КОМПЬЮТЕРНОЕ МОДЕЛИРОВАНИЕ

УДК 004.424.25

МЕТОД ПОСТРОЕНИЯ БИНАРНЫХ ДИАГРАММ ДЛЯ РЕАЛИЗАЦИИ ТАБЛИЦ РЕШЕНИЙ

В.В. МУРОМЦЕВ
Е.М. МАМАТОВ

Белгородский государственный национальный исследовательский университет

e-mail: muromtsev@bsu.edu.ru

В работе предложен метод программирования таблиц решений, основанный на построении бинарных диаграмм. Отличительной особенностью метода является то, что таблица решений представляется системой логических функций. Затем строится бинарная диаграмма решений, представляющая эту систему, при этом анализируются структурные свойства системы с целью уменьшения числа условных вершин диаграммы.

Ключевые слова: таблица решений, бинарная диаграмма решений, логическая функция.

Постановка задачи

Одной из важнейших характеристик программного обеспечения (ПО) является простота его модификации. Для создания легко модифицируемых программ существуют различные методы и приемы. Одним из таких приемов является использование таблиц решений на начальных этапах проектирования ПО. Такие таблицы позволяют в компактной форме установить связь между условиями и действиями. Пример представлен в табл. 1.

Таблица 1

Специальный клиент (C_1)	*N*	*N*	*Y*	*
Приоритетный заказ (C_2)	*N*	*Y*	-	*
Международный заказ (C_3)	-	*Y*	-	*
Плата	150	100	70	80
Код предупредительного сигнала	0	0	1	2

Таблица решений разделена на две части двойной чертой. Выше этой черты в первом столбце перечислены логические условия, а все остальные столбцы задают варианты выполнения условий. Условие может быть выполнено (Y или 1), не выполнено (N или 0) или не важно (-). Ниже двойной черты в первом столбце перечислены действия, а все остальные столбцы определяют надо или не надо выполнять соответствующее действие для варианта выполнения условий. Так в приведенном примере второй столбец определяет следующее правило - если клиент не специальный и заказ не приоритетный, то пла-

та составит 150 единиц и сигнал не выдается (код сигнала равен 0). Третий столбец определяет правило - если клиент не специальный и заказ приоритетный и международный, то плата составит 100 и сигнал не выдается. Четвертый столбец определяет, что для специального клиента при любом заказе плата 70, кроме того, если заказ делает специальный клиент, то выдается предупредительный сигнал с кодом 1. Последний столбец определяет правило «иначе», т.е. во всех остальных случаях плата составит 80 и выдается предупредительный сигнал с кодом 2.

В рассмотренном примере все условия логического типа. Такие таблицы называют «таблицы с ограниченным входом». Также используются таблицы, в которых условия не являются логическими, например, условие может быть перечисляемого типа с числом элементов больше двух. Такие таблицы называют «таблицы с расширенным входом». Далее речь будет идти о таблицах с ограниченным входом.

Кроме таблиц решений для определения связи между условиями и действиями также часто используют бинарные диаграммы решений (БДР) [1] или соответствующие им блок-схемы, диаграммы деятельности, конечные автоматы. Преимущества БДР в том, что по ним легко синтезировать программы. Однако таблицы решений обладают рядом преимуществ перед БДР. Так при описании сложных условий таблица решений компактнее и проще для понимания. Поэтому использование в проектах таблиц решений позволят сократить разрыв между программистами и их заказчиками. Таблицы решений могут успешно сочетаться с другими средствами описания проектов, например, с диаграммами UML. Также следует отметить, что таблицы решений являются более общей формой, чем БДР. Так таблица решений, описывающая некоторую ситуацию, содержит в себе все возможные для этой ситуации БДР.

Таким образом, при определении связи между условиями и действиями удобнее использовать таблицы решений. Далее следует выполнить программирование таблицы решений, т.е. нужно разработать программу, реализующую определенную связь между условиями и действиями. Тривиальный способ написания такой программы сводится к реализации каждого варианта выполнения условий с помощью условного оператора. Однако такой способ может оказаться неприемлемым, т.к. потребует большого числа проверок условий.

Задачу программирования таблиц можно решить путем синтеза логически эквивалентной БДР, отвечающей заданным критериям, и потом синтезировать программу, реализующую полученную БДР. В статье рассматривается один из подходов к синтезу БДР, при этом преследуется цель уменьшения числа условных вершин диаграммы.

Представление таблицы решений системой логических функций

Таблицу решений будем рассматривать как четверку $T = (C, A, S, O)$, где C - множество условий, A - множество действий, $S : W^n \to A_1 \times \ldots \times A_k$ - функция, задающая для каждого варианта выполнения условий необходимые действия, $n = |C|$ - число условий, $k = |A|$ - число действий, $W = \{"N","MY","M-"\}$, $A_i, i = 1,2,\ldots,L$ _ множества возможный значений для i-го действия, $O \in A_1 \times \ldots \times A_k$ - правило «иначе», т.е. действия, которые необходимо выполнить если не подходит ни один из определенных вариантов выполнения условий.

Пусть X и $Y = \{y_1, y_2, \ldots\}$, $|X| - |Y|$ - область определения и область значений функции S соответственно. Тогда алгоритм представления таблицы решений системой логических функций в дизъюнктивной нормальной форме будет следующий:

1. Для $i = 1,\ldots,m$, где $m = |Y|$ выполнить:
   - Найти $P = \{x \mid S(x) = y_i\}$ – прообраз элемента y_i.
   - Для каждого $x = (x_1,\ldots,x_n) \in P$ выписать элементарную конъюнкцию $c_1^{\sigma_1} \cdot \ldots \cdot c_n^{\sigma_n}$, где $c_j \in C$; $\sigma_j = 1$, если $x_j = Y$; $\&j = o$, если $x_j = N$ и буква $c_j^{\sigma_j}$ отсутствует в конъюнкции, если $x_j = "-"$.

- Все выписанные элементарные конъюнкции объединить дизъюнкцией. В результате получим логическую функцию f_i.

2. Все логические функции f_i, $i = 1,..,m$ объединить в систему. Для каждой функции f_i запомнить соответствующие действия y_i.

Представим в виде системы логических функций рассмотренную ранее таблицу решений. Обозначим условие «Специальный клиент» через c_1, «Приоритетный заказ» - c_2, «Международный заказ» - c_3. После таких обозначений $C = \{c_1, c_2, c_3\}$. Также для этого примера $A_1 = \{150,100,70,80\}$, $A_2 = \{0,1,2\}$, $O = (80,2)$, $S = \{(("N","N","-"), (50,0) (("N","Y","Y"), (100,0)), (("Y","-","-"), (70,1)$

Область определения функции S : $X = \{("N","N","-"), ("N","Y","Y"), ("Y","-","-")\}$.

Область значений функции S: $Y = \{(150,0), (100,0), (70,1)\}$. В данном примере $|X| = |Y|$, но как было отмечено выше в общем случае $|X| \le |Y|$.

Для $i = 1$ прообраз $y_1 = (150,0)$ будет следующим $P = \{("N","N","-")\}$. Множество P состоит из одного элемента - вектора $("N","N","-")$. Этому вектору соответствует следующая элементарная конъюнкция - $c_1^0 \bullet c_2^0 = \bar{c}_1 \bullet \bar{c}_2$. В результате получили логическую функцию $f = \bar{c}_1 \bullet \bar{c}_2$. Выполнив аналогичные действия для $i = 2,3$, получим систему из трех логических функций: $f_1 = \bar{c}_1 \bullet \bar{c}_2$, $f_2 = \bar{c}_1 \bullet \bar{c}_2 \bullet c_3$, $f_3 = c_1$.

Построение бинарной диаграммы решений

Пусть исходной таблице решений соответствует система $X = \{f_1, \ldots, f_m\}$ логических функций от n переменных. БДР, представляющей систему X будем называть ациклический граф, в котором:

1. Имеется ровно одна вершина без входящих в нее дуг (начальная вершина).
2. Все вершины по числу выходящих дуг делятся на два типа:
   - вершины, помеченные переменной c_j, $j \in \{1,...,n\}$, из которых выходит две дуги, помеченные символами 0 и 1 (условные вершины);
   - вершины, помеченные вектором $(a_1, \ldots, a_n)$, $a_i \in \{0,1\}$, $i = 1,...,m$, и не имеющие выходящих дуг (заключительные вершины).

Как было отмечено выше, задача программирования таблиц решений сводится к поиску логически эквивалентной БДР, отвечающей заданным критериям. Возникает вопрос - можно ли осуществить данный поиск путем перебора всех возможных БДР? Оценим трудоемкость такого поиска.

Для системы логических функций (таблицы решений) с одной логической переменной (с одним условием) возможна БДР единственной структуры первого уровня - $P(1) = 1$. Диаграмма, имеющая структуру первого уровня ($n = 1$) представлена на рис.1а.

Диаграмма, имеющая структуру второго уровня ($n = 2$) представлена на рис.1б. Такую диаграмму можно рассматривать как начальную вершину, из которой выходят дуги в диаграммы, имеющие структуру первого уровня. Поскольку в качестве переменной, анализируемой в начальной вершине можно взять любую из двух переменных, а структура диаграммы первого уровня единственная, то число возможных БДР со структурой второго уровня равно $P(2) = 2 \cdot 1 \cdot 1 = 2 \bullet P(1)^2 = 2$.

Структура БДР третьего уровня ($n = 3$) представлена на рис.1в. Такую диаграмму можно рассматривать как начальную вершину, из которой выходят дуги в диаграммы, имеющие структуру второго уровня. Поскольку в качестве переменной, анализируемой в начальной вершине можно взять любую из трех переменных, а число диаграмм имеющих структуру второго уровня равно $P(2) = 2$, то число возможных БДР со структурой третьего уровня равно $P(3) = 3 \bullet 2 \bullet 2 = 3 \bullet P(2)^2 = 12$.

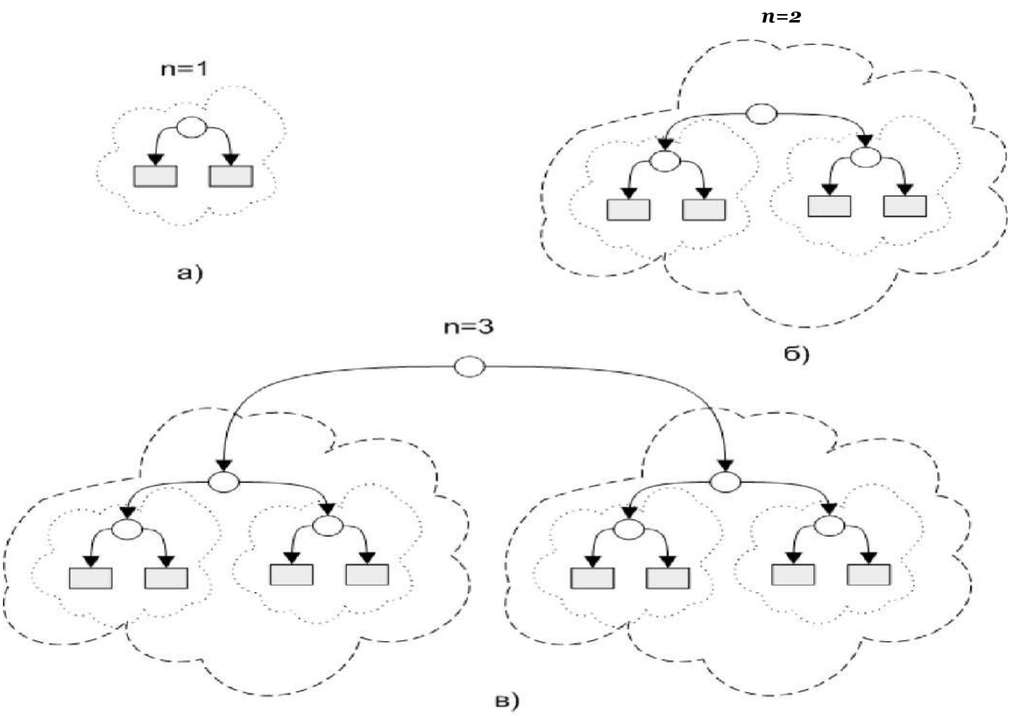


Рис. 1. Структуры БДР

Таким образом, $P(n)$ - число возможных БДР, имеющих структуру n -го уровня, т.е. БДР с n логическими переменными выражается следующим рекуррентным соотношением:

1) $P(1) = 1$;

2) $P(n) = n \cdot P(n-1)^2$.

Уже при $n = 7$ число $P(7)$ превысит $19 \cdot 10^{26}$. Поэтому поиск некоторой оптимальной БДР путем полного перебора всех вариантов невозможен.

Известны методы, позволяющие находить точное решение без обязательного полного перебора всех вариантов. Однако эти методы в некоторых случаях все равно приводят к необходимости проверки всех возможностей и к комбинаторному взрыву, что ограничивает их применение на практике. Поэтому больший интерес представляет вопрос разработки эвристических методов построения рациональных БДР, позволяющих получить результат за приемлемое время. Такие методы также существуют. В [2] можно найти обзор методов построения блок-схем логически эквивалентных таблицам решений.

Следует отметить, что существующие эвристические методы, как правило, отличаются от точных методов тем, что на каждом уровне дерева поиска некоторым образом ограничивается число исследуемых вариантов.

В работе предлагается метод, входом которого является не таблица решений, а эквивалентная ей система логических функций. Далее для этой системы строится БДР, при этом анализируются структурные свойства системы с целью уменьшения числа условных вершин диаграммы.

В основу алгоритма построения БДР, представляющего систему X положено разложение каждой функции системы X по некоторой переменной c_j, $j \in \{1,...,n\}$:

/г (0) = /г к – С;-:А $c_{J+1},...,c_n$ b /г () = f $c_{j-1}, c_{j+1},...,c_n$), * = m. Такое разложение отображается на графе (V, E), где $V = \{a, b, c\}$, $E = \{(a, b), (a, c)\}$ следующим образом: вершина a помечается системой X и символом c_j, вершины b и c помечаются систе-

мами $X^0 = \{ (0),..., f_m(0)\}$ и $X^1 = \{(1),..., f_m(1)\}$ соответственно. Кроме того, дуга (a, b) помечается символом «0», а дуга $^{(a,c)}$ - символом «1».

Пусть в результате разложения получена некоторая система $\{<,...,<\}$, < е {0,1}, $i = 1,...,m$, т.е. получена заключительная вершина БДР, тогда справедливы следующие утверждения:

1. Если < = 1, p е {1,..., m} и < = 0, $i = 1,..., m$, i Ф p, то получена заключительная вершина, соответствующая действиям y_p.
2. Если < = 0, $i = 1,..., m$, то получена заключительная вершина БДР, соответствующая правилу «иначе».
3. Если < = 1, <z_z = 1, p,r е {1,..., m}, то исходные данные содержат ошибку (не ясно, какое из действий ^{y}p или $Уг$ следует выполнить).

Учитывая эти утверждения, разработан базовый алгоритм построения БДР, представляющей систему логических функций (рис. 2). В этом алгоритме остается открытым вопрос выбора переменной разложения. Стратегия выбора может меняться от исходных данных и заданных критериев оптимальности БДР. Например, в качестве дополнительных данных для каждой комбинации условий в таблице решений может быть задана вероятность появления данной комбинации и заданы затраты на проверку каждого условия. В этом случае в качестве критерия оптимальности может выступить минимум затрат на принятие решений.

В рассматриваемом методе преследуется цель уменьшения числа условных вершин в БДР, что приводит к сокращению, как требуемой памяти, так и к снижению времени выполнения соответствующей программы. Число вершин в БДР зависит от выбора переменных разложения. Для уменьшения вершин предложена стратегия, заключающаяся в выборе переменной разложения $^{v}1$ е C, которая соответствует вершине двудольного графа G с наибольшим числом инцидентных ребер. Граф $G = (V, E)$, строится следующим образом: $V = C$ u X - множество вершин, а множество дуг E формируется по правилу: {vj, v_2}е E если v_1 е C, v_2 е X и буква vf, <е {0,1} есть в записи функции $^{v}2$.

Для графа G справедливы следующие утверждения:

1. После выполнения разложения системы X по переменной c,, граф G трансформируется в граф $G' = G - c$,.
2. Каждой заключительной вершине в БДР соответствует изолированная вершина в графе G.

Рассмотрим пример применения предложенной стратегии для рассматриваемой таблицы решений (табл.1), которой соответствует система из трех логических функций $^{f} = \bullet C_2$, $^{f_2} = {}_{c_1} \bullet {}_{c_2} \bullet {}_{c_3}$, $^{f_3} = {}_{c_1}$.

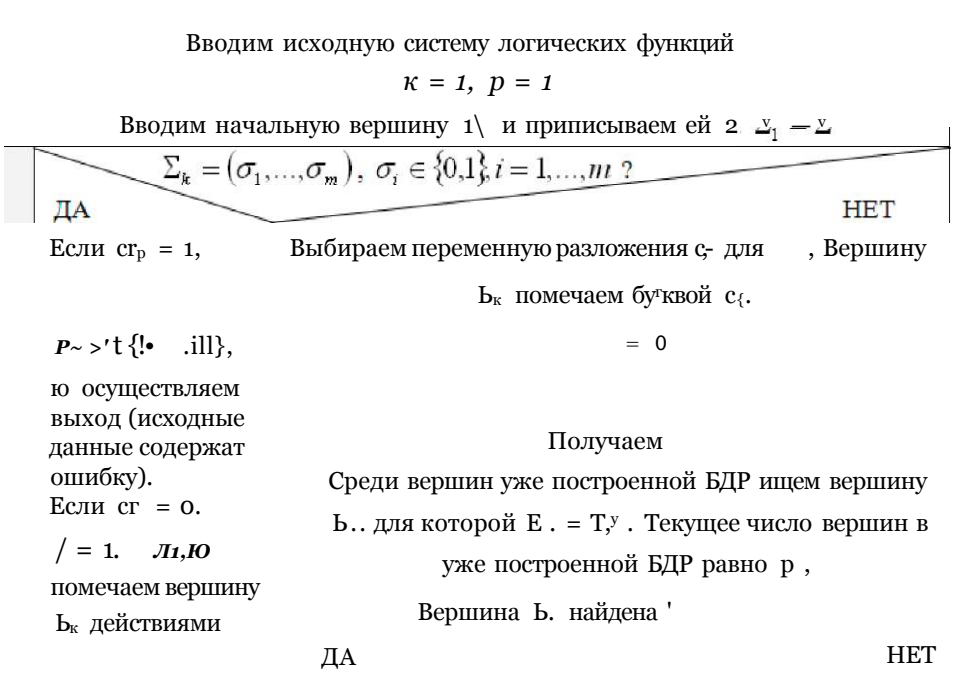


Рис. 2. Базовый алгоритм построения БДР

Граф G, соответствующий этой системе представлен на рис. за. На первом шаге в качестве переменной разложения выбирается переменная C_1, т.к. соответствующей ей вершине инцидентно наибольшее число ребер. После выполнения разложения по c_1, граф принимает вид как на рис.зб. Таким образом, на втором шаге в качестве переменной разложения следует выбрать c_2.

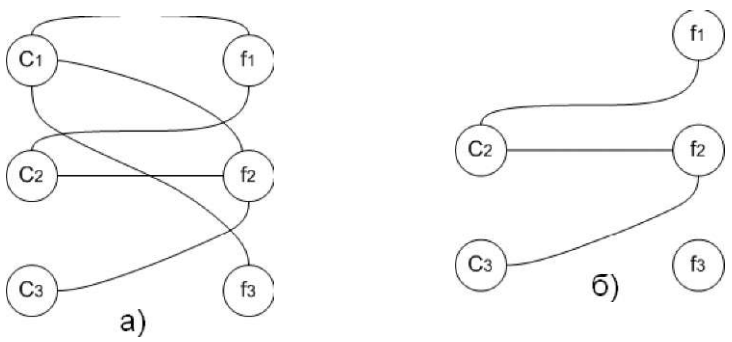


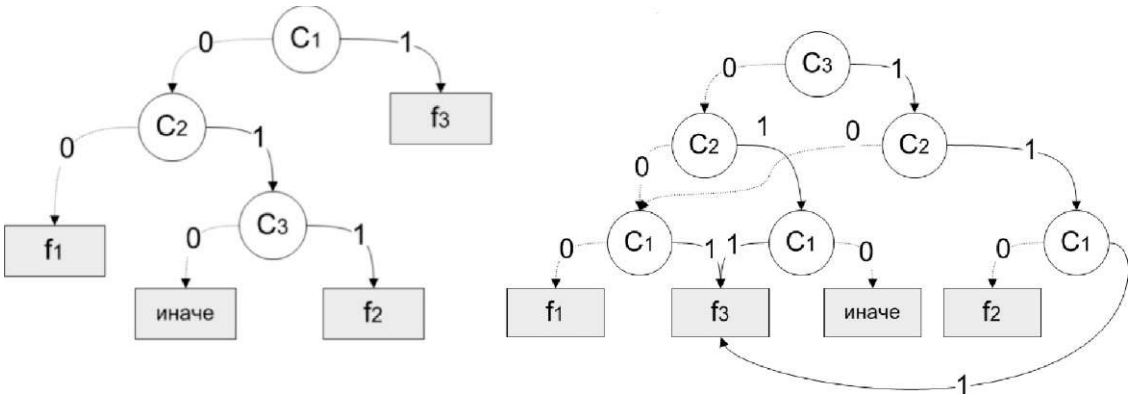
Рис. 3. Преобразование графа G

Итак, в этом примере анализ структуры системы логических функций показал следующий порядок выбора переменных разложения: c_1, c_2, c_3. При таком порядке получается БДР, состоящая из трех условных вершин (рис.4а).

Для сравнения на рис.4б показана неудачная БДР, состоящая из 6 условных вершин. Данная БДР получилась при следующем порядке выбора переменных разложения: c_3, c_2, c_2, c_1, c_1, c_1

БДР, представленной на рис.4а, соответствует следующая программа:

```
/1/   if      then    {toll=70;  signal=i} else
/2/   if c2   then
/3/           if c3   then    {toll=100;  signal=0 }
              else    {toll=80;  signal=2 }
      else    {toll=150;   signal=0  }
```



а) б)

Рис. 4. Бинарные диаграммы решений

В этой программе каждой условной вершине БДР соответствует условный оператор, в котором анализируется единственное условие.

Тривиальная программная реализация рассматриваемой таблицы решений будет следующей:

```
/1/   if c1 • c2          then   {toll=150;   signal=0  }      else
/2/   if c1 • c2 • c3     then   {toll=100;   signal=0  }      else
/3/   if c1               then   {toll=70;    signal=1}
                          else   {toll=80;    signal=2  }
```

В представленных программах каждому условному оператору присвоен номер - /1/, /2/, /3/, который будет использоваться в табл. 2.

Таблица 2

Условия			Число логических проверок					
			Тривиальная программная реализация таблицы решений			Программная реализация БДР на рис.4а		
c_1	c_2	c_3	/1/	/2/	/3/	/1/	/2/	/3/
0	0	0	2			1	1	
0	0	1	2			1	1	
0	1	0	2	3	1	1	1	1
0	1	1	2	3		1	1	1
1	0	0	1	1	1	1		
1	0	1	1	1	1	1		
1	1	0	1	1	1	1		
1	1	1	1	1	1	1		
ИТОГО			27			14		

В этой таблице сведены результаты выполнения условных операторов для всех возможных вариантов условий. При подсчете числа логических проверок h, $i = 1,\ldots,2^n$ подразумевалось, что вычисление логических условий осуществляется по короткой схеме. Видно, что при тривиальной программной реализации таблицы решений для анализа всех возможных вариантов выполнения условий требуется 27 проверок, а при программной реализации БДР, представленной на рис 4а, достаточно 14 проверок.

Для оценки качества БДР с точки зрения числа логических проверок полезно следующее утверждение: максимальное число логических проверок в БДР для всех возможных вариантов условий равно $2^n \cdot n$, где n - число условий (переменных в исходной системе логических функций).

При реализации БДР, представленной на рис.4б, требуется максимальное число логических проверок. Это число равно 24, что меньше, чем число проверок, требующихся при тривиальной реализации таблицы решений.

Для оценки качества программ, реализующих таблицы решений, кроме минимальности числа проверок для всех возможных вариантов условий можно использовать и другие критерии. Например, различным вариантам условий могут быть заданы вероятности их возникновения p_i, $i = 1,\ldots,2^n$. Тогда в качестве критерия можно использовать $\sum_{i=1}^{2^n} p_i \cdot \ldots \to \min$. Также различным условиям может быть задана некоторая весовая функция.

Выводы

В работе предложен метод программирования таблиц решений. Отличительной особенностью метода является то, что вначале таблица решений представляется системой логических функций. Далее строится БДР, представляющая эту систему, при этом контролируются ошибки в исходной таблице решений и анализируются структурные свойства системы с целью уменьшения числа вершин в БДР. Полученная БДР реализуется программно. Данная реализация не представляет трудности. В примере, рассмотренном в работе, при программной реализации БДР использовался компиляционный подход. При таком подходе для каждой БДР строится своя программа, в которой каждой условной вершине диаграммы соответствует условный оператор. Также для реализации БДР можно использовать интерпретационный подход. В этом случае программа единая. Работа такой программы заключается в перемещении от начальной вершины БДР до заключительной вершины. Перемещение осуществляется в зависимости от условия в текущей вершине. Такой подход может оказаться предпочтительным при аппаратной реализации БДР.

Предложенный метод основан на анализе структурных свойств системы логических функций. Также следует отметить, что при построении БДР структурный анализ имеет смысл проводить не только для всей системы логических функций, но и для отдельных функций, входящих в эту систему. Это также позволяет сокращать число условных вершин в БДР. Однако этот вопрос, как и вопрос использования других критериев оптимальности выходит за рамки данной статьи.

Исследование выполнено в рамках Государственного задания Министерства образования и науки РФ на выполнение НИР подведомственным вузам в 2013 году. Проект № 8.8600.2013.

Список литературы

1. Knuth D. Fun With Binary Decision Diagrams (BDDs). [Электронный ресурс]: видео лекция. - Режим доступа: - http://myvideos.stanford.edu/player/slplayer.aspx?coll=ea60314a-53b3-4be2-8552-dcf190ca0c0b&co=18bcd3a8-965a-4a63-a516-a1ad74af1119&o=true, свободный.

2. Хамби Э. Программирование таблиц решений. - Пер. с англ. - М.:Мир, 1976.

THE METHOD OF CONSTRUCTING BINARY DECISION DIAGRAM FOR THE PROGRAMMING OF DECISION TABLES

V.V. MUROMTSEV
E.M. MAMATOV

Belgorod State National Research University

e-mail: muromtsev@bsu.edu.ru

The paper proposes a method of programming decision tables based on building the binary decision diagram. A distinctive feature of the method is that the table of solutions of a system of logical functions. Then builds the binary decision diagram solutions representing this system. By analyzing the structural properties of the system to reduce the number of conditional tops the diagram.

Keywords: decisions table, binary decision diagram, logic function.