

Antwoorden Languages & Machines (Oefening 2)

1. Originele grammatica G (uit opgave 4.4 van het boek):

$$G = \begin{cases} S \rightarrow AB | BCS \\ A \rightarrow aA | C \\ B \rightarrow bB | \lambda \\ C \rightarrow cC | \lambda \end{cases}$$

G_1 : (nieuw startsymbool, S_0 , niet recursief maken)

$$G_1 = \begin{cases} S_0 \rightarrow AB | BCS \\ S \rightarrow AB | BCS \\ A \rightarrow aA | C \\ B \rightarrow bB | \lambda \\ C \rightarrow cC | \lambda \end{cases}$$

G_2 : (non-contracting maken door null-regels te elimineren).

De null regels zijn: $\{C, B, A, S, S_0\}$

$$G_2 = \begin{cases} S_0 \rightarrow A | B | C | S | AB | BC | BS | CS | BCS | \lambda \\ S \rightarrow A | B | C | S | AB | BC | BS | CS | BCS \\ A \rightarrow aA | a | C \\ B \rightarrow bB | b \\ C \rightarrow cC | c \end{cases}$$

G_3 : (chain rules elimineren).

De niet-triviale kettingen zien er als volgt uit:

$$\begin{aligned} \text{chain}(S) &= \{A, B, C, S\} \\ \text{chain}(S_0) &= \{A, B, C, S, S_0\} \end{aligned}$$

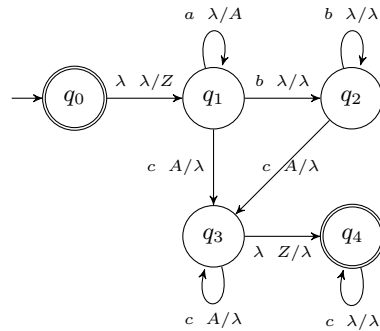
$$G_3 = \begin{cases} S_0 \rightarrow aA | a | cC | c | bB | b | AB | BC | BS | CS | BCS | \lambda \\ S \rightarrow aA | a | cC | c | bB | b | AB | BC | BS | CS | BCS \\ A \rightarrow aA | a | cC | c \\ B \rightarrow bB | b \\ C \rightarrow cC | c \end{cases}$$

G_4 : (rechterkanten fatsoeneren)

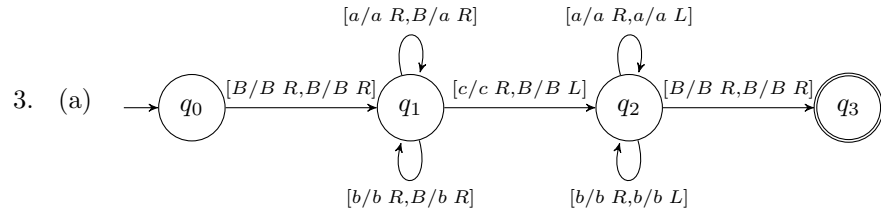
$$G_4 = \begin{cases} S_0 \rightarrow XA | a | ZC | c | YB | b | AB | BC | BS | CS | US | \lambda \\ S \rightarrow XA | a | ZC | c | YB | b | AB | BC | BS | CS | US \\ A \rightarrow XA | a | ZC | c \\ B \rightarrow YB | b \\ C \rightarrow ZC | c \\ X \rightarrow a \\ Y \rightarrow b \\ Z \rightarrow c \\ U \rightarrow BC \end{cases}$$

2. $L = \{a^i b^* c^j \mid j \geq i \geq 0\}$

(a) Een deterministische PDA voor L :



Er wordt eerst een Z op de stack gezet, om later te testen dat de stack (bijna) leeg is. In q_1 worden a 's gelezen, en geteld op de stack. Dan kan er een willekeurig aantal b 's volgen, die verder niet hoeven te worden geteld. Na de eerste c vanaf q_1 of q_2 springen we naar q_3 . Hier worden net zoveel c 's gelezen als er a 's waren. Als de stack leeg is springen we naar q_4 . Daar mogen nog meer c 's volgen. De automaat is deterministisch, omdat er vanuit geen enkele toestand overlappende transitie mogelijk zijn.



(b) Er wordt eerst een maximaal woord $w \in \{a, b\}^*$ van tape 1 naar tape 2 gekopieerd. Na een c wordt gecheckt of het resterende woord w^R op tape 1 matcht met het omgekeerde woord op tape 2. De berekening is:

$[q_0; *BaabcbaaB; *BBBBB]$
 $[q_1; B * aabcbaaB; B * BBBB]$
 $[q_1; Ba * abcbaaB; Ba * BBB]$
 $[q_1; Baa * bcbaaB; Baa * BB]$
 $[q_1; Baab * cbaaB; Baab * B]$
 $[q_2; Baabc * baaB; Baa * bB]$
 $[q_2; Baabcb * aaB; Ba * abB]$
 $[q_2; Baabcba * aB; B * aabB]$
 $[q_2; Baabcbaa * B; *BaabB]$
 $[q_3; BaabcbaaB*; B * aabB]$

De TM termineert in de accepterende toestand q_3 , dus het $aabcbaa$ wordt geaccepteerd.

- (c) Deze TM termineert altijd (na 1 pass over het woord op tape 1), dus hij beslist deze taal.
- (d) Deze TM is deterministisch, want het symbool op tape 1 bepaalt uniek welke transitie wordt genomen.