Министерство образования и науки Кыргызской Республики

Кыргызско-Германский институт прикладной информатики

Лабораторная работа №4

по теме «Грамматики и конечные автоматы»

Выполнила: студентка группы AIN-1-22

Тахирова К.

Проверила:

Доцент КГИПИ

Кошоева Б. Б.

Оглавление

[1 Цель работы 3](#_Toc181734049)

[2 Постановка задачи 3](#_Toc181734050)

[3 Математическая модель автомата 3](#_Toc181734051)

[4 Способы задания автомата 3](#_Toc181734052)

[4.1 Табличный способ 3](#_Toc181734053)

[4.2 Графический способ 4](#_Toc181734054)

[4.3 Матричный способ 4](#_Toc181734055)

[5 Программное решение 5](#_Toc181734056)

[6 Вывод 8](#_Toc181734057)

# Цель работы

Целью данной лабораторной работы является изучение методов задания конечных автоматов и разработка программного модуля для работы с матрицей выходов.

# Постановка задачи

* Построить таблицы переходов и выходов, а также ориентированный граф переходов и матрицу соединений заданного автомата.
* Написать и отладить программный модуль формирования матрицы выходов заданного автомата.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Вариант | Тип автомата | Количество входных сигналов | Количество состояний | Количество выходов |
| 12 | Мура | 5 | 3 | 3 |

# Математическая модель автомата

Математическое описание автомата: 𝐴 = (𝑆, 𝛴, 𝛥, 𝛿, 𝑠0)

* S — конечное множество состояний {S0, S1, S2, S3}
* Σ — конечное множество входных символов (алфавит) {X0, X1}
* Δ — конечное множество выходных символов {Y0, Y1}
* 𝛿: 𝑆 × 𝛴 → 𝑆 — переходная функция,
* 𝑆0 ∈ 𝑆 — начальное состояние.

В автомате Мура выходной сигнал определяется только текущим состоянием, а не входным сигналом.

# Способы задания автомата

## Табличный способ

Параметры автомата:

* **Состояния (S)**: , ,
* **Входные сигналы (X)**: , , , ,
* **Выходы (Y)**: , ,

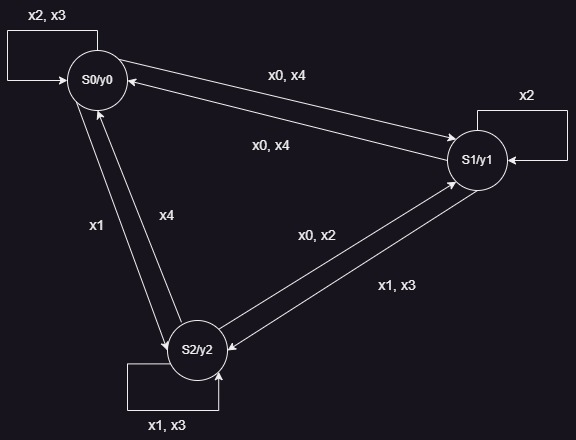
Таблица переходов:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Текущее состояние | Входной сигнал | Входной сигнал | Входной сигнал | Входной сигнал | Входной сигнал |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

Таблица выходов:

|  |  |  |  |
| --- | --- | --- | --- |
| Состояние | Выход | Выход | Выход |
|  | 1 | 0 | 0 |
|  | 0 | 1 | 0 |
|  | 0 | 0 | 1 |

## Графический способ



## Матричный способ

Матрица соединений будет представлять состояние переходов и выходов:

C = ,

где элемент ​ (например, ) соответствует переходу из состояния в состояние ​ с указанием входного сигнала и выходного сигнала. Можно представить эту матрицу следующим образом, где элемент матрицы содержит информацию о переходах:

C =

|  |  |  |  |
| --- | --- | --- | --- |
| Состояние |  |  |  |
|  | 1 | 1 | 1 |
|  | 1 | 1 | 1 |
|  | 1 | 1 | 1 |

# Программное решение

package tahirova\_ain1;  
  
import org.graphstream.graph.Edge;  
import org.graphstream.graph.Graph;  
import org.graphstream.graph.implementations.SingleGraph;  
  
public class FiniteAutomaton {  
 private static final String[] *STATES* = {"S0", "S1", "S2"};  
 private static final String[] *INPUTS* = {"x0", "x1", "x2", "x3", "x4"};  
 private static final String[] *OUTPUTS* = {"y0", "y1", "y2"};  
  
 //[текущее состояние][входной сигнал] -> следующее состояние  
 private static final int[][] *TRANSITION\_TABLE* = {  
 {1, 2, 0, 0, 1}, // S0  
 {0, 2, 1, 2, 0}, // S1  
 {1, 2, 1, 2, 0} // S2  
 };  
  
 //[состояние] -> выходной сигнал  
 private static final int[] *OUTPUT\_TABLE* = {0, 1, 2}; // y0 для S0, y1 для S1, y2 для S2  
  
 public static void main(String[] args) {  
 System.*out*.println("Таблица переходов:");  
 *printTransitionTable*();  
 System.*out*.println("\nТаблица выходов:");  
 *printOutputTable*();  
 System.*out*.println("\nМатрица соединений:");  
 *printConnectionMatrix*();  
  
 *displayGraph*();  
 }  
  
 private static void printTransitionTable() {  
 System.*out*.print("Текущее состояние\t");  
 for (String input : *INPUTS*) {  
 System.*out*.printf("%-16s", input);  
 }  
 System.*out*.println();  
  
 for (int currentState = 0; currentState < *STATES*.length; currentState++) {  
 System.*out*.printf("%-20s", *STATES*[currentState]);

for (int input = 0; input < *INPUTS*.length; input++) {  
 int nextState = *TRANSITION\_TABLE*[currentState][input];  
 System.*out*.printf("%-16s", *STATES*[nextState]);  
 }

System.*out*.println();  
 }  
}  
  
private static void printOutputTable() {  
 System.*out*.printf("%-16s%-10s%n", "Состояние", "Выход");  
 for (int currentState = 0; currentState < *STATES*.length; currentState++) {  
 String outputSignal = *OUTPUTS*[*OUTPUT\_TABLE*[currentState]];  
 System.*out*.printf("%-16s%-10s%n", *STATES*[currentState], outputSignal);  
 }  
}  
  
private static void printConnectionMatrix() {  
 System.*out*.print(" ");  
 for (String input : *INPUTS*) {  
 System.*out*.printf("%-15s", input);  
 }  
 System.*out*.println();  
  
 for (int currentState = 0; currentState < *STATES*.length; currentState++) {  
 System.*out*.printf("%-5s", *STATES*[currentState]);  
  
 for (int input = 0; input < *INPUTS*.length; input++) {  
 int nextState = *TRANSITION\_TABLE*[currentState][input];  
 String outputSignal = *OUTPUTS*[*OUTPUT\_TABLE*[nextState]];  
  
 System.*out*.printf("(%d / %s, %s) ", nextState, *INPUTS*[input], outputSignal);  
 }  
 System.*out*.println();  
 }  
}

private static void displayGraph() {  
 System.*setProperty*("org.graphstream.ui", "swing");  
 Graph graph = new SingleGraph("FiniteAutomaton", false, true);  
  
 for (int i = 0; i < *STATES*.length; i++) {  
 graph.addNode(*STATES*[i]).setAttribute("ui.label", *STATES*[i] + " / " + *OUTPUTS*[i]);  
 if (i == 0) {  
 graph.getNode(*STATES*[i]).setAttribute("ui.class", "initial");  
 }  
 }  
  
 int initialOffset = 10;  
 int horizontalIncrement = 15;  
 int verticalIncrement = 5;  
   
 for (int currentState = 0; currentState < *STATES*.length; currentState++) {  
 for (int input = 0; input < *INPUTS*.length; input++) {  
 int nextState = *TRANSITION\_TABLE*[currentState][input];  
 String fromState = *STATES*[currentState];  
 String toState = *STATES*[nextState];

String edgeId = fromState + "-" + *INPUTS*[input] + "-" + currentState + "-" + input;

if (graph.getEdge(edgeId) == null) {  
 Edge edge = graph.addEdge(edgeId, fromState, toState, true);  
 if (edge != null) {  
 String label = *INPUTS*[input] + " / " + *OUTPUTS*[*OUTPUT\_TABLE*[nextState]];  
 edge.setAttribute("ui.label", label);  
  
 // Alternate label position between above and below, with increasing offset  
 if ((currentState + input) % 2 == 0) {  
 edge.setAttribute("ui.style", "text-alignment: above; text-offset: " + initialOffset + "px, " + verticalIncrement + "px;");  
 } else {  
 edge.setAttribute("ui.style", "text-alignment: under; text-offset: " + (initialOffset + horizontalIncrement) + "px, -" + verticalIncrement + "px;");  
 }  
  
 initialOffset += horizontalIncrement;  
 verticalIncrement += 5;  
 }  
 }  
 }  
 }  
  
 graph.setAttribute("ui.stylesheet",  
 "node { " +  
 "fill-color: lightblue;" +  
 "size: 40px;" +  
 "text-size: 16;" +  
 "text-color: black;" +  
 "text-background-mode: plain;" +  
 "text-padding: 10;" +  
 "}" +  
 "node.initial { " +  
 "fill-color: lightgreen;" +  
 "}" +  
 "edge {" +  
 "text-size: 14;" +  
 "text-color: black;" +  
 "arrow-size: 12px, 8px;" +  
 "text-offset: 15, 25;" +  
 "text-background-mode: plain;" +  
 "text-padding: 5;" +  
 "}");  
  
 graph.display().enableAutoLayout();  
 }  
}

# Вывод

Для данной лабораторной работы, целью которой было исследование и разработка системы автоматов с применением матричного способа представления, можно сделать следующие выводы:

1. **Матричный способ представления автоматов** показал свою эффективность в структурировании данных о переходах и выходах автомата, предоставляя наглядную и упорядоченную модель. Матрица соединений позволяет четко задать переходы между состояниями в зависимости от входных сигналов и соответствующие выходы, что значительно облегчает анализ и обработку автомата.
2. **Реализация таблиц переходов и выходов** подтвердила, что для автоматов Мура выходной сигнал зависит исключительно от состояния автомата, что упрощает вычисления и делает этот тип автоматов более предсказуемым при анализе их работы.
3. **Программная реализация** алгоритма для формирования матрицы переходов и выходов позволила на практике оценить работу автоматов Мура, а также визуализировать граф переходов, что способствовало более глубокому пониманию процессов, происходящих в автомате.

В результате выполнения работы были получены важные навыки и знания, касающиеся построения и анализа автоматов с использованием различных методов представления, включая табличный, матричный и графический.