Нижегородский государственный технический университет им. Р.Е. Алексеева.

Институт радиоэлектроники и информационных технологий.

Кафедра «Прикладная информатика и математика».

Отчёт по лабораторной работе:

«Минимизация конечных автоматов»

Выполнил: студент 2 курса группы 23-ПМ-1 Гугин И.В.

Научный руководитель: Санников Н.А.

Нижний Новгород

2024

Содержание

[**Вступление** 3](#_Toc182485468)

[**Цель работы:** 3](#_Toc182485469)

[**Задачи:** 3](#_Toc182485470)

[**Теоретическая справка** 4](#_Toc182485471)

[**Определение 3.1**. 4](#_Toc182485472)

[**Определение 3.2**. 4](#_Toc182485473)

[**Определение 3.3**. 4](#_Toc182485474)

[**Алгоритм 3.1. Устранение недостижимых состояний КА** 4](#_Toc182485475)

[**Пример 3.1**. 5](#_Toc182485476)

[**Алгоритм 3.2. Объединение эквивалентных состояний КА** 6](#_Toc182485477)

[**Пример 3.2.** 7](#_Toc182485478)

[**Реализация:** 8](#_Toc182485479)

[**Результат работы:** 9](#_Toc182485480)

[**Вывод:** 11](#_Toc182485481)

# Вступление

## Цель работы:

закрепить понятия «недостижимые состояния автомата», «эквивалентные состояния автомата», «минимальный конечный автомат».

## Задачи:

1) ввод исходного конечного автомата и вывод на экран его графа;

2) устранение недостижимых состояний конечного автомата;

3) исключение эквивалентных состояний конечного автомата;

4) вывод на экран графа минимального конечного автомата.

5)Сделать проверку используя грамматику моего варианта(7):

|  |  |
| --- | --- |
| |  | | --- | |  | |

# Теоретическая справка

Конечный автомат может содержать лишние состояния двух типов: недостижимые и эквивалентные состояния.

## Определение 3.1.

Два различных состояния  и  в конечном автомате  называются *n*-эквивалентными, *n*∈*N*∪{0}, если, находясь в одном их этих состояний и получив на вход любую цепочку символов *ω*: *ω* ∈*VT\**, *|ω|*≤*n*, автомат может перейти в одно и то же множество конечных состояний.

Определение 3.2.Состояние *q* КА называется недостижимым, если к нему нет пути из начального состояния автомата.

Определение 3.3.КА, не содержащий недостижимых и эквивалентных состояний, называется приведенным или минимальным КА.

## Алгоритм 3.1. Устранение недостижимых состояний КА

Вход: КА .

Выход: КА .

Шаг 1. Поместить начальное состояние КА в список достижимых состояний , т.е. .

Шаг 2. Для новых элементов списка достижимых состояний пополнить список группой их состояний-приемников, отсутствующих в нем, т.е. .

Шаг 3. Повторить шаг 2, пока список достижимых состояний не перестанет меняться. То есть, если , то *i*:=*i*+1, иначе .

Шаг 4. Исключить из множества *Q* состояний КА все состояния, отсутствующие в списке *Qд* достижимых состояний, т.е. .

Шаг 5. Исключить недостижимые заключительные состояния и пары функции переходов, содержащие недостижимые состояния, т.е. , .

Пример 3.1.Устранить недостижимые состояния КА , где *Q* = {*A*, *B*, *C*, *D*, *E*, *F*, *G*}, *T* = {*a*, *b*}, *H* = {*A*}, *Z* = {*D*, *E*} и функция переходов задана таблицей 3.1. Граф исходного КА *М* представлен на рисунке 3.1.

Таблица 3.1 – Функция переходов конечного автомата *M*

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| *F* | *A* | *B* | *C* | *D* | *E* | *F* | *G* |
| *a* | *B* |  |  | *C* | *B* | *D* | *F* |
| *b* | *C* | *D* | *E* | *E* | *D* | *G* | *E* |

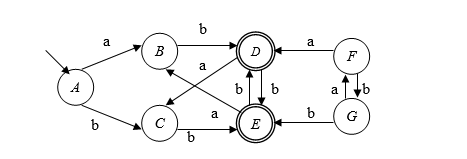


Рисунок 3.1 – Граф исходного конечного автомата *М*

Последовательность устранения недостижимых состояний КА имеет вид:

*Q*0 = {*A*};

*Q*1 = {*A*, *B*, *C*};

*Q*2 = {*A*, *B*, *C*, *D*, *E*};

*Q*3 = {*A*, *B*, *C*, *D*, *E*}; т.к. *Q*2 = *Q*3, то *Q*д = {*A*, *B*, *C*, *D*, *E*}.

*Q*н = {*F*, *G*}; *****=*** {*A*, *B*, *C*, *D*, *E*}; **= {*D*, *E*}.

Функция переходов автомата  представлена в таблице 3.2.

Таблица 3.2 - Функция переходов автомата 

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| *F* | *A* | *B* | *C* | *D* | *E* |
| *a* | *B* |  |  | *C* | *B* |
| *b* | *C* | *D* | *E* | *E* | *D* |

Граф КА  после устранения недостижимых состояний представлен на рисунке 3.2.

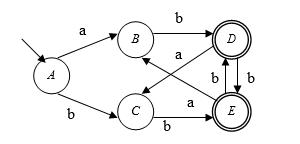


Рисунок 3.2 - Граф КА  после устранения недостижимых состояний

## Алгоритм 3.2. Объединение эквивалентных состояний КА

Вход: КА без недостижимых состояний.

Выход: минимальный КА .

Шаг 1. На первом шаге строим нулевое разбиение R(0), состоящее из двух классов эквивалентности: заключительные состояния КА - Z и не заключительные - Q-Z.

Шаг 2. На очередном шаге построения разбиения R(n) в классы эквивалентности включить те состояния, которые по одинаковым входным символам переходят в n-1 эквивалентные состояния, т.е.

.

Шаг 3. До тех пор, пока R(n)  R(n-1) полагаем n:=n+1 и идем к шагу 2.

Шаг 4. Переобозначить оставшиеся неразбитые группы состояний и включить их в таблицу новых обозначений состояний автомата.

Шаг 5. Определить эквивалентный КА в новых обозначениях.

Пример 3.2.Минимизировать конечный автомат из примера 3.1.

Последовательность построения разбиений будет иметь вид:

*R*(0) = {{*A*, *B*, *C*}, {*D*, *E*}}, *n* = 0;

*R*(1) = {{*A*}, {*B*, *C*}, {*D*, *E*}}, *n* = 1;

*R*(2) = {{*A*}, {*B*, *C*}, {*D*, *E*}}, *n*=2.

Т.к. *R*(1) = *R*(2), то искомое разбиение построено.

Переобозначим оставшиеся неразбитые группы состояний:

X={B, C}, Y={D, E}.

Получим минимальный автомат , где **={*A*, *X*, *Y*}, **={*Y*}.

Функция переходов автомата представлена в таблице 3.3.

Таблица 3.3 - Функция переходов автомата 

|  |  |  |  |
| --- | --- | --- | --- |
|  | *A* | *X* | *Y* |
| *a* | *X* |  | *X* |
| *b* | *X* | *Y* | *Y* |

Граф переходов конечного автомата после его минимизации показан на рисунке 3.3.

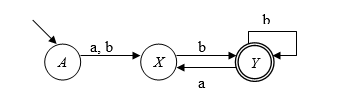


Рисунок 3.3 – Граф минимального КА 

# Реализация:

Программа написана на языке С++. Для хранения информации о переходах будем использовать 3 вектора. В первом хранится то, откуда был произведён переход, в 2 с помощью какого символа, в 3 куда. Например первый переход из Y в V с помощью 1, тогда в первый вектор запишем Y, в второй 1, в третий V. Если нам понадобиться доступ к этому переходу мы сможем обратиться по соответственному номеру. При запуске программы будет запрошен ввод ввода – ручной или из файла. Если выбран из файла, то будет произведён поиск необходимой информации, если её нет будет предложена создать файл. Если выбран ручной ввод то, будет запрошен список точек и символов перехода, для дальнейшего использования в проверках и выборе символов для новых точек при переходе к ДКА. Информация о точках и символах перехода так же храниться в двух соответствующих векторах. Позже будет произведён отбор достижимых точек путем поиска тех, в которых нет входа, после нахождения такой точки цикл будет начат заново, а все точки для которых есть вход(не учитывая начальную) будут записаны с отдельный вектор для дальнейшей работы. Далее по стандартному алгоритму будет произведена замена некоторых точек с целью минимизации. В итоге программа покажет все переходы минимизированного графа.

# Результат работы:

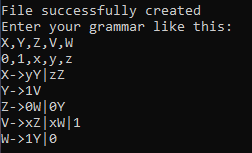
При запуске программы у пользователя будет спрошено, как он хочет ввести данные, если выбран первый режим работы, то ручной, если второй то из файла:



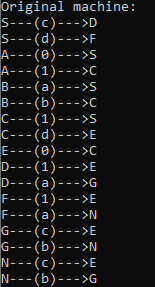
Если выбран режим ввода из файла, но файл не найден программа предложит его создать:



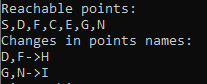
При успешном создании файла будет выведено соответствующее уведомление и показан пример ввода грамматики:



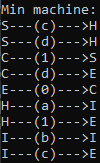
При загрузке из файла/ручном вводе грамматики будут выведен начальный список переходов графа:



После обработки данных будет выведен сначала список достижимых точек и всех замен, произведённых программой:



А после и все переходы для минимального графа:



# Вывод:

В данной лабораторной работе было реализована программа способная принять формальную грамматику как в ручном режиме, так и чтением из файла. Построение по этой грамматики графа, его минимизация и составление грамматики для данного графа. Проверка его работы используя заранее выбранную граматику.