CS5003
Quiz 1001

Foundations of C.S.

Spring, 2017

PRINT NAME: _____
SIGN: _____

1. (4 points) Give an equivalent grammar to

$$\begin{aligned} G: S &\rightarrow BS \mid SB \mid B \\ A &\rightarrow aA \mid b \\ B &\rightarrow aB \mid bA \mid C \\ C &\rightarrow cC \mid A \end{aligned}$$

with a non-recursive start and no chain rules.

♣ No variables nullable, NULL $= \emptyset$, but it does has a recursive start, we transform to:

$$\begin{aligned} S &\rightarrow BS' \mid S'B \mid B \\ S' &\rightarrow BS' \mid S'B \mid B \\ A &\rightarrow aA \mid b \\ B &\rightarrow aB \mid bA \mid C \\ C &\rightarrow cC \mid A \end{aligned}$$

and compute chain$(S) = \{S, A, B, C\}$, chain$(S) = \{S', A, B, C\}$, chain$(A) = \{A\}$, chain$(B) = \{A, B, C\}$, and chain$(B) = \{A, C\}$. which gives us

$$\begin{aligned} S &\rightarrow BS' \mid S'B \mid aA \mid b \mid aB \mid bA \mid cC \\ S' &\rightarrow BS' \mid S'B \mid aA \mid b \mid aB \mid bA \mid cC \\ A &\rightarrow aA \mid b \\ B &\rightarrow aA \mid b \mid aB \mid bA \mid cC \\ C &\rightarrow aA \mid b \mid cC \qquad \clubsuit. \end{aligned}$$

2. (6 points) Construct an equivalent non-contracting grammar with non-recursive start. Also give a regular expression for the language.

$$\begin{aligned} G: S &\rightarrow ABCA \mid aBC \\ A &\rightarrow aA \mid BCCC \\ B &\rightarrow bBBC \mid \lambda \\ C &\rightarrow cCCB \mid \lambda \end{aligned}$$

with a non-recursive start and no chain rules.

♣ Let's find the regular expression first. $B \stackrel{*}{\Rightarrow} b(b \cup c)^* \cup \lambda$ and $C \stackrel{*}{\Rightarrow} c(b \cup c)^* \cup \lambda$ So BC and CB both derive to $(b \cup c)^*$. A derives to $a^*(b \cup c)^*$, so, lastly, S derives to $a^*(b \cup c)^* a^*(b \cup c)^*$. From this regular expression it is very easy to construct a grammar of

the proper type: (built from scratch, not converted. If you noticed, the problem asked you to construct it.)

$$\begin{aligned} G: S &\rightarrow aA \mid bB \mid cB \mid aC \mid bD \mid cD \mid a \mid b \mid c \mid \lambda \\ A &\rightarrow aA \mid bB \mid cB \mid a \mid b \mid c \\ B &\rightarrow bB \mid cB \mid aC \mid a \mid b \mid c \\ C &\rightarrow aC \mid bD \mid cD \mid a \mid b \mid c \\ D &\rightarrow cD \mid dD \mid c \mid d \end{aligned}$$ ♣

Just for fun, I'll do the conversion also. The start is not recursive, but NULL $= \{S, A, B, C\}$, so the grammar converts to

$$\begin{aligned} G: S &\rightarrow ABCA \mid BCA \mid ACA \mid ABA \mid BCA \mid AB \mid AC \mid AA \mid BC \mid BA \mid CA \mid A \mid B \mid C \\ &\quad \mid aBC \mid aB \mid aC \mid a \\ A &\rightarrow aA \mid a \mid BCCC \mid BCC \mid BC \mid B \mid CCC \mid CC \mid C \\ B &\rightarrow bBBC \mid bBC \mid bBB \mid bB \mid bC \mid b \\ C &\rightarrow cCCB \mid cCB \mid cCC \mid cC \mid cB \mid c \end{aligned}$$

Unfortunately that introduces some chain rules: chain$(S) = \{S, A, B, C\}$, chain$(A) = \{A, B, C\}$, chain$(B) = \{B\}$, and chain$(C) = \{C\}$. So the final form is

$$\begin{aligned} G: S &\rightarrow ABCA \mid BCA \mid ACA \mid ABA \mid BCA \mid AB \mid AC \mid AA \mid BC \mid BA \mid CA \mid aBC \mid aB \mid \\ &\quad aC \mid a \mid aA \mid BCCC \mid BCC \mid BC \mid CCC \mid CC \mid bBBC \mid bBC \mid bBB \mid bB \mid bC \mid b \mid \\ &\quad cCCB \mid cCB \mid cCC \mid cC \mid cB \mid c \\ A &\rightarrow aA \mid a \mid BCCC \mid BCC \mid BC \mid CCC \mid CC \mid bBBC \mid bBC \mid bBB \mid bB \mid bC \mid b \mid \\ &\quad cCCB \mid cCB \mid cCC \mid cC \mid cB \mid c \\ B &\rightarrow bBBC \mid bBC \mid bBB \mid bB \mid bC \mid b \\ C &\rightarrow cCCB \mid cCB \mid cCC \mid cC \mid cB \mid c \end{aligned}$$

From this form it is also possible, to get to the regular expression.