**Н. И. Туккель**
**А. А. Шалыто,**
*доктор технических наук, профессор*

# Реализация вычислительных алгоритмов на основе автоматного подхода

Показано, что произвольная схема алгоритма (программы) может быть реализована оператором do-while, телом которого является оператор switch, изоморфно реализующий граф переходов конечного автомата, построенный формально по указанной схеме. Эта реализация проще получаемой при использовании метода Ашкрофта–Манны.

В [11] была предложена парадигма автоматного программирования, в рамках которой построение и реализация алгоритма выполняются в виде конечного автомата, базирующегося на понятии "состояние". В этой работе указанный подход был использован применительно к системам логического управления, в которых входные и выходные воздействия являются двоичными переменными.

В [14] этот подход был расширен для обеспечения возможности программной реализации событийных ("реактивных") систем, в которых в качестве входных воздействий добавлены события, а входные переменные и выходные воздействия реализуются функциями.

Для указанных классов систем выделение состояний является естественным, так как они в значительной мере определяются физическими состояниями объектов управления [13].

Для вычислительных алгоритмов выделение состояний не столь естественно, однако и для этого класса задач автоматный подход может быть применен [7].

Такой подход позволяет унифицировать процесс построения структурированных программ с визуализацией их состояний, что имеет важное значение особенно для целей обучения [4]. Это не удается обеспечить при традиционном процедурном подходе, так как в нем в качестве базовых используются понятия "условие" и "действие", а не "состояние".

Программы, построенные на основе явного выделения состояний, назовем автоматными.

Психологические трудности [7], возникающие при непосредственном построении алгоритмов для вычислительных задач с применением автоматов, связаны, в частности, с непривычной реализацией циклов. Эти трудности привели к тому, что в [6] автоматный подход был использован только для построения одного из алгоритмов поиска подстрок, в то время как для десятков других алгоритмов, описанных в этой книге, автоматы не применялись ни при алгоритмизации, ни при реализации.

Разделим рассматриваемый класс алгоритмов на два подкласса: без использования событий (вычислительные алгоритмы) и с их использованием (событийно-вычислительные алгоритмы). Настоящая работа посвящена первому из указанных подклассов. В этом подклассе будем рассматривать схемы алгоритмов с одним входом и одним выходом, в которых через каждую вершину проходит путь от входа схемы к ее выходу (отсутствуют недостижимые вершины и бесконечные циклы).

Покажем, что произвольную схему алгоритма (программы), принадлежащую этому подклассу, можно реализовать оператором do-while, телом которого является оператор switch. Последний изоморфно реализует граф переходов конечного автомата, формально построенный по исходной схеме. Отметим, что, в отличие от [14], в рассматриваемом подклассе алгоритмов при реализации каждого автомата используется один оператор switch.

Эта реализация проще получаемой при использовании метода Ашкрофта–Манны [1,3], который требует введения состояния для каждой вершины неструктурированной схемы алгоритма и приводит к построению его структурированной схемы с числом вершин, определяемым соотношением:

$$B = 4У + 3О + 2,$$

где $У$ и $О$ — соответственно количество условных и операторных вершин в неструктурированной схеме алгоритма.

Подход к структурированию программ с использованием булевых признаков [3] в настоящей работе не рассматривается, так как он является эвристическим.

## Изложение метода

Ниже излагается метод построения по схеме алгоритма (программы) графа переходов, по которому, в свою очередь, строится структурированная программа, являющаяся автоматной. Если задан текст программы, то для применения предлагаемого метода необходимо предварительно построить ее схему, которую назовем традиционной. Она может быть как структурированной, так и неструктурированной, но не содержит явно выделенных состояний.

Метод состоит из перечисленных ниже этапов.

1. Используя метод, предложенный в [2] для аппаратных реализаций схем алгоритмов, в заданную схему в зависимости от выбранного для ее замены типа автомата (Мура или Мили) вводятся пометки, соответствующие номерам его состояний. При этом для автоматов обоих типов начальной и конечной вершинам схемы присваивается номер $0$, что обеспечивает построение "замкнутого" графа переходов. При построении автомата Мура $k$-й группе соединенных последовательно операторных вершин (она может состоять также и из одной вершины) присваивается номер $k$. При построении

автомата Мили соответствующий номер присваивается точке, следующей за последней из последовательно соединенных операторных вершин группы, причем указанные точки для различных групп могут совпадать. Это приводит к тому, что автомат Мили имеет число состояний, не превышающее их число в эквивалентном автомате Мура. Номера, применяемые для построения автомата Мили, будем указывать на схеме алгоритма в скобках, а номера для построения автомата Мура — без скобок. Используемая нумерация начальной и конечной вершин отличается от [1, 3], где этим вершинам присваиваются различные номера, а получающийся при этом граф переходов разомкнут.

2. Обеспечение "замкнутости" графа переходов и выбранная стандартная программная реализация могут привести к необходимости введения дополнительного состояния (как для автоматов Мили, так и для автоматов Мура), если схема алгоритма содержит контур без операторных вершин и пометок других состояний, замыкающийся на дуге, исходящей из начальной вершины.

3. В случае построения автомата Мили в схеме алгоритма за счет дублирования некоторых операторных вершин уменьшается число точек, следующих за ними, что обеспечивает сокращение числа состояний автомата этого типа.

4. В схеме программы выделяются пути между смежными точками (пометками), включая пути начинающиеся и оканчивающиеся в одной и той же точке. Для каждого пути выписываются условия перехода и выполняемые при этом действия. На основе выделенных состояний и переходов (включая петли) строится граф переходов автомата выбранного типа.

5. В случае построения автомата Мили его граф переходов, если это возможно, упрощается за счет переноса одинаковых действий с дуг, входящих в вершину, непосредственно в нее самое. При этом строится граф переходов смешанного автомата.

6. Построенный граф переходов изоморфно реализуется с помощью оператора switch языка Си или его аналога из других языков программирования.

7. Полученный фрагмент программы используется в качестве тела оператора do-while, условием выхода из которой является нахождение автомата в состоянии *0*.

При необходимости после получения графа переходов по нему может быть построена структурированная схема программы, соответствующая оператору do-while, тело которого изоморфно графу переходов и содержит дешифратор состояний. Схемы с такой структурой назовем автоматными.

В отдельных случаях в построенной схеме может быть уменьшено число вершин за счет потери указанного изоморфизма (но не структурированности), по аналогии с тем, как это выполняется в [8] для схем, построенных методом Ашкрофта–Манны, с помощью перехода к рекурсивным структурированным схемам.

Завершив изложение метода, отметим, что подход, описанный в [2], состоит только из первого и четвертого этапов.

С помощью предлагаемого метода в настоящей работе решаются две задачи: преобразование традиционных программ (обычно структурированных) в автоматные (примеры 1–5) и преобразование неструктурированных схем алгоритмов в автоматные (примеры 6–12).

Перейдем к рассмотрению этих задач.

### Преобразование традиционных программ в автоматные

Пример 1. Задана программа поиска максимума в одномерном массиве:

```
void main ()
{
  enum  { n = 8 } ;
  int   a[n] = { 44, 55, 12, 42, 94, 18, 6, 67 } ;
  int   max = a [0] ;
  int   i ;

  for( i = 1 ; i < n ; i++ )
   if( a[i] > max )
    max = a[i] ;
}
```

По ней требуется построить программу, соответствующую автомату Мура.

Схема этой программы, в которой без скобок указаны пометки, соответствующие состояниям автомата Мура, приведена на рис. 1.

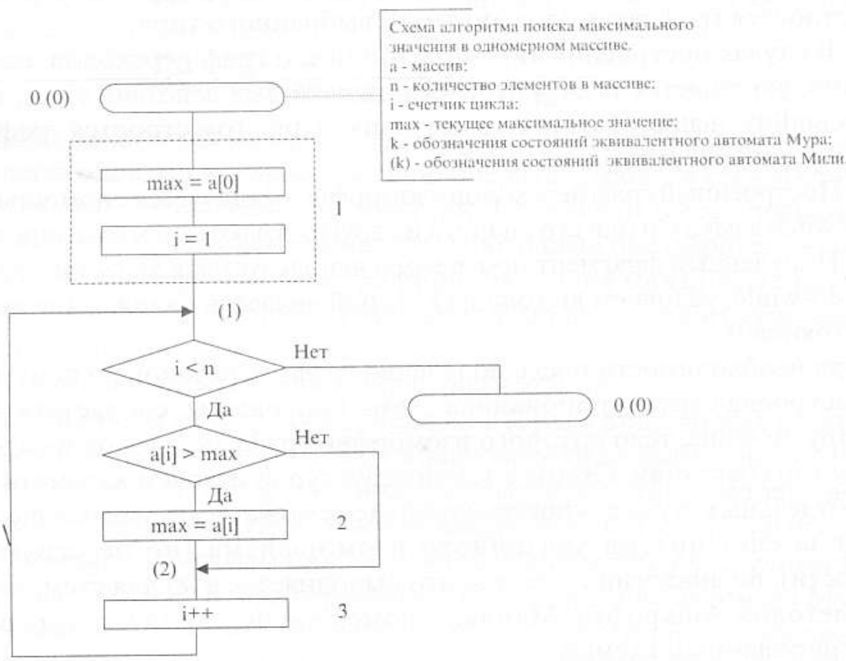


Рис. 1

Граф переходов автомата Мура с четырьмя вершинами, построенный по этой схеме, приведен на рис. 2. На этом графе пометки некоторых дуг упрощены (вычеркнуты) за счет введения приоритетов и пометки "иначе". Пометка петли исключена, так как ей вместе с петлей при программной реализации соответствует оператор break.

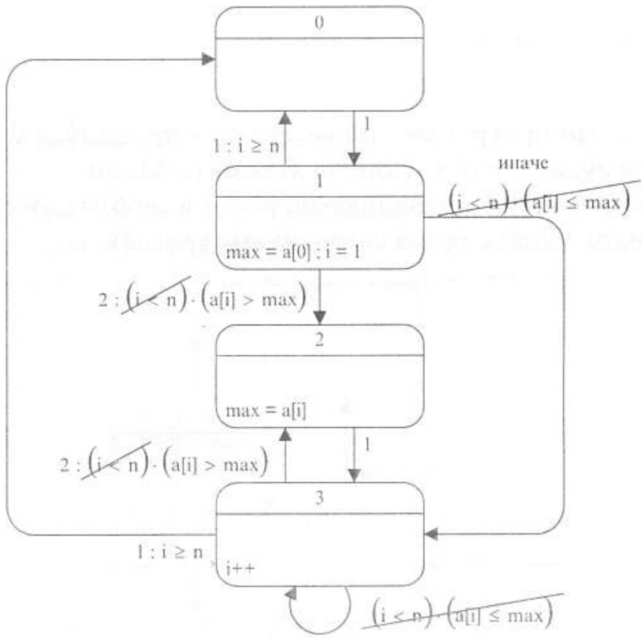


Рис. 2

Используя этот граф переходов, построим автоматную программу требуемого типа:

```
void main()
{
  int   y = 0 ;
  enum  { n = 8 } ;
  int   a[n] = { 44, 55, 12, 42, 94, 18, 6, 67 } ;
  int   max ;
  int   i ;
  do
    switch( y )
    {
case 0:
y = 1 ;
break ;
case 1:
max = a[0] ; i = 1 ;
if( i >= n )            y = 0 ;
else
if( a[i] > max )            y = 2 ;
else                    y = 3 ;
break ;
case 2:
max = a[i] ;            y = 3 ;
```

```
break ;
case 3:
i++ ;
if( i >= n )                    y = 0 ;
else
if( a[i] > max )                    y = 2 ;
break ;
   }
  while( y != 0 ) ;
}
```

Пример 2. По программе, приведенной в предыдущем примере, построить программу, соответствующую автомату Мили.

Используя пометки, указанные на рис. 1 в скобках, построим граф переходов автомата Мили с тремя состояниями (рис. 3).

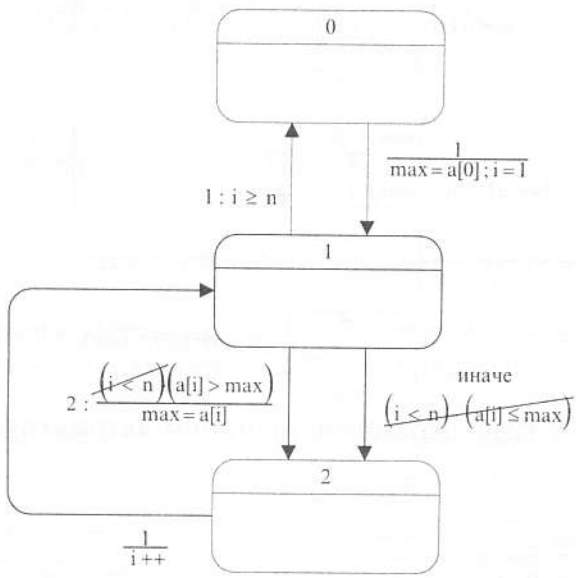


Рис. 3

Используя этот граф переходов, построим автоматную программу требуемого типа:

```
void main()
{
  int       y = 0 ;
  enum      { n = 8 } ;
  int       a[n] = { 44, 55, 12, 42, 94, 18, 6, 67 } ;
  int       max ;
  int       i ;

  do
    switch( y )
    {
case 0:
max = a[0] ; i = 1 ;           y = 1 ;
break ;
```

```
case 1:
if( i >= n )              y = 0 ;
else
if( a[i] > max )
{ max = a[i] ;            y = 2 ; }
else                      y = 2 ;
break ;
case 2:
i++ ;                         y = 1 ;
break ;
    }
  while( y != 0 ) ;
}
```

Пример 3. Преобразуем схему программы на рис. 1 с целью получения автомата Мили с двумя состояниями и построим по этой схеме автоматную программу. Для этого продублируем операторную вершину с пометкой "i++" (рис. 4).

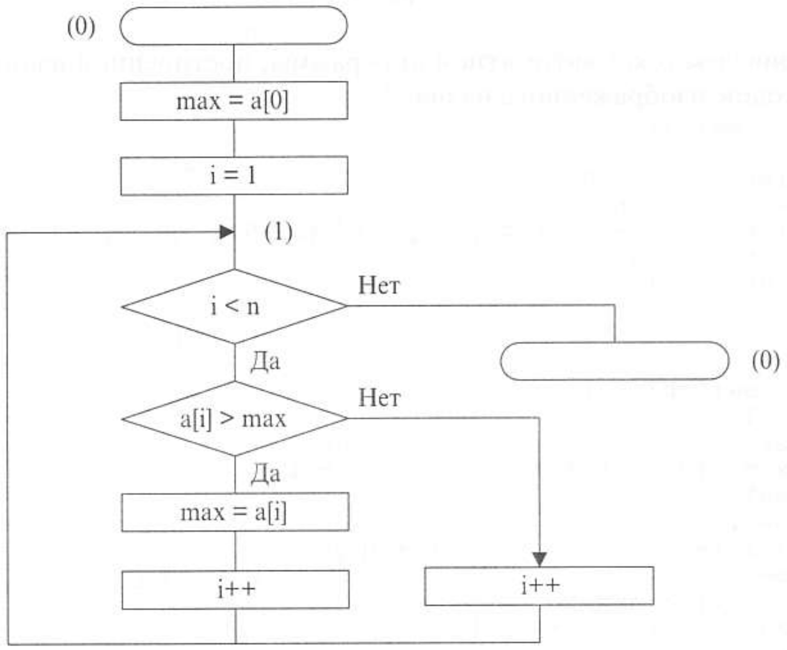


Рис. 4

Схеме программы на рис. 4 соответствует граф переходов автомата Мили с двумя состояниями (рис. 5).

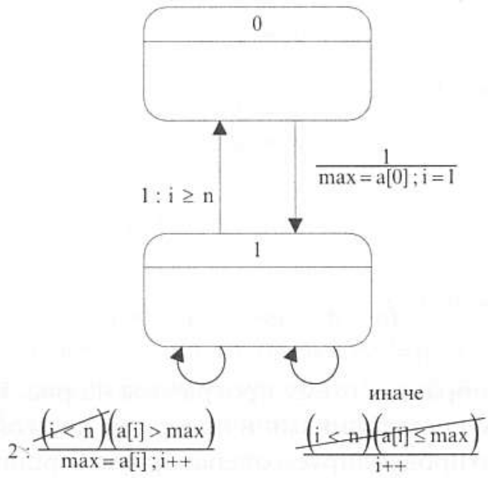


Рис. 5

Приведем текст автоматной программы, построенной на основе графа переходов, изображенного на рис. 5:

```
void main()
{
  int     y = 0 ;
  enum    { n = 8 } ;
  int     a[n] = { 44, 55, 12, 42, 94, 18, 6, 67 } ;
  int     max ;
  int     i ;


  do
    switch( y )
    {
case 0:
max = a[0] ; i = 1 ;          y = 1 ;
break ;
case 1:
if( i >= n )             y = 0 ;
else
if( a[i] > max )
{ max = a[i] ; i++ ; }
else
  i++ ;
break ;
   }
  while( y != 0 ) ;
}
```

Пример 4. Задана программа поиска максимума в двумерном массиве:

```
void main()
{
  enum    { m = 2, n = 4 } ;
  int     a[m][n] = { 44, 55, 12, 42, 94, 18, 6, 67 } ;
  int     max = a[0][0] ;
```

```
int    i, j ;

for( i = 0 ; i < m ; i++ )
  for( j = 0 ; j < n ; j++ )
    if( a[i][j] > max )
      max = a[i][j] ;
}
```

По ней требуется построить программу, соответствующую автомату Мили.

Если по схеме этой программы построить граф переходов автомата Мили, то он будет содержать четыре вершины. Дублирование операторной вершины с пометкой “j++” приводит к построению схемы программы, приведенной на рис. 6.

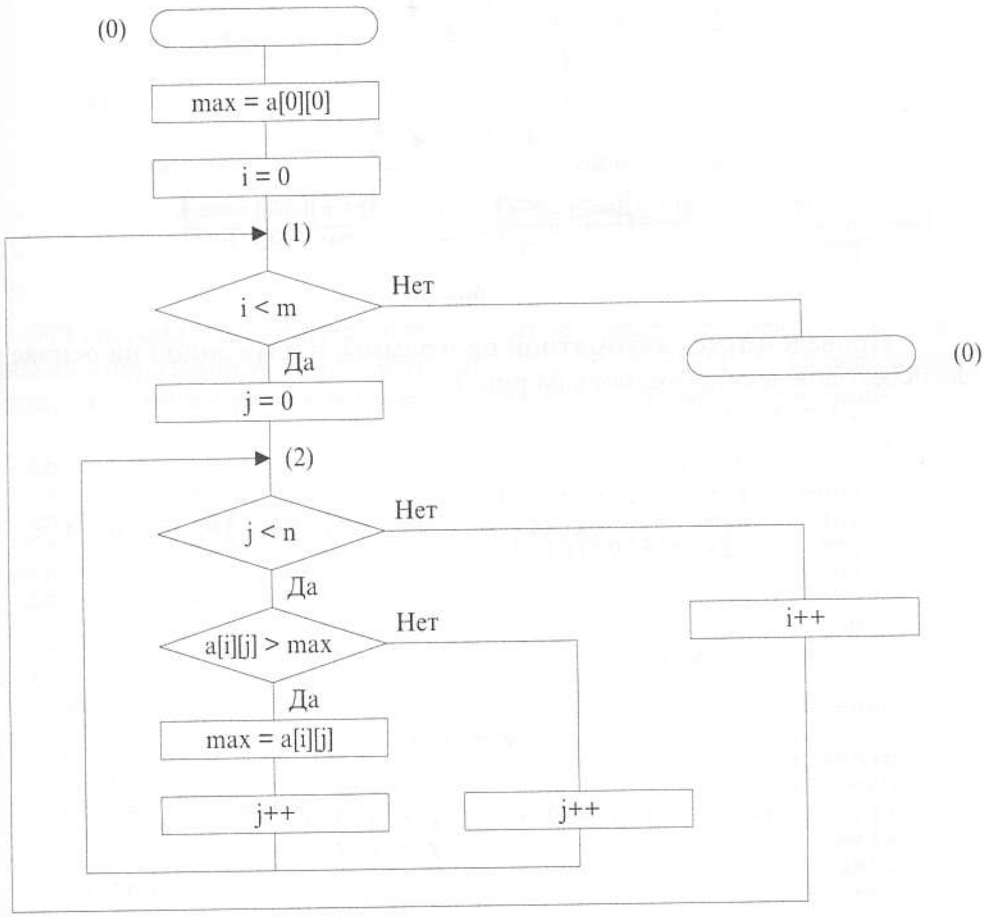


Рис. 6

Эта схема реализуется графом переходов автомата Мили с тремя состояниями (рис. 7).

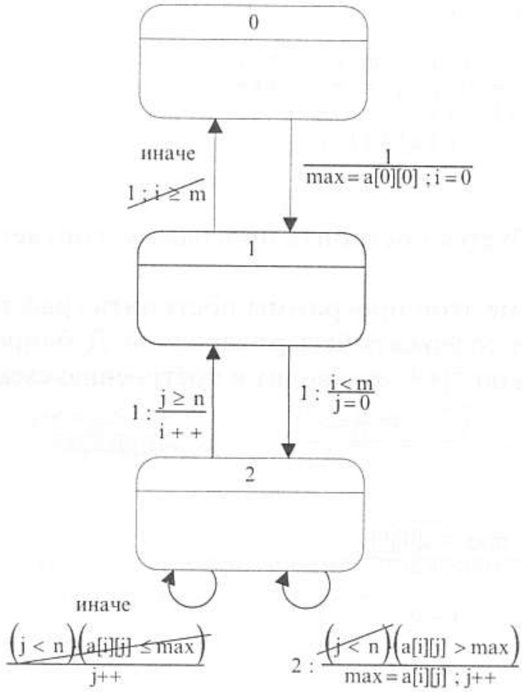


Рис. 7

Приведем текст автоматной программы, построенной на основе графа переходов, изображенного на рис. 7:

```
void main()
{
  int    y = 0 ;
  enum   { m = 2, n = 4 } ;
  int    a[m][n] = { 44, 55, 12, 42, 94, 18, 6, 67 } ;
  int    max = a[0][0] ;
  int    i, j ;

  do
    switch( y )
    {
case 0:
i = 0 ;                   y = 1 ;
break ;
case 1:
if( i < m )     { j = 0 ;      y = 2 ; }
else                           y = 0 ;
break ;
case 2:
if( j >= n )    { i++ ;        y = 1 ; }
else
if( a[i][j] > max )
{ max = a[i][j] ; j++ ; }
else
j++ ;
break ;
```

```
    }
  while( y != 0 ) ;
}
```

Пример 5. Автомат Мили, граф переходов которого приведен на рис.7, можно декомпозировать на два автомата Мили, которые обозначим A0 и A1. Каждый из автоматов содержит по два состояния. При этом автомат A1 вложен в автомат A0 (рис. 8).

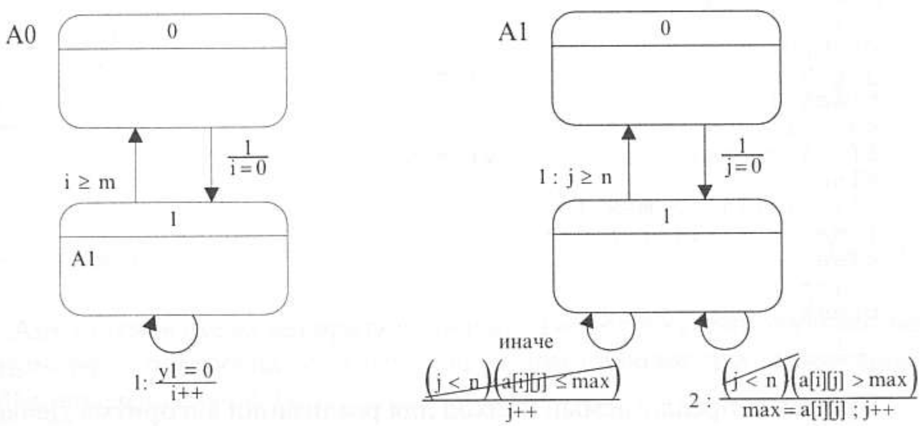


Рис. 8

Эта система взаимосвязанных автоматов может быть реализована вложенными операторами switch. Однако "более изоморфной" является программа, в которой каждый автомат реализуется отдельной функцией:

```
int       y0 = 0 ;
int       y1 = 0 ;
enum      { m = 2, n = 4 } ;
int       a[m][n] = { 44, 55, 12, 42, 94, 18, 6, 67 } ;
int       max ;
int       i, j ;

void main()
{
  max = a[0][0] ;

  do
    A0() ;
  while( y0 != 0 ) ;
}

void A0()
{
  switch( y0 )
  {
case 0:
i = 0 ;                        y0 = 1 ;
break ;
case 1:
```

```
A1() ;
if( y1 == 0 )    i++
else
if( i >= m )           y0 = 0 ;
break ;
 }
}
void A1()
{
  switch( y1 )
  {
case 0:
j = 0 ;               y1 = 1 ;
break ;
case 1:
if( j >= n )          y1 = 0 ;
else
if( a[i][j] > max )
{ max = a[i][j] ; j++ ; }
else
  j++ ;
break ;
  }
}
```

Применяя предлагаемый подход для реализации алгоритма Дейкстры, предназначенного для поиска кратчайших расстояний в положительном взвешенном графе от вершины с номером ноль до всех остальных вершин, удается построенную традиционно структурированную программу, содержащую цикл for, в который вложены два таких же цикла, преобразовать в автоматную программу, состоящую из одной конструкции do-while, телом которой является конструкция switch, реализующая автомат Мили с четырьмя состояниями [5].

В заключение этой части работы отметим, что изложенный подход напоминает известный метод Ашкрофта–Манны [1], предназначенный для структурирования неструктурированных программ, который по этой причине (в отличие от предлагаемого) к структурированным программам не применяется.

## Преобразование неструктурированных схем алгоритмов в автоматные

Предлагаемый подход при структурировании неструктурированных схем алгоритмов является более эффективным по сравнению с методом Ашкрофта–Манны ввиду явного использования автоматов разных типов и меньшего числа состояний в них. Продемонстрируем это на примерах.

Пример 6. Построить по неструктурированной схеме алгоритма (рис. 9), приведенной в [8], автоматную схему.

Используя изложенный метод, построим граф переходов автомата Мили (рис. 10).

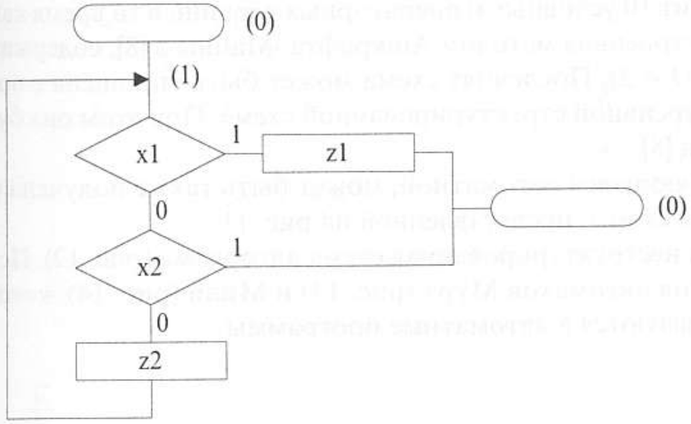


Рис. 9

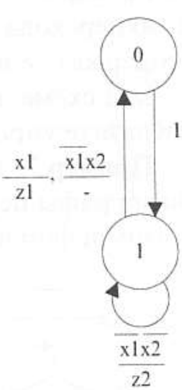


Рис. 10

Автоматная схема алгоритма, являющаяся структурированной, которая изоморфно построена по этому графу с помощью метода, изложенного в [11,13], приведена на рис. 11.

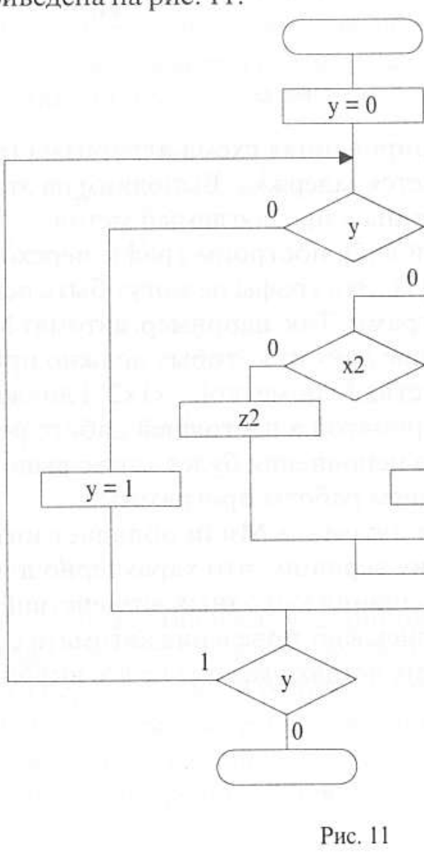


Рис. 11

Эта схема содержит 10 условных и операторных вершин, в то время как аналогичная схема, построенная методом Ашкрофта–Манны в [8], содержит 16 таких вершин ($У = О = 2$). Последняя схема может быть упрощена с помощью перехода к рекурсивной структурированной схеме. При этом она будет содержать 8 вершин [8].

Эта схема, не являющаяся автоматной, может быть также получена и в результате упрощения схемы, представленной на рис. 11.

Пример 7. Задана неструктурированная схема алгоритма (рис. 12). Построим графы переходов автоматов Мура (рис. 13) и Мили (рис. 14), которые изоморфно преобразуются в автоматные программы.

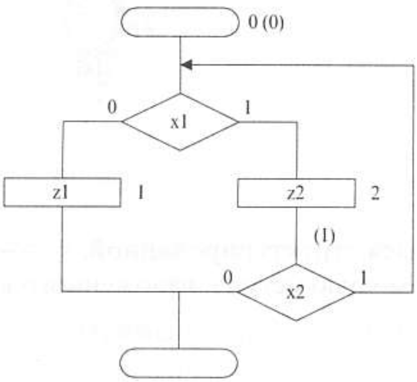


Рис. 12

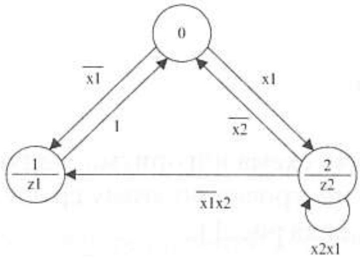


Рис. 13

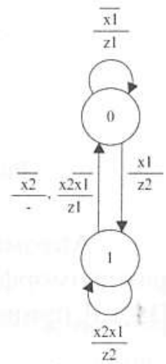


Рис. 14

Пример 8. Задана неструктурированная схема алгоритма (рис. 15), в которой при x1 = x2 = 1 осуществляется задержка. Выполним на этом примере обоснование введения второго этапа в предлагаемый метод.

Применяя метод, изложенный в [2], построим графы переходов автоматов Мура (рис. 16) и Мили (рис. 17). Эти графы не могут быть использованы для построения автоматных программ. Так, например, автомат Мили имеет три петли, однократное исполнение двух из которых должно приводить к завершению алгоритма, а третья петля (с пометкой "x1x2") должна исполняться многократно. Однако при принятой в настоящей работе реализации третья петля вместо многократного исполнения будет также выполнена однократно с последующим завершением работы программы.

Полученный граф переходов автомата Мили обладает интересным свойством: он содержит только одну вершину, что характерно для автоматов без памяти. Однако за счет умолчания выходных воздействий (прочерка) на одной из петель этот граф описывает поведение автомата с памятью. Кроме того, отметим, что и на других петлях указаны не все выходные переменные.

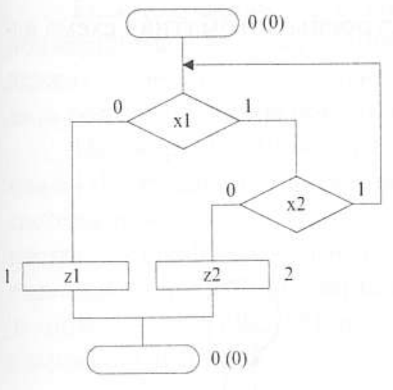


Рис. 15

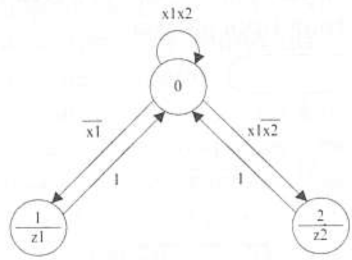


Рис. 16

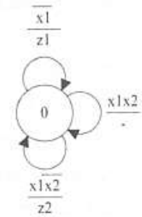


Рис. 17

Пример 9. Задана неструктурированная схема алгоритма (рис. 15). Так как она содержит контур без операторных вершин, замыкающийся на дуге, исходящей из начальной вершины, то в соответствии со вторым этапом предлагаемого метода введем пометку для дополнительного состояния автоматов Мура и Мили сразу после начальной вершины схемы (рис. 18). Отметим, что эта пометка достаточно естественна для построения автомата Мили, но нетрадиционна для автомата Мура.

Построенные графы переходов автоматов Мура и Мили приведены на рис. 19 и 20 соответственно. По этим графам, как изложено выше, могут быть построены автоматные схемы алгоритмов и автоматные программы.

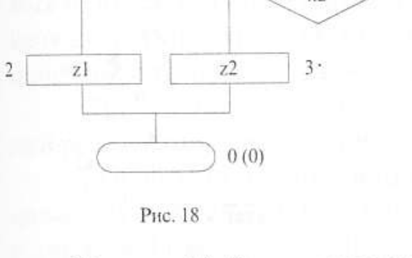


Рис. 18

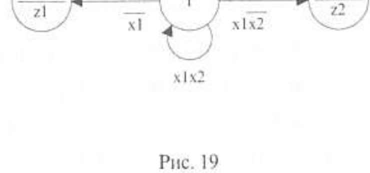


Рис. 19

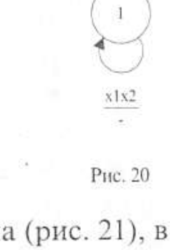


Рис. 20

Пример 10. Задана неструктурированная схема алгоритма (рис. 21), в которой при x2 = x3 = 1 запоминаются значения выходных воздействий, вычисленные в первом блоке. В отличие от предыдущего примера, наличие "пустого" контура в схеме не требует введения дополнительного состояния, так как этот контур замыкается не после начальной вершины. Граф переходов автомата Мили, построенный для этой схемы, приведен на рис. 22. По этому графу, как изложено выше, могут быть построены автоматная схема алгоритма и автоматная программа.

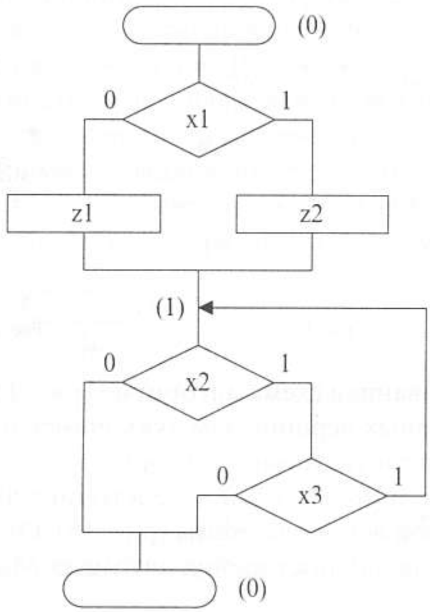


Рис. 21

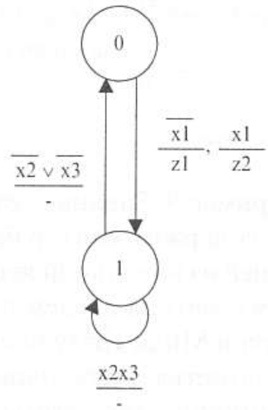


Рис. 22

Пример 11. Задана неструктурированная и непланарная схема алгоритма (рис. 23). Граф переходов автомата Мили, построенный по этой схеме и содержащий всего лишь два состояния, приведен на рис. 24. По этому графу, как изложено выше, могут быть построены автоматная схема алгоритма и автоматная программа.

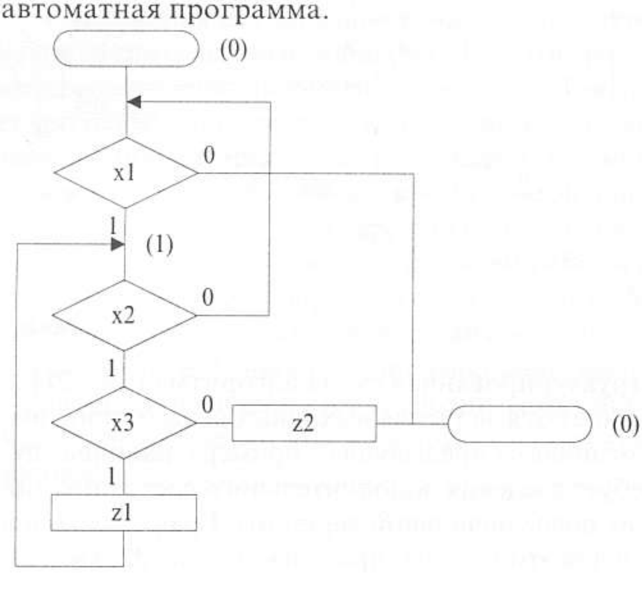


Рис. 23

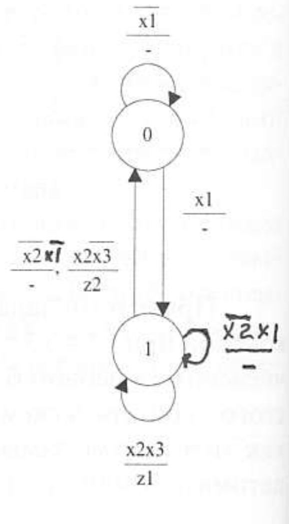


Рис. 24

Рассмотренная схема содержит контур, замыкающийся после начальной вершины и не содержащий операторных вершин. Так как этот контур содержит пометку “(1)”, соответствующую состоянию автомата Мили, то вводить дополнительное состояние не требуется.

Пример 12. Задана неструктурированная схема алгоритма, приведенная в [3] и представленная на рис. 25. Введя пометку для дополнительного состояния в соответствии со вторым этапом предлагаемого метода, можно построить граф переходов автомата Мили с четырьмя состояниями. Продублировав операторную вершину с пометкой “z3” в соответствии с третьим этапом метода (рис. 26), построим граф переходов автомата Мили с тремя состояниями (рис. 27).

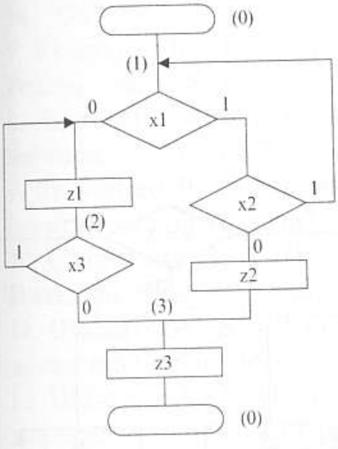


Рис. 25

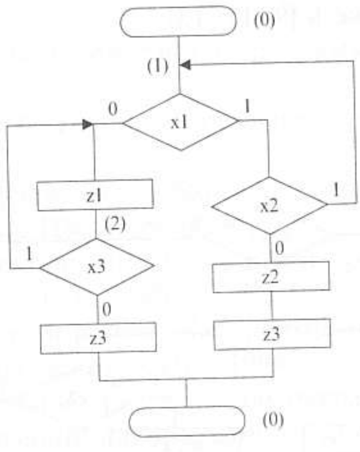


Рис. 26

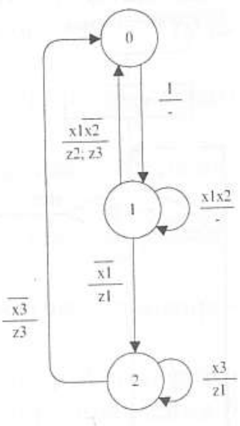


Рис. 27

Автоматная схема алгоритма, изоморфная этому графу, приведена на рис. 28 (с.52).

Построенная структурированная схема алгоритма содержит 16 условных и операторных вершин, в то время как аналогичная схема, построенная методом Ашкрофта–Манны в [3] (с учетом исправления допущенной там ошибки), содержит 23 таких вершины (У = О = 3).

Приведенная на рис. 28 схема соответствует предлагаемой программной реализации с использованием операторов do-while и switch.

Все рассмотренные в примерах схемы алгоритмов и программ имели начальную и конечную вершины, что характерно для вычислительных алгоритмов. Предложенный метод может быть использован также и для преобразования алгоритмов, содержащих бесконечный внешний цикл. При этом граф переходов, построенный по такой схеме, будет обладать свойством недостижимости начальной вершины.

Сопоставление рассмотренных в настоящей работе традиционных схем алгоритмов и графов переходов показывает, что последние более компактны и понятнее специфицируют алгоритм. Это связано с тем, что построение графа переходов сводится к исследованию всех путей между соседними пометками в традиционной схеме алгоритма и компактному их представлению в виде помеченных дуг и петель графа.

Преимущество автоматного подхода по сравнению с традиционным достигается за счет добавления к понятиям "входное и выходное воздействия" понятия "состояние", а по сравнению с методом Ашкрофта–Манны – за счет сокращения числа состояний.

Можно считать, что предложенный подход для рассматриваемых задач обеспечивает преобразование процедурных знаний в декларативные [10].

Другие аспекты взаимосвязи схем алгоритмов и программ с графами переходов рассмотрены в [12] (www.softcraft.ru, раздел "Программирование/Автоматы"), а также в [9, 11, 13].

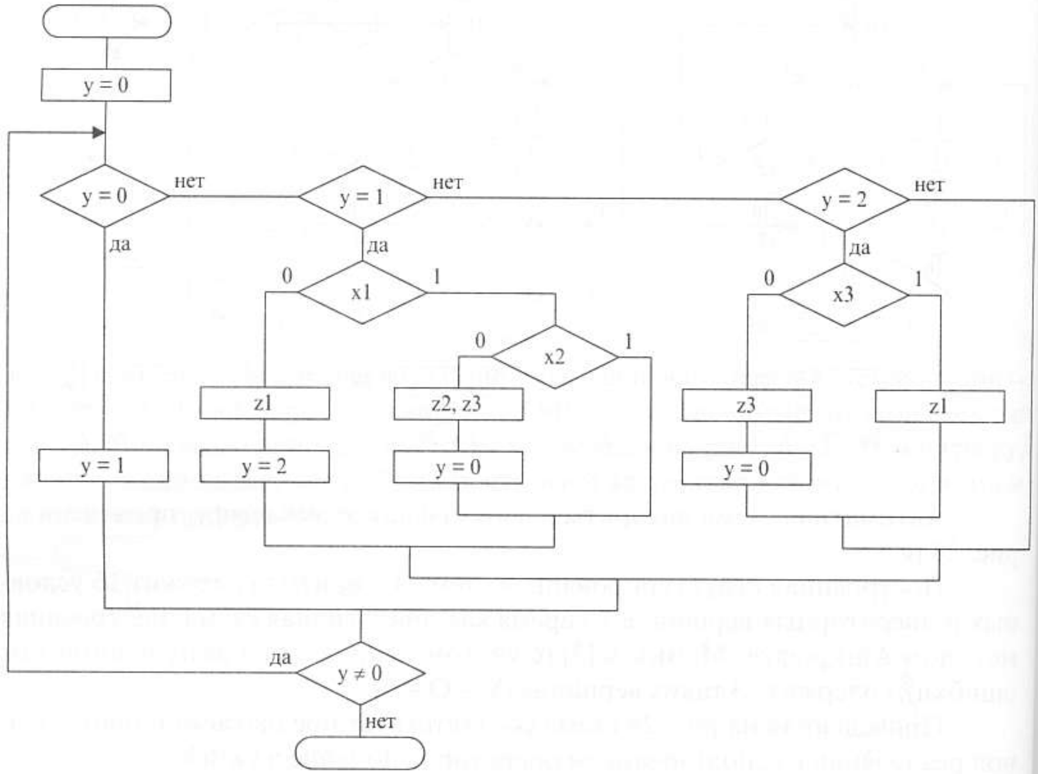


Рис. 28

## Список литературы

1. Aschcroft E., Manna Z. The translation of "goto" program into "while" program // Proceeding of 1971 IFIP Congress.

2. Баранов С. И. Синтез микропрограммных автоматов (граф-схемы и автоматы). Л., 1979.

3. Йодан Э. Структурное проектирование и конструирование программ. М., 1979.
4. Казаков М. А., Столяр С. Е. Визуализаторы алгоритмов как элемент технологии преподавания дискретной математики и программирования // Тезисы докладов Международной научно-методической конференции “Телематика–2000”. СПб., 2000.
5. Казаков М. А., Шалыто А. А., Туккель Н. И. Использование автоматного подхода для реализации вычислительных алгоритмов // Труды Международной научно-методической конференции “Телематика–2001”. СПб., 2001.
6. Кормен Т., Лейзерсон Ч., Ривеста Р. Алгоритмы. Построение и анализ. М., 1999.
7. Кузнецов Б. П. Психология автоматного программирования // BYTE/ Россия. 2000. № 11.
8. Лингер Р., Миллс Х., Уитт Б. Теория и практика структурного программирования. М., 1982.
9. Любченко В. С. Мы выбираем, нас выбирают… (к проблеме выбора алгоритмической модели) // Мир ПК. 1999. № 3.
10. Станкевич Л. А. Интеллектуальные технологии представления знаний. Интеллектуа́льные системы. СПб., 2000.
11. Шалыто А. А. SWITCH-технология. Алгоритмизация и программирование задач логического управления. СПб., 1998.
12. Шалыто А. А. Использование граф-схем алгоритмов и графов переходов при программной реализации алгоритмов логического управления // Автоматика и телемеханика. 1996. № 6, 7.
13. Шалыто А. А. Логическое управление. Методы аппаратной и программной реализации алгоритмов. СПб., 2000.
14. Шалыто А. А., Туккель Н. И. SWITCH-технология – автоматный подход к созданию программного обеспечения “реактивных” систем // Промышленные АСУ и контроллеры. 2000. № 10.