Содержание

[1 Теоретическая часть 5](#_Toc465108414)

[1.1 Понятие конечного автомата 5](#_Toc465108415)

[1.2 Детерминированный и недетерминированный конечный автомат 6](#_Toc465108416)

[1.3 Способы описания конечных автоматов 8](#_Toc465108417)

[2 Практическая часть 9](#_Toc465108418)

[2.1 Постановка задачи 9](#_Toc465108419)

[2.2 Программная реализация 9](#_Toc465108420)

[2.3 Текст программы 10](#_Toc465108421)

[Заключение 12](#_Toc465108422)

[Список литературы 13](#_Toc465108423)

Введение

В качестве примера, иллюстрирующего сущность конечного автомата, часто приводят автомат для продажи каких-либо товаров, например кофе. Принимаемые автоматом купюры соответствуют заранее записанным входным параметрам, так называемым, символам. Автомат в своей памяти хранит информацию о принятой до настоящего момента общей сумме, к которой будет прибавляться новая сумма принятой купюры. Эта хранящаяся в памяти прежняя сумма соответствует состоянию конечного автомата. Когда автомат принимает деньги, составляющие цену порции кофе, срабатывает механическое устройство, выдающее покупателю товар, и принятая автоматом сумма сбрасывается в ноль.

Некоторые автоматы не просто продают какой-то один вид товара. В них имеются дисплеи, на которых отображается внесенная сумма. Указываемая на дисплее сумма соответствует состоянию, набор внесенных купюр соответствует входному символу, а выдаваемый автоматом товар – выходному символу. Если даже бросить монеты в автомат наугад, но сумма превысит определенную величину, товар будет выдан вместе со сдачей. Брошенная в автомат сумма должна иметь верхний предел, то есть не может быть неограниченно большой. Таким образом, число состояний, число входных символов (число комбинаций купюр) и число выходных символов (число выдаваемых разновидностей товара вместе со сдачей) являются конечными.

Как только автомат установлен, он включается в сеть обслуживания, которая следит за регулярным изъятием денег и пополнением автомата товаром. Поэтому время работы автомата можно считать неограниченным.

Устройство выполнения операций, устройство управления и оперативное (основное) запоминающее устройство, входящие в состав машины Тьюринга, делают ее машиной с ограниченным числом состояний, то есть самым настоящим конечным автоматом. Другими словами, при запоминании конечного числа каких-то элементов можно характеристики этих элементов представить в виде набора состояний конечного автомата. Бесконечное число состояний или, говоря языком логики, сколь угодно большое наперед заданное число состояний запомнить невозможно.

В конечном автомате нельзя запомнить число состояний, большее числа, заранее заданного для этого автомата, которое определяется параметрами рабочих лент. Хотя число разновидностей символов, используемых на лентах, является конечным, число ячеек может быть бесконечным, как бесконечен сам натуральный ряд чисел. Как в двоичной, так и в десятичной системах счисления на рабочих лентах можно записать последовательность цифр с требуемым числом разрядов и, таким образом, запомнить любое самое большое число данной разрядности.

Целью данной курсовой работы является ЛИСП-реализация конечных автоматов.

1. Теоретическая часть
   1. Понятие конечного автомата

Конечный автомат – в теории алгоритмов математическая абстракция, позволяющая описывать пути изменения состояния объекта в зависимости от его текущего состояния и входных данных, при условии, что общее возможное количество состояний конечно. Конечный автомат является частным случаем абстрактного автомата.

Существуют различные варианты задания конечного автомата. Например, конечный автомат может быть задан с помощью пяти параметров:

(1)

где

– конечное множество состояний автомата;

– начальное состояние автомата ;

– множество заключительных (или допускающих) состояний, таких что ;

– допустимый входной алфавит (конечное множество допустимых входных символов), из которого формируются строки, считываемые автоматом;

δ – заданное отображение множества во множество подмножеств . Иногда δ называют функцией переходов автомата.

(2)

Автомат начинает работу в состоянии , считывая по одному символу входной строки. Считанный символ переводит автомат в новое состояние из Q в соответствии с функцией переходов. Если по завершении считывания входного слова (цепочки символов) автомат оказывается в одном из допускающих состояний, то слово «принимается» автоматом. В этом случае говорят, что оно принадлежит языку данного автомата. В противном случае слово «отвергается».

Также для автомата можно определить язык (множество слов) в алфавите , который он представляет – так называется множество слов, при вводе которых автомат переходит из начального состояния в одно из состояний множества .

Конечные автоматы широко используются на практике, например в синтаксических, лексических анализаторах, и тестировании программного обеспечения на основе моделей.

* 1. Детерминированный и недетерминированный конечный автомат

Конечные автоматы подразделяются на детерминированные и недетерминированные.

Детерминированным конечным автоматом (ДКА) называется такой автомат, в котором для каждой последовательности входных символов существует лишь одно состояние, в которое автомат может перейти из текущего.

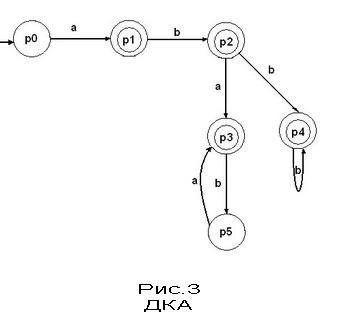


Рисунок 1.1 – Детерминированный конечный автомат

Недетерминированный конечный автомат (НКА) является обобщением детерминированного автомата. Недетерминированность автоматов достигается двумя способами.

1. Существуют переходы, помеченные пустой цепочкой (Рисунок 1.2);
2. Из одного состояния выходит несколько переходов, помеченных одним и тем же символом (Рисунок 1.3).

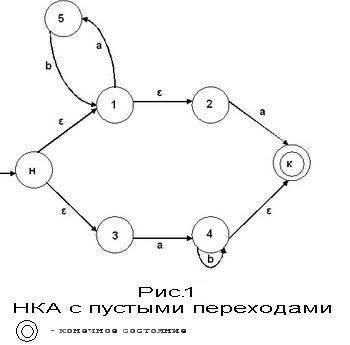


Рисунок 1.2 – Недетерминированный конечный автомат с пустыми переходами

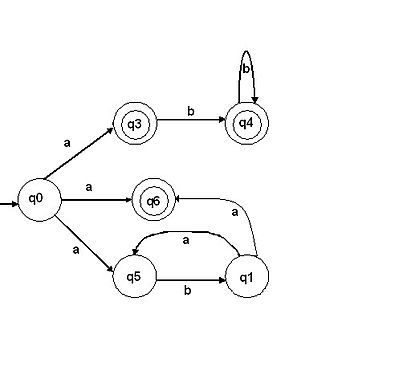


Рисунок 1.3 – Недетерминированный конечный автомат с несколькими переходами

Теоретически, любой недетерминированный конечный автомат может быть преобразован в детерминированный так, чтобы языки этих автоматов совпадали. Такие автоматы называются эквивалентными. Однако, поскольку количество состояний в эквивалентном детерминированном конечном автомате растёт экспоненциально с ростом количества состояний недетерминированного конечного автомата, на практике подобная детерминизация не всегда возможна. Кроме того, конечные автоматы с выходом в общем случае не поддаются детерминизации.

Поэтому, несмотря на большую сложность недетерминированных конечных автоматов, для задач, связанных с обработкой текста, преимущественно применяются именно недетерминированные конечные автоматы.

* 1. Способы описания конечных автоматов

Диаграмма состояний (или иногда граф переходов) – графическое представление множества состояний и функции переходов. Представляет собой нагруженный однонаправленный граф.

* Вершины – состояния конечного автомата;
* Дуги – переходы из одного состояния в другое;
* Веса – символы, при которых осуществляется данный переход.

Если переход из состояния в может быть осуществлен при появлении одного из нескольких символов, то над дугой должны быть надписаны все эти символы.

Таблица переходов – табличное представление функции (2).

* Строки – состояние конечного автомата;
* Столбцы – допустимые входные символы;
* Ячейки – действие, которое должен выполнить автомат, при переходе в это состояние и в то же время когда на входе находится этот символ.

1. Практическая часть
   1. Постановка задачи

Необходимо подсчитать количество слов начинающихся с двух русских строчных гласных букв подряд.

* 1. Программная реализация

Слова в файле могут быть записаны в несколько строк, в каждой строке может располагаться разное количество слов, слова обычно разделяются пробелом, однако при наборе текста могли быть допущены ошибки и вместо пробела был введен другой печатный символ.

Таким образом, в исходных данных выделяют две части:

* base – базовое состояние;
* word – начало слова (первый буквенный символ после знаков препинания, пробела или цифры).

Построим диаграмму состояний конечного автомата для решения задачи:

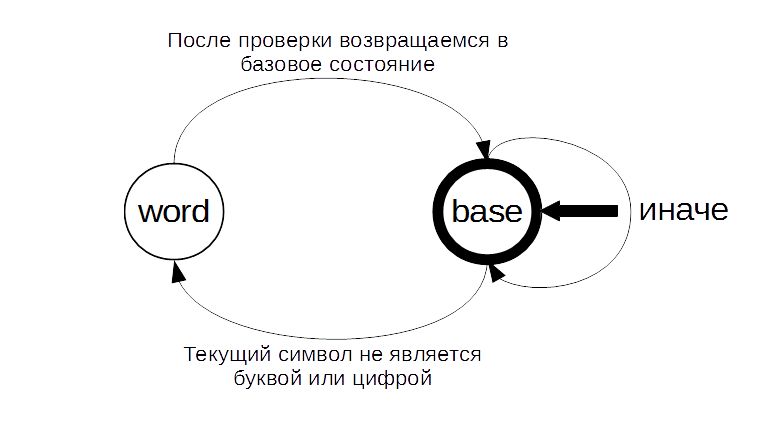


Рисунок 2.1 – Диаграмма состояний

Таким образом, только находясь в состоянии word, программа должна искать и подсчитывать слова, начинающиеся с двух русских гласных строчных букв. Программа переходит в состояние word, когда обнаружит первый символ нового слова, которое начинается с гласной буквы. После чего проверяет следующую букву и не зависимо от результата проверки, переходит в состояние base, где ожидает начала следующего слова, начинающегося с гласной буквы.

Поскольку в конечном автомате всего 2 состояния, в программной реализации тип, используемый для переменной состояния, будет логическим (boolean).

Построен конечный автомат без выхода ,

– входной алфавит совпадает с множеством символов кодовой таблицы

– декартово множество произведений, функция перехода состояний

– начальное состояние

– множество финальных состояний

Функция перехода состояний можно задать с помощью диаграммы состояний или таблицы состояний.

Построим таблицу переходов конечного автомата для решения задачи:

Таблица 1 – Таблица переходов

|  |  |  |  |
| --- | --- | --- | --- |
|  | Первая буква слова | Другая буква | Не буква |
| **→ \* base** | – | base | word |
| **word** | base | – | base |

* 1. Текст программы

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Text;

using System.Windows.Forms;

using System.IO;

namespace Работа\_с\_файлами

{

public partial class Form1 : Form

{

String MyFileName = "";

public Form1()

{

InitializeComponent();

}

private void button1\_Click(object sender, EventArgs e)//процедура выбора файла

{

OpenFileDialog MyOpenFileDialog = new OpenFileDialog();

MyOpenFileDialog.Filter = "Text|\*.txt";

if (MyOpenFileDialog.ShowDialog() == DialogResult.OK)

{

MyFileName = MyOpenFileDialog.FileName;

StreamReader MyStreamReader = new StreamReader(MyFileName,System.Text.Encoding.Default);

textBox1.Text = MyStreamReader.ReadToEnd();

MyStreamReader.Close();

button2.Enabled = true;

}

}

private void button2\_Click(object sender, EventArgs e)//процедура подсчета количества слов

{

Byte k\_wordsglas = 0;//счетчики

Boolean State = true;//переменная состояний

String Glas = "уеыаоэяию";//список русских гласный строчных букв

String specsymbols = " /?.>,<'\"]}[{;:\\|=+-\_)(\*&^%$#@!~`№0123456789";//список "не букв"

StreamReader MyStreamReader = new StreamReader(MyFileName, System.Text.Encoding.Default);//начало чтения файла

String s = MyStreamReader.ReadLine();

while(s!=null)//перебор строк

{

State = true;//состояние word

if (specsymbols.Contains(Convert.ToString(s[0])))//если первый символ не буква (первая проверка для состояния base)

State = false;//базовое состояние

for (Byte i = 0; i < s.Length - 1; i++)//перебор символов

{

if (State==true)//если состояние word

{

if (Glas.Contains(Convert.ToString(s[i])))//если текущий символ - гласная строчная буква

if (Glas.Contains(Convert.ToString(s[i + 1])))//если следующий символ - гласная строчная буква

k\_wordsglas++;//увеличение счетчика

State = false;//переход в базовое состояние

}

else

if (specsymbols.Contains(Convert.ToString(s[i])))//если текущий символ не буква

State = true;//переход в состояние word

}

s = MyStreamReader.ReadLine();//чтение следующий строки

}

MyStreamReader.Close();//завершение чтения файла

label1.Text = "Количество слов начинающихся двух русских строчных гласных букв: " + Convert.ToString(k\_wordsglas);//вывод количества на форму

}

}

}

Заключение

Мышление в терминах конечных автоматов (то есть разбиение исполнения программы на шаги автомата и передача информации от шага к шагу через состояние) необходимо при построении событийно-ориентированных приложений. В этом случае программирование в стиле конечных автоматов оказывается единственной альтернативой порождению множества процессов или потоков управления.

Часто понятие состояний и машин состояний используется для спецификации программ. Так, при проектировании программного обеспечения с помощью UML для описания поведения объектов используются диаграммы состояний. Кроме того, явное выделение состояний используется в описании сетевых протоколов.

Итогом работы можно считать созданную функциональную модель реализации конечных автоматов. Созданная функциональная модель и ее программная реализация могут служить органической частью решения более сложных задач.

Список литературы

1. Хювенен Э. Мир Лиспа [Текст] / Э. Хювенен, Й. Сеппянен. – М.: Мир, 1990. – 460 с.
2. Дехтярь, М.И. Введение в схемы, автоматы и алгоритмы. [Электронный ресурс] / М.И. Дехтярь. – М.: Наука, 2002. С. 642.
3. Мозговой, М.В. Классика программирования: алгоритмы, языки, автоматы, компиляторы. Практический подход. / М.В. Мозговой. – М.: Наука и Техника, 2006. С. 320.
4. Семакин, И.Г. Основы программирования. [Текст] / И.Г. Семакин, А.П. Шестаков. – М.: Мир, 2006. C. 346.
5. Симанков, В.С. Основы функционального программирования [Текст] / В.С. Симанков, Т.Т. Зангиев, И.В. Зайцев. – Краснодар: КубГТУ, 2002. – 160 с.
6. Степанов, П.А. Функциональное программирование на языке Lisp. [Электронный ресурс] / П.А. Степанов, А.В. Бржезовский. – М.: ГУАП, 2003. С. 79.
7. Бронштейн, И.Н. Справочник по математике для инженеров и учащихся втузов [Текст] / И.Н. Бронштейн, К.А. Семендяев. – М.: Наука, 2007. – 708 с.
8. Конечный автомат [Электронный ресурс] – Режим доступа: http://ru/wikipedia.org/wiki/Конечный\_автомат.