TOC ASSIGNMENT - 2

Navdeep K

S20190010099

UnderGraduate-2, Computer Science Engineering

Indian Institute of Information Technology, Sri City, Chittor, Andhra Pradesh, India

navdeep.k19@iiits.in

***Abstract*— This document provides solutions to the TOC Assignment -2**

***Keywords*— Algorithms to convert PDA to CFG and their comparisons and complexities.**

I. Algorithm 1

We will try to slice up the machine into various components (each of which has a corresponding language), and then put them back together again using a CFG.

(P,Q,t) = {x∈Σ* | x is consumed in moving from state P to state Q in the machine, with the symbol t being taken from the top of the stack in the process}

Now, given any PDA, we construct a context-free grammar which accepts the same language as follows:

- The terminal symbols are just the input symbols of the PDA
- The non-terminal symbols are all triples of the form (P,Q,t) for each state P and Q, and each stack symbol t
- If S and F are the start and finish states respectively of the PDA, then the start symbol of the CFG is (S,F,ε)

The production rules are as follows:

- For each transition of the form (P,a,t) -> $(R,B_1,B_2...B_j)$, add all rules of the form (P,Q_j,t) -> $a.(R,Q_1,B_1).(Q_1,Q_2,B_2)....(Q_{j-1},Q_j,B_j)$

  where the Q_i can be any state in the machine

- For each transition of the form (P,a,t)->(R,ε), add all rules of the form (P,R,t)->a

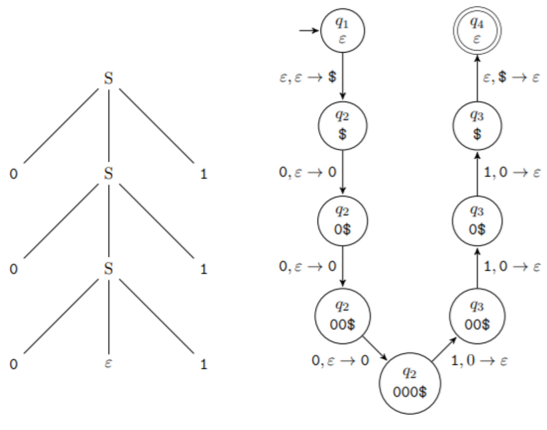
II. Algorithm 2

Prerequisites for the PDA P = (Q, Σ, Γ, δ, q0, {qaccept}):

1. Single accept state
2. Empties stack before accepting
3. Each transition either pushes one symbol to the stack, or pops one symbol off the stack, but not both or none.

We construct a CFG G that has the following rules:

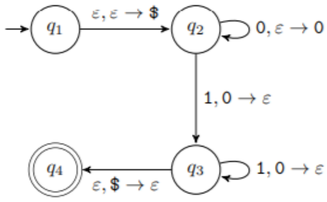
1. ∀p ∈ Q put rule App → ε
2. ∀p, q, r ∈ Q put rule Apq → AprArq
3. ∀p, r, s, q ∈ Q put rule Apq → aArsb
   if • (r, u) ∈ δ(p, a, ε) and • (q, ε) ∈ δ(s, b, u).



CFG derivation PDA run

Examples :

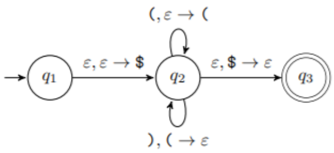
1.



The produced CFG will be:

$$
\begin{aligned}
A_{11} &\to \varepsilon \\
A_{22} &\to \varepsilon \\
A_{33} &\to \varepsilon \\
A_{44} &\to \varepsilon \\
A_{11} &\to A_{11}A_{11} \mid A_{12}A_{21} \mid A_{13}A_{31} \mid A_{14}A_{41} \\
A_{12} &\to A_{11}A_{12} \mid A_{12}A_{22} \mid A_{13}A_{32} \mid A_{14}A_{42} \\
A_{13} &\to A_{11}A_{13} \mid A_{12}A_{23} \mid A_{13}A_{33} \mid A_{14}A_{43} \\
&\dots \\
A_{42} &\to A_{41}A_{12} \mid A_{42}A_{22} \mid A_{43}A_{32} \mid A_{44}A_{42} \\
A_{43} &\to A_{41}A_{13} \mid A_{42}A_{23} \mid A_{43}A_{33} \mid A_{44}A_{43} \\
A_{44} &\to A_{41}A_{14} \mid A_{42}A_{24} \mid A_{43}A_{34} \mid A_{44}A_{44} \\
A_{23} &\to 0A_{22}1 \mid 0A_{23}1 \\
A_{14} &\to \varepsilon A_{23}\varepsilon \\
S &\to A_{14}
\end{aligned}
$$

2.



The produced CFG (after eliminating unreachable non-terminals) will be:

$$
\begin{aligned}
A_{13} &\to \varepsilon A_{22}\varepsilon \\
A_{22} &\to A_{22}A_{22} \mid \varepsilon \mid (A_{22}) \\
S &\to A_{13}
\end{aligned}
$$

III. Comparison of the Algorithms

- Time complexity of 1st Algorithm is O(mn)

  ~Where m is no. of times we get transition and n is no. of states.

- Time complexity of 2nd Algorithm is $O(n^3)$

  ~Where n is the total no. of states.

- In the first Algorithm we do conversion based on the transitions and states.
- Where as in the second Algorithm conversion is done based on only the no. of states

References

[1] https://www.cs.nuim.ie/~jpower/Courses/Previous/parsing/node36.html

[2] https://www3.nd.edu/~dchiang/teaching/theory/2016/notes/week06/week06b.pdf

[3] https://my.eng.utah.edu/~cs3100/lectures/l18/pda-notes.pdf

[4] https://courses.cs.washington.edu/courses/cse322/06sp/handouts/PDA2CFGexample.pdf