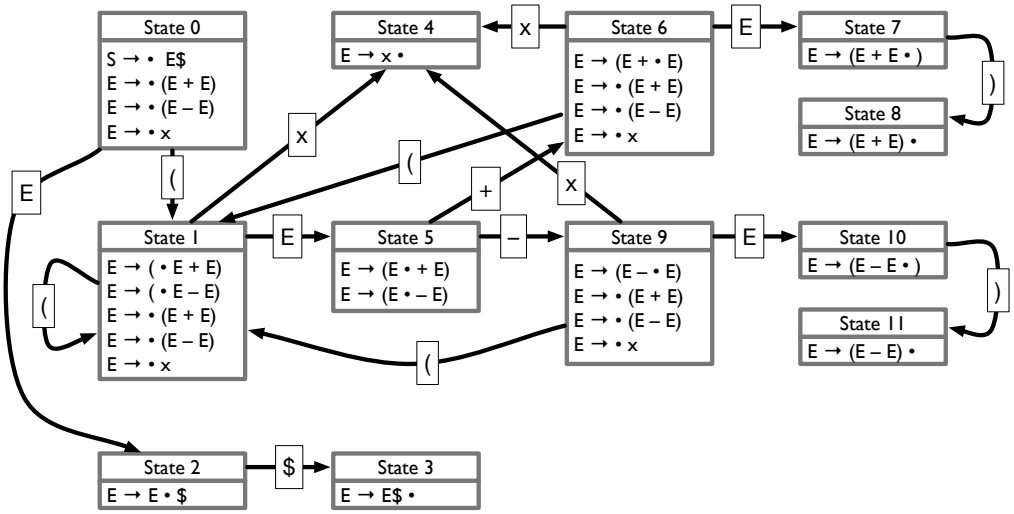
ECE 468
Problem Set 3: LR(0) parsers **(Solutions)**

1. For the following sub-problems, consider the following context-free grammar:

$$
\begin{aligned}
S &\rightarrow E\$ && (1)\\
E &\rightarrow (E+E) && (2)\\
E &\rightarrow (E-E) && (3)\\
E &\rightarrow x && (4)\\
& && (5)
\end{aligned}
$$

   (a) Build the CFSM for this grammar. Label the states in the machine as "shift", "reduce", or "accept" states.

   Answer:



   States 0, 1, 2, 5, 6, 7, 9, and 10 are shift states. States 4, 8, and 11 are reduce states. State 3 is the accept state. (Your state numbering may be different than mine)

   (b) Show the actions the parser would take while parsing $(x + (x - x))\$$. For shift actions, just say "shift". For reduce actions, say "reduce" and indicate which production you are reducing. You do not have to indicate which state you wind up in after the reduction is complete.

   Answer:

   I will indicate state numbers that are shifted to, etc. during this parse. I am also assuming the four productions are numbered 1 through 4.

State stack	Remaining input	Action
0	$(x + (x - x))$\$	Shift 1
0 1	$x + (x - x))$\$	Shift 4
0 1 4	$+(x - x))$\$	Reduce 4, goto 5
0 1 5	$+(x - x))$\$	Shift 6
0 1 5 6	$(x - x))$\$	Shift 1
0 1 5 6 1	$x - x))$\$	Shift 4
0 1 5 6 1 4	$-x))$\$	Reduce 4, goto 5
0 1 5 6 1 5	$-x))$\$	Shift 9
0 1 5 6 1 5 9	$x))$\$	Shift 4
0 1 5 6 1 5 9 4	$))$\$	Reduce 4, goto 10
0 1 5 6 1 5 9 10	$))$\$	Shift 11
0 1 5 6 1 5 9 10 11	$)$\$	Reduce 3, goto 7
0 1 5 6 7	$)$\$	Shift 8
0 1 5 6 7 8	\$	Reduce 2, goto 2
0 2	\$	Shift 3
0 2 3		Accept!

Let us decode the 5 reduce actions that we took during the parse, one by one, since they can be a little confusing.

i. **Reduce 4, goto 5**. Here, we reduced rule 4 ($E \to x$), which means that we pretend that when we saw the x, we knew all along that it was an E. So we back up one state (to state 1), which is where we were before we saw the x, and assume that instead we knew it was an E, which takes us to state 5.

ii. **Reduce 4, goto 5**. This is the same process as the previous one.

iii. **Reduce 4, goto 10**. This time, when we back up one state to before we saw the x, we find ourselves in state 9. In state 9, when we see an E, we go to state 10.

iv. **Reduce 3, goto 7**. When we find ourselves in state 11, we have recognized the right hand side of rule 3 ($E \to (E - E)$). So we back up to before we saw $(E - E)$, or *5* states (to state 6), and assume we knew all along that all of those symbols were actually an E. Thus, we wind up in state 7.

v. **Reduce 2, goto 2**. Again, we back up 5 states (to state 0) and assume we saw an E, taking us to state 2.

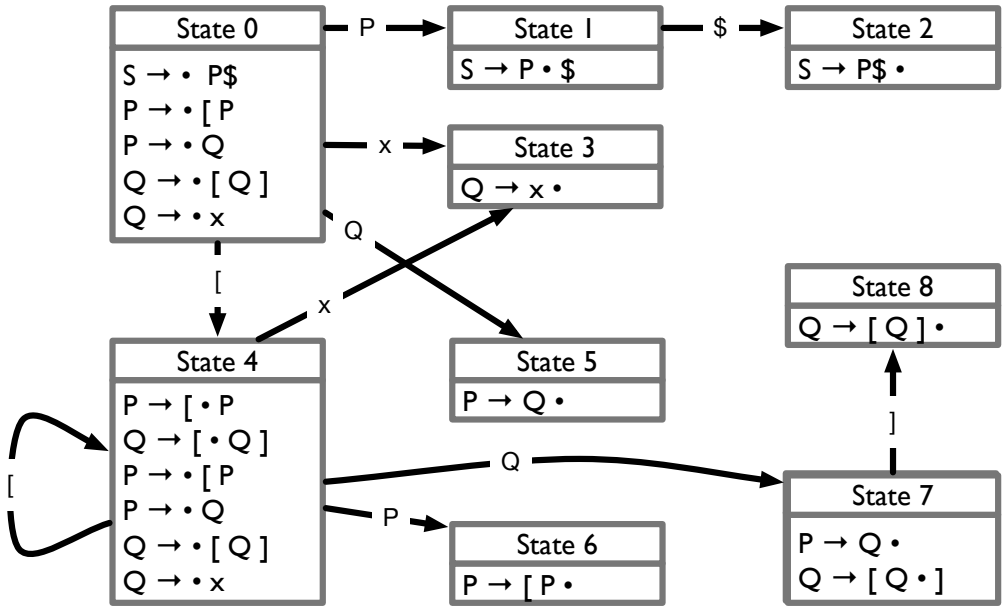
2. for the following sub-problems, consider the following grammar:

$$S \to P\$ \quad (1)$$
$$P \to [P \quad (2)$$
$$P \to Q \quad (3)$$
$$Q \to [Q] \quad (4)$$

$$Q \rightarrow x \tag{5}$$

(a) Build the CFSM for this grammar.

Answer: We went through this one in class.



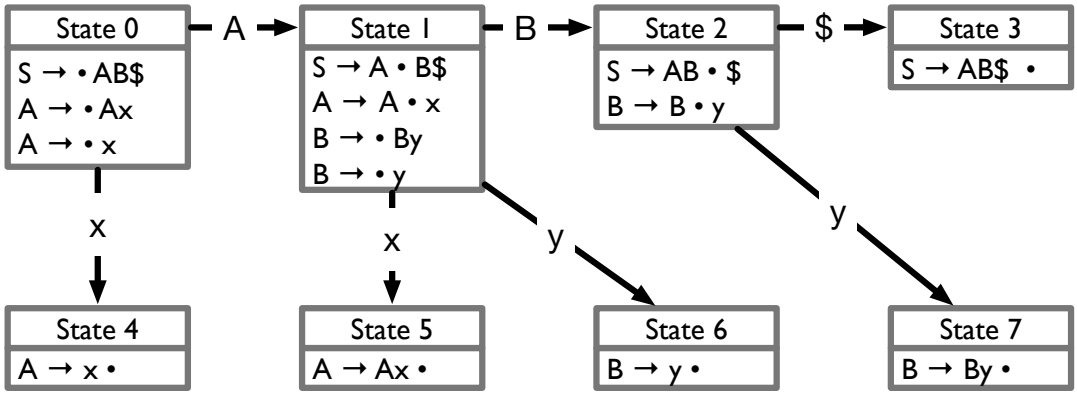
(b) Is this grammar an LR(0) grammar? Why or why not?

Answer: This is *not* an LR(0) grammar: State 7 has a shift/reduce conflict.

(c) Consider the string [x\$. If the grammar *is* LR(0), show the actions your parser would take while parsing the string. If the grammar *is not* LR(0), explain, using this string as an example, why the parser gets stuck.

Answer: After reading in the partial string [x, the parser will find itself in State 7, where it doesn't know whether to reduce (turning the [Q into a P) or shift. In state 7, it does not know whether the remainder of the string looks like]\$ (in which case it should shift) or just \$ (in which case it should reduce), and so it gets stuck.

3. For the following sub-problems, consider the CFSM below:



(a) What is the grammar that this CFSM was derived from?

Answer: This one is a little tricky. One way that you can figure this out is to realize that every rule in the grammar has to show up in at least one state in the CFSM, so if you work through the machine and see all the rules that show up, you'll figure out what the grammar is. There's an easier trick, though: In an LR(0) grammar, *every rule has one reduce state*, and *each reduce state only applies to one rule* so you can work through the reduce states (states that don't have any outgoing edges) to figure out what the grammar is:

$$
\begin{aligned}
S &\rightarrow AB\$ \\
A &\rightarrow x \\
A &\rightarrow Ax \\
B &\rightarrow y \\
B &\rightarrow By
\end{aligned}
$$

(b) Show the action the parser would take when parsing the string $xxyy\$$. For shift states, list the action as "Shift X" where X is the state that the parser moves to (for example, "Shift 5"). For Reduce states, list the action as "Reduce P, goto Y", where P is the production being reduced, and Y is the state the parser is in after completing the reduction (for example, "Reduce $A \rightarrow Ax$, goto 1").

Answer:

State stack	**Remaining input**	**Action**
0	$xxyy\$$	Shift 4
0 4	$xyy\$$	Reduce $A \to x$, goto 1
0 1	$xyy\$$	Shift 5
0 1 5	$yy\$$	Reduce $A \to Ax$, goto 1
0 1	$yy\$$	Shift 6
0 1 6	$y\$$	Reduce $B \to y$, goto 2
0 1 2	$y\$$	Shift 7
0 1 2 7	\$	Reduce $B \to By$, goto 2
0 1 2	\$	Shift 3
0 1 2 3		Accept!