# COM 2030 Exercise sheet 2: Pushdown Automata and Context-Free Languages Solutions

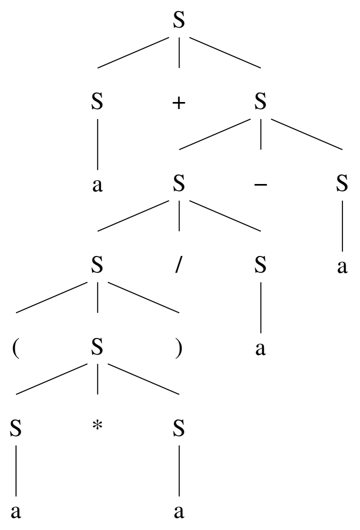
1. Consider the language of legal algebraic expressions generated by th context-free grammar $G$ with start symbol $S$, nonterminals $\{S\}$, terminals $\{a, (, ), +, -, *, /\}$ and rewrite rules:

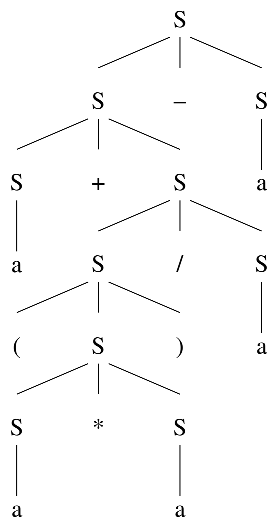
$$\begin{array}{lcl} S & \rightarrow & S+S \\ S & \rightarrow & S-S \\ S & \rightarrow & S*S \\ S & \rightarrow & S/S \\ S & \rightarrow & (S) \\ S & \rightarrow & a \end{array}$$

(This is sometimes abbreviated as: $S \rightarrow S+S \mid S-S \mid S*S \mid S/S \mid (S) \mid a$ )

Now consider the string $a+(a*a)/a-a$:

(a) draw two distinct derivation trees for this string;





(b) for one of the derivation trees in (a), provide a leftmost derivation and a rightmost derivation of the string;

Leftmost and rightmost derivations for first derivation tree:

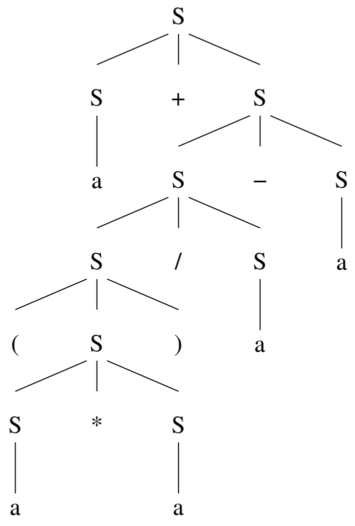
$$\begin{aligned}
S &\Rightarrow S+S & \qquad S &\Rightarrow S+S \\
&\Rightarrow a+S & &\Rightarrow S+S-S \\
&\Rightarrow a+S-S & &\Rightarrow S+S-a \\
&\Rightarrow a+S/S-S & &\Rightarrow S+S/S-a \\
&\Rightarrow a+(S)/S-S & &\Rightarrow S+S/a-a \\
&\Rightarrow a+(S*S)/S-S & &\Rightarrow S+(S)/a-a \\
&\Rightarrow a+(a*S)/S-S & &\Rightarrow S+(S*S)/a-a \\
&\Rightarrow a+(a*a)/S-S & &\Rightarrow S+(S*a)/a-a \\
&\Rightarrow a+(a*a)/a-S & &\Rightarrow S+(a*a)/a-a \\
&\Rightarrow a+(a*a)/a-a & &\Rightarrow a+(a*a)/a-a
\end{aligned}$$

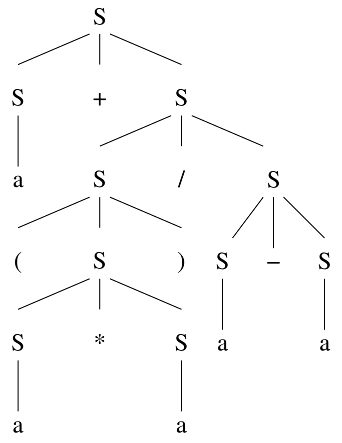
(c) how many distinct derivation trees are there for this string ? – justify your answer;

Five. There are three ambiguous binary operators in $a+(a*a)/a-a$. Picking $+$ or $-$ as the first branching point leaves an expression with two ambiguous binary operators (hence 4 derivation trees). Picking / as the first operator leaves no ambiguity. Thus, there are $(2 \times 2)+1=5$ ways of assigning an unambiguous structure to the expression.

(d) do any of the distinct derivation trees lead to different results if used to control the evaluation of the string interpreted as an algebraic expression ?

Yes. Consider these two derivation trees:

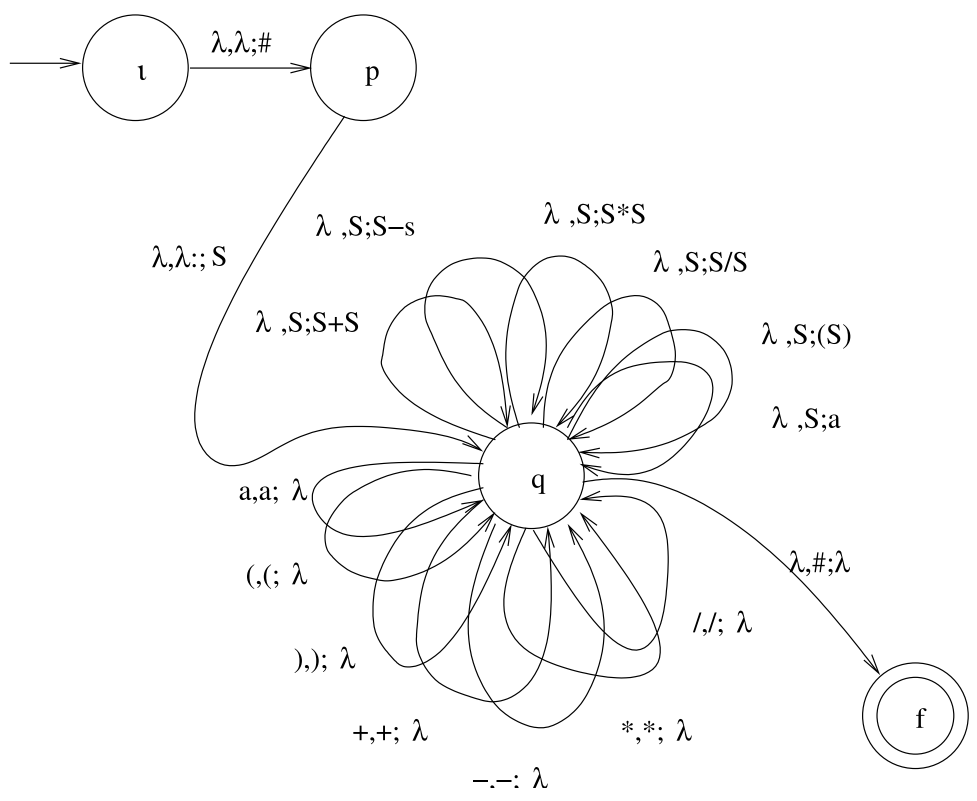




Picking, e.g. $a=5$ the first tree evaluates to $5+(((5*5)/5)-5)=5$ while the second evaluates to $5+((5*5)/(5-5))$ which is undefined.

(e) construct a transition diagram for a pushdown automaton which accepts the language generated by $G$ by leftmost derivation of its strings;

The following transition diagram accepts the language and obtains leftmost derivations.



*Note that the following points are important:*

- *pushing # then start symbol of grammar*
- *the grammar rule transitions (those above state $q$ in figure)*
- *the terminal symbol transitions (those below state $q$ in figure)*
- *popping and moving to halt state.*

(f) show a sequence of transitions the PDA in (d) could execute in order to accept the string, using the five-tuple notation for PDA transitions introduced in Lecture 5.

In analysing the string $a + (a * a)/a - a$ the machine's behaviour could be represented thus (top of stack is at the left).

| Contents of Stack | Remaining Input | Transition Executed |
|---|---|---|
| $\lambda$ | $a+(a*a)/a-a$ | $(\iota,\lambda,\lambda;p,\#)$ |
| $\#$ | $a+(a*a)/a-a$ | $(p,\lambda,\lambda;q,S)$ |
| $S\#$ | $a+(a*a)/a-a$ | $(q,\lambda,S;q,S+S)$ |
| $S+S\#$ | $a+(a*a)/a-a$ | $(q,\lambda,S;q,a)$ |
| $a+S\#$ | $a+(a*a)/a-a$ | $(q,a,a;q,\lambda)$ |
| $+S\#$ | $+(a*a)/a-a$ | $(q,+,+;q,\lambda)$ |
| $S\#$ | $(a*a)/a-a$ | $(q,\lambda,B;q,S-S)$ |
| $S-S\#$ | $(a*a)/a-a$ | $(q,\lambda,S;q,S/S)$ |
| $S/S-S\#$ | $(a*a)/a-a$ | $(q,\lambda,S;q,(S))$ |
| $(S)/S-S\#$ | $(a*a)/a-a$ | $(q,(,(;q,\lambda)$ |
| $S)/S-S\#$ | $a*a)/a-a$ | $(q,\lambda,S;q,S*S)$ |
| $S*S)/S-S\#$ | $a*a)/a-a$ | $(q,\lambda,S;q,a)$ |
| $a*S)/S-S\#$ | $a*a)/a-a$ | $(q,a,a;q,\lambda)$ |
| $*S)/S-S\#$ | $*a)/a-a$ | $(q,*,*;q,\lambda)$ |
| $S)/S-S\#$ | $a)/a-a$ | $(q,\lambda,S;q,a)$ |
| $a)/S-S\#$ | $a)/a-a$ | $(q,a,a;q,\lambda)$ |
| $)/S-S\#$ | $)/a-a$ | $(q,),);q,\lambda)$ |
| $/S-S\#$ | $/a-a$ | $(q,/,/;q,\lambda)$ |
| $S-S\#$ | $a-a$ | $(q,\lambda,S;q,a)$ |
| $a-S\#$ | $a-a$ | $(q,a,a;q,\lambda)$ |
| $-S\#$ | $-a$ | $(q,-,-;q,\lambda)$ |
| $S\#$ | $a$ | $(q,\lambda,S;q,a)$ |
| $a\#$ | $a$ | $(q,a,a;q,\lambda)$ |
| $\#$ | $\lambda$ | $(q,\lambda,\#;f,\lambda)$ |

2. Let $G$ be the grammar $(\{S,A,B\},\{a,b,c,+,(,)\},S,R\}$ where the rules in $R$ are:

$$\begin{array}{lllllllllll} S & \to & S+A & A & \to & AB & B & \to & (S) & B & \to & b \\ S & \to & A & A & \to & B & B & \to & a & B & \to & c \end{array}$$

This grammar generates the language containing simple arithmetic expressions with addition and multiplication (represented by adjacency) operations and variables $a$, $b$, and $c$.

Using the constructive technique described in the proof of Proposition 7.2 and the remarks immediately following, reduce this grammar to Chomsky normal form, showing the steps in your reduction and explaining why they preserve the equivalence of the reduced grammar.

Given the initial grammar

$$\begin{array}{llllllllllll} S & \to & S+A & A & \to & AB & B & \to & (S) & B & \to & b \\ S & \to & A & A & \to & B & B & \to & a & B & \to & c \end{array}$$

Step 1. Introduce the rules $C{\to}+$, $D{\to}($ and $E{\to})$ and produce the grammar with rules:

$$\begin{array}{llllllllllll} S & \to & SCA & A & \to & AB & B & \to & DSE & B & \to & b \\ S & \to & A & A & \to & B & B & \to & a & B & \to & c \\ C & \to & + & D & \to & ( & E & \to & ) \end{array}$$

*it is important to:*

- *demonstrate the idea of getting terminals into unary rules and out of rules with nonterminals*
- *get the right rules*

Step 2. Now add nonterminals $X$ and $Y$ and reduce right sides to length two:

$$\begin{array}{llllllllllll} S & \to & SX & A & \to & AB & B & \to & DY & B & \to & b \\ S & \to & A & A & \to & B & B & \to & a & B & \to & c \\ C & \to & + & D & \to & ( & E & \to & ) & X & \to & CA \\ Y & \to & SE \end{array}$$

*it is important to:*

- *demonstrate the idea of reducing rules with nonterminals to length two*
- *get the right rules*

Step 3. To eliminate $S{\to}A$ and $A{\to}B$ define the sets:

$$\begin{array}{lllllllll} U(S) & = & \{S\} & U(C) & = & \{C\} & U(X) & = & \{X\} \\ U(A) & = & \{S,A\} & U(D) & = & \{D\} & U(Y) & = & \{Y\} \\ U(B) & = & \{S,A,B\} & U(E) & = & \{E\} \end{array}$$

Then add the rules:

$$\begin{array}{lllllllll} S & \to & a & S & \to & b & S & \to & c \\ A & \to & a & A & \to & b & A & \to & c \end{array}$$

*it is important to:*

- *demonstrate the idea of adding rules* $Y \rightarrow x$ *for all nonterminals* $Y$ *in* $U(X)$ *where* $X \rightarrow x$
- *get the right rules*

and

$$S \rightarrow AB \qquad S \rightarrow DY \qquad A \rightarrow DY$$

to the set derived in Step 2.

*it is important to:*

- *demonstrate the idea of adding rules* $Y \rightarrow WZ$ *for all nonterminals* $Y$ *in* $U(X)$ *where* $X \rightarrow WZ$
- *get the right rules*

3. (a) Show that if the language $L$ is a deterministic context-free language then the language

$$MIN(L) = \{x \mid x \in L \text{ and no } w \in L \text{ is a proper prefix of } x\}$$

is also a deterministic context-free language. (A *proper prefix* in a string $x$ is any initial substring $s$ of $x$ such that $s \neq x$).

Suppose $L$ is a deterministic context-free language. Then there exists a DPDA $M$ which accepts $L$. A new machine $M'$ may be defined as follows: $M'$ is just like $M$ except that for each accept state $f \in F$ of $M$, if there exist any outgoing arcs from $f$, these are removed in $M'$. Note that $M'$ is deterministic.

We must prove that

i. every string accepted by $M'$ is in $MIN(L)$; and
ii. every string in $MIN(L)$ is accepted by $M'$.

i. Every string in $MIN(L)$ is accepted by $M'$.
Suppose $x$ is a string in $MIN(L)$. Then $x$ is accepted by $M$ (since $MIN(L) \subset L$ and $M$ accepts $L$). Since $M$ is deterministic, there is exactly one path through $M$ from the start state to an accept state whose arcs are labelled with the symbols in $x$ – call this path $p$. Further, since $x$ is in $MIN(L)$, no intermediate state on $p$ can be an accept state (otherwise $M$ would accept a proper prefix of $x$ and $x$ would not be in $MIN(L)$).
However, since $p$ contains no intermediate states which are accept states, it follows that $p$ will be present in $M'$ as well, since $M'$ is just like $M'$ save that outgoing arcs from accept states have been removed. Therefore $x$ is accepted by $M'$.

ii. Every string accepted by $M'$ is in $MIN(L)$.
Suppose $x$ is a string accepted by $M'$. Then there is a path $p$ through $M'$ from the start state to an accept state whose arcs are labelled with the symbols in $x$. This same path $p$ occurs in $M$, since $M'$ is the same as $M$, save for outgoing arcs from accept states in $M$ having been removed in $M'$ (i.e. all paths in $M'$ are in $M$, though the converse is not necessarily true).
Thus, $M$ accepts $x$ and so $x$ is in $L$. Note that $p$ is the only path by which $x$ is accepted by $M$, since $M$ is deterministic. Further, there can be no proper prefix of $x$ which is also accepted by $M$ since for $p$ to occur in both $M'$ and $M'$, no intermediate state on $p$ can be an accept state in $M$. So $x$ is also in $MIN(L)$.

*it is important to:*

- *show the construction*
- *argue that every string accepted by $M'$ is in $MIN(L)$*
- *argue that every string in $MIN(L)$ is accepted by $M'$.*

(b) Prove that the language $\{x^m y^n x^m y^n x^m y^n : m, n \in \mathbf{N}\}$ is not context-free (Hint: Use the Pumping Lemma).

The Pumping Lemma (Lecture 7) states that if $L$ is a context-free language with infinitely many strings then there must be a string in $L$ of the form $svuwt$, where $s$, $v$, $u$, $w$, and $t$ are substrings, at least one of $v$ and $w$ is nonempty, and $sv^kuw^kt$ is in $L$ for each $k \in \mathbf{N}^+$.

Suppose $L = \{x^m y^n x^m y^n x^m y^n : m, n \in \mathbf{N}\}$ is context-free. Since $L$ has infinitely many strings, it follows by the Pumping Lemma that there must be a string $z = svuwt \in L$ with at least one of $v$ and $w$ nonempty such that $sv^kuw^kt$ is in $L$ for each $k \in \mathbf{N}^+$.

*it is important to:*

- *assume the language $L = \{x^m y^n x^m y^n x^m y^n\}$ is context free (ie for setting up a proof by contradiction).*
- *observe that $L$ has an infinite number of strings*
- *deduce that $L$ must contain a string of the sort specified by the pumping lemma.*

Are there strings in $L$ that can be pumped as the lemma requires ? There are several possibilities:

i. one of the nonempty pumped segments $v$ and $w$ (both cannot be empty) contains only $x$'s or $y$'s. In this case a pumped string could be produced which contained a substring with more than $m$ $x$'s or $n$ $y$'s.
ii. one of the nonempty pumped segments $v$ and $w$ contains both $x$'s and $y$'s. In this case a pumped string could be produced which contained $k$ substrings of $x$'s each occurring before a substring of $y$'s, for arbitrary $k \geq 3$. But the strings in $L$ contain at most three substrings of $x$'s occurring before substrings of $y$'s.