďaka symbolu  $Z_0$  na vrchu zásobníka. V tomto stave je možné prečítať ľubovoľnú postupnosť  $c$ -čok na vstupe, t.j.  $c^m$ .
- Dokopy teda ak je na vstupe  $a^n b^n c^m$ , tak existuje spôsob, ako ho prečítať a skončiť v akceptačnom stave.

$$(q_0, a^n b^n c^m, Z_0) \vdash^* (q_0, b^n c^m, a^n Z_0) \vdash (q_1, b^{n-1} c^m, a^{n-1} Z_0) \vdash^* (q_1, c^m, Z_0) \vdash \\ \vdash (q_2, c^m, Z_0) \vdash^* (q_2, \varepsilon, Z_0)$$



Neformálny dôkaz toho, že každý akceptovaný reťazec je zároveň reťazec v tvare  $a^n b^n c^m$ :

1. Predpokladajme, že automat akceptuje nejaký reťazec  $w$ . To znamená, že  $(q_0, w, Z_0) \vdash^* (q_2, \varepsilon, Z_0)$
2. Z pohľadu na automat vidíme, že do akceptačného stavu  $q_2$  prejdeme zo stavu  $q_1$  len ak je na vrchu zásobníka  $Z_0$ .
3. Ak by v zásobníku boli v stave  $q_1$  iné symboly, než  $Z_0$ , tak tam musí byť symbol  $a$ . Ten je možné odstrániť len čítaním  $b$  zo vstupu v stave  $q_1$ . Ale  $a$  sa do zásobníka dostane len tak, že v stave  $q_0$  čítame zo vstupu  $a$ . Aby sme prečítaním  $b$ -čok v  $q_1$  odstránili všetky  $a$  zo zásobníka, tak sme museli prečítať práve  $a^n b^n$ .
4. Následne je po prečítaní  $a^n b^n$  možné prejsť do  $q_2$ . Každý reťazec, ktorý sa celý prečíta môže teoreticky na konci obsahovať nejaké  $c$  symboly, t.j.  $c^m$ .
5. Preto každý akceptovaný reťazec zároveň spĺňa tvar  $a^n b^n c^m$ .

