Theory of Computing

CSE-203

Mealy & Moore machine, Chomsky Normal Form, Ambiguity

Finite automata may have outputs corresponding to each transition. There are two types of finite state machines that generate output −

* Mealy Machine
* Moore machine

**Mealy Machine**

A Mealy Machine is an FSM whose output depends on the present state as well as the present input. It can be described by a 6 tuple (Q, ∑, O, δ, X, q0) where –

* **Q** is a finite set of states.
* **∑** is a finite set of symbols called the input alphabet.
* **O** is a finite set of symbols called the output alphabet.
* **δ** is the input transition function where δ: Q × ∑ → Q
* **X** is the output transition function where X: Q × ∑ → O
* **q0** is the initial state from where any input is processed (q0 ∈ Q).

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| The state table of a Mealy Machine is shown below −   |  |  |  |  |  | | --- | --- | --- | --- | --- | | **Present state** | **Next state** | | | | | **input = 0** | | **input = 1** | | | **State** | **Output** | **State** | **Output** | | → a | b | x1 | c | x1 | | b | b | x2 | d | x3 | | c | d | x3 | c | x1 | | d | d | x3 | d | x2 | | The state diagram of the above Mealy Machine is −  State Diagram of Mealy Machine |

**Moore Machine**

Moore machine is an FSM whose outputs depend on only the present state.

A Moore machine can be described by a 6 tuple (Q, ∑, O, δ, X, q0) where −

* **Q** is a finite set of states.
* **∑** is a finite set of symbols called the input alphabet.
* **O** is a finite set of symbols called the output alphabet.
* **δ** is the input transition function where δ: Q × ∑ → Q
* **X** is the output transition function where X: Q → O
* **q0** is the initial state from where any input is processed (q0 ∈ Q).

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| The state table of a Moore Machine is shown below –   |  |  |  |  | | --- | --- | --- | --- | | **Present state** | **Next State** | | **Output** | | **Input = 0** | **Input = 1** | | → a | b | c | x2 | | b | b | d | x1 | | c | c | d | x2 | | d | d | d | x3 | | The state diagram of the above Moore Machine is –  Moore Machine State Diagram |

**Mealy Machine vs. Moore Machine**

The following table highlights the points that differentiate a Mealy Machine from a Moore Machine.

|  |  |
| --- | --- |
| **Mealy Machine** | **Moore Machine** |
| Output depends both upon present state and present input. | Output depends only upon the present state. |
| Generally, it has fewer states than Moore Machine. | Generally, it has more states than Mealy Machine. |
| Output changes at the clock edges. | Input change can cause change in output change as soon as logic is done. |
| Mealy machines react faster to inputs | In Moore machines, more logic is needed to decode the outputs since it has more circuit delays. |

**Moore Machine to Mealy Machine**

Algorithm 1

**Input** − Moore Machine

**Output** − Mealy Machine

**Step 1** − Take a blank Mealy Machine transition table format.

**Step 2** − Copy all the Moore Machine transition states into this table format.

**Step 3** − Check the present states and their corresponding outputs in the Moore Machine state table; if for a state Qi output is m, copy it into the output columns of the Mealy Machine state table wherever Qi appears in the next state.

**Example 1:**

Let us consider the following Moore machine −

|  |  |  |  |
| --- | --- | --- | --- |
| **Present State** | **Next State** | | **Output** |
| **a = 0** | **a = 1** |
| → a | d | b | 1 |
| b | a | d | 0 |
| c | c | c | 0 |
| d | b | a | 1 |

Now we apply Algorithm 4 to convert it to Mealy Machine.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Step 1 & 2** −   |  |  |  |  |  | | --- | --- | --- | --- | --- | | **Present State** | **Next State** | | | | | **a = 0** | | **a = 1** | | | **State** | **Output** | **State** | **Output** | | → a | d |  | b |  | | b | a |  | d |  | | c | c |  | c |  | | d | b |  | a |  | | **Step 3** −   |  |  |  |  |  | | --- | --- | --- | --- | --- | | **Present State** | **Next State** | | | | | **a = 0** | | **a = 1** | | | **State** | **Output** | **State** | **Output** | | => a | d | 1 | b | 0 | | b | a | 1 | d | 1 | | c | c | 0 | c | 0 | | d | b | 0 | a | 1 | |

**Mealy Machine to Moore Machine**

Algorithm 2

**Input** − Mealy Machine

**Output** − Moore Machine

**Step 1** − Calculate the number of different outputs for each state (Qi) that are available in the state table of the Mealy machine.

**Step 2** − If all the outputs of Qi are same, copy state Qi. If it has n distinct outputs, break Qi into n states as Qin where **n** = 0, 1, 2.......

**Step 3** − If the output of the initial state is 1, insert a new initial state at the beginning which gives 0 output.

**Example 2:** Let us consider the following Mealy Machine −

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Present State** | **Next State** | | | |
| **a = 0** | | **a = 1** | |
| **Next State** | **Output** | **Next State** | **Output** |
| → a | d | 0 | b | 1 |
| b | a | 1 | d | 0 |
| c | c | 1 | c | 0 |
| d | b | 0 | a | 1 |

Here, states ‘a’ and ‘d’ give only 1 and 0 outputs respectively, so we retain states ‘a’ and ‘d’. But states ‘b’ and ‘c’ produce different outputs (1 and 0). So, we divide **b** into **b0, b1** and **c** into **c0, c1**.

|  |  |  |  |
| --- | --- | --- | --- |
| **Present State** | **Next State** | | **Output** |
| **a = 0** | **a = 1** |
| → a | d | b1 | 1 |
| b0 | a | d | 0 |
| b1 | a | d | 1 |
| c0 | c1 | C0 | 0 |
| c1 | c1 | C0 | 1 |
| d | b0 | a | 0 |

**Chomsky Normal Form**

A CFG is in Chomsky Normal Form if the Productions are in the following forms −

* A → a
* A → BC
* S → ε

where A, B, and C are non-terminals and **a** is terminal.

**Algorithm to Convert into Chomsky Normal Form −**

**Step 1** − If the start symbol **S** occurs on some right side, create a new start symbol **S’** and a new production **S’→ S**.

**Step 2** − Remove Null productions. (Using the Null production removal algorithm discussed earlier)

**Step 3** − Remove unit productions. (Using the Unit production removal algorithm discussed earlier)

**Step 4** − Replace each production **A → B1…Bn** where **n > 2** with **A → B1C** where **C → B2 …Bn**. Repeat this step for all productions having two or more symbols in the right side.

**Step 5** − If the right side of any production is in the form **A → aB** where a is a terminal and **A, B** are non-terminal, then the production is replaced by **A → XB** and **X → a**. Repeat this step for every production which is in the form **A → aB**.

**Problem 1:**

Convert the following CFG into CNF

S → ASA | aB, A → B | S, B → b | ε

Solution

**(1)** Since **S** appears in R.H.S, we add a new state **S0** and **S0→S** is added to the production set and it becomes −

S0→S, S→ ASA | aB, A → B | S, B → b | ∈

**(2)** Now we will remove the null productions −

B → ∈ and A → ∈

After removing B → ε, the production set becomes −

S0→S, S→ ASA | aB | a, A → B | S | ∈, B → b

After removing A → ∈, the production set becomes −

S0→S, S→ ASA | aB | a | AS | SA | S, A → B | S, B → b

**(3)** Now we will remove the unit productions.

After removing S → S, the production set becomes −

S0→S, S→ ASA | aB | a | AS | SA, A → B | S, B → b

After removing S0→ S, the production set becomes −

S0→ ASA | aB | a | AS | SA, S→ ASA | aB | a | AS | SA

A → B | S, B → b

After removing A→ B, the production set becomes −

S0 → ASA | aB | a | AS | SA, S→ ASA | aB | a | AS | SA

A → S | b

B → b

After removing A→ S, the production set becomes −

S0 → ASA | aB | a | AS | SA, S→ ASA | aB | a | AS | SA

A → b |ASA | aB | a | AS | SA, B → b

**(4)** Now we will find out more than two variables in the R.H.S

Here, S0→ ASA, S → ASA, A→ ASA violates two Non-terminals in R.H.S.

Hence we will apply step 4 and step 5 to get the following final production set which is in CNF −

S0→ AX | aB | a | AS | SA

S→ AX | aB | a | AS | SA

A → b |AX | aB | a | AS | SA

B → b

X → SA

**(5)** We have to change the productions S0→ aB, S→ aB, A→ aB

And the final production set becomes −

S0→ AX | YB | a | AS | SA

S→ AX | YB | a | AS | SA

A → b A → b |AX | YB | a | AS | SA

B → b

X → SA

Y → a

**Ambiguity**

**Problem 2:**

Explain why the grammar below is ambiguous.

S → 0A | 1B

A → 0AA | 1S | 1

B →1BB | 0S | 0

Solution

The grammar is ambiguous because we can find strings which have multiple derivations:

S ⇒ 0A⇒00AA⇒001S1⇒0011B1⇒001101

S ⇒ 0A⇒00AA⇒0011S⇒00110A⇒001101

**Problem 3:**

Given the following ambiguous context free grammar

S → Ab | aaB

A → a | Aa

B →b

Solution

Find the string s generated by the grammar that has two leftmost derivations. Show the derivations. The string s = aab has the following two leftmost derivations

S ⇒ aaB ⇒ aab

S ⇒ AB ⇒ AaB ⇒ aaB ⇒ aab

Homework

Construct context free grammars to accept the following languages. Σ= {0, 1}

1. {w | w starts and ends with the same symbol}
2. {w | w is odd}
3. {w | w is odd and its middle symbol is 0}
4. {0n1n | n > 0} U {0n12n | n>0}
5. {0i1j2k | i≠j or j≠k}
6. Binary strings with twice as many 1s as 0s.