Formal Languages and Automata Theory
CS 303

Swadhin Kumar Barisal

Department of CSE, S'O'A deemed tobe University

November 22, 2020

Lecture No.: 49 & 50

What We Discussed

- Turing Machine
  - Instantaneous Description
  - Language
  - Modified Chomsky Hierarchy
  - Deterministic Turing Machine

Today's Agenda

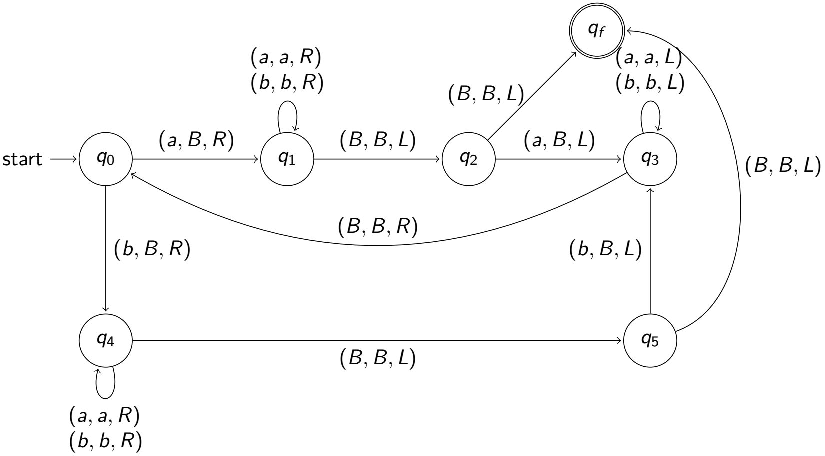
1. Turing Machine
   - Deterministic Turing Machine
   - Nondeterministic Turing Machine
2. Variation on Turing Machine
   - Two-stack PDA

Examples on Deterministic Turing Machine

Q4: Construct a Deterministic Turing Machine for
$L = \{ww^R \cup waw^R \cup wbw^R : w \in \{a, b\}^+\}$.

Examples on Deterministic Turing Machine

Q4: Construct a Deterministic Turing Machine for
$L = \{ww^R \cup waw^R \cup wbw^R : w \in \{a, b\}^+\}$.

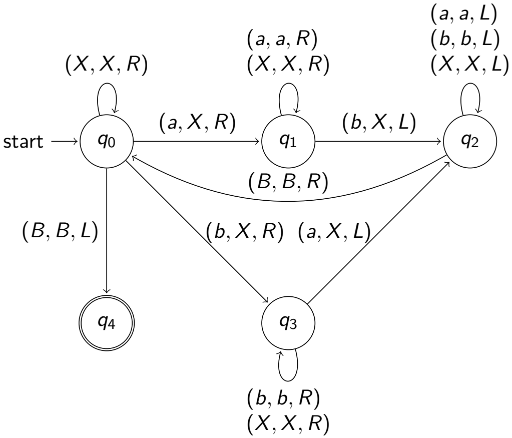


Examples on Deterministic Turing Machine

Q5: Construct a Deterministic Turing Machine for
$L = \{N_a(w) = N_b(w) : w \in \{a, b\}^*\}$.

Examples on Deterministic Turing Machine

Q5: Construct a Deterministic Turing Machine for
$L = \{N_a(w) = N_b(w) : w \in \{a, b\}^*\}$.

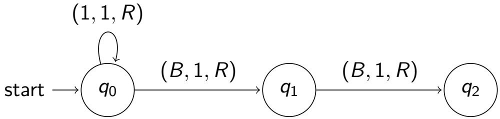


Examples on Deterministic Turing Machine

Q6: Construct a Deterministic Turing Machine to perform the function $f(x) = x + 2$, where x is a positive integer.

Examples on Deterministic Turing Machine

Q6: Construct a Deterministic Turing Machine to perform the function $f(x) = x + 2$, where x is a positive integer.

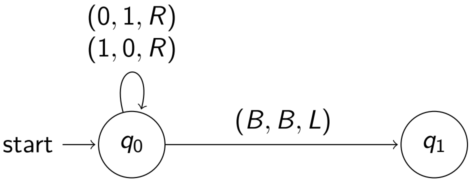


Examples on Deterministic Turing Machine

Q7: Construct a Deterministic Turing Machine, which performs 1's complement on a binary string.

Examples on Deterministic Turing Machine

Q7: Construct a Deterministic Turing Machine, which performs 1's complement on a binary string.

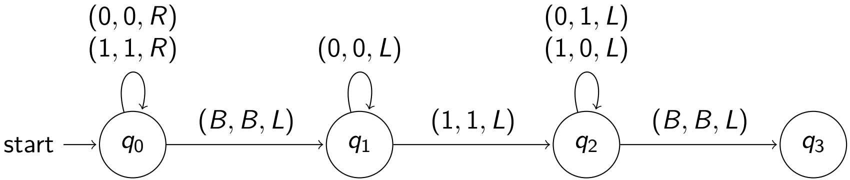


Examples on Deterministic Turing Machine

Q8: Construct a Deterministic Turing Machine, which performs 2's complement on a binary string.

Examples on Deterministic Turing Machine

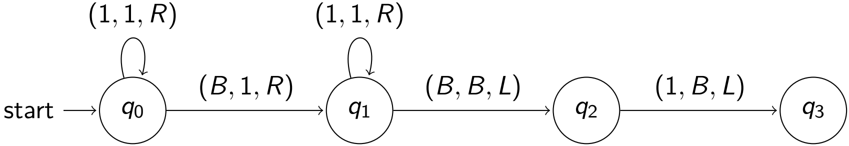
Q8: Construct a Deterministic Turing Machine, which performs 2's complement on a binary string.



Q9: Construct a Deterministic Turing Machine to perform $f(x, y) = x + y$, where x and y are two positive integers.

Examples on Deterministic Turing Machine

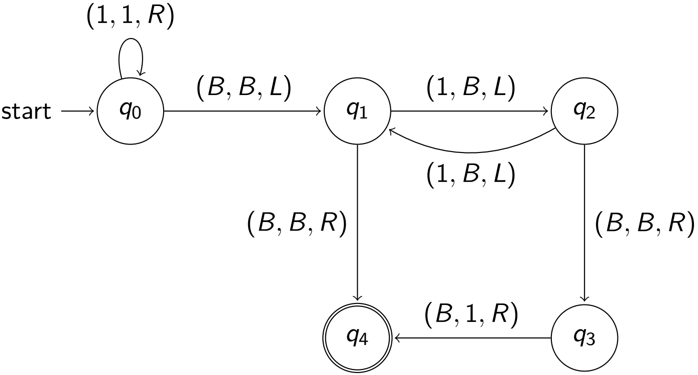
Q9: Construct a Deterministic Turing Machine to perform $f(x, y) = x + y$, where x and y are two positive integers.



Q10: Construct a Deterministic Turing Machine to test whether an integer is even or odd.

Examples on Deterministic Turing Machine

Q10: Construct a Deterministic Turing Machine to test whether an integer is even or odd.



Examples on Deterministic Turing Machine

Q1: Construct a Deterministic Turing Machine for $L = \{a^n b^n : n \geq 1\}$.
Q2: Construct a Deterministic Turing Machine for $L = \{a^n b^n c^n : n \geq 1\}$.
Q3: Construct a Deterministic Turing Machine for $L = \{ww^R : w \in \{a, b\}^+\}$.
Q4: Construct a Deterministic Turing Machine for $L = \{ww^R \cup waw^R \cup wbw^R : w \in \{a, b\}^+\}$.
Q5: Construct a Deterministic Turing Machine, which performs 2's complement on a binary string.
Q6: Construct a Deterministic Turing Machine for $L = \{N_a(w) = N_b(w) : w \in \Sigma^*\}$.
Q7: Construct a Deterministic Turing Machine to perform the function $f(x) = x + 2$, where x is a positive integer.
Q8: Construct a Deterministic Turing Machine, which performs 1's complement on a binary string.
Q9: Construct a Deterministic Turing Machine for $L = \{a^n b^{2n} : n \geq 1\}$.
Q10: Construct a Deterministic Turing Machine for $L = \{wcw^R : w \in \{a, b\}^+\}$.
Q11: Construct a Deterministic Turing Machine to concatenate two strings *aaaa* and *aaaaaa*.
Q12: Construct a Deterministic Turing Machine to perform $f(x, y) = x - y$, where x and y are two positive integers and $x > y$.

Examples on Deterministic Turing Machine

Q13: Construct a Deterministic Turing Machine for $L = \{N_a(w) + N_b(w) \text{ is even} : w \in \Sigma^*\}$.
Q14: Construct a Deterministic Turing Machine for $L = \{N_a(w) + N_b(w) \text{ is odd} : w \in \Sigma^*\}$.
Q15: Construct a Deterministic Turing Machine for $L = \{ww : w \in \Sigma^*\}$.
Q16: Construct a Deterministic Turing Machine to test a string of balanced parenthesis.
Q17: Construct a Deterministic Turing Machine to perform $f(w) = w^R$, where $w \in \{a, b\}^+$.
Q18: Construct a Deterministic Turing Machine for $L = \{a^n b^n a^n : n \geq 1\}$.
Q19: Construct a Deterministic Turing Machine for $L = \{a^n b^n a^n : n \geq 1\}$.
Q20: Construct a Deterministic Turing Machine, which acts as an eraser.
Q21: Construct a Deterministic Turing Machine for $L = \{a^{m+n} b^m c^m : m, n \geq 1\}$.
Q22: Construct a Deterministic Turing Machine for $L = \{a^n b^n c^m d^m : m, n \geq 1\}$.
Q23: Construct a Deterministic Turing Machine for

$$f(x, y) = \begin{cases} x - y \text{ if } x > y \\ 0 \quad \text{ if } x \leq y \end{cases}$$

.

Examples on Nondeterministic Turing Machine

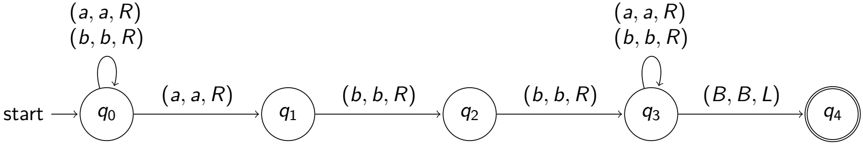
Q1: Construct a Nondeterministic Turing Machine for $L = \{\{a, b\}^* abb\{a, b\}^*\}$.

Theorem

Every non-deterministic Turing Machine has an equivalent deterministic Turing Machine.

Examples on Nondeterministic Turing Machine

Q1: Construct a Nondeterministic Turing Machine for $L = \{\{a, b\}^* abb\{a, b\}^*\}$.

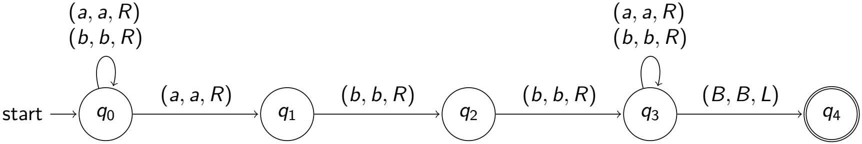


Theorem

Every non-deterministic Turing Machine has an equivalent deterministic Turing Machine.

Examples on Nondeterministic Turing Machine

Q1: Construct a Nondeterministic Turing Machine for $L = \{\{a, b\}^* abb\{a, b\}^*\}$.



Theorem

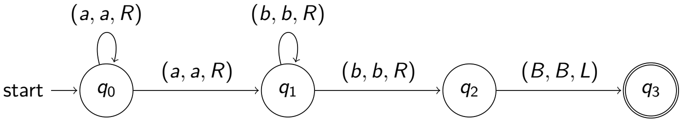
Every non-deterministic Turing Machine has an equivalent deterministic Turing Machine.

Examples on Nondeterministic Turing Machine

Q2: Construct a Nondeterministic Turing Machine for $L = \{a^n b^m : n, m \geq 1\}$.

Examples on Nondeterministic Turing Machine

Q2: Construct a Nondeterministic Turing Machine for $L = \{a^n b^m : n, m \geq 1\}$.



Variation on Turing Machine

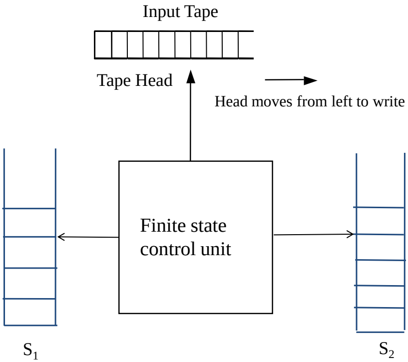
1. Deterministic Turing Machine
2. Nondeterministic Turing Machine
3. Two-stack PDA and Turing Machine
4. Linear Bounded Automaton (LBA)
5. Multi-tape Turing Machine
6. Multi-head Turing Machine
7. Enumerator
8. *k*-dimensional Turing Machine
9. Universal Turing Machine

Two-stack PDA and Minsky Theorem[1]

- The language $L = \{a^n b^n c^n : n \geq 1\}$ is a not context free language. Therefore, the PDA (with one stack) is helpless. However, a two-stack PDA (even, more than two stacks PDA) can solve this problem. Two-stack accepts all the CFLs and CSLs.

Theorem (Minsky Theorem)

Any language accepted by a two-stack PDA can also accepted by some Turing Machine and any language accepted by a Turing Machine can be accepted by some two-stack PDA.



[1] Chapter 26: Daniel I. A. Cohen: Introduction to Computer Theory

Two-stack PDA and Minsky Theorem[2]

Definition

Formally, we can define a Two-stack PDA (2PDA) by the 9-tuple $\mathcal{P} = (Q, \Sigma, \Gamma_1, \Gamma_2, \delta, q_0, z_1, z_2, F)$, where

1. Q is a finite set called the **set of states**,
2. Σ is a finite set called the **input alphabet**,
3. Γ_1 is an alphabet of **stack symbols** for stack S_1,
4. Γ_2 is an alphabet of **stack symbols** for stack S_2,
5. **Transition function**
   - $\delta : (Q \times (\Sigma \cup \{\epsilon\}) \times \Gamma_1 \times \Gamma_2) \rightarrow (Q \times \Gamma_1^* \times \Gamma_2^*)$
6. $q_0 \in Q$, called the **initial/start** state of ,
7. $z_1 \in \Gamma$ is the **initial stack symbol** stack S_1,
8. $z_2 \in \Gamma$ is the **initial stack symbol** stack S_2, and
9. $F \subseteq Q$, called the set of **final/accept** states.

[2] Chapter 26: Daniel I. A. Cohen: Introduction to Computer Theory

Two-stack PDA and Minsky Theorem[2]

Definition

Formally, we can define a Two-stack PDA (2PDA) by the 9-tuple $\mathcal{P} = (Q, \Sigma, \Gamma_1, \Gamma_2, \delta, q_0, z_1, z_2, F)$, where

1. Q is a finite set called the **set of states**,
2. Σ is a finite set called the **input alphabet**,
3. Γ_1 is an alphabet of **stack symbols** for stack S_1,
4. Γ_2 is an alphabet of **stack symbols** for stack S_2,
5. **Transition function**
   - $\delta : (Q \times (\Sigma \cup \{\epsilon\}) \times \Gamma_1 \times \Gamma_2) \rightarrow (Q \times \Gamma_1^* \times \Gamma_2^*)$
6. $q_0 \in Q$, called the **initial/start** state of ,
7. $z_1 \in \Gamma$ is the **initial stack symbol** stack S_1,
8. $z_2 \in \Gamma$ is the **initial stack symbol** stack S_2, and
9. $F \subseteq Q$, called the set of **final/accept** states.

[2] Chapter 26: Daniel I. A. Cohen: Introduction to Computer Theory

Two-stack PDA and Minsky Theorem[2]

Definition

Formally, we can define a Two-stack PDA (2PDA) by the 9-tuple $\mathcal{P} = (Q, \Sigma, \Gamma_1, \Gamma_2, \delta, q_0, z_1, z_2, F)$, where

1. Q is a finite set called the **set of states**,
2. Σ is a finite set called the **input alphabet**,
3. Γ_1 is an alphabet of **stack symbols** for stack S_1,
4. Γ_2 is an alphabet of **stack symbols** for stack S_2,
5. **Transition function**
   - $\delta : (Q \times (\Sigma \cup \{\epsilon\}) \times \Gamma_1 \times \Gamma_2) \to (Q \times \Gamma_1^* \times \Gamma_2^*)$
6. $q_0 \in Q$, called the **initial/start** state of ,
7. $z_1 \in \Gamma$ is the **initial stack symbol** stack S_1,
8. $z_2 \in \Gamma$ is the **initial stack symbol** stack S_2, and
9. $F \subseteq Q$, called the set of **final/accept** states.

[2] Chapter 26: Daniel I. A. Cohen: Introduction to Computer Theory

Two-stack PDA and Minsky Theorem[2]

Definition

Formally, we can define a Two-stack PDA (2PDA) by the 9-tuple $\mathcal{P} = (Q, \Sigma, \Gamma_1, \Gamma_2, \delta, q_0, z_1, z_2, F)$, where

1. Q is a finite set called the **set of states**,
2. Σ is a finite set called the **input alphabet**,
3. Γ_1 is an alphabet of **stack symbols** for stack S_1,
4. Γ_2 is an alphabet of **stack symbols** for stack S_2,
5. **Transition function**
   - $\delta : (Q \times (\Sigma \cup \{\epsilon\}) \times \Gamma_1 \times \Gamma_2) \rightarrow (Q \times \Gamma_1^* \times \Gamma_2^*)$
6. $q_0 \in Q$, called the **initial/start** state of ,
7. $z_1 \in \Gamma$ is the **initial stack symbol** stack S_1,
8. $z_2 \in \Gamma$ is the **initial stack symbol** stack S_2, and
9. $F \subseteq Q$, called the set of **final/accept** states.

[2] Chapter 26: Daniel I. A. Cohen: Introduction to Computer Theory

Two-stack PDA and Minsky Theorem[2]

Definition

Formally, we can define a Two-stack PDA (2PDA) by the 9-tuple $\mathcal{P} = (Q, \Sigma, \Gamma_1, \Gamma_2, \delta, q_0, z_1, z_2, F)$, where

1. Q is a finite set called the **set of states**,
2. Σ is a finite set called the **input alphabet**,
3. Γ_1 is an alphabet of **stack symbols** for stack S_1,
4. Γ_2 is an alphabet of **stack symbols** for stack S_2,
5. **Transition function**
   - $\delta : (Q \times (\Sigma \cup \{\epsilon\}) \times \Gamma_1 \times \Gamma_2) \to (Q \times \Gamma_1^* \times \Gamma_2^*)$
6. $q_0 \in Q$, called the **initial/start** state of ,
7. $z_1 \in \Gamma$ is the **initial stack symbol** stack S_1,
8. $z_2 \in \Gamma$ is the **initial stack symbol** stack S_2, and
9. $F \subseteq Q$, called the set of **final/accept** states.

[2] Chapter 26: Daniel I. A. Cohen: Introduction to Computer Theory

Two-stack PDA and Minsky Theorem[2]

Definition

Formally, we can define a Two-stack PDA (2PDA) by the 9-tuple $\mathcal{P} = (Q, \Sigma, \Gamma_1, \Gamma_2, \delta, q_0, z_1, z_2, F)$, where

1. Q is a finite set called the **set of states**,
2. Σ is a finite set called the **input alphabet**,
3. Γ_1 is an alphabet of **stack symbols** for stack S_1,
4. Γ_2 is an alphabet of **stack symbols** for stack S_2,
5. **Transition function**
   - $\delta : (Q \times (\Sigma \cup \{\epsilon\}) \times \Gamma_1 \times \Gamma_2) \to (Q \times \Gamma_1^* \times \Gamma_2^*)$
6. $q_0 \in Q$, called the **initial/start** state of ,
7. $z_1 \in \Gamma$ is the **initial stack symbol** stack S_1,
8. $z_2 \in \Gamma$ is the **initial stack symbol** stack S_2, and
9. $F \subseteq Q$, called the set of **final/accept** states.

[2] Chapter 26: Daniel I. A. Cohen: Introduction to Computer Theory

Two-stack PDA and Minsky Theorem[2]

Definition

Formally, we can define a Two-stack PDA (2PDA) by the 9-tuple $\mathcal{P} = (Q, \Sigma, \Gamma_1, \Gamma_2, \delta, q_0, z_1, z_2, F)$, where

1. Q is a finite set called the **set of states**,
2. Σ is a finite set called the **input alphabet**,
3. Γ_1 is an alphabet of **stack symbols** for stack S_1,
4. Γ_2 is an alphabet of **stack symbols** for stack S_2,
5. **Transition function**
   - $\delta : (Q \times (\Sigma \cup \{\epsilon\}) \times \Gamma_1 \times \Gamma_2) \to (Q \times \Gamma_1^* \times \Gamma_2^*)$
6. $q_0 \in Q$, called the **initial/start** state of ,
7. $z_1 \in \Gamma$ is the **initial stack symbol** stack S_1,
8. $z_2 \in \Gamma$ is the **initial stack symbol** stack S_2, and
9. $F \subseteq Q$, called the set of **final/accept** states.

[2] Chapter 26: Daniel I. A. Cohen: Introduction to Computer Theory

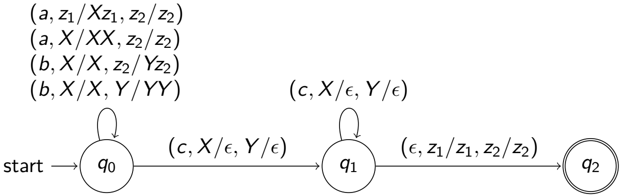
Two-stack PDA

Q1: Construct a two-stack PDA to accept the CSL $L = \{a^n b^n c^n : n \geq 1\}$ by final state.

- $\delta(q_0, a, z_1, z_2) = (q_0, Xz_1, z_2)$
- $\delta(q_0, a, X, z_2) = (q_0, XX, z_2)$
- $\delta(q_0, b, X, z_2) = (q_0, X, Yz_2)$
- $\delta(q_0, b, X, Y) = (q_0, X, YY)$
- $\delta(q_0, c, X, Y) = (q_1, \epsilon, \epsilon)$
- $\delta(q_1, c, X, Y) = (q_1, \epsilon, \epsilon)$
- $\delta(q_1, \epsilon, z_1, z_2) = (q_2, z_1, z_2)$

Two-stack PDA

Q1: Construct a two-stack PDA to accept the CSL $L = \{a^n b^n c^n : n \geq 1\}$ by final state.



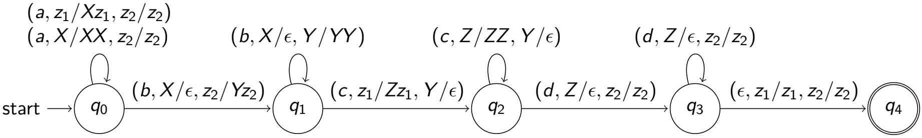
- $\delta(q_0, a, z_1, z_2) = (q_0, Xz_1, z_2)$
- $\delta(q_0, a, X, z_2) = (q_0, XX, z_2)$
- $\delta(q_0, b, X, z_2) = (q_0, X, Yz_2)$
- $\delta(q_0, b, X, Y) = (q_0, X, YY)$
- $\delta(q_0, c, X, Y) = (q_1, \epsilon, \epsilon)$
- $\delta(q_1, c, X, Y) = (q_1, \epsilon, \epsilon)$
- $\delta(q_1, \epsilon, z_1, z_2) = (q_2, z_1, z_2)$

Q2: Construct a two-stack PDA to accept the CSL $L = \{a^n b^n c^n d^n : n \geq 1\}$ by final state.

- $\delta(q_0, a, z_1, z_2) = (q_0, Xz_1, z_2)$, $\delta(q_0, a, X, z_2) = (q_0, XX, z_2)$
- $\delta(q_0, b, X, z_2) = (q_1, \epsilon, Yz_2)$, $\delta(q_1, b, X, Y) = (q_1, \epsilon, YY)$
- $\delta(q_1, c, z_1, Y) = (q_2, Zz_1, \epsilon)$, $\delta(q_2, c, Z, Y) = (q_2, ZZ, \epsilon)$
- $\delta(q_2, d, Z, z_2) = (q_3, \epsilon, z_2)$, $\delta(q_2, d, Z, z_2) = (q_3, \epsilon, z_2)$
- $\delta(q_3, \epsilon, z_1, z_2) = (q_4, z_1, z_2)$

Two-stack PDA

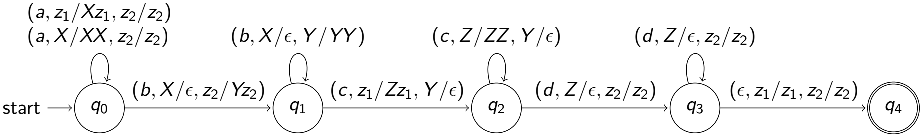
Q2: Construct a two-stack PDA to accept the CSL $L = \{a^n b^n c^n d^n : n \geq 1\}$ by final state.



- $\delta(q_0, a, z_1, z_2) = (q_0, Xz_1, z_2)$, $\delta(q_0, a, X, z_2) = (q_0, XX, z_2)$
- $\delta(q_0, b, X, z_2) = (q_1, \epsilon, Yz_2)$, $\delta(q_1, b, X, Y) = (q_1, \epsilon, YY)$
- $\delta(q_1, c, z_1, Y) = (q_2, Zz_1, \epsilon)$, $\delta(q_2, c, Z, Y) = (q_2, ZZ, \epsilon)$
- $\delta(q_2, d, Z, z_2) = (q_3, \epsilon, z_2)$, $\delta(q_2, d, Z, z_2) = (q_3, \epsilon, z_2)$
- $\delta(q_3, \epsilon, z_1, z_2) = (q_4, z_1, z_2)$

Two-stack PDA

Q2: Construct a two-stack PDA to accept the CSL $L = \{a^n b^n c^n d^n : n \geq 1\}$ by final state.



- $\delta(q_0, a, z_1, z_2) = (q_0, Xz_1, z_2)$, $\delta(q_0, a, X, z_2) = (q_0, XX, z_2)$
- $\delta(q_0, b, X, z_2) = (q_1, \epsilon, Yz_2)$, $\delta(q_1, b, X, Y) = (q_1, \epsilon, YY)$
- $\delta(q_1, c, z_1, Y) = (q_2, Zz_1, \epsilon)$, $\delta(q_2, c, Z, Y) = (q_2, ZZ, \epsilon)$
- $\delta(q_2, d, Z, z_2) = (q_3, \epsilon, z_2)$, $\delta(q_2, d, Z, z_2) = (q_3, \epsilon, z_2)$
- $\delta(q_3, \epsilon, z_1, z_2) = (q_4, z_1, z_2)$

Thank You