Noida Institute of Engineering and Technology, Gr. Noida

Theory of Automata and Formal Languages (ACSE0404)

Unit: IV

Push Down Automata

B. Tech
(Data Science)
4th Semester

Evaluation Scheme

NOIDA INSTITUTE OF ENGINEERING & TECHNOLOGY, GREATER NOIDA
(An Autonomous Institute)

B. TECH (IT)
EVALUATION SCHEME
SEMESTER-IV

Subject Codes	Subject Name	Periods			Evaluation Scheme				End Semester		Total	Credit
		L	T	P	CT	TA	TOTAL	PS	TE	PE		
AAS0402	Engineering Mathematics-IV	3	1	0	30	20	50		100		150	4
AASL0401	Technical Communication	2	1	0	30	20	50		100		150	3
AIT0401	Software Engineering	3	0	0	30	20	50		100		150	3
ACSE0403A	Operating Systems	3	0	0	30	20	50		100		150	3
ACSE0404	Theory of Automata and Formal Languages	3	0	0	30	20	50		100		150	3
ACSAI0402	Database Management Systems	3	1	0	30	20	50		100		150	4
AIT0451	Software Engineering Lab	0	0	2				25		25	50	1
ACSE0453A	Operating Systems Lab	0	0	2				25		25	50	1
ACSAI0452	Database Management Systems Lab	0	0	2				25		25	50	1
ACSE0459	Mini Project using Open Technology	0	0	2				50			50	1
ANC0402 / ANC0401	Environmental Science*/Cyber Security*(Non Credit)	2	0	0	30	20	50		50		100	0
	MOOCs** (For B.Tech. Hons. Degree)											
	GRAND TOTAL										**1100**	**24**

Subject Syllabus

B. TECH. SECOND YEAR			
Course Code	**ACSE0404**	**L T P**	**Credits**
Course Title	**Theory of Automata and Formal Languages**	**3 0 0**	**3**

Course objective:

To teach mathematical foundations of computation including automata theory, provide the design concepts of abstract computation model of finite automata, push down automata and turing Machine and familiarize the notions of algorithm, decidability, complexity, and computability.

Pre-requisites:

- Discrete Mathematics
- Fundamental of Computer System

Course Contents / Syllabus

UNIT-I	Basic Concepts of Formal Language and Automata Theory	8 Hours

Introduction to Theory of Computation- Alphabet, Symbol, String, Formal Languages, Grammar, Derivation and Language generation by Grammar, Chomsky Hierarchy, Finite Automata, Deterministic Finite Automaton (DFA)- Definition, Representation, Acceptability of a String and Language, Non-Deterministic Finite Automaton (NFA), Equivalence of DFA and NFA, NFA with ∈-Transition, Equivalence of NFA's with and without ∈-Transition, Finite Automata with output- Moore Machine, Mealy Machine, Equivalence of Moore and Mealy Machine, Minimization of Finite Automata, Myhill-Nerode Theorem, Simulation of DFA and NFA.

UNIT-II	Regular Language and Finite Automata	8 Hours

Regular Expressions, Transition Graph, Kleen's Theorem, Finite Automata and Regular Expression- Arden's theorem, Algebraic Method Using Arden's Theorem, Regular Grammars-Right Linear and Left Linear grammars, Conversion of FA into Regular grammar and Regular grammar into FA, Regular and Non-Regular Languages- Closure properties of Regular Languages, Pigeonhole Principle, Pumping Lemma, Application of Pumping Lemma.

Decidability- Decision properties, Finite Automata and Regular Languages, Simulation of Transition Graph and Regular language.

UNIT-III	Context Free Language and Grammar	8 Hours

Context Free Grammar (CFG)-Definition, Derivations, Languages, Derivation Trees and Ambiguity, Simplification of CFG, Normal Forms- Chomsky Normal Form (CNF), Greibach Normal Form (GNF), Pumping Lemma for CFL, Closure properties of CFL, Decision Properties of CFL

Subject Syllabus

UNIT-IV	Push Down Automata	8 Hours
Pushdown Automata- Definition, Representation, Instantaneous Description (ID), Acceptance by PDA, Nondeterministic Pushdown Automata (NPDA)- Definition, Moves, Pushdown Automata and Context Free Language, Pushdown Automata and Context Free Grammar, Two stack Pushdown Automata.		
UNIT-V	**Turing Machine and Undecidability**	**8 Hours**
Turing Machine Model, Representation of Turing Machines, Language Acceptability of Turing Machines, Techniques for Turing Machine Construction, Variations of Turing Machine, Turing Machine as Computer of Integer Functions, Universal Turing machine, Linear Bounded Automata, Church's Thesis, Recursive and Recursively Enumerable language, Closure Properties of Recursive and Recursively Enumerable Languages, Non-Recursively Enumerable and Non-Recursive Languages, Undecidability, Halting Problem, Undecidability of Halting Problem, Post's Correspondence Problem.		

Course outcome: After completion of this course students will be able to:

CO 1	Design and Simplify automata for formal languages and transform non-deterministic finite automata to deterministic finite automata.	K6
CO 2	Identify the equivalence between the regular expression and finite automata and apply closure properties of formal languages to construct finite automata for complex problems.	K3
CO 3	Define grammar for context free languages and use pumping lemma to disprove a formal language being context- free.	K3
CO 4	Design pushdown automata (PDA) for context free languages and Transform the PDA to context free grammar and vice-versa.	K6
CO 5	Construct Turing Machine for recursive and recursive enumerable languages. Identify the decidable and undecidable problems.	K6

Text books:

(1) Introduction to Automata theory, Languages and Computation, J.E. Hopcraft, R. Motwani, and Ullman. 3rd edition, Pearson Education Asia.
(2) Theory of Computer Science-Automata Language and Computation, K.L.P. Mishra, and N. Chandrasekharan, 3rd Edition, PHI.
(3) An Introduction to Formal Languages and Automata, P. Linz, 6th Edition, Jones & Bartlett Learning Publication.

Reference Books:

(1) Finite Automata and Formal Languages- A simple Approach, A. M. Padma Reddy, Cengage Learning Inc.
(2) Elements and Theory of Computation, C Papadimitrou and C. L. Lewis, PHI.
(3) Introduction to languages and the theory of computation, J Martin, 3rd Edition, Tata McGraw Hill.
(4) Introduction to The Theory of Computation, M Sipser, 3rd Edition, Cengage Learning Inc.

Branch Wise Applications

Computer Science

- Automaton is nothing but a machine which accepts the strings of a language L over an input alphabet Σ. There are four different types of Automata that are mostly used in the theory of computation (TOC).
- Finite-state machine (FSM).
- Pushdown automata (PDA).
- Linear-bounded automata (LBA).
- Turing machine (TM).

Course Objective

The primary objective of this course is to introduce students to the foundations of computability theory. The other objectives include:

- Introduce concepts in automata theory and theory of computation
- Identify different formal language classes and their relationships
- Design grammars and recognizers for different formal languages
- Prove or disprove theorems in automata theory using its properties
- Determine the decidability and intractability of computational problems

Course Outcome

Course outcome: After completion of this course students will be able to:		
CO 1	Design and Simplify automata for formal languages and transform non-deterministic finite automata to deterministic finite automata.	K6
CO 2	Identify the equivalence between the regular expression and finite automata and apply closure properties of formal languages to construct finite automata for complex problems.	K3
CO 3	Define grammar for context free languages and use pumping lemma to disprove a formal language being context- free.	K3
CO 4	Design pushdown automata (PDA) for context free languages and Transform the PDA to context free grammar and vice-versa.	K6
CO 5	Construct Turing Machine for recursive and recursive enumerable languages. Identify the decidable and undecidable problems.	K6

Program Outcomes (POs)

Engineering Graduates will be able to:

1. Engineering knowledge: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
2. Problem analysis: Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.
3. Design/development of solutions: Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.
4. Conduct investigations of complex problems: Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.

Program Outcomes (POs)

Contd..

5. Modern tool usage: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.
6. The engineer and society: Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.
7. Environment and sustainability: Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.
8. Ethics: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.

Program Outcomes (POs)

Contd..

9. Individual and team work: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.
10. Communication: Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.
11. Project management and finance: Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.
12. Life-long learning: Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

CO-PO correlation matrix

	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12
CO1	2	2	3	3	2	2	-	-	2	1	-	3
CO2	1	3	2	3	2	2	-	1	1	1	2	2
CO3	2	2	3	2	2	2	-	2	2	1	2	3
CO4	2	2	2	3	2	2	-	2	2	1	1	3
CO5	3	2	2	2	2	2	-	2	1	1	1	2
Average	2	2.2	2.4	2.6	2	2	-	1.4	1.6	1	1.2	2.6

CO-PO correlation matrix

	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12
CO1	2	2	3	3	2	2	-	-	2	1	-	3

Program Specific Outcomes

On successful completion of graduation degree the Computer Science & Engineering graduates will be able to:

PSO1:	identify, analyze real world problems and design their ethical solutions using artificial intelligence, robotics, virtual/augmented reality, data analytics, block chain technology, and cloud computing.
PSO2:	Design and develop the hardware sensor devices and related interfacing software systems for solving complex engineering problems.
PSO3:	understand inter-disciplinary computing techniques and to apply them in the design of advanced computing.
PSO4:	Conduct investigation of complex problem with the help of technical, managerial, leadership qualities, and modern engineering tools provided by industry sponsored laboratories.

CO-PSO correlation matrix

CO	PSO			
	PSO1	PSO2	PSO3	PSO4
CO1	2	2	2	2
CO2	2	2	1	1
CO3	2	2	1	1
CO4	2	2	1	1
CO5	2	2	2	2
Average	2	2	1.4	1.4

Program Educational Objectives (PEOs)

PEO1: To have an excellent scientific and engineering breadth so as to comprehend, analyze, design and solve real-life problems using state-of-the-art technologies.

PEO2: To lead a successful career in industries, to pursue higher studies or to support entrepreneurial endeavors so that engineering graduates can face the global challenges.

PEO3: To effectively bridge the gap between industry and academia through effective communication skill, professional attitude, ethical values and a desire to learn.

PEO4: To provide highly competitive environment and solidarity to students for successful professional career as engineer, scientist, entrepreneur and bureaucrats for the betterment of society

CO-PEO correlation matrix

CO	PEO			
	PEO1	PEO2	PEO3	PEO4
CO1	2	2	1	2
CO2	2	2	1	1
CO3	2	2	2	2
CO4	2	2	1	1
CO5	2	2	1	2
Average	2	2	1.2	2

Result Analysis

Subject Result:

Section A: 96.65%
Section B: 95.14%
Section C: 84.51%

End Semester Question Paper Template

Printed page: …. **Subject Code: ……..**

Roll No: | | | | | | | | | | | | |

NOIDA INSTITUTE OF ENGINEERING AND TECHNOLOGY ,GREATER NOIDA

(An Autonomous Institute Affiliated to AKTU, Lucknow)

B.Tech/B.Voc./MBA/MCA/M.Tech (Integrated)

(SEM:…..THEORY EXAMINATION(2020-2021)

Subject ……..

Time: 3Hours Max. Marks:100

General Instructions:

- All questions are compulsory. Answers should be brief and to the point.
- This Question paper consists of .....pages & ...8.....questions.
- It comprises of three Sections, A, B, and C. You are to attempt all the sections.
- Section A -Question No- 1 is objective type questions carrying 1 mark each, Question No- 2 is very short answer type carrying 2 mark each. You are expected to answer them as directed.
- Section B - Question No-3 is Long answer type -I questions with external choice carrying 6 marks each. You need to attempt any five out of seven questions given.
- Section C -Question No. 4-8 are Long answer type –II (within unit choice) questions carrying 10marks each. You need to attempt any one part *a* or *b*.
- Students are instructed to cross the blank sheets before handing over the answer sheet to the invigilator.
- No sheet should be left blank. Any written material after a blank sheet will not be evaluated/checked.

End Semester Question Paper Template

		SECTION – A		CO
1.	Attempt all parts-		[10×1=10]	
	1-a.	Question-	(1)	
	1-b.	Question-	(1)	
	1-c.	Question-	(1)	
	1-d.	Question-	(1)	
	1-e.	Question-	(1)	
	1-f.	Question-	(1)	
	1-g.	Question-	(1)	
	1-h.	Question-	(1)	
	1-i.	Question-	(1)	
	1-j.	Question-	(1)	
2.	Attempt all parts-		[5×2=10]	CO
	2-a.	Question-	(2)	
	2-b.	Question-	(2)	
	2-c.	Question-	(2)	
	2-d.	Question-	(2)	
	2-e.	Question-	(2)	

End Semester Question Paper Template

	SECTION – B			**CO**
3.	**Answer any five of the following-**		**[5×6=30]**	
	3-a.	**Question-**	**(6)**	
	3-b.	**Question-**	**(6)**	
	3-c.	**Question-**	**(6)**	
	3-d.	**Question-**	**(6)**	
	3-e.	**Question-**	**(6)**	
	3-f.	**Question-**	**(6)**	
	3-g.	**Question-**	**(6)**	
	SECTION – C			**CO**
4	**Answer any one of the following-**		**[5×10=50]**	
	4-a.	**Question-**	**(10)**	
	4-b.	**Question-**	**(10)**	
5.	**Answer any one of the following-**			
	5-a.	**Question-**	**(10)**	
	5-b.	**Question-**	**(10)**	
6.	**Answer any one of the following-**			
	6-a.	**Question-**	**(10)**	
	6-b.	**Question-**	**(10)**	
7.	**Answer any one of the following-**			
	7-a.	**Question-**	**(10)**	
	7-b.	**Question-**	**(10)**	
8.	**Answer any one of the following-**			
	8-a.	**Question-**	**(10)**	
	8-b.	**Question-**	**(10)**	

Prerequisite for the Course

- Basics operations of mathematics.
- Discrete mathematics.
- Predicate logic.

Recap

- CFG is used to generate context free languages.
- We need an automata that can accept CFL.
- Simplification of grammar includes
  - Reduction of grammar
  - Elimination of Null production
  - Elimination of Unit production
- We can normalize a CFG in CNF and GNF.

Brief Introduction about the Subject with videos

- (73) TAFL1:Theory of Automata and Formal Language| Course Overview of automata, TOC Lectures in Hindi – YouTube

- (73) TAFL2:Theory of Automata and Formal Language| Theory of Computation Tutorial Syllabus - YouTube

Contents

Topics	Duration (in Hours)
Instant Description , Acceptance by PDA	1
Non deterministic Pushdown Automata	3
Pushdown Automata and context free language	2
Pushdown Automata and context free Grammer	2
Two stack Pushdown Automata	1

Topics	Duration (in Hours)
Assignment, Tutorials and Quiz	1

Objective of Unit

Objective of the course is to make students able to:

- Instant Description , Acceptance by PDA
- .Non deterministic Pushdown Automata
- Pushdown Automata and context free language
- Pushdown Automata and context free Grammar
- Two stack Pushdown Automata

Topic Objective

Objective of the Topic

The objective of the topic is to make the student able to:

- Learn the tuples of pushdown automata
- Recognize the language accepted by PDA.
- Design determinstc and non-deterministic PDA
- Realize the expressive power of DPDA, NPDA and two stack PDA
- Convert FA to RE and vice-versa.

Objective mapping with CO

Topic mapping with Course Outcome

Topic	CO1	CO2	CO3	CO4	CO5
Finite Automata	3	-	-	-	-

Why PDA ?

- Finite automata accept the regular languages.

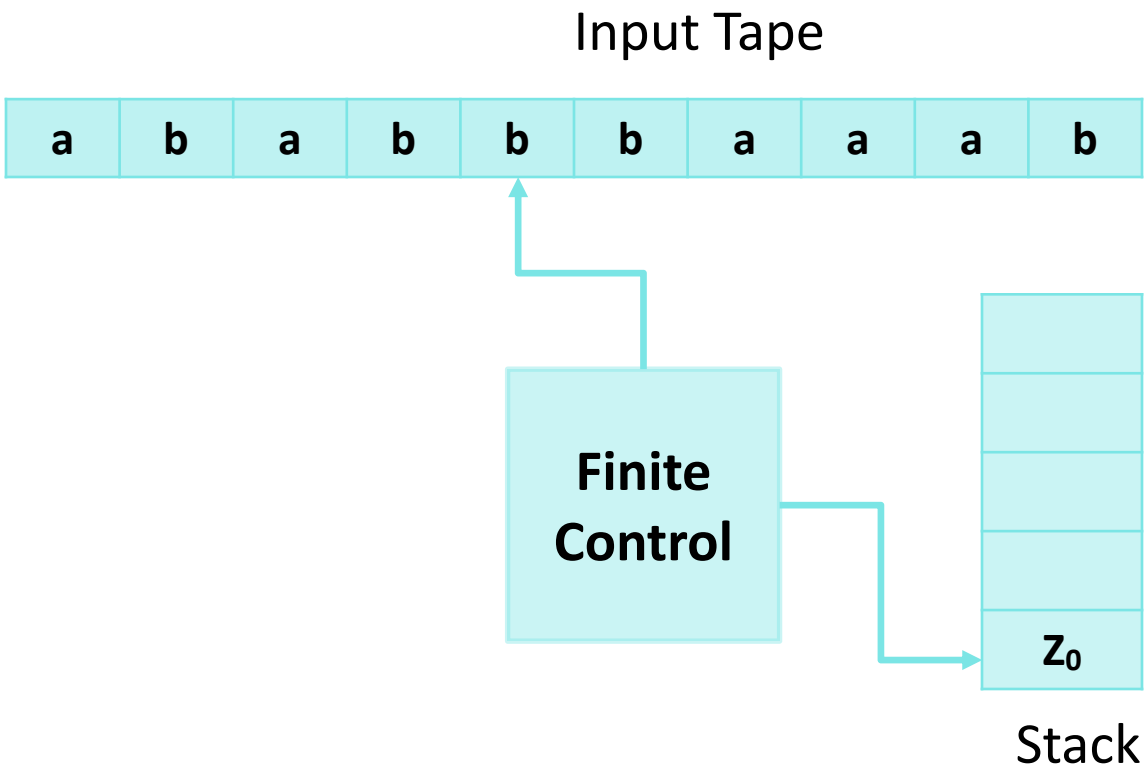
  e.g. $\{ 0^n \mid n \in \mathbb{N} \}$, $\{ 0^m 1^n \mid n \in \mathbb{N} \}$
- We may need unbounded memory to recognize context-free languages.

  e.g. $\{ 0^n 1^n \mid n \in \mathbb{N} \}$ requires unbounded counting.
- How do we build an automaton with finitely many states but unbounded memory?

  The answer is Pushdown Automata (PDA).

Why PDA ??

- Informally a pushdown automata (PDA) is an NFA + Stack.



Pushdown Automata

- A pushdown automaton (PDA) is a finite automaton equipped with a stack-based memory.
- Each transition
  - is based on the current input symbol and the top of the stack,
  - optionally pops the top of the stack, and
  - optionally pushes new symbols onto the stack.
- Initially, the stack holds a special symbol $\mathbf{Z_0}$ that indicates the bottom of the stack.

Formal Definition of PDA

PDA is a machine with 7-tuples **$M= (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$,** where

- Q is a finite set of states,
- Σ is an alphabet,
- Γ is the stack alphabet of symbols that can be pushed on to the stack,
- **$\delta : Q \times \{ \Sigma \cup \varepsilon \} \times \Gamma \rightarrow (Q \times \Gamma^*)$** is the transition function,
- $q_0 \in Q$ is the start state,
- $Z_0 \in \Gamma$ is the stack start symbol, and
- $F \subseteq Q$ is the set of accepting states.

Acceptance by PDA

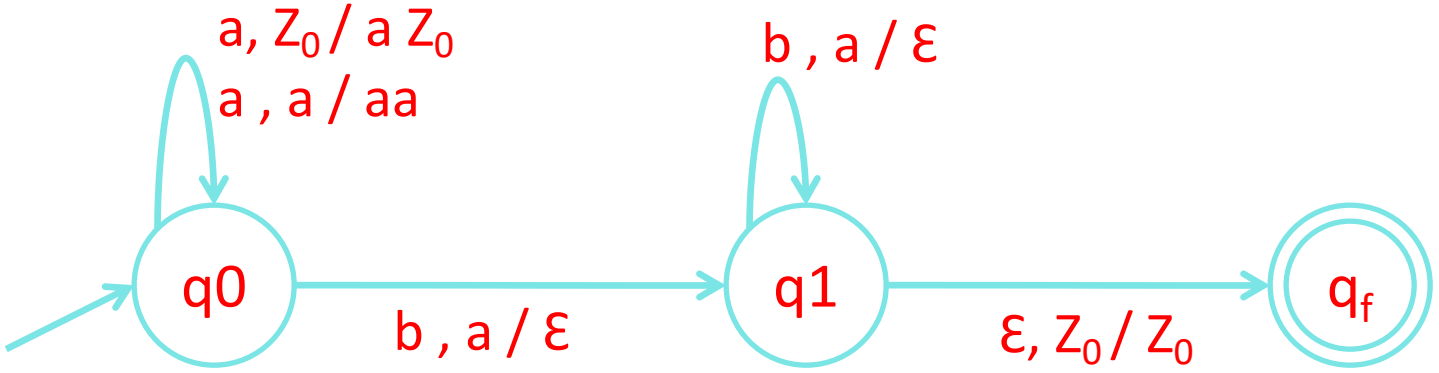
There are two ways for acceptance by PDA

- Acceptance by Final state
- Acceptance by Empty stack

PDA

PDA for L= $\{a^n b^n \mid n > 0\}$.

Acceptance by Final state



$\delta(q_0, a, Z_0) = (q_0, aZ_0)$

$\delta(q_0, a, a) = (q_0, aa)$

$\delta(q_0, b, a) = (q_1, \varepsilon)$

$\delta(q_1, b, a) = (q_1, \varepsilon)$

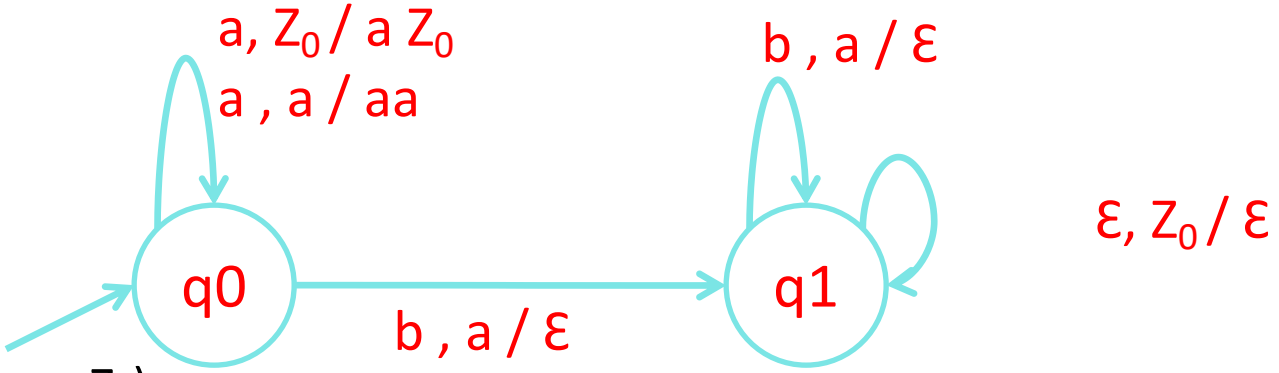
$\delta(q_1, \varepsilon, Z_0) = (q_f, Z_0)$

$M = \{\{q_0, q_1, q_f\}, \{a, b\}, \{a, Z_0\}, \delta, q_0, Z_0, q_f\}$

PDA

PDA for L= $\{a^n b^n \mid n > 0\}$.

Acceptance by Empty stack



$\delta(q_0, a, Z_0) = (q_0, aZ_0)$

$\delta(q_0, a, a) = (q_0, aa)$

$\delta(q_0, b, a) = (q_1, \varepsilon)$

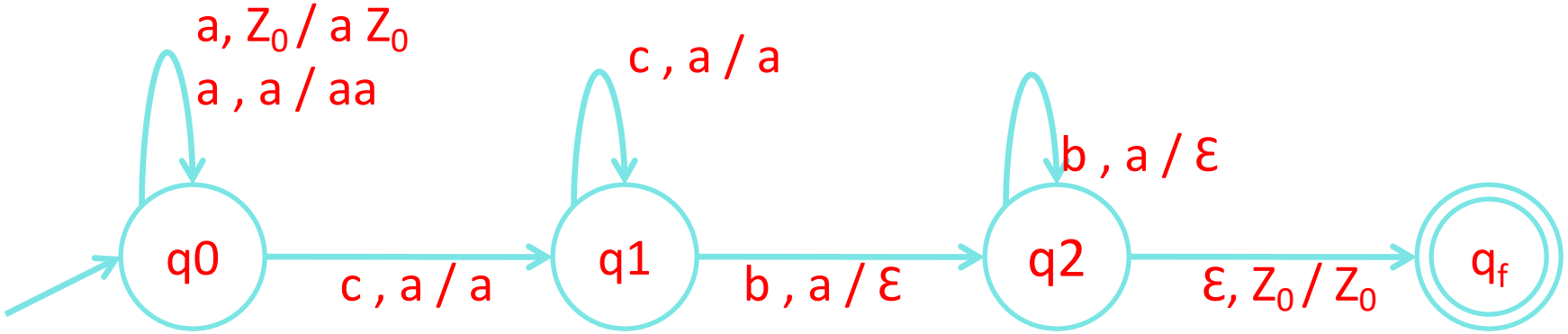
$\delta(q_1, b, a) = (q_1, \varepsilon)$

$\delta(q_1, \varepsilon, Z_0) = (q_1, \varepsilon)$

$M = \{\{q_0, q_1\}, \{a, b\}, \{a, Z_0\}, \delta, q_0, Z_0, \phi\}$

PDA

PDA for L= $\{a^n c^m b^n \mid m, n > 0\}$.



$\delta(q_0, a, Z_0) = (q_0, aZ_0)$

$\delta(q_0, a, a) = (q_0, aa)$

$\delta(q_0, c, a) = (q_1, a)$

$\delta(q_1, c, a) = (q_1, a)$

$\delta(q_1, b, a) = (q_2, \varepsilon)$

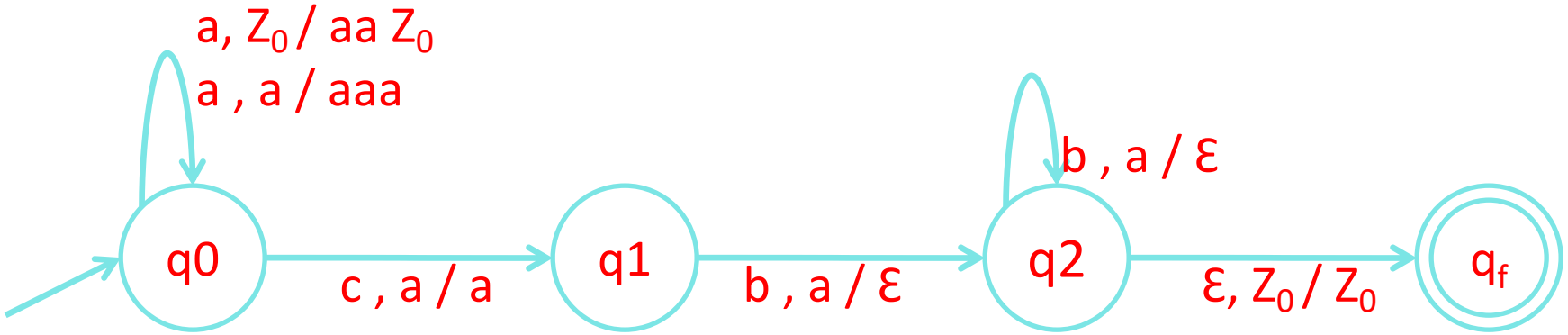
$\delta(q_2, b, a) = (q_2, \varepsilon)$

$\delta(q_2, \varepsilon, Z_0) = (q_f, Z_0)$

$M = \{\{q_0, q_1, q_2, q_f\}, \{a, b, c\}, \{a, Z_0\}, \delta, q_0, Z_0, q_f\}$

PDA

PDA for L= $\{a^n c b^{2n} \mid n > 0\}$.



$\delta(q_0, a, Z_0) = (q_0, aaZ_0)$

$\delta(q_0, a, a) = (q_0, aaa)$

$\delta(q_0, c, a) = (q_1, a)$

$\delta(q_1, b, a) = (q_2, \varepsilon)$

$\delta(q_2, b, a) = (q_2, \varepsilon)$

$\delta(q_2, \varepsilon, Z_0) = (q_f, Z_0)$

$M = \{\{q_0, q_1, q_2, q_f\}, \{a, b, c\}, \{a, Z_0\}, \delta, q_0, Z_0, q_f\}$

Example of PDA (CO4)

- In order to remember the first half of the strings that is umber of **a**'s , machine will PUSH all the **a**'s into the stack.
- For example as given in the image after reading first 5 **a**'s the stack contains 5 **a**'s.
- While reading the second half of the input string containing **b**'s, the machine POP out an **a** from the stack for every **b** as input.
- After reading 5 **b**'s, input will finish and the stack will become empty. This will indicate that the input string is of the form $a^n b^n$.

Instantaneous Description of a PDA

- For a FA, the only thing of interest about the FA is its state.
- For a PDA, we want to know its state and the entire content of its stack.
  - Often the stack is one of the most useful pieces of information, since it is not bounded in size.
- We can represent the instantaneous description (ID) of a PDA by the following triple (q,w,γ):
  - q is the state
  - w is the remaining input
  - γ is the stack contents
- By convention the top of the stack is shown at the left end of γ and the bottom at the right end.

Instantaneous Description of a PDA

- Write down the IDs or moves for input string w = “aabb” of PDA as M = ($\{q_0, q_1, q_2\}$, {a, b}, {a, b, Z_0}, δ, q0, Z_0, $\{q_2\}$), where δ is defined by following rules:

- $\delta(q_0, a, Z_0) = \{(q_0, aZ_0)\}$ Rule (1)
  $\delta(q_0, a, a) = \{(q_0, aa)\}$ Rule (2)
  $\delta(q_0, b, a) = \{(q_1, \lambda)\}$ Rule (3)
  $\delta(q_1, b, a) = \{(q_1, \lambda)\}$ Rule (4)
  $\delta(q_1, \lambda, Z_0) = \{(q_2, \lambda)\}$ Rule (5)
  $\delta(q_0, \lambda, Z_0) = \{(q_2, \lambda)\}$ Rule (6)

- Also check string w is accepted by PDA or not?

Instantaneous Description of a PDA

- **Solution:** Instantaneous Description for string w = “aabb”

- $(q_0, aabb, Z_0)$ $\vdash (q_0, abb, aZ_0)$ as per Rule (1)
  $\vdash (q_0, bb, aaZ_0)$ as per Rule (2)
  $\vdash (q_1, b, aZ_0)$ as per Rule (3)
  $\vdash (q_1, \lambda, Z_0)$ as per Rule (3)
  $\vdash (q2, \lambda, \lambda)$ as per Rule (5)

- Finally PDA reached a configuration of (q_2, λ, λ) i.e. the input tape is empty or input string w is completed, PDA stack is empty and PDA has reached a final state. So the string ‘w’ is **accepted**.

The Language of a PDA

Objective: To understand the language accepted by PDA.

A language of a PDA can be accepted by two ways:

- Through final State
- Through empty stack

It is also possible to convert between the two states:

- From Final State to empty stack
- From empty stack to final state

Acceptance by Empty Stack

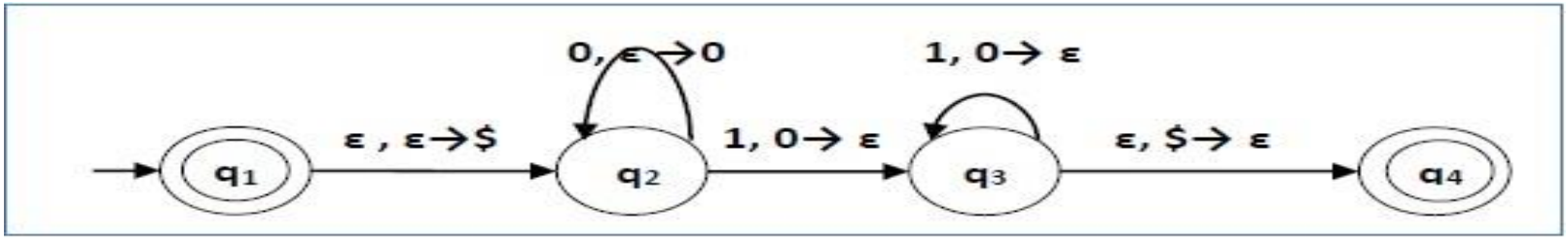
- Let the PDA , M = (Q, Σ, Γ, δ, q0, z0, Φ) then the language accepted by an empty stack is given by:

$$L(M) = \left\{ w \mid (q_0, w, z_0) \overset{*}{\underset{M}{\Rightarrow}} (q_1, \varepsilon, \varepsilon) \right\}$$

Where q1 is any stat [illegible] on application of input string w.

Example

Construct a PDA that accepts L = {0n 1n | n ≥ 0}



PDA for L= {0ⁿ 1ⁿ | n≥0}

Acceptance by Final State

- Let the PDA , M = (Q, Σ, Γ, δ, q0, z0, Φ) then the language accepted by M through a final state is given by:

$$L(M) = \left\{ w \mid (q_0, w, z_0) \overset{*}{\underset{M}{\Rightarrow}} (q_1, \varepsilon, \alpha) \right\}$$

- Where state q1 ϵ F, α, the final contents of the stack are irrelevant as a string is accepted through a final state.

Topic objective

Objective: To understand the designing of Non Deterministic PDA.

- The non-deterministic pushdown automata is very much similar to NFA.
- The CFG which accepts deterministic PDA accepts non-deterministic PDAs as well. Similarly, there are some CFGs which can be accepted only by NPDA and not by DPDA. Thus NPDA is more powerful than DPDA.

Non-deterministic Pushdown Automata

- Design PDA for Palindrome strips.

1. $\delta(q1, a, Z) = (q1, aZ)$
2. $\delta(q0, b, Z) = (q1, bZ)$
3. $\delta(q0, a, a) = (q1, aa)$
4. $\delta(q1, a, b) = (q1, ab)$
5. $\delta(q1, a, b) = (q1, ba)$
6. $\delta(q1, b, b) = (q1, bb)$

Pushing the symbols onto the stack

7. $\delta(q1, a, a) = (q2, \varepsilon)$
8. $\delta(q1, b, b) = (q2, \varepsilon)$
9. $\delta(q2, a, a) = (q2, \varepsilon)$
10. $\delta(q2, b, b) = (q2, \varepsilon)$
11. $\delta(q2, \varepsilon, Z) = (q2, \varepsilon)$

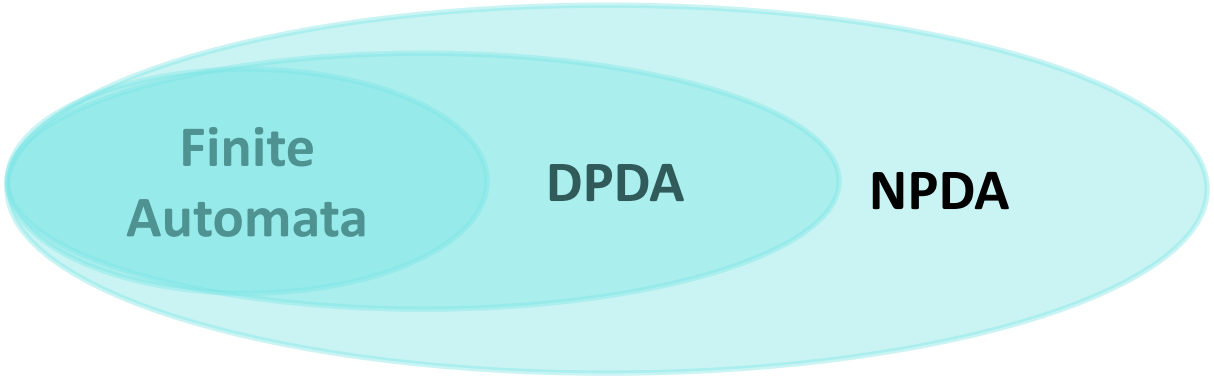
Popping the symbols on reading the same kind of symbol

Non-deterministic PDA

Power ??

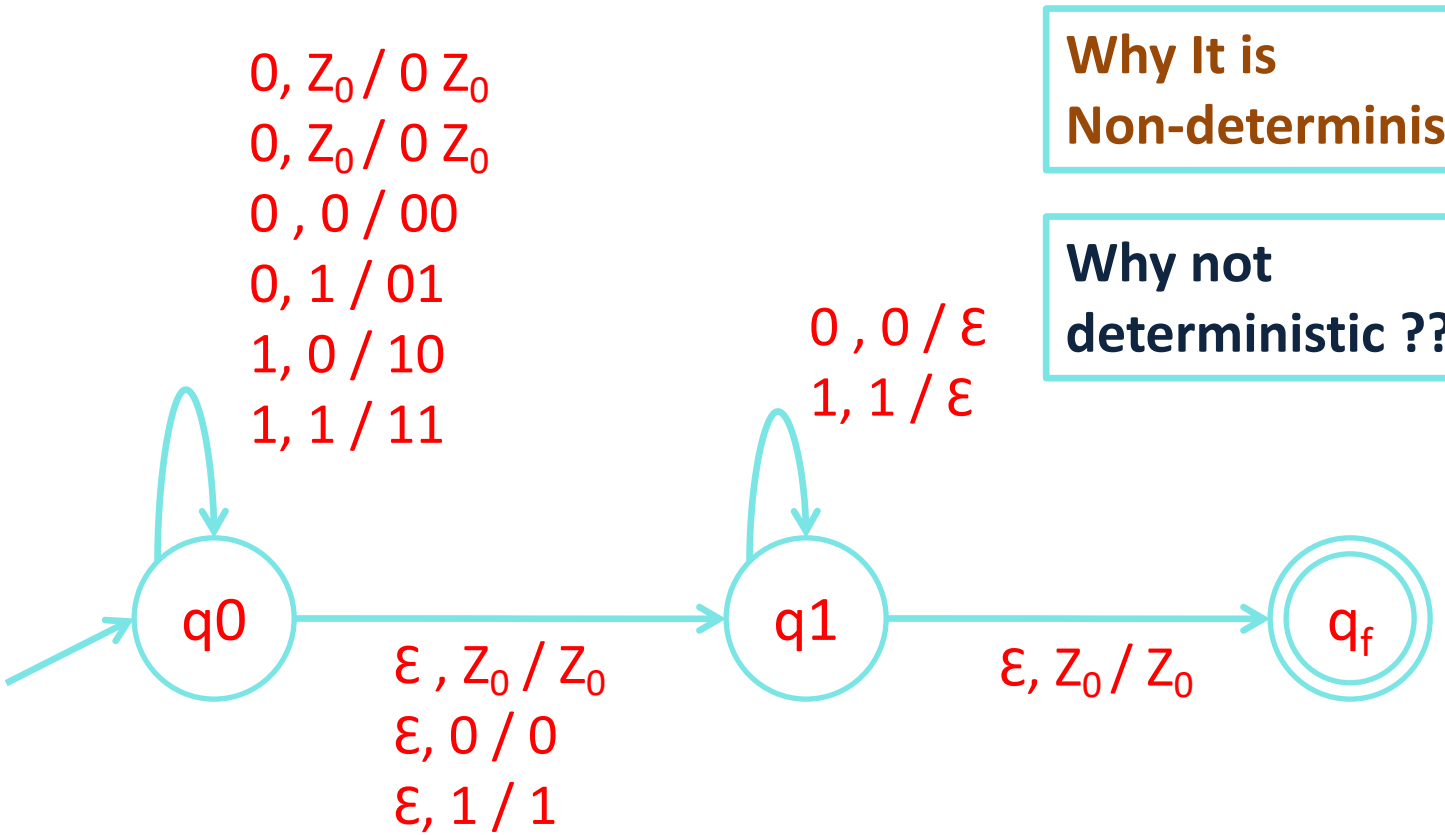
P (NFA) = P (DFA)

P (NPDA) ≥ P (DPDA)



Non-deterministic PDA

- Construct a PDA for L = {ww^R | w ∈ (0, 1) }.

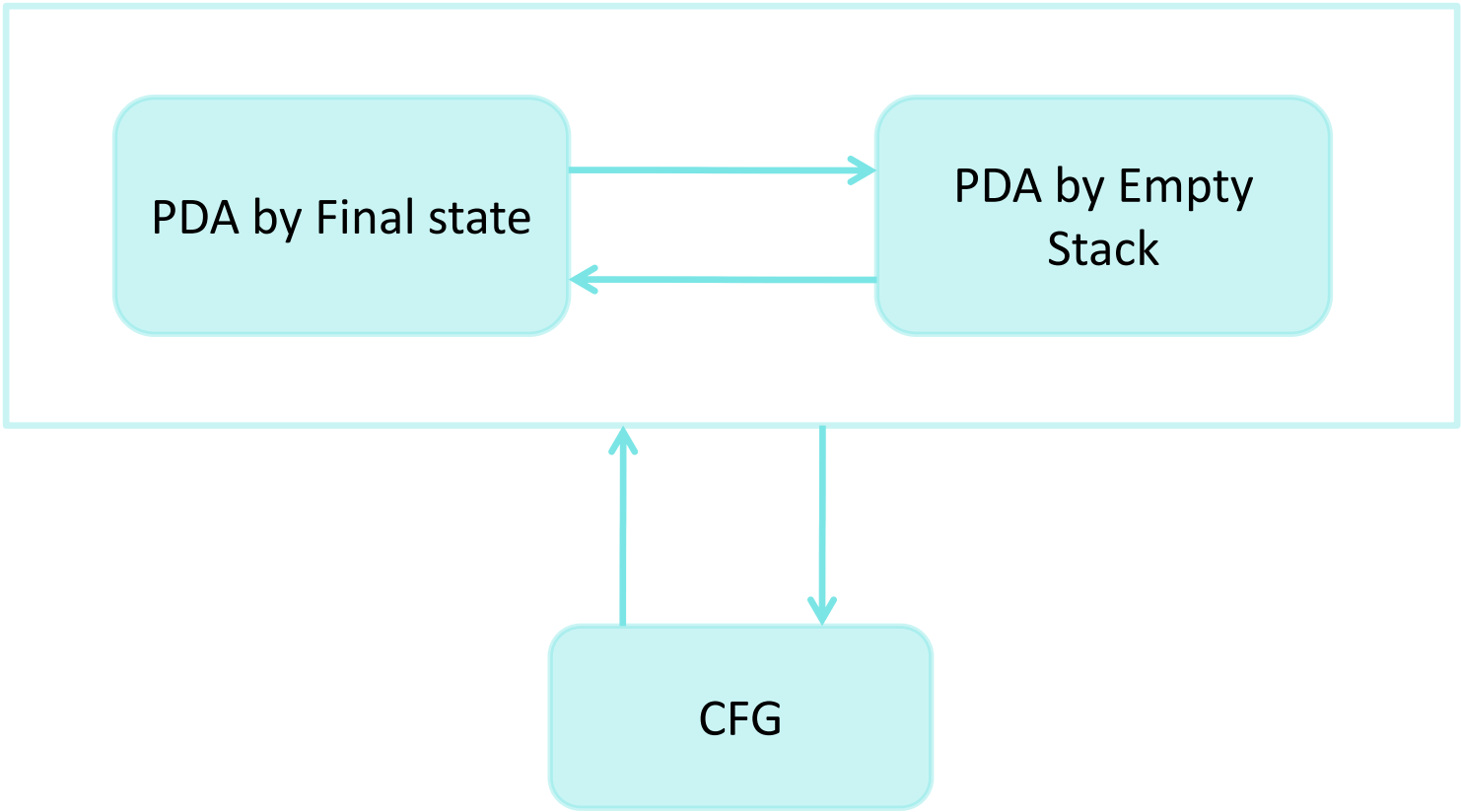


Why It is Non-deterministic ??

Why not deterministic ??

CFG = PDA ➡ CFL

Equivalence of CFG and PDA



CFG to PDA

Let G = (V, T, P, S) be a CFG, we can construct a PDA M= { {q}, T, {V $\cup$ T}, δ, q, S, ϕ } where

δ Is defined as follows:

i. For each variable A and production A $\rightarrow \alpha$, add production

$$\delta (q, \varepsilon, A) = (q, \alpha)$$

ii. For each terminal a, add production

$$\delta (q, a, a) = (q, \varepsilon)$$

CFG to PDA

Example: Construct PDA for given CFG

$S \rightarrow 0BB$

$B \rightarrow 0S \mid 1S \mid 0$

Also check whether 010^4 is being accepted by PDA or not?

Solution: Productions:

$\delta(q, \varepsilon, S) = (q, 0BB)$

$\delta(q, \varepsilon, B) = \{ (q, 0S), (q, 1S), (q, 0) \}$

$\delta(q, 0, 0) = (q, \varepsilon)$

$\delta(q, 1, 1) = (q, \varepsilon)$

Instantaneous Description

$\delta(q, 010000, S)$ |-- (q, 010000, 0BB)
|-- (q, 10000, BB)
|-- (q, 10000, 1SB)
|-- (q, 0000, SB)
|-- (q, 0000, 0BBB)
|-- (q, 000, BBB)
|-- (q, 000, 0BB)
|-- (q, 00, BB)
|$\overline{*}$ (q, 0, 0)
|-- (q, ε)

PDA to CFG

If A = (Q, Σ, Γ, δ, q0, Z0, F) is a PDA then there exists a Context-free grammar G such that L(G) =N(A).

PDA to CFG

Construction of G.

We define G = (V, T, P, S), where

$V = \{S\} \cup \{ [q, Z, q'] \mid q, q' \in Q, Z \in \Gamma\}$

The productions in P follow the rules **R1** to **R3**:

R 1: For start symbol S

$S \rightarrow [qo, Z_0, q']$ for every $q \in Q$.

R2: For $\delta(q, a, Z) = (q', \in)$

$[q, Z, q'] \rightarrow a$

R3: For $\delta(q, a, Z) = (q1, Z1\ Z2 \ldots\ldots Zm)$

$[q, Z, q'] \rightarrow a\ [q1, Z1, q2][q2, Z2, q3] \ldots [qm, Zm, q']$

where each of the states q', q2, .. qm' can be any state in Q.

PDA to CFG

Example: Construct a context-free grammar G which accepts N(A),

where A = ({qo}, {a, b}, {Z, Z_0}, δ, qo, Z_0, ϕ) and

δ is given by

δ(qo, a, Z_0) = (qo, ZZ_0)

δ(qo, a, Z) = (qo, ZZ)

δ(q0, b, Z) = (q0, Ɛ)

δ(qo, Ɛ, Z_0) = (qo, Ɛ)

PDA to CFG

Solution:

$$S \rightarrow [q0, Z_0, q0]$$

$\delta(qo, a, Z_0) = (qo, ZZ_0)$

$$[q0, Z_0, q0] \rightarrow a\ [q0, Z_0, q0]\ [q0, Z_0, q0]$$

$\delta(qo, a, Z) = (qo, ZZ)$

$$[q0, Z, q0] \rightarrow a\ [q0, Z, q0]\ [q0, Z, q0]$$

$\delta(q0, b, Z) = (q0, \varepsilon)$

$$[q0, Z_0, q0] \rightarrow b$$

$\delta(qo, \varepsilon, Z_0) = (qo, \varepsilon)$

$$[q0, Z_0, q0] \rightarrow \varepsilon$$

PDA to CFG

Question: Construct a context-free grammar G which accepts N(A),

where A = ({q_0, q1}, {a, b}, {Z, Z_0}, δ, q_0, Z_0, ϕ) and

δ is given by

$\delta(q_0, b, Z_0) = (q_0, ZZ_0)$

$\delta(q_0, \varepsilon, Z_0) = (q_0, \varepsilon)$

$\delta(q_0, b, Z) = (q_0, ZZ)$

$\delta(q_0, a, Z) = (q_1, Z)$

$\delta(q_1, b, Z) = (q_1, \varepsilon)$

$\delta(q_1, a, Z_0) = (q_1, Z_0)$

Construction of PDA from CFG: Example

Q: Find a PDA for the given grammar:

S → **0S1|00|11**

Solution:

The equivalent PDA, M is given by:

M = ({q},{0,1}, {0,1,S}, δ, q, S, Φ), where δ is given by:

δ(q, ε, S) = {(q,0S1), (q,00), (q,11)}

δ(q, 0, 0) = {(q, ε)}

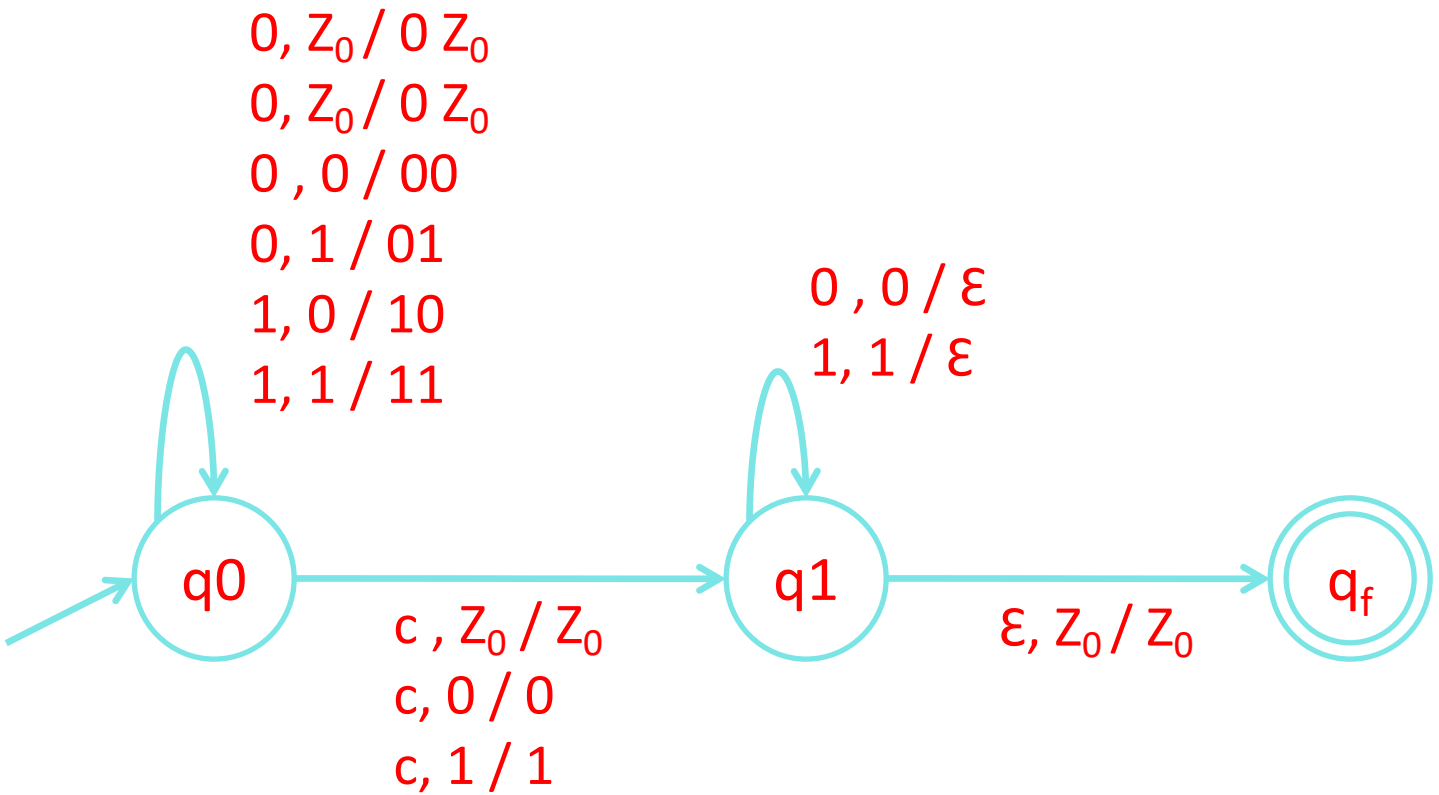
δ(q, 1, 1) = {(q, ε)}

For CFG to PDA follow the following link:

https://www.youtube.com/watch?v=J_6JJH34XyQ

Deterministic PDA

- Construct a PDA for L = {wcw^R | w ∈ (0, 1) }.



Difference between DPDA and NPDA

DPDA	NPDA
In NPDA, there may exits exactly one transition for each input symbol.	In PDA, there may exits more than one transition for each input symbol
Table contains single entities	Table may contains multiple defined entities.
Less powerful than NPDA.	More powerful than DPDA.
Ex: PDA for L= $\{wcw^R \mid w \in (0, 1)^*\}$	**Ex:** PDA for L= $\{ww^R \mid w \in (0, 1)^*\}$

PDA Examples with Solutions

Q: Let L={$a^m b^n$ | n<m} . Construct a PDF for following Language.

Solution:

Sequence of a's should be pushed onto the stack in state q0.

δ(q0,a,z0) = (q0, az0)

δ(q0,a,a) = (q0, aa)

An a should be popped for every b as input till the end of input.

δ(q0,b,a) = (q1,ε)

δ(q1,b,a) = (q1,ε)

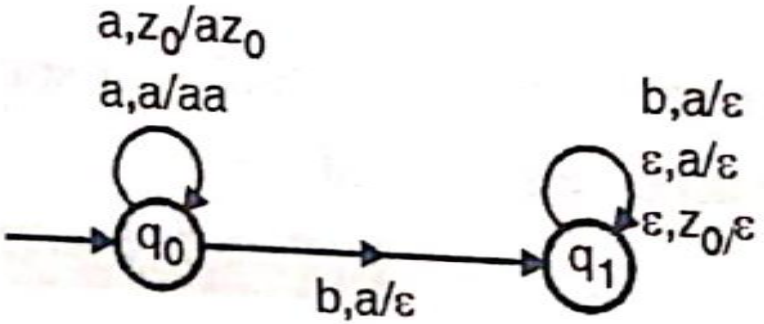
PDA Examples with Solutions

Additional m-n a's should be popped from the stack.

$\delta(q1, \varepsilon, a) = (q1, \varepsilon)$

Finally the symbol z0 should be popped out to make the state empty.

$\delta(q1, \varepsilon, z0) = (q1, \varepsilon)$



PDA Examples with Solutions

Q: Design a PDA for accepting the set of all strings over an alphabet {a,b} with an equal number of a's and b's.

Solution:

Stack will be used to store excess of a's over b's or excess of b's over a's out of input seen so far.

PDA Examples with Solutions

1. Stack contains z0 (topmost symbols) and the input is a:

δ(q0, a, z0) = (qo, az0)

2. Stack contains z0 and the input is b:

δ(q0, b, z0) = (qo, bz0)

3. Stack contains a and the input is a.

δ(q0, a, a) = (qo, aa)

4. Stack contains a and the input is b

δ(q0, a, b) = (qo, ε)

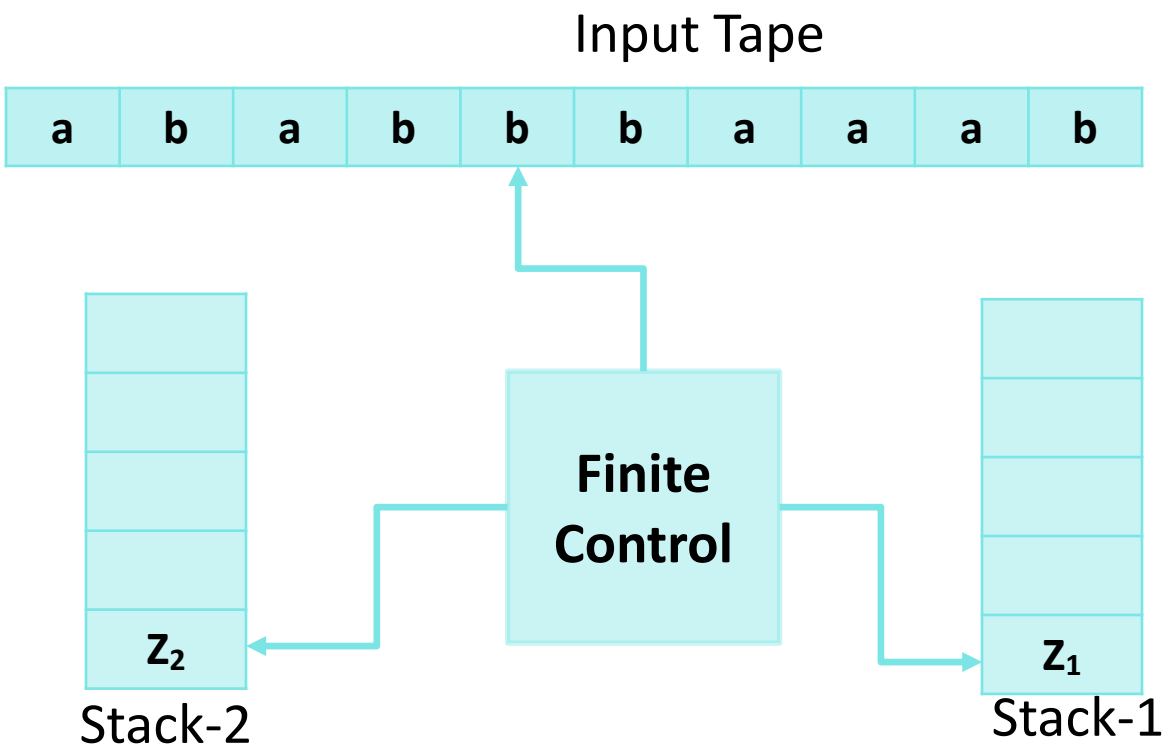
5. Stack contains b and the input is a

δ(q0, b, a) = (qo, ε)

6. Stack contains b and the input is b

δ(q0, b, b) = (qo, bb)

Two stack PDA



Two stack PDA

- A two stack PDA is deterministic.
- Two stack PDA is more powerful than PDA.
- Its power is equal to that of A Turing machine.
- It can Perform PUSH / POP operation on either or both of the stacks.

Two stack PDA

Formally, a two stack PDA is defined by the 9-tuples

M= {Q, Σ, Γ1, Γ2, δ, q0, Z1, Z2, F}, where

- Q is a finite set of states,
- Σ is an alphabet,
- Γ1, Γ2 is the stack alphabet
- **$\delta : Q \times \{ \Sigma \cup \in \} \times \Gamma 1 \; X \; \Gamma 2, \rightarrow (Q \times \Gamma 1^* X \; \Gamma 2^*)$**
- q0 ∈ Q is the start state,
- Z1 ∈ Γ1, is the stack start symbol of PDS 1, and
- Z2 ∈ Γ2, is the stack start symbol of PDS 2
- F ⊆ Q is the set of accepting states.

Two stack PDA

Example: Construct a two stack PDA that accepts L = { $a^n b^n c^n$ | n>0}.

Solution:

δ(q0, a, Z0, Z0) = (q0, aZ0, aZ0)

δ(q0, a, a, a) = (q0, aa, aa)

δ(q0, b, a, a) = (q1, Ɛ, a)

δ(q1, b, a, a) = (q1, Ɛ, a)

δ(q1, c, Z0, a) = (q2, Z0, Ɛ)

δ(q2, c, Z0, a) = (q2, Z0, Ɛ)

δ(q2, Ɛ, Z0, Z0) = (q0, Ɛ, Ɛ)

Recap

We learnt:

- Nondeterministic Pushdown Automata (NPDA),
- Deterministic Pushdown Automata(DPDA)
- Pushdown Automata for Context Free Languages,
- Context Free grammars for Pushdown Automata,
- Two stack Pushdown Automata,

Video Links

- Self Made Video Link:
  - https://www.youtube.com/playlist?list=PLKhyqvN-JcP5AF4UzRz9wxFVBm63igm7y
- NPTEL Video Links

  https://youtu.be/7ZbDEfnYwAo

  https://youtu.be/_3Fi5jBhinE

  https://youtu.be/q82IcIjS1y8

Daily Quiz

- Which of the following automata takes stack as auxiliary storage?
  A. Finite automata
  B. **Push down automata**
  C. Turing machine
  D. All of the mentioned

- A push down automata can be represented as:
  PDA= ε-NFA +[stack] State true or false:
  A**. true**
  B. false

- A PDA machine configuration (p, w, y) can be correctly represented as:
  A**. (current state, unprocessed input, stack content)**
  B. (unprocessed input, stack content, current state)
  C. (current state, stack content, unprocessed input)
  D. none of the mentioned

Daily Quiz

- Push down automata accepts _____ languages.
  A. Type 3
  B. **Type 2**
  C. Type 1
  D. Type 0

- A string is accepted by a PDA when
  A. Stack is empty
  B. Acceptance state
  C. **Both A and B.**
  D. None of the mentioned

Daily Quiz

- If the PDA does not stop on an accepting state and the stack is not empty, the string is:
  A. **rejected**
  B. goes into loop forever
  C. both A and B.
  D. none of the mentioned

- A language accepted by Deterministic Push down automata is closed under which of the following?
  A. **Complement**
  B. Union
  C. Both (A. and (B.
  D. None of the mentioned

Daily Quiz

- A language is accepted by a push down automata if it is:
  A. regular
  B. context free
  C. **both (A. and (B.**
  D. none of the mentioned

- What you can say about following statement.
  Statement: The operations of PDA never work on elements, other than the top.
  A. **True**
  B. False
  C. may be
  D. can't say

Weekly Assignment

1. Construct DPDA and NPDA for L= $a^n b^n$, n>=1. **[C04]**
2. Design a DPDA by using acceptance by Final state for L= wCw^R, w Є {a/b)$^+$. **[C04]**
3. Design DPDA for L= $a^n b^m c^m d^n$, m, n>=1 **[C04]**
4. Design DPDA for L= $a^n b^m a^n$, m, n>=0 **[C04]**
5. NPDA for ww^R, w Є {0/1)*, using acceptance by Null store. **[C04]**
6. Construct DPDA to accept L={x: x Є (a/b)*where $n_a(x) > n_b(x)$} **[C04]**
7. Construct PDA to accept L={w: w Є (a/b)*where $n_a(w) = n_b(w)$} **[C04]**
8. Design a DPDA to accept L= { $a^m b^m c^n$: m, n>=1} **[C04]**
9. Design a PDA which can detect both even and odd palindrome. **[C04]**
10. Convert the following CFG into PDA: S→0S1/A, A→1A0/S/ε **[C04]**

Weekly Assignment

11. Design PDA for the following languages **[C04]**

a. $L=a^m b^n$ i) $n>m$ ii) $m>n$

b. $L=\{ a^n b^n c^m d^m, m,n>=0\}$

c. $L=\{a^{2n}bc$ where $n>=0\}$

d. $L=\{(ab)^n$ where $n>=1\}$

e. $L=\{a^n bc^{2n}$ where $n>=1\}$

f. $L=(a^{n+1}b^{2n}$ where $n>=0\}$

g. $L=\{a^n b^m c^{m-n}$ where $n>=0$, $m>=0$, $m>=n\}$

h. $L=\{a^n b^m a^n$ where $m,n>=1\}U$ $\{a^n c^n$ where $n>=1\}$

MCQ s

- Consider the languages:

  L1 = {wwR |w ∈ {0, 1}*}

  L2 = {w#wR | w ∈ {0, 1}*}, where # is a special symbol

  L3 = {ww | w ∈ (0, 1}*)

  Which one of the following is TRUE?

  A. L1 is a deterministic CFL

  B. L2 is a deterministic CFL

  C. L3 is a CFL, but not a deterministic CFL

  D. L3 is a deterministic CFL

MCQ s

- A context free grammar can be recognized by
  - A. Push down automata
  - B. 2 way linearly bounded automata
  - **C. both a and b**
  - D. None of the mentioned

- PDA is more powerful than
  - A. Turing machine
  - B. **Finite automata**
  - C. Both (A. and (B.
  - D. None of these

MCQ s

- Which of the following statement is false?
  A. Let L is a language accepted by a PDA P then there exist a CFG G L such that L(G) =N(P)
  B. If L is a CFL then there exists a push down automata P accepting CFL L by empty stack i.e. L = N(P)
  C. If L is a language accepted by PDA A by final state there exist a PDA B that accepts L by empty stack such that L =L(A) = N(B).
  D. All of these

- Which of the following are decision properties?
  A. Emptiness
  B. Infiniteness
  C. Membership
  D. **All of the mentioned**

MCQ s

- A CFG is not closed under
  A. Dot operation
  B. Union Operation
  C. Concatenation
  D. Iteration

- A class of language that is closed under
  A. union and complementation has to be closed under intersection
  B. intersection and complement has to be closed under union
  C. union and intersection has to be closed under complementation
  D. **both (A. and (B**.

Old Question Papers

https://drive.google.com/drive/folders/19Eia3VHCl3627foiH6V_j-p4X9ZkyyC7?usp=sharing

Expected Questions for University Exam

- Describe the instantaneous description of a PDA.
- State the pumping lemma for the context free languages.
- Explain Two Stacks PDA.
- Is context free language closed under union? If yes, give an example.
- Design a PDA which accepts set of balanced paranthesis ({ { } }).
- Construct PDA for the language L = $\{a^{2n}cb^n \mid n \geq 1\}$
- Convert the grammar S$\rightarrow$ aAA, A $\rightarrow$aS | bS | a to a PDA that accepts the same language by empty stack.
- Construct a PDA from the following CFG.
- G = ({S, X}, {a, b}, P, S) where the productions are −
- S → XS | ε , X → aXb | Xb | ab

Summary

- PDA is an automata that accepts CFL.
- CFL is generated by CFG.
- A CFL is closed under: Union, concatenation, Kleene Closure and Reversal.
- A CFL is not closed under Intersection, Complement and Difference.
- Pumping lemma, here, is used to prove that a language is not CFL.
- Decision properties: Membership, Finiteness, Emptiness.
- NPDA is more powerful than DPDA.
- The power of two stack PDA is same as that of Turing machine.

References

- Hopcroft, J. E., Motwani, R., & Ullman, J. D. (2001). Introduction to automata theory, languages, and computation. *Acm Sigact News*, *32*(1), 60-65.
- Linz, P. (2006). *An introduction to formal languages and automata*. Jones & Bartlett Learning.
- Mishra, K. L. P., & Chandrasekaran, N. (2006). *Theory of Computer Science: Automata, Languages and Computation*. PHI Learning Pvt. Ltd..

Thank You