МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное общеобразовательное учреждение высшего образования

«Ижевский государственный технический университет имени М.Т. Калашникова»

Кафедра «Программное обеспечение»

ОТЧЕТ

по индивидуальному заданию №8

по дисциплине «Теория автоматов и формальных языков»

Выполнил: студент группы Б04-191-3 Р.А. Гумметов

Принял: д.т.н., профессор М.А. Сенилов

Ижевск 2020

СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 3](#_Toc41391264)

[1. ПОСТАНОВКА ЗАДАЧИ 4](#_Toc41391265)

[2. ИНДИВИДУАЛЬНОЕ ЗАДАНИЕ. ПОСТРОЕНИЕ ПРАВОЛИНЕЙНОЙ ГРАММАТИКИ 6](#_Toc41391266)

[3. ПОСТРОЕНИЕ АВТОМАТНОЙ ГРАММАТИКИ ПО ПРАВОЛИНЕЙНОЙ 9](#_Toc41391267)

[4. ПОСТРОЕНИЕ НЕДЕТЕРМИНИРОВАННОГО КОНЕЧНОГО АВТОМАТА 10](#_Toc41391268)

[5. СВЕДЕНИЕ НЕДЕТЕРМИНИРОВАННОГО КОНЕЧНОГО АВТОМАТА К ДЕТЕРМИНИРОВАННОМУ 12](#_Toc41391269)

[6. ПОСТРОЕНИЕ МИНИМАЛЬНОГО АВТОМАТА 16](#_Toc41391270)

[7. ОПИСАНИЕ ПРОГРАММЫ, РЕАЛИЗУЮЩЕЙ РАСПОЗНАЮЩИЙ АВТОМАТ 19](#_Toc41391271)

[7.1 Вводная часть 19](#_Toc41391272)

[7.2 Функциональное назначение 19](#_Toc41391273)

[7.3 Описание информации 19](#_Toc41391274)

[7.4 Описание логики 20](#_Toc41391275)

[8. ОПИСАНИЕ КОНТРОЛЬНОГО ПРИМЕРА 22](#_Toc41391276)

[8.1 Назначение 22](#_Toc41391277)

[8.2 Исходные данные 22](#_Toc41391278)

[8.3 Результаты расчета 22](#_Toc41391279)

[8.4 Результаты испытания программы 24](#_Toc41391280)

[ЗАКЛЮЧЕНИЕ 25](#_Toc41391281)

[СПИСОК ЛИТЕРАТУРЫ 26](#_Toc41391282)

[ПРИЛОЖЕНИЯ 27](#_Toc41391283)

[Текст программы 27](#_Toc41391284)

[Результаты расчета на ЭВМ контрольного примера 28](#_Toc41391285)

ВВЕДЕНИЕ

Цель курсовой работы состоит в изучении способов задания языков грамматиками, распознающими автоматами, построении модели конечного автомата, распознающего заданный язык и его программная реализация.

В наше время, конечные автоматы имеют широкое распространение в компиляторах языков, поэтому программная реализация конечного автомата приобретает высокое значение. Также они применяются для создания лингвистических процессоров, для описания и обработки формальных языков.

Каждый автомат имеет конечное число входов, воспринимающих информацию, изображаемую конечным числом символов из некоторого алфавита, и конечное число выходов для выдачи информации.

Выходная информация автомата зависит не только от входной информации, но и от внутреннего состояния автомата. Конечный автомат имеет конечное число состояний.

Автоматы часто представляют сетями. Для автомата характерен последовательный способ функционирования: автомат последовательно переходит из состояния в состояние с заданной функцией перехода и осуществляет очередной шаг.

1. ПОСТАНОВКА ЗАДАЧИ

Необходимо построить праволинейную грамматику на основе индивидуального задания и приведенного ниже определения формальной грамматики. Затем по праволинейной грамматике построить автоматную грамматику. Построить недетерминированный конечный автомат по полученной автоматной грамматике. Преобразовать недетерминированный конечный автомат в детерминированный. Минимизировать полученный автомат, построить таблицу и граф переходов минимального автомата.

Входными данными для автомата является цепочка (строки, вводимые с клавиатуры) из терминальных символов. На выходе автомата выдается состояние - отвергающее или допускающее входную цепочку.

Задана формальная грамматика G = <Vt, Vn, S, P>, где

Vt = {C1, C2,…, C18} – терминальный словарь,

Vn = {S, A, B, C, D, E, F}- нетерминальный словарь,

S - начальный символ грамматики, S Vn,

P - множество правил вывода

Правила вывода имеют следующий вид:

S → C1 C2 C3 A;

S → C1 C4 C5 B;

S → C6 C;

S → C7 F;

A → C8 D;

A → C9;

B → C8 E;

B → C9;

C → C8 E;

C → C9;

D → C10 S;

D → C11;

E → C10 S;

E → C11;

F → C12 C13 C14 C15;

F → C16 C13 C14 C15;

F → C17 C18 C15.

2. ИНДИВИДУАЛЬНОЕ ЗАДАНИЕ. ПОСТРОЕНИЕ ПРАВОЛИНЕЙНОЙ ГРАММАТИКИ

Индивидуальным заданием являются две таблицы (см. табл., 1,2) и правила вывода R. Необходимо поставить в соответствие терминальным символам Ci грамматики G терминальные символы Xi грамматики G’. Для этого во вторую строку таблицы 1 записываются первые 18 символов фамилии, имени и отчества с обязательными пробелами между ними. В третью строку необходимо занести для каждого из 18 символов строки символы из алфавита {X0, X1, X2, X3, X4, X5, X6, X7} в соответствии с таблицей 2.

Таблица 1. Индивидуальная таблица

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| ci | c1 | c2 | c3 | c4 | c5 | c6 | c7 | c8 | c9 | c10 | c11 | c12 | c13 | c14 | c15 | c16 | c17 | c18 |
| si | Г | У | М | М | Е | Т | О | В | \_ | Р | У | С | Т | А | М | \_ | А | Л |
| xi | X4 | X7 | X3 | X3 | X6 | X5 | X4 | x2 | x5 | X0 | X7 | X4 | x5 | X1 | x3 | x5 | X1 | x7 |

Таблица 2. Таблица соответствия между буквами алфавита и терминальными символами грамматики G’

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| А | Б | В | Г | Д | Е | Ж | З | И | Й | К | Л | М | Н | О | П |
| x1 | x5 | x2 | x4 | x6 | x6 | x4 | x3 | x3 | x0 | x7 | x0 | x3 | x7 | x4 | x5 |
| P | С | Т | У | Ф | Х | Ц | Ч | Ш | Щ | Ь | Ы | Э | Ю | Я | \_ |
| x0 | x4 | x5 | x7 | x2 | x5 | x1 | x2 | x2 | x0 | x6 | x1 | x1 | x3 | x7 | x5 |

Наконец, задана формальная грамматика: G=<Vt, Vn, S, R>, где Vt={c1, c2, c3, … , c18} – терминальный словарь; Vn={S, A, B, C, D, E, F} – нетерминальный словарь; S∈Vn – начальный символ грамматики; R – множество правил вывода, которые имеют следующий вид:

1) S→x4 x7 x3 A;

2) S → x4 x3 x6 B;

3) S → x5;

4) S → x7 F;

5) A → x2 D;

6) A → x5;

7) B → x2 E;

8) B → x5;

9) C → x2 E;

10) C → x5;

11) D → x0 S;

12) D → x7;

13) E → x0 S;

14) E → x7;

15) F → x4 x5 x1 x3;

16) F → x5 x5 x1 x3;

17) F → x1 x7 x3;

G'=<V't, V'n, S, R'>, где V't={x0, …, x7} – новый терминальный словарь; V'n=Vn; R' – множество правил вывода, получаемых из заданных заменой символов из алфавита Vt символами из алфавита V't в соответствии с таблицей 1. В данном примере они имеют вид:

S→x4 x7 x3 A | x4 x3 x6 B | x5 C | s7 F;

A → x2 D | x5;

B → x2 E | x5;

C → x2 E | x5;

D → x0 S | x7;

E → x0 S | x7;

F → x4 x5 x1 x3 | x5 x5 x1 x3 | x1 x7 x3;

Граф полученной грамматики представлен на рис. 1.

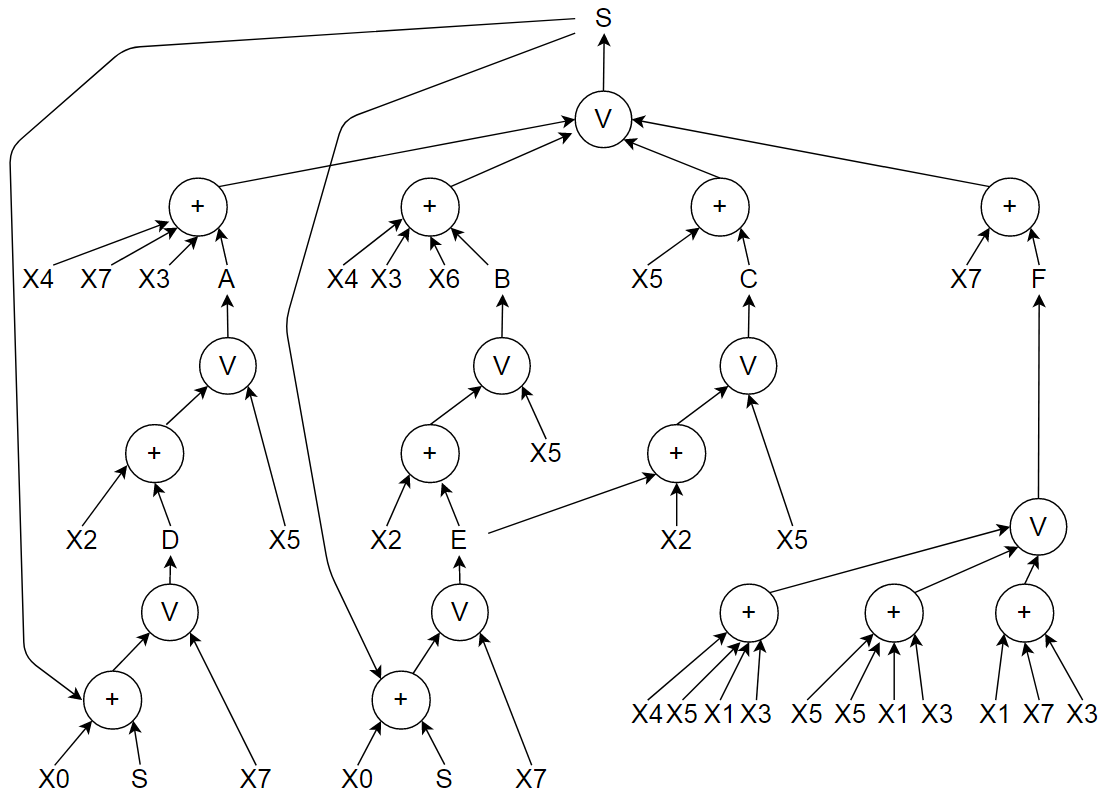


Рис. 1. Граф-грамматики

3. ПОСТРОЕНИЕ АВТОМАТНОЙ ГРАММАТИКИ ПО ПРАВОЛИНЕЙНОЙ

Этот этап выполняется путем расширения нетерминального словаря способом, вытекающим из возможности преобразования праволинейной грамматики в модифицированную автоматную G'' = <V't, V''n, S, R''>

Получим множество R'' правил вывода:

S→x4 S1; S1→x7 S2; S2→x3 A;

S→x4 S3; S3→x3 S4; S4→x6 B;

S→x5 C;

S→x7 F;

A→x2 D; A→x5 A1; A1→ε;

B→x2 E; B→x5 B1; B1→ε;

C→x2 E; C→x5 C1; C1→ε;

D→x0 S; D→x7 D1; D1→ε;

E→x0 S; E→x7 E1; E1→ ε;

F→x4 F1; F1→x5 F2; F2→x1 F3; F3→x3 F4; F4→ε;

F→x5 F5; F5→x5 F6; F5→x1 F7; F7→x3 F8; F8→ε;

F→x1 F9; F9→x7 F10; F10→x3 F11; F11→ε

Таким образом, нетерминальный словарь теперь имеет вид V''n = {S, S1, S2, S3, S4, A, A1, B, B1, C, C1, D, D1, E, E1, F, F1, F2, F3, F4, F5, F6, F7, F8, F9, F10, F11} и его мощность |V''n| равна 27.

4. ПОСТРОЕНИЕ НЕДЕТЕРМИНИРОВАННОГО КОНЕЧНОГО АВТОМАТА

Построим на основе грамматики G'' конечный автомат. При этом нетерминальным символам грамматики V''n поставим в соответствие состояния автомата (оставим для них те же обозначения). Начальному нетерминалу S поставим в соответствие начальное состояние автомата S. Правилам вывода поставим в соответствие переходы автомата.

В результате получим таблицу 3 – таблицу переходов недетерминированного конечного автомата.

Таблица 3. Таблица переходов недетерминированного частичного автомата

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | X0 | X1 | X2 | X3 | X4 | X5 | X6 | X7 |  |
| S | Er | Er | F | Er | S1,S3 | C | Er | F | 0 |
| S1 | Er | Er | Er | Er | Er | Er | Er | S2 | 0 |
| S2 | Er | Er | Er | A | Er | Er | Er | Er | 0 |
| S3 | Er | Er | Er | S4 | Er | Er | Er | Er | 0 |
| S4 | Er | Er | Er | Er | Er | Er | B | Er | 0 |
| A | Er | Er | D | Er | Er | A1 | Er | Er | 0 |
| A1 | Er | Er | Er | Er | Er | Er | Er | Er | 1 |
| B | Er | Er | E | Er | Er | B1 | Er | Er | 0 |
| B1 | Er | Er | Er | Er | Er | Er | Er | Er | 1 |
| C | Er | Er | E | Er | Er | C1 | Er | Er | 0 |
| C1 | Er | Er | Er | Er | Er | Er | Er | Er | 1 |
| D | S | Er | Er | Er | Er | Er | Er | D1 | 0 |
| D1 | Er | Er | Er | Er | Er | Er | Er | Er | 1 |
| E | S | Er | Er | Er | Er | Er | Er | E1 | 0 |
| E1 | Er | Er | Er | Er | Er | Er | Er | Er | 1 |
| F | Er | F9 | Er | Er | F1 | F5 | Er | Er | 0 |
| F1 | Er | Er | Er | Er | Er | F2 | Er | Er | 0 |
| F2 | Er | F3 | Er | Er | Er | Er | Er | Er | 0 |
| F3 | Er | Er | Er | F4 | Er | Er | Er | Er | 0 |
| F4 | Er | Er | Er | Er | Er | Er | Er | Er | 1 |
| F5 | Er | Er | Er | Er | Er | F6 | Er | Er | 0 |
| F6 | Er | F7 | Er | Er | Er | Er | Er | Er | 0 |
| F7 | Er | Er | Er | F8 | Er | Er | Er | Er | 0 |
| F8 | Er | Er | Er | Er | Er | Er | Er | Er | 1 |
| F9 | Er | Er | Er | Er | Er | Er | Er | F10 | 0 |
| F10 | Er | Er | Er | F11 | Er | Er | Er | Er | 0 |
| F11 | Er | Er | Er | Er | Er | Er | Er | Er | 1 |

Граф переходов автомата, построенный по таблице 3, показан на рис. 2.

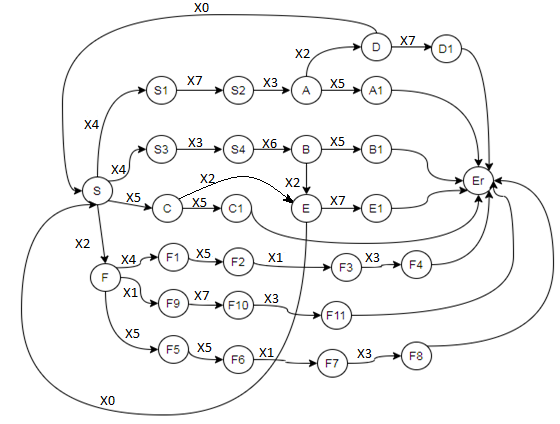


Рис. 2. Граф переходов исходного (недетерминированного) автомата

5. СВЕДЕНИЕ НЕДЕТЕРМИНИРОВАННОГО КОНЕЧНОГО АВТОМАТА К ДЕТЕРМИНИРОВАННОМУ

Процедура построения детерминированного автомата Ад, эквивалентного недетерминированному автомату Ан, задается следующими шагами:

1. Пометить первую строку таблицы переходов для Ад множеством начальных состояний автомата Ан. Применить к этому множеству шаг 2.

2. По данному множеству состояний В, помечающему строку таблицы переходов автомата Ад, для которой переходы еще не вычислены, вычислить те состояния автомата Ан, которые могут быть достигнуты из В с помощью каждого входного символа х, и поместить полученные множества состояний в соответствующие ячейки таблицы для автомата Ад. Символически это выражается так: если d – функция недетерминированных переходов, то функция детерминированных перехода d' задается формулой d'(B, x) = {S | S Î d (T, x), " Т Î В}

3. Для каждого нового множества, порожденного переходами на шаге 2, посмотреть, имеется ли уже в Ад строка, помеченная этим множеством. Если нет, то создать новую строку и пометить ее этим множеством. Если множество уже использовалось как метка, никаких действий не требуется.

4. Если в таблице автомата Ад есть строка, для которой еще не вычислены переходы, вернуться назад и применить к этой строке шаг 2. Если все переходы вычислены, перейти к шагу 5.

5. Пометить строку как допускающее состояние автомата Ад тогда и только тогда, когда она содержит допускающее состояние недетерминированного автомата. В противном случае пометить как отвергающее состояние.

Описанная процедура гарантирует, что детерминированный автомат не содержит недостижимых состояний.

В результате применения этого алгоритма от недетерминированного автомата (табл. 3, рис. 2) можно перейти к эквивалентному детерминированному автомату, таблица переходов которого приведена в таблице 4.

Таблица 4. Таблица переходов детерминированного автомата

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | X0 | X1 | X2 | X3 | X4 | X5 | X6 | X7 |  |
| {S} | {Er} | {Er} | {Er} | {Er} | {S1, S3} | {C} | {Er} | {F} | 0 |
| {S1, S3} | {Er} | {Er} | {Er} | {Er} | {Er} | {Er} | {Er} | {S2, S4} | 0 |
| {S2, S4} | {Er} | {Er} | {Er} | {A} | {Er} | {Er} | {B} | {Er} | 0 |
| {A} | {Er} | {Er} | {D} | {Er} | {Er} | {A1} | {Er} | {Er} | 0 |
| {A1} | {Er} | {Er} | {Er} | {Er} | {Er} | {Er} | {Er} | {Er} | 1 |
| {B} | {Er} | {Er} | {E} | {Er} | {Er} | {B1} | {Er} | {Er} | 0 |
| {B1} | {Er} | {Er} | {Er} | {Er} | {Er} | {Er} | {Er} | {Er} | 1 |
| {C} | {Er} | {Er} | {E} | {Er} | {Er} | {C1} | {Er} | {Er} | 0 |
| {C1} | {Er} | {Er} | {Er} | {Er} | {Er} | {Er} | {Er} | {Er} | 1 |
| {D} | {S} | {Er} | {Er} | {Er} | {Er} | {Er} | {Er} | {D1} | 0 |
| {D1} | {Er} | {Er} | {Er} | {Er} | {Er} | {Er} | {Er} | {Er} | 1 |
| {E} | {S} | {Er} | {Er} | {Er} | {Er} | {Er} | {Er} | {E1} | 0 |
| {E1} | {Er} | {Er} | {Er} | {Er} | {Er} | {Er} | {Er} | {Er} | 1 |
| {F} | {Er} | {F9} | {Er} | {Er} | {F1} | {F5} | {Er} | {Er} | 0 |
| {F1} | {Er} | {Er} | {Er} | {Er} | {Er} | {F2} | {Er} | {Er} | 0 |
| {F2} | {Er} | {F3} | {Er} | {Er} | {Er} | {Er} | {Er} | {Er} | 0 |
| {F3} | {Er} | {Er} | {Er} | {F4} | {Er} | {Er} | {Er} | {Er} | 0 |
| {F4} | {Er} | {Er} | {Er} | {Er} | {Er} | {Er} | {Er} | {Er} | 1 |
| {F5} | {Er} | {Er} | {Er} | {Er} | {Er} | {F6} | {Er} | {Er} | 0 |
| {F6} | {Er} | {F7} | {Er} | {Er} | {Er} | {Er} | {Er} | {Er} | 0 |
| {F7} | {Er} | {Er} | {Er} | {F8} | {Er} | {Er} | {Er} | {Er} | 0 |
| {F8} | {Er} | {Er} | {Er} | {Er} | {Er} | {Er} | {Er} | {Er} | 1 |
| {F9} | {Er} | {Er} | {Er} | {Er} | {Er} | {Er} | {Er} | {F10} | 0 |
| {F10} | {Er} | {Er} | {Er} | {F11} | {Er} | {Er} | {Er} | {Er} | 0 |
| {F11} | {Er} | {Er} | {Er} | {Er} | {Er} | {Er} | {Er} | {Er} | 1 |
| {Er} | {Er} | {Er} | {Er} | {Er} | {Er} | {Er} | {Er} | {Er} | 0 |

Перейдем к более простым обозначениям состояний автомата:

{S}→Y; {S1, S3}→Y1; {S2, S4}→ Y2;

{A}→Y3; {A1}→Y4; {B}→Y5; {B1}→Y6;

{C}→Y7; {C1}→Y8; {D}→Y9; {D1}→Y10;

{E}→Y11; {E1}→Y12; {F}→Y13; {F1}→Y14;

{F2}→Y15; {F3}→Y16; {F4}→Y17; {F5}→Y18;

{F6}→Y19; {F7}→Y20; {F8}→Y21; {F9}→Y22;

{F10}→Y23; {F11}→Y24; {Er}→Er

В новых обозначениях таблица переходов автомата приведена в табл. 5.

Таблица 5. Таблица переходов детерминированного автомата (новые обозначения состояний)

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | X0 | X1 | X2 | X3 | X4 | X5 | X6 | X7 |  |
| Y | Er | Er | Er | Er | Y1 | Y7 | Er | Y13 | 0 |
| Y1 | Er | Er | Er | Er | Er | Er | Er | Y2 | 0 |
| Y2 | Er | Er | Er | Y3 | Er | Er | Y5 | Er | 0 |
| Y3 | Er | Er | Y9 | Er | Er | Y4 | Er | Er | 0 |
| Y4 | Er | Er | Er | Er | Er | Er | Er | Er | 1 |
| Y5 | Er | Er | Y11 | Er | Er | Y6 | Er | Er | 0 |
| Y6 | Er | Er | Er | Er | Er | Er | Er | Er | 1 |
| Y7 | Er | Er | Y11 | Er | Er | Y8 | Er | Er | 0 |
| Y8 | Er | Er | Er | Er | Er | Er | Er | Er | 1 |
| Y9 | Y | Er | Er | Er | Er | Er | Er | Y10 | 0 |
| Y10 | Er | Er | Er | Er | Er | Er | Er | Er | 1 |
| Y11 | Y | Er | Er | Er | Er | Er | Er | Y12 | 0 |
| Y12 | Er | Er | Er | Er | Er | Er | Er | Er | 1 |
| Y13 | Er | Y22 | Er | Er | Y14 | Y18 | Er | Er | 0 |
| Y14 | Er | Er | Er | Er | Er | Y15 | Er | Er | 0 |
| Y15 | Er | Y16 | Er | Er | Er | Er | Er | Er | 0 |
| Y16 | Er | Er | Er | Y17 | Er | Er | Er | Er | 0 |
| Y17 | Er | Er | Er | Er | Er | Er | Er | Er | 1 |
| Y18 | Er | Er | Er | Er | Er | Y19 | Er | Er | 0 |
| Y19 | Er | Y20 | Er | Er | Er | Er | Er | Er | 0 |
| Y20 | Er | Er | Er | Y21 | Er | Er | Er | Er | 0 |
| Y21 | Er | Er | Er | Er | Er | Er | Er | Er | 1 |
| Y22 | Er | Er | Er | Er | Er | Er | Er | Y23 | 0 |
| Y23 | Er | Er | Er | Y24 | Er | Er | Er | Er | 0 |
| Y24 | Er | Er | Er | Er | Er | Er | Er | Er | 1 |
| Er | Er | Er | Er | Er | Er | Er | Er | Er | 0 |

Граф переходов автомата, построенный по таблице 5, показан на рис. 3.

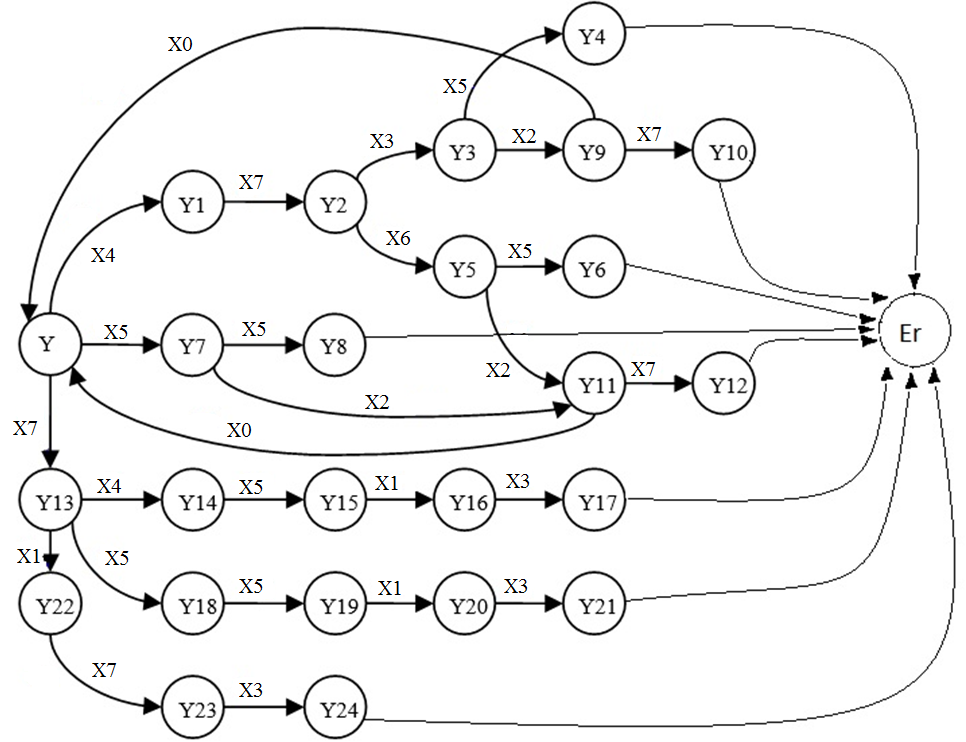


Рис. 3. Граф переходов детерминированного автомата, эквивалентного исходному

# 6. ПОСТРОЕНИЕ МИНИМАЛЬНОГО АВТОМАТА

Произвольный конечный автомат можно превратить в эквивалентный ему минимальный, выбрасывая недостижимые и объединяя эквивалентные состояния. Разбиение состояний на классы эквивалентности можно осуществить с помощью метода разбиения.

Метод разбиения заключается в разбиении множества состояний на непересекающиеся подмножества или блоки, такие, что неэквивалентные состояния попадают в разные блоки.

Начальное разбиение P0 заключается в разбиении всего множества состояний на подмножества допустимых и недопустимых состояний:

Р0=({Y, Y1, Y2, Y3, Y5, Y7, Y9, Y11, Y13, Y14, Y15, Y16, Y18, Y19, Y20, Y22, Y23, Er}, {Y4, Y6, Y8, Y10, Y12, Y17, Y21, Y24})

Далее произведем разбиение блока 1 по входу X5 и получим разбиение

P1=({Y, Y1, Y2, Y9, Y11, Y13, Y14, Y15, Y16, Y18, Y19, Y20, Y22, Y23, Er}, {}, {Y3, Y5, Y7}, {Y4, Y6, Y8, Y10, Y12, Y17, Y21, Y24})

Произведем разбиение блока 1 по входу X0, получим разбиение

P2 = ({Y, Y1, Y2, Y13, Y14, Y15, Y16, Y18, Y19, Y20, Y22, Y23, Er}, { Y9, Y11}, {Y3, Y5, Y7}, {Y4, Y6, Y8, Y10, Y12, Y17, Y21, Y24})

Произведем разбиение блока 1 по входу X1, получим разбиение

P3 = ({Y, Y1, Y2, Y14, Y16, Y18, Y20, Y22, Y23, Er}, {Y13}, {Y15, Y19}, { Y9, Y11}, {Y3, Y5, Y7}, {Y4, Y6, Y8, Y10, Y12, Y17, Y21, Y24})

Произведем разбиение блока 1 по входу X3, получим разбиение

P4 = ({Y, Y1, Y14, Y18, Y22, Er}, {Y2}, {Y16, Y20, Y23}, {Y13}, {Y15, Y19}, { Y9, Y11}, {Y3, Y5, Y7}, {Y4, Y6, Y8, Y10, Y12, Y17, Y21, Y24})

Произведем разбиение блока 1 по входу X4, получим разбиение

P5 = ({Y1, Y14, Y18, Y22, Er}, {Y2}, {Y}, {Y16, Y20, Y23}, {Y13}, {Y15, Y19}, { Y9, Y11}, {Y3, Y5, Y7}, {Y4, Y6, Y8, Y10, Y12, Y17, Y21, Y24})

Произведем разбиение блока 1 по входу X7, получим разбиение

P6 = ({Y14, Y18, Er}, {Y1}, {Y22}, {Y2}, {Y}, {Y16, Y20, Y23}, {Y13}, {Y15, Y19}, { Y9, Y11}, {Y3, Y5, Y7}, {Y4, Y6, Y8, Y10, Y12, Y17, Y21, Y24})

Произведем разбиение блока 1 по входу X5, получим разбиение

P7 = ({Er}, {Y14, Y18}, {Y1}, {Y22}, {Y2}, {Y}, {Y16, Y20, Y23}, {Y13}, {Y15, Y19}, { Y9, Y11}, {Y3, Y5, Y7}, {Y4, Y6, Y8, Y10, Y12, Y17, Y21, Y24})

Множество Р8 не допускает дальнейшего разбиения ни по одному входу, оно содержит подмножества (блоки) эквивалентных состояний, которые и являются состояниями минимального автомата.

Введем обозначения для этих подмножеств – состояний минимального автомата (табл. 6).

Таблица 6. Состояния минимального автомата

|  |  |
| --- | --- |
| Блок эквивалентных состояний | Состояние минимального автомата |
| {Y} | 1 |
| {Y1} | 2 |
| {Y2} | 3 |
| {Y3, Y5, Y7} | 4 |
| { Y9, Y11} | 5 |
| {Y13} | 6 |
| {Y14, Y18} | 7 |
| {Y15, Y19} | 8 |
| {Y16, Y20, Y23} | 9 |
| {Y22} | 10 |
| {Y4, Y6, Y8, Y10, Y12, Y17, Y21, Y24} | 11 |
| Er | Er |

Таблица переходов минимального автомата показана в табл. 7.

Таблица 7. Таблица переходов минимального автомата

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | X0 | X1 | X2 | X3 | X4 | X5 | X6 | X7 |  |
| 1 | Er | Er | Er | Er | 2 | 4 | Er | 6 | 0 |
| 2 | Er | Er | Er | Er | Er | Er | Er | 3 | 0 |
| 3 | Er | Er | Er | 4 | Er | Er | 4 | Er | 0 |
| 4 | Er | Er | 5 | Er | Er | 11 | Er | Er | 0 |
| 5 | 1 | Er | Er | Er | Er | Er | Er | 11 | 0 |
| 6 | Er | 10 | Er | Er | 7 | 7 | Er | Er | 0 |
| 7 | Er | Er | Er | Er | Er | 8 | Er | Er | 0 |
| 8 | Er | 9 | Er | Er | Er | Er | Er | Er | 0 |
| 9 | Er | Er | Er | 11 | Er | Er | Er | Er | 0 |
| 10 | Er | Er | Er | Er | Er | Er | Er | 9 | 0 |
| 11 | Er | Er | Er | Er | Er | Er | Er | Er | 1 |
| Er | Er | Er | Er | Er | Er | Er | Er | Er | 0 |

Граф переходов минимального автомата, построенный по табл. 7, показан на рис. 4.

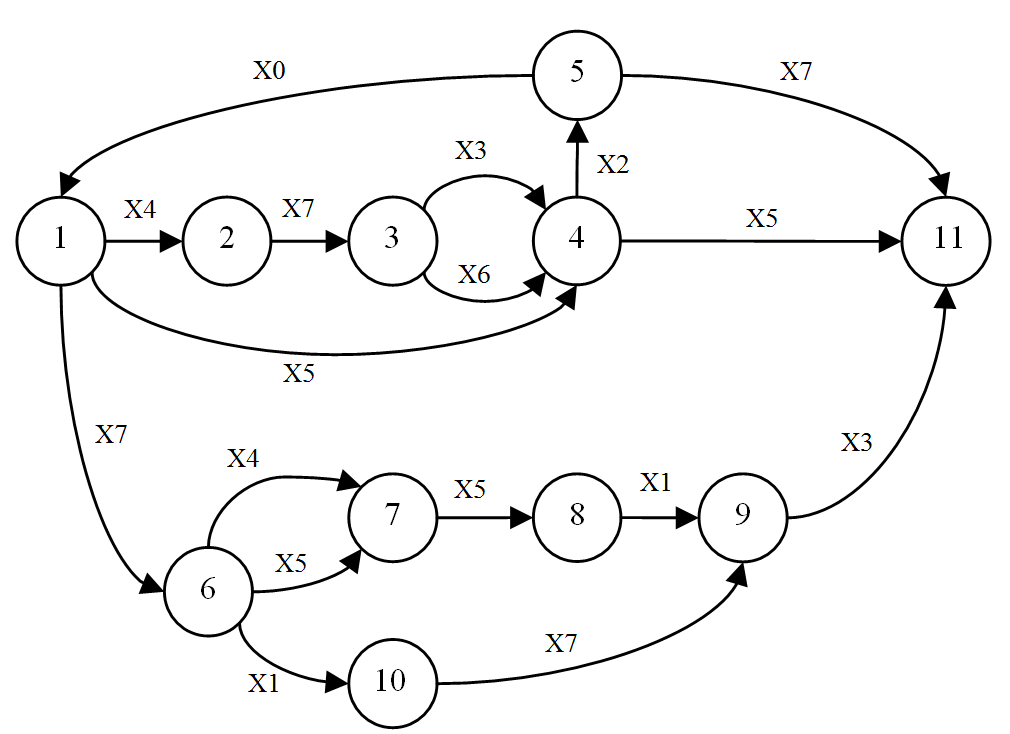


Рис. 4. Граф переходов минимального распознающего автомата

7. ОПИСАНИЕ ПРОГРАММЫ, РЕАЛИЗУЮЩЕЙ РАСПОЗНАЮЩИЙ АВТОМАТ

## 7.1 Вводная часть

Для проверки правильности построенного конечного распознавателя, написана программа. Программа реализует работу распознающего автомата и производит распознавание вводимых с клавиатуры цепочек. Программа написана на языкеC#.

## 7.2 Функциональное назначение

Программа имитирует работу конечного автомата. Программа применяется для распознавания входных цепочек символов праволинейной грамматики.

Для функционирования программы необходима любая ЭВМ, имеющая

транслятор языка C#.

Для работы программы требуются устройство ввода(клавиатура) и устройство вывода(дисплей)

## 7.3 Описание информации

В качестве входной информации выступают строки, вводимые с клавиатуры, состоящие из символов исходной грамматики и являющиеся строкой для распознавания. Информация о допустимости цепочек выводится на дисплей. Входные данные имеют формат: хАхВхС , где А, В, С - числа от 1 до 7. Если цепочка допустима, то на экране появится сообщение «Цепочка допустима». Если цепочка недопустима, то на экране появится сообщение «Цепочка не допустима».

## 7.4 Описание логики

Логику написанной программы иллюстрирует схема программы, представленная на рис. 5, рис. 6 и рис. 7.

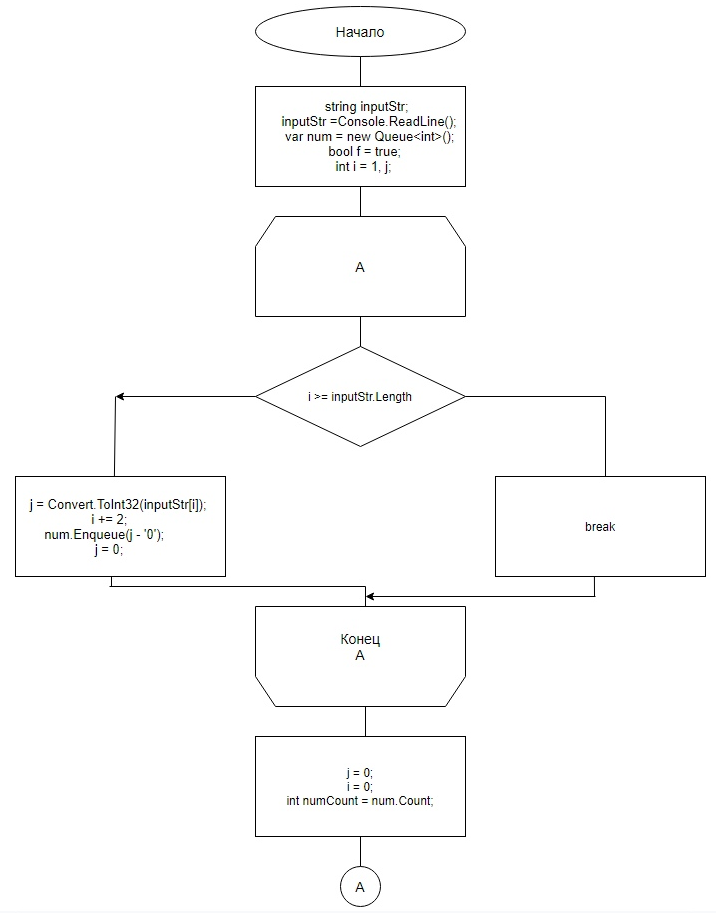


Рис. 5. Схема программы

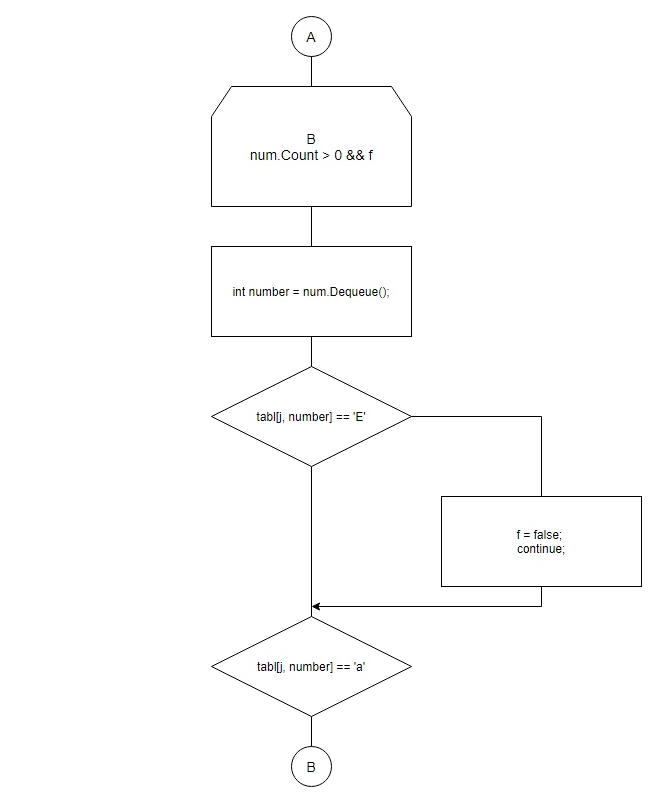


Рис. 6. Схема программы (продолжение)

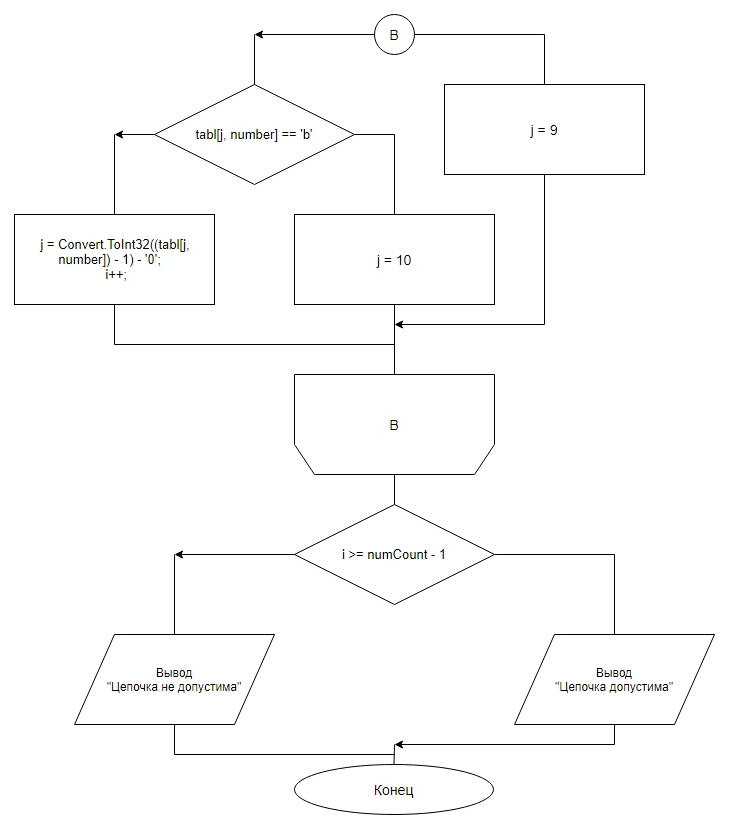


Рис. 7. Схема программы (продолжение)

# 8. ОПИСАНИЕ КОНТРОЛЬНОГО ПРИМЕРА

## 8.1 Назначение

Контрольный пример предназначен для тестирования программы, реализующей конечный автомат.

## 8.2 Исходные данные

Исходные данные - цепочка символов. В нее входят символы из множества: {x1,x2,x3,x4,x5,x6,x7}. Построим цепочки символов, для контрольного примера, исходя из право-линейной грамматики. Для проверки правильности работы автомата нужно проверить его с помощью допустимых цепочек. Что бы получить допустимую цепочку символов необходимо взять одно из правил, в левой части которого стоит начальный символ S. Выписать все терминальные символы из этого правила и если в конце стоит нетерминал, то перейти к одному из правил, в левой части которого стоит этот нетерминал. Выписать терминальные символы из этого правила и если в конце стоит нетерминал, то перейти к новому правилу и т.д., пока мы не дойдем до правила, правая часть которого кончается терминалом.

8.3 Результаты расчета

Итак, получаем допустимые цепочки:

1) S⇒x4x7x3A⇒x4x7x3x5

↑1 ↑6

Отсюда получаем цепочку: x4x7x3x5

2) S⇒x7F⇒x7x4x5x1x3

↑4 ↑15

Отсюда получаем цепочку: x7x4x5x1x3

3) S⇒x7F⇒x7x1x7x3

↑4  ↑17

Отсюда получаем цепочку: x7x1x7x3

Для полной проверки автомата получим несколько недопустимых цепочек. Их можно получить, если выписывать терминалы, не доходя до терминала, который стоит последним в правиле.

Или же если записать терминал, которого нет в правой части ни одного из правил, в левой части которых стоит необходимый нетерминал.

Недопустимые цепочки:

4) x5x2x1

5) x4x7x3x0

6) x4x7x5

Проверим полученные цепочки на минимальном автомате, построив соответствующие последовательности переходов автомата:

1) x4x7x3x5

1-2-3-4-11

11 - допускающее состояние, значит, цепочка допускается

2) x7x4x5x1x3

1-6-7-8-9-11

11- допускающее состояние, значит, цепочка допускается

3) x7x1x7x3

1-6-10-9-11

11 - допускающее состояние, значит, цепочка допускается

4) x5x2x1

1-4-5-Er

Er - отвергающее состояние, значит, цепочка отвергается

5) x4x7x3x0

1-2-3-4-Er

Er - отвергающее состояние, значит, цепочка отвергается

6) x4x7x5

1-2-3-Er

Er - отвергающее состояние, значит, цепочка отвергается

## 8.4 Результаты испытания программы

Результаты испытания программы представлены в приложении 2. Результаты совпадают с рассчитанными, отсюда следует, что программа, реализующая работу конечного автомата, работает правильно.

Таблица 8. Сравнение результатов работы программы с результатами работы автомата

|  |  |  |  |
| --- | --- | --- | --- |
| № | Входная цепочка | Результат работы программы | Результат работы автомата |
| 1 | x4x7x3x5 | Цепочка допускается | Цепочка допускается |
| 2 | x7x4x5x1x3 | Цепочка допускается | Цепочка допускается |
| 3 | x7x1x7x3 | Цепочка допускается | Цепочка допускается |
| 4 | x5x2x1 | Цепочка не допускается | Цепочка не допускается |
| 5 | x4x7x3x0 | Цепочка не допускается | Цепочка не допускается |
| 6 | x4x7x5 | Цепочка не допускается | Цепочка не допускается |

# ЗАКЛЮЧЕНИЕ

В ходе выполнения индивидуального задания была построена праволинейная грамматика и ее граф. В дальнейшем по ней была построена автоматная грамматика, из которой в свою очередь был построен недетерминированный конечный автомат. Недетерминированный конечный автомат был сведен к эквивалентному детерминированному. Я произвел минимизацию детерминированного автомата методом разбиения.

Грамматики и автоматы сопровождены графическими изображениями. Результаты минимизации автоматов с помощью теории автоматов совпали с результатами программы реализующей распознающий автомат. Результаты программной реализации удовлетворительны.

# СПИСОК ЛИТЕРАТУРЫ

1. Методические указания для самостоятельной работы студентов по дисциплине" Теория вычислительных процессов и структур ". Ч1/ Ижевск. гос.техн.университет; Сост. Сенилов М.А. ИжГТУ, 2000.

# ПРИЛОЖЕНИЯ

Приложение 1

## Текст программы

using System;

using System.Collections.Generic;

class Program

{

static void Main()

{

var tabl = new char[,]

{

{'E', 'E', 'E', 'E', '2', '4', 'E', '6' },

{'E', 'E', 'E', 'E', 'E', 'E', 'E', '3' },

{'E', 'E', 'E', '4', 'E', 'E', '4', 'E' },

{'E', 'E', '5', 'E', 'E', 'b', 'E', 'E' },

{'1', 'E', 'E', 'E', 'E', 'E', 'E', 'b' },

{'E', 'a', 'E', 'E', '7', '7', 'E', 'E' },

{'E','E','E','E','E', '8', 'E','E' },

{'E', '9', 'E','E','E','E','E','E' },

{'E','E','E', 'b', 'E','E','E','E' },

{'E','E','E','E','E','E','E', '9' },

{'E','E','E','E','E','E','E','E' },

{'E','E','E','E','E','E','E','E' }

};

string inputStr;

inputStr = Console.ReadLine();

var num = new Queue<int>();

bool f = true;

int i = 1, j;

while (true)

{

if (i >= inputStr.Length) break;

j = Convert.ToInt32(inputStr[i]);

i += 2;

num.Enqueue(j - '0');

j = 0;

}

j = 0;

i = 0;

int numCount = num.Count;

while (num.Count > 0 && f)

{

int number = num.Dequeue();

if (tabl[j, number] == 'E') { f = false; continue; }

if (tabl[j, number] == 'a') j = 9;

else if (tabl[j, number] == 'b') j = 10;

else j = Convert.ToInt32((tabl[j, number]) - 1) - '0';

i++;

}

if (i >= numCount - 1) Console.WriteLine("Цепочка допустима");

else Console.WriteLine("Цепочка не допустима");

}

}

Приложение 2

## Результаты расчета на ЭВМ контрольного примера

Результаты испытания программы представлены на рис.4-9.

1.png

Рис. 1. Контрольный пример 1

2.png

Рис.2. Контрольный пример 2

3.png

Рис.3. Контрольный пример 3

4.png

Рис.4. Контрольный пример 4

5.png

Рис.5. Контрольный пример 5

6.png

Рис.6. Контрольный пример 6