CS 321H Homework 4

Lyell Read

---

1. (5 pts) Convert the grammar below to CNF.

$G = (V, T, S, P)$ where

$V = \{S, A, B, C, D\}$

$T = \{0, 1, 2\}$

$P =$

$$
\begin{array}{lcl}
\mathrm{S} & \rightarrow & \mathrm{A} \;\;|\;\; \mathrm{ABD} \;\;|\;\; \mathrm{0BB} \\
\mathrm{A} & \rightarrow & \mathrm{0} \;\;|\;\; \mathrm{BAA} \\
\mathrm{B} & \rightarrow & \mathrm{BB} \;\;|\;\; \mathrm{1} \;\;|\;\; \mathrm{2} \;\;|\;\; \lambda \\
\mathrm{C} & \rightarrow & \mathrm{CD} \;\;|\;\; \mathrm{0} \\
\mathrm{D} & \rightarrow & \mathrm{D1} \;\;|\;\; \mathrm{DD}
\end{array}
$$

Answer:

- Eliminate start symbols from RHS (none)
- Remove lambda symbols

$$
\begin{array}{lcl}
\mathrm{S} & \rightarrow & \mathrm{A} \\
\mathrm{S} & \rightarrow & \mathrm{ABD} \\
\mathrm{S} & \rightarrow & \mathrm{aBB} \\
\mathrm{A} & \rightarrow & \mathrm{b} \\
\mathrm{A} & \rightarrow & \mathrm{BAA} \\
\mathrm{B} & \rightarrow & \mathrm{BB} \\
\mathrm{B} & \rightarrow & \mathrm{b} \\
\mathrm{B} & \rightarrow & \mathrm{c} \\
\mathrm{D} & \rightarrow & \mathrm{Db} \\
\mathrm{D} & \rightarrow & \mathrm{DD} \\
\mathrm{S} & \rightarrow & \mathrm{AD} \\
\mathrm{S} & \rightarrow & \mathrm{a} \\
\mathrm{S} & \rightarrow & \mathrm{aB} \\
\mathrm{A} & \rightarrow & \mathrm{AA} \\
\mathrm{B} & \rightarrow & \mathrm{B}
\end{array}
$$

- Eliminate unit productions

$$
\begin{array}{lcl}
\mathrm{S} & \rightarrow & \mathrm{ABD} \\
\mathrm{S} & \rightarrow & \mathrm{aBB} \\
\mathrm{A} & \rightarrow & \mathrm{b} \\
\mathrm{A} & \rightarrow & \mathrm{BAA} \\
\mathrm{B} & \rightarrow & \mathrm{BB} \\
\mathrm{B} & \rightarrow & \mathrm{b} \\
\mathrm{B} & \rightarrow & \mathrm{c} \\
\mathrm{D} & \rightarrow & \mathrm{Db} \\
\mathrm{D} & \rightarrow & \mathrm{DD} \\
\mathrm{S} & \rightarrow & \mathrm{AD} \\
\mathrm{S} & \rightarrow & \mathrm{a} \\
\mathrm{S} & \rightarrow & \mathrm{aB} \\
\mathrm{A} & \rightarrow & \mathrm{AA}
\end{array}
$$

$$\begin{array}{lcl} S & \rightarrow & b \\ S & \rightarrow & BAA \\ S & \rightarrow & AA \end{array}$$

- Eliminate useless productions

$$\begin{array}{lcl} S & \rightarrow & aBB \\ S & \rightarrow & a \\ S & \rightarrow & aB \\ S & \rightarrow & b \\ S & \rightarrow & BAA \\ S & \rightarrow & AA \\ A & \rightarrow & AA \\ B & \rightarrow & c \\ B & \rightarrow & b \\ B & \rightarrow & BB \\ A & \rightarrow & BAA \\ A & \rightarrow & b \end{array}$$

Convert to Chomsky

$$\begin{array}{lcl} S & \rightarrow & Ba \ | \ D2 \\ D2 & \rightarrow & BB \\ S & \rightarrow & a \\ S & \rightarrow & Ba \ | \ B \\ Ba & \rightarrow & a \\ S & \rightarrow & b \\ S & \rightarrow & BD1 \\ S & \rightarrow & AA \\ A & \rightarrow & b \\ A & \rightarrow & BD1 \\ D1 & \rightarrow & AA \\ A & \rightarrow & AA \\ B & \rightarrow & BB \\ B & \rightarrow & b \\ B & \rightarrow & c \end{array}$$

2. (10 pts) Consider the CNF grammar

$G = (V, T, S, P)$ where

$V = \{S, A, B, C, D\}$

$T = \{a, b, c\}$,

$S = S$

$P =$

$$\begin{array}{lcl} S & \rightarrow & AB \ | \ AD \ | \ AC \\ A & \rightarrow & AA \ | \ a \\ B & \rightarrow & BB \ | \ AB \ | \ b \\ C & \rightarrow & AC \ | \ DC \ | \ c \\ D & \rightarrow & DD \ | \ b \ | \ c \end{array}$$

Use the CYK algorithm to determine if the strings $w_1 =$ babbc and $w_2 =$ aaaabb are in the language $L(G)$. Show the DP table. If the string is in $L(G)$ construct the parse tree.

Response:

- $w_1 = babbc$

i/j	1	2	3	4	5
1	B,D				
2		A			
3			B,D		
4				B,D	
5					C,D

i/j	1	2	3	4	5
1	B,D	∅			
2		A	B,S		
3			B,D	B,D	
4				B,D	C,D
5					C,D

i/j	1	2	3	4	5
1	B,D	∅	B		
2		A	B,S	S,B	
3			B,D	B,D	C,D
4				B,D	C,D
5					C,D

i/j	1	2	3	4	5
1	B,D	∅	B	B	
2		A	B,S	S,B	S
3			B,D	B,D	C,D
4				B,D	C,D
5					C,D

i/j	1	2	3	4	5
1	B,D	∅	B	B	∅
2		A	B,S	S,B	S
3			B,D	B,D	C,D
4				B,D	C,D
5					C,D

String not accepted, as there is no S in top right corner.

- $w_2 = aaaabb$

i/j	1	2	3	4	5	6
1	A					
2		A				
3			A			
4				A		
5					B,D	
6						B,D

i/j	1	2	3	4	5	6
1	A	A				
2		A	A			
3			A	A		
4				A	S	
5					B,D	B,D
6						B,D

i/j	1	2	3	4	5	6
1	A	A	A			
2		A	A	A		
3			A	A	S	
4				A	S	S
5					B,D	B,D
6						B,D

i/j	1	2	3	4	5	6
1	A	A	A	A		
2		A	A	A	S	
3			A	A	S	S
4				A	S	S
5					B,D	B,D
6						B,D

i/j	1	2	3	4	5	6
1	A	A	A	A	S	
2		A	A	A	S	S
3			A	A	S	S
4				A	S	S
5					B,D	B,D
6						B,D

i/j	1	2	3	4	5	6
1	A	A	A	A	S	S
2		A	A	A	S	S
3			A	A	S	S
4				A	S	S
5					B,D	B,D
6						B,D

String accepted, as S is in top right corner.

Parse Tree:

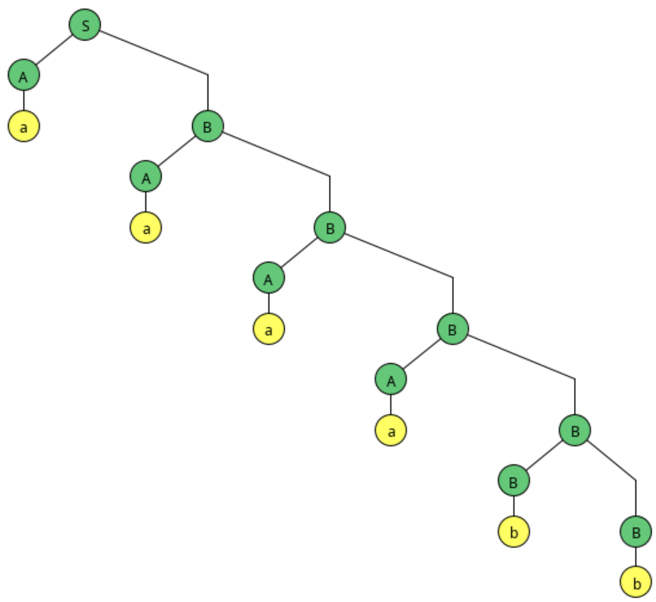

Figure 1: Parse Tree

3. (15 pts) Construct npda's that accept the following languages on $\Sigma = \{a, b\}$. Give both a verbal explanation on how your npda works and the formal definition including the transition function and/or transition graph. You must use JFLAP. Submit the transition graph in the HW pdf and the JFLAP code file for each problem.

- a. $L = \{a^n b^{2n} : n \geq 0\}$

NPDA:

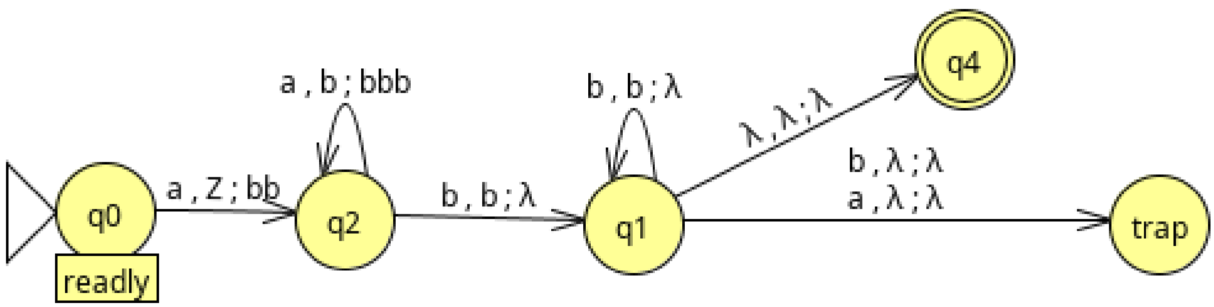

Figure 2: NPDA for Excercise 3a

Explanation:

This NPDA works as a result of the stack being used as a tally of the required remaining `b` characters - that is, every time it reads an `a`, it adds two `b` characters to the stack, and when it reads the first `b`, it begins pulling from the stack.

- b. $L = \{w : n_a(w) = 2n_b(w)\}$

NPDA:

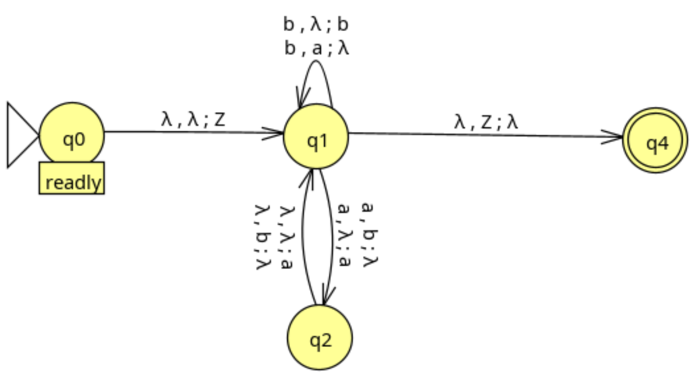

Figure 3: NPDA for Excercise 3b

Credit: https://scholar.harvard.edu/files/harrylewis/files/ps4a_solutions.pdf

Explanation:

This NPDA operates very simply and beautifully to ensure that when the stack is empty, the number of `b` characters is double the number of `a` characters. It does this using the nondeterminism of an NDPA. After pushing an end of stack character, the program decides whether to push a `b` or pop an `a` when it locates a `b` at the start of the string. It also makes a choice between the following:

1. Push a `b` and push another `b`
2. Push a `b` and pop an `a`
3. Push an `a` and push a `b`
4. Push an `a` and pop an `a`

when it encounters an `a` character at the start of the string. Finally, when it encounters the end of stack character at the top of the stack, it will succeed.

- c. $L = \{wcw^R : w \in \{a, b\}*\}$

NPDA:

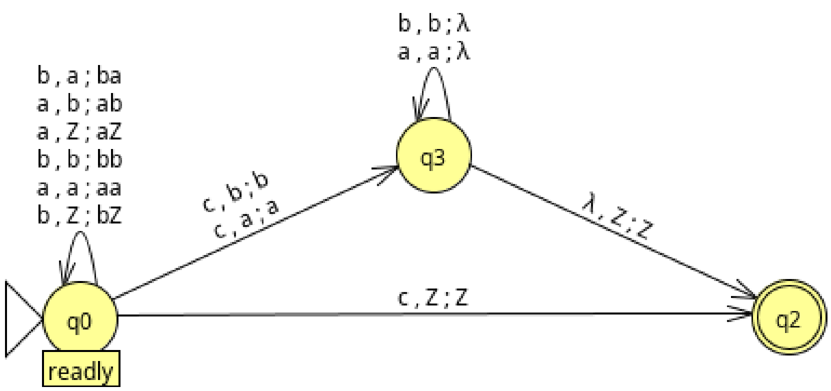

Figure 4: NPDA for Excercise 3c

Sources:

https://www.geeksforgeeks.org/npda-for-accepting-the-language-l-wwr-w-ab/

http://www.cs.sjsu.edu/faculty/pollett/154.3.07s/Hw4.pdf

Explanation:

This NPDA is essentially a palindrome checker that specifies that there must be a `c` in the middle of the palindrome, and the word alphabet is $\Sigma = \{a, b\}$. It does this by pushing a copy of the start of the palindrome, up to `c` on the stack, and verifying that the same string is read after `c`, before finishing. Even knowing that $R \geq 0$, we include a case to catch the string `c`, as this would be a string where $R = 1, w = \lambda$.