**ОГЛАВЛЕНИЕ**

[ПЕРЕЧЕНЬ УСЛОВНЫХ ОБОЗНАЧЕНИЙ 2](#_Toc479722877)

[РЕФЕРАТ ДИПЛОМНОЙ РАБОТЫ](#_Toc479722877) 3

[РЭФЕРАТ ДЫПЛОМНАЙ РАБОТЫ 4](#_Toc479722877)

[DIPLOMA WORK ABSTRACT 5](#_Toc479722877)

[ВВЕДЕНИЕ 6](#_Toc479722877)

ГЛАВА [1. ЭЛЕМЕНТЫ ТЕОРИИ АБСТРАКТНЫХ АВТОМАТОВ 7](#_Toc479722878)

[1.1 Основы теории булевых функций 7](#_Toc479722879)

[1.2 Основы прикладной теории автоматов 10](#_Toc479722879)

[1.3 Абстрактные автоматы 12](#_Toc479722879)

[1.4 Минимизация полностью определенных автоматов 16](#_Toc479722879)

[1.5 Минимизация частичных автоматов 21](#_Toc479722879)

ГЛАВА [2. ГЕНЕРАЦИЯ ПСЕВДОСЛУЧАЙНЫХ АВТОМАТОВ 2](#_Toc479722878)5

[2.1 Описание алгоритма генерации 25](#_Toc479722882)

[2.2 Примеры выполнения задачи 26](#_Toc479722882)

ГЛАВА [3. ОПИСАНИЕ ПРОГРАММЫ 33](#_Toc479722878)

[3.1 Назначение программы 3](#_Toc479722882)3

[3.2 Блок-схема программы 3](#_Toc479722882)3

[3.3 Инструкция пользователя 3](#_Toc479722882)4

[ЗАКЛЮЧЕНИЕ 39](#_Toc479722894)

[СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ 40](#_Toc479722894)

[ПРИЛОЖЕНИЕ А 4](#_Toc479722894)1

[ПРИЛОЖЕНИЕ Б 4](#_Toc479722894)2

ПРИЛОЖЕНИЕ В [4](#_Toc479722895)4

# ПЕРЕЧЕНЬ УСЛОВНЫХ ОБОЗНАЧЕНИЙ

БФ – булева функция

ЧБФ – частичные булевы функции

СБФ – симметрическая булева функция

ТМ – троичная матрица

ДНФ – дизъюнктивная нормальная форма

МАА – минимизация абстрактных автоматов.

# РЕФЕРАТ ДИПЛОМНОЙ РАБОТЫ

Дипломная работа содержит:

- 42 страницы,

- 9 иллюстраций (рисунков),

- 27 таблиц,

- 3 приложения.

Ключевые слова: БУЛЕВА ФУНКЦИЯ, ЧАСТИЧНЫЙ АВТОМАТ МИЛИ, ЧАСТИЧНЫЙ АВТОМАТ МУРА, ГЕНЕРАЦИЯ ЧАСТИЧНЫХ АВТОМАТОВ, МИНИМИЗАЦИЯ ЧАСТИЧНЫХ АВТОМАТОВ

В данной дипломной работе рассматриваются вопросы, связанные с минимизацией и генерацией частичных автоматов Мили и Мура.

Целью данной дипломной работы является:

1. разработать алгоритмы для минимизации частичных автоматов и генерации псевдослучайных частичных автоматов, пригодных для минимизации;
2. реализовать полученные алгоритмы на языке C# и объединить их в виде программного комплекса;
3. провести опытную эксплуатацию программного комплекса и подготовить необходимую документацию для внедрения данного комплекса в учебном процессе отделения математической электроники.

Работа, прежде всего, носит практический характер, т.к. разработанный программный комплекс в дальнейшем может использоваться в учебном процессе.

Работа выполнена автором самостоятельно.

# РЭФЕРАТ ДЫПЛОМНАЙ РАБОТЫ

Дыпломная праца змяшчае:

- 42 старонкі,

- 9 ілюстрацый (малюнкаў),

- 27 табліц,

- 3 дадатку.

Ключавыя словы: БУЛЕВА ФУНКЦЫЯ, ЧАСТКОВЫ АУТАМАТ МIЛI, ЧАСТКОВЫ АУТАМАТ МУРА, ГЕНЕРАЦЫЯ ЧАСТКОВЫХ АУТАМАТАУ, МIНIМIЗАЦЫЯ ЧАСТКОВЫХ АУТАМАТАУ.

У дадзенай дыпломнай працы разглядаюцца пытанні, звязаныя з мінімізацыяй і генерацыяй частковых аўтаматаў Мілі і Мура.

Мэтай дадзенай дыпломнай працы з'яўляецца:

1. распрацаваць алгарытмы для мінімізацыі частковых аўтаматаў і генерацыі псеўдавыпадковых частковых аўтаматаў, прыдатных для мінімізацыі;
2. рэалізаваць атрыманыя алгарытмы на мове C # і аб'яднаць іх у выглядзе праграмнага комплексу;
3. правесці доследную эксплуатацыю праграмнага комплексу і падрыхтаваць неабходную дакументацыю для ўкаранення дадзенага комплексу ў навучальным працэсе аддзялення матэматычнай электронікі.

Праца, перш за ўсё, носіць практычны характар, бо распрацаваны праграмны комплекс у далейшым можа выкарыстоўвацца ў навучальным працэсе.

Праца выканана аўтарам самастойна.

# DIPLOMA WORK ABSTRACT

The thesis contains:

- 42 pages,

- 9 illustrations (drawings),

- 27 tables,

- 3 applications.

Keywords: BOOLEVA FUNCTION, PARTIAL MILI AUTOMAT, PARTIAL MUR AUTOMAT, GENERATION OF PARTIAL AUTOMATORS, MINIMIZATION OF PARTIAL AUTOMATORS

In this thesis, the issues related to the minimization and generation of the Mili and Moore partial automata are considered.

The purpose of this thesis is:

1. develop algorithms for minimizing partial automata and generating pseudo-random partial automata suitable for minimization;
2. implement the resulting algorithms in C # and combine them as a software package;
3. to conduct an experimental operation of the software complex and prepare the necessary documentation for the implementation of this complex in the educational process of the Department of Mathematical Electronics.

First of all, work is of a practical nature, because The developed software package can be used in the future in the educational process.

The work was done by the author himself.

# ВВЕДЕНИЕ

Сложные дискретные устройства получили весьма широкое применение в вычислительной технике. Они применяются для преобразования информации и её обработки. Для описания работы данных устройств используется широкий класс математических моделей – абстрактные автоматы – простейшие модели вычислительного устройства, предназначенные для обработки входной и формирования выходной информации.

Начальный этап в разработке этой теории был положен еще в середине 50-х годов прошлого века, и не смотря на свою, на первый взгляд, простоту, модель конечного автомата оказалась весьма удобной и легко применимой не только в информатике, но и в большом количестве инженерных задач.

В вычислительной технике современные устройства подразделяются на устройства комбинационной логики и последовательные устройства (или устройства с памятью). Первые из них зависят только от входных переменных. Вторые из них представляют более сложный класс устройств: устройства с памятью. В них значения на выходах зависят и от того, какие сигналы проходили ранее.

В данной дипломной работе требуется:

1. разработать алгоритм минимизация частичных автоматов Мили и Мура;
2. разработать алгоритм генерация не минимальных частичных автоматов, которые поддаются для минимизации;
3. создать программный комплекс на языке C# с использованием WPF-технологии, а также провести опытную эксплуатацию данного программного продукта;
4. провести адаптацию полученной программы для использования в учебном процессе отделения математической электроники.

# ГЛАВА 1 ЭЛЕМЕНТЫ ТЕОРИИ АБСТРАКТНЫХ АВТОМАТОВ

В этой главе будут изложены термины, понятия и определения, необходимые для понимания и выполнения дальнейшей работы.

## 1.1 Основы теории булевых функций

Функции от любого конечного числа двоичных переменных способны принимать лишь два значения: 0 и 1, принято называть булевыми функциями.

Известно, что под дискретным устройством понимается такое техническое устройство, структура и поведение которого описывается булевыми переменными.

**Определение 1.1.** Переменная *x*, принимающая значение из множества {0,1}, называется *булевой (логической, двоичной)*.

**Определение 1.2.** Функция F, зависящая от булевых переменных *x1*, *x2*, …, *xn*, и принимающая значения из множества {0,1}, называется булевой.

Такая функция обозначается, как F = F (*x1*, *x2*, …, *xn*).

*Табличный способ задания булевых функций*

При табличном способе задания булева функция (БФ) F = F (*x1*, *x2*, …, *xn*) представляется в виде таблицы, в левой части которой, в порядке естественной упорядоченности сверху вниз выписываются наборы значений переменных x1, x2, …, xn, на которых определены данные функции, а в правой части таблицы против каждого набора значений переменных выписываются соответствующие значения функции F. Такая таблица называется таблицей истинности булевой функции

F = F(*x1* , *x2* , … , *xn*).

В таблице 1.1 представлена булева функция трех переменных F = F (*x1*, *x2*, *x3*) посредством таблицы истинности.

Таблица 1.1 – Таблица истинности функции

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  | *F* |
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 1 |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 0 |

Приведенная таблица 1.1 истинности является одномерной. Основной недостаток данной таблицы – число строк, которое увеличивается экспоненциально в зависимости от числа переменных n.

В теории булевых функций особое значение имеют функции одного и двух переменных.

Значение всех булевых функций одной переменной представлены в таблице 1.2, а функции двух переменных – в таблице 1.3.

Таблица 1.2 – Булевы функции одной переменной

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *X* |  |  |  |  |
| 0 | 0 | 0 | 1 | 1 |
| 1 | 0 | 1 | 0 | 1 |

Из построенных функций функции ɡ*1* и ɡ*2* представляют собою константы 0 и 1. Функция ɡ*2* повторяет значение переменной *x* и потому просто совпадает с ней. Единственной нетривиальной функцией является функция ɡ*3*. Будем называть эту функцию отрицанием переменной *x* и обозначать через *.*

**Теорема 1.1.** Имеется точно различных булевых функций от n переменных (n = 1, 2, …).

Число булевых функций от двух переменных равно, согласно *теореме1.1,*  = = 16. Выпишем свободную таблицу всех этих функций (таблица 1.3).

Таблица 1.3 – Булевы функции двух переменных

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| *x* | *y* | *ƒ1* | *ƒ2* | *ƒ3* | *ƒ4* | *ƒ5* | *ƒ6* | *ƒ7* | *ƒ8* | *ƒ9* | *ƒ10* | *ƒ11* | *ƒ12* | *ƒ13* | *ƒ14* | *ƒ15* | *ƒ16* |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 0 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 |
| 1 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 |

Функции *ƒ*1*, ƒ*4*, ƒ*6*, ƒ*11*, ƒ*13*, ƒ*16*,* будут вырожденными.Легко увидеть, что *ƒ*1*=0, ƒ*4*=x, ƒ*6*=y, ƒ*11*=, ƒ*13*=, ƒ*16*=*1*.*

Остальные функции будут невырожденными. Введем для них специальные названия и обозначения.

Функция *ƒ*2 носит название *конъюнкции*, или *произведения*, или логического И. Для ее обозначения мы будем пользоваться знаком умножения, который, как и в обычной алгебре, применительно к буквенным выражениям будем опускать: *ƒ*2 = *x* × *y* = *x y*. Если значения переменных *x* и *y* функции *ƒ*2 понимать, как обычные числа 0 и 1, то значение функции *ƒ*2 можно находить, как легко проверить, с помощью обычного перемножения соответствующих значений переменных *x* и *y*.

Функции *ƒ*3 и *ƒ*5 можно называть отрицаниями импликации. Иногда они называются функциями запрета. Их можно обозначит следующим образом: *ƒ3* = *x*, *ƒ5* = *y* .

Функция *ƒ7* носит название *функции неравнозначности*, или суммы по модулю два: *ƒ7* = *x* *y.*

Функция *ƒ8* носит название *дизъюнкции* или *логического ИЛИ*.

Для ее обозначения мы будем употреблять знак дизъюнкции, применяющийся в математической логике: *ƒ8* = *x ˅ y*.

Функцию *ƒ9* будем называть *отрицанием дизъюнкции* или *«стрелкой Пирса»* и обозначать через *x* ↓ *y* .

Функция *ƒ10* носит название функции равнозначности, или логической эквивалентности, или просто эквивалентности и обозначатся через *x* ~ *y*.

Функция *ƒ12* и *ƒ14* носят название импликации. Для их обозначения употребляют стрелку, соединяющую между собой переменные *x* и *y, расположение в определенном порядке: ƒ12* = *y → x,*

*ƒ14* = *x → y.*

Функция *ƒ15* будем называть «*штрихом Шеффера»* или *отрицание конъюнкции.* Для ее обозначения используется обычно вертикальная черта, разделяющая переменные: *ƒ15* = *x* | *y.* Можно обозначать эту функцию следующим образом: .

Значение функций одного или двух переменных в общей теории булевых функции состоит в том, что из них может быть построена любая булева функция.

## 1.2 Основы прикладной теории автоматов

Для написания данного раздела были использованы источники [1–2].

Абстрактные автоматы образуют фундаментальный класс дискретных моделей как самостоятельная модель, и как основная компонента машин Тьюринга, автоматов с магазинной памятью, конечных автоматов и других преобразователей информации.

Модель абстрактного автомата широко используется, как базовая, для построения дискретных моделей автоматов, распознающих, порождающих и преобразующих последовательности символов.

Термин "автомат" используется в двух аспектах. С одной стороны, автомат – устройство, выполняющее некоторые функции без участия человека (например, ЭВМ). С другой стороны, термин "автомат" – математическое понятие и обозначает математическую модель реальных технических процессов.

В общем случае автомат представляется как "черный ящик" и полностью описывается совокупностью следующих шести объектов:

1) множество входных сигналов *Х*;

2) множество выходных сигналов *Y*;

3) множество состояний автомата *А*;

4) начальное состояние автомата *a0* как элемент множества *А*;

5) функция перехода из одного состояния в другое *f (a, x)*;

6) функция выходов автомата.

**Определение 1.3.** *Автоматом* называется дискретный преобразователь информации, способный принимать состояния из некоторого множества А, переходить под воздействием входных сигналов из одного состояния в другое и выдавать выходные сигналы.

**Определение 1.4.** *Цифровой автомат* – устройство, предназначенное для преобразования цифровой информации.

В начальный момент времени *t0* автомат находится в состоянии *a0*. В каждыймомент времени *t*, определяемый интервалом дискретности, автомат под воздействием входного сигнала *x(t)* скачкообразно переходит из состояния *ai(t)* в состояние *ai(t+1)* и выдает соответствующий выходной сигнал *y (t).*

Понятие состояние автомата используется для описания систем, выходы которых зависят не только от входных сигналов в данный момент времени, но и от некоторой предыстории, – сигналов, которые поступили на входы системы ранее, т.е. состояние – некоторая память о прошлом.

Цифровые автоматы условно подразделяются на два класса. В синхронных автоматах моменты времени, в которые фиксируются состояния автомата, задаются специальным устройством – генератором синхроимпульсов. В асинхронных автоматах моменты перехода автомата из одного состояния в другое заранее не определены и зависят от каких-то событий.

## 1.3 Абстрактные автоматы

Рассмотрим устройство, которое имеет один вход и один выход. Предположим, что устройство работает в некотором идеализированном дискретном времени.







Рисунок 1.1 – Устройство S

Так как устройство S работает в идеализированном дискретном времени, то выделим на шкале времени последовательные моменты времени t = 0, 1, 2, 3, …, т.е.



0 1 2 3

Предположим, что в некоторый момент времени на вход  поступает входной сигнал – одна из букв (входного) алфавита, а на его выходе появляется выходной сигнал – буква (выходного) алфавита *W*.

Под *алфавитом* понимается непустое множество попарно различных символов (элементов) – букв этого алфавита. Конечная упорядоченная последовательность букв образует слово в данном алфавите.

Пусть,  и устройство  таково, что в ответ на  на выходе появляется (таблица 1.4).

Таблица 1.4 – Таблица переходов автомата

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | 0 | 1 | 2 | 3 | 4 | 5 |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |

Отсюда видно, что реакция устройства на одну и ту же входную букву может быть различной. Например, входные и выходные сигналы в моменты времени 1,2 и 3,4. Другими словами, сигнал на выходе устройства в момент времени *t* зависит не только от входного сигнала *zi* в тот же момент времени, но и от его предыстории. Таким образом, в устройстве должна быть память о том, какое слово поступало на его вход до рассматриваемого момента времени.

Математической моделью подобных устройств является абстрактный автомат, изучаемый в теории автоматов. Термин «абстрактный автомат» используется в связи с идеализированным дискретным временем, а также потому, что в этой модели абстрагируются от реальной физической природы входных и выходных сигналов, рассматривая их как буквы некоторых алфавитов.

В настоящее время в вычислительной технике получили наибольшее распространение абстрактные автоматы Мили и Мура.

**Определение 1.5.** Абстрактный автомат Мили определяется как совокупность шести объектов:

,

где  – множество внутренних состояний (алфавит состояний);

 – множество входных сигналов (входной алфавит);

 – множество выходных сигналов (выходной алфавит);

функция переходов  определяет внутреннее состояние  в момент времени в зависимости от состояния  и входного сигнала , т.е. . Другими словами, функция переходов  ставит паре  внутреннее состояние , в которое переходит автомат  из состояния  под воздействием входного сигнала ;

функция переходов  ставит паре «входной сигнал  – состояние » в соответствие выходной сигнал , т.е. ;

 – начальное состояние абстрактного автомата , т.е. состояние, в котором находится автомат в момент времени . Другими словами, .

Из определения 1.5 следует, что  и .

Если на вход абстрактного автомата Мили , установленного в начальное состояние , подать последовательность входных сигналов , то на его выходе будет формироваться выходное слово такой же длины. Относя к каждому входному слову соответствующее выходное слово, получаем отображение, индуцируемое абстрактным автоматом Мили.

**Определение 1.6.** Автомат  называется конечным, если множества  являются конечными.

**Определение 1.7.** Автомат  называется детерминированным, если в нем выполняется условие однозначности переходов: находясь в некотором внутреннем состоянии , автомат  не может перейти в более чем одно состояние  под воздействием одного и того же входного сигнала .

**Определение 1.8.** Автомат  называется инициальным, если в нем выделено начальное состояние .

**Определение 1.9.** Автомат  называется полностью определенным, если область определения функции переходов  и функции выходов  совпадает с множеством всевозможных пар , где  и .

**Определение 1.10.** Автомат  называется не полностью определенным (или частичным), если функция переходов  или функции выходов  определена не на всех парах , где  и .

**Определение 1.11.** Абстрактный автомат Мура определяется как совокупность шести объектов:

,

где  – множество внутренних состояний (алфавит состояний);

 – множество входных сигналов (входной алфавит);

 – множество выходных сигналов (выходной алфавит);

функция переходов  определяет внутреннее состояние  в момент времени в зависимости от состояния  и входного сигнала , т.е. . Другими словами, функция переходов  ставит паре  внутреннее состояние , в которое переходит автомат  из состояния  под воздействием входного сигнала ;

функция переходов  ставит паре «входной сигнал  – состояние » в соответствие выходной сигнал , т.е. ;

 – начальное состояние абстрактного автомата , т.е. состояние, в котором находится автомат в момент времени . Другими словами, .

## 1.4 Минимизация полностью определенных автоматов

Автоматы являются математическими моделями технических устройств, предназначенных для переработки дискретной информации. Входной и выходной алфавиты автоматов определяются перерабатываемой информацией, в то время как на внутренний алфавит (алфавит состояний) обычно никаких условий не накладывается. Нужное преобразование информации может быть осуществлено автоматами с различным числом состояний. Поэтому возникает задача построения автомата, для которого это число минимально.

Здесь будем рассматривать только полностью определенные абстрактные автоматы.

**Определение 1.12.** Автомат , эквивалентный заданному автомату  и имеющий при этом наименьшее число состояний, называется минимальным (для автомата *S*).

Рассмотрим множество *A* всех состояний некоторого автомата *S*. Так как отношение эквивалентности состояний является рефлексивным, симметричным и транзитивным, то оно разбивает (причем однозначно) множество  на *классы эквивалентности* – классы попарно эквивалентных состояний.

Задача минимизации абстрактного автомата *S* (т.е. задача построения минимального автомата, эквивалентного) осуществляется в два этапа. На первом этапе множество *A* разбивается на классы эквивалентности  , а на втором этапе на основе этого разбиения строится искомый минимальный автомат . Если при выполнении первого этапа окажется, что , то исходный автомат  является минимальным, т.е. .

Наиболее распространенным методом минимизации абстрактных автоматов является метод Ауфенкампа и Хона (метод последовательных разбиений).

Рассмотрим применения данного метода на примере минимизации следующего абстрактного автомата Мили.

*Пример* 1.1*.* Пусть ,, и совмещенная таблица переходов и выходов имеет виде таблицы 1.5

Таблица 1.5 – Совмещенная таблица переходов и выходов

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

Этап 1 (разбиение множества состояний  на классы эквивалентности). Для определения всех пар  эквивалентных состояний автомата составим треугольную таблицу специального вида, клетки которой соответствуют различным неупорядоченным парам состояний , где . Если для состояний  и  существует входной сигнал , приводящий различным значениям выхода, т.е. , то соответствующую клетку треугольной таблицы отмечаем символом «крестик». В противном случае – в соответствующую клетку таблицы записывает всевозможные различные пары состояний , где ,  и . При этом необходимо учитывать следующие требования:  и .

Заметим, что число пар состояний  может быть от 0 до .

Далее в треугольной таблице следует вычеркнуть те клетки, в которых присутствуют пары состояний, соответствующие уже вычеркнутым клеткам.

В данном примере следует вычеркнуть клетку , так как в ней содержится пара состояний , и клетку , поскольку она содержит пару . Обе эти клетки выделены в треугольной таблице.

Таблица 1.6 – Треугольная матрица

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

После выполнения процедуры «дополнительного вычеркивания» клеток треугольная матрица принимает окончательный вид, изображенный в таблице 1.7.

Не вычеркнутые клетки результирующе таблицы соответствуют всем парам эквивалентных состояний исходного автомата Мили. При этом, если все клетки таблицы оказались вычеркнутыми, то это означает, что исходный автомат не минимизируется, т.е. .

Таблица 1.7 – Окончательный вид треугольной матрицы

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

В рассматриваемом примере выписываем всевозможные пары эквивалентных состояний: , ,  и . Далее, формируем классы эквивалентности, которые образуются попарно эквивалентными состояниями. Здесь имеем **, ** и  ****.

Заметим, что состояние  всегда должно попадать в класс .

Этап 2 (построение совмещенной таблицы переходов и выходов минимального автомата ).

Совмещенная таблица переходов и выходов автомата формируется на основе использования построенных на 1-м этапе классов эквивалентности , где , согласно следующим правилам:

1) Каждому классу эквивалентности  ставится в соответствие состояние  автомата S\*, где . Причем, обязательно . Тогда *b­1* начальное состояние автомата S\*.

2) Функция переходов  автомата S\* определяется так: если в автомате S для любого состояния  из класса эквивалентности  имеет место переход  и , то в автомате S\* выполняется переход . Здесь необходимо рассмотреть все классы эквивалентности, т.е. .

3) Функция выходов  минимального автомата S\* определяется таким образом: , где любое состояние из класса эквивалентности  и .

В данном примере положим ,  и . В таком случае совмещенная таблица переходов и выходов минимального автомата будет иметь вид таблицы 1.8.

Таблица 1.8 – Совмещенная таблица минимального автомата 

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

## 1.5 Минимизация частичных автоматов

Первым этапом метода минимизации частичного автомата, суть которого состоит в построении покрывающего автомата и минимально возможным числом состояний, является нахождение максимальной группировки состояний, а на втором этапе метода на основе найденной группировки строится минимальный покрывающий автомат.

На первом этапе метода формируется максимальная группировка на основе построенных пар совместимых состояний. Для построения всех таких пар может быть использован метод, применяемый раньше для выявления всех пар эквивалентных состояний полностью определенных автоматов Мили и Мура.

Выполнение метода минимизации частичного автомата проиллюстрируем на следующем примере.

*Пример* 1.2. Пусть , ,  и совмещенная таблица переходов и выходов частичного автомата Мили  выглядит как в таблице 1.9.

Таблица 1.9 – Совмещенная таблица переходов

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

Требуется построить минимальный покрывающий автомат .

Решение*.* Первоначально формируется треугольная таблица специального вида. При этом клетка  вычеркивается в том случае, когда существует входной сигнал  такой, что значения функции выходов ,  определены и .

В противном случае, в данную клетку заносятся всевозможные пары состояний , где  и , при условии, что значения функции переходов ,  определены  и , где .

Отсюда получаем следующие шесть пар совместимых состояний: , , ,  и . Отсюда видно, что отношение совместимости, введенное на множестве внутренних состояний , не является транзитивным. Построение максимальной группировки осуществляется на основе просмотра слева – направо результирующей треугольной таблицы. За время этого просмотра образуются некоторые системы подмножеств состояний. В качестве исходной системы берется система, состоящая из множества внутренних состояний .

Предположим, что после просмотра -го столбца таблицы образована система подмножеств , где . При переходе к му столбцу выделяются все состояния, несовместимые с  (это легко сделать, исходя из треугольной таблицы). Если подмножество  не содержит одновременно  и несовместимые с ним состояния, то оно переписывается без изменения. В противном случае из  образуется два новых подмножества: одно – путем удаления состояния , другое – путем удаления всех состояний, несовместимых с ним.

Таблица 1.10 – Результирующая треугольная таблица

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

Проделав эту процедуру для всех   и устранив не максимальные подмножества (содержащихся в других), получаем систему подмножеств, которая является результатом просмотра го столбца. Совокупность подмножеств, образованная после просмотра последнего столбца результирующей таблицы, является максимальной группировкой.

В рассматриваемом примере максимальная группировка строится как показано в таблице 1.11.

Следовательно, максимальная группировка содержит четыре максимальные группы совместимости: ,, и .

Группа совместимости  не является максимальной, поскольку она содержится в группе совместимости .

Таблица 1.11 – Построение максимальной группировки

|  |  |
| --- | --- |
| № шага | Система подмножеств |
| 0 |  |
| 1 | , |
| 2 | ,, |
| 3 | ,, |
| 4 | ,,, |
| 5 | ,,,, |

На втором этапе формируется непосредственно минимальный покрывающий автомат *S*\*, число внутренних состояний которого определяется числом максимальных групп совместимости. Для этого каждой максимальной группе совместимости ставится в соответствие состояния *b*1, *b*2, … минимального автомата *S*\*. При этом состояние *b*1 ставится в соответствие группе совместимости, содержащей состояние *a1*. Если таких максимальных групп совместимости несколько, то – любой из них (для однозначности решения будем ставить *b*1 в соответствие группе с наименьшим номером).

Итак, получаем ,,, . В таком случае совмещенная таблица переходов и выходов минимального покрывающего автомата *S*\* будет иметь вид как отражено в таблице 1.12.

Таблица 1.12 – Совмещенная таблица минимального автомата 

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Z A |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

В дальнейшем материал данного раздела будет использован при создании соответствующих алгоритмов.

# ГЛАВА 2 ГЕНЕРАЦИЯ ПСЕВДОСЛУЧАЙНЫХ АВТОМАТОВ

Далее в этой главе будет рассмотрен алгоритм генерации псевдослучайных автоматов Мили и Мура.

## 2.1 Описание алгоритма генерации

Алгоритм генерации состояний частичного автомата заключается в выборе любого состояния автомата (автомат может быть не минимальным) и его дублирования. Таким образом, получается, что в таблицу будет добавлен еще один столбец, однако у него будет новое состояние (новый индекс). Но важно учесть какие значения в таблице могут быть изменены, т.е. чтобы при процедуре минимизации эти два состояния объединялись. Затем случайным образом заменяем их из списка возможных вариантов.

Таблица 2.1 – Добавление нового состояния автомата 

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

Применяем алгоритм минимизации и треугольную таблицу Ауфенкампа. Туда записываются состояния с потенциалом на соединение. Таким образом, в новом столбце обязательно такие же выходные сигналы *w*, что и в выбранном состоянии. Далее, после таблицы, определяются классы эквивалентных состояний, которые могут быть объединены при минимизации. Таким образом, в таблице везде, где встречается выбранное состояние, его можно заменить на любой состояние из класса, в котором оно присутствует.

## 2.2 Примеры выполнения задачи

Приведем примеры минимизации и генерации частичных автоматов Мили и Мура. Основной задачей этого раздела является проверка и отражение работоспособности приложения. Для этого будет выполнено несколько контрольных примеров, вводимые данные и результаты работы приложения будут вынесены в виде таблиц.

*Пример* 2.1. Для минимизации используем частичный автомат Мили, описанный в примере 1.2 (глава 1, раздел 1.5).

Пусть , , 

Таблица 2.2 – Отмеченная таблица переходов примера 2.1

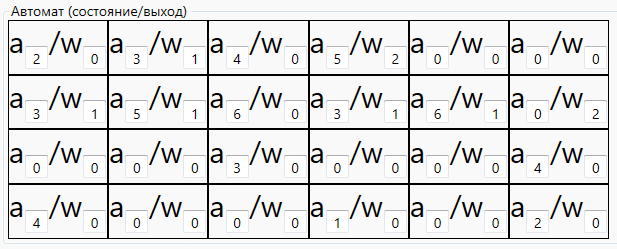
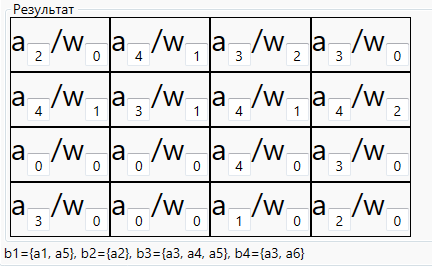


Таблица 2.3 – Минимальный покрывающий автомат примера 2.1



*Пример* 2.2. Пусть автомат Мили имеет следующие параметры: , , .

Таблица 2.4 – Отмеченная таблица переходов примера 2.2

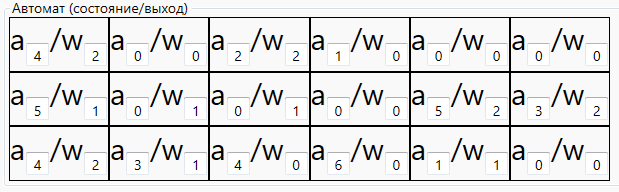
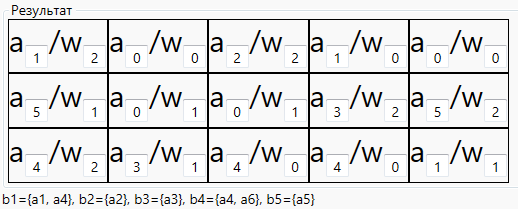


Таблица 2.5 – Минимальный покрывающий автомат примера 2.2



*Пример* 2.3. Пусть автомат Мура имеет следующие параметры: , , .

Таблица 2.6 – Отмеченная таблица переходов примера 2.3

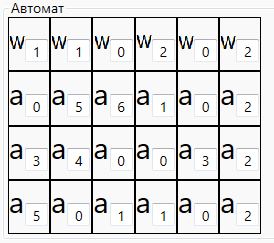
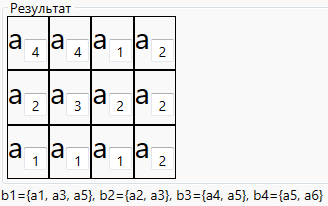


Таблица 2.7 – Минимальный покрывающий автомат примера 2.3



*Пример* 2.4. Пусть автомат Мура имеет следующие параметры: ,, .

Таблица 2.8 – Отмеченная таблица переходов примера 2.4

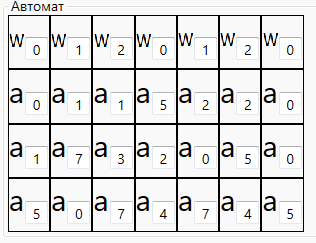
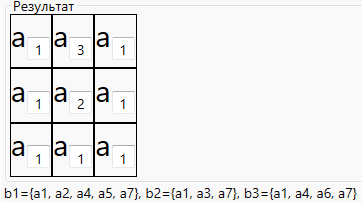


Таблица 2.9 – Минимальный покрывающий автомат примера 2.4



*Пример* 2.5. Пусть автомат Мура имеет следующие параметры: ,, .

Таблица 2.10 – Отмеченная таблица переходов примера 2.5

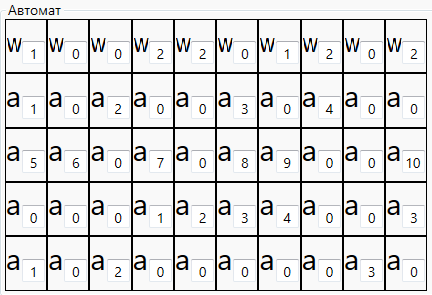
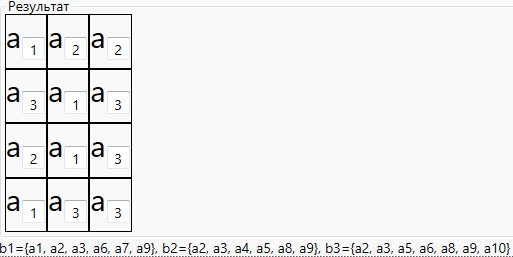


Таблица 2.11 – Минимальный покрывающий автомат примера 2.5



*Пример* 2.6. Пусть автомат Мили имеет следующие параметры:  , .

Таблица 2.12 – Отмеченная таблица переходов примера 2.6

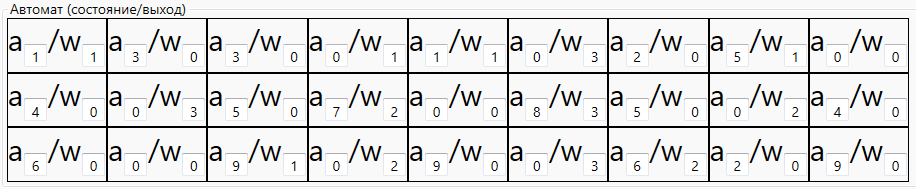
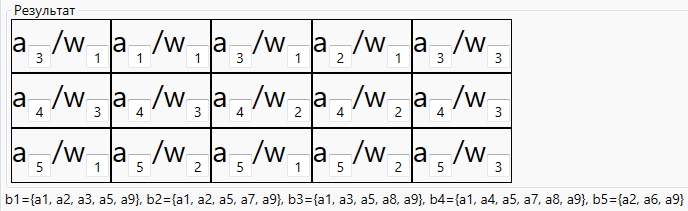
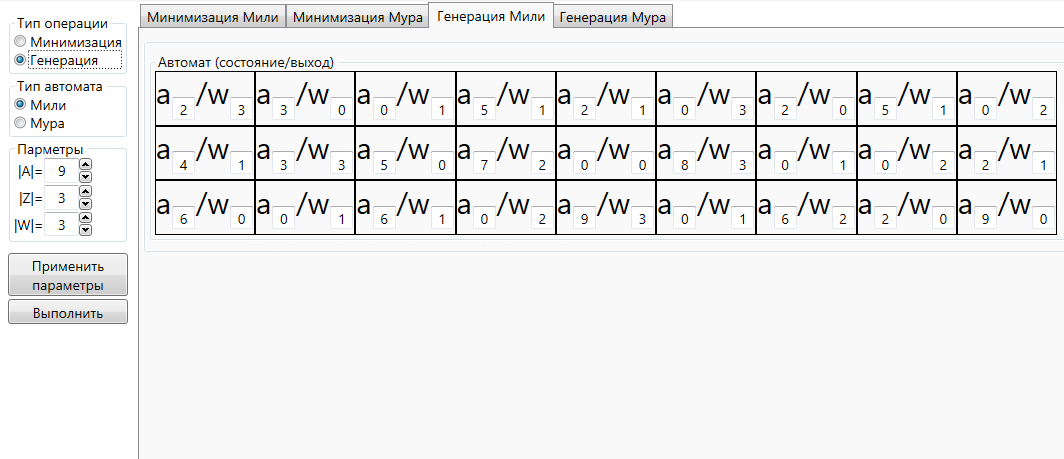


Таблица 2.13 – Минимальный покрывающий автомат примера 2.6



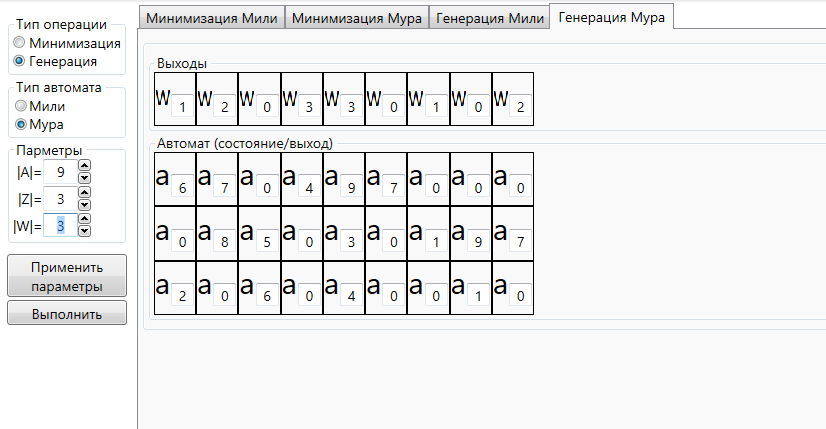
*Пример* 2.7. Пусть необходимо сгенерировать частичный автомат Мили со следующими параметрами:  , .

Таблица 2.14 – Отмеченная таблица переходов примера 2.7



*Пример* 2.8. Пусть необходимо сгенерировать частичный автомат Мура со следующими параметрами:  , .

Таблица 2.15 – Отмеченная таблица переходов примера 2.8



Как видно из примеров, приложение имеет понятный вывод результатов. Однако для удобства работы пользователя необходимо изучить работу приложения. Этому будет посвящен следующий раздел.

# ГЛАВА 3 ОПИСАНИЕ ПРОГРАММЫ

Целью этой главы является рассмотрение учебного комплекса программ. В нем реализованы минимизация частичных автоматов Мили и Мура, их генерация.

## 3.1 Назначение программы

Программный комплекс, разработанный в результате выполнения дипломной работы, предназначен для:

1. минимизации частичных автоматов Мили и Мура;
2. генерации псевдослучайных частичных автоматов Мили и Мура.

## 3.2 Блок-схема программы

Блок-схема данного комплекса представлена на рисунке 3.1.

Начало

Ввод начальных параметров

Генерация заданного автомата

Формирование таблиц для задания автомата

Ввод частичного автомата

Конец

Минимизация частичного автомата

Рисунок 3.1 – Блок-схема программного комплекса

## 3.3 Инструкция пользователя

Как результат выполнения теоретической части, была реализована программа для минимизации и генерации псевдослучайных частичных автоматов Мили и Мура. Данный программный продукт имеет графический интерфейс и может работать в любой ОС семейства Windows, предъявляя лишь требования к наличию на целевой машине .net framework 3.5 и выше.

Разработанная программа имеет интуитивно понятный графический интерфейс.

После загрузки приложения пользователю предоставляется возможность выбора выполняемых операций (минимизация или генерация) для указанного типа автомата (Мили или Мура) и ввести основные параметры (рисунок 3.2).

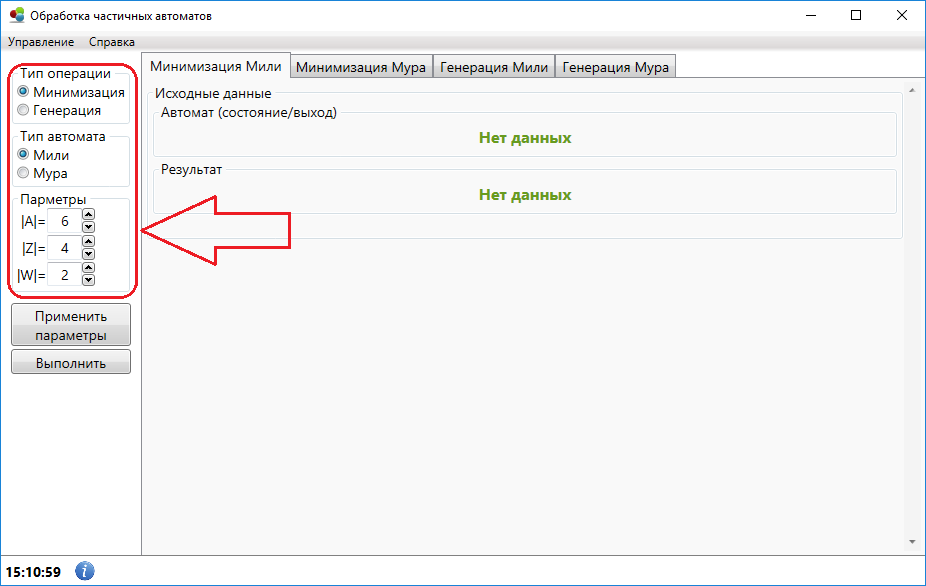


Рисунок 3.2 – Ввод параметров для начала работы

Пользователь должен ввести число входных, выходных переменных, число внутренних состояний автомата:

1. |А| – мощность множества внутренних состояний;
2. |Z| – мощность множества входных переменных;
3. |W| – мощность множества выходных переменных.

Следует отметить, что программа имеет ограничение по числу состояний и выходов (не более 10). Затем нужно указать тип выполняемой операции и для какого типа автомата она будет выполняться (рисунок 3.3) и нажать кнопку «Применить параметры» (рисунок 3.4).

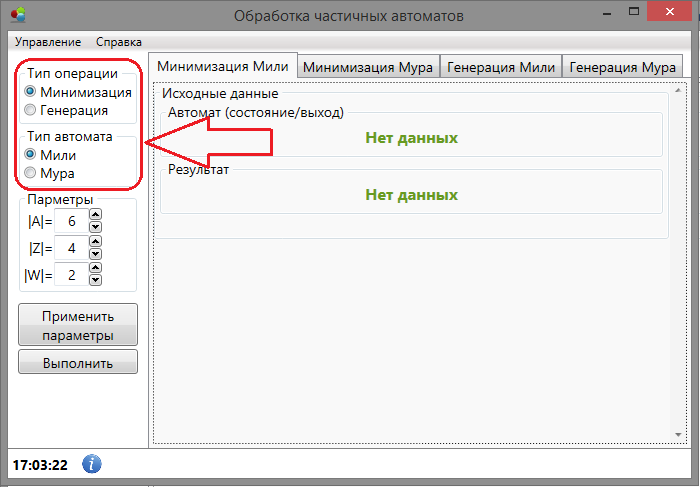


Рисунок 3.3 – Выбор выполняемой операции и типа автомата

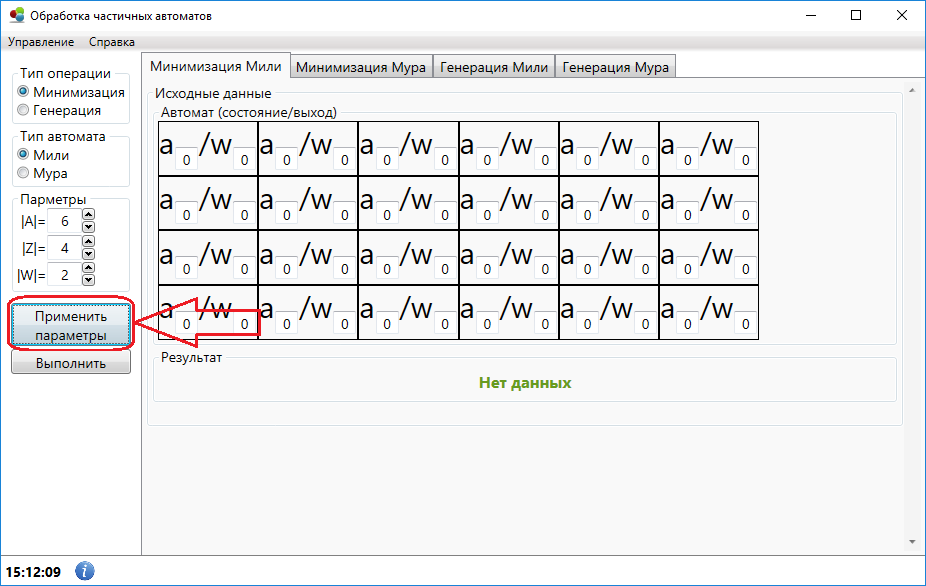


Рисунок 3.4 – Применение указанных параметров для начальной инициализации приложения

После этого будет открыта соответствующая вкладка, где в зависимости от операции будет возможно либо увидеть результат процесса генерации частичного автомата, либо будет предложено заполнить таблицу переходов автомата для минимизации (рисунок 3.5).

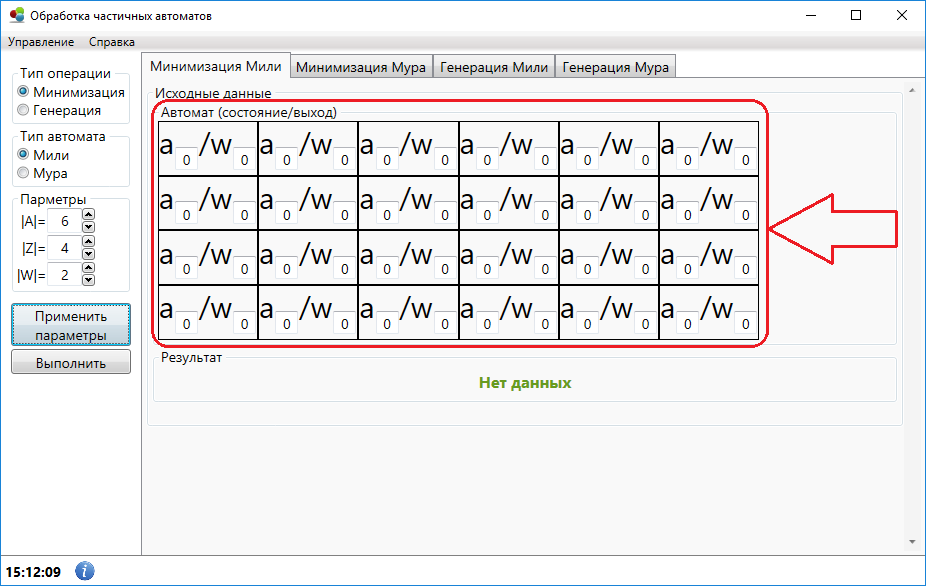


Рисунок 3.5 – Отмеченная таблица переходов

После заполнения таблицы переходов следует нажать кнопку «Выполнить» (рисунок 3.6), после чего ниже можно будет увидеть результат (рисунок 3.7).

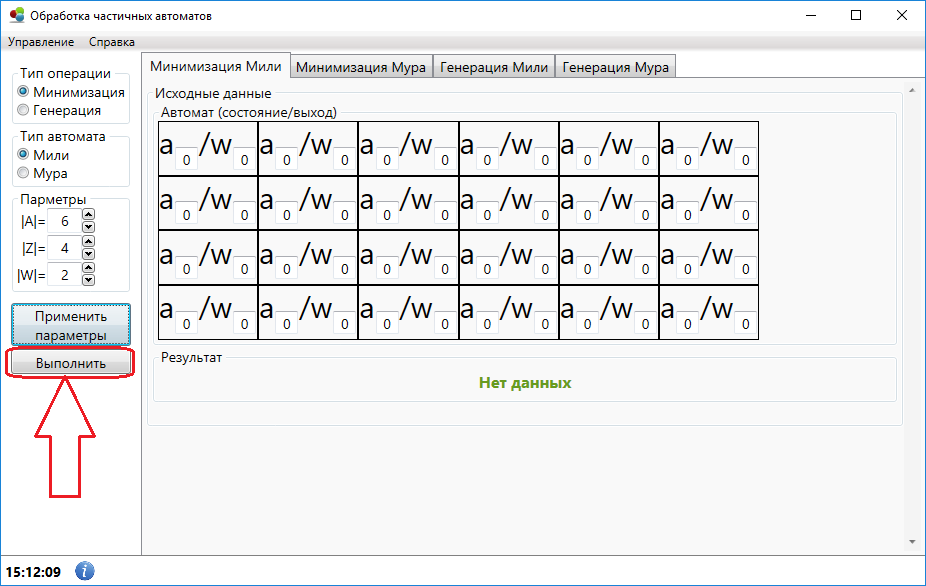


Рисунок 3.6 – Кнопка «Выполнить»

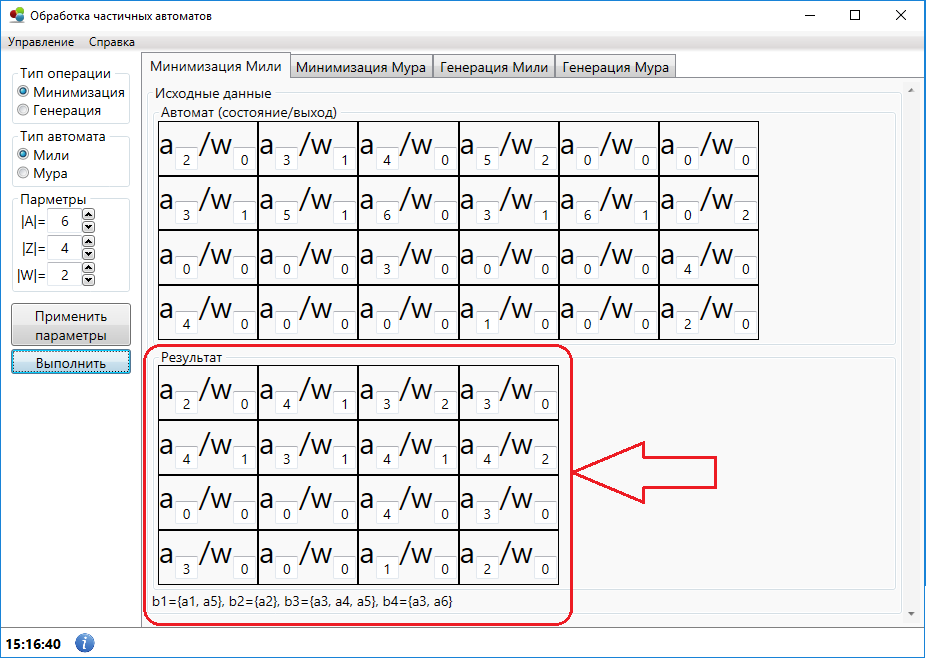


Рисунок 3.7 – Результат минимизации автомата Мили

Так же программа предусмотрена область уведомлений и система оповещения пользователей об ошибках, выдается звуковой сигнал и поясняющее сообщение в области уведомлений (рисунок 3.8).

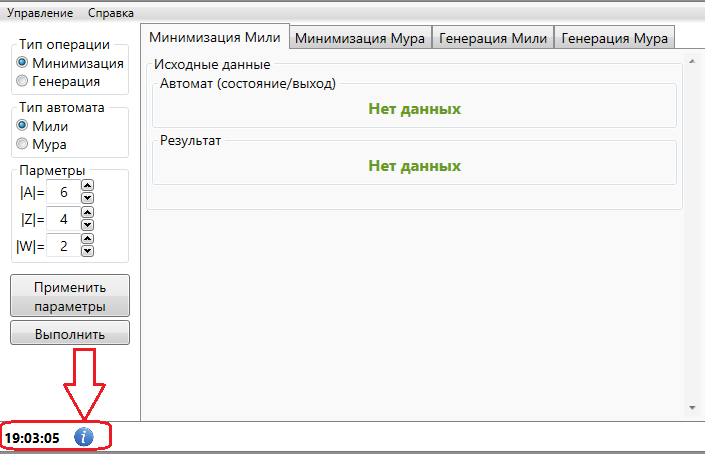


Рисунок 3.8 – Области уведомлений

# ЗАКЛЮЧЕНИЕ

При написании дипломной работы были проработаны научно-технические источники в области теории абстрактных автоматов.

Результатом работы является разработка двух алгоритмов: алгоритм минимизации частичных автоматов Мили и Мура (раздел 1, глава 1.4 -1.5), а также алгоритм для генерации соответствующих не минимальных псевдослучайных автоматов (раздел 2, глава 2.1).

Это, в свою очередь, позволило использовать данные алгоритмы для реализации программного учебного комплексе. Сам программный комплекс был реализован на языке C# с использованием технологии WPF и среды разработчика Visual Studio 2015. Полученный в результате проделанной работы комплекс позволяет выполнять операции по минимизации и генерации псевдослучайных автоматов Мили и Мура. Разумным ограничением на мощности множеств *A*, *W*, *Z* было положено число 10, однако для опытного исследования вполне достаточно рассматривать автоматы с числом состояний, не превышающим 7-8.

Данный программный комплекс является в будущем может быть использован в учебном процессе отделения математической электроники. Положительный опыт использования подобных программных комплексов был получен ранее при использовании в учебном процессе результатов дипломной работы Митина Александра Семеновича («Преобразование и минимизация абстрактных автоматов»).

С целью адаптации данного программного средства к учебному процессу была написана инструкция пользователя и добавлена справочная информация, содержащая необходимые теоретические сведения.

# СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ

1. Супрун В.П., Автоматика и вычислительная техника. — Рига: АН Латвийской ССР, 1984.
2. Супрун В.П., Конспект лекций «Прикладная теории автоматов».
3. Супрун В.П., Тихомирова Н.Н. Программное обеспечение ЭВМ. Выпуск 73: Автоматизированный синтез комбинационных сетей из программируемых логических матриц. — Минск: АН БССР. Ин-т математики, Белорусский государственный университет им. В.И. Ленина.
4. Поттосин Ю.В., Дискретная математика и теория проектирования цифровых устройств и систем. — Минск: БГУИР, 2007.