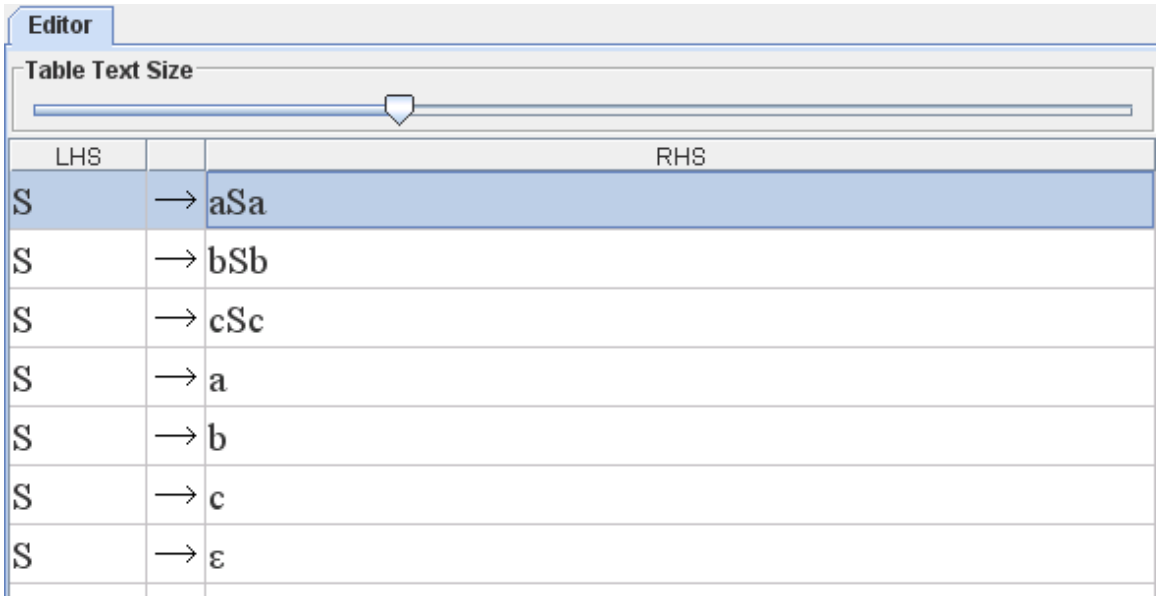
SJSU Students
CS 154 Section 01
1:30-2:45
04/06/11

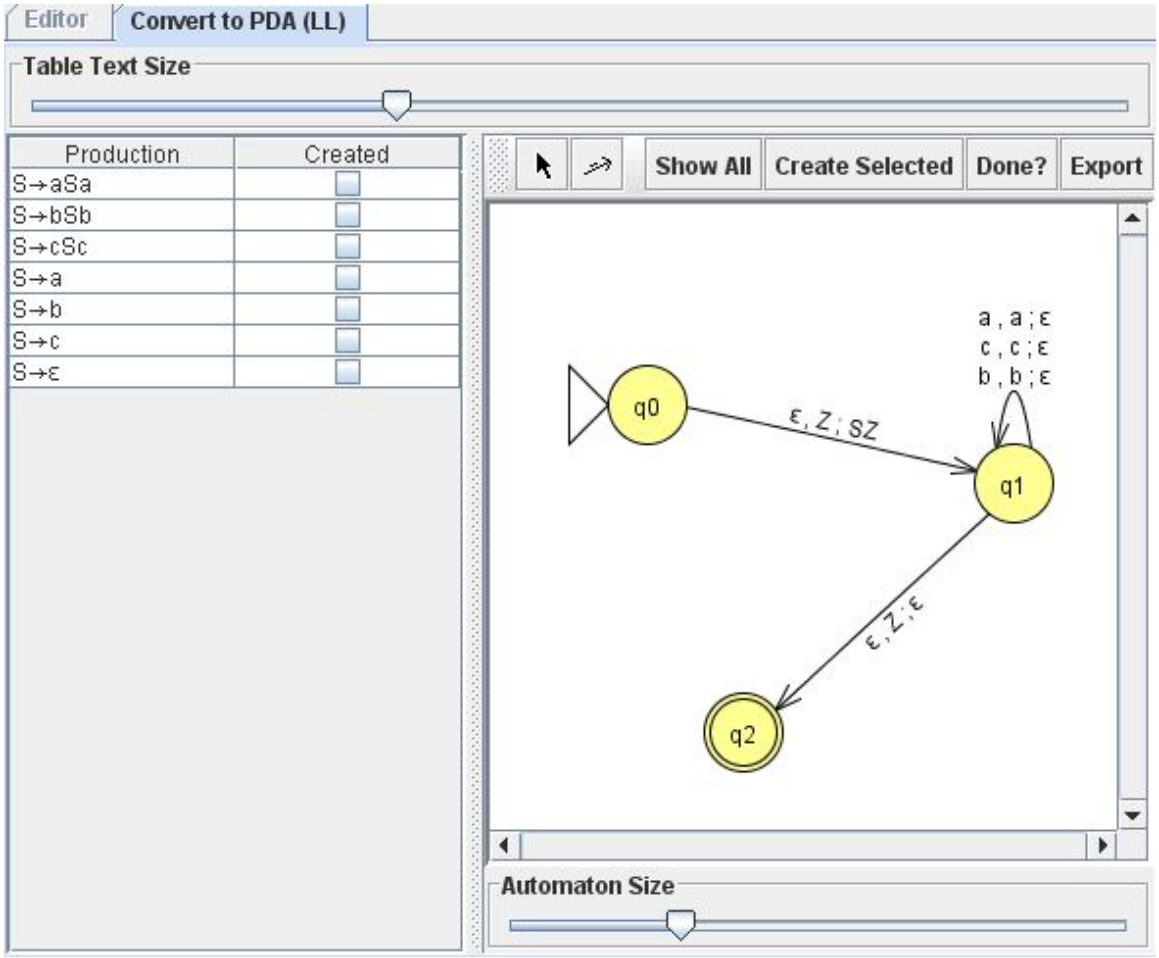
Homework #3

1. Context free grammar for palindromes:

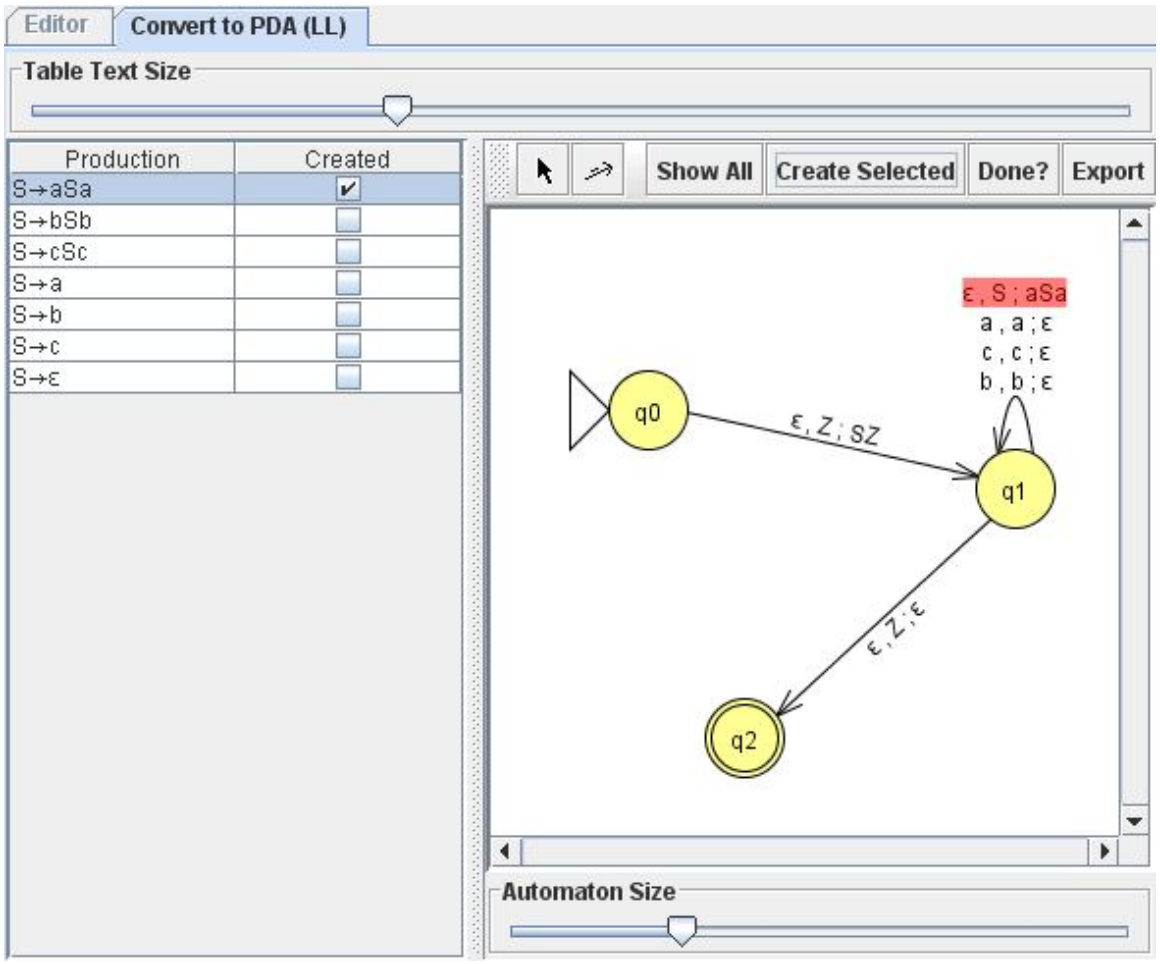
S --> aSa | bSb | cSc | a | b | c | ε



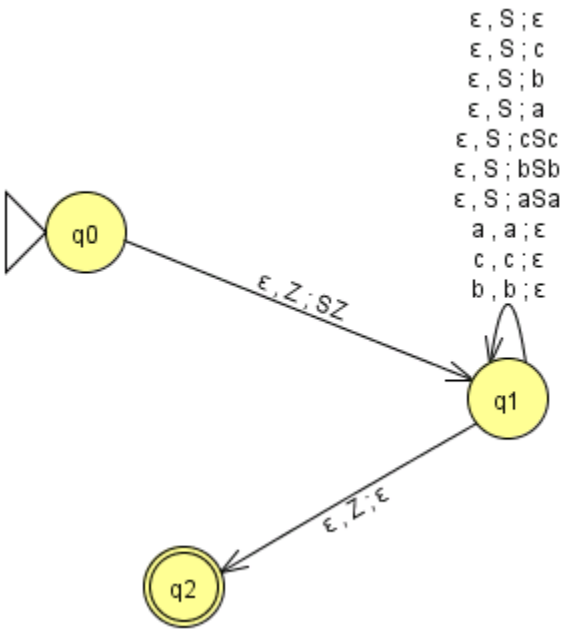
First of all, to enter the grammar into JFLAP, select grammar from the main JFLAP menu. Then, add each rule into the boxes provided. The left most box takes the left side of the productions. Next, the middle box will always take the arrow for the production. Finally, the right box will take the right side of the production.



After entering the grammar, click convert from the topmost menu and choose convert to PDA. The above is picture is what is obtained after converting. In the first transition from q0 to q1, S, the string that needs to be processed by the automaton, and the variable Z are pushed onto the stack. The transition also says that ε is all that is required to get from q0 to q1. Now, notice that the only transitions on q1 are those that involve popping symbols off the stack. For example: in state q1 a, a; ε will pop ‘a’ off the stack if ‘a’ is the next symbol that needs to processed. The final transition between q1 and q2 can only be reached if it is the end of the string being processed. Also, this transition says that the transition can only be followed to q2 if the variable Z can be popped off the stack. So, in order to reach the accept state, the stack must contain only the variable Z, while the entirety of S must be popped off.



The next step involves adding the transitions in state q1 that push variables and terminals onto the stack. Having rules to replace every variable in S with terminals is necessary to get to the accept state, since every transition in q1 that pops a variable off the stack requires a terminal input. For example: a, a; ε will only work if there is variable 'a' in the input and a variable 'a' on the stack. To add the necessary transitions, click on one of the production rules in the table on the left and then click the 'Create Selected' button. This will add the transition for the production in q1. As shown in the picture, for the first production S → aSa, the transition rule generated is: ε, S; aSa. This rule tells us that on an empty string input, pop S off the stack and push aSa onto the stack.



After repeating the process of adding production as transitions, the PDA pictured above is generated.

2. Context free grammar:

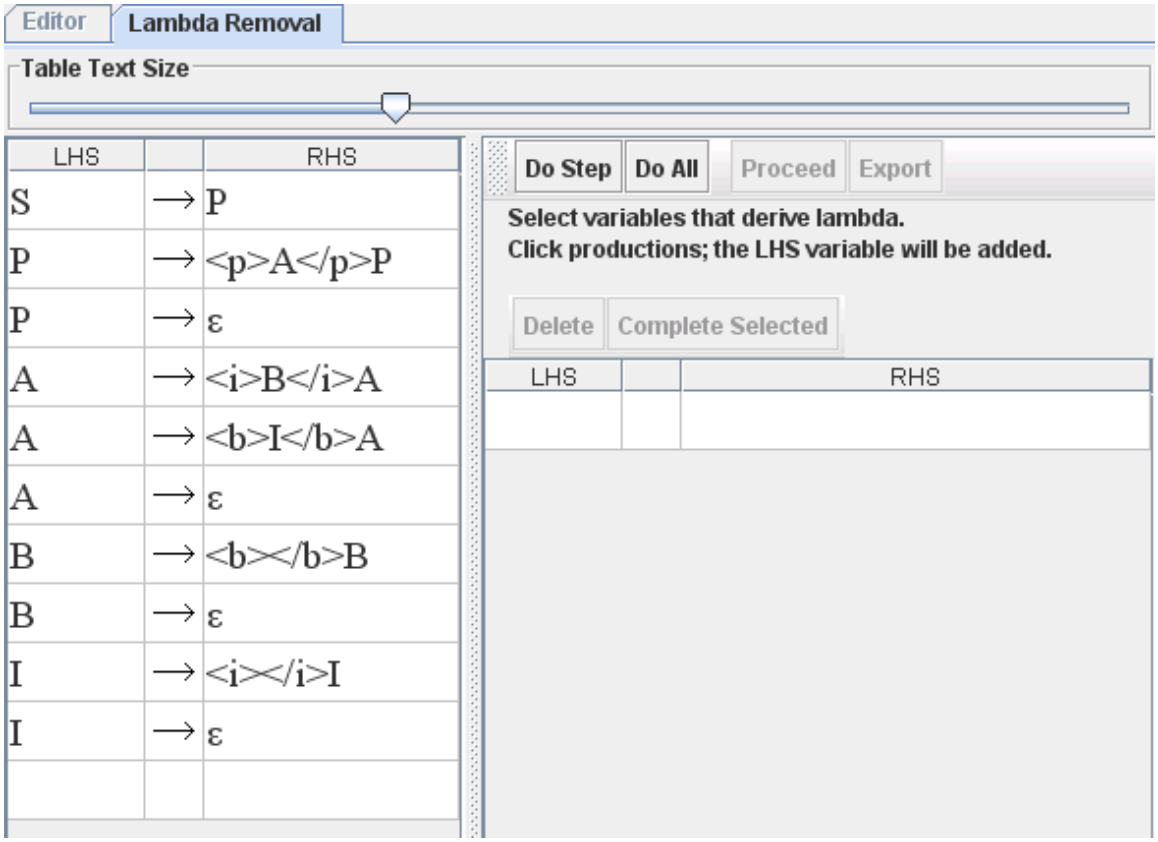
S → P

P → A P | ε

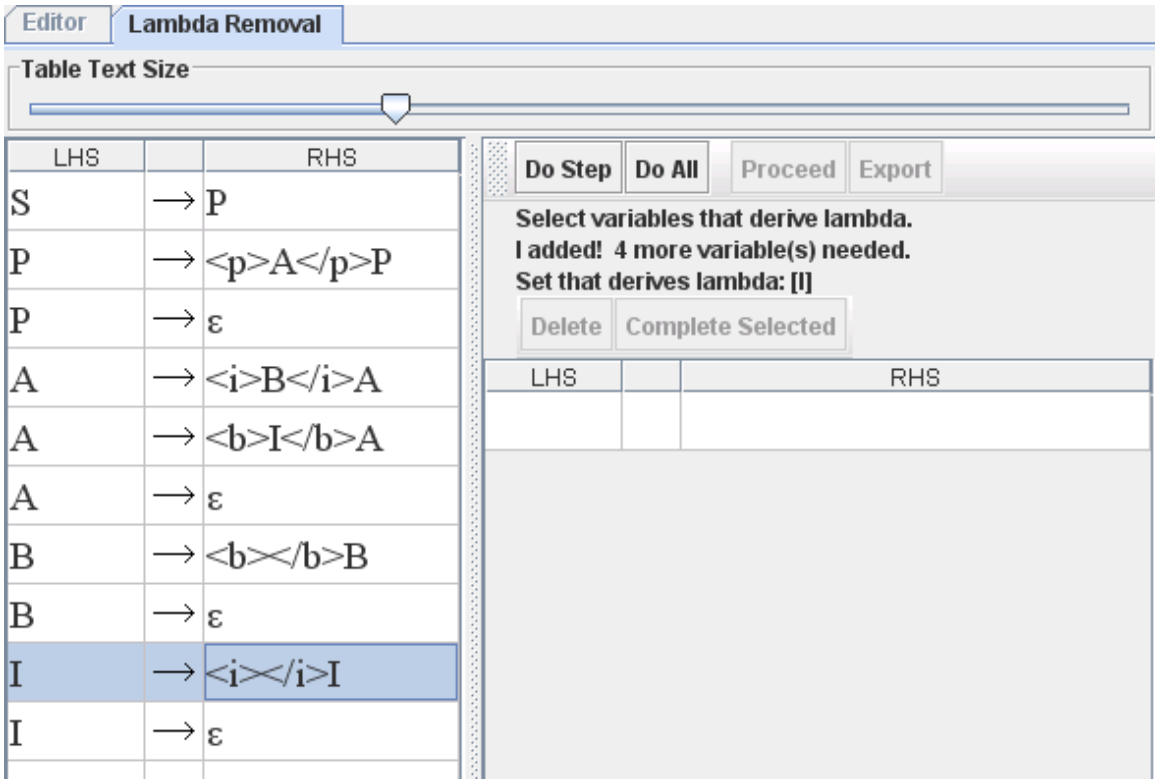
A → BA | IA | ε

B → B | ε

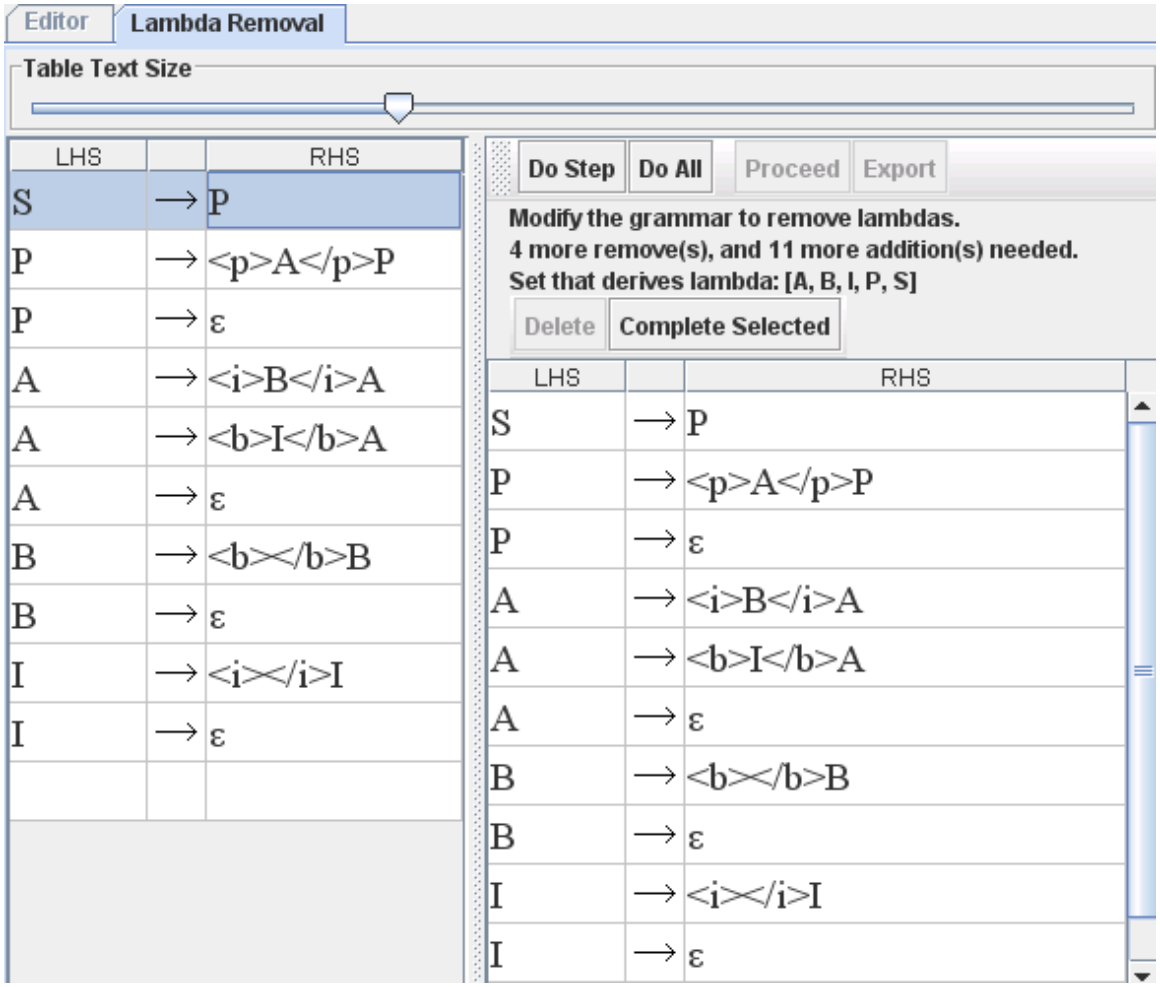
I → I | ε



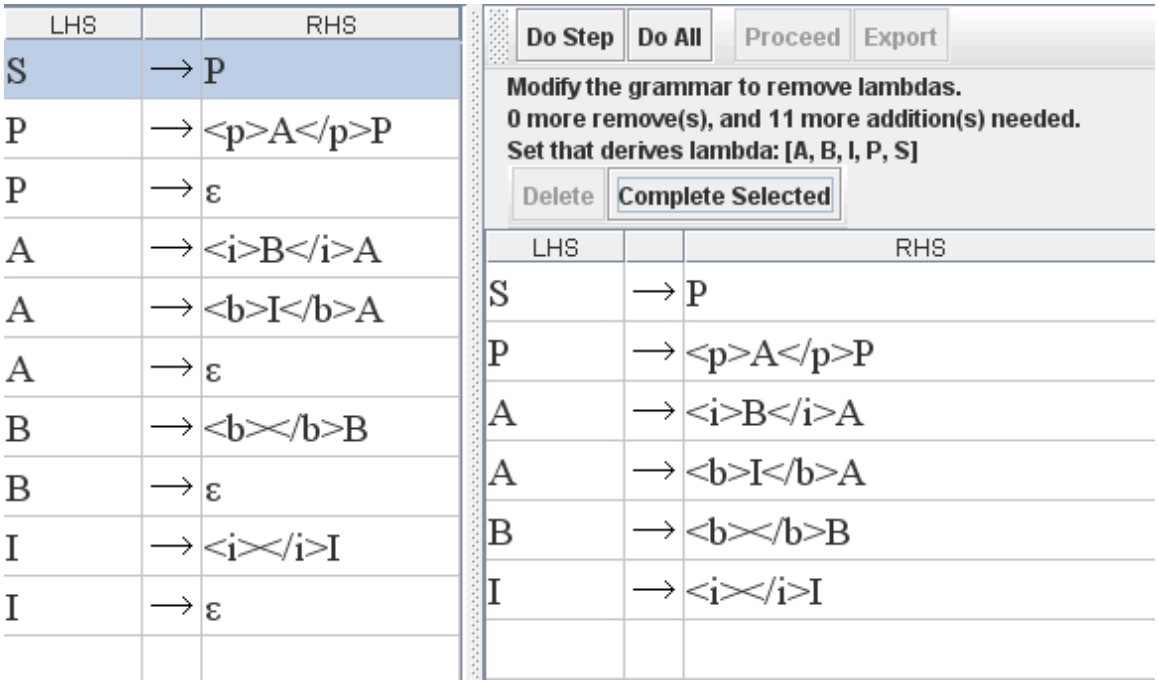
Once again, the first step is to enter the grammar into JFLAP. After that, click 'Transform Grammar' from the Convert menu at the top. This will cause the above image to appear.



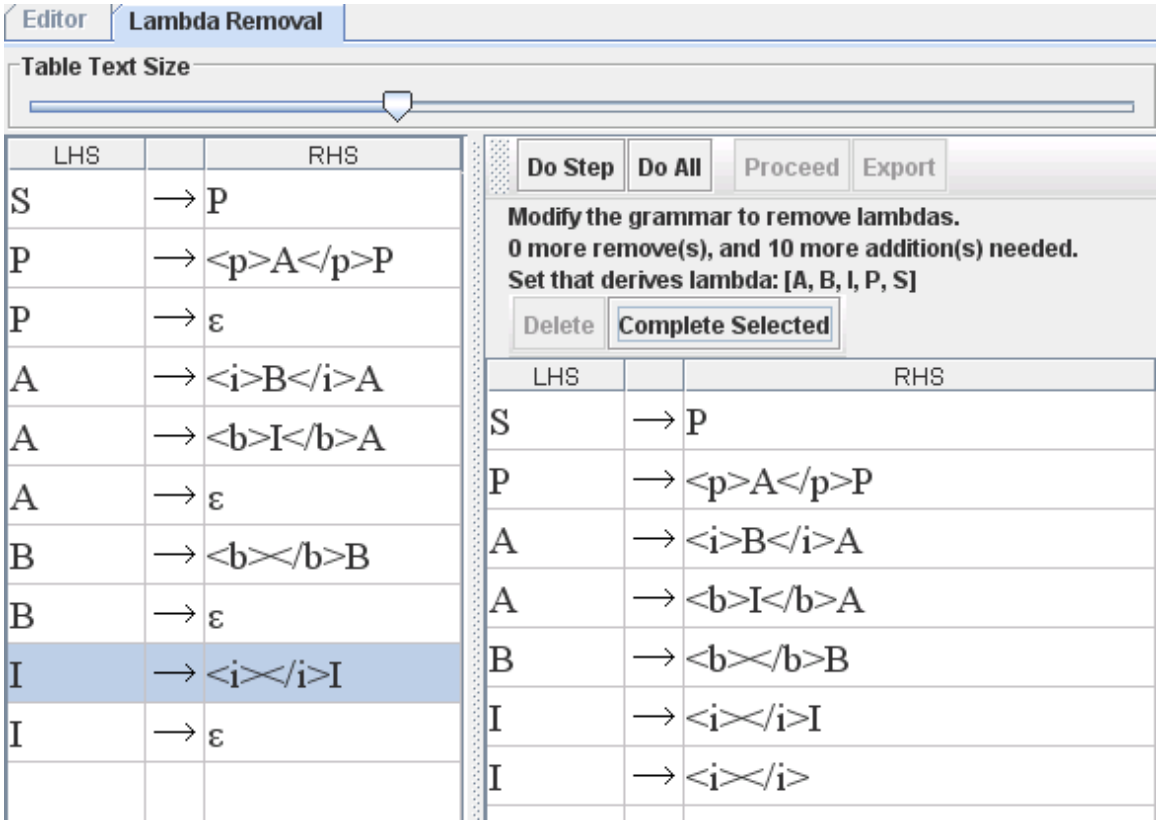
Now, variables that derive ε must be selected. To select a variable that derives ε, just click on the production rule, where that variable is on the left side of the arrow.



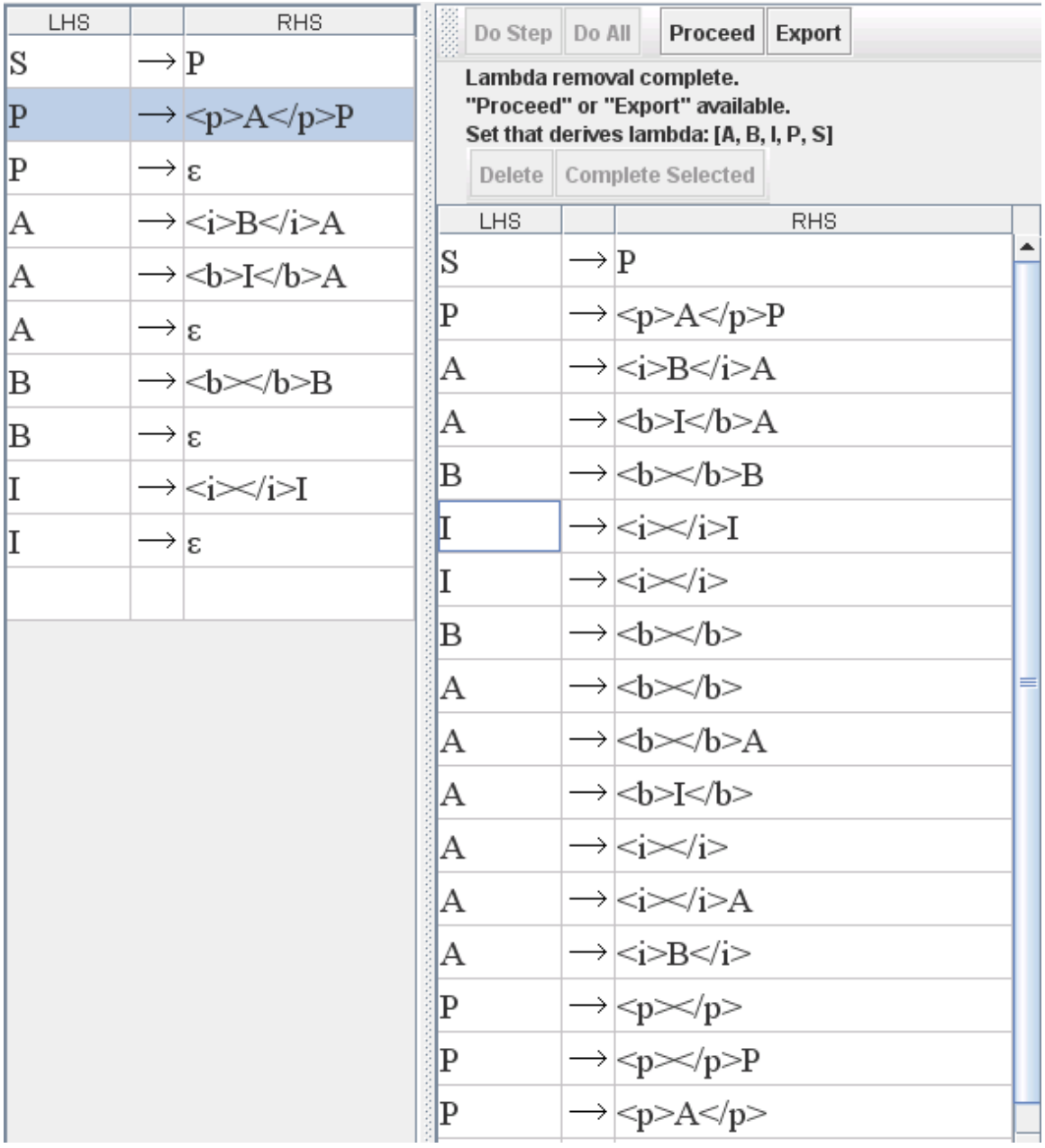
After all the variables that derive ε have been selected, the above image appears.



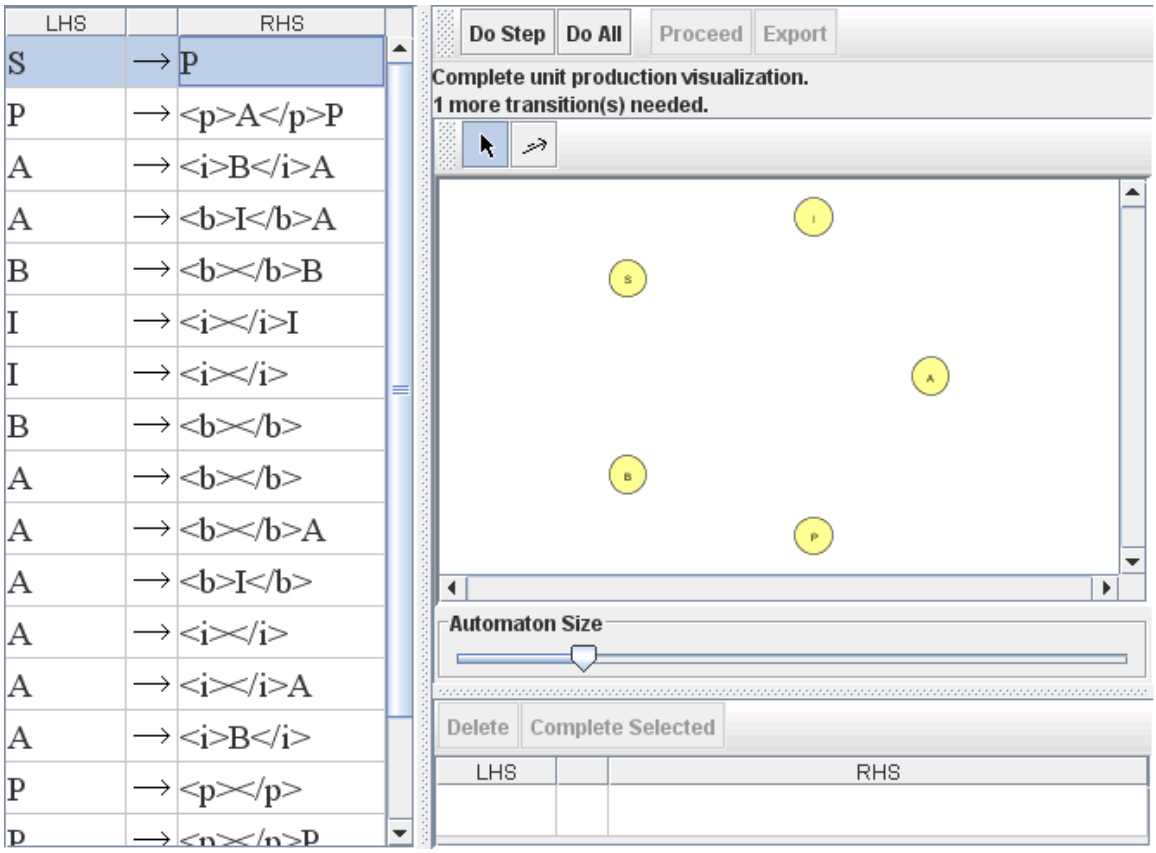
Next, delete all the rules that lead to ε in the table of productions that was generated on the right. This can be done by selecting a rule and clicking the delete button.



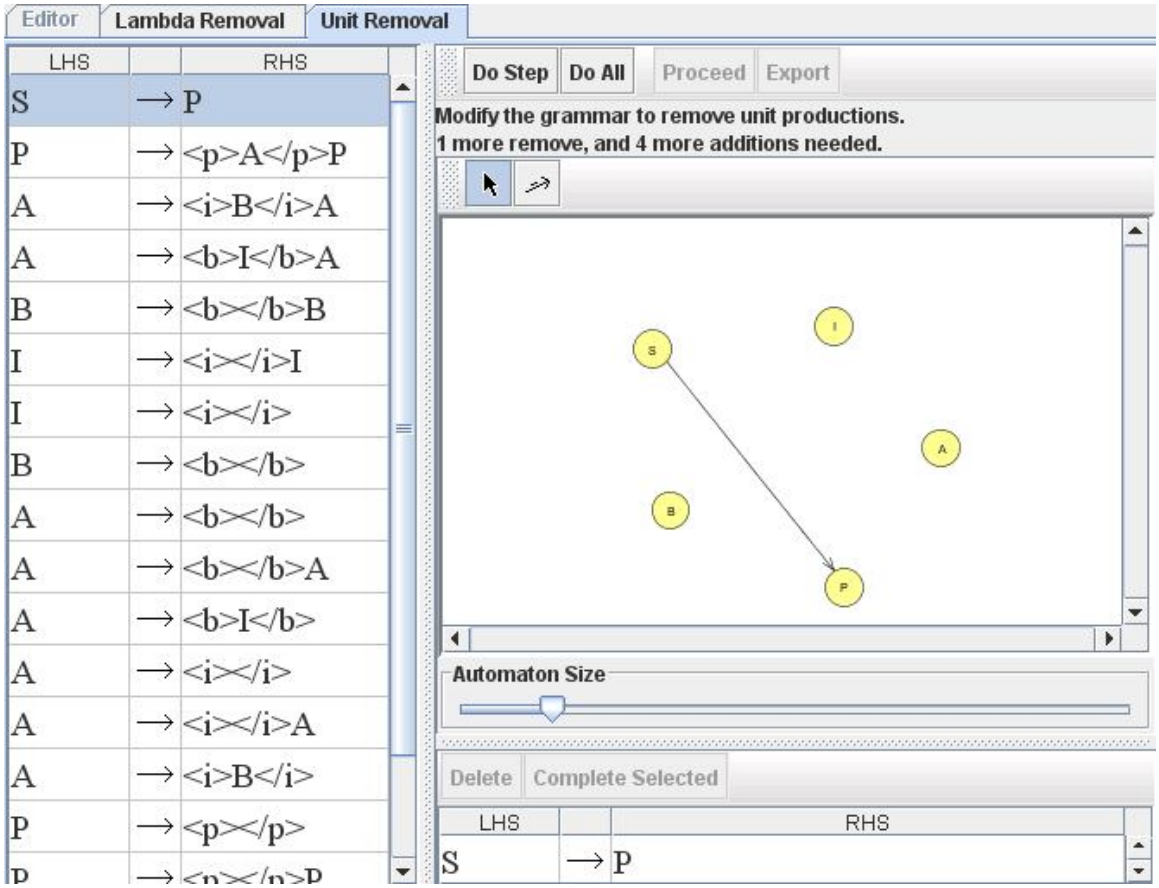
Now, for every ε rule that was removed, new productions need to be added so that the new grammar accepts the same set of strings as the original grammar. This can be manually done, by adding on the right. The alternate method is to just select a production rule in the left table and hitting the Complete Selected button. In the picture above a new rule for the variable 'I' is added. This is because the production I → ε was deleted. So, the original production rule, which was I → I | ε must become I → I | . This is why I → was added in the table on the right.



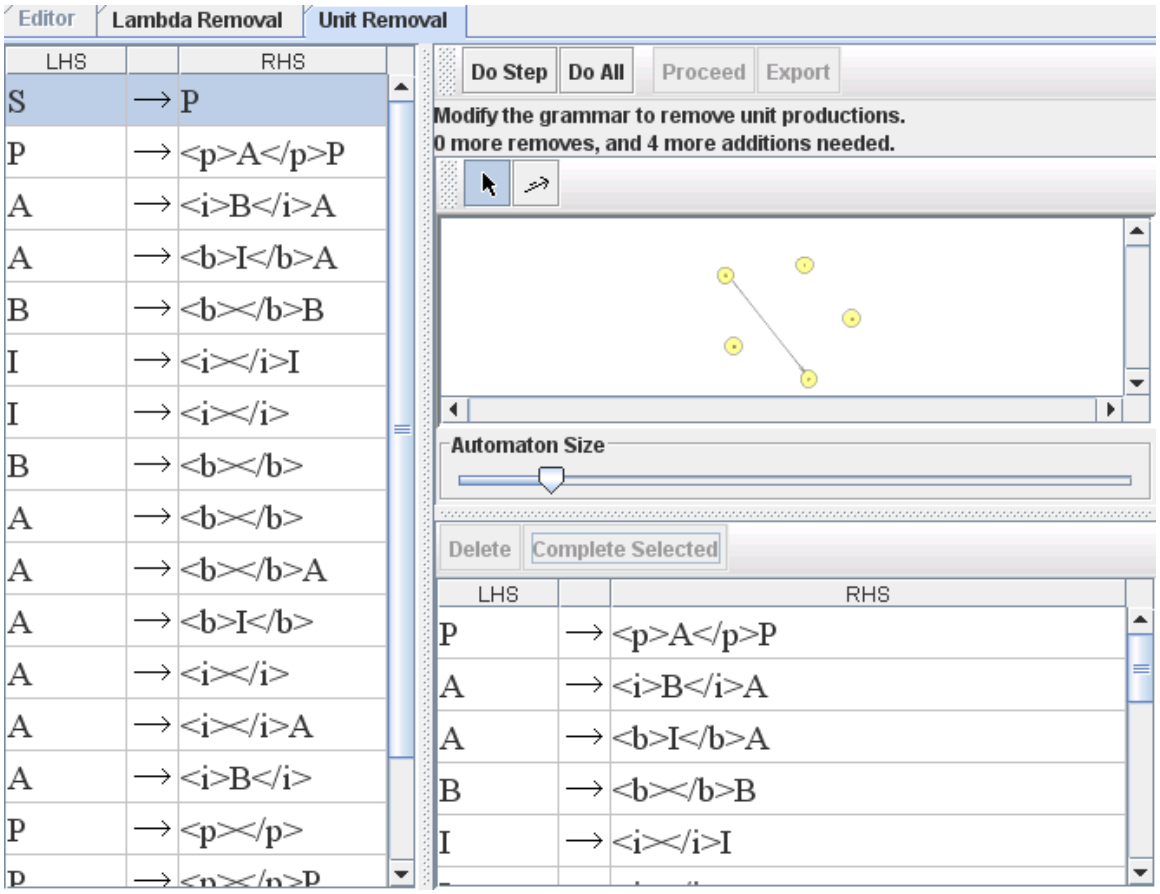
After adding all of the necessary rules, the above image is produced. Now, all ε rules have been removed and the new grammar has been fully modified to accept the same set of strings as the original language. Now the next step is to remove unit rules.



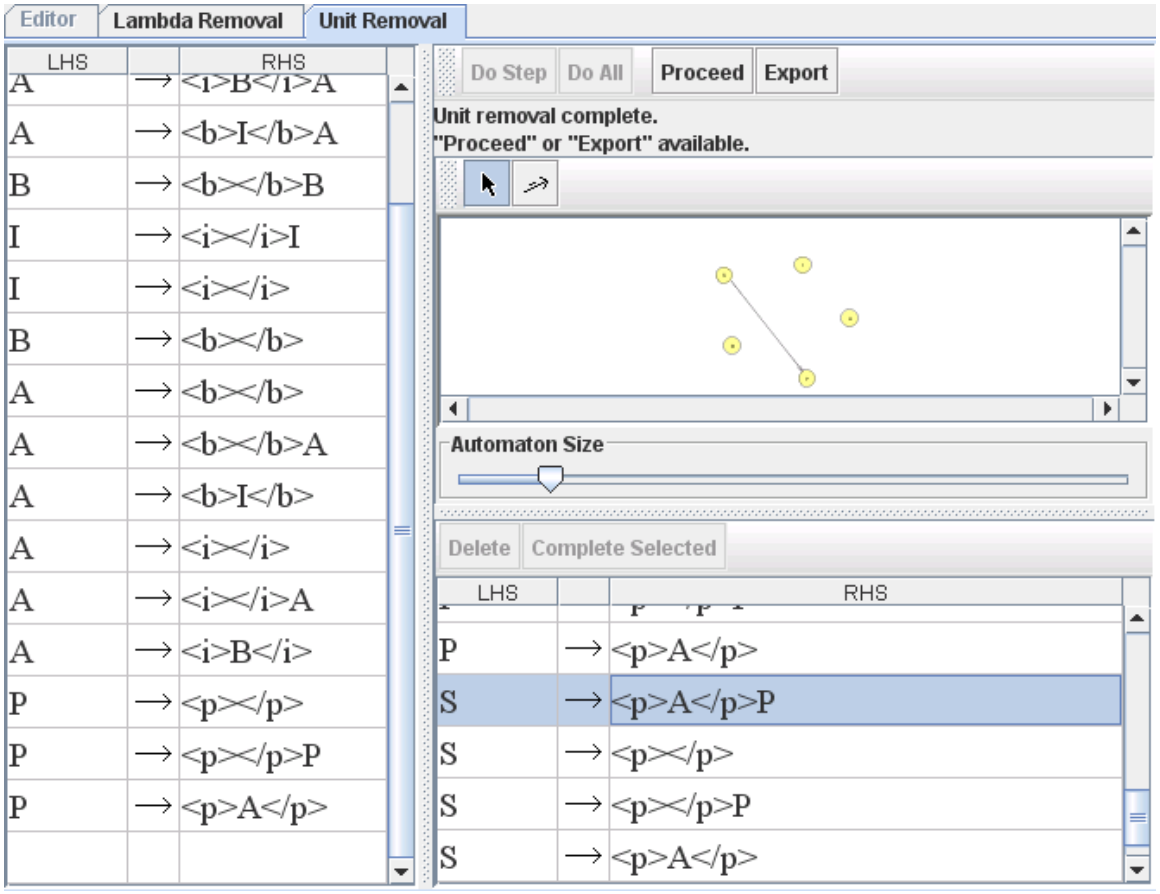
After clicking the Proceed button, the above image is displayed. The graph on the right has a node for every variable in the grammar.



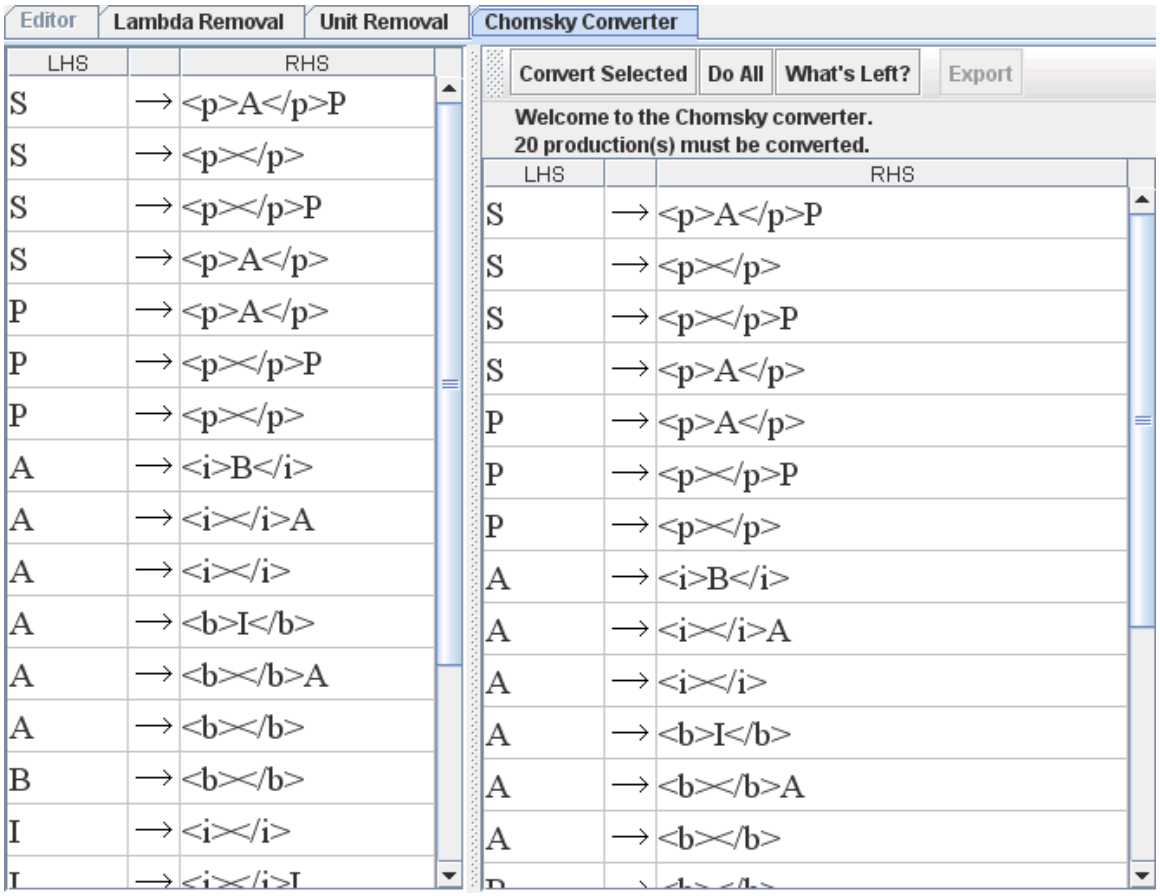
Now for each unit rule add a transition arrow. So, for the unit rule S → P, add a transition arrow from state S to state P as shown above.



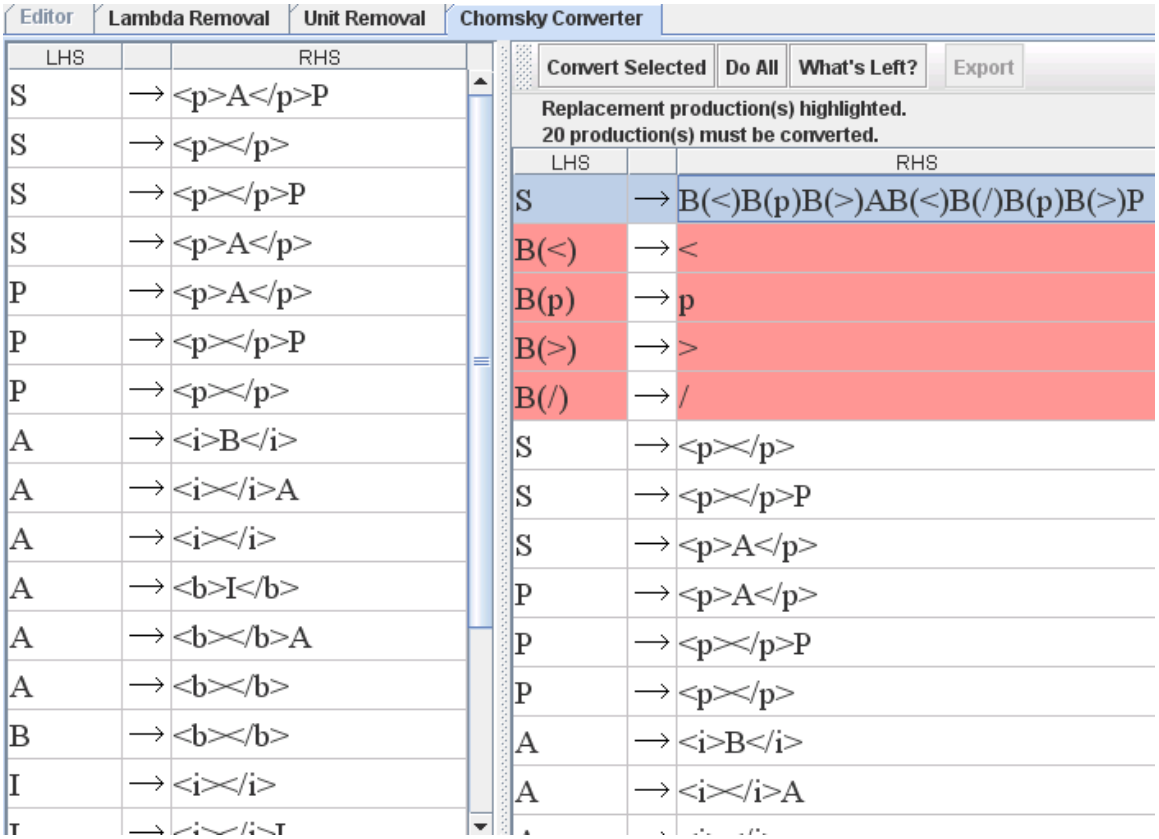
After that, delete any unit production rules in the bottom right table the same way ε rules were deleted. So, S → P is deleted from the table.



Now, just like in Epsilon removal, additional rules need to be added to the grammar to make it equal to the original grammar. So, since S → P was deleted, a new production with S on the left needs to be added for every production that had P on the left. This is why there are four new rules with S on the left side added into the new grammar.



After that, useless productions would be removed. However, since our grammar did not contain any useless production, JFLAP proceeds straight to the Chomsky Converter.



Now every production rule must have the form: A → BC or A → a. In these rules, A, B, and C are variables and ‘a’ is a terminal. One way to do this is to add more production rules manually and introduce new variables to get each production into CNF. The other way is to select a production and hit the ‘Convert Selected’ button. After converting every production that is not in CNF, the final grammar is obtained and the conversion to CNF is completed.

When the starting grammar for problem #2 is converted into CNF, JFLAP throws an error saying: 26 variables available, but 73 needed. So, the grammar entered into JFLAP needs be a little simplified to produce fewer variables in the CNF.

Context free grammar entered into JFLAP:

S → P

P → CADP | ε

A → FBGA | JIKA | ε

B → JKB | ε

I → FGI | ε

C → p

D → /p

F → i

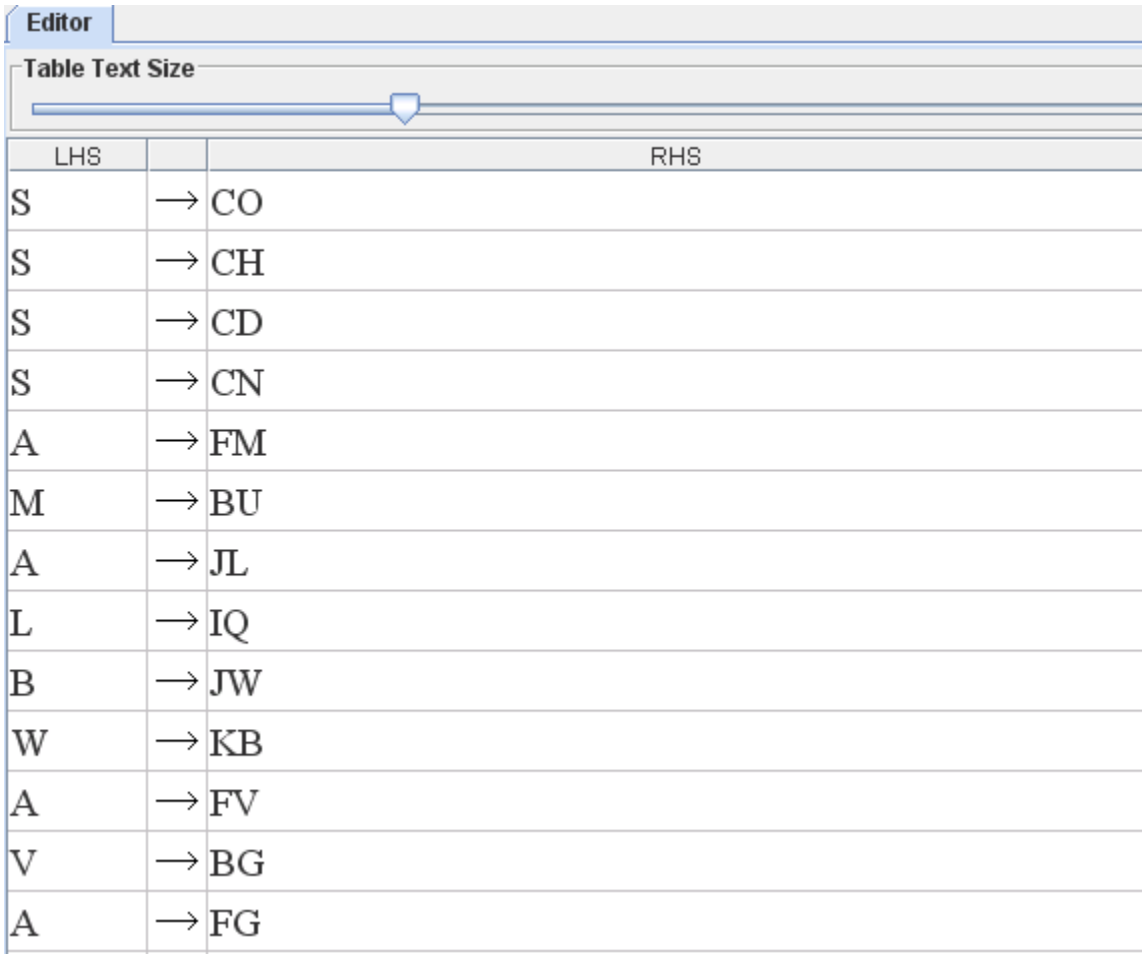
G → /i

J → b

K → /b

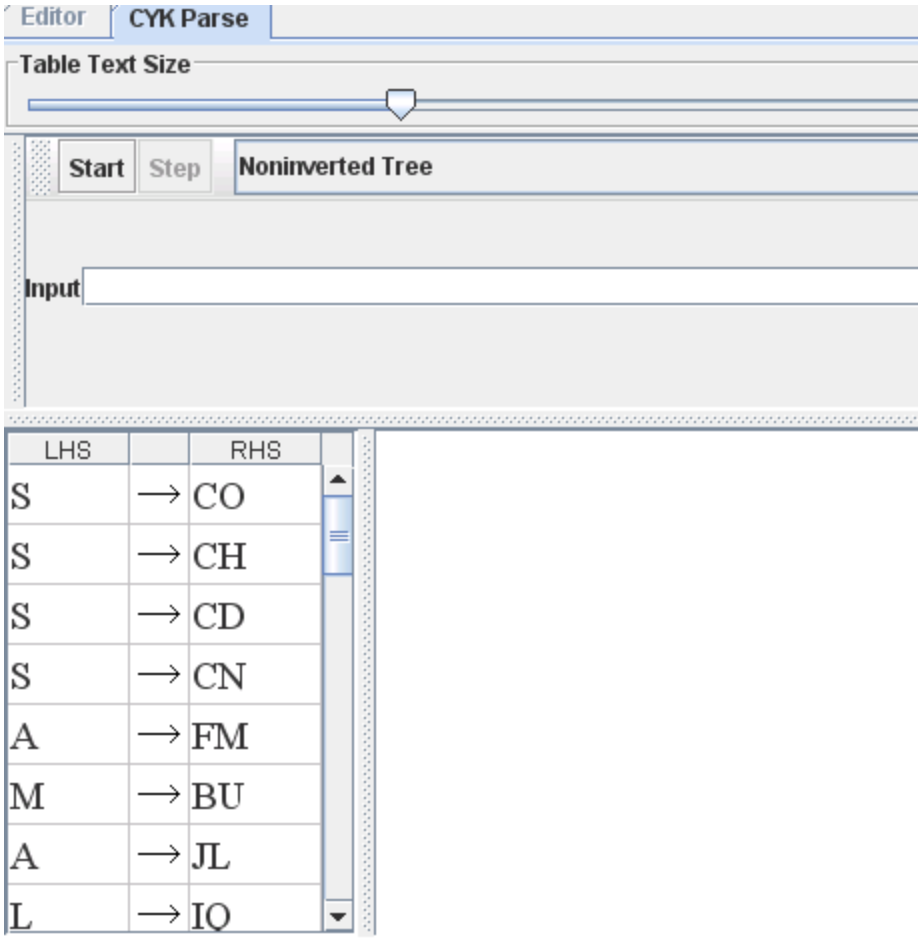
Note: in this grammar, the terminals < and > were excluded, so that JFLAP would not throw the error previously mentioned. So, those terminals are just assumed to be around all occurrences of the terminals p, /p, b, /b, i, and /i. The CNF generated from this grammar was used for problem #3.

3.

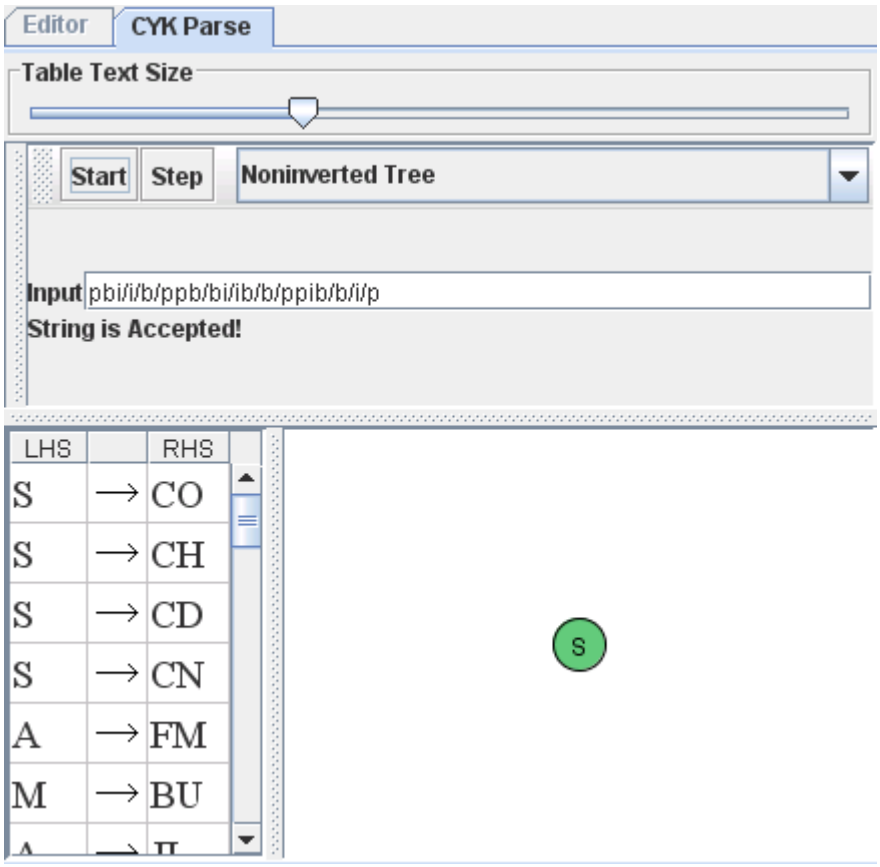


LHS		RHS
S	→	CO
S	→	CH
S	→	CD
S	→	CN
A	→	FM
M	→	BU
A	→	JL
L	→	IQ
B	→	JW
W	→	KB
A	→	FV
V	→	BG
A	→	FG

Once again, we start by inputting our grammar into JFLAP. This time it's the CNF of the grammar from problem 2.



After that, go to the input menu and choose CYK Parse to get to screen shown in the above image.



The next step is to enter the string that will be CYK parsed. In our case, since the grammar was modified so that JFLAP would not throw an error, the input string that is provided in the homework problem needs to also be modified. So, the terminals < and > need to discarded from the string.

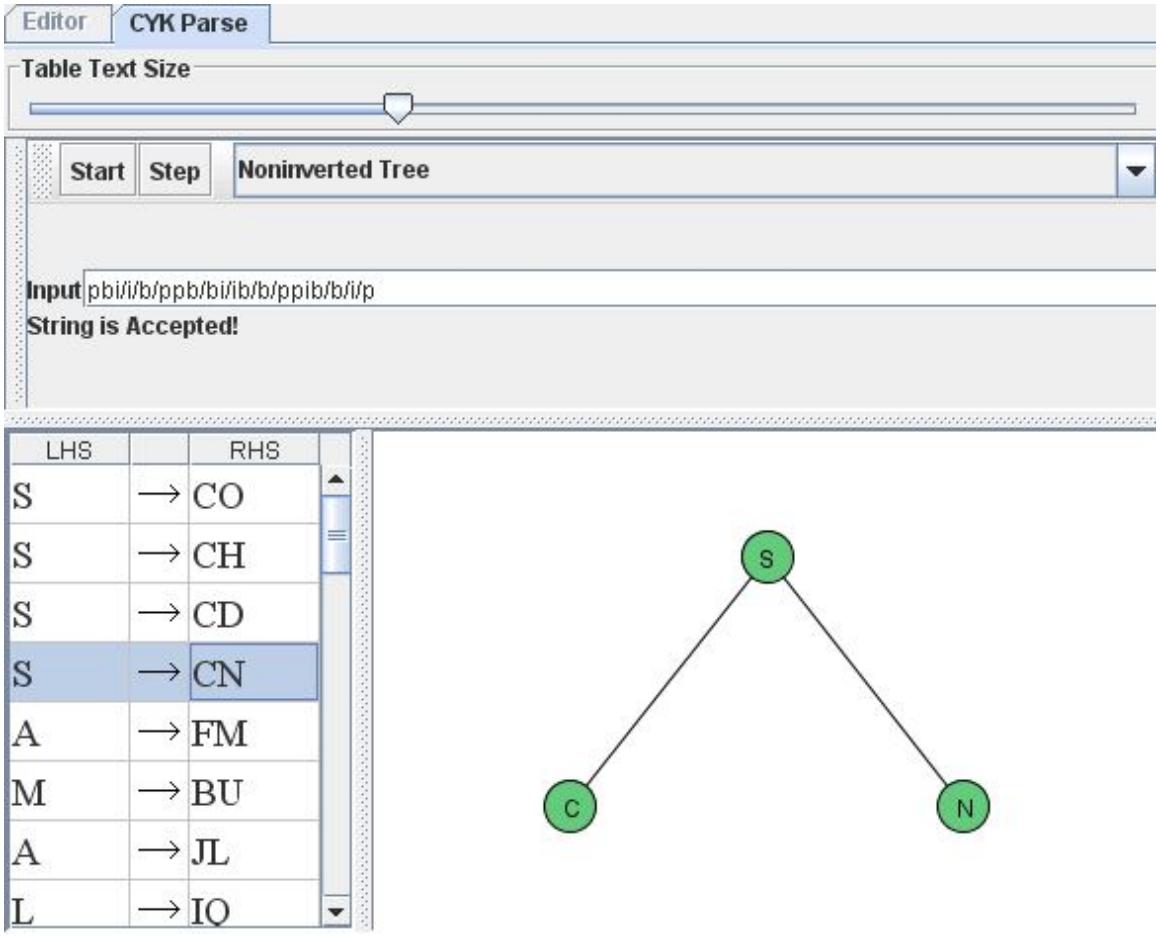
Homework input string provided:

Homework input string modified:

pbi/i/b/ppb/bi/ib/b/ppib/b/i/p

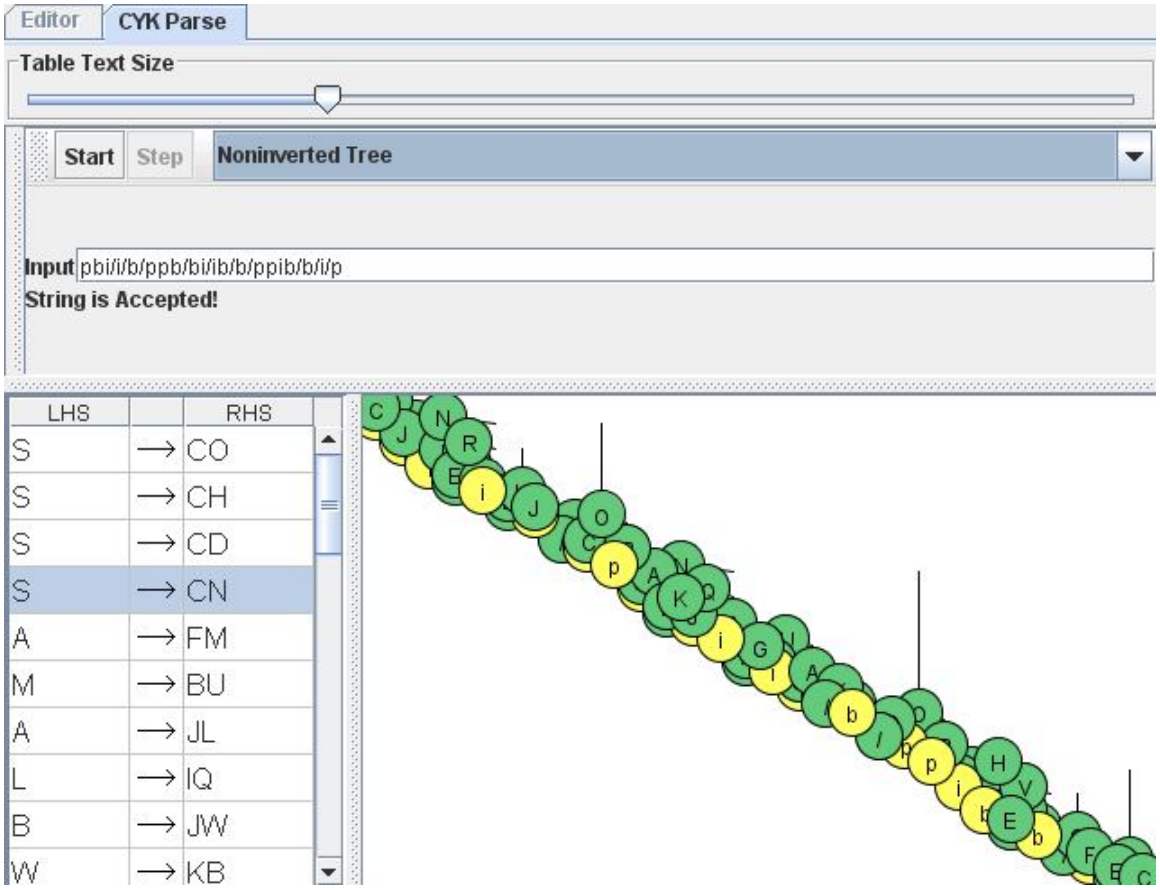
The modified version is the one we will be inputting into JFLAP.

The picture above is obtained after entering in the string and hitting the “Start” button. JFLAP immediately gives a confirmation that the string is accepted by the CNF grammar.

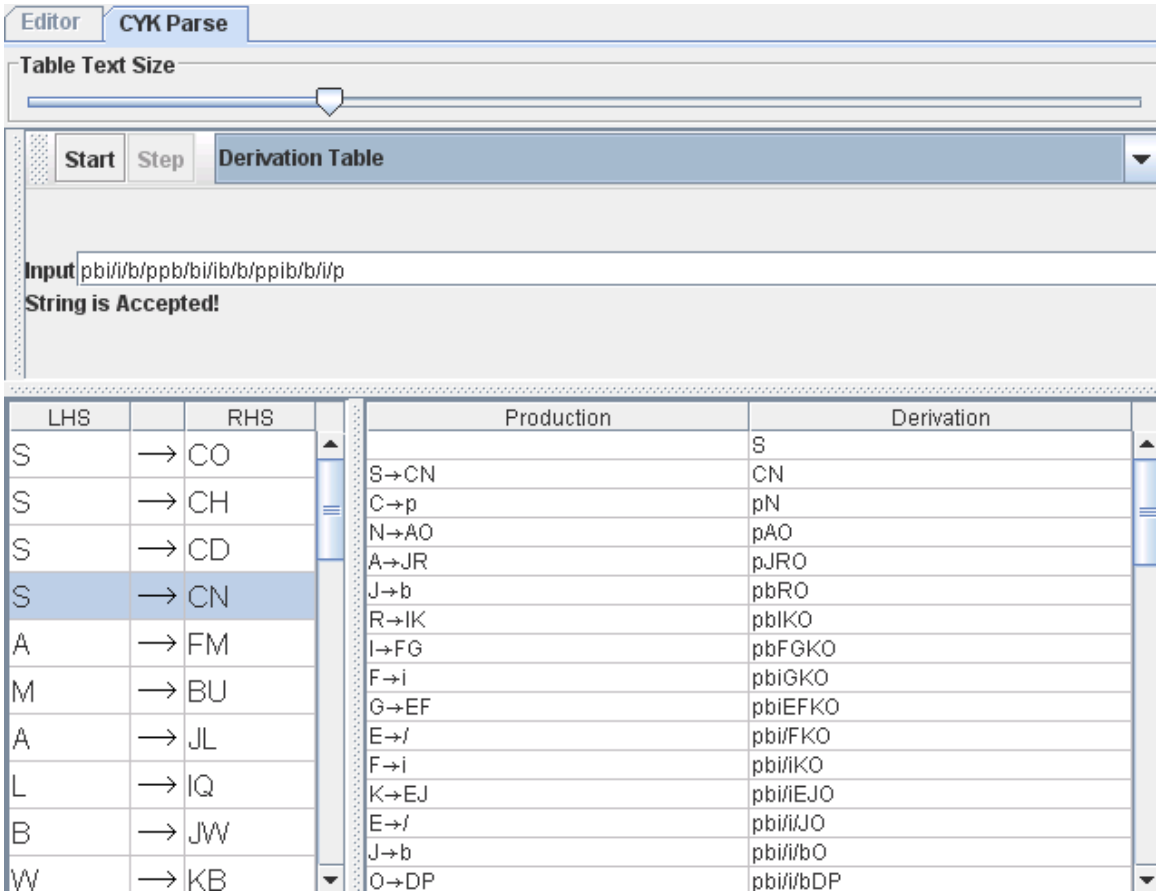


After that, click on the Step button to see which production was first applied on the start symbol in the derivation. So, in the above picture, to derive the input string from our start symbol S, the first production that must be used is

S → CN.

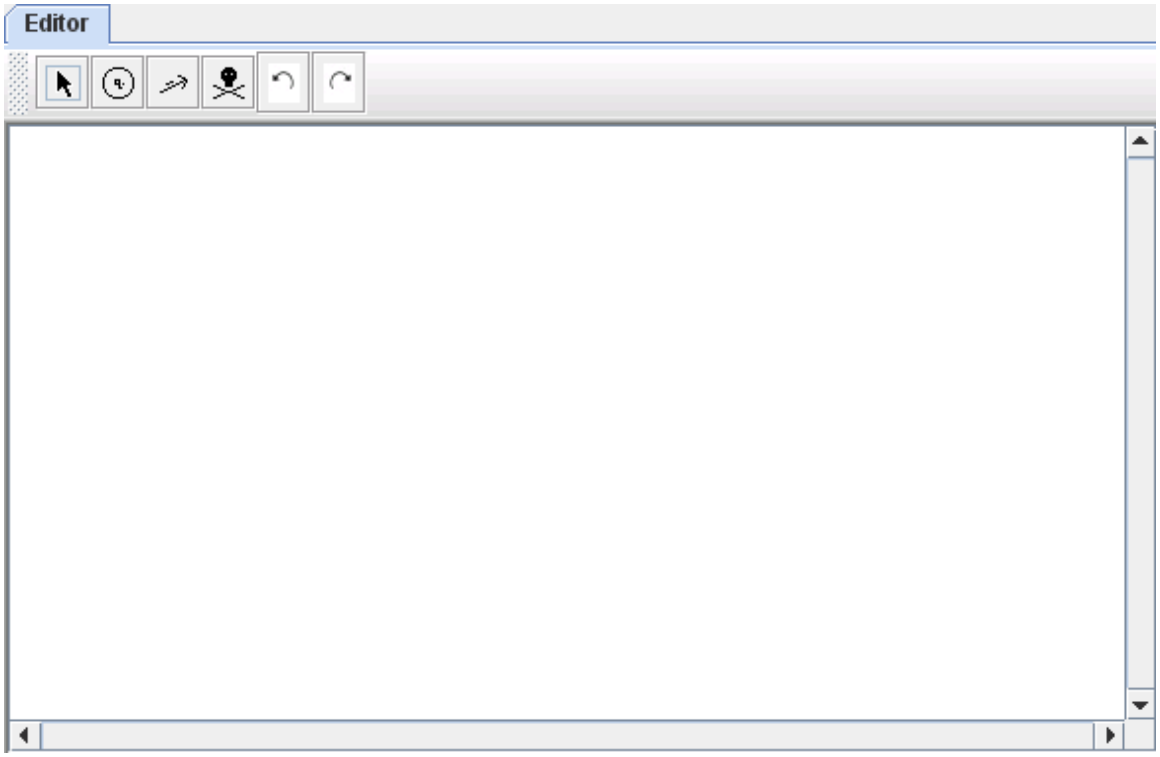


After clicking Step as many times as necessary, the entire tree is obtained.

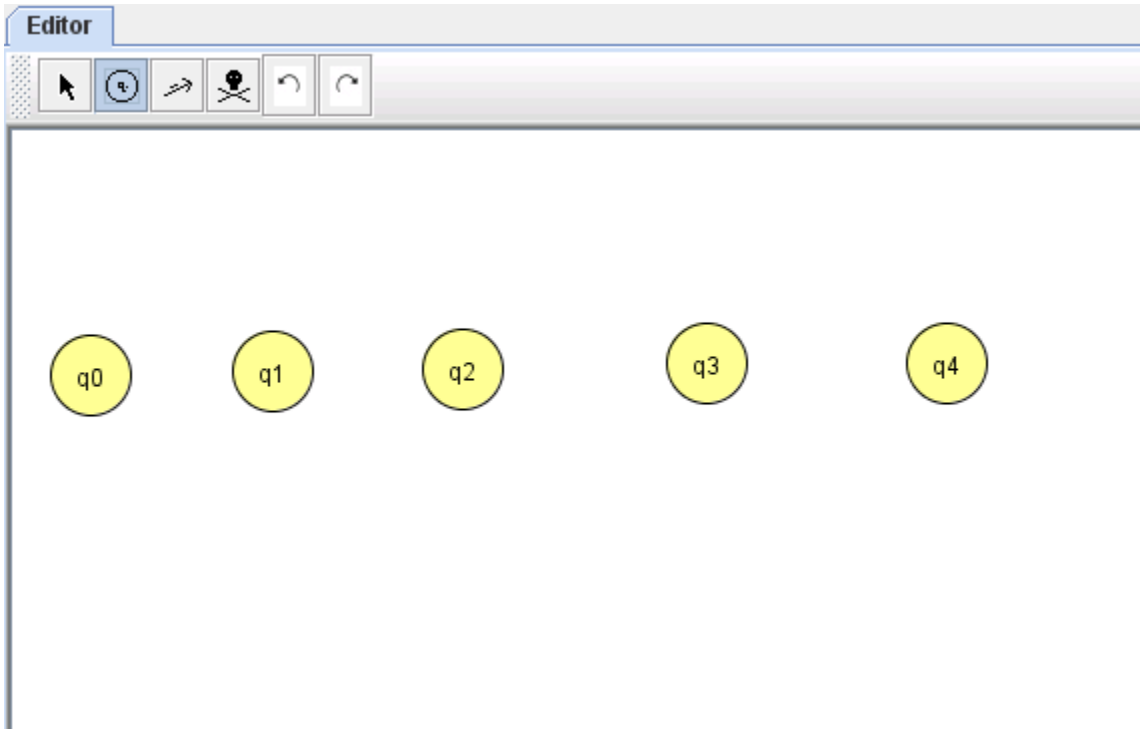


The tree can be impossible to read if there are too many derivations, so an alternate method of viewing the productions used is to select ‘Derivation Table’ from the drop down menu next to the Start and Step buttons.

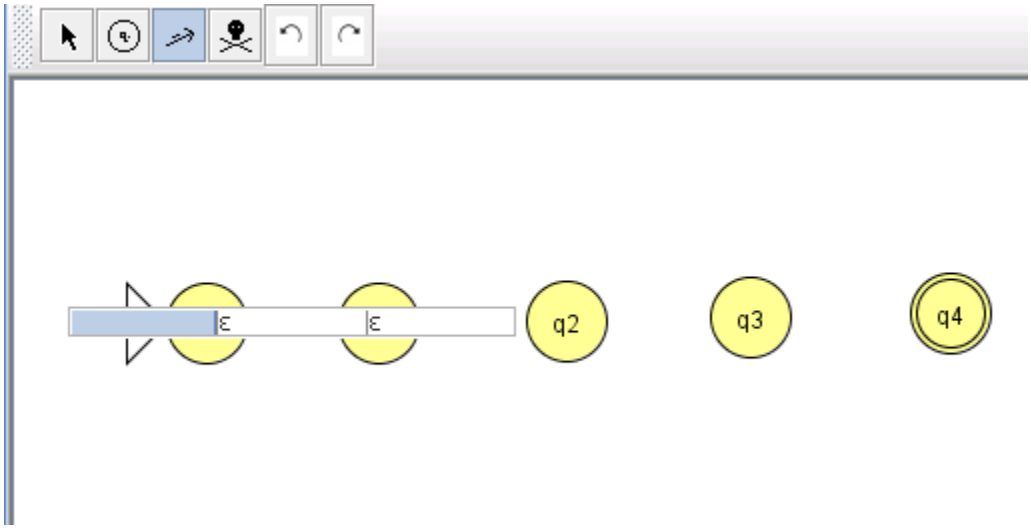
4.



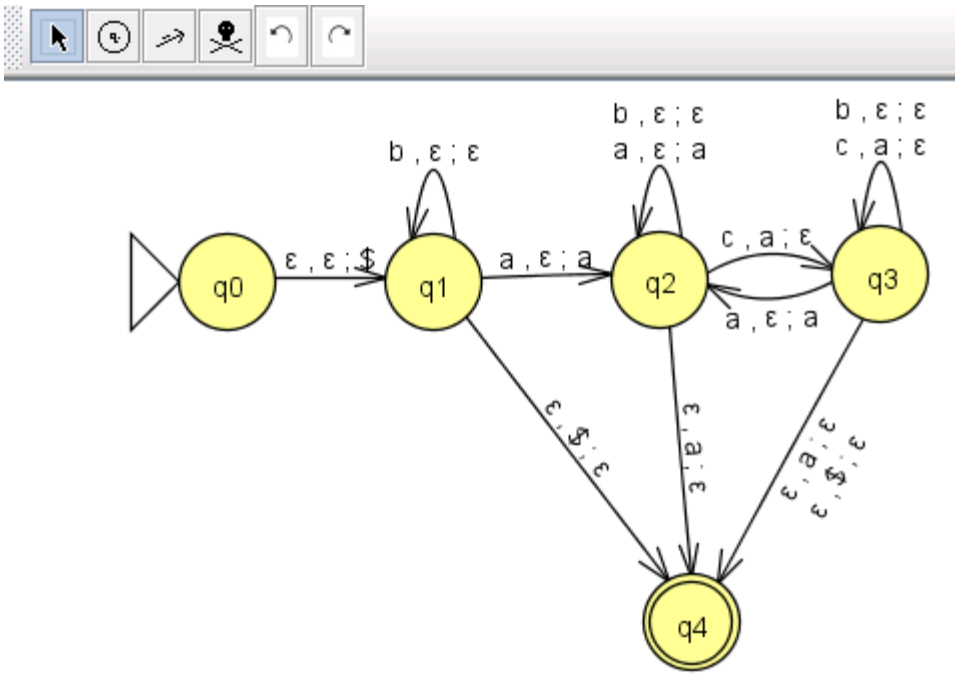
First choose Pushdown Automaton from the start menu.



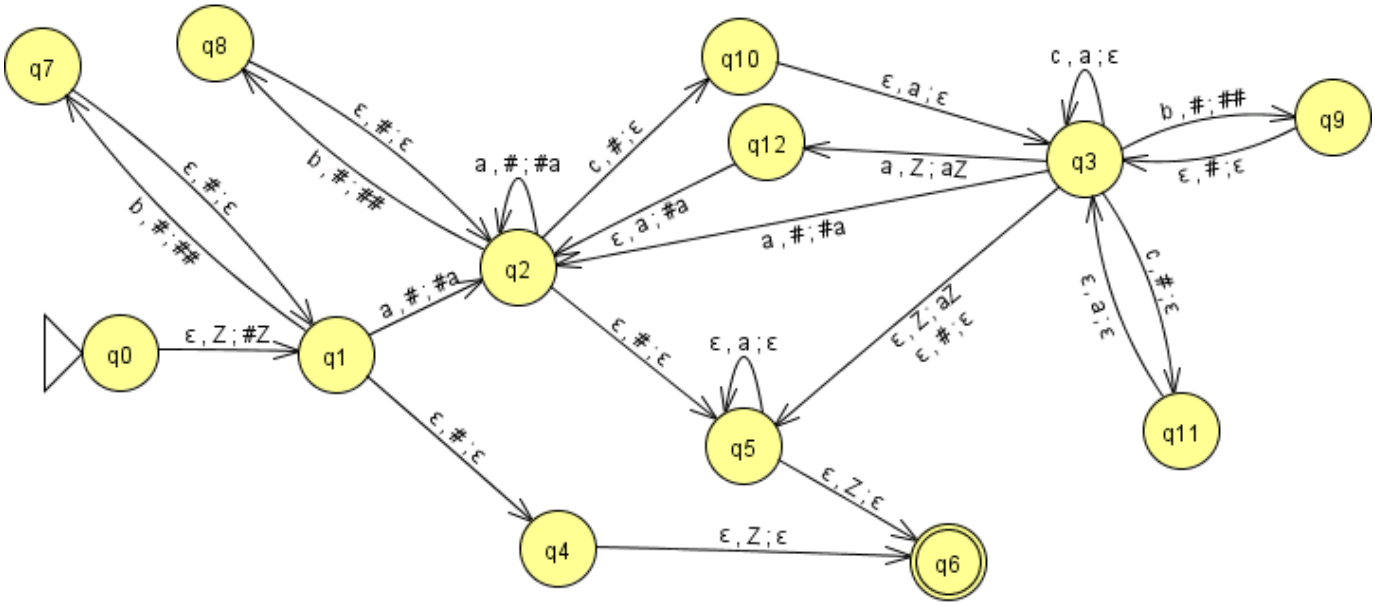
Next create the states for the automaton using the state creator.



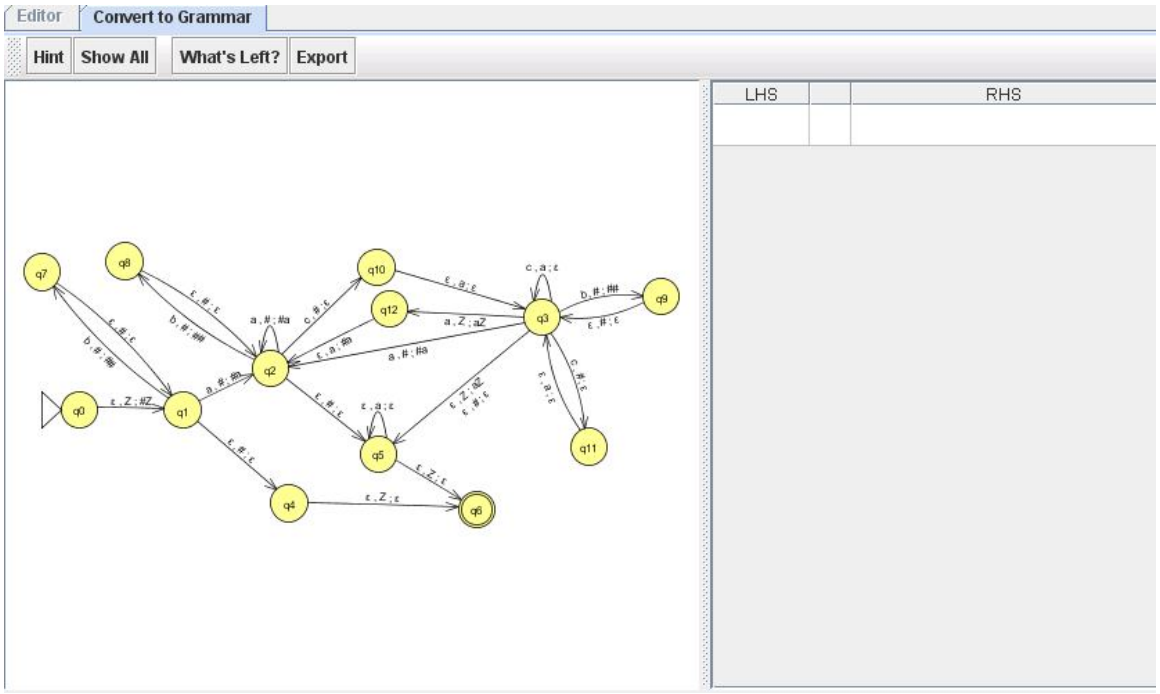
Next create the transitions using the transition creator. When inputting the parameters for the transition arrows, the first parameter represents the input terminal being read in. The second parameter is the symbol being popped off the stack. The third parameter is the symbol being pushed onto the stack.



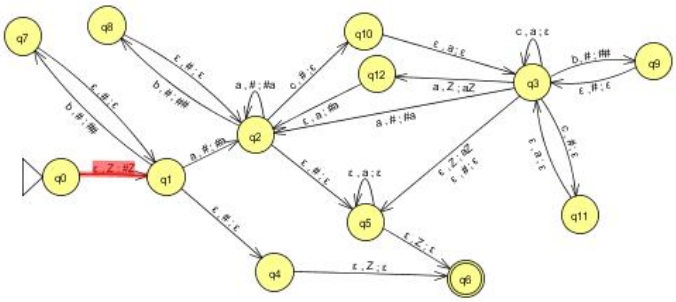
Here's what the PDA looks like after all the transition arrows have been added in.



However, JFLAP forces transition arrows to pop 1 symbol and push 0 or 2 symbols. This restriction forced me to change the PDA to the one pictured above.

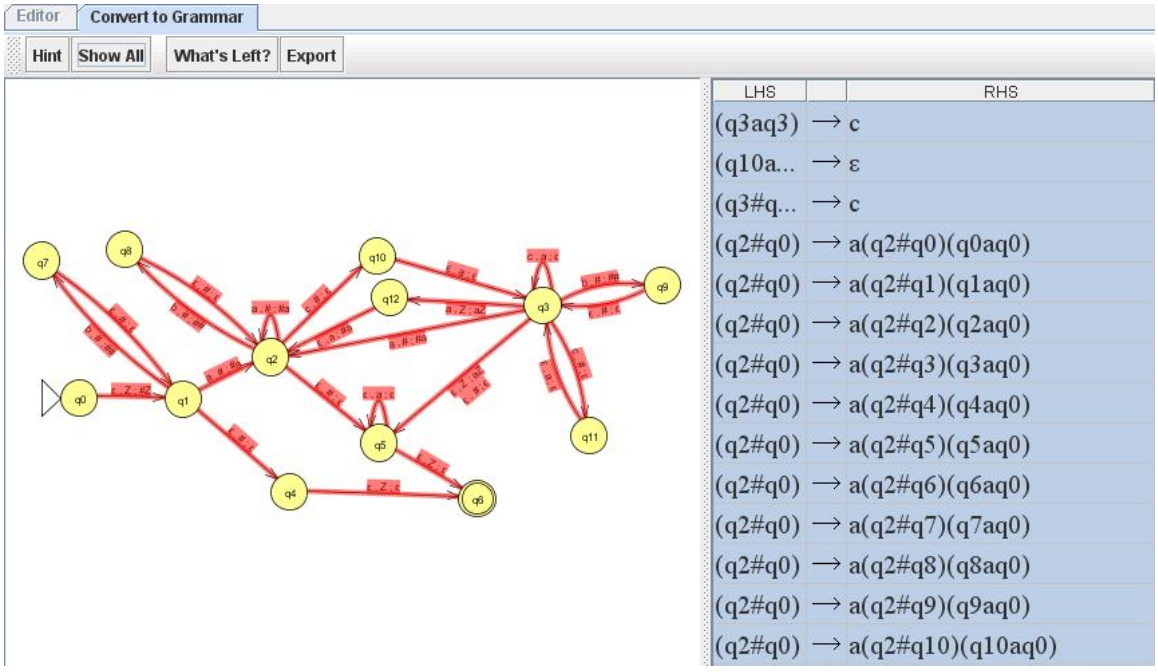


The first step after entering the PDA into JFLAP is to click on the Convert menu and choose ‘Convert to grammar’. After that you will arrive at the screen pictured above.



LHS		RHS
(q0Zq0)	→	(q1#q0)(q0Zq0)
(q0Zq0)	→	(q1#q1)(q1Zq0)
(q0Zq0)	→	(q1#q2)(q2Zq0)
(q0Zq0)	→	(q1#q3)(q3Zq0)
(q0Zq0)	→	(q1#q4)(q4Zq0)
(q0Zq0)	→	(q1#q5)(q5Zq0)
(q0Zq0)	→	(q1#q6)(q6Zq0)
(q0Zq0)	→	(q1#q7)(q7Zq0)
(q0Zq0)	→	(q1#q8)(q8Zq0)
(q0Zq0)	→	(q1#q9)(q9Zq0)
(q0Zq0)	→	(q1#q10)(q10Zq0)
(q0Zq0)	→	(q1#q11)(q11Zq0)
(q0Zq0)	→	(q1#q12)(q12Zq0)
(q0Zq1)	→	(q1#q0)(q0Zq1)

Clicking on a transition adds all possible grammar rules associated with that transition. Because the grammar rules produced by this method were obtained using a brute force method, not all of them are useful.



LHS		RHS
(q3aq3)	→	c
(q10a...	→	ε
(q3#q...	→	c
(q2#q0)	→	a(q2#q0)(q0aq0)
(q2#q0)	→	a(q2#q1)(q1aq0)
(q2#q0)	→	a(q2#q2)(q2aq0)
(q2#q0)	→	a(q2#q3)(q3aq0)
(q2#q0)	→	a(q2#q4)(q4aq0)
(q2#q0)	→	a(q2#q5)(q5aq0)
(q2#q0)	→	a(q2#q6)(q6aq0)
(q2#q0)	→	a(q2#q7)(q7aq0)
(q2#q0)	→	a(q2#q8)(q8aq0)
(q2#q0)	→	a(q2#q9)(q9aq0)
(q2#q0)	→	a(q2#q10)(q10aq0)

To display all of the rules, click the show all button.

S	⟶	LG
S	⟶	OH
S	⟶	MA
C	⟶	c
J	⟶	c
U	⟶	JK
U	⟶	FC
F	⟶	aFC
F	⟶	aJK
F	⟶	bIF
D	⟶	BN
B	⟶	aBN
B	⟶	bIB
J	⟶	bIJ
G	⟶	aUG
G	⟶	aDA
L	⟶	aFC
L	⟶	aJK
M	⟶	aBN
L	⟶	bEL
O	⟶	bEO
M	⟶	bEM
O	⟶	ε
G	⟶	NA
E	⟶	ε
I	⟶	ε
N	⟶	ε
B	⟶	ε
A	⟶	ε
H	⟶	ε
K	⟶	ε

After that click export to get rid of useless grammar rules and display the final grammar.

5.

a. We know that $f(n) = \Omega(n^2)$. So $n^2 = O(f(n))$. This tells us that n^2 grows as f or slower. So, f grows as fast as n^2 or faster. Therefore we know that the string $w = 0^{n\wedge 2}$ is in the language. We can rewrite w as $w = (0^n)^n$. Now we make $w = (0^p)^p$ so that w is both in the language and has a length greater than p. By the pumping lemma, let w = xyz. Also, $|xy| <= p$, $|y| > 0$, and xy^iz is recognized by the language for all $i >= 0$. This means $x = 0^k$ and $y = 0^j$ where $k + j <= p$ and $j > 0$. Then take i = 0. This means that $xz = (0^{p-j})^p$ is also in the language. However, since we know that $j > 0$, $p - j$ must be less than p. So, $(p - j)p < p^2$. This results in $(0^{p-j})^p$ having fewer 0s than $(0^p)^p$. However, we know from before that f(n) must grow as fast as n^2 or faster. So, this string does not take the form $w = 0^{f(n)}$. This contradicts the pumping lemma and so the language is not regular.

b. Consider the string $w = a^pbca^pbc$. This string is in the language and has a length greater than p. By the pumping lemma, let w = xyz. Also, $|xy| <= p$, $|y| > 0$, and xy^iz is recognized by the language for all $i >= 0$. This means $x = a^k$ and $y = a^j$ where $k + j <= p$ and $j > 0$. Then take i = 0. This means that $xz = a^{p-j}bca^pbc$ is also in the language. However, now there are fewer a's present on the left of the string. So, a couple of a's from the right half of the string will need to be moved to the left half to compensate for the smaller length. This causes the left half and right of the string to no longer to equivalent. So, it does not take the form ww. So, the string contradicts the pumping lemma and so the language is not regular.

c. Consider the string $w = 0^p\#0^{4p}\#0^{8p}$. This string is in the language and has a length greater than p. By the pumping lemma, let w = xyz. Also, $|xy| <= p$, $|xy| <= p$, $|y| > 0$, and xy^iz is recognized by the language for all $i >= 0$. This means $x = 0^k$ and $y = 0^j$ where $k + j <= p$ and $j > 0$. Then take $i = 0$. This means that $xz = 0^{p-j}\#0^{4p}\#0^{8p}$ is also in the language. However, since $j > 0$, we know that $p – j$ is not equal to p. So, 4p does not equal 4(p - j) and 8p does not equal 8(p - j). So, the xz does have form $0^p\#0^{4p}\#0^{8p}$. Therefore, the string contradicts the pumping lemma and so the language is not regular.

6. The limerick that will be used is:

there once was a dinosaur
who decided to give up his roar,
but when he did
he flipped his lid
and started to roar even more!

Letter	Aux	Production Table	Hash
t	t	{}	{}
h	th	{}	{th}
e	the	{}	{th, he}
r	ther	{}	{th, he, er}
e	there	{}	{th, he, er, re}
_	there_	{}	{th, he, er, re, e_}
o	there_o	{}	{th, he, er, re, e_, _o}
n	there_on	{}	{th, he, er, re, e_, _o, on}
c	there_once	{}	{th, he, er, re, e_, _o, on, nc}
e	there_once	{}	{th, he, er, re, e_, _o, on, nc, ce}
_	therAoncA	{A -> e_}	{th, he, er, rA, Ao, on, nc, cA}
w	therAoncAw	{A -> e_}	{th, he, er, rA, Ao, on, nc, cA, Aw}
a	therAoncAwa	{A -> e_}	{th, he, er, rA, Ao, on, nc, cA, Aw, wa}
s	therAoncAwas	{A -> e_}	{th, he, er, rA, Ao, on, nc, cA, Aw, wa, as}
_	therAoncAwas_	{A -> e_}	{th, he, er, rA, Ao, on, nc, cA, Aw, wa, as, s_}
a	therAoncAwas_a	{A -> e_}	{th, he, er, rA, Ao, on, nc, cA, Aw, wa, as, s_, _a}
_	therAoncAwas_a_	{A -> e_}	{th, he, er, rA, Ao, on, nc, cA, Aw, wa, as, s_, _a, a_}
d	therAoncAwas_a_d	{A -> e_}	{th, he, er, rA, Ao, on, nc, cA, Aw, wa, as, s_, _a, a_, _d}
I	therAoncAwas_a_di	{A -> e_}	{th, he, er, rA, Ao, on, nc, cA, Aw, wa, as, s_, _a, a_, _d, di}
n	therAoncAwas_a_din	{A -> e_}	{th, he, er, rA, Ao, on, nc, cA, Aw, wa, as, s_, _a, a_, _d, di, in}
o	therAoncAwas_a_dino	{A -> e_}	{th, he, er, rA, Ao, on, nc, cA, Aw, wa, as, s_, _a, a_, _d, di, in, no}
s	therAoncAwas_a_dinos	{A -> e_}	{th, he, er, rA, Ao, on, nc, cA, Aw, wa, as, s_, _a, a_, _d, di, in, no, os}
a	therAoncAwas_a_dinosa	{A -> e_}	{th, he, er, rA, Ao, on, nc, cA, Aw, wa, as, s_, _a, a_, _d, di, in, no, os, sa}
u	therAoncAwas_a_dinosau	{A -> e_}	{th, he, er, rA, Ao, on, nc, cA, Aw, wa, as, s_, _a, a_, _d, di, in, no, os, sa, au}
r	therAoncAwas_a_dinosaur	{A -> e_}	{th, he, er, rA, Ao, on, nc, cA, Aw, wa, as, s_, _a, a_, _d, di, in, no, os, sa, au, ur}
_	therAoncAwas_a_dinosaur_	{A -> e_}	{th, he, er, rA, Ao, on, nc, cA, Aw, wa, as, s_, _a, a_, _d, di, in, no, os, sa, au, ur, r_}
w	therAoncAwas_a_dinosaur_w	{A -> e_}	{th, he, er, rA, Ao, on, nc, cA, Aw, wa, as, s_, _a, a_, _d, di, in, no, os, sa, au, ur, r_, _w}
h	therAoncAwas_a_dinosaur_wh	{A -> e_}	{th, he, er, rA, Ao, on, nc, cA, Aw, wa, as, s_, _a, a_, _d, di, in, no, os, sa, au, ur, r_, _w, wh}
o	therAoncAwas_a_dinosaur_who	{A -> e_}	{th, he, er, rA, Ao, on, nc, cA, Aw, wa, as, s_, _a, a_, _d, di, in, no, os, sa, au, ur, r_, _w, wh, ho}

_	therAoncAwas_a_dinosaur_who_	{A -> e_}	{th, he, er, rA, Ao, on, nc, cA, Aw, wa, as, s_, _a, a_, _d, di, in, no, os, sa, au, ur, r_, _w, wh, ho, o_}
d	therAoncAwas_aBinosaur_whoB	{A -> e_, B -> _d}	{th, he, er, rA, Ao, on, nc, cA, Aw, wa, as, s_, _a, aB, Bi, in, no, os, sa, au, ur, r_, _w, wh, ho, oB}
e	therAoncAwas_aBinosaur_whoBe	{A -> e_, B -> _d}	{th, he, er, rA, Ao, on, nc, cA, Aw, wa, as, s_, _a, aB, Bi, in, no, os, sa, au, ur, r_, _w, wh, ho, oB, Be}
c	therAoncAwas_aBinosaur_whoBe c	{A -> e_, B -> _d}	{th, he, er, rA, Ao, on, nc, cA, Aw, wa, as, s_, _a, aB, Bi, in, no, os, sa, au, ur, r_, _w, wh, ho, oB, Be, ec}
i	therAoncAwas_aBinosaur_whoBe ci	{A -> e_, B -> _d}	{th, he, er, rA, Ao, on, nc, cA, Aw, wa, as, s_, _a, aB, Bi, in, no, os, sa, au, ur, r_, _w, wh, ho, oB, Be, ec, ci}
d	therAoncAwas_aBinosaur_whoBe cid	{A -> e_, B -> _d}	{th, he, er, rA, Ao, on, nc, cA, Aw, wa, as, s_, _a, aB, Bi, in, no, os, sa, au, ur, r_, _w, wh, ho, oB, Be, ec, ci, id}
e	therAoncAwas_aBinosaur_whoBe cide	{A -> e_, B -> _d}	{th, he, er, rA, Ao, on, nc, cA, Aw, wa, as, s_, _a, aB, Bi, in, no, os, sa, au, ur, r_, _w, wh, ho, oB, Be, ec, ci, id, de}
d	therAoncAwas_aBinosaur_whoBe cided	{A -> e_, B -> _d}	{th, he, er, rA, Ao, on, nc, cA, Aw, wa, as, s_, _a, aB, Bi, in, no, os, sa, au, ur, r_, _w, wh, ho, oB, Be, ec, ci, id, de, ed}
_	therAoncAwas_aBinosaur_whoBe cided_	{A -> e_, B -> _d}	{th, he, er, rA, Ao, on, nc, cA, Aw, wa, as, s_, _a, aB, Bi, in, no, os, sa, au, ur, r_, _w, wh, ho, oB, Be, ec, ci, id, de, ed, d_}
t	therAoncAwas_aBinosaur_whoBe cided_t	{A -> e_, B -> _d}	{th, he, er, rA, Ao, on, nc, cA, Aw, wa, as, s_, _a, aB, Bi, in, no, os, sa, au, ur, r_, _w, wh, ho, oB, Be, ec, ci, id, de, ed, d_, _t}
o	therAoncAwas_aBinosaur_whoBe cided_to	{A -> e_, B -> _d}	{th, he, er, rA, Ao, on, nc, cA, Aw, wa, as, s_, _a, aB, Bi, in, no, os, sa, au, ur, r_, _w, wh, ho, oB, Be, ec, ci, id, de, ed, d_, _t, to}
_	therAoncAwas_aBinosaur_whoBe cided_to_	{A -> e_, B -> _d}	{th, he, er, rA, Ao, on, nc, cA, Aw, wa, as, s_, _a, aB, Bi, in, no, os, sa, au, ur, r_, _w, wh, ho, oB, Be, ec, ci, id, de, ed, d_, _t, to, _o}
g	therAoncAwas_aBinosaur_whoBe cided_to_g	{A -> e_, B -> _d}	{th, he, er, rA, Ao, on, nc, cA, Aw, wa, as, s_, _a, aB, Bi, in, no, os, sa, au, ur, r_, _w, wh, ho, oB, Be, ec, ci, id, de, ed, d_, _t, to, _o, _g}
i	therAoncAwas_aBinosaur_whoBe cided_to_gi	{A -> e_, B -> _d}	{th, he, er, rA, Ao, on, nc, cA, Aw, wa, as, s_, _a, aB, Bi, in, no, os, sa, au, ur, r_, _w, wh, ho, oB, Be, ec, ci, id, de, ed, d_, _t, to, _o, _g, gi}
v	therAoncAwas_aBinosaur_whoBe cided_to_giv	{A -> e_, B -> _d}	{th, he, er, rA, Ao, on, nc, cA, Aw, wa, as, s_, _a, aB, Bi, in, no, os, sa, au, ur, r_, _w, wh, ho, oB, Be, ec, ci, id, de, ed, d_, _t, to, _o, _g, gi, iv}
e	therAoncAwas_aBinosaur_whoBe cided_to_give	{A -> e_, B -> _d}	{th, he, er, rA, Ao, on, nc, cA, Aw, wa, as, s_, _a, aB, Bi, in, no, os, sa, au, ur, r_, _w, wh, ho, oB, Be, ec, ci, id, de, ed, d_, _t, to, _o, _g, gi, iv, ve}
_	therAoncAwas_aBinosaur_whoBe cided_to_givA	{A -> e_ marked, B -> _d}	{th, he, er, rA, Ao, on, nc, cA, Aw, wa, as, s_, _a, aB, Bi, in, no, os, sa, au, ur, r_, _w, wh, ho, oB, Be, ec, ci, id, de, ed, d_, _t, to, _o, _g, gi, iv, vA}
u	therAoncAwas_aBinosaur_whoBe cided_to_givAu	{A -> e_ marked, B -> _d}	{th, he, er, rA, Ao, on, nc, cA, Aw, wa, as, s_, _a, aB, Bi, in, no, os, sa, au, ur, r_, _w, wh, ho, oB, Be, ec, ci, id, de, ed, d_, _t, to, _o, _g, gi, iv, vA, Au}
p	therAoncAwas_aBinosaur_whoBe cided_to_givAup	{A -> e_ marked, B -> _d}	{th, he, er, rA, Ao, on, nc, cA, Aw, wa, as, s_, _a, aB, Bi, in, no, os, sa, au, ur, r_, _w, wh, ho, oB, Be, ec, ci, id, de, ed, d_, _t, to, _o, _g, gi, iv, vA, Au, up}
_	therAoncAwas_aBinosaur_whoBe cided_to_givAup_	{A -> e_ marked, B -> _d}	{th, he, er, rA, Ao, on, nc, cA, Aw, wa, as, s_, _a, aB, Bi, in, no, os, sa, au, ur, r_, _w, wh, ho, oB, Be, ec, ci, id,

			de, ed, d_, _t, to, _o, _g, gi, iv, vA, Au, up, p_}
h	therAoncAwas_aBinosaur_whoBecided_to_givAup_h	{A -> e_ marked, B -> _d}	{th, he, er, rA, Ao, on, nc, cA, Aw, wa, as, s_, _a, aB, Bi, in, no, os, sa, au, ur, r_, _w, wh, ho, oB, Be, ec, ci, id, de, ed, d_, _t, to, _o, _g, gi, iv, vA, Au, up, p_, _h}
i	therAoncAwas_aBinosaur_whoBecided_to_givAup_hi	{A -> e_ marked, B -> _d}	{th, he, er, rA, Ao, on, nc, cA, Aw, wa, as, s_, _a, aB, Bi, in, no, os, sa, au, ur, r_, _w, wh, ho, oB, Be, ec, ci, id, de, ed, d_, _t, to, _o, _g, gi, iv, vA, Au, up, p_, _h, hi}
s	therAoncAwas_aBinosaur_whoBecided_to_givAup_his	{A -> e_ marked, B -> _d}	{th, he, er, rA, Ao, on, nc, cA, Aw, wa, as, s_, _a, aB, Bi, in, no, os, sa, au, ur, r_, _w, wh, ho, oB, Be, ec, ci, id, de, ed, d_, _t, to, _o, _g, gi, iv, vA, Au, up, p_, _h, hi, is}
_	therAoncAwaCaBinosaur_whoBecided_to_givAup_hiC	{A -> e_ marked, B -> _d, C -> s_}	{th, he, er, rA, Ao, on, nc, cA, Aw, wa, aC, Ca, aB, Bi, in, no, os, sa, au, ur, r_, _w, wh, ho, oB, Be, ec, ci, id, de, ed, d_, _t, to, _o, _g, gi, iv, vA, Au, up, p_, _h, hi, iC}
r	therAoncAwaCaBinosaur_whoBecided_to_givAup_hiCr	{A -> e_ marked, B -> _d, C -> s_}	{th, he, er, rA, Ao, on, nc, cA, Aw, wa, aC, Ca, aB, Bi, in, no, os, sa, au, ur, r_, _w, wh, ho, oB, Be, ec, ci, id, de, ed, d_, _t, to, _o, _g, gi, iv, vA, Au, up, p_, _h, hi, iC, Cr}
o	therAoncAwaCaBinosaur_whoBecided_to_givAup_hiCro	{A -> e_ marked, B -> _d, C -> s_}	{th, he, er, rA, Ao, on, nc, cA, Aw, wa, aC, Ca, aB, Bi, in, no, os, sa, au, ur, r_, _w, wh, ho, oB, Be, ec, ci, id, de, ed, d_, _t, to, _o, _g, gi, iv, vA, Au, up, p_, _h, hi, iC, Cr, ro}
a	therAoncAwaCaBinosaur_whoBecided_to_givAup_hiCroa	{A -> e_ marked, B -> _d, C -> s_}	{th, he, er, rA, Ao, on, nc, cA, Aw, wa, aC, Ca, aB, Bi, in, no, os, sa, au, ur, r_, _w, wh, ho, oB, Be, ec, ci, id, de, ed, d_, _t, to, _o, _g, gi, iv, vA, Au, up, p_, _h, hi, iC, Cr, ro, oa}
r	therAoncAwaCaBinosaur_whoBecided_to_givAup_hiCroar	{A -> e_ marked, B -> _d, C -> s_}	{th, he, er, rA, Ao, on, nc, cA, Aw, wa, aC, Ca, aB, Bi, in, no, os, sa, au, ur, r_, _w, wh, ho, oB, Be, ec, ci, id, de, ed, d_, _t, to, _o, _g, gi, iv, vA, Au, up, p_, _h, hi, iC, Cr, ro, oa, ar}
,	therAoncAwaCaBinosaur_whoBecided_to_givAup_hiCroar,	{A -> e_ marked, B -> _d, C -> s_}	{th, he, er, rA, Ao, on, nc, cA, Aw, wa, aC, Ca, aB, Bi, in, no, os, sa, au, ur, r_, _w, wh, ho, oB, Be, ec, ci, id, de, ed, d_, _t, to, _o, _g, gi, iv, vA, Au, up, p_, _h, hi, iC, Cr, ro, oa, ar, r\, , }
_	therAoncAwaCaBinosaur_whoBecided_to_givAup_hiCroar,_	{A -> e_ marked, B -> _d, C -> s_}	{th, he, er, rA, Ao, on, nc, cA, Aw, wa, aC, Ca, aB, Bi, in, no, os, sa, au, ur, r_, _w, wh, ho, oB, Be, ec, ci, id, de, ed, d_, _t, to, _o, _g, gi, iv, vA, Au, up, p_, _h, hi, iC, Cr, ro, oa, ar, r\, , \,_}
b	therAoncAwaCaBinosaur_whoBecided_to_givAup_hiCroar,_b	{A -> e_ marked, B -> _d, C -> s_}	{th, he, er, rA, Ao, on, nc, cA, Aw, wa, aC, Ca, aB, Bi, in, no, os, sa, au, ur, r_, _w, wh, ho, oB, Be, ec, ci, id, de, ed, d_, _t, to, _o, _g, gi, iv, vA, Au, up, p_, _h, hi, iC, Cr, ro, oa, ar, r\, , \,_, _b}
u	therAoncAwaCaBinosaur_whoBecided_to_givAup_hiCroar,_bu	{A -> e_ marked, B -> _d, C -> s_}	{th, he, er, rA, Ao, on, nc, cA, Aw, wa, aC, Ca, aB, Bi, in, no, os, sa, au, ur, r_, _w, wh, ho, oB, Be, ec, ci, id, de, ed, d_, _t, to, _o, _g, gi, iv, vA, Au, up, p_, _h, hi, iC, Cr, ro, oa, ar, r\, , \,_, _b, bu}
t	therAoncAwaCaBinosaur_whoBecided_to_givAup_hiCroar,_but	{A -> e_ marked, B -> _d, C -> s_}	{th, he, er, rA, Ao, on, nc, cA, Aw, wa, aC, Ca, aB, Bi, in, no, os, sa, au, ur, r_, _w, wh, ho, oB, Be, ec, ci, id, de, ed, d_, _t, to, _o, _g, gi, iv, vA, Au, up, p_, _h, hi, iC, Cr, ro, oa, ar, r\, , \,_, _b, bu, ut}
_	therAoncAwaCaBinosaur_whoBecided_to_givAup_hiCroar,_but_	{A -> e_ marked, B -> _d, C -> s_}	{th, he, er, rA, Ao, on, nc, cA, Aw, wa, aC, Ca, aB, Bi, in, no, os, sa, au, ur, r_, _w, wh, ho, oB, Be, ec, ci, id, de, ed, d_, _t, to, _o, _g, gi, iv, vA, Au, up, p_, _h, hi, iC, Cr, ro, oa, ar, r\, , \,_, _b, bu, ut, t_}
w	therAoncAwaCaBinosaurDhoBeci	{A -> e_ marked,	{th, he, er, rA, Ao, on, nc, cA, Aw, wa, aC, Ca, aB, Bi, in,

	ded_to_givAup_hiCroar,_butD	B -> _d, C -> s_, D -> _w}	no, os, sa, au, ur, rD, Dh, ho, oB, Be, ec, ci, id, de, ed, d_, _t, to, _o, _g, gi, iv, vA, Au, up, p_, _h, hi, iC, Cr, ro, oa, ar, r\, , \,_, _b, bu, ut, tD}
h	therAoncAwaCaBinosaurEoBecided_to_givAup_hiCroar,_butE	{A -> e_ marked, B -> _d, C -> s_, D -> _w, E -> Dh}	{th, he, er, rA, Ao, on, nc, cA, Aw, wa, aC, Ca, aB, Bi, in, no, os, sa, au, ur, rE, Eo, oB, Be, ec, ci, id, de, ed, d_, _t, to, _o, _g, gi, iv, vA, Au, up, p_, _h, hi, iC, Cr, ro, oa, ar, r\, , \,_, _b, bu, ut, tE}
e	therAoncAwaCaBinosaurEoBecided_to_givAup_hiCroar,_butEe	{A -> e_ marked, B -> _d, C -> s_, D -> _w, E -> Dh}	{th, he, er, rA, Ao, on, nc, cA, Aw, wa, aC, Ca, aB, Bi, in, no, os, sa, au, ur, rE, Eo, oB, Be, ec, ci, id, de, ed, d_, _t, to, _o, _g, gi, iv, vA, Au, up, p_, _h, hi, iC, Cr, ro, oa, ar, r\, , \,_, _b, bu, ut, tE, Ee}
n	therAoncAwaCaBinosaurEoBecided_to_givAup_hiCroar,_butEen	{A -> e_ marked, B -> _d, C -> s_, D -> _w, E -> Dh}	{th, he, er, rA, Ao, on, nc, cA, Aw, wa, aC, Ca, aB, Bi, in, no, os, sa, au, ur, rE, Eo, oB, Be, ec, ci, id, de, ed, d_, _t, to, _o, _g, gi, iv, vA, Au, up, p_, _h, hi, iC, Cr, ro, oa, ar, r\, , \,_, _b, bu, ut, tE, Ee, en}
_	therAoncAwaCaBinosaurEoBecided_to_givAup_hiCroar,_butEen_	{A -> e_ marked, B -> _d, C -> s_, D -> _w, E -> Dh}	{th, he, er, rA, Ao, on, nc, cA, Aw, wa, aC, Ca, aB, Bi, in, no, os, sa, au, ur, rE, Eo, oB, Be, ec, ci, id, de, ed, d_, _t, to, _o, _g, gi, iv, vA, Au, up, p_, _h, hi, iC, Cr, ro, oa, ar, r\, , \,_, _b, bu, ut, tE, Ee, en, n_}
h	therAoncAwaCaBinosaurEoBecided_to_givAupFiCroar,_butEenF	{A -> e_ marked, B -> _d, C -> s_, D -> _w, E -> Dh, F -> _h}	{th, he, er, rA, Ao, on, nc, cA, Aw, wa, aC, Ca, aB, Bi, in, no, os, sa, au, ur, rE, Eo, oB, Be, ec, ci, id, de, ed, d_, _t, to, _o, _g, gi, iv, vA, Au, up, pF, Fi, iC, Cr, ro, oa, ar, r\, , \,_, _b, bu, ut, tE, Ee, en, nF}
e	therAoncAwaCaBinosaurEoBecided_to_givAupFiCroar,_butEenFe	{A -> e_ marked, B -> _d, C -> s_, D -> _w, E -> Dh, F -> _h}	{th, he, er, rA, Ao, on, nc, cA, Aw, wa, aC, Ca, aB, Bi, in, no, os, sa, au, ur, rE, Eo, oB, Be, ec, ci, id, de, ed, d_, _t, to, _o, _g, gi, iv, vA, Au, up, pF, Fi, iC, Cr, ro, oa, ar, r\, , \,_, _b, bu, ut, tE, Ee, en, nF, Fe}
_	therAoncAwaCaBinosaurEoBecided_to_givAupFiCroar,_butEenFA	{A -> e_ marked, B -> _d, C -> s_, D -> _w, E -> Dh, F -> _h}	{th, he, er, rA, Ao, on, nc, cA, Aw, wa, aC, Ca, aB, Bi, in, no, os, sa, au, ur, rE, Eo, oB, Be, ec, ci, id, de, ed, d_, _t, to, _o, _g, gi, iv, vA, Au, up, pF, Fi, iC, Cr, ro, oa, ar, r\, , \,_, _b, bu, ut, tE, Ee, en, nF, FA}
d	therAoncAwaCaBinosaurEoBecided_to_givAupFiCroar,_butEenFAd	{A -> e_ marked, B -> _d, C -> s_, D -> _w, E -> Dh, F -> _h}	{th, he, er, rA, Ao, on, nc, cA, Aw, wa, aC, Ca, aB, Bi, in, no, os, sa, au, ur, rE, Eo, oB, Be, ec, ci, id, de, ed, d_, _t, to, _o, _g, gi, iv, vA, Au, up, pF, Fi, iC, Cr, ro, oa, ar, r\, , \,_, _b, bu, ut, tE, Ee, en, nF, FA, Ad}
i	therAoncAwaCaBinosaurEoBecided_to_givAupFiCroar,_butEenFAdi	{A -> e_ marked, B -> _d, C -> s_, D -> _w, E -> Dh, F -> _h}	{th, he, er, rA, Ao, on, nc, cA, Aw, wa, aC, Ca, aB, Bi, in, no, os, sa, au, ur, rE, Eo, oB, Be, ec, ci, id, de, ed, d_, _t, to, _o, _g, gi, iv, vA, Au, up, pF, Fi, iC, Cr, ro, oa, ar, r\, , \,_, _b, bu, ut, tE, Ee, en, nF, FA, Ad, di}
d	therAoncAwaCaBinosaurEoBecGed_to_givAupFiCroar,_butEenFAdG	{A -> e_ marked, B -> _d, C -> s_, D -> _w, E -> Dh, F -> _h, G -> id}	{th, he, er, rA, Ao, on, nc, cA, Aw, wa, aC, Ca, aB, Bi, in, no, os, sa, au, ur, rE, Eo, oB, Be, ec, cG, Ge, ed, d_, _t, to, _o, _g, gi, iv, vA, Au, up, pF, Fi, iC, Cr, ro, oa, ar, r\, , \,_, _b, bu, ut, tE, Ee, en, nF, FA, Ad, dG}
_	therAoncAwaCaBinosaurEoBecGed_to_givAupFiCroar,_butEenFAdG_	{A -> e_ marked, B -> _d, C -> s_, D -> _w, E -> Dh, F -> _h, G -> id}	{th, he, er, rA, Ao, on, nc, cA, Aw, wa, aC, Ca, aB, Bi, in, no, os, sa, au, ur, rE, Eo, oB, Be, ec, cG, Ge, ed, d_, _t, to, _o, _g, gi, iv, vA, Au, up, pF, Fi, iC, Cr, ro, oa, ar, r\, , \,_, _b, bu, ut, tE, Ee, en, nF, FA, Ad, dG, G_}
h	therAoncAwaCaBinosaurEoBecGed_to_givAupFiCroar,_butEenFAdGF	{A -> e_ marked, B -> _d, C -> s_, D -> _w, E -> Dh, F -> _h, G -> id}	{th, he, er, rA, Ao, on, nc, cA, Aw, wa, aC, Ca, aB, Bi, in, no, os, sa, au, ur, rE, Eo, oB, Be, ec, cG, Ge, ed, d_, _t, to, _o, _g, gi, iv, vA, Au, up, pF, Fi, iC, Cr, ro, oa, ar, r\, , \,_, _b, bu, ut, tE, Ee, en, nF, FA, Ad, dG, GF}

e	therAoncAwaCaBinosaurEoBecGed_to_givAupFiCroar,_butEenFAdGFe	{A -> e_ marked, B -> _d, C -> s_, D -> _w, E -> Dh, F -> _h, G -> id}	{th, he, er, rA, Ao, on, nc, cA, Aw, wa, aC, Ca, aB, Bi, in, no, os, sa, au, ur, rE, Eo, oB, Be, ec, cG, Ge, ed, d_, _t, to, _o, _g, gi, iv, vA, Au, up, pF, Fi, iC, Cr, ro, oa, ar, r\, , \,_, _b, bu, ut, tE, Ee, en, nF, FA, Ad, dG, GF, Fe}
_	therAoncAwaCaBinosaurEoBecGed_to_givAupFiCroar,_butEenHdGH	{A -> e_ marked, B -> _d, C -> s_, D -> _w, E -> Dh, F -> _h, G -> id, H -> FA}	{th, he, er, rA, Ao, on, nc, cA, Aw, wa, aC, Ca, aB, Bi, in, no, os, sa, au, ur, rE, Eo, oB, Be, ec, cG, Ge, ed, d_, _t, to, _o, _g, gi, iv, vA, Au, up, pF, Fi, iC, Cr, ro, oa, ar, r\, , \,_, _b, bu, ut, tE, Ee, en, nH, Hd, dG, GH}
f	therAoncAwaCaBinosaurEoBecGed_to_givAupFiCroar,_butEenHdGHf	{A -> e_ marked, B -> _d, C -> s_, D -> _w, E -> Dh, F -> _h, G -> id, H -> FA}	{th, he, er, rA, Ao, on, nc, cA, Aw, wa, aC, Ca, aB, Bi, in, no, os, sa, au, ur, rE, Eo, oB, Be, ec, cG, Ge, ed, d_, _t, to, _o, _g, gi, iv, vA, Au, up, pF, Fi, iC, Cr, ro, oa, ar, r\, , \,_, _b, bu, ut, tE, Ee, en, nH, Hd, dG, GH, Hf}
l	therAoncAwaCaBinosaurEoBecGed_to_givAupFiCroar,_butEenHdGHfl	{A -> e_ marked, B -> _d, C -> s_, D -> _w, E -> Dh, F -> _h, G -> id, H -> FA}	{th, he, er, rA, Ao, on, nc, cA, Aw, wa, aC, Ca, aB, Bi, in, no, os, sa, au, ur, rE, Eo, oB, Be, ec, cG, Ge, ed, d_, _t, to, _o, _g, gi, iv, vA, Au, up, pF, Fi, iC, Cr, ro, oa, ar, r\, , \,_, _b, bu, ut, tE, Ee, en, nH, Hd, dG, GH, Hf, fl}
i	therAoncAwaCaBinosaurEoBecGed_to_givAupFiCroar,_butEenHdGHfli	{A -> e_ marked, B -> _d, C -> s_, D -> _w, E -> Dh, F -> _h, G -> id, H -> FA}	{th, he, er, rA, Ao, on, nc, cA, Aw, wa, aC, Ca, aB, Bi, in, no, os, sa, au, ur, rE, Eo, oB, Be, ec, cG, Ge, ed, d_, _t, to, _o, _g, gi, iv, vA, Au, up, pF, Fi, iC, Cr, ro, oa, ar, r\, , \,_, _b, bu, ut, tE, Ee, en, nH, Hd, dG, GH, Hf, fl, li}
p	therAoncAwaCaBinosaurEoBecGed_to_givAupFiCroar,_butEenHdGHflip	{A -> e_ marked, B -> _d, C -> s_, D -> _w, E -> Dh, F -> _h, G -> id, H -> FA}	{th, he, er, rA, Ao, on, nc, cA, Aw, wa, aC, Ca, aB, Bi, in, no, os, sa, au, ur, rE, Eo, oB, Be, ec, cG, Ge, ed, d_, _t, to, _o, _g, gi, iv, vA, Au, up, pF, Fi, iC, Cr, ro, oa, ar, r\, , \,_, _b, bu, ut, tE, Ee, en, nH, Hd, dG, GH, Hf, fl, li, ip}
p	therAoncAwaCaBinosaurEoBecGed_to_givAupFiCroar,_butEenHdGHflipp	{A -> e_ marked, B -> _d, C -> s_, D -> _w, E -> Dh, F -> _h, G -> id, H -> FA}	{th, he, er, rA, Ao, on, nc, cA, Aw, wa, aC, Ca, aB, Bi, in, no, os, sa, au, ur, rE, Eo, oB, Be, ec, cG, Ge, ed, d_, _t, to, _o, _g, gi, iv, vA, Au, up, pF, Fi, iC, Cr, ro, oa, ar, r\, , \,_, _b, bu, ut, tE, Ee, en, nH, Hd, dG, GH, Hf, fl, li, ip, pp}
e	therAoncAwaCaBinosaurEoBecGed_to_givAupFiCroar,_butEenHdGHflippe	{A -> e_ marked, B -> _d, C -> s_, D -> _w, E -> Dh, F -> _h, G -> id, H -> FA}	{th, he, er, rA, Ao, on, nc, cA, Aw, wa, aC, Ca, aB, Bi, in, no, os, sa, au, ur, rE, Eo, oB, Be, ec, cG, Ge, ed, d_, _t, to, _o, _g, gi, iv, vA, Au, up, pF, Fi, iC, Cr, ro, oa, ar, r\, , \,_, _b, bu, ut, tE, Ee, en, nH, Hd, dG, GH, Hf, fl, li, ip, pp, pe}
d	therAoncAwaCaBinosaurEoBecGJ_to_givAupFiCroar,_butEenHdGHflippJ	{A -> e_ marked, B -> _d, C -> s_, D -> _w, E -> Dh, F -> _h, G -> id, H -> FA, J -> ed}	{th, he, er, rA, Ao, on, nc, cA, Aw, wa, aC, Ca, aB, Bi, in, no, os, sa, au, ur, rE, Eo, oB, Be, ec, cG, GJ, J_, _t, to, _o, _g, gi, iv, vA, Au, up, pF, Fi, iC, Cr, ro, oa, ar, r\, , \,_, _b, bu, ut, tE, Ee, en, nH, Hd, dG, GH, Hf, fl, li, ip, pp, pJ}

In the table above, spaces in the sentences were replaced by '_' for clarity.

The grammar outputted for this would be:

{A -> e_,

B -> _d,

C -> s_,

D -> _w,

E -> Dh,

F -> _h,

G -> id,

H -> FA,

J -> ed,

S -> therAoncAwaCaBinosaurEoBecGJ_to_givAupFiCroar,_butEenHdGHflippJ}

The final compressed string would be:

Re_R_dRs_R_wR#3hR_hRidR#5#0Red (continued on next line)

Rther#0onc#0wa#2a#1inosaur#4o#1ec#6#8_to_giv#0up#5i#2roar,_but#4en#7d#6#7flipp#8