

**Министерство науки и образования
Российской Федерации
Пензенский государственный университет**

Н.П.Вашкевич

НЕДЕТЕРМИНИРОВАННЫЕ АВТОМАТЫ В ПРОЕКТИРОВАНИИ СИСТЕМ ПАРАЛЛЕЛЬНОЙ ОБРАБОТКИ

**Рекомендовано учебно-методическим объединением вузов
по университетскому политехническому образованию
в качестве учебного пособия для студентов,
обучающихся по направлению 654600 –
«Информатика и вычислительная техника»**

**Издательство
Пензенского государственного университета
ПЕНЗА 2004**

УДК 519.713

В23

Рецензенты:

Кафедра «Вычислительные машины и системы»
Пензенской государственной технологической академии

Кандидат технических наук, доцент,
начальник кафедры «Артиллерийское вооружение и информатика»
Пензенского артиллерийского инженерного института
П.И.Тобольченко

Н.П.Вашкевич Недетерминированные автоматы в проектировании систем параллельной обработки [Текст]/Н.П.Вашкевич: учебное пособие. – Пенза: Изд-во Пенз.гос.ун-та, 2004. – 280 с.: 50 ил., 24 табл., библиогр. 84 назв.

Представлен основной математический аппарат, связанный с формальным представлением параллельных управляющих алгоритмов в виде модели недетерминированных автоматов (НДА) и эквивалентные преобразования этой модели. В качестве основной модели НДА принята система рекуррентных канонических уравнений, описывающих все реализуемые в системе управления частные события и язык граф-схем алгоритмов с параллельными ветвями. Эквивалентные преобразования модели НДА базируются на использовании операций детерминизации, минимизации, кодирования и композиции НДА. Рассмотрены методы формального описания алгоритмов взаимодействия параллельных процессов, связанных с их синхронизацией и обменом информацией. Формальные представления алгоритмов взаимодействия процессами иллюстрируются на базе решения «классических» задач управления процессами: обращение к общему ресурсу для n процессов, производители-потребители, читатели-писатели, обедающие философы, а также задач идентификации языков. Рассмотрены также методы структурной реализации параллельных алгоритмов.

Пособие подготовлено на кафедре «Вычислительная техника», предназначено для студентов специальности 220100 при изучении ими дисциплин «Теория автоматов» и «Недетерминированные автоматы в системах логического управления», а также может быть использовано студентами всех специальностей направления «Информатика и вычислительная техника» при изучении ими родственных дисциплин.

Учебное пособие подготовлено при выполнении фундаментальной НИР по заказу Минобразования и науки РФ, шифр Т02-03,3-2476.

УДК 519.713

Издательство Пензенского государственного
университета, 2004
Вашкевич Н.П., 2004

ПРЕДИСЛОВИЕ

Повышение производительности вычислительных машин и систем автоматики на их основе во многом определяется эффективностью организации логического управления в системах, учитывающих реализацию параллелизма на всех уровнях управления. Особенно это относится к мультипроцессорным системам, в том числе к системам реального времени, предназначенным для управления в робототехнических системах, когда возникают задачи управления параллельными взаимодействующими процессами и ресурсами. При создании таких систем одной из актуальных задач является разработка формальных методов описания алгоритмов управления параллельными взаимодействующими процессами, их эквивалентных преобразований и методов их структурной реализации на разных уровнях.

Настоящее учебное пособие посвящено вопросам синтеза таких систем логического управления, который базируется на использовании моделей недетерминированных конечных автоматов (НДА) и их стандартной аналитической форме представления в виде системы рекуррентных канонических уравнений, описывающих все реализуемые в системе управления частные события.

В такой математической модели НДА в отличие от модели детерминированного автомата (ДА) предложено использовать представления функций переходов в автомате не в терминах состояний, а в терминах частных событий, реализуемых в автомате. При этом каждому НДА может быть поставлен в соответствие эквивалентный ему детерминированный автомат, в котором состояния в общем случае будут представлены совокупностью событий, имеющих место в один и тот же момент времени в НДА. Т.е. ДА является частным случаем НДА, в котором не соблюдаются условия однозначности переходов, проявляющейся в том, что под действием одного и того же входного сигнала возможен переход от некоторого исходного события к нескольким событиям, совокупность которых и будет представлять собой одно из состояний ДА.

Таким образом, «недетерминированность» автомата нельзя смешивать со «случайностью», характеризующую вероятностный автомат, в котором автомат может перейти в одно из следующих состояний с фиксированными вероятностями [6]. Указанная особенность модели НДА и определяет «скрытый» в ней параллелизм.

Такой подход к формализации сложных параллельных алгоритмов управления с взаимодействующими ветвями позволяет создать набор методов и средств для компактного формального описания алгоритмов управления и их эквивалентных преобразований для построения возможных реализаций с учетом требований повышения производительности, надежности и уменьшения аппаратных затрат.

Математические модели устройств управления, принятые в работе и основанные на использовании концепции параллелизма и недетерминизма, являются достаточно простыми и универсальными, позволяющими их использовать для широкого круга применения как по управлению процессами и ресурсами, так и для моделирования событий, начиная от простейших устройств и кончая функциями управления операционных систем.

В первых трех главах учебного пособия рассмотрены основные понятия и определения из теории недетерминированных автоматов (НДА) и методы их эквивалентных преобразований на основе операций детерминизации, минимизации кодирования и композиции.

В четвертой главе рассматриваются начальные языки, используемые для представления управляющих алгоритмов параллельной обработки. К ним относятся: язык регулярных выражений алгебры событий (РВАС), язык исчисления предикатов первого порядка с ограниченными кванторами и его связь с языком РВАС, язык операторных схем алгоритмов с параллельными ветвями. Рассматриваются также и обычные (непараллельные) языки – язык граф-схем алгоритмов (ГСА) и язык логических схем алгоритмов (ЛСА), т.к. они являются основой для построения параллельных языков. Для каждого начального языка рассматривается методика преобразования описания на начальном языке на описание на стандартном языке в виде системы канонических уравнений для всех событий, реализуемых в алгоритме управления.

Отличительной особенностью формализации управляющих алгоритмов на начальных языках заключается в том, что в качестве входного алфавита используется множество элементарных двоичных сигналов. При этом реализуемые в автомате события могут зависеть либо от всех, либо от части элементарных входных сигналов и эта зависимость событий от элементарных входных сигналов на разных шагах работы алгоритма управления может быть различной. Поэтому в качестве входных сигналов на каждом шаге работы алгоритма управления используются частные входные входы, которые образуются сочетаниями только тех элементарных двоичных входных сигналов, которые действуют на данном переходе. Это обстоятельство, наряду с использованием для описания управляющих алгоритмов частных событий, а не состояний, позволяет значительно уменьшить критичность формального описания к ее размерности, определяемой как произведение состояний системы на число входных сигналов.

В пятой главе рассматриваются вопросы структурной реализации алгоритмов логического управления, заданных моделью НДА. При этом отмечаются недостатки классических методов структурного синтеза систем управления на основе использования моделей детерминированных конечных автоматов. Обращается особое внимание структурной реализации систем управления на основе разбиения частных событий, реализуемых в автомате,

на группы несовместимых событий, что позволяет построить распределенные системы параллельной обработки. Рассматриваются также методы одноуровневой и двухуровневой структурной реализации систем управления для унитарного способа кодирования частных событий.

Шестая глава посвящена вопросам формализации алгоритмов управления взаимодействующими процессами для организации параллельной обработки информации. В главе в начале представлена методика формализации простейших базовых структур управления взаимодействующими процессами с использованием стандартного языка НДСКУ. На основе базовых структур управления показана методика формализации главной функции управления взаимодействующими процессами – взаимоисключение критических интервалов (участков), обеспечивающих доступ к общим разделяемым данным (общему ресурсу) для 2-х и n процессов. В последующих разделах главы рассмотрены методы формализации алгоритмов управления взаимодействующими параллельными процессами, связанные с решением задач синхронизации процессов и обменом сообщениями. Методы формализации алгоритмов управления процессами иллюстрируются на примерах решения известных «классических» задач: обращение к общему ресурсу для n процессов, производители-потребители, читатели-писатели, обедающие философы. По каждой задаче представлены все основные уравнения, определяющие реализуемые в алгоритме частные события, которые могут быть взяты за основу для практической структурной реализации системы управления.

В седьмой главе рассматриваются вопросы проектирования алгоритмов для решения задач идентификации языков. Представлена формулировка различных задач распознавания цепочек-образов и методика разработки алгоритмов для решения этих задач для однопроцессорной и многопроцессорной реализации. Разработка параллельных алгоритмов идентификации языков сопровождается соответствующим примером.

В разделе «Заключение» приведены наиболее важные достоинства способа формализации алгоритмов логического управления на разных уровнях преобразования информации в вычислительных системах на основе использования модели НДА и ее аналитического представления на языке НДСКУ в виде систем канонических уравнений, описывающих все реализуемые в алгоритме управления частные события.

При создании учебного пособия предполагалось, что читателю известны основные положения булевой алгебры, теории множеств, а также методы минимизации булевых функций и основные понятия конечных цифровых автоматов и синтеза комбинационных схем. Работа имеет прикладной характер, поэтому в ней отсутствуют какие-либо теоремы, связанные с теорией эквивалентных преобразований. В связи с этим все основные положения, излагаемые в пособии, иллюстрируются соответствующими примерами.

Учебное пособие написано по материалам научных работ автора в области теории НДА и систем логического управления и материалов лекций по дисциплинам: «Прикладная теория цифровых автоматов», «Теория автоматов» и «Недетерминированные автоматы в системах логического управления». Глава 7 написана совместно с Е.И. Калиниченко. Автор выражает глубокую благодарность и признательность коллегам по работе, которые критически прочитали различные части рукописи и внесли полезные предложения. Особо хотелось бы поблагодарить Е.И.Калиниченко, В.С.Князькова, В.Г.Пучкова, Н.Н.Коннова, С.А.Зинкина, М.М.Бутаева, А.В.Никишина.

Выражаю особую благодарность О.Н.Сергеевой-Дружининой и А.В.Прошкину за быструю и качественную перепечатку рукописи.

Глава 1. Основные понятия и определения из теории недетерминированных автоматов

1.1. Определение и понятия недетерминированного автомата

Недетерминированные автоматы являются общим случаем конечных детерминированных цифровых автоматов, основные понятия и определения которых рассмотрены во многих источниках, в том числе и в [1, 2].

Под НДА понимается математическая модель устройства переработки цифровой (дискретной) информации, для которой не выполняются условия однозначности функций переходов. Кроме того, для НДА, как и для детерминированного конечного цифрового автомата, могут не соблюдаться также и условия полноты переходов. Термин «недетерминированный», как было отмечено в [3], наводит на мысль как о чем-то случайном, однако в НДА ничего случайного нет. В данном случае под НДА понимается просто формализм для определения (распознавания) множества цепочек (слов), а слово «автомат» в названии НДА присутствует потому, что формализм является обобщением формализма, используемого для определения обычного детерминированного конечного автомата. Таким образом, НДА не следует смешивать с вероятностным автоматом, в котором переходы осуществляются в любое из состояний с некоторой вероятностью.

Понятие НДА было впервые введено в работах американских ученых М.О.Рабина и Д. Скотта [4]. В дальнейшем понятие НДА успешно использовалось в работах, связанных с решением теоретических вопросов информатики, например, в [5,6,7]. Однако в этих работах НДА рассматривались в основном с точки зрения использования такой математической модели для построения различных распознавателей и компиляторов. В частности, в работах [5, 8] НДА были представлены под названием «источники». При этом под источником понимается ориентированный граф, однозначно определяющий событие S в некотором алфавите $[Z]$, порождаемое множеством всех путей из начальных вершин в заключительные.

В работах [9] и [10] НДА был представлен системой предикатных формул на языке исчисления предикатов 1-го порядка, а также системой бескванторных предикатных формул, формализующих все реализуемые частные события алгоритма функционирования цифрового автомата, полученных после спуска кванторов.

В дальнейших работах автора [2,11] модель НДА, представленная в виде стандартной и универсальной системы рекуррентных бескванторных предикатных формул, названная системой канонических уравнений недетерминированного автомата (НД SKU), стала основой для синтеза систем логического управления процессами и объектами.

Использование понятий НДА как математической модели в виде НДСКУ, как будет показано ниже, является эффективным средством для формализации описания сложных алгоритмических процессов управления, в том числе алгоритмов управления с параллельными взаимодействующими ветвями. Используя различные эквивалентные преобразования такого описания, можно построить различные структуры устройств управления для систем параллельной обработки, позволяющие увеличить их производительность и уменьшить аппаратные затраты.

Наиболее простое определение НДА основывается на языке графов. НДА называется произвольная диаграмма B над входным алфавитом, в которой выделены множества начальных и финальных вершин. Под диаграммой B понимается направленный граф, вершины которого помечены буквами из алфавита $B = [b_0, b_1, \dots, b_m]$ (или цифрами). Каждая вершина графа отмечена соответствующими выходными сигналами из абстрактного выходного алфавита $W = [w_0, w_1, \dots, w_y]$ или совокупностью выходных двоичных элементарных сигналов из структурного алфавита $Y = [y_1, y_2, \dots, y_N]$. Ребра графа помечены буквами из абстрактного входного алфавита $Z = [z_0, z_1, \dots, z_F]$ или частными входными сигналами, являющихся сочетаниями двоичных элементарных сигналов из структурного алфавита $X = [x_1, x_2, \dots, x_L]$.

Понятие «произвольный» здесь означает, что граф может не удовлетворять условиям автоматности [5]. В частности, допускаются следующие особенности:

- а) из некоторой вершины могут выходить несколько ребер, помеченных одной и той же буквой входного алфавита $[Z]$ или одним и тем же частным входным сигналом из алфавита $[X]$ (нарушение однозначности);
- б) из ряда вершин может вообще не выходить ни одно ребро, помеченное некоторой буквой входного алфавита (нарушение полной определенности);
- в) имеется несколько начальных вершин.

Автоматная интерпретация НДА предполагает следующее его функционирование. Если в некоторый момент времени (t) возбуждена вершина с номером $i(b_i)$ и имеет место входной сигнал z_k , то в момент времени ($t+1$) будут возбуждены все вершины, в которые ведут ребра с пометкой z_k . Если таких ребер нет, то не будет возбуждена ни одна вершина.

Так как в каждый момент времени (t) может быть возбуждено несколько вершин, то для определения вершин, которые будут возбуждены в момент времени ($t+1$), необходимо объединить множество вершин, возбуждаемых из каждой возбужденной вершины. В момент времени $t=0$ возбуждены все начальные вершины.

В каждом такте будут выдаваться выходные сигналы, которыми были отмечены одновременно возбужденные вершины.

Таким образом, недетерминированный автомат Мура это пятерка:

$$M=[S, X_{i,j}, \delta, S_0, Y_j],$$

где $[S] = [S_1, S_2, \dots, S_m]$ - множество частных событий;

$[X_{i,j}]$ - множество частных входных сигналов, образуемых сочетаниями элементарных двоичных входных сигналов из алфавита $[X]$;

$[Y_j]$ - множество выходных сигналов, отмечающих частные события S_j и образуемых совокупностью двоичных элементарных выходных сигналов из алфавита $[Y]$;

S_0 - начальное событие (возможно существование нескольких начальных событий);

δ - функция переходов НДА, которую в обобщенном виде можно представить так:

$$\delta[(S_{i,1}, \dots, S_{i,k}), X_{i,j}](t)=[(S_{j,1}, \dots, S_{j,l}), (Y_{j,1}, \dots, Y_{j,l})](t+1),$$

откуда видно, что функция переходов δ определяет переходы от i -го множества частных событий $(S_{i,1}, \dots, S_{i,k})$ к j -му множеству частных событий $(S_{j,1}, \dots, S_{j,l})$, одновременное существование которых в автомате возможно, под действием частного входного сигнала $X_{i,j}$. Необходимо иметь в виду, что в некоторых случаях i -е и/или j -е множества частных событий для отдельных переходов могут содержать в себе только один элемент.

Отметим, что **недетерминированность модели автомата Мура** будет проявляться лишь в возможном переходе под действием входного сигнала от какого-либо частного события S_i (или событий) **к нескольким частным событиям** (или событию S_j).

В то время как в **детерминированном автомате** (если под событием понимать переход автомата в определенное состояние a_j), всегда реализуются переходы под действием сигнала $X_{i,j}$ от события S_i только к **одному** событию S_j .

По НДА может быть построен эквивалентный ему конечный детерминированный автомат с помощью алгоритма детерминизации. Один из вариантов алгоритма детерминизации был предложен в работе [12], где он представлен в виде алгоритма синтеза таблицы переходов детерминированного цифрового автомата Мура, заданного моделью НДА в виде системы рекуррентных бескванторных предикатных формул, описывающих все реализуемые в автомате частные события. В дальнейшей работе [2] алгоритм детерминизации усовершенствовался применительно к синтезу микропрограммных управляющих автоматов. Алгоритмы детерминизации НДА были также рассмотрены, например, в работах [5, 6, 8].

В процессе детерминизации каждому возможному множеству одновременно возбужденных вершин (событий) ставится в соответствие

вполне определенное состояние автомата и определяются переходы между этими состояниями. НДА является более компактным представлением автомата, поскольку при детерминизации НДА с m реализуемыми в нем событиями (состояниями) эквивалентный ему детерминированный автомат может иметь не более $M \leq 2^m$ состояний, что будет доказано в дальнейшем. С другой стороны, можно утверждать, что детерминированный автомат с M состояниями может быть представлен НДА с $\lceil \log_2 M \rceil$ состояниями. Получение такого НДА возможно, например, путем кодирования детерминированного автомата состояниями НДА. Отсюда следует, что операция детерминизации над НДА и операция кодирования эквивалентного ему детерминированного автомата являются операциями обратными друг другу.

В том случае, когда в графе автомата выделено единственное начальное состояние (событие) и при любых входных последовательностях сигналов никогда не будут одновременно возбуждены две и более вершин (или одновременно существовать два или более события), такой автомат будет являться детерминированным. В этом случае каждой вершине графа или событию, реализуемому в автомате, будет соответствовать свое вполне определенное состояние автомата.

Над НДА возможно выполнение большого количества операций, из которых, кроме отмеченных выше операций детерминизации и кодирования, можно выделить: операцию стягивания начальных и финальных вершин, которые отмечены одним и тем же выходным сигналом; операции минимизации, отрицания, конкатенации, суммирования, объединения и др. Указанные операции над НДА рассматриваются как операции над СКУ, описывающими поведение НДА. Они используются для преобразования СКУ с целью построения структуры автомата, удовлетворяющей определенным требованиям. Это оказалось возможным благодаря использованию модели НДА в виде СКУ, представляющую собой простую стандартную и универсальную систему рекуррентных бескванторных предикатных формул, правая часть которых представляет собой ДНФ булевых функций.

1.2. Общие сведения о выборе языков для представления недетерминированных автоматов

НДА, также как и конечные цифровые автоматы, используются в качестве математической модели для формального описания алгоритмов логического управления в системах обработки дискретной информации. При этом наиболее эффективно они могут быть использованы в системах, реализующих параллельную обработку информации. Для формального описания таких математических моделей используются языки двух типов: начальные и стандартные, которые отличаются друг от друга способом задания функций переходов.

Начальные языки характеризуются тем, что функции переходов автоматов в описании их закона функционирования на этих языках представлены не в явном виде. К числу таких языков относятся, например, такие известные языки, как язык регулярных выражений алгебры событий (РВАС), язык логики одноместных предикатов, языки операторных схем алгоритмов (ОСА), языки операторных схем алгоритмов с параллельными ветвями (ОСАП) и др.

Стандартные языки, в отличие от начальных, характеризуются тем, что в описании закона функционирования автоматов на этих языках, функции переходов представлены в явном виде. К числу таких языков относятся: различные типы таблиц переходов и выходов, матрицы переходов, направленные графы и системы канонических уравнений.

Стандартный язык, представляя функции переходов автомата в явном виде, позволяет непосредственно перейти к структурной реализации управляющего устройства (автомата). С другой стороны, на стандартном языке в большинстве случаев затруднительно выполнять «перевод» неформального словесного описания алгоритма функционирования автомата на формальное описание, т.е. говорят, что стандартный язык обладает небольшими выразительными возможностями.

В связи с этим для первоначального описания алгоритма логического управления по его словесному неформальному описанию, представленному в техническом задании, целесообразно использовать начальные языки, к которым предъявляются следующие основные требования [8, 13, 14]:

- Язык должен обладать достаточно широкими выразительными возможностями и быть удобным не только для перевода на этот язык часто встречающихся словесных описаний алгоритма управления, но и должен быть достаточно удобным переход от начального формального описания к каноническим уравнениям стандартного языка. Кроме того, язык должен также содержать необходимые средства и методы для эквивалентных преобразований и выполнения контрольных процедур описания на начальном языке.

- Описание алгоритма управления, полученное на основе использования начального языка и представленное в виде системы уравнений, формализующих все реализуемые в алгоритме события, должно быть выражено в виде регулярных формул. Это требование следует из известной теоремы С.К.Клини [15], в которой утверждается, что **для того чтобы события были представлены в конечном автомате, необходимо и достаточно, чтобы они были регулярными**, т.е. описание таких событий должны быть получены из описания элементарных событий с использованием только трех операций алгебры событий: дизъюнкции, конкатенации и итерации. Таким образом, когда задание на реализацию алгоритма управления будет выражено на языке регулярных формул, то говорят, что проблема распознавания и синтеза управляющего устройства в соответствии с заданным алгоритмом не возникает [13].

Выбор того или иного начального языка для формального описания алгоритмов управления зависит в первую очередь от двух основных факторов: от сферы применения (области применения) языка и от традиций и привычек проектировщика в описании условий работы автомата [8, 14]. Например, для решения задач, связанных с распознаванием и преобразованием языков целесообразно использовать язык РВАС, а различные варианты языков операторных схем алгоритмов (ОСА) наиболее целесообразно использовать при решении вопросов проектирования систем логического управления объектами и процессами.

В нашей работе в качестве базового языка для формального представления управляющих алгоритмов (по аналогии с выбором базового языка в [14]) предлагается использовать язык граф-схем алгоритмов с параллельными ветвями (ГСАП) совместно с языком НД СКУ, являющимся его аналитической интерпретацией [16–21].

Такой выбор языков для формализации алгоритмов управления позволяет наглядно представить общую структуру модели алгоритма управления в виде графа НДА, в котором предусмотрены, кроме четырех типов вершин, имеющих в обычном языке ГСА, разветвительные и соединительные вершины, обеспечивающие разветвление алгоритмического процесса и соединение параллельных ветвей. При этом изменяя логические условия выхода алгоритмического процесса за соединительную вершину, можно решать многие вопросы синхронизации и взаимодействия параллельных ветвей.

Язык НД СКУ, являясь аналитической интерпретацией графического представления общей структуры алгоритма управления, позволяет детально представить аналитическое описание всех реализуемых в алгоритме частных событий в виде стандартной системы канонических бескванторных рекуррентных уравнений, в том числе частных событий, которые обеспечивают организацию взаимодействия параллельных ветвей алгоритма управления и их синхронизацию.

В связи с тем, что правая часть уравнений канонической системы НД СКУ представляет ДНФ булевых функций двух типов переменных, то над системой уравнений НД СКУ для целей ее структурной реализации (аппаратно, программно или аппаратно-программно) могут быть выполнены все необходимые эквивалентные преобразования, в том числе и преобразования в соответствии с законами булевой алгебры.

Кроме того, учитывая, что управляющий алгоритм аналитически представлен системой канонических уравнений НД СКУ, формализующих все частные события НДА, а не состояния детерминированного аппарата, то вопросы структурной реализации систем логического управления менее критичны к размерности решаемых задач, т.к. представление алгоритма управления в виде модели НДА является самым минимальным по сложности по сравнению с представлением алгоритма управления на основе использования модели детерминированного автомата.

Следует также отметить, что выбор указанных языков представления алгоритма управления предопределил также широкое использование для структурного синтеза систем управления прямые таблицы переходов для частных событий (НД ПТП), реализуемых в алгоритме управления (в виде списочной структуры). Использование НД ПТП позволяет выполнять при необходимости преобразование описания алгоритма управления с одного языка на другой, избавляться от недостижимых событий, выполнять проверку алгоритма управления на корректность и выполнять многие другие эквивалентные преобразования алгоритма управления, связанные с минимизацией, детерминизацией, кодированием и другими операциями над событиями.

В связи с тем, что язык НД СКУ является базовым, на который осуществляется перевод алгоритмов управления со всех начальных языков, то его рассмотрению посвящены два следующих раздела.

1.3. Аналитическое представление недетерминированных автоматов на стандартном языке

Если с каждой вершиной направленного графа НДА связать предикатную переменную $S_j(t)$, с каждым входным сигналом — предикатную переменную $X_{i,j}(t)$, а с каждым выходным сигналом — предикатную переменную $Y_j(t)$, где t пробегает ряд целых неотрицательных чисел ($t = 0, 1, 2, \dots$), то функционирование НДА может быть задано системой рекуррентных бескванторных предикатных формул вида (1.1), описывающих все реализуемые в автомате частные события S_j :

$$S_j^{Y_j}(t+1) = \bigvee_{i,j} X_{i,j}(t) \& S_i(t) \vee \bigvee_j X_{j,j}(t) \& S_j(t) \quad (1.1)$$

$$S_0(0) = I, \quad j = \overline{0, m},$$

где $(m+1)$ — число вершин графа (число всех событий, реализуемых НДА);

$S_i(t)$ — сокращенное обозначение выражения, описывающего событие S_i , непосредственно предшествующее событию S_j , при этом каждая предикатная формула $S_i(t)$ должна быть либо тождественно истинной предикатной формулой, либо удовлетворять условию $S_i(t) \neq S_j(t)$ и определяться также выражением вида (1.1);

$X_{i,j}$ — частный входной сигнал, представляющий собой сочетание элементарных входных сигналов из алфавита $[X]$, в результате воздействия которых осуществляется переход от вершины графа НДА с номером i к вершине графа с номером j (в новых обозначениях переход от события S_i к событию S_j);

Y_j — частный выходной сигнал, представляющий собой сочетание элементарных выходных сигналов из алфавита $[Y]$, отмечающих вершину графа b_j (или событие S_j).

Систему уравнений вида (1.1) будем называть в дальнейшем **простой канонической формой** или сокращенно — системой канонических уравнений (СКУ). При этом, если система уравнений описывает НДА, то сокращенное обозначение ее будет — НД СКУ. Кроме того, так как способы представления цифровых автоматов принято называть также и языками представления автоматов, то и в нашей работе, когда это целесообразно, будем называть систему уравнений вида (1.1) сокращенно — язык СКУ.

Как видно из (1.1) СКУ определяет все переходы в НДА и является отмеченной НД СКУ автомата Мура, так как каждое событие S_j отмечено своим частным выходным сигналом Y_j . При этом первая часть уравнений (1.1) позволяет формализовать описание условий первоначального появления или зарождения события S_j , а вторая - определить условие возобновления или сохранения его. Следует иметь ввиду, что описание некоторых событий S_j системы (1.1) могут и не содержать вторую часть уравнений, определяющую сохранение события S_j , в то время как присутствие первой части в каждом уравнении системы (1.1) обязательно.

Из системы уравнений (1.1) также следует, что введенное понятие «событие», принципиально ничем не отличается от понятия события, принятого в теории конечных цифровых автоматов [1]. Действительно, следуя [1], в нашем рассмотрении событие, например, S_j — это также множество слов, каждое из которых состоит из последней буквы ($X_{i,j}$) этого слова и начального отрезка слова до этой буквы, обозначенного сокращенно S_j , и определяемое как событие, непосредственно предшествующее событию S_j .

Представление функционирования НДА системой уравнений (1.1) соответствует описанию, которое является результатом преобразования регулярных выражений алгебры событий (РВАС) произвольной формы к виду, представленному в развернутой форме в виде системы канонических уравнений. Действительно, используя правило разворачивания итерации и закона коммутативности для итерации, в алгебре событий [1] выражение для события вида

$$S_\alpha = S_\beta \{R\}, \alpha \neq \beta \quad (1.2)$$

преобразуется в выражение с отсутствием итерационных скобок

$$S_\alpha = S_\beta \vee S_\alpha R \quad (1.3)$$

Откуда следует, что выражения (1.2) и (1.3) являются различными формами определения одного и того же события S_α . При этом выражение (1.2)

называют свернутой формой, а выражение (1.3) – развернутой формой определенного регулярного выражения алгебры событий.

В связи с этим каждую предикатную формулу $S_j(t+1)$ системы (1.1) можно привести к следующей свернутой форме регулярного выражения, имеющего вид:

$$\begin{aligned} S_j(t) &= \bigvee_{i,j} S_i(t-1) \& X_{i,j}(t) \vee \bigvee_j S_j(t-1) \& X_{j,j}(t) = \\ &= \bigvee_{i,j} S_i X_{i,j} \vee \bigvee_j S_j X_{j,j} = \bigvee_{i,j} S_i X_{i,j} \{X_{j,j}\} \end{aligned} \quad (1.4)$$

Выполняя все необходимые операции свертывания в уравнениях системы (1.1), которые содержат вторую часть описания, определяющую условия сохранения события, и выполняя m -раз операции подстановки свернутых выражений вместо непосредственно предшествующих событий S_i в выражения типа (1.4), получим представление исходной системы (1.1) в виде выражений алгебры событий, выраженные через частные входные сигналы и начальное событие S_0 с использованием только 3-х операций алгебры событий: дизъюнкции, конкатенации и итерации.

Таким образом, учитывая изложенное, можно утверждать, что система уравнений (1.1), представляющая простую каноническую форму, описывающая все реализуемые в автомате частные события, на основании теоремы С.К.Клини представима в конечном цифровом автомате, а следовательно и в НДА.

Примечание. В главе 4, посвященной начальным языкам, будет подробно рассмотрен алгоритм преобразования регулярных выражений алгебры событий в описание событий в виде системы канонических уравнений.

Для построения управляющего автомата в виде модели автомата Мура необходимо, кроме функций переходов, представленных уравнениями (1.1), получить и систему выходных функций (СВФ). Она может быть получена из (1.1) путем объединения тех событий S_j , в состав отмеченных частных выходных сигналов которых входит рассматриваемый элементарный выходной сигнал, т. е. СВФ для автомата Мура будет иметь вид

$$y_k(t+1) = \bigvee_{\forall S_j (Y_j \subset y_k)} S_j^{Y_j}(t+1), k = \overline{1, N} \quad (1.5)$$

или

$$y_k(t) = \bigvee_{\forall S_j (Y_j \subset y_k)} S_j^{Y_j}(t), k = \overline{1, N}, \quad (1.6)$$

где переменные в левой и правой частях уравнений взяты для одного и того же момента времени.

Система выходных функций для модели автомата Мили эквивалентного ему автомата Мура, может быть получена из отмеченной НД СКУ вида (1.1) путем объединения выражений для тех событий S_j , в состав отмеченных частных выходных сигналов которых входит рассматриваемый

элементарный выходной сигнал, т.е. для автомата Мили СВФ будет иметь вид

$$y_k(t) = \bigvee_{\forall S_j (y_j \subset y_k)} S_j^{Y_j}(t+1), k = \overline{1, N}, \quad (1.7)$$

где в правую часть выражения (1.7) необходимо сделать подстановку из (1.1).

Сравнивая выражения (1.5) и (1.7), подтверждаем тот факт, что выходной сигнал автомата Мура отстает на один такт по сравнению с выходным сигналом эквивалентного ему автомата Мили. Кроме того, следует иметь в виду, что выходные сигналы автомата Мура отличаются от выходных сигналов эквивалентного ему автомата Мили по времени действия, а именно: выходной сигнал автомата Мура действует от момента появления события и до его исчезновения, а выходной сигнал автомата Мили действует от момента появления входного сигнала и до его исчезновения.

Система функций переходов для автомата Мили, эквивалентного ему автомата Мура, может быть получена из системы уравнений (1.1) путем объединения тех событий НД СКУ автомата Мура, переходы из которых полностью совпадают.

Пример 1.1. Пусть НДА Мура задан графом (рис.1.1), в котором в качестве входного и выходного алфавитов используются структурные алфавиты:

$$X = [x_0, x_1, x_2, x_3, x_4], Y = [y_0, y_1, y_2, y_3, y_4],$$

где x_0 - сигнал приведения автомата в исходное состояние.

Построить для этого графа НД СКУ и СВФ для моделей автомата Мура и Мили.

Как видно из рисунка, представленный графом НДА Мура является также и не полностью определенным, так как из вершины 3 не удовлетворяются условия полноты переходов.

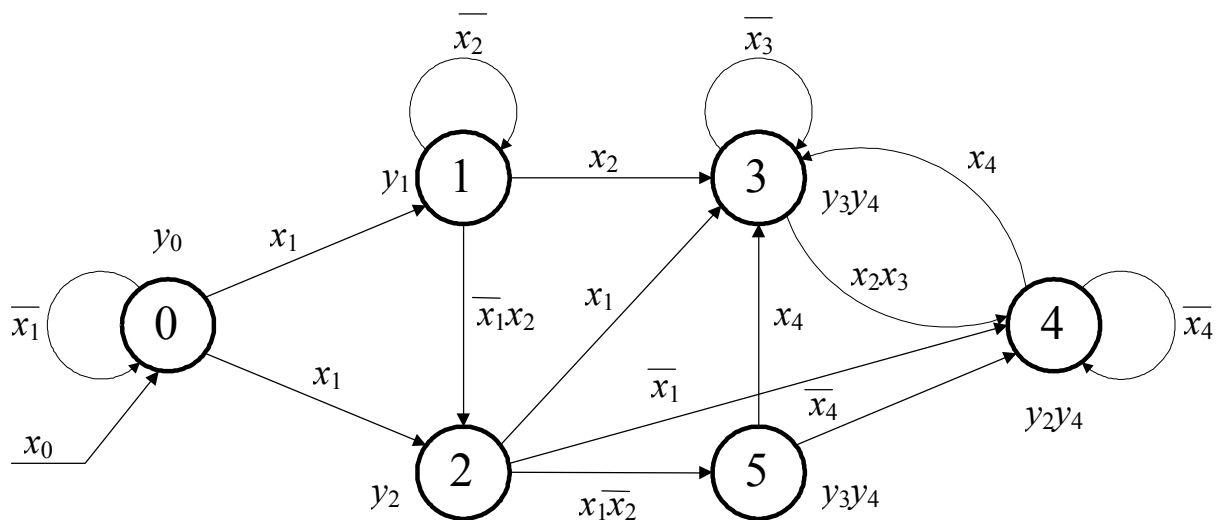


Рис.1.1. Граф недетерминированного автомата Мура

Поставив в соответствие каждой вершине графа событие перехода в эту вершину, получим следующую отмеченную недетерминированную СКУ не полностью определенного НДА Мура:

$$\begin{aligned}
 S_0^{y_0}(t+1) &= S_0(t) \& \bar{x}_1(t) \vee x_0(t); \\
 S_1^{y_1}(t+1) &= S_0(t) \& x_1(t) \vee S_1(t)x(t); \\
 S_2^{y_2}(t+1) &= S_0(t) \& x_1(t) \vee S_1(t) \& \bar{x}_1(t)x_2(t); \\
 S_3^{y_3y_4}(t+1) &= S_1(t) \& x_2(t) \vee S_2(t) \& x_1(t) \vee S_3(t) \& \bar{x}_3(t) \vee \\
 &\vee (S_4 \vee S_5)(t) \& x_4(t); \\
 S_4^{y_2y_4}(t+1) &= S_2(t) \& \bar{x}_1(t) \vee S_3(t) \& x_2(t) \& x_3(t) \vee \\
 &\vee (S_4 \vee S_5)(t) \& \bar{x}_4(t); \\
 S_5^{y_1y_3}(t+1) &= S_2(t) \& x_1(t) \& x(t).
 \end{aligned} \tag{1.8}$$

Примечание: В дальнейшем, если позволяет контекст, знаки $\&$ и время t в правой части СКУ для простоты будем опускать.

По НД СКУ автомата Мура в соответствии с (1.6) получим СВФ не полностью определенного НДА Мура (1.9):

$$\begin{aligned}
 y_0 &= S_0 & y_1 &= S_1 \vee S_5 & y_2 &= S_2 \vee S_4 \\
 y_3 &= S_3 \vee S_5 & y_4 &= S_3 \vee S_4
 \end{aligned} \tag{1.9}$$

НД СКУ автомата Мили, эквивалентного автомату Мура, получим из (1.8) путем объединения тех событий НД СКУ автомата Мура, переходы из которых полностью совпадают. К таким событиям относятся события S_4 и S_5 , переходы из которых совпадают: $(S_4 \vee S_5)x_4 \rightarrow S_3$, $(S_4 \vee S_5)\bar{x}_4 \rightarrow S_4$. При этом для всех букв левой части уравнений для НД СКУ автомата Мили опускаются отмечающие их выходные сигналы. Делая замену переменных $S_4 \vee S_5 = S_4$ и их подстановку в (1.8), получим НД СКУ не полностью определенного автомата Мили:

$$\begin{aligned}
 S_0(t+1) &= S_0\bar{x}_1 \vee x_0; \\
 S_1(t+1) &= S_0x_1 \vee S_1\bar{x}_2; \\
 S_2(t+1) &= S_0x_1 \vee S_1\bar{x}_1x_2; \\
 S_3(t+1) &= S_1x_2 \vee S_2x_1 \vee S_3\bar{x}_3 \vee S_4x_4; \\
 S_4(t+1) &= S_2(\bar{x}_1 \vee \bar{x}_2) \vee S_3x_2x_3 \vee S_4\bar{x}_4.
 \end{aligned} \tag{1.10}$$

По НД СКУ автомата Мура в соответствии с (1.7) и учитывая замену $S_4 \vee S_5 = S_4$, получим СВФ не полностью определенного НДА Мили:

$$\begin{aligned}
y_0 &= S_0 \bar{x}_1 \vee x_n; \\
y_1 &= S_0 x_1 \vee S_1 \bar{x}_2 \vee S_2 x_1 \bar{x}_2; \\
y_2 &= S_0 x_1 \vee S_1 \bar{x}_1 x_2 \vee S_2 \bar{x}_1 \vee S_3 x_2 x_3 \vee S_4 \bar{x}_4; \\
y_3 &= S_1 x_2 \vee S_2 x_1 \vee S_3 \bar{x}_3 \vee S_4 x_4; \\
y_4 &= S_1 x_2 \vee S_2 \vee S_4 \vee S_3 (x_2 \vee \bar{x}_3).
\end{aligned} \tag{1.11}$$

1.4. Расширение выразительных возможностей языка НД СКУ и его отличительные особенности

Следует отметить, что в ряде случаев исходный алгоритм функционирования автомата целесообразно сразу записать в форме НД СКУ, т. е. использовать СКУ как начальный язык. В этом случае НД СКУ, представленную в простой канонической форме уравнениями вида (1.1), целесообразно представить в более обобщенной форме с целью увеличения «выразительных» возможностей языка СКУ. В частности, такую НД СКУ можно представить в виде системы рекуррентных предикатных формул в следующем виде:

$$\begin{aligned}
S_j^Y(t+1) &= \bigvee_{i,j} X_{i,j}(t) \& R_i(t) \& S_i(t) \vee \bigvee_j X_{j,j}(t) \& R_j(t) \& S_j(t), \\
S_0(0) &= 1, j = \overline{0, m},
\end{aligned} \tag{1.12}$$

где $R_i = \tilde{S}_{i,1} \& \dots \& \tilde{S}_{i,k}$, $R_j = \tilde{S}_{j,1} \& \dots \& \tilde{S}_{j,k}$; знак « $\sim$ » означает, что переменные в R_i и R_j могут быть взяты с отрицанием или без него.

Из структуры уравнений системы (1.12) следует, что расширение выразительных возможностей для формального описания событий достигается за счет описания более сложных условий зарождения и сохранения событий S_j , которые могут зависеть не только от значений внешних частных входных сигналов $X_{i,j}$, но и от значений совокупности некоторых частных событий или внутренних переменных, непосредственно предшествующих событию S_j . К числу таких частных событий или внутренних переменных, образующих комбинационные события R_i и R_j , могут относиться, например, такие, которые определяют условия готовности исходных данных для их обработки в операционных блоках и их готовности к приему результатов обработки исходных данных, а также переменные, отражающие временной фактор, определяющий начало или конец выполнения отдельных операций (микроопераций) и др.

При формализации частных событий системы (1.12) необходимо иметь в виду, что описание каждого частного события S_j должно в общем случае состоять из трех частей [21]:

- описание события в момент его наступления;
- описание условий, при которых событие сохраняется;
- описание события, непосредственно предшествующего событию

S_j .

Отметим также, что описание некоторых событий могут и не иметь описаний условий его сохранения, тогда как две другие части описания события S_j должны быть обязательно. В противном случае описание события S_j может преобразоваться в булеву функцию, реализуемую простой комбинационной схемой.

Полученная система уравнений (1.12) должна быть проверена на отсутствие недостижимых событий. Для этой цели строится прямая таблица переходов недетерминированного автомата Мура (НД ПТП) для всех частных событий, представленных уравнениями (1.12) и реализуемых в автомате.

НД ПТП Мура в общем случае может иметь пять основных колонок (рис.1.2).

	$S_i(t)$	$R_i(t)$	$X_{i,j}(t)$	$S_j(t+1)$	$Y_j(t+1)$
1	2	3	4	5	6
1	S_0				

Рис.1.2. Прямая таблица переходов для событий
НДА Мура (НД ПТП)

Построение НД ПТП должно выполняться по следующему правилу: построение первой строки таблицы начинается с исходного события S_i , которое соответствует событию S_0 , а для всех последующих строк в качестве исходного события S_i используется то, которым уже была отмечена колонка (5) в предыдущих строках таблицы. Такой порядок построения НД ПТП выявляет недостижимые события, то есть те, которые не имеют событий предшественников.

Построенная НД ПТП может быть в дальнейшем использована для преобразования системы уравнений (1.12), связанных с выполнением операций минимизации, детерминизации, кодирования и др.

Отличительными особенностями языка НД СКУ являются:

а) Язык НД СКУ является языком, определяющим в явном виде функции переходов в автомате. Эти функции выражаются в виде переходов от события к событию (или событиям) или переходов от событий к событию

(или событиям). Причем эти переходы для каждого входного сигнала являются *однозначными*.

б) Недетерминированная СКУ представляется системой «рекуррентных» формул. Это означает, что каждое событие S_j из множества реализуемых в автомате событий, может быть определено через событие S_i (события), имеющее место в предшествующий момент автоматного времени. При этом в начале работы автомата может быть истинным то событие (события), для которого в предшествующий момент времени было истинным событие S_0 .

в) В названии системы НД СКУ присутствует слово «каноническая». Этим названием подчеркивается тот факт, что все переменные в правой части уравнений НД СКУ взяты для одного и того же момента времени (t).

г) Учитывая, что правая часть формул НД СКУ представляет собой ДНФ булевых функций от двух типов переменных, представляющих как входные сигналы, так и события, то к этим формулам могут быть применены все законы и методы преобразования булевых функций.

д) Учитывая также, что для систем (1.1) и (1.12) можно было бы вместо знака равенства поставить знак $\asymp$, являющийся знаком сокращенного обозначения, означающим, что предикат, стоящий слева от этого знака, является сокращенным обозначением предикатной формулы, расположенной от него справа [15, стр. 222, 226], то возможно применять к формулам НД СКУ все законы и методы преобразования, принятые в исчислении высказывания и предикатов.

е) Исходя из изложенного, целесообразно математическую модель НДА, представленную на языке НД СКУ, называть в дальнейшем *автоматной моделью НДА*, хотя во многих работах по теории цифровых автоматов слово «автоматность» всегда отождествлялось с понятием детерминированного автомата. При введении такого названия автор исходил из того, что в названии математической модели автомата словом «автоматная» в первую очередь подчеркивается *задание функций переходов в автомате в явном виде*.

ж) В связи с тем, что НДА является обобщением конечного детерминированного автомата, то математическую модель детерминированного автомата (ДА), представленную на языке СКУ, будем называть *автоматной моделью ДА*. Для такого автомата СКУ будет представлять собой функции переходов в автомате, когда переход в любое состояние интерпретируется как событие.

В заключение отметим, что в ряде работ автора приведены алгоритмы синтеза автоматных моделей управляющих автоматов, заданных на некоторых начальных языках: операторных схем алгоритмов (ОСА), в том числе с параллельными ветвями (ОСАП) [2, 16—19], на языке регулярных выражений алгебры событий [2, 20], на языке исчисления предикатов 1-го порядка [9].

1.5. Иерархия входных сигналов и событий, реализуемых в системах логического управления

В данном разделе уточняются и систематизируются введенные ранее некоторые понятия и определения и вводятся ряд новых понятий и определений, связанных с представлениями входных сигналов и событий, реализуемых в системах управления. Эти понятия и определения будут в дальнейшем использоваться в последующих разделах пособия при рассмотрении вопросов, связанных с эквивалентными преобразованиями НДА, формализацией алгоритмов управления и структурного синтеза систем управления.

Частный входной сигнал — сигнал, определяемый сочетанием (конъюнкцией) тех элементарных двоичных входных сигналов из алфавита $[X]$, взятых с отрицанием или без них, которые действуют *в одной из любых ветвей алгоритма управления на данном его шаге* и которые определяют вполне определенное (частное) событие. Будем обозначать эти сигналы буквой $X_{i,j}$, означающей, что под действием этого сигнала происходит переход от события S_i к событию S_j в пределах одного шага алгоритма управления. Под шагом алгоритма понимается путь от одной операторной вершины к другой, проходящей только через логические условия.

Полный входной сигнал в алфавите $[X]$ - сигнал, определяемый сочетанием (конъюнкцией) тех элементарных двоичных входных сигналов, взятых с отрицанием или без них, действующих *во всех ветвях на данном шаге алгоритма управления*, которые определяют вполне определенное состояние автомата (полное событие). Будем обозначать этот сигнал символом $X(a_m, a_s)$, означающим, что под действием этого сигнала происходит переход ДА из состояния a_m в состояние a_s в пределах одного шага алгоритма управления.

Полный входной сигнал в алфавите $[Z]$. Этому сигналу будет соответствовать полный входной сигнал в алфавите $[X]$, определяемый сочетанием всех L элементарных двоичных входных сигналов из алфавита $[X]$, взятых с отрицанием или без них, действующих *во всех ветвях и на всех шагах алгоритма управления*. Максимально возможное число таких сигналов равно 2^L . Будем обозначать такие сигналы символом $Z(a_m, a_s)$, означающий, что под действием этого сигнала происходит переход ДА из состояния a_m в состояние a_s в пределах одного шага алгоритма управления.

Пустой входной сигнал — специально введенный абстрактный входной сигнал, называемый пустой буквой e (пустое слово нулевой длины)» позволяющий считать, что абстрактный автомат (математическая модель автомата) получает входные сигналы во все множества времени $t = 0, 1, 2, \dots$, несмотря на то, что реальный автомат в какие-то моменты времени может и не получать никаких реальных входных сигналов [1]. Таким образом,

появление буквы e на входе автомата свидетельствует о том, что на вход автомата в действительности ничего не подается. Это нашло отражение в законе нейтральности пустого слова: $eS = Se = S$.

Запрещенный входной сигнал — сигнал, который на вход автомата никогда не подается. Выражение для такого сигнала может быть определено как отрицание от дизъюнкции всех входных сигналов $X_{i,j}$, действующих на переходе. Для запрещенного входного сигнала событие перехода будет неопределенным.

Выражение для запрещенных входных сигналов может быть определено как отрицание от дизъюнкции всех входных сигналов, действующих на переходах алгоритма управления.

В зависимости от типа математической модели автомата можно выделить три вида запрещенных входных сигналов:

- **Частный запрещенный входной сигнал** в одной из ветвей алгоритма управления на определенном его шаге в алфавите $[X]$. Например, пусть задан фрагмент ПТП автомата, для которого не выполняются условия полноты переходов.

$S_i(t)$	$x_2(t)$	$S_j^1(t+1)$
	$x_1 \bar{x}_2(t)$	$S_j^2(t+1)$

Для этого фрагмента ПТП $x_{i,j}^1 \vee x_{i,j}^2 \neq 1$. Полнота переходов может быть обеспечена вводом запрещенного входного сигнала, для которого событие перехода будет неопределенным.

$$x_{\text{запр.}} = \overline{x_2 \vee x_1 \bar{x}_2} = \bar{x}_1 \bar{x}_2$$

Скорректированный фрагмент ПТП будет иметь вид:

$S_i(t)$	$x_2(t)$	$S_j^1(t+1)$
	$x_1 \bar{x}_2(t)$	$S_j^2(t+1)$
	$(\bar{x}_1 \bar{x}_2)^*(t)$	$S(-)$

Здесь знаком $*$ будем обозначать запрещенный входной сигнал, а знаком $S(-)$ — неопределенное событие.

- **Полный запрещенный входной сигнал** для всех ветвей алгоритма управления на определенном его шаге в алфавите $[X]$.

- **Полный абстрактный запрещенный входной сигнал** для всех ветвей алгоритма управления на всех его шагах в алфавите $[Z]$.

Покажем на примере взаимосвязь между отдельными видами запрещенных входных сигналов и пустым входным сигналом.

Пусть алфавит абстрактных входных сигналов содержит следующие пять сигналов $[Z] = [z_0, z_1, z_2, z_3, z_4]$. Тогда при построении структурной схемы автомата абстрактные входные сигналы должны кодироваться тремя двоичными входными сигналами из алфавита $[X] = [x_1, x_2, x_3]$.

Для произвольного кодирования получаем.

$$\begin{aligned} z_0 &= \bar{x}_1 \bar{x}_2 \bar{x}_3, & z_3 &= \bar{x}_1 x_2 x_3, \\ z_1 &= \bar{x}_1 \bar{x}_2 x_3, & z_4 &= x_1 \bar{x}_2 \bar{x}_3, \\ z_2 &= \bar{x}_1 x_2 \bar{x}_3, \end{aligned}$$

Неиспользуемые кодовые группы имеют вид:

$$z_5 = x_1 \bar{x}_2 \bar{x}_3, \quad z_6 = x_1 x_2 \bar{x}_3, \quad z_7 = x_1 x_2 x_3.$$

Эти неиспользуемые кодовые группы и составят полные запрещенные входные сигналы. Они определяются также как отрицание от дизъюнкции действующих входных сигналов:

$$e_3 = \overline{z_0 \vee z_1 \vee z_2 \vee z_3 \vee z_4} = x_1 \bar{x}_2 x_3 \vee x_1 x_2 \bar{x}_3 \vee x_1 x_2 x_3.$$

Пустой абстрактный входной сигнал для данного алфавита входных сигналов определится как отрицание от дизъюнкции всех абстрактных входных сигналов, включая запрещенные:

$$e = \overline{z_0 \vee z_1 \vee z_2 \vee z_3 \vee z_4 \vee e_3} = \bar{1} = 0$$

S-событие — это частное событие, определяемое частными входными сигналами, отмеченное соответствующей совокупностью выходных сигналов из алфавита $[Y]$ и формализуемое в соответствии с формулами (1.1) и (1.12). Множество *S*-событий определяет функции переходов в НДА Мура.

a-событие — это полное событие, определяемое полными входными сигналами в алфавите $[X]$, которому соответствует возможное сочетание частных событий, одновременное существование которых в автомате возможно. Этому сочетанию частных событий ставится в соответствие одно вполне определенное состояние ДА Мура, отмеченное совокупностью выходных сигналов, отмечающих соответствующие частные события. Множество *a*-событий определяют функции переходов ДА Мура.

Q-событие — это событие, определяющее переходы элементарного автомата (триггера), входящего в состав памяти автомата. Модель автомата, представленная системой уравнений для *Q*-событий (СКУ для *Q*-событий), будет недетерминированной относительно *Q*-событий, хотя система, полученная в результате кодирования полных *a*-событий, представляет реальный ДА. Отметим, что детерминизация СКУ для *Q*-событий даст нам СКУ для *a*-событий.

Промежуточное событие — это событие, обычно не отмеченное реальным выходным сигналом, которое играет вспомогательную роль при определении отмеченных и реализуемых в автомате событий.

Пустое (или тупиковое) событие S_* — это событие, из которого не может быть получено ни одно из реализуемых в автомате событий, кроме начального события S_0 , являющегося исходным для всех событий в автомате. Пустое событие может быть определено выражением, являющимся отрицанием от дизъюнкции всех реализуемых в автомате событий. Пустому событию может быть поставлено в соответствие одно состояние автомата, которое по аналогии будем называть пустым (или тупиковым) состоянием.

Неопределенное событие — это событие, которому соответствует на переходе запрещенный входной сигнал, в результате чего функция переходов на этом шаге алгоритма будет не определена и может быть доопределена совершение произвольно. Это позволит выполнять упрощение формул, определяющих функции переходов автоматов. Необходимо иметь в виду, что $S_i \& S(-) \neq 0$, тогда как $S_i \& S_* = 0$.

Недостижимое событие — это событие, которое не может быть получено из начального события при любой последовательности входных сигналов, т. е. такое событие не имеет событий - предшественников.

Контрольные вопросы к главе 1.

1. В чем состоит основная особенность модели НДА, отличающая ее от модели ДА ?
2. Как определяется частный входной сигнал и чем он отличается от полного входного сигнала ?
3. Каким образом можно определить запрещенный входной сигнал и чем он отличается от пустого входного сигнала ?
4. Из каких составных частей состоит общий вид описания любого из событий, представляющих управляющий алгоритм в виде модели НДА в стандартной канонической форме ? Каким образом можно управлять началом наступления события и продолжительностью его существования ?
5. Как могут быть представлены функции переходов в модели НДА ?
6. Как построить НДА Мили, эквивалентный заданному НДА Мура ?
7. В чем отличие пустого события от неопределенного события ?
8. Как связаны между собой операция кодирования и операция детерминизации над событиями, определяющими алгоритм управления на разных этапах проектирования систем управления, и как такая связь может быть использована для организации контрольных процедур при проектировании систем управления ?

Глава 2. Детерминизация недетерминированных автоматов

В предыдущей главе было отмечено, что над НДА возможно выполнение большого количества операций, основными из которых являются операции детерминизации, минимизации и кодирования. Эти операции над НДА рассматриваются как операции над системами канонических уравнений (СКУ), описывающими поведение НДА. Перечисленные основные операции над СКУ и некоторые дополнительные используются для эквивалентных преобразований НДА с целью построения структур управляющих автоматов, удовлетворяющих заданным определенным требованиям.

Будем рассматривать алгоритм детерминизации НДА Мура, имея в виду, что для НДА Мили алгоритм детерминизации в принципе будет такой же.

Максимально возможное число состояний детерминированного автомата, полученного в результате детерминизации эквивалентного ему НДА, представленного m частными событиями, не может быть больше, чем 2^m и определится из выражения:

$$M \leq \sum_{k=0}^m C_m^k = 2^m \quad (2.1)$$

где C_m^k — число сочетаний из m элементов по k .

Величина M может быть определена так же, как определяется в булевой алгебре максимальное число наборов значений двоичных переменных, на которых задается булева функция m переменных. В нашем случае под булевыми переменными будем понимать сокращенные обозначения частных событий S_j , число которых равно m . Если каждому набору значений переменных S_j будет соответствовать состояние детерминированного автомата, то $M \leq 2^m$.

Основу алгоритма детерминизации НД СКУ составляет процесс отыскания на каждом шаге работы алгоритма функционирования автомата множества сочетаний частных событий, одновременное существование которых в автомате возможно. Такому сочетанию частных событий, названному в главе 1 полным событием (a -событием), ставится в соответствие вполне определенное состояние детерминированного автомата, отмеченное совокупностью выходных сигналов, отмечающих соответствующие частные события.

Для более эффективного решения задачи детерминизации НДА более наглядного представления этого процесса исходную НД СКУ целесообразно представить в виде прямой таблицы переходов НДА, на основании которой будем строить таблицу переходов детерминированного автомата по

алгоритму, который, как было отмечено ранее, базируется на алгоритме синтеза цифровых автоматов Мура, предложенном в [12].

2.1. Алгоритм детерминизации НДА

Алгоритм построения прямой таблицы переходов детерминированного автомата Мура включает следующие этапы:

1) Строится прямая таблица переходов НДА Мура и заготавливается прямая таблица переходов детерминированного автомата Мура. По прямой таблице переходов исходного НДА (ПТП НДА) отыскивается совокупность (сочетание) исходных — начальных событий и/или частных событий, выводимых из них. Этой совокупности (сочетанию) начальных событий, ставится в соответствие начальное состояние детерминированного автомата, а их обозначения записываются во второй столбец в первой строке прямой таблицы переходов детерминированного автомата Мура (ПТП ДА Мура).

2) Для каждого исходного (вначале начального) частного события, входящего в совокупность (сочетание) частных событий, записанных во втором столбце ПТП ДА Мура, выписываются в третий столбец подмножества частных входных сигналов, которые вызывают переходы из этих исходных событий.

В каждом из подмножеств частных входных сигналов находят все не эквивалентные между собой и не эквивалентные нулю сочетания.

3) Выполняют анализ и корректировку состава каждого из полученных подмножеств сочетаний частных входных сигналов, вызывающих переходы из исходного события, на основании которого:

а) оставляют для дальнейшего рассмотрения сочетания, не являющиеся выводимыми ни из какого одного из сочетаний рассматриваемого подмножества и те из выводимых, для которых их истинность не влечет истинность хотя бы одного из сочетаний, из которого это сочетание само выводимо;

б) исключают из дальнейшего рассмотрения те из выводимых сочетаний частных входных сигналов рассматриваемого подмножества, истинность которых всегда влечет истинность хотя бы одного из сочетаний, из которых они выводимы;

в) выполняют проверку состава частных входных сигналов рассматриваемого подмножества на полноту состава. Нарушение полноты состава частных входных сигналов будет иметь место в том случае, если логическая сумма всех частных входных сигналов данного подмножества не равна единице. Такой состав частных входных сигналов не удовлетворяет условиям полноты переходов в автомате для рассматриваемого исходного события.

Для устранения неполноты состава частных входных сигналов определяются дополнительные частные входные сигналы, которые для рассматриваемого подмножества будут запрещенными. Для них

соответствующие события перехода будут неопределенными. Запрещенные входные сигналы определятся из выражения, представляющего собой отрицание от дизъюнкции всех частных входных сигналов рассматриваемого подмножества.

4) Для всех откорректированных подмножеств частных входных сигналов, действующих на данном шаге алгоритма работы ДА Мура, находится множество всех не эквивалентные между собой и не эквивалентные нулю сочетания частных входных сигналов.

5) Выполняется анализ и корректировка полученного множества сочетаний частных входных сигналов, действующих на данном переходе ДА Мура, на основании которого:

а) оставляют для дальнейшего рассмотрения и записываются в третий столбец ПТП ДА сочетания, не являющиеся выводимыми ни из какого одного из сочетаний рассматриваемого множества и те из выводимых, для которых их истинность не влечет истинность хотя бы одного из сочетаний, из которого это сочетание само выводимо;

б) исключают из дальнейшего рассмотрения те из выводимых сочетаний, истинность которых влечет истинность хотя бы одного из сочетаний, из которых они выводимы;

в) к оставленным выводимым сочетаниям применяют операцию развертывания, умножая их на выражение вида $(x_i \vee \bar{x}_i) \cdots (x_j \vee \bar{x}_j)$, где $x_i, \dots, x_j$ — элементарные входные сигналы, которые не вошли в выводимое сочетание элементарных входных сигналов по сравнению с сочетанием, из которого они выводятся. Такая операция необходима для обеспечения условий однозначности на рассматриваемом переходе ДА (шаге алгоритма). Из полученных в результате выполнения операции развертывания частных входных сигналов удаляются те из них, которые эквивалентны ранее полученным. В том случае, если выводимое сочетание частных входных сигналов выводимо из нескольких сочетаний частных входных сигналов, то при операции развертывания необходимо использовать те элементарные входные сигналы $x_i, \dots, x_j$, которые входят в сочетание с наибольшим числом букв.

В результате выполнения п.5 алгоритма детерминизации будут получены все полные входные сигналы $X(a_m, a_s)$, действующие на рассматриваемом шаге работы алгоритма ДА.

Примечание: в том случае, если детерминизация НДА используется только с целью определения состава всех полных событий, то операцию развертывания к выводимым сочетаниям частных входных сигналов можно не применять, так как новых полных событий не получим. Для таких входных сигналов в столбце 4 будем ставить дополнительно знак ().

б) Для каждого полного входного сигнала $X(a_m, a_s)$, записанного в третьем столбце, определяют, используя исходную ПТП НДА, совокупность (сочетание) частных событий, которые выводимы из каждого исходного

(вначале начального) события. Этой совокупности (сочетанию) частных событий ставится в соответствие определенное состояние ДА (полное событие a_s), а их обозначения записываются в четвертом столбце таблицы переходов ДА.

В том случае, если на данном шаге работы автомата в какой-либо строке ПТП ДА имеет место запрещенный входной сигнал, то в четвертом столбце в рассматриваемой строке ставится черточка (или событие $S_{(-)}$), означающая, что событие или совокупность событий (полное событие) на этом переходе является неопределенным.

Если для какого-нибудь сочетания частных входных сигналов (полного входного сигнала) и для исходных частных событий, входящих в рассматриваемую совокупность, невыводимо ни одно из частных событий НДА, то четвертый столбец в рассматриваемой строке отмечается обозначением пустого события S_* .

7) Каждое из полученных новых сочетаний частных событий (полное событие) записывается во второй столбец очередной строки таблицы переходов автомата и вновь повторяются этапы по п.п. 2, 3, 4, 5, 6, 7 до тех пор, пока не будут исчерпаны все новые сочетания частных событий, записываемые в четвертый столбец.

8) Отмечаются все совокупности (сочетания) частных событий (полные события) соответствующими обозначениями частных выходных сигналов.

Сочетания частных промежуточных событий (состояния автомата), которые не отмечены никакими выходными сигналами, отмечаются знаком (). Так как знак () не противоречит никакому частному выходному сигналу, то при необходимости можно дополнительно отметить эти состояния ДА любым (в том числе и пустым) выходным сигналом.

9) После окончания построения ПТП ДА ее еще раз проверяют на однозначность и полноту переходов. Для однозначности переходов должно выполняться условие равенства нулю всех попарных произведений входных сигналов на каждом шаге работы алгоритма ДА, а для полноты переходов логическая сумма всех полных входных сигналов на каждом шаге работы алгоритма ДА должна равняться единице.

По полученной прямой таблице переходов детерминированного автомата Мура нетрудно построить детерминированную СКУ по следующему алгоритму:

1) В прямой таблице переходов, полученной после детерминизации, отыскиваются все пути, которые оканчиваются одинаковыми полными событиями (состояниями) $a_s(t+1)$, и составляются для них конъюнкции из обозначений полных событий (состояний) $a_m(t)$, стоящих в начале пути перехода, с соответствующими сочетаниями частных входных $X(a_m, a_s)$ сигналов, действующих на переходе от a_m к a_s .

2) Полученные конъюнкции $a_m(t) \& X(a_m, a_s)(t)$ объединяют знаком дизъюнкции. Такую операцию выполняют для всех полных событий (состояний).

3) Все полные события (состояния), представленные в левой части уравнений СКУ, отмечаются соответствующими сочетаниями выходных сигналов.

Пример 2.1. Сформировать полные входные сигналы для одного из шагов алгоритма работы ДА.

Пусть на данном шаге алгоритма работы автомата заданы следующие два подмножества частных входных сигналов.

$$[X]_1 = [x_1 \bar{x}_3, \bar{x}_1 x_2, x_3, \bar{x}_3],$$

$$[X]_2 = [x_1, \bar{x}_1].$$

В соответствии с п.2 алгоритма детерминизации получим следующие сочетания частных входных сигналов в подмножестве $[X]_1$:

$$[x_1 \bar{x}_3, \bar{x}_1 x_2 x_3, \bar{x}_1 x_2 \bar{x}_3, \bar{x}_1 x_2, x_3, \bar{x}_3]$$

Из анализа полученных сочетаний частных входных сигналов следует, что в данном сочетании имеет место:

- сочетания, которые не являются выводимыми ни из какого одного из сочетаний подмножества $[X]_1$. Для нашего примера к таким сочетаниям относятся: $x_1 \bar{x}_3$, $\bar{x}_1 x_2 x_3$, $\bar{x}_1 x_2 \bar{x}_3$; такие сочетания остаются для дальнейшего рассмотрения;
- сочетание $\bar{x}_1 x_2$, которое выводимо из сочетаний $\bar{x}_1 x_2 x_3$ и $\bar{x}_1 x_2 \bar{x}_3$. Так как истинность сочетания $\bar{x}_1 x_2$ всегда влечет истинность одного из двух сочетаний, из которых выводим частный входной сигнал $\bar{x}_1 x_2$, то данный входной сигнал исключается из дальнейшего рассмотрения;
- элементарные сочетания – входной сигнал x_3 , выводимый из сочетания $\bar{x}_1 x_2 x_3$ и входной сигнал $\bar{x}_3$, выводимый из сочетаний $x_1 \bar{x}_3$ и $\bar{x}_1 x_2 \bar{x}_3$. Истинность сигнала x_3 еще не означает истинность сочетания $\bar{x}_1 x_2 x_3$, а истинность сигнала $\bar{x}_3$ еще не означает всегда истинность хотя бы одного из сочетаний $x_1 \bar{x}_3$ и $\bar{x}_1 x_2 \bar{x}_3$. Поэтому сигналы x_3 и $\bar{x}_3$ остаются для дальнейшего рассмотрения.

Таким образом, откорректированное подмножество частных входных сигналов $[X]_1$ будет иметь вид:

$$[X]_1 = [x_1 \bar{x}_3, \bar{x}_1 x_2 x_3, \bar{x}_1 x_2 \bar{x}_3, x_3, \bar{x}_3]$$

Сочетания всех частных входных сигналов для двух заданных подмножеств $[X]_1$ и $[X]_2$ после корректировки подмножества $[X]_1$ будут представлены следующим множеством:

$$[X]_{1,2} = [x_1 \bar{x}_3, x_1 x_3, \bar{x}_1 x_2 x_3, \bar{x}_1 x_2 \bar{x}_3, \bar{x}_1 x_3, \bar{x}_1 \bar{x}_3, x_3, \bar{x}_3, x_1, \bar{x}_1]$$

В соответствии с п.5а и 5б алгоритма детерминизации из полученного множества сочетаний частных входных сигналов остаются для дальнейшего рассмотрения следующие шесть сочетаний:

$$[x_1 \bar{x}_3, x_1 x_3, \bar{x}_1 x_2 x_3, \bar{x}_1 x_2 \bar{x}_3, \bar{x}_1 x_3, \bar{x}_1 \bar{x}_3]$$

К двум последним сочетаниям частных входных сигналов в соответствии с п.5в алгоритма детерминизации необходимо применить операцию развертывания, так как каждое из них является выводимым из другого сочетания рассматриваемого множества и их истинность не влечет истинность сочетаний, из которых они выводимы. Для нашего примера сочетание $\bar{x}_1 x_3$ выводимо из сочетания $\bar{x}_1 x_2 x_3$, а сочетание $\bar{x}_1 \bar{x}_3$ выводимо из сочетания $\bar{x}_1 x_2 \bar{x}_3$. Операции развертывания будут иметь вид:

$$x_1 \bar{x}_3 (x_2 \vee \bar{x}_2) \text{ и } \bar{x}_1 \bar{x}_3 (x_2 \vee \bar{x}_2)$$

В результате имеем после учета эквивалентных, ранее полученных сигналов, следующие полные входные сигналы, которые действуют на данном шаге алгоритма работы ДА:

$$[x_1 \bar{x}_3, x_1 x_3, \bar{x}_1 x_2 x_3, \bar{x}_1 x_2 \bar{x}_3, \bar{x}_1 (x_2) x_3, \bar{x}_1 (x_2) \bar{x}_3]$$

где переменная в скобках свидетельствует о применении операции развертывания.

Пример 2.2. Сформировать полные входные сигналы для одного из шагов алгоритма работы ДА, когда на данном шаге алгоритма работы автомата заданы два подмножества частных входных сигналов, одно из которых не удовлетворяет условиям полноты.

$$[X]_1 = [\bar{x}_2, x_2, \bar{x}_1 x_2]$$

$$[X]_2 = [\bar{x}_3, x_2 x_3]$$

Операции определения сочетаний частных входных сигналов в заданных подмножествах новых частных входных сигналов не дает, поэтому в соответствии с п.2в алгоритма детерминизации выполняем проверку заданных подмножеств на полноту состава входных сигналов. Из двух подмножеств второе подмножество $[X]_2$ не удовлетворяет условиям полноты, так как дизъюнкция входных сигналов не равна единице:

$$\bar{x}_3 \vee x_2 x_3 = \bar{x}_3 \vee x_2.$$

Запрещенный входной сигнал для подмножества $[X]_2$, определяемый как отрицание от дизъюнкции входных сигналов, имеет вид:

$$\overline{\bar{x}_3 \vee x_2} = \bar{x}_2 x_3$$

Таким образом, скорректированные подмножества частных входных сигналов примут следующий вид:

$$[X]_1 = [\bar{x}_2, x_2, \bar{x}_1 x_2]$$

$$[X]_2 = [\bar{x}_3, x_2 x_3, (\bar{x}_2 x_3)^*]$$

Сочетания всех частных входных сигналов для откорректированных подмножеств $[X]_1$ и $[X]_2$ будут представлены следующим множеством:

$$[X]_{1,2} = [\bar{x}_2 \bar{x}_3, x_2 \bar{x}_3, \bar{x}_1 x_2 \bar{x}_3, x_2 x_3, \bar{x}_1 x_2 x_3, \bar{x}_1 x_2, (\bar{x}_2 x_3)^*, \bar{x}_2, x_2, \bar{x}_3] .$$

После удаления выводимых сочетаний, истинность которых всегда влечет истинность одного из сочетаний, из которых эти сочетания выводимы, для дальнейшего рассмотрения остаются следующие шесть сочетаний:

$$[x_2 x_3, x_2 \bar{x}_3, \bar{x}_1 x_2 \bar{x}_3, \bar{x}_2 \bar{x}_3, \bar{x}_1 x_2 x_3, (\bar{x}_2 x_3)^*] .$$

После применения к первым двум выводимым сочетаниям частных входных сигналов операции развертывания получим следующие полные входные сигналы, действующие на данном шаге алгоритма работы ДА:

$$[(x_1)x_2 x_3, (x_1)x_2 \bar{x}_3, \bar{x}_1 x_2 \bar{x}_3, \bar{x}_2 \bar{x}_3, \bar{x}_1 x_2 x_3, (\bar{x}_2 x_3)^*] .$$

2.2. Пример детерминизации НДА с построением СКУ и СВФ для автоматов Мура и Мили

В качестве примера детерминизации используется НДА Мура, заданный графом (рис.1.1). В результате детерминизации необходимо получить:

— прямую таблицу переходов ДА Мура, эквивалентного заданному НДА;

— системы канонических уравнений ДА Мура и ДА Мили;

— системы выходных функций (СВФ) ДА Мура и ДА Мили.

а) По графу НДА Мура (рис. 1.1) строим прямую таблицу переходов исходного НДА (ПТП НДА) Мура, которая будет иметь вид (табл. 2.1), где для запрещенного входного сигнала $(\bar{x}_2 x_3)^*$ принято событие перехода S_4 . В нашем примере при построении ПТП ДА событие, принятое в качестве неопределенного будем заключать в скобки, то есть имеем: $S_{(-)} = (S_4)$.

Таблица 2.1

Шаг алгоритма	Исходное частное событие и отмечающий его выходной сигнал $S_i(t)(Y_i)$	Частный входной сигнал на переходе $X_{i,j}(t)$	Событие перехода $S_j(t+1)$
1	2	3	4
1	$S_0(y_0)$	$\bar{x}_1$ x_1	S_0 S_1, S_2
2	$S_1(y_1)$	$\bar{x}_2$ x_2 $\bar{x}_1 x_2$	S_1 S_3 S_2
3	$S_2(y_2)$	$\bar{x}_1$ x_1 $x_1 \bar{x}_2$	S_4 S_3 S_5
4	$S_3(y_3, y_4)$	$\bar{x}_3$ $x_2 x_3$ $(\bar{x}_2 x_3)^*$	S_3 S_4 $S_{(-)} = S_4$
5	$S_4(y_2, y_4)$	$\bar{x}_4$ x_4	S_4 S_3
6	$S_5(y_1, y_3)$	$\bar{x}_4$ x_4	S_4 S_3

б) В соответствии с алгоритмом детерминизации на основании табл.2.1 получим следующую прямую таблицу переходов ДА Мура (табл.2.2).

Таблица 2.2

Шаг алго- ритма	Сочетание (конъюнкция) частных исходных событий в момент времени (t)	Подмножества частных входных сигналов на переходе $[X_{i,j}]$	Сочетание (конъюнкция) частных событий в момент времени (t+1)
	Полное событие $a_m(t)(Y_m)$	Сочетание (конъюнкция) частных входных сигналов на переходе (полный входной сигнал) $X(a_m, a_s)(t)$	Полное событие на переходе $a_s(t+1)$
1	$\frac{S_0(y_0)}{a_0}$	$\bar{x}_1$ x_1	S_0/a_0 S_1S_2/a_1
2	$\frac{S_1S_2(y_1y_2)}{a_1}$	$\frac{[\bar{x}_2, x_2, \bar{x}_1 x_2],}{[\bar{x}_1, x_1, x_1 \bar{x}_2]}$ $\bar{x}_1 \bar{x}_2$ $\bar{x}_1 x_2$ $x_1 \bar{x}_2$ $x_1 x_2$	S_1S_4/a_2 $S_2S_3S_4/a_3$ $S_1S_3S_5/a_4$ S_3/a_5
3	$\frac{S_1S_4(y_1y_2y_4)}{a_2}$	$\frac{[\bar{x}_2, x_2, \bar{x}_1 x_2],}{[\bar{x}_4, x_4]}$ $\bar{x}_2 \bar{x}_4$ $\bar{x}_2 x_4$ $(x_1)x_2 \bar{x}_4$ $(x_1)x_2 x_4$ $\bar{x}_1 x_2 \bar{x}_4$ $\bar{x}_1 x_2 x_4$	S_1S_4/a_2 S_1S_3/a_6 S_3S_4/a_7 S_3/a_5 $S_2S_3S_4/a_3$ S_2S_3/a_8

Продолжение таблицы 2.2

Шаг алго- ритма	Сочетание (конъюнкция) частных исходных событий в момент времени (t)	Подмножества частных входных сигналов на переходе $[X_{i,j}]$	Сочетание (конъюнкция) частных событий в момент времени (t+1)
	Полное событие $a_m(t)(Y_m)$	Сочетание (конъюнкция) частных входных сигналов на переходе (полный входной сигнал) $X(a_m, a_s)(t)$	Полное событие на переходе $a_s(t+1)$
4	$\frac{S_2 S_3 S_4 (y_2 y_3 y_4)}{a_3}$	$[\bar{x}_1, x_1, x_1 \bar{x}_2],$ $[\bar{x}_3, x_2 x_3, (\bar{x}_2 x_3)^*], [\bar{x}_4, x_4]$ $\bar{x}_1 \bar{x}_3 \bar{x}_4$ $x_1 (x_2) \bar{x}_3 \bar{x}_4$ $x_1 \bar{x}_2 \bar{x}_3 \bar{x}_4$ $\bar{x}_1 x_2 x_3 \bar{x}_4$ $x_1 x_2 x_3 \bar{x}_4$ $\bar{x}_1 \bar{x}_3 x_4$ $x_1 (x_2) \bar{x}_3 x_4$ $x_1 \bar{x}_2 \bar{x}_3 x_4$ $\bar{x}_1 x_2 x_3 x_4$ $x_1 x_2 x_3 x_4$ $\bar{x}_1 \bar{x}_2 x_3 \bar{x}_4$ $x_1 \bar{x}_2 x_3 \bar{x}_4$ $\bar{x}_1 \bar{x}_2 x_3 x_4$ $x_1 \bar{x}_2 x_3 x_4$	$S_3 S_4 / a_7$ $S_3 S_4 / a_7$ $S_3 S_4 S_5 / a_{10}$ S_4 / a_9 $S_3 S_4 / a_7$ $S_3 S_4 / a_7$ S_3 / a_5 $S_3 S_5 / a_{11}$ $S_3 S_4 / a_7$ $S_3 S_4 / a_7$ $(S_4) / a_9$ $S_3 (S_4) S_5 / a_{10}$ $S_3 (S_4) / a_7$ $S_3 (S_4) S_5 / a_{10}$

Продолжение таблицы 2.2

Шаг алго- ритма	Сочетание (конъюнкция) частных исходных событий в момент времени (t)	Подмножества частных входных сигналов на переходе $[X_{i,j}]$	Сочетание (конъюнкция) частных событий в момент времени (t+1)
	Полное событие $a_m(t)(Y_m)$	Сочетание (конъюнкция) частных входных сигналов на переходе (полный входной сигнал) $X(a_m, a_s)(t)$	Полное событие на переходе $a_s(t+1)$
5	$\frac{S_1 S_3 S_5 (y_1 y_3 y_4)}{a_4}$	$\begin{aligned} & [\bar{x}_2, x_2, \bar{x}_1 x_2], \\ & \frac{[\bar{x}_3, x_2 x_3, (\bar{x}_2 x_3)^*], [\bar{x}_4, x_4]}{\bar{x}_2 \bar{x}_3 \bar{x}_4} \\ & (x_1) x_2 \bar{x}_3 \bar{x}_4 \\ & \bar{x}_1 x_2 \bar{x}_3 \bar{x}_4 \\ & \bar{x}_1 x_2 x_3 \bar{x}_4 \\ & (x_1) x_2 x_3 \bar{x}_4 \\ & \bar{x}_2 \bar{x}_3 x_4 \\ & (x_1) x_2 \bar{x}_3 x_4 \\ & \bar{x}_1 x_2 \bar{x}_3 x_4 \\ & \bar{x}_1 x_2 x_3 x_4 \\ & (x_1) x_2 x_3 x_4 \\ & \bar{x}_2 x_3 \bar{x}_4 \\ & \bar{x}_2 x_3 x_4 \end{aligned}$	$\begin{aligned} & S_1 S_3 S_4 / a_{12} \\ & S_3 S_4 / a_7 \\ & S_2 S_3 S_4 / a_3 \\ & S_2 S_3 S_4 / a_3 \\ & S_3 S_4 / a_7 \\ & S_1 S_3 / a_6 \\ & S_3 / a_5 \\ & S_2 S_3 / a_8 \\ & S_2 S_3 S_4 / a_3 \\ & S_3 S_4 / a_7 \\ & S_1(S_4) / a_2 \\ & S_1 S_3(S_4) / a_{12} \end{aligned}$
6	$\frac{S_3(y_3 y_4)}{a_5}$	$\begin{aligned} & \frac{[\bar{x}_3, x_2 x_3, (\bar{x}_2 x_3)^*]}{\bar{x}_3} \\ & x_2 x_3 \\ & (\bar{x}_2 x_3) \end{aligned}$	$\begin{aligned} & S_3 / a_5 \\ & S_4 / a_9 \\ & (S_4) / a_9 \end{aligned}$

Продолжение таблицы 2.2

Шаг алго- ритма	Сочетание (конъюнкция) частных исходных событий в момент времени (t)	Подмножества частных входных сигналов на переходе $[X_{i,j}]$	Сочетание (конъюнкция) частных событий в момент времени (t+1)
	Полное событие $a_m(t)(Y_m)$	Сочетание (конъюнкция) частных входных сигналов на переходе (полный входной сигнал) $X(a_m, a_s)(t)$	Полное событие на переходе $a_s(t+1)$
7	$\frac{S_1 S_3 (y_1 y_3 y_4)}{a_6}$	$\frac{[\bar{x}_2, x_2, \bar{x}_1 x_2], [\bar{x}_3, x_2 x_3, (\bar{x}_2 x_3)^*]}{\bar{x}_2 \bar{x}_3 (x_1) x_2 \bar{x}_3 \bar{x}_1 x_2 \bar{x}_3 (x_1) x_2 x_3 \bar{x}_1 x_2 x_3 \bar{x}_2 x_3}$	$S_1 S_3 / a_6$ S_3 / a_5 $S_2 S_3 / a_8$ $S_3 S_4 / a_7$ $S_2 S_3 S_4 / a_3$ $S_1 (S_4) / a_2$
8	$\frac{S_3 S_4 (y_2 y_3 y_4)}{a_7}$	$\frac{[\bar{x}_3, x_2 x_3, (\bar{x}_2 x_3)^*], [\bar{x}_4, x_4]}{\bar{x}_3 \bar{x}_4 \bar{x}_3 x_4 x_2 x_3 \bar{x}_4 x_2 x_3 x_4 \bar{x}_2 x_3 \bar{x}_4 \bar{x}_2 x_3 x_4}$	$S_3 S_4 / a_7$ S_3 / a_5 S_4 / a_9 $S_3 S_4 / a_7$ S_4 / a_9 $S_3 S_4 / a_7$

Продолжение таблицы 2.2

Шаг алго- ритма	Сочетание (конъюнкция) частных исходных событий в момент времени (t)	Подмножества частных входных сигналов на переходе $[X_{i,j}]$	Сочетание (конъюнкция) частных событий в момент времени (t+1)
	Полное событие $a_m(t)(Y_m)$	Сочетание (конъюнкция) частных входных сигналов на переходе (полный входной сигнал) $X(a_m, a_s)(t)$	Полное событие на переходе $a_s(t+1)$
9	$\frac{S_2 S_3 (y_2 y_3 y_4)}{a_8}$	$[\bar{x}_1, x_1, x_1 \bar{x}_2],$ $\frac{[\bar{x}_3, x_2 x_3, (\bar{x}_2 x_3)^*]}{\bar{x}_1 \bar{x}_3}$ $x_1 (x_2) \bar{x}_3$ $x_1 \bar{x}_2 \bar{x}_3$ $\bar{x}_1 x_2 x_3$ $x_1 x_2 x_3$ $\bar{x}_1 \bar{x}_2 x_3$ $x_1 \bar{x}_2 x_3$	$S_3 S_4 / a_7$ S_3 / a_5 $S_3 S_5 / a_{11}$ S_4 / a_9 $S_3 S_4 / a_7$ $(S_4) / a_9$ $S_3 (S_4) S_5 / a_{10}$
10	$\frac{S_4 (y_2 y_4)}{a_9}$	$\bar{x}_4$ x_4	S_4 / a_9 S_3 / a_5
11	$\frac{S_3 S_4 S_5 (y_1 y_2 y_3 y_4)}{a_{10}}$	$\frac{[\bar{x}_3, x_2 x_3, (\bar{x}_2 x_3)^*], [\bar{x}_4, x_4]}{\bar{x}_3 \bar{x}_4}$ $\bar{x}_3 x_4$ $x_2 x_3 \bar{x}_4$ $x_2 x_3 x_4$ $\bar{x}_2 x_3 \bar{x}_4$ $\bar{x}_2 x_3 x_4$	$S_3 S_4 / a_7$ S_3 / a_5 S_4 / a_9 $S_3 S_4 / a_7$ $(S_4) / a_9$ $S_3 (S_4) / a_7$

Продолжение таблицы 2.2

Шаг алгоритма	Сочетание (конъюнкция) частных исходных событий в момент времени (t)	Подмножества частных входных сигналов на переходе $[X_{i,j}]$	Сочетание (конъюнкция) частных событий в момент времени (t+1)
	Полное событие $a_m(t)(Y_m)$	Сочетание (конъюнкция) частных входных сигналов на переходе (полный входной сигнал) $X(a_m, a_s)(t)$	Полное событие на переходе $a_s(t+1)$
12	$\frac{S_3 S_5 (y_1 y_3 y_4)}{a_{11}}$	$\frac{[\bar{x}_3, x_2 x_3, (\bar{x}_2 x_3)^*], [\bar{x}_4, x_4]}{\bar{x}_3 \bar{x}_3}$ $\bar{x}_3 x_4$ $x_2 x_3 \bar{x}_4$ $x_2 x_3 x_4$ $\bar{x}_2 x_3 \bar{x}_4$ $\bar{x}_2 x_3 x_4$	$S_3 S_4 / a_7$ S_3 / a_5 S_4 / a_9 $S_3 S_4 / a_7$ $(S_4) / a_9$ $S_3 (S_4) / a_7$
13	$\frac{S_1 S_3 S_4 (y_1 y_2 y_3 y_4)}{a_{12}}$	$\frac{[\bar{x}_2, x_2, \bar{x}_1 x_2],}{[\bar{x}_3, x_2 x_3, (\bar{x}_2 x_3)^*], [\bar{x}_4, x_4]}$ $\bar{x}_2 \bar{x}_3 \bar{x}_4$ $(x_1) x_2 \bar{x}_3 \bar{x}_4$ $\bar{x}_1 x_2 \bar{x}_3 \bar{x}_4$ $\bar{x}_1 x_2 x_3 \bar{x}_4$ $(x_1) x_2 x_3 \bar{x}_4$ $\bar{x}_2 \bar{x}_3 x_4$ $(x_1) x_2 \bar{x}_3 x_4$ $\bar{x}_1 x_2 \bar{x}_3 x_4$ $\bar{x}_1 x_2 x_3 x_4$ $(x_1) x_2 x_3 x_4$ $\bar{x}_2 x_3 \bar{x}_4$ $\bar{x}_2 x_3 x_4$	$S_1 S_3 S_4 / a_{12}$ $S_3 S_4 / a_7$ $S_2 S_3 S_4 / a_3$ $S_2 S_3 S_4 / a_3$ $S_3 S_4 / a_7$ $S_1 S_3 / a_6$ S_3 / a_5 $S_2 S_3 / a_8$ $S_2 S_3 S_4 / a_3$ $S_3 S_4 / a_7$ $S_1 (S_4) / a_2$ $S_1 S_3 (S_4) / a_{12}$

в) Отметим некоторые особенности, связанные с оформлением табл.2.2. Для отдельных шагов алгоритма работы автомата (шаги алгоритма 3—5, 7, 9, 13) к частным выводимым входным сигналам была применена

операция развертывания. Результат этой операции после удаления входных сигналов, эквивалентных ранее полученным, отмечен заключением элементарных входных сигналов в круглые скобки.

В том случае, если детерминизация НД СКУ выполняется в соответствии с примечанием п. 5 алгоритма детерминизации, то операция развертывания к выводимым входным сигналам не применяется, и для таких входных сигналов ставится в столбце 4 дополнительный знак (). Например, для двух переходов третьего шага алгоритма работы автомата табл. 2.2 имела бы вид

a_2	x_2x_4 x_2x_4	$S_3S_4()/a_7$ $S_3()/a_5$
-------	----------------------	---------------------------------

Для некоторых переходов (шаги алгоритма 4—8, 10—12) сочетания частных входных сигналов включают и запрещенный входной сигнал $(\overline{x_2} \ x_3)^*$, для которого событие перехода может быть определено произвольно. В нашем случае в качестве события перехода для такого сигнала принято событие S_4 . В общем случае вместо неопределенного события перехода, обычно отмечаемого черточкой, принимается такое событие, которое может привести к упрощению выражений СКУ и СВФ.

г) По полученной прямой таблице переходов детерминированного автомата Мура (табл. 2.2), пользуясь рассмотренным ранее алгоритмом построения СКУ по прямой таблице переходов и учитывая указанные выше замечания по оформлению табл. 2.2, получим отмеченную детерминированную СКУ, определяющую все переходы в автомате Мура, заданного графом НДА (рис.1.1). После преобразования этой СКУ с целью ее упрощения за счет минимизации булевых функций, представленных дизъюнкцией частных входных сигналов при одних и тех же состояниях, отмеченная СКУ полностью определенного детерминированного автомата Мура примет следующий вид:

$$\begin{aligned}
a_0^y(t+1) &= a_0 \bar{x}_1 \vee x_0; \\
a_1^y y_2(t+1) &= a_0 x_1; \\
a_2^y y_2 y_4(t+1) &= a_1 \bar{x}_1 \bar{x}_2 \vee a_2 \bar{x}_2 \bar{x}_4 \vee (a_4 \vee a_{12}) \bar{x}_2 x_3 \bar{x}_4 \vee a_6 \bar{x}_2 x_3; \\
a_3^y y_3 y_4(t+1) &= a_1 \bar{x}_1 x_2 \vee a_2 \bar{x}_1 x_2 \bar{x}_4 \vee (a_4 \vee a_{12}) (\bar{x}_1 x_2 \bar{x}_4 \vee \\
&\vee \bar{x}_1 x_2 x_3) \vee a_6 \bar{x}_1 x_2 x_3; \\
a_4^y y_3 y_4(t+1) &= a_1 x_1 \bar{x}_2; \\
a_5^y y_3 y_4(t+1) &= a_1 x_1 x_2 \vee a_2 x_1 x_2 x_4 \vee (a_4 \vee a_{12} \vee a_3) x_1 x_2 \bar{x}_3 x_4 \vee \\
&\vee a_5 \bar{x}_3 \vee (a_6 \vee a_8) x_1 x_2 \bar{x}_3 \vee (a_7 \vee a_{10} \vee a_{11}) \bar{x}_3 x_4 \vee a_9 x_4; \\
a_6^y y_3 y_4(t+1) &= a_2 \bar{x}_2 x_4 \vee (a_4 \vee a_{12}) \bar{x}_2 \bar{x}_3 x_4 \vee a_6 \bar{x}_2 \bar{x}_3; \\
a_7^y y_3 y_4(t+1) &= a_2 x_1 x_2 \bar{x}_4 \vee a_3 (\bar{x}_1 \bar{x}_3 \vee \bar{x}_1 x_4 \vee x_1 x_2 x_3 \vee x_1 x_2 \bar{x}_4) \vee \\
&(a_4 \vee a_{12}) (x_1 x_2 \bar{x}_4 \vee x_1 x_2 x_3) \vee a_6 x_1 x_2 x_3 \vee \\
&\vee (a_7 \vee a_{10} \vee a_{11}) (\bar{x}_3 \bar{x}_4 \vee x_3 x_4) \vee a_8 (\bar{x}_1 \bar{x}_3 \vee x_1 x_2 x_3); \\
a_8^y y_3 y_4(t+1) &= a_2 \bar{x}_1 x_2 x_4 \vee (a_4 \vee a_{12}) \bar{x}_1 x_2 \bar{x}_3 x_4 \vee a_6 \bar{x}_1 x_2 \bar{x}_3; \\
a_9^y y_4(t+1) &= a_3 \bar{x}_1 x_3 \bar{x}_4 \vee a_5 x_3 \vee (a_7 \vee a_{10} \vee a_{11}) x_3 \bar{x}_4 \vee \\
&\vee a_8 \bar{x}_1 x_3 \vee a_9 \bar{x}_4; \\
a_{10}^y y_2 y_3 y_4(t+1) &= a_3 (x_1 \bar{x}_2 x_3 \vee x_1 \bar{x}_2 \bar{x}_4) \vee a_8 x_1 \bar{x}_2 x_3; \\
a_{11}^y y_3 y_4(t+1) &= a_3 x_1 \bar{x}_2 \bar{x}_3 x_4 \vee a_8 x_1 \bar{x}_2 \bar{x}_3; \\
a_{12}^y y_2 y_3 y_4(t+1) &= (a_4 \vee a_{12}) \bar{x}_2 \bar{x}_3 \bar{x}_4 \vee \bar{x}_2 x_3 x_4.
\end{aligned} \tag{2.2}$$

д) Система канонических уравнений детерминированного полностью определенного автомата Мили, эквивалентного автомату Мура, может быть получена из отмеченной СКУ (2.2) путем объединения тех состояний (табл. 2.2), для которых все переходы полностью совпадают. Для нашего примера это соответствует двум группам состояний: (a_4, a_{12}) и (a_7, a_{10}, a_{11}) . Делая необходимые замены и подстановки переменных вида: $a_4 \vee a_{12} = a_4$ и $(a_7 \vee a_{10} \vee a_{11}) = a_7$ в (2.2), получим следующую СКУ автомата Мили:

$$\begin{aligned}
a_0(t+1) &= a_0 \bar{x}_1 \vee x_0; \\
a_1(t+1) &= a_0 x_1; \\
a_2(t+1) &= a_1 \bar{x}_1 \bar{x}_2 \vee a_2 \bar{x}_2 \bar{x}_4 \vee a_4 \bar{x}_2 x_3 \bar{x}_4 \vee a_6 \bar{x}_2 x_3; \\
a_3(t+1) &= a_1 \bar{x}_1 x_2 \vee a_2 \bar{x}_1 x_2 \bar{x}_4 \vee a_4 (\bar{x}_1 x_2 \bar{x}_4 \vee \bar{x}_1 x_2 x_3) \vee a_6 \bar{x}_1 x_2 x_3; \\
a_4(t+1) &= a_1 x_1 \bar{x}_2 \vee a_4 (\bar{x}_2 \bar{x}_3 \bar{x}_4 \vee \bar{x}_2 x_3 x_4); \\
a_5(t+1) &= a_1 x_1 x_2 \vee a_2 x_1 x_2 x_4 \vee (a_3 \vee a_4) x_1 x_2 \bar{x}_3 x_4 \vee \\
&\vee a_5 \bar{x}_3 \vee a_7 \bar{x}_3 x_4 \vee (a_6 \vee a_8) x_1 x_2 \bar{x}_3 \vee a_9 x_4; \\
a_6(t+1) &= a_2 \bar{x}_2 x_4 \vee a_4 \bar{x}_2 \bar{x}_3 x_4 \vee a_6 \bar{x}_2 \bar{x}_3; \\
a_7(t+1) &= a_2 x_1 x_2 \bar{x}_4 \vee a_3 (\bar{x}_1 x_4 \vee \bar{x}_3 \bar{x}_4 \vee x_1 \bar{x}_2 \vee x_1 x_3) \vee \\
&\vee a_6 x_1 x_2 x_3 \vee a_7 (\bar{x}_3 \bar{x}_4 \vee x_3 x_4) \vee a_8 (x_1 \bar{x}_3 \vee x_1 x_3 \vee x_1 \bar{x}_2); \\
a_8(t+1) &= a_2 \bar{x}_1 x_2 x_4 \vee a_4 \bar{x}_1 x_2 \bar{x}_3 x_4 \vee a_6 \bar{x}_1 x_2 \bar{x}_3; \\
a_9(t+1) &= a_3 \bar{x}_1 x_3 \bar{x}_4 \vee a_5 x_3 \vee a_7 x_3 \bar{x}_4 \vee a_8 \bar{x}_1 x_3 \vee a_9 \bar{x}_4.
\end{aligned} \tag{2.3}$$

е) Система выходных функций для детерминированного полностью определенного автомата Мура в соответствии с (2.2) и (1.3) будет иметь вид:

$$\begin{aligned}
y_0 &= a_0; \\
y_1 &= a_1 \vee a_2 \vee a_4 \vee a_6 \vee \bigvee_{k=10}^{k=12} a_k; \\
y_2 &= \bigvee_{k=1}^{k=3} a_k \vee \bigvee_{k=7}^{k=10} a_k \vee a_{12}; \\
y_3 &= \bigvee_{k=3}^{k=8} a_k \vee \bigvee_{k=10}^{k=12} a_k; \\
y_4 &= \bigvee_{k=2}^{k=12} a_k.
\end{aligned} \tag{2.4}$$

ж) Систему выходных функций для детерминированного полностью определенного автомата Мили, эквивалентного автомату Мура, получим исходя из (2.2), (1.4) и принятой системы замены переменных. Она будет иметь такой вид:

$$\begin{aligned}
y_0 &= a_0 \bar{x}_1 \vee x_n; \\
y_1 &= a_0 x_1 \vee (a_1 \vee a_2 \vee a_4 \vee a_6) \bar{x}_2 \vee (a_3 \vee a_8) x_1 \bar{x}_2; \\
y_2 &= a_0 x_1 \vee a_1 \bar{x}_1 \vee a_2 (\bar{x}_1 x_2 \vee \bar{x}_4) \vee a_3 (\bar{x}_1 \vee \bar{x}_4) \vee \\
&\vee a_4 (\bar{x}_1 x_2 \vee \bar{x}_2 \bar{x}_4 \vee \bar{x}_2 x_3) \vee a_6 (x_3 \vee \bar{x}_1 x_2) \vee a_5 x_3 \vee \\
&\vee a_7 \bar{x}_4 \vee a_8 (\bar{x}_1 \vee x_3) \vee a_9 \bar{x}_4; \\
y_3 &= a_1 (x_1 \vee x_2) \vee a_2 (x_2 \vee x_4) \vee a_3 (x_1 \vee \bar{x}_3 \vee x_4) \vee \\
&\vee a_4 (x_2 \vee \bar{x}_3 \vee x_4) \vee a_5 \bar{x}_3 \vee a_6 (x_2 \vee \bar{x}_3) \vee a_7 (\bar{x}_3 \vee x_4) \vee \\
&\vee a_8 (x_1 \vee \bar{x}_3) \vee a_9 x_4; \\
y_4 &= \bigvee_{k=1}^{k=9} a_k.
\end{aligned} \tag{2.5}$$

Контрольные вопросы к главе 2.

1. В чем состоит основной принцип, используемый для построения алгоритма детерминизации НДА?
2. Какие требования предъявляются к построению исходной ПТП НДА, чтобы избежать недостижимых событий?
3. Как можно проверить на каждом шаге алгоритма работы автомата подмножества частных входных сигналов на однозначность и полноту?
4. Что из себя представляют выводимые частные входные сигналы и с какой целью некоторые из них при детерминизации подвергаются коррекции путем использования операции развертывания?
5. В каких случаях при детерминизации можно не выполнять операции развертывания над выводимыми частными входными сигналами?
6. Какой порядок определения полных входных сигналов при детерминизации НДА на каждом шаге работы алгоритма управления? Каким требованиям должны удовлетворять полные входные сигналы?
7. В каких случаях при детерминизации НДА на некоторых шагах работы алгоритма управления могут появиться пустые и неопределенные события?

Глава 3. Минимизация систем канонических уравнений

В данном разделе рассматриваются вопросы минимизации систем канонических уравнений, являющихся математической моделью как детерминированных, так и недетерминированных автоматов.

При формализации алгоритма функционирования автомата на начальных языках, в первую очередь, составляется формальное описание словесного алгоритма и проверяется его адекватность с помощью моделирования. При этом не особенно обращается внимание на возможную избыточность описания. Так при использовании языка ОСА в описании могут быть, например, избыточные операторные и условные вершины. При реализации алгоритма желательно устранение этой избыточности, т. к. она ведет, как правило, к увеличению стоимости реализации. В связи с этим после синтеза СКУ ее необходимо минимизировать.

Минимизация СКУ включает два этапа. Один из них используется для минимизации каждого из уравнений СКУ за счет минимизации булевых функций, представляющих собой ДНФ частных входных сигналов, входящих в уравнения СКУ перед одинаковыми предшествующими событиями. Такая минимизация может быть выполнена с использованием известных методов минимизации булевых функций. В примере детерминированной СКУ, построенной в предыдущем разделе, такая минимизация выполнялась. Второй этап минимизации СКУ связан с минимизацией числа событий, представленных в СКУ. В качестве основного метода минимизации числа событий СКУ в данном пособии рассматривается метод минимизации на основе определения эквивалентного разбиения событий. Кроме этого метода будет рассмотрен еще один метод минимизации СКУ, который базируется на использовании дополнительной информации о входных сигналах, называемой «учет распределения сдвигов» [23-26]. Этот метод в ряде случаев приводит как к минимизации числа событий, так и минимизации частных входных сигналов в СКУ.

В том случае, если СКУ недетерминирована, ее минимизацию целесообразно выполнить до операции детерминизации, т. к. в противном случае детерминизация может потребовать большой объем работ. По этой же причине, если функционирование автомата задано на языке ОСАП, то минимизацию НД СКУ целесообразно выполнять отдельно для каждой частной СКУ, формализующей события в соответствующей ветви алгоритма.

Прежде чем приступить к выполнению основных этапов минимизации СКУ необходимо убедиться, что исходная СКУ не содержит *недостижимых событий*. Для этого необходимо по исходной СКУ построить прямую таблицу переходов, которая должна строиться по такой же процедуре, что и прямая таблица переходов при детерминизации НДА, т. е. построение

прямой таблицы переходов должно начинаться с исходного начального события. При этом очередное событие может быть занесено в первый столбец таблицы в качестве исходного только в том случае, если оно имело место в один из предшествующих шагов алгоритма работы автомата (было зафиксировано в качестве события перехода). Указанная процедура построения прямой таблицы переходов позволит избавиться от всех недостижимых событий. В то же время такая таблица является и исходной информацией для выполнения последующих основных этапов минимизации СКУ.

3.1. Минимизация СКУ на основе определения эквивалентного разбиения событий

3.1.1. Некоторые понятия и определения

Для минимизации числа событий СКУ можно воспользоваться одним из методов, предложенных для минимизации числа состояний автомата, заданного классической таблицей переходов и выходов, основанного на определении эквивалентного разбиения автомата [27]. При использовании таких методов возникает необходимость в их коррекции, которая обусловлена применением частных входных сигналов при описании событий. Прежде чем приступить к рассмотрению методики определения эквивалентного разбиения автомата, введем некоторые понятия и определения, полезные при дальнейшем рассмотрении вопросов минимизации СКУ. Эти понятия и определения базируются на заимствовании аналогичных понятий, применительно к минимизации внутренних состояний автомата [27].

Эквивалентность событий. События S_α и S_β эквивалентны, если при переходе из них под воздействием любой входной последовательности сигналов, формируются одинаковые выходные последовательности сигналов. Эквивалентность событий S_α и S_β будем обозначать равенством $S_\alpha = S_\beta$. Отношения эквивалентности обладают свойствами: рефлексивности ($S_\alpha = S_\alpha$), симметричности (если $S_\alpha = S_\beta$, то $S_\beta = S_\alpha$) и свойством транзитивности (если $S_\alpha = S_\beta$ и $S_\beta = S_\gamma$, то $S_\alpha = S_\gamma$).

K -эквивалентность событий. События S_α и S_β k -эквивалентны, если при переходе из них под воздействием любой входной последовательности сигналов длины k , формируются одинаковые выходные последовательности сигналов. Если события S_α и S_β не являются k -эквивалентными, то они называются k -различимыми.

Преимники (или последователи) событий. K -м преимником события S_α по отношению к входной последовательности сигналов длины k называют событие, которое формируется из события S_α под воздействием входной последовательности сигналов длины k .

Определение эквивалентности одних событий при установленной эквивалентности других событий.

а) Если события S_α и S_β являются k -эквивалентными и если их k -е преемники по отношению к любой входной последовательности сигналов длины k являются эквивалентными, то $S_\alpha = S_\beta$. С другой стороны, можно утверждать, что, если события S_α и S_β являются k -эквивалентными, а их k -е преемники по отношению к любой входной последовательности сигналов длины k являются k -различимыми, то $S_\alpha \neq S_\beta$.

б) Если события S_α и S_β являются эквивалентными, то их k -е преемники по отношению к любой входной последовательности сигналов длины k и для любого k являются эквивалентными.

k -эквивалентное разбиение событий СКУ на классы.

Все множество событий может быть разбито на классы по признакам:

а) Все события, принадлежащие одному классу, должны быть k -эквивалентны, их называют также смежными.

б) Все события, принадлежащие к различным классам, должны быть k -различимыми.

Такое разбиение называют k -эквивалентным разбиением событий СКУ.

Эквивалентные разбиения событий СКУ. Эквивалентным разбиением событий СКУ называется k -эквивалентное разбиение их, когда в каждом из классов этого разбиения смежные события эквивалентны. Такое разбиение в принципе может быть получено, если последовательно получать k -эквивалентные разбиения событий, начиная с $k=1$ для автомата Мили и $k=0$ для автомата Мура. Такие разбиения последовательно получают до тех пор, пока первый раз не получается разбиение, которое совпадает с предыдущим. Это разбиение событий СКУ и будет эквивалентным разбиением. Для определения такого разбиения событий могут быть использованы различные методы. Рассмотрим один из них, который основан на использовании так называемой таблицы пар [27].

3.1.2. Определение эквивалентного разбиения событий СКУ на основе использования таблицы пар

Такое разбиение выполняется последовательным изменением таблицы пар путем вычеркивания по определенному алгоритму тех пар событий, которые не являются эквивалентными. Оставшиеся не вычеркнутые пары будут образовывать все пары эквивалентных событий.

Методику построения таблицы пар и ее использования для определения эквивалентного разбиения событий СКУ будем рассматривать на основе примера автомата Мура, заданного прямой таблицей переходов (табл. 3.1), в которой все события как исходные, так и события перехода, отмечены выходными сигналами.

Таблица 3.1

Шаг алгоритма	Исходное событие $S_i(t)(Y_i)$	Входной сигнал $X_{i,j}(t)$	Событие перехода $S_j(t+1)(Y_j)$
1	$S_0(y_0)$	1	$S_1(y_1)$
2	$S_1(y_1)$	x_1 $\bar{x}_1$	$S_2(y_2)$ $S_3(y_3)$
3	$S_2(y_2)$	1	$S_3(y_3)$
4	$S_3(y_3)$	1	$S_4(y_1)$
5	$S_4(y_1)$	x_1 $\bar{x}_1 x_2$ $\bar{x}_1 \bar{x}_2$	$S_5(y_2)$ $S_6(y_3)$ $S_7(y_3)$
6	$S_5(y_2)$	1	$S_8(y_1)$
7	$S_6(y_3)$	x_3 $\bar{x}_3$	$S_9(y_4)$ $S_{10}(y_5)$

Шаг алгоритма	Исходное событие $S_i(t)(Y_i)$	Входной сигнал $X_{i,j}(t)$	Событие перехода $S_j(t+1)(Y_j)$
8	$S_7(y_3)$	x_3 $\bar{x}_3$	$S_9(y_4)$ $S_{10}(y_5)$
9	$S_8(y_1)$	x_1 $\bar{x}_1$	$S_5(y_2)$ $S_7(y_3)$
10	$S_9(y_4)$	1	$S_k(y_k)$
11	$S_{10}(y_5)$	x_3 $\bar{x}_3$	$S_9(y_4)$ $S_{10}(y_5)$

Такое представление исходной прямой таблицы переходов ЦА Мура позволит использовать ее для построения таблицы пар при определении эквивалентного разбиения событий как для ЦА Мура, так и для эквивалентного ему автомата Мили. Разница будет заключаться лишь в построении основного столбца таблицы пар: для автомата Мура в основном столбце размещают все пары 0-эквивалентных событий S_α и S_β (для $\alpha \neq \beta$), а для автомата Мили — все пары 1-эквивалентных событий.

Напомним, что под 0-эквивалентными событиями понимаются те события, которые отмечены во втором столбце прямой таблицы переходов одинаковыми выходными сигналами, а под 1-эквивалентными событиями понимают события (во втором столбце), при переходе из которых под действием одного и того же входного сигнала формируются одинаковые выходные сигналы (в четвертом столбце).

Для нашего примера имеются три класса 0-эквивалентных событий: (S_1, S_4, S_8) , (S_2, S_5) , (S_3, S_6, S_7) и три класса 1-эквивалентных событий: (S_1, S_4, S_8) , (S_0, S_3, S_5) , (S_6, S_7, S_{10}) .

Определение эквивалентного разбиения событий для построения минимального ЦА Мили.

Рассматриваемая методика определения эквивалентного разбиения событий на основе использования ПТП ЦА Мура позволяет, наряду с процедурой минимизации числа событий, выполнить также и одновременное преобразование автомата Мура в эквивалентный ему автомат Мили с минимальным числом событий.

Для более компактного представления таблицы пар ее целесообразно представлять в виде подтаблиц, число которых определится для ЦА Мили числом классов 1-эквивалентных событий.

Как было отмечено выше, для автомата Мили основной столбец таблицы пар будет содержать все пары 1-эквивалентных событий. Во все

последующие колонки таблицы пар записываются пары событий, которые являются первыми преемниками по отношению к конкретному входному сигналу. Для этой цели колонки таблицы пар обозначают теми входными сигналами, которые не являются выводимыми ни из каких других входных сигналов.

Для их определения необходимо:

- сначала найти по исходной ПТП (табл.3.1) все возможные сочетания входных сигналов для каждого конкретного класса 1-эквивалентных событий;
- из полученных сочетаний входных сигналов используют для обозначения колонок соответствующей подтаблицы те из них, которые не являются выводимыми ни из каких других сочетаний сигналов. Например, сигнал $\bar{x}_1$ выводим из сигналов $\bar{x}_1 x_2$ и $\bar{x}_1 \bar{x}_2$, поэтому сигналом $\bar{x}_1$ не отмечается столбец таблицы пар.

Предусматривается также одна колонка, для которой переходы к первым преемникам не зависят от входного сигнала. Учитывая изложенное, таблица пар будет иметь следующий вид (табл. 3.2). В каждой клетке для упрощения таблицы записываются только индексы, определяющие номера событий.

Таблица 3.2

Классы 1-эквивалентных событий	Пары 1-эквивалентных событий	$\bar{x}_1 x_2$	$\bar{x}_1 \bar{x}_2$	x_1	x_3	$\bar{x}_3$	1
	1	2	3	4	5	6	7
S_1, S_4, S_8	1-4	3-6	3-7	2-5			1
	1-8	3-7	3-7	2-5			2
	4-8	6-7	7-7	5-5			3
S_0, S_3, S_5	0-3						4
	0-5						5
	3-5						6
S_6, S_7, S_{10}	6-7				9-9	10-10	7
	6-10				9-9	10-10	8
	7-10				9-9	10-10	9

Алгоритм нахождения эквивалентного разбиения событий будет включать следующие этапы:

1) Последовательно по строкам отыскиваются отличающиеся пары событий (S_α и S_β , где $\alpha \neq \beta$), которые отсутствуют в первом основном столбце таблицы пар. Если в какой-либо строке имеется хотя бы одна такая пара, то в этой строке зачеркивается пара в первом столбце. Такие строки, в которых зачеркнуты пары в первом столбце таблицы, называют выделенными строками. Для нашего примера это пары (3-6), (3-7), (2-5). Для них зачеркиваем в первом столбце таблицы пары (1-4) и (1-8), они выделены в таблице жирным шрифтом.

2) Отыскиваются невыделенные строки, в которых имеются пары, зачеркнутые в первом столбце на предыдущем этапе. Если такие строки имеются, то для них зачеркиваются пары в первом столбце. Для нашего примера это строки 4 и 5, для них зачеркиваем пары в первом столбце (0-3) и (0-5). Такой процесс повторяется до тех пор, пока на очередном этапе не обнаруживаются невыделенные строки, в которых имеются пары, зачеркнутые в первом столбце на предыдущем этапе. Оставшиеся не зачеркнутые пары образуют все пары эквивалентных событий. Для нашего примера это пары: (S_4, S_8), (S_3, S_5), (S_6, S_7), (S_6, S_{10}), (S_7, S_{10}).

Рассмотренная методика определения пар эквивалентных событий, основанная на последовательном зачеркивании пар неэквивалентных событий, базируется на использовании введенных ранее свойств (а) и (б) по определению эквивалентности одних событий при установленной эквивалентности других событий.

Учитывая свойства транзитивности для эквивалентных событий, а также состав событий, которые не вошли в пары эквивалентных событий, получим для нашего примера следующее эквивалентное разбиение событий ЦА Мили: (S_0), (S_1), (S_2), (S_3, S_5), (S_4, S_8), (S_6, S_7, S_{10}), (S_9), (S_k).

Определение эквивалентного разбиения событий для построения минимального ЦА Мура.

В том случае, если требуется построить минимальный ЦА Мура, то для определения эквивалентного разбиения событий, как было отмечено ранее, необходимо по исходной ПТП ЦА Мура (табл.3.1) найти все классы 0-эквивалентных событий, которые будут использованы для построения основного столбца таблицы пар.

Дальнейшая методика построения таблицы пар для автомата Мура ничем не отличается от рассмотренной методики для автомата Мили.

Для нашего примера таблица пар для определения эквивалентного разбиения событий для построения минимального ЦА Мура будет иметь следующий вид (табл.3.3).

Таблица 3.3

Классы 0-эквивалентных событий	Пары 0-эквивалентных событий	$\overline{x_1} x_2$	$\overline{x_1} \overline{x_2}$	x_1	x_3	$\overline{x_3}$	1
	1	2	3	4	5	6	7
S_1, S_4, S_8	1-4	3-6	3-7	2-5			1
	1-8	3-7	3-7	2,5			2
	4-8	6-7	7-7	5,5			3
S_2, S_5	2-5						4
S_3, S_6, S_7	3-6				4-9	4-10	5
	3-7				4-9	4-10	6
	3-7				9-9	10-10	7

На первом шаге алгоритма отыскания эквивалентного разбиения событий находим пары (3-8) (4-9), которые отсутствуют в основном столбце. Для таких пар зачеркиваются соответствующие им пары в основном столбце, к ним относятся пары: (2-5), (3-6), (3-7).

На втором шаге алгоритма отыскиваются строки, в которых имеются пары, зачеркнутые в основном столбце на первом шаге алгоритма. Для нашего примера это строки 1-я и 2-я, для них зачеркиваются пары (1-4) и (1-8).

Для нашего примера на этом шаге алгоритма отыскания эквивалентного разбиения событий свою работу заканчивает, так как больше не обнаруживаются невыделенные строки, в которых имеются пары, зачеркнутые в основном столбце на предыдущем шаге алгоритма. Оставшиеся незачеркнутые пары событий образуют пары эквивалентных событий, к ним относятся пары: (S_4, S_8) и (S_6, S_7) .

Учитывая состав событий, которые не вошли в пары эквивалентных событий, получим следующее эквивалентное разбиение событий для ЦА Мура для нашего примера:

$$(S_0), (S_1), (S_2), (S_3), (S_4, S_8), (S_6, S_7), (S_9), (S_k), (S_5), (S_{10}).$$

3.1.3. Определение минимальной прямой таблицы переходов и минимальной SKU для автомата Мили и Мура

Цифровой автомат Мили. Обозначив произвольно каждый класс эквивалентного разбиения событий ЦА Мили своим символом и заменив в исходной ПТП (табл.3.1) каждое обозначение событий, входящего в класс, обозначением введенного символа для этого класса и выполнив зачеркивание повторяющихся строк, получим минимальную прямую таблицу переходов ЦА Мили. Примем для нашего примера следующие обозначения для классов эквивалентного разбиения:

$$(S_0)=a_0; \quad (S_1)=a_1; \quad (S_2)=a_2; \quad (S_3, S_5)=a_3; \\ (S_4, S_8)=a_4; \quad (S_6, S_7, S_{10})=a_5; \quad (S_9)=a_6; \quad (S_k)=a_7.$$

После подстановки введенных обозначений в табл. 3.1 и вычеркивания повторяющихся строк, получим следующую минимальную прямую таблицу переходов для автомата Мили (табл.3.4).

Таблица 3.4

Шаг алгоритма	$a_i(t)$	$X_{i,j}(t)$	$a_j(t+1)$	$Y_{i,j}(t)$
1	2	3	4	5
1	a_0	1	a_1	y_1
2	a_1	x_1	a_2	y_2
		$\bar{x}_1$	a_3	y_3
3	a_2	1	a_3	y_3
4	a_3	1	a_4	y_1
5	a_4	x_1	a_3	y_2
		$\bar{x}_1$	a_5	y_3
6	a_5	x_3	a_6	y_4
		$\bar{x}_3$	a_5	y_5
7	a_6	1	a_7	y_k

Минимальную СКУ для автомата Мили и соответствующую СВФ можно получить одним из двух способов. В первом случае, если минимальная прямая таблица переходов уже построена, то по ней не трудно построить минимальную СКУ и СВФ. Во втором случае, если минимальная прямая таблица переходов не строилась, а исходный автомат был задан СКУ и СВФ, то минимальную СКУ и СВФ можно построить путем склеивания (объединения) правых частей уравнений для одинаково обозначенных событий и подстановки новых обозначений в исходные СКУ и СВФ. Для нашего примера имеем:

$$S_0=a_0; \quad S_1=a_1; \quad S_2=a_2; \quad S_3 \vee S_5=a_3; \quad S_4 \vee S_8=a_4;$$

$$S_6 \vee S_7 \vee S_{10}=a_5; \quad S_9=a_6; \quad S_k=a_7.$$

Откуда получим минимальные СКУ и СВФ автомата Мили:

$$a_1(t+1) = a_0;$$

$$y_1 = a_0 \vee a_3;$$

$$a_2(t+1) = a_1 x_1;$$

$$y_2 = (a_1 \vee a_4) x_1;$$

$$a_3(t+1) = a_2 \vee a_1 \bar{x}_1 \vee a_4 x_1;$$

$$y_3 = a_1 \bar{x}_1 \vee a_2 \vee a_4 \bar{x}_1;$$

$$a_4(t+1) = a_3;$$

$$y_4 = a_5 x_3;$$

$$a_5(t+1) = a_4 \bar{x}_1 \vee a_5 \bar{x}_3;$$

$$y_5 = a_5 \bar{x}_3;$$

$$a_6(t+1) = a_5 x_3;$$

$$y_k = a_6.$$

$$a_7(t+1) = a_6.$$

Цифровой автомат Мура. Минимальная таблица переходов автомата Мура для нашего примера с учетом эквивалентности двух пар событий и введенных обозначений для классов эквивалентного разбиения:

$$(S_0)=a_0, \quad (S_1)=a_1, \quad (S_2)=a_2, \quad (S_3)=a_{3,1}, \quad (S_4, S_8)=a_4, \\ (S_6, S_7)=a_{5,1}, \quad (S_9)=a_6, \quad (S_k)=a_7, \quad (S_5)=a_{3,2}, \quad (S_{10})=a_{5,2}$$

будет иметь следующий вид (табл.3.5):

Таблица 3.5

	$a_i(t)$	$X_{i,j}(t)$	$a_j(t+1)(Y_j)$
1	2	3	4
1	a_0	1	$a_1(y_1)$
2	a_1	x_1	$a_2(y_2)$
		$\bar{x}_1$	$a_{3,1}(y_3)$
3	a_2	1	$a_{3,1}(y_3)$
4	$a_{3,1}$	1	$a_4(y_1)$
5	$a_{3,2}$	1	$a_4(y_1)$
6	a_4	x_1	$a_{3,2}(y_2)$
		$\bar{x}_1$	$a_{5,1}(y_3)$
7	$a_{5,1}$	x_3	$a_6(y_4)$
		$\bar{x}_3$	$a_{5,2}(y_5)$
8	$a_{5,2}$	x_3	$a_6(y_4)$
		$\bar{x}_3$	$a_{5,2}(y_5)$
9	a_6	1	$a_7(y_k)$

В соответствии с табл.3.5 минимальная отмеченная СКУ и СВФ автомата Мура имеют вид:

$$\begin{aligned} a_1^{y_1}(t+1) &= a_0; & y_1 &= a_1; \\ a_2^{y_2}(t+1) &= a_1 x_1; & y_2 &= a_2 \vee a_{3,2}; \\ a_{3,1}^{y_3}(t+1) &= a_1 \bar{x}_1 \vee a_2; & y_3 &= a_{3,1} \vee a_{5,1}; \\ a_{3,2}^{y_2}(t+1) &= a_4 x_1; & y_4 &= a_6; \\ a_4^{y_1}(t+1) &= a_{3,1} \vee a_{3,2}; & y_5 &= a_{5,2}; \\ a_{5,1}^{y_3}(t+1) &= a_4 \bar{x}_1; & y_k &= a_7. \\ a_{5,2}^{y_5}(t+1) &= (a_{5,1} \vee a_{5,2}) \bar{x}_3; \\ a_6^{y_4}(t+1) &= (a_{5,1} \vee a_{5,2}) x_3; \\ a_7^{y_k}(t+1) &= a_6. \end{aligned}$$

В заключение отметим, что в случае, если структуру управляющего автомата необходимо строить на основе модели ЦА Мили с минимальным числом событий, а исходный алгоритм управления задан моделью ЦА Мура, то минимизацию числа событий ЦА Мили можно выполнить на основе

использования модели исходного ЦА Мура без предварительного преобразования его в эквивалентный ЦА Мили. Это обеспечивается тем, что рассмотренная методика минимизации числа событий автоматов на основе использования ПТП ЦА Мура позволяет объединить события как за счет непосредственно минимизации событий ЦА Мура (для нашего примера: объединение событий $S_4 \vee S_8 = a_4$ и $S_6 \vee S_7 = a_5$), так и за счет одновременного преобразования ЦА Мура в эквивалентный ему ЦА Мили (для нашего примера: объединение событий $a_{5,1} \vee a_{5,2} = a_5 = S_6 \vee S_7 \vee S_{10}$ и $a_{3,1} \vee a_{3,2} = a_3 = S_3 \vee S_5$).

3.2. Минимизация СКУ на основе учета распределения сдвигов

3.2.1. Методика минимизации СКУ

В реальных цифровых управляющих автоматах во многих случаях в каждом такте работы автомата могут изменяться не все элементарные входные сигналы по сравнению со значениями их в предшествующем такте. Некоторые элементарные входные сигналы в соответствии с алгоритмом функционирования автомата каждый раз после реализации определенного события S_α могут сохранять свое значение, полученное в предыдущем такте. Такую дополнительную информацию о работе автомата при описании алгоритма его функционирования на языке ОСА, называемую распределением сдвигов, используют для упрощения записи ОСА [23-26].

В тех случаях, когда по условиям распределения сдвигов после появления любого события возможно изменение всех элементарных входных сигналов, никакого упрощения записи СКУ рассматриваемым способом получить нельзя.

Дальнейшее рассмотрение методики минимизации СКУ будет базироваться на работе [28].

Если условия распределения сдвигов заданы, то минимизация СКУ с учетом распределения сдвигов может быть достигнута или за счет сокращения числа реализуемых в автомате событий или их отдельных ветвей, или за счет сокращения записи частных входных сигналов, когда некоторые элементарные входные сигналы, входящие в частный входной сигнал, окажутся равными единице.

Будем предполагать, что СКУ состоит из нескольких уравнений, каждое из которых описывает событие, состоящее из нескольких ветвей:

$$S_\alpha = \bigvee_{i=1}^k S_\alpha^i,$$

где α - номер события, входящего в СКУ;

k — число ветвей в событии S_α .

Любая i -я ветвь события S_α может быть представлена в следующем виде:

$$S_\alpha^i(t+1) = X_\alpha^i(t) \& S_{\beta,i}(t), \quad (3.1)$$

где X_α^i - частный входной сигнал в i -й ветви события S_α ;

$S_{\beta,i}$ - сокращенное обозначение события, непосредственно предшествующего событию S_α в i -й ветви.

Событие $S_{\beta,i}$, в общем случае также может содержать несколько ветвей, каждая из которых может быть представлена в виде формулы, аналогичной формуле (3.1), т. е. имеем:

$$S_{\beta,i} = \bigvee_{j=1}^{\tau} S_{\beta,i}^j, S_{\beta,i}^j(t+1) = X_{\beta,i}^j(t) \& S_{\varepsilon,j}(t),$$

где τ - число ветвей в событии $S_{\beta,i}$.

Пусть распределение сдвигов для любого события $S_{\beta,i}$ будет задано в виде следующего множества элементарных входных сигналов:

$$\Delta_{\beta,i} = [x_t, \dots, x_l, \dots, x_p, \dots]. \quad (3.2)$$

Это означает, что после выполнения события $S_{\beta,i}$ указанные в множестве $\Delta_{\beta,i}$ элементарные входные сигналы сохраняют свое значение в следующем такте, т. е. в данном случае для удобства дальнейшего изложения множество $\Delta_{\beta,i}$ интерпретируется по иному, чем в [24, 25].

Для анализа заданной СКУ с целью ее минимизации необходимо для каждой ветви события S_α , например, для S_α^i определить ту составляющую часть входного сигнала X_α^i , значение которой остается неизменным после выполнения события $S_{\beta,i}$. Здесь следует иметь в виду, что если $X_\alpha^i = 1$, то дальнейший анализ с целью упрощения S_α^i не производится.

Первый этап анализа S_α^i может быть выполнен с помощью следующей операции пересечения:

$$[X_\alpha^i] \cap [\Delta_{\beta,i}] \quad (3.3)$$

где $[X_\alpha^i]$ — множество элементарных входных сигналов, входящих в частный входной сигнал X_α^i ;

$\Delta_{\beta,i} = [\tilde{x}_t, \dots, \tilde{x}_l, \dots, \tilde{x}_p, \dots]$ — множество элементарных входных сигналов, входящих в (3.2), но включающее также кроме переменных и их инверсии.

Переменные, полученные в результате выполнения операции пересечения (3.3), объединим знаком конъюнкции. В результате получим часть входного сигнала X_α^i , которая остается в соответствии с заданным $\Delta_{\beta,i}$

неизменной после выполнения события $S_{\beta,i}$, являющегося непосредственно предшествующим событию S_{α}^i . Обозначим этот частный входной сигнал символом $X_{\alpha}^i(\Delta_{\beta,i})$.

Второй этап анализа с целью упрощения записи S_{α}^i производится, если выражение (3.3) не является пустым, т. е. не равно $\emptyset$. На этом этапе необходимо отыскать общую часть частных входных сигналов для событий всех ветвей, образующих событие $S_{\beta,i}$. Для этого может быть использована следующая операция пересечения:

$$[X_{\beta,i}^1] \cap [X_{\beta,i}^2] \cap \dots \cap [X_{\beta,i}^j] \cap \dots \cap [X_{\beta,i}^{\tau}] \quad (3.4)$$

где $[X_{\beta,i}^j]$ - множество элементарных входных сигналов, входящих в сигнал

$X_{\beta,i}^j$, который в свою очередь входит в описание j -й ветви события $S_{\beta,i}$. Необходимо при этом иметь ввиду, что, если $X_{\beta,i}^j = 1$, то множество $[X_{\beta,i}^j]$ включает в себя все элементарные входные сигналы, как с отрицанием, так и без отрицаний (так как дизъюнкция всех 2^L конститuent единицы равна единице).

Дальнейший анализ может идти в двух направлениях: первое направление, когда число параллельных ветвей в событии $S_{\beta,i}$ не менее двух, причем, не менее одного входного сигнала $X_{\beta,i}^j \neq 1$; второе направление, когда для всех ветвей в событии $S_{\beta,i}$, все входные сигналы $X_{\beta,i}^j = 1$.

Для первого направления анализа переменные, полученные в результате выполнения операции (3.4) и объединенные знаком конъюнкции, обозначим символом $X_{\beta,i}^{1,2,\dots,\tau}$. Этот входной сигнал является общей частью входных сигналов всех ветвей события $S_{\beta,i}$.

Если выражение (3.4) не является пустым, то в результате пересечения и конъюнкции двух полученных выше входных сигналов найдем условия упрощения записи исследуемого события

$$[X_{\alpha}^i(\Delta_{\beta,i})] \cap [X_{\beta,i}^{1,2,\dots,\tau}]; \quad (3.5,a)$$

$$X_{\alpha}^i(\Delta_{\beta,i}) \& X_{\beta,i}^{1,2,\dots,\tau}, \quad (3.5,b)$$

где оба множества в (3.5,a) включают в себя элементарные входные сигналы как с отрицанием, так и без отрицания.

Если выражение (3.5,a) равно $\emptyset$, то запись события S_{α}^i не упрощается, в противном случае могут быть два варианта: если конъюнкция (3.5,b) равна нулю, то входной сигнал $X_{\alpha}^i = 0$, а следовательно, и событие $S_{\alpha}^i = 0$; если конъюнкция (3.5,b) представляет некоторое выражение, не равное нулю, то

входящие в это выражение элементарные входные сигналы с отрицанием или без отрицания заменяются единицей в записи входного X_{α}^i сигнала события S_{α}^i .

Для второго направления анализа необходимо найти пересечение

$$[X_{\alpha}^i] \cap [\Delta_{\beta,i}] \cap [\Delta_{\varepsilon,j}], \quad (3.6)$$

где $\Delta_{\varepsilon,j}$ - распределение сдвигов для события $S_{\varepsilon,j}$, являющегося для события $S_{\beta,i}^j$ непосредственно предшествующим.

Для простоты дальнейшего анализа будем рассматривать событие $S_{\beta,i}$, состоящим только из одной ветви. Переменные, полученные в результате выполнения операции пересечения (3.6), объединим знаком конъюнкции и обозначим символом $X_{\alpha}^i(\Delta_{\beta,i}; \Delta_{\varepsilon,j})$. В дальнейшем находим выражение $X_{\varepsilon,j}^{1,2,\dots,\gamma}$, аналогичное выражению $X_{\beta,i}^{1,2,\dots,\tau}$, но полученное, исходя из анализа следующего пересечения:

$$[X_{\varepsilon,j}^1] \cap [X_{\varepsilon,j}^2] \cap \dots \cap [X_{\varepsilon,j}^p] \cap \dots \cap [X_{\varepsilon,j}^{\gamma}], \quad (3.7)$$

где $[X_{\varepsilon,j}^p]$ - множество элементарных входных сигналов, входящих в сигнал $X_{\varepsilon,j}^p$, который в свою очередь входит в описание p -й ветви события $S_{\varepsilon,j}$;

γ - число ветвей события $S_{\varepsilon,j}$.

Если выражение (3.7) не является пустым, то аналогично выражениям (3.5,а) и (3.5,б) определим пересечение и конъюнкцию

$$[X_{\alpha}^i(\Delta_{\beta,i}; \Delta_{\varepsilon,j})] \cap [X_{\varepsilon,j}^{1,2,\dots,\gamma}], \quad (3.8,а)$$

$$X_{\alpha}^i(\Delta_{\beta,i}; \Delta_{\varepsilon,j}) \& X_{\varepsilon,j}^{1,2,\dots,\gamma}. \quad (3.8,б)$$

В дальнейшем, используя (3.8,а) и (3.8.б), поступают аналогично рассмотренному выше. В том случае, если найдено, что событие S_{α}^i равно нулю, то необходимо произвести коррекцию СКУ, т. е. вычеркнуть строку, описывающую это событие и приравнять нулю ветви тех событий, в которые S_{α}^i входит в качестве непосредственно предшествующего события.

3.2.2. Пример минимизации СКУ

Проиллюстрируем рассмотренную методику минимизации СКУ с учетом заданного распределения сдвигов на примере СКУ, полученной по МСА из [23].

$$\begin{aligned}
S_1(t+1) &= \bar{x}_1 S_0 \vee \bar{x}_1 \bar{x}_4 S_3 \vee \bar{x}_1 x_2 \bar{x}_3 S_1; \\
S_2(t+1) &= S_5; \\
S_3(t+1) &= x_1 x_2 S_1 \vee x_1 \bar{x}_2 S_0; \\
S_4(t+1) &= \bar{x}_2 x_4 S_2 \vee \bar{x}_1 x_4 S_3; \\
S_5(t+1) &= x_1 x_2 S_0 \vee x_2 x_4 S_2 \vee x_2 x_4 S_4 \vee \bar{x}_1 x_2 x_3 S_1; \\
S_6(t+1) &= \bar{x}_4 S_2 \vee \bar{x}_2 S_1 \vee x_1 S_3 \vee (\bar{x}_2 \vee \bar{x}_4) S_4.
\end{aligned} \tag{3.9}$$

Распределение сдвигов для заданной СКУ:

$$\begin{aligned}
\Delta_0 &= [\emptyset]; & \Delta_4 &= [\tilde{x}_4]; \\
\Delta_1 &= [\tilde{x}_1]; & \Delta_5 &= [\tilde{x}_2, \tilde{x}_3]; \\
\Delta_2 &= [\tilde{x}_1, \tilde{x}_2, \tilde{x}_3, \tilde{x}_4]; & \Delta_6 &= [\emptyset]. \\
\Delta_3 &= [\tilde{x}_1, \tilde{x}_2, \tilde{x}_4];
\end{aligned} \tag{3.10}$$

Произведем поочередный анализ событий исходной СКУ (3.9):

1) Событие S_1 состоит из трех ветвей:

а) Ветвь $S_1^1(t+1) = \bar{x}_1 S_0$. Для этой ветви имеем:
 $[X_1^1] \cap [\Delta_0] = [\bar{x}_1] \cap [\emptyset] = \emptyset$. Поэтому запись этой ветви остается без изменения.

б) Ветвь $S_1^2(t+1) = \bar{x}_1 \bar{x}_4 S_3$. Для этой ветви имеем:
 $[X_1^2] \cap [\Delta_3] = [\bar{x}_1, \bar{x}_4]$, $X_1^2(\Delta_3) = \bar{x}_1 \bar{x}_4$.

Для события S_3 находим $[X_3^1] \cap [X_3^2] = [x_1]$, $X_3^{1,2} = x_1$.

Откуда $[X_1^2(\Delta_3)] \cap [X_3^{1,2}] \neq [\emptyset]$, $X_1^2(\Delta_3) \& X_3^{1,2} = 0$, поэтому $X_1^2 = 0$ и $S_1^2 = 0$.

в) Ветвь $S_1^3(t+1) = \bar{x}_1 x_2 \bar{x}_3 S_1$. Для этой ветви имеем:
 $[X_1^3] \cap [\Delta_1] = [\bar{x}_1]$, $X_1^3(\Delta_1) = \bar{x}_1$.

Для события S_1 находим $[X_1^1] \cap [X_1^2] \cap [X_1^3] = [\bar{x}_1]$, $X_3^{1,2,3} = \bar{x}_1$. Откуда $[X_1^3(\Delta_1)] \cap [X_3^{1,2,3}] \neq [\emptyset]$, $X_1^3(\Delta_1) \& X_3^{1,2,3} = \bar{x}_1$, поэтому новая запись входного сигнала для события S_1^3 будет иметь вид $X_1^3 = x_2 \bar{x}_3$.

2) Событие S_2 имеет одну ветвь, для которой $X_2^1 = 1$, поэтому дальнейший анализ не производится.

3) Событие S_3 состоит из двух ветвей:

а) Ветвь $S_3^1(t+1) = x_1 x_2 S_1$. Для этой ветви имеем:
 $[X_3^1] \cap [\Delta_1] = [x_1]$, $X_3^1(\Delta_1) = x_1$.

Для события S_1 в п.1,в получено $X_1^{1,2,3} = \bar{x}_1$. Откуда $[X_3^1(\Delta_1)] \cap [X_1^{1,2,3}] = [\emptyset]$, $X_3^1(\Delta_1) \& X_1^{1,2,3} = 0$, поэтому $X_3^1 = 0$ и ветвь $S_3^1 = 0$.

б) Ветвь $S_3^2(t+1) = x_1 \bar{x}_2 S_0$. Для этой ветви имеем: $[X_3^2] \cap [\Delta_0] = \emptyset$, поэтому дальнейший анализ не производится.

4) Событие S_4 состоит из двух ветвей:

а) Ветвь $S_4^1(t+1) = \bar{x}_2 x_4 S_2$. Для этой ветви имеем:
 $[X_4^1] \cap [\Delta_2] = [\bar{x}_2, x_4]$, $X_4^1(\Delta_2) = \bar{x}_2 x_4$.

Для события S_2 имеем $X_2^1 = 1$. Поэтому рассматриваем событие S_5 , являющееся непосредственно предшествующим событию S_2 .

В этом случае согласно операции (3.6) получим
 $[X_4^1] \cap [\Delta_2] \cap [\Delta_5] = [\bar{x}_2]$, $X_4^1(\Delta_2, \Delta_5) = \bar{x}_2$.

Далее для всех ветвей события S_5 имеем:
 $[X_5^1] \cap [X_5^2] \cap [X_5^3] \cap [X_5^4] = [x_2]$, $X_5^{1,2,3,4} = x_2$.

Откуда $[X_4^1(\Delta_2, \Delta_5)] \cap [X_5^{1,2,3,4}] \neq \emptyset$, $X_4^1(\Delta_2, \Delta_5) \& X_5^{1,2,3,4} = 0$, поэтому $X_4^1 = 0$ и ветвь $S_4^1 = 0$.

б) Ветвь $S_4^2(t+1) = \bar{x}_1 x_4 S_3$. Для этой ветви имеем:
 $[X_4^2] \cap [\Delta_3] = [\bar{x}_1, x_4]$, $X_4^2(\Delta_3) = \bar{x}_1 x_4$.

Для события S_3 в п. 1,б получено $X_3^{1,2} = x_1$. Откуда
 $[X_4^2(\Delta_3)] \cap [X_3^{1,2}] \neq \emptyset$, $X_4^2(\Delta_3) \& X_3^{1,2} = 0$, поэтому $X_4^2 = 0$ и $S_4^2 = 0$.

5) Событие S_5 состоит из четырех ветвей:

а) Ветвь $S_5^1(t+1) = x_1 x_2 S_0$. Для этой ветви имеем $[X_5^1] \cap [\Delta_0] = \emptyset$. Поэтому запись этой ветви остается без изменения.

б) Ветвь $S_5^2(t+1) = x_2 x_4 S_2$. Для этой ветви имеем:
 $[X_5^2] \cap [\Delta_2] = [x_2, x_4]$, $X_5^2(\Delta_2) = x_2 x_4$.

Для события S_2 , как и в п.4,а находим
 $[X_5^2] \cap [\Delta_2] \cap [\Delta_5] = [x_2]$, $X_5^2(\Delta_2, \Delta_5) = x_2$. Откуда
 $[X_5^2(\Delta_2, \Delta_5)] \cap [X_5^{1,2,3,4}] \neq \emptyset$, $X_5^2(\Delta_2, \Delta_5) \& X_5^{1,2,3,4} = x_2$, поэтому запись входного сигнала X_5^2 будет иметь вид $X_5^2 = x_4$.

в) Ветвь $S_5^3(t+1) = x_2 x_4 S_4$. Для этой ветви ранее было найдено $S_4 = 0$, поэтому $S_5^3 = 0$.

г) Ветвь $S_5^4(t+1) = \bar{x}_1 x_2 x_3 S_1$. Для этой ветви имеем:
 $[X_5^4] \cap [\Delta_1] = [\bar{x}_1]$, $X_5^4(\Delta_1) = \bar{x}_1$. Для события S_1 в п.1,в было найдено, что
 $X_1^{1,2,3} = \bar{x}_1$.

Откуда $[X_5^4(\Delta_1)] \cap [X_1^{1,2,3}] \neq \emptyset$, $X_5^4(\Delta_1) \& X_1^{1,2,3} = \bar{x}_1$, поэтому новая запись входного сигнала для события S_5^4 будет иметь вид $X_5^4 = x_2 x_3$.

б) Событие S_6 будет состоять из трех ветвей, т.к. было ранее найдено, что $S_4 = 0$.

а) Ветвь $S_6^1(t+1) = \bar{x}_4 S_2$. Для этой ветви имеем:
 $[X_6^1] \cap [\Delta_2] = [\bar{x}_4]$, $X_6^1(\Delta_2) = \bar{x}_4$.

Дальнейший анализ выполняем как в п.4,а. Так как $[X_6^1] \cap [\Delta_2] \cap [\Delta_5] = \emptyset$, то запись этой ветви остается без изменения.

б) Ветвь $S_6^2(t+1) = \bar{x}_2 S_1$. Для этой ветви имеем $[X_6^2] \cap [\Delta_1] = \emptyset$, поэтому запись этой ветви остается без изменения.

в) Ветвь $S_6^3(t+1) = x_1 S_3$. Для этой ветви имеем:
 $[X_6^3] \cap [\Delta_3] = [x_1]$, $X_6^3(\Delta_3) = x_1$.

Для события S_3 в п.1,б получено $X_3^{1,2} = x_1$. Откуда $[X_6^3(\Delta_3)] \cap [X_3^{1,2}] \neq \emptyset$, $X_6^3(\Delta_3) \& X_3^{1,2} = x_1$, поэтому новая запись входного сигнала для события S_6^3 будет иметь вид: $X_6^3 = 1$.

После произведенного анализа исходной СКУ с целью ее минимизации за счет учета распределения сдвигов ее новая запись, где введена замена $S_i = a_i$; будет иметь следующий вид (3.11):

$$\begin{aligned} a_1(t+1) &= \bar{x}_1 a_0 \vee x_2 \bar{x}_3 a_1; \\ a_2(t+1) &= a_5; \\ a_3(t+1) &= x_1 \bar{x}_2 a_0; \\ a_4(t+1) &= 0; \\ a_5(t+1) &= x_1 x_2 a_0 \vee x_4 a_2 \vee x_2 x_3 a_1; \\ a_6(t+1) &= \bar{x}_4 a_2 \vee \bar{x}_2 a_1 \vee a_3. \end{aligned} \tag{3.11}$$

Контрольные вопросы к главе 3.

1. Чем характеризуются основные этапы минимизации СКУ, представляющей алгоритм функционирования автомата?
2. Назовите условия эквивалентности событий. Чем отличаются Q-эквивалентные события от 1-эквивалентных событий?
3. Укажите признаки, характеризующие разбиение событий на классы при решении задач минимизации событий.
4. Чем характеризуются эквивалентные разбиения событий?
5. На чем основан метод таблиц пар определения эквивалентного разбиения событий?
6. Как обозначаются столбцы таблицы пар, используемой для минимизации числа событий?

7. В чем отличие таблиц пар, используемых для минимизации цифрового автомата Мили и цифрового автомата Мура?

8. Что подразумевается под понятием «учет распределения сдвигов» при определении входных сигналов автомата?

9. Как могут быть определены условия для упрощения записи событий при использовании информации, определяющей «учет распределения сдвигов»?

Глава 4. Начальные языки, используемые для представления управляющих алгоритмов параллельной обработки информации

В качестве начальных языков для формального представления параллельных алгоритмических процессов и их отдельных ветвей в данной работе рассматриваются:

- язык регулярных выражений алгебры событий (РВАС);
- язык исчисления предикатов первого порядка с ограниченными кванторами и его связь с языком РВАС;
- язык операторных схем алгоритмов с параллельными ветвями (ОСАП), базирующийся на его графическую интерпретацию – язык граф-схем алгоритмов с параллельными ветвями (ГСАП). Кроме того, рассматриваются и обычные непараллельные языки – язык граф-схем алгоритмов (ГСА) и язык логических схем алгоритмов (ЛСА), т.к. они являются основой для построения параллельных языков ОСАП.

Для каждого начального языка, кроме основных понятий и определений, рассматривается методика их использования для представления алгоритмических процессов и методика преобразования описания алгоритмических процессов на начальном языке к описанию на стандартном языке в виде системы канонических уравнений для всех событий, реализуемых в алгоритме управления преобразованием информации.

4.1. Язык регулярных выражений алгебры событий

4.1.1. Основные понятия и определения языка РВАС

Язык регулярных выражений алгебры событий (РВАС) нашел применение при построении различных управляющих и операционных устройств, например, для построения синтаксических анализаторов и устройств преобразования цифровой информации, для описания протоколов в компьютерных сетях и др.

Язык РВАС базируется на использовании алгебры событий, что позволило представлять бесконечные события, реализуемые в устройствах преобразования информации, конечными выражениями. В данном случае под преобразованием информации понимается процесс отображения множества слов в некотором конечном входном алфавите в множество слов в конечном выходном алфавите. Особенно удобен язык РВАС для решения задач распознавания цепочек слов и их принадлежности к тому или иному языку.

В связи с тем, что в соответствии с известными теоремами С.К.Клини [15] в конечном цифровом автомате представимы только регулярные

события, то язык РВАС имеет исключительно важное значение в теории и практике разработки способов описания законов функционирования цифровых автоматов на других начальных языках, когда решаются вопросы о представлении таких описаний в цифровых автоматах.

Язык РВАС и его использование для синтеза цифровых автоматов был впервые фундаментально представлен в монографии академика В.М.Глушкова «Синтез цифровых автоматов» [1]. Следуя [1], введем некоторые понятия и определения из теории языка РВАС.

Событием в любом конечном входном алфавите $[Z]$ называют произвольное множество слов в этом алфавите.

Событие $S_\alpha^{W_\beta}$, имеющее номер α , будет представленным буквой W_β в любом заданном отображении φ с алфавитами $[Z]$ и $[W]$, если для каждого входного слова $p(t)$ этого события из алфавита $[Z]$ соответствующее ему выходное слово $q(t)$ будет оканчиваться буквой W_β , входящей в алфавит $[W]$, т.е. имеем $q(t) = \varphi[p(t)]$. Такому определению события не противоречит определение события, данное в главе 2 данного пособия.

Элементарными событиями в алфавите $Z = [z_1, z_2, \dots, z_F]$ называются события, состоящие из одной буквы алфавита $[Z]$ и событие $S=e$, где e – пустое слово.

Множество всех событий, каждое из которых отмечено одним из выходных сигналов алфавита $[W]$ называют каноническим множеством событий, соответствующих отображению φ .

Для всякого автоматного отображения φ определяемое им множество событий должно удовлетворять условиям автоматности, которые формулируются таким образом:

1) События, входящие в автоматное множество событий, попарно не пересекаются и не содержат пустого слова (условие однозначности).

2) Начальные отрезки слов, принадлежащие какому-нибудь событию из автоматного множества событий, за исключением пустого слова, принадлежат каким-либо событиям из того же самого множества (условие полноты).

Под **алгеброй событий** в алфавите $[Z]$ понимается множество всех событий в этом алфавите, на котором определены две двухместные операции – дизъюнкция и умножение и одна одноместная операция, называемая итерацией.

Дизъюнкция двух событий $S_1 \vee S_2$ - это теоретико-множественное объединение этих событий.

Произведение (или конкатенация) двух событий $S_1 S_2$ - это событие, состоящее из всех слов вида $k_1 k_2$, где k_1 - любое слово из S_1 , а k_2 - любое слово события S_2 . При этом $S_1 S_2 \neq S_2 S_1$, т.е. эта операция отличается от операции логического умножения. В дальнейшем, если такая операция

встречается в каком-либо выражении одновременно с операцией логического умножения, то последнюю операцию будем обозначать символом $\&$.

Итерация события S , записываемая в виде $\{S\}$ – это событие, состоящее из пустого слова e и всевозможных слов вида $p_1 p_2 \dots p_n$, где $p_1, p_2, \dots, p_n$ – произвольные слова события S , а n – любое натуральное число $n = 1, 2, \dots$

$$\{S\} = e \vee S \vee SS \vee SSS \vee \dots$$

В алгебре событий при отсутствии в выражении обычных скобок, изменяющих обычный порядок действий, сначала должны выполняться итерации, затем конкатенация и потом дизъюнкция.

Регулярное событие – это событие, которое можно получить из элементарных событий в результате применения конечного числа раз трех основных операций алгебры событий. Необходимо иметь в виду, что не все события являются регулярными. Например, не являются регулярным событием множество всех последовательностей $z_1 z_1 \dots z_1 z_2$, таких что их длина выражается простым числом [44].

К числу не основных операций в алгебре событий относят операцию дополнения события и двуместную операцию пересечения событий.

Дополнением $\bar{S}$ события S в некотором алфавите $[Z]$ называется множество всех слов в алфавите $[Z]$, которые не вошли в событие S .

Пересечением $S_1 \& S_2$ событий S_1 и S_2 называется событие, состоящее из всех слов, входящих одновременно в оба события S_1 и S_2 .

Регулярное выражение алгебры событий – это всякое представление регулярного события через элементарные события и указанные операции алгебры событий, включая и не основные операции.

Так как одно и тоже регулярное выражение допускает много различных представлений, то в ряде случаев может возникнуть проблема эквивалентных преобразований регулярных выражений. Такие преобразования могут быть выполнены с помощью следующих тождественных соотношений [1]:

- | | |
|--|------------------------------------|
| 1. $P \vee Q = Q \vee P$ | - коммутативность дизъюнкции; |
| 2. $P \vee (Q \vee S) = (P \vee Q) \vee S$ | - ассоциативность дизъюнкции; |
| 3. $P \vee P \vee P \vee \dots \vee P = P$ | - идемпотентность дизъюнкции; |
| 4. $P (Q S) = (P Q) S$ | - ассоциативность умножения; |
| 5. $P (Q \vee S) = PQ \vee PS$ | - левая и правая дистрибутивность |
| $(P \vee Q) S = PS \vee QS$ | умножения по отношению к |
| | дизъюнкции; |
| 6. $\{\{P\}\} = \{P\}$ | - идемпотентность итерации; |
| 7. $\{P\} = e \vee P\{P\}$ | - правило разворачивания итерации; |

- | | |
|----------------------------|---|
| 8. $P \{P\} = \{P\} P$ | - закон коммутативности для итерации; |
| 9. $\{P\} \{P\} = \{P\}$ | - закон мультипликативного поглощения для итерации; |
| 10. $\{P\} \vee P = \{P\}$ | - закон дизъюнктивного поглощения для итерации; |
| 11. $eS = Se = S$ | - закон нейтральности пустого слова; |
| 12. $\{e\} = e$. | |

В этих соотношениях P, Q, S – обозначают произвольные события, а e – событие, состоящее из одного пустого слова.

Приведем некоторые дополнительные тождественные соотношения алгебры событий, полученные в [30]:

1. $\{\{P\} \{Q\}\} = \{P \vee Q\}$;
2. $\{P \vee \{Q\} \{S\}\} = \{P \vee Q \vee S\}$;
3. $\{P \vee Q\} \vee Z(P, Q) = \{P \vee Q\}$;

где $Z(P, Q)$ – регулярное выражение, построенное из P и Q с помощью трех основных операций алгебры событий;

4. $\{P \vee \{Q\}\} = \{P \vee Q\}$;
5. $\{P \vee Q \vee Z(P, Q)\} = \{P \vee Q\}$;
6. $\{P \vee Q\} \{Z(P, Q)\} = \{P \vee Q\}$.

Можно доказать следующее важное соотношение, позволяющее раскрывать итерационные скобки:

$$\text{если } P = S\{Q\}, \text{ то } P = S \vee PQ. \quad (4.1)$$

Действительно, используя тождество $\{Q\} = e \vee Q\{Q\}$ и закон нейтральности пустого слова, получим

$$P = S(e \vee Q\{Q\}) = Se \vee SQ\{Q\} = S \vee SQ\{Q\}.$$

Применяя правило $Q\{Q\} = \{Q\}Q$, будем иметь: $P = S \vee S\{Q\}Q$, где заменяя $S\{Q\} = P$, получим искомое соотношение (4.1). С другой стороны, если $e \notin Q$, то уравнение $P = QP \vee S$ имеет решение $P = \{Q\}S$.

Здесь уместно еще раз обратить внимание на фундаментальное значение для теории конечных цифровых автоматов известных теорем С.К.Клини [13, 15], в которых было впервые доказано, что класс событий, представимых в конечных цифровых автоматах, совпадает с классом регулярных событий. Это означает, что:

1) Событие, представленное в произвольном конечном автомате Мили или Мура некоторым множеством выходных сигналов или состояний, обязательно регулярно.

2) Любое регулярное событие может быть представлено в конечном автомате Мили или Мура некоторым множеством выходных сигналов или состояний.

Приведем примеры регулярных выражений для входного алфавита $[Z]=[z_1, z_2, \dots, z_F]$ некоторых характерных событий, которые могут быть полезными для описания законов функционирования автоматов [31]:

1. Всеобщее или универсальное событие – это событие, состоящее из всех возможных слов входного алфавита $[Z]$:

$$I = \{z_1 \vee z_2 \vee \dots \vee z_F\} \quad (4.2)$$

2. Событие, включающее все возможные слова, состоящие из букв $z_i, \dots, z_j$:

$$S = \{z_i \vee \dots \vee z_j\}, \text{ где } (z_i, \dots, z_j) \in [Z] \quad (4.3)$$

3. Событие, содержащее все слова, оканчивающиеся отрезком p . Например, для $p = z_i z_j$ имеем:

$$S = \{z_1 \vee z_2 \vee \dots \vee z_i \vee z_j \vee \dots \vee z_F\} z_i z_j = I z_i z_j \quad (4.4)$$

4. Событие, содержащее все слова, в которых хотя бы один раз встречается отрезок слова p в любом месте. Например, для $p = z_i z_j$ имеем:

$$S = I p I = I z_i z_j \quad (4.5)$$

5. Событие, состоящее из всех слов, имеющих начальные и конечные отрезки p_1 и p_2 соответственно. Например, для $p_1 = z_i$, $p_2 = z_j z_k$ имеем:

$$S = p_1 I p_2 = z_i I z_j z_k \quad (4.6)$$

6. Событие, содержащее все слова длины r :

$$S = S_1 S_2 \dots S_j \dots S_r, \text{ где } S_j = z_1 \vee z_2 \vee \dots \vee z_F, j = \overline{1, r} \quad (4.7)$$

7. Событие, содержащее все слова длиной, кратной r :

$$S = \{S_1 S_2 \dots S_j \dots S_r\}, \text{ где } S_j = z_1 \vee z_2 \vee \dots \vee z_F, j = \overline{1, r} \quad (4.8)$$

8. Событие, состоящее из всех слов алфавита $Z = [z_1, z_2]$, не содержащих серии из r букв z_1 и оканчивающихся буквой z_2 :

$$S = \left\{ z_2 \vee z_1 z_2 \vee z_1 z_1 z_2 \vee \dots \vee \underbrace{z_1 z_1 \dots z_1}_{(r-1) \text{ букв}} z_2 \right\} \quad (4.9)$$

Как было отмечено выше (гл.1), для многих практических реальных автоматов в качестве входного (выходного) алфавитов целесообразно использовать множество элементарных входных (выходных) сигналов. В этом случае при формализации закона функционирования автомата на языке

регулярных выражений алгебры событий под входной буквой будем понимать также частный входной сигнал, который равен конъюнкции (комбинации) тех элементарных входных сигналов, которые действуют в данный момент автоматного времени одновременно. Такую конъюнкцию будем заключать в круглые скобки. Например, для события, записанного в виде выражения:

$$S_{\alpha}^{W_{\beta}} = S_0(x_1x_2)\{(x_2\bar{x}_3x_4)\}(\bar{x}_4)$$

элементарные конъюнкции (x_1x_2) , $(x_2\bar{x}_3x_4)$, $(\bar{x}_4)$ - представляют собой частные входные сигналы, действующие на входе автомата в соответствующие такты его работы и могут быть обозначены одной буквой, например: $\chi_i = x_1x_2$; $\chi_j = x_2\bar{x}_3x_4$; $\chi_k = \bar{x}_4$.

Приведем пример составления регулярного выражения, определяющего закон функционирования цифрового автомата.

П р и м е р 4.1. Пусть необходимо описать автомат, выдающий сигнал W_k в течении одного такта всякий раз, когда происходит изменение входной буквы с z_1 на z_2 . Формулируя это условие по другому, можно сказать, что сигнал W_k должен выдаваться в ответ на любые входные последовательности, кончающиеся последовательностью z_1z_2 . Фраза «любые входные последовательности» формализуется так называемым всеобщим или универсальным событием, состоящим из всех возможных слов в алфавите $Z = [z_1, z_2]$. Тогда событие, в ответ на которое должен выдаваться сигнал W_k , на основании (4.2) и (4.4) будет описываться регулярным выражением $S^{W_k} = \{z_1 \vee z_2\}z_1z_2$.

4.1.2. Построение SKU по регулярному выражению алгебры событий и ее детерминизация

Предлагаемый способ построения SKU по регулярному выражению алгебры событий основывается на следующих предложениях [32,33]. Любое регулярное выражение алгебры событий, представляющее в цифровом автомате событие $S_{\alpha}^{W_{\beta}}$, в общем случае может быть записано в виде

$$S_{\alpha}^{W_{\beta}} = \bigvee_{i=1}^k S_{\alpha}^i, \quad (4.10)$$

где α - номер события;

k - число параллельных ветвей алгоритма, определяющих событие S_{α} (число дизъюнктивных членов регулярного выражения);

S_{α}^i - обозначение события, описывающего i - ю ветвь алгоритма.

Событие S_α^i , в свою очередь, может быть сведено к произведению элементарных одноэлементных событий. Действительно, если произвести замену итерации в описании события S_α^i там, где она есть, то получим

$$S_\alpha^i = S_0 \underbrace{z_\varepsilon \cdots z_p \cdots z_r z_\gamma}_{n \text{ букв}} \quad (4.11)$$

Исходя из интерпретации регулярного выражения алгебры событий, выражение (4.11) может быть представлено в виде следующей системы регулярных выражений:

$$\begin{aligned} S_{\alpha,0}^i &= S_{\alpha,1}^i z_\gamma, \\ S_{\alpha,1}^i &= S_{\alpha,2}^i z_r, \\ &= = = = = = \\ S_{\alpha,\beta}^i &= S_{\alpha,\beta+1}^i z_p, \\ &= = = = = = \\ S_{\alpha,n-1}^i &= S_0 z_\varepsilon \end{aligned} \quad (4.12)$$

где β - порядковый номер буквы z_p , начиная с конца выражения (4.11) (β называется рангом события и используется в качестве второго нижнего индекса);

S_0 - обозначение начального события, равного $\{e\}$; поскольку $\{e\}=e$ и $eP=Pe=P$ (закон нейтральности пустого слова), то событие S_0 может быть всегда добавлено в регулярное выражение;

$S_{\alpha,\beta+1}^i$ - обозначение события, непосредственно предшествующего событию $S_{\alpha,\beta}^i$;

n - число входных букв в выражении (4.11).

Если в выражении (4.11) была произведена замена вида

$$z_p = \{R\},$$

где R - произвольное регулярное выражение, то используя тождество (4.1), получим

$$S_{\alpha,\beta}^i = S_{\alpha,\beta+1}^i \{R\} = S_{\alpha,\beta+1}^i \vee S_{\alpha,\beta}^i R \quad (4.13)$$

Применяя рассмотренные операции к выражению (4.13), можно также свести его к выражениям типа (4.10), (4.11) и (4.12).

Исходя из изложенного, алгоритм построения СКУ по заданным регулярным выражениям исходной системы событий будет содержать следующие этапы:

1) В каждом из регулярных выражений исходной системы событий $S_1, S_2, \dots, S_\alpha, \dots, S_M$, отмеченными соответствующими выходными сигналами, выделяют параллельные ветви, описывающие события $S_\alpha (\alpha = \overline{1, M})$.

2) Для каждой параллельной ветви $S_\alpha^i (i = \overline{1, k})$ каждого из регулярных выражений исходной системы событий выполняют операцию замены итерационных скобок (внешних) там, где они имеются, одной из вспомогательных переменных, которые будут играть роль дополнительных входных букв. Например, $\{R_{\alpha, p}^i\} = z_{\alpha, p}^i$, где $R_{\alpha, p}^i$ - регулярное выражение, входящее в итерационные скобки i -ой ветви события S_α , p - порядковый номер итерационных скобок в i -ой ветви.

3) Каждое событие S_α^i преобразуют в систему рекуррентных регулярных соотношений вида (4.12).

4) Выполняют подстановку вместо буквы типа $z_{\alpha, p}^i$ соответствующие итерационные скобки $\{R_{\alpha, p}^i\}$ и осуществляют их раскрытие в соответствии с тождеством (4.1).

5) Если содержимое итерационных скобок какого-либо частного события из полученной системы соотношений вида (4.12) включает более одной буквы, в т.ч. и итерационные скобки (внутренние), то для этого события повторяют все операции, предусмотренные п.п. 1-5 и т.д. до тех пор, пока не будут раскрыты все итерационные скобки в регулярных выражениях всех событий, входящих в исходную систему.

Примечание: Для уменьшения числа итерационных скобок целесообразно использовать дополнительные тождественные соотношения алгебры событий.

6) Выполняют для каждого события S_α исходной системы подстановку вместо обозначения событий S_α^i соответствующих им выражений.

Аналогичную подстановку делают и в том случае, если в процессе преобразований описания некоторых частных событий будут содержать обозначения непосредственно предшествующих событий без входных букв. В этом случае необходимо сделать подстановку вместо обозначения таких событий соответствующих им выражений.

7) Во всех описаниях, полученных частных событий, заменяют знаки конкатенации на знаки конъюнкции и, вводя дискретное время, как принято в (2.3), получают искомую СКУ, в которой обозначения событий исходной системы отмечаются соответствующими выходными сигналами.

Примечание: Принятый способ нумерации событий и букв удобен при выполнении всех приведенных выкладок и не является обязательным. При небольшом числе событий может оказаться более удобным нумеровать события с использованием одного индекса.

Пример 4.2. Рассмотрим без дополнительных пояснений пример построения СКУ по следующему регулярному выражению:

$$S_{\alpha}^W = S_0 z_2 \{z_1 \vee z_2\} z_2 z_1 \vee S_0 \{z_1 z_2\} = S_{\alpha,1}^W \vee S_{\alpha,2}^W,$$

$$S_{\alpha,1}^W = S_0 z_2 \{z_1 \vee z_2\} z_2 z_1 = S_3 z_1,$$

$$S_3 = S_0 z_2 \{z_1 \vee z_2\} z_2 = S_4 z_2,$$

$$S_4 = S_0 z_2 \{z_1 \vee z_2\} = S_0 z_2 \vee S_4 (z_1 \vee z_2),$$

$$S_{\alpha,2}^W = S_0 \{z_1 z_2\} z_2 = S_5 z_2,$$

$$S_5 = S_0 \{z_1 z_2\} = S_0 \vee S_5 z_1 z_2 = S_0 \vee S_6 z_2, S_0 \Rightarrow S_5,$$

$$S_6 = S_5 z_1,$$

где $\Rightarrow$ - знак выводимости.

Окончательно СКУ будет иметь вид:

$$\begin{aligned} S_{\alpha}^W(t+1) &= S_3(t) \& z_1(t) \vee S_5(t) \& z_2(t), \\ S_3(t+1) &= S_4(t) \& z_2(t), \\ S_4(t+1) &= S_0(t) \& z_2(t) \vee S_4(t) \& [z_1(t) \vee z_2(t)], \\ S_5(t+1) &= S_0(t) \vee S_6(t) \& z_2(t), \\ S_6(t+1) &= S_5(t) \& z_1(t). \end{aligned} \tag{4.14}$$

В общем случае СКУ, полученная по регулярным выражениям алгебры событий недетерминирована. Это относится не только к СКУ, полученной по системе регулярных выражений с общими начальными событиями, но и к СКУ, полученной по одному регулярному выражению, включающему только одну ветвь с итерационной скобкой. В связи с этим для таких СКУ в целях дальнейшего синтеза автомата может потребоваться их детерминизация.

Если в качестве входных сигналов в исходных регулярных выражениях алгебры событий используются частные входные сигналы из структурного алфавита, то детерминизацию СКУ необходимо проводить в соответствии с алгоритмом, предложенным в гл.2.

Если в качестве входных сигналов в исходных регулярных выражениях алгебры событий используются, как обычно принято, полные входные сигналы из абстрактного алфавита $[Z]$, то алгоритм детерминизации упрощается, т.к. на входе автомата в любой момент автоматного времени может появиться только одна из букв входного алфавита $[Z]$, но не их сочетание.

В связи с отмеченными замечаниями при построении прямой таблицы переходов детерминированного автомата для каждой строки таблицы в столбце 3 выписываются все входные сигналы из абстрактного алфавита $[Z]$, а для обычной таблицы переходов детерминированного автомата каждая

строка должна отмечаться своим абстрактным входным сигналом, а число строк таблицы определится числом всех входных сигналов.

Для упрощения процесса детерминизации при построении таблиц переходов автомата также целесообразно представить исходную СКУ в виде вспомогательной прямой таблицы переходов для событий.

При этом следует иметь в виду, что при построении по СКУ вспомогательной прямой таблицы переходов (НД ПТП) и таблицы переходов (ТП) детерминированного автомата первый столбец в первой строке таблиц должен обозначаться следующим образом:

а) если S_0 входит в качестве предшествующего события в какое-либо из уравнений СКУ и из S_0 не выводимо ни одно из событий СКУ, то $a_0 = S_0$;

б) если S_0 входит в качестве предшествующего события в какое-либо из уравнений СКУ и из S_0 выводимы события $S_i, \dots, S_j$, то $a_0 = S_0 \& S_i \& \dots \& S_j$;

в) если S_0 не входит в качестве предшествующего события ни в одно из уравнений СКУ и из S_0 выводимы события $S_i, \dots, S_j$, то $a_0 = S_i \& \dots \& S_j$.

Пример 4.3. Выполним детерминизацию СКУ, полученную в предыдущем примере, путем построения обычной таблицы переходов детерминированного автомата.

По СКУ (4.14) строим вспомогательную прямую таблицу переходов для частных событий (табл.4.1), в которой в столбец 3 каждой строки записываются те входные сигналы, которые имеют место в исходной СКУ на рассматриваемом переходе.

В соответствии с общими положениями алгоритма детерминизации, а также замечаниями отмеченными выше, и на основании табл.4.1, таблица переходов детерминированного автомата Мура будет иметь вид (табл.4.2):

Таблица 4.1

1	2	3	4
1	$S_0 \& S_5$	z_1 z_2	S_6 $S_4 \& S_\alpha$
2	S_6	z_2	S_5
3	S_4	z_1 z_2	S_4 $S_3 \& S_4$
4	S_5	z_1 z_2	S_6 S_α
5	S_3	z_1	S_α
6	S_α	z_1 z_2	$*$ $*$

Таблица 4.2

w	e	e	w	e	e	e	w	e
a z	$S_0 \& S_5$	S_6	$S_4 \& S_\alpha$	S_5	S_4	$S_4 \& S_3$	S_α	$*$
z_1	S_6	$*$	S_4	S_6	S_4	$S_4 \& S_\alpha$	$*$	$*$
z_2	$S_4 \& S_\alpha$	S_5	$S_4 \& S_3$	S_α	$S_4 \& S_3$	$S_4 \& S_3$	$*$	$*$

П р и м е р 4.4. Предположим, что необходимо описать множество всех последовательностей из нулей и единиц, которые имеют три подряд стоящих единицы, но не кончающиеся на 01 и не состоящие лишь из единиц. По полученному описанию построить отмеченную таблицу переходов (ТП) цифрового автомата Мура.

В данном примере рассматривается представление двоичных слов в виде событий на языке РВАС с использованием дополнительных операций алгебры событий с построением SKU и ТП ЦА Мура, реализующего представляющее событие.

Решение:

1) закодируем исходные двоичные цифры следующим образом:
 $0 = x_0$, $1 = x_1$;

2) множество всех последовательностей, которые имеют подряд три единицы, можно записать так:

$$\{x_0 \vee x_1\}x_1x_1x_1\{x_0 \vee x_1\};$$

3) множество всех последовательностей, кончающееся на 01, запишется таким образом:

$$\{x_0 \vee x_1\}x_0x_1;$$

4) множество всех последовательностей, состоящее лишь из единиц, имеет вид:

$$x_1\{x_1\};$$

5) таким образом все искомое множество двоичных последовательностей можно представить в виде следующего события:

$$S_\alpha^y = \{x_0 \vee x_1\}x_1x_1x_1\{x_0 \vee x_1\} \& (\overline{\{x_0 \vee x_1\}x_0x_1 \vee x_1\{x_1\}}) = S_1 \& \bar{S}_2;$$

6) на основе рассмотренного алгоритма преобразования РВАС описание события S_α^y преобразуется в следующую СКУ:

$$S_1(t+1) = S_3(t) \& x_1(t) \vee S_1(t) \& [x_0(t) \vee x_1(t)],$$

$$S_3(t+1) = S_4(t) \& x_1(t),$$

$$S_4(t+1) = S_5(t) \& x_1(t),$$

$$S_5(t+1) = S_0(t) \vee S_5(t) \& [x_0(t) \vee x_1(t)],$$

$$S_2(t+1) = S_0(t) \& x_1(t) \vee S_6(t) \& x_1(t) \vee S_7(t) \& x_1(t),$$

$$S_6(t+1) = S_5(t) \& x_0(t),$$

$$S_7(t+1) = S_0(t) \& x_1(t) \vee S_7(t) \& x_1(t);$$

7) для построения отмеченной таблицы переходов ЦА Мура по СКУ строится вспомогательная НД ПТП ЦА Мура, в которой первый столбец отмечается совокупностью событий S_0 и S_5 , т.к. событие S_5 выводимо из события S_0 , т.к. имеем $S_0 \Rightarrow S_5$.

Таблица 4.3

1	2	3	4
1	$S_0 \& S_5$	x_0 x_1	$S_5 \& S_6$ $S_5 \& S_7 \& S_2 \& S_4$
2	S_5	x_0 x_1	$S_5 \& S_6$ $S_5 \& S_4$
3	S_6	x_1	S_2
4	S_7	x_1	$S_2 \& S_7$
5	S_4	x_1	S_3
6	S_3	x_1	S_1
7	S_1	x_0 x_1	S_1 S_1

8) искомая отмеченная детерминированная ТП ЦА Мура, построенная на основе НД ПТП, будет иметь следующий вид:

Таблица 4.4

	e	e	e	e
a x	$S_0 \& S_5$	$S_5 \& S_6$	$S_5 \& S_7 \& S_4 \& S_2$	$S_5 \& S_4 \& S_2$
x_0	$S_5 \& S_6$	$S_5 \& S_6$	$S_5 \& S_6$	$S_5 \& S_6$
x_1	$S_5 \& S_7 \& S_4 \& S_2$	$S_5 \& S_4 \& S_2$	$S_5 \& S_4 \& S_7 \& S_2 \& S_3$	$S_5 \& S_4 \& S_3$

e	e	e
$S_5 \& S_4 \& S_7 \& S_2 \& S_3$	$S_5 \& S_4 \& S_3$	$S_5 \& S_4 \& S_3 \& S_2 \& S_7 \& S_1$
$S_5 \& S_6$	$S_5 \& S_6$	$S_5 \& S_6 \& S_1$
$S_5 \& S_4 \& S_3 \& S_2 \& S_7 \& S_1$	$S_5 \& S_4 \& S_3 \& S_1$	$S_5 \& S_4 \& S_3 \& S_1 \& S_2 \& S_7$

y	y	e
$S_5 \& S_4 \& S_3 \& S_1$	$S_5 \& S_6 \& S_1$	$S_5 \& S_4 \& S_2 \& S_1$
$S_5 \& S_6 \& S_1$	$S_5 \& S_6 \& S_1$	$S_5 \& S_6 \& S_1$
$S_5 \& S_4 \& S_2$	$S_5 \& S_4 \& S_2 \& S_1$	$S_5 \& S_4 \& S_3 \& S_1$

Цифровой автомат, построенный в соответствии с полученной ТП ЦА Мура, при подаче на вход последовательности двоичных цифр, представленных событием S_a^y , будет выдавать выходной сигнал $y=1$; для

всех других последовательностей, не соответствующих событию S_α , ЦА будет выдавать пустой сигнал.

4.2. Язык исчисления предикатов

Как следует из [29,34] язык исчисления предикатов имеет ряд преимуществ по сравнению с другими начальными языками, в том числе и с языком РВАС.

Во-первых, язык исчисления предикатов естественно описывает словесные алгоритмы функционирования устройств цифровой вычислительной техники.

Во-вторых, язык исчисления предикатов значительно шире, т.е. выразительнее языка РВАС.

В-третьих, алгоритмические проблемы проектируемых цифровых устройств могут формулироваться и решаться на том же языке, на котором решается проблема синтеза абстрактного автомата.

В связи с тем, что любой новый предлагаемый для использования начальный язык требует доказательств представимости в цифровом автомате событий, описываемых на этом языке, то в начале этого раздела будет показано, что все регулярные выражения алгебры событий, в том числе элементарное одноэлементное событие и событие с итерацией сложного регулярного выражения, могут быть описаны вполне определенным частным видом формул исчисления предикатов, не имеющих кванторов по функциональной переменной, т.е. частным видом формул исчисления предикатов первого порядка с ограниченными кванторами [34].

Полученный частный вид формул исчисления предикатов первого порядка, не имеющих кванторов по функциональной переменной, будет принят в качестве основы для описания событий, реализуемых в алгоритме управления преобразованием информации [22].

4.2.1. Описание регулярных выражений алгебры событий рекурсивными предикатами

При описании событий в алфавите $[Z]=[z_1, z_2, \dots, z_F]$ будем учитывать два условия:

1) Всякий абстрактный автомат получает входные сигналы во все моменты времени $t=1,2,3,\dots$. Если же реальный автомат в какие-то моменты времени не получает сигналов из алфавита $[Z]$, то для описания такого случая будем использовать пустой входной сигнал. В исчислении предикатов пустой входной сигнал (пустое слово) $e(\tau)$ может быть описан предикатной формулой

$$e(\tau) = \overline{z_1(\tau) \vee z_2(\tau) \vee \dots \vee z_F(\tau)}. \quad (4.15)$$

2) На вход абстрактного автомата в любой момент времени может поступать только один сигнал z_i . Это условие в исчислении предикатов может быть описано предикатной формулой $\chi_i(\tau)$, определяемой выражением:

$$\chi_i(\tau) = z_i(\tau) \& e_{z_i}(\tau), \quad (4.16)$$

где $e_{z_i}(\tau)$ есть результат подстановки нуля вместо предиката $z_i(\tau)$ в формулу для пустой буквы $e(\tau)$. Используя обозначение подстановки, принятое в [35], получим

$$e_{z_i}(\tau) = \int_0^\vee Z_i(\tau) e(\tau). \quad (4.17)$$

Описание события S_0 , заданное в алгебре событий пустым словом, формулами исчисления предикатов.

Событие S_0 на основании закона нейтральности пустого слова в алгебре событий можно записать так:

$$S_0 = \{e\}. \quad (4.18)$$

Выполнив описание выражения (4.18) в исчислении предикатов, получим

$$S_0(t) = (\forall \tau) [\tau \leq t \rightarrow e(\tau)]. \quad (4.19)$$

Выражение (4.19) определяет предикат $S_0(t)$ для всех значений вещественной переменной τ из области натуральных чисел, удовлетворяющих неравенству $0 < \tau \leq t$.

Вводя сокращенное обозначение ограниченных кванторов [34], окончательно получим

$$S_0(t) = (\forall_{\tau \leq t}) e(\tau). \quad (4.20)$$

Описание элементарного одноэлементного события $S = z_i$.

Учитывая закон нейтральности пустого слова, исходное выражение можно переписать так:

$$S = \{e\}_{z_i} \{e\}. \quad (4.21)$$

Выражение (4.21) в исчислении предикатов, с учетом 2-го условия, может быть описано следующим образом:

$$S(t) = (\exists \tau) [\tau \leq t \& z_i(t) \& e_{z_i}(\tau) \& (\forall \tau_1) [\tau < \tau_1 \leq t \rightarrow e(\tau_1)] \& (\forall \delta) [\delta < \tau \rightarrow e(\delta)]]]. \quad (4.22)$$

Легко видеть, что полученная предикатная формула является утверждением того, что для $\tau \leq t$ существует единственное τ , такое, что $z_i(\tau)$ [29] и утверждением того, что для всех τ , удовлетворяющих неравенству $\tau \leq t$, $e_{z_i}(\tau)$.

Используя сокращенное обозначение ограниченных кванторов и выражение (4.16), перепишем (4.22) так:

$$S(t) = (\exists \tau)_{\tau \leq t} \chi_i(\tau) \& (\forall \tau_1)_{\tau < \tau_1 \leq t} e(\tau_1) \& e(\delta)_{\delta < \tau} \quad (4.23)$$

или используя дополнительно (4.20) и учитывая, что неравенство $\delta < \tau$ эквивалентно неравенству $\delta \leq \tau - 1$, получим:

$$S(t) = (\exists \tau)_{\tau \leq t} \chi_i(\tau) \& (\forall \tau_1)_{\tau < \tau_1 < t} e(\tau_1) \& S_0(\tau - 1) \quad (4.24)$$

Из выражения (4.24) следует, что описание события S состоит из 3-х частей:

1. описание события в момент его наступления;
2. описание условия, при котором событие S сохраняется;
3. описание события, непосредственно предшествующего событию S .

Описание события S , заданного дизъюнкцией элементарных одноэлементных событий $S = z_i \vee z_j$.

На основании закона нейтральности пустого слова перепишем исходное выражение в следующей эквивалентной форме:

$$S = \{e\}z_i\{e\} \vee \{e\}z_j\{e\} \quad (4.25)$$

Поступив также как и в предыдущем случае, получим:

$$\begin{aligned} S(t) = & (\exists \tau)_{\tau \leq t} \chi_i(\tau) \& (\forall \tau_1)_{\tau < \tau_1 \leq t} e(\tau_1) \& S_0(\tau - 1) \vee \\ & \vee (\exists \tau)_{\tau \leq t} \chi_j(\tau) \& (\forall \tau_1)_{\tau < \tau_1 \leq t} e(\tau_1) \& S_0(\tau - 1) \end{aligned} \quad (4.26)$$

Выполнив в полученном выражении переименование связанных переменных [29, 39] и производя необходимые преобразования, можно выражению 4.26 придать следующий вид:

$$S(t) = (\exists \tau)_{\tau \leq t} [\chi_i(\tau) \vee \chi_j(\tau)] \& (\forall \tau_1)_{\tau < \tau_1 \leq t} e(\tau_1) \& S_0(\tau - 1) \quad (4.27)$$

Описание события S , заданного регулярным выражением алгебры событий $z_i z_j$, т.е.

$$S = z_i z_j,$$

которое на основании нейтральности пустого слова в алгебре событий можно переписать так:

$$S = \{e\}z_i\{e\}z_j\{e\}. \quad (4.28)$$

Выполнив описание выражения (4.28) в исчислении предикатов, получим:

$$S(t) = (\exists \tau)_{\tau \leq t} [\chi_j(\tau) \& (\forall \tau_1)_{\tau < \tau_1 \leq t} e(\tau_1) \& (\exists \delta)_{\delta < \tau} [\chi_i(\delta) \& (\forall \delta_1)_{\delta < \delta_1 < \tau} e(\delta_1) \& (\forall \sigma)_{\sigma < \delta} e(\sigma)]]]. \quad (4.29)$$

Если не пользоваться скобками, определяющими область действия кванторов существования, считая при этом, что действие квантора распространяется на всю конъюнкцию предикатов, расположенных правее квантора, и учесть (4.20), то выражение (4.29) примет вид:

$$S(t) = (\exists \tau)_{\tau \leq t} \chi_i(\tau) \& (\forall \tau_1)_{\tau < \tau_1 \leq t} e(\tau_1) \& (\exists \delta)_{\delta < \tau} \chi_i(\delta) \& (\forall \delta_1)_{\delta < \delta_1 < \tau} e(\delta_1) \& S_0(\delta - 1). \quad (4.30)$$

Обозначив на основании (4.24) связную группу предикатов

$$(\exists \delta)_{\delta < \tau} \chi_i(\delta) \& (\forall \delta_1)_{\delta < \delta_1 < \tau} e(\delta_1) \& S_0(\delta - 1)$$

через $S_1(\tau - 1)$, получим следующую форму записи выражения (4.30):

$$S(t) = (\exists \tau)_{\tau \leq t} \chi_i(\tau) \& (\forall \tau_1)_{\tau < \tau_1 \leq t} e(\tau_1) \& S_1(\tau - 1). \quad (4.31)$$

Из (4.31) следует, что описание выражения $S = z_i z_j$, также как и описание выражения $S = z_i$, состоит из аналогичных трех частей.

Описание события S , заданное регулярным выражением алгебры событий $\{z_i\}$, т.е.

$$S = \{z_i\}.$$

На основании правила развертывания итерации, закона коммутативности для итерации и закона нейтральности пустого слова исходное выражение в алгебре событий может быть переписано в следующей эквивалентной форме:

$$S = \{e\} \vee \{z_i\} z_i \{e\}. \quad (4.32)$$

Учитывая исходную формулу описываемого выражения, можно (4.32) переписать и так:

$$S = \{e\} \vee S z_i \{e\}. \quad (4.33)$$

Выполнив описание выражения (4.33) формулами исчисления предикатов, получим:

$$S(t) = S_0(t) \vee (\exists \tau)_{\tau \leq t} \chi_i(\tau) \& (\forall \tau_1)_{\tau < \tau_1 \leq t} e(\tau_1) \& S(\tau - 1). \quad (4.34)$$

Описание дополнения $\bar{S}$ события S .

Дополнение $\bar{S}$ события S в исчислении предикатов описывается обычным образом на основании правил исчисления предикатов. Так, если

$$S(t) = (\exists \tau)_{\tau \leq t} \chi_j(\tau) \& (\forall \tau_1)_{\tau < \tau_1 \leq t} e(\tau_1) \& S_1(\tau - 1), \text{ то}$$

$$\bar{S}(t) = (\forall \tau)_{\tau \leq t} [\bar{\chi}_j(\tau) \vee (\exists \tau_1)_{\tau < \tau_1 \leq t} \bar{e}(\tau_1) \vee \bar{S}_1(\tau - 1)]. \quad (4.35)$$

Описание регулярных выражений алгебры событий произвольной формы формулами исчисления предикатов.

В разделе 4.1.2 было показано, что любое РВАС, заданное произведением элементарных одноэлементных событий может быть представлено в виде системы регулярных выражений (4.12), в которой любое входящее в систему регулярное выражение, представляется произведением двух переменных:

$$S_{\alpha, \beta}^i = S_{\alpha, \beta+1}^i z_p, \quad (4.36)$$

где $S_{\alpha, \beta+1}^i$ - сокращенное обозначение события, являющегося непосредственно предшествующим событию $S_{\alpha, \beta}^i$, а z_p - буква входного алфавита $[Z]$.

Выражения типа (4.36) на основании предыдущих результатов могут быть представлены формулами исчисления предикатов в следующем виде:

$$S_{\alpha, \beta}^i(t) = (\exists \tau)_{\tau \leq t} \chi_p(\tau) \& (\forall \tau_1)_{\tau < \tau_1 \leq t} e(\tau_1) \& S_{\alpha, \beta+1}^i(\tau - 1). \quad (4.37)$$

На основании полученных в разделе 4.2 результатов, можно сформулировать следующее предложение:

Любое регулярное выражение алгебры событий, заданное произведением элементарных одноэлементных событий $z_1, z_2, \dots, z_F$ может быть описано в алфавите $[Z]$ системой одноместных формул исчисления предикатов первого порядка с ограниченными кванторами вида (4.37), общее число формул системы определяется числом букв входного алфавита $[Z]$, входящих в исходное произведение [22].

В том случае, когда в исходном выражении для произведения элементарных одноэлементных событий (4.11) была выполнена замена, например, вида $z_p = \{R\}$, то используя правила разворачивания итерации и на основании (4.34) получим:

$$S_{\alpha, \beta}^i(t) = S_{\alpha, \beta+1}^i(t) \vee (\exists \tau)_{\tau \leq t} R(\tau) \& (\forall \tau_1)_{\tau < \tau_1 \leq t} \& S_{\alpha, \beta}^i(\tau - 1), \quad (4.38)$$

где R – любое регулярное выражение алгебры событий.

В том случае, когда выражение для R представлено не в виде одной буквы, то используют операции подстановок, позволяющие получить описания событий, заданных произвольными регулярными выражениями алгебры событий. Например, пусть

$$R = z_i \vee z_k z_r. \quad (4.39)$$

Требуется построить предикатную формулу $S_{\alpha,\beta}^i(t)$, получающуюся из (4.38) путем подстановки вместо группы предикатов с переменными в называющей форме, описывающих левую часть формулы (4.39), группу предикатов в называющей форме, описывающих правую часть формулы (4.39). Тогда событие $S_{\alpha,\beta}^i(t)$ запишется так:

$$S_{\alpha,\beta}^i(t) = S_{\alpha,\beta+1}^i(t) \vee (\exists \tau)[\tau \leq t \& \chi_i(\tau) \& (\forall \tau_1)_{\tau < \tau_1 \leq t} e(\tau_1) \vee \\ \vee (\exists \delta)_{\delta \leq t} \chi_r(\delta) \& (\forall \delta_1)_{\delta < \delta_1 \leq t} e(\delta_1) \& \tau < \delta \& \chi_k(\tau) \& (\forall \tau_1)_{\tau < \tau_1 \leq \delta} e(\tau_1)] \& S_{\alpha,\beta}^i(\tau - 1)].$$

Выполнив необходимые преобразования, в том числе частичное переименование связанной переменной, окончательно получим:

$$S_{\alpha,\beta}^i(t) = S_{\alpha,\beta+1}^i(t) \vee (\exists \tau)_{\tau \leq t} \chi_i(\tau) \& (\forall \tau_1)_{\tau < \tau_1 \leq t} e(\tau_1) \& S_{\alpha,\beta}^i(\tau - 1) \vee \\ \vee (\exists \tau)_{\delta \leq t} \chi_r(\tau) \& (\forall \tau_1)_{\delta < \tau_1 \leq t} e(\tau_1) \& (\exists \tau)_{\sigma < \tau} \chi_k(\sigma) \& (\forall \sigma_1)_{\sigma < \sigma_1 \leq \tau} e(\sigma_1) \& S_{\alpha,\beta}^i(\sigma - 1). \quad (4.40)$$

Обозначив на основании (4.24) связную группу предикатов

$$(\exists \sigma)_{\sigma < \tau} \chi_k(\sigma) \& (\forall \sigma_1)_{\sigma < \sigma_1 \leq \tau} e(\sigma_1) \& S_{\alpha,\beta}^i(\sigma - 1) \quad (4.41)$$

через $S_1(\tau - 1)$, получим следующую форму записи выражения (4.40):

$$S_{\alpha,\beta}^i(t) = S_{\alpha,\beta+1}^i(t) \vee (\exists \tau)_{\tau \leq t} \chi_i(\tau) \& (\forall \tau_1)_{\tau < \tau_1 \leq t} e(\tau_1) \& S_{\alpha,\beta}^i(\tau - 1) \vee \\ \vee (\exists \tau)_{\tau \leq t} \chi_r(\tau) \& (\forall \tau_1)_{\tau < \tau_1 \leq t} e(\tau_1) \& S_1(\tau - 1). \quad (4.42)$$

Откуда следует, что событие $S_{\alpha,\beta}^i$ будет представлено двумя предикатными формулами, одна из которых (4.42), а другая, определяемая связной группой предикатов (4.41), после переименования связанной переменной будет иметь вид:

$$S_1(t) = (\exists \tau)_{\tau \leq t} \chi_k(\tau) \& (\forall \tau_1)_{\tau < \tau_1 \leq t} e(\tau_1) \& S_{\alpha,\beta}^i(\tau - 1) \quad (4.43)$$

Таким образом, исходя из полученных результатов, справедливо следующее утверждение:

Любое регулярное выражение алгебры событий S_α , заданное в алфавите $[Z]$, может быть рекурсивно описано в этом же алфавите системой (A_α) предикатных формул, полученных из выражений типа (4.37) при помощи необходимого количества подстановок. При этом любые предикатные формулы системы имеют одинаковую структуру и представляются в виде выражений, состоящих из трех частей, как это было показано, например, в (4.37).

Из сформулированного утверждения и изложенного выше можно вывести следующее следствие:

Если в исчислении предикатов первого порядка некоторый предикат $P(t)$ можно эквивалентными преобразованиями в алфавите $[Z]$ привести к виду (A_α) , то предикат $P(t)$ в этом же алфавите $[Z]$ описывает вполне определенное регулярное выражение алгебры событий и на основании первой теоремы С.К.Клини [13] представим в конечном цифровом автомате.

4.2.2. Формализация словесных алгоритмов, определяющих реализуемые в цифровом автомате события, на основе использования языка исчисления предикатов

Для удобства и упорядочения формы записи исходных предикатных формул, формализующих словесные алгоритмы, введем понятие номера и ранга рассматриваемого события и номера ветви алгоритма, определяющей это событие [12].

Номером события S будем называть натуральное число $0, 1, 2, \dots, \alpha, \dots$, приписываемое к обозначению события в качестве первого нижнего индекса.

Рангом события S_α будем называть натуральным число $0, 1, 2, \dots, \beta, \dots$, приписываемое к обозначению события в качестве второго нижнего индекса.

Событием нулевого (старшего) ранга относительно описываемого события S_α будем называть самое это событие S_α , а также события, непосредственно определяющие событие S_α как произвольную двузначную функцию [36].

Событием первого ранга относительно события S_α будем называть каждое событие, непосредственно предшествующее событию S_α .

Событием $(\beta+1)$ -го ранга относительно события S_α будем называть каждое событие, непосредственно предшествующее любому из событий β -го ранга относительно события S_α .

Если событие $S_{\alpha,\beta}$ непосредственно определяется логической суммой других событий, то для отличия таких дизъюнктивных членов введем первый верхний индекс, который будем называть номером ветви алгоритма, определяющего событие S_α . При этом различные ветви алгоритма должны иметь различные номера, т.е. обозначения событий, порожденных различным образом из исходного, при одинаковых нижних индексах должны иметь различные первые верхние индексы.

Для формального описания на языке исчисления предикатов словесных алгоритмов, определяющих реализуемые в цифровом автомате события, можно воспользоваться результатами, получаемыми в разделе 4.2.1 относительно представления любого регулярного выражения алгебры

событий в виде системы рекурсивно определяемых предикатных формул, имеющих одинаковую структуру вида (4.37).

Такая система в расширенном исчислении предикатов может быть преобразована в более удобную для формализации словесных алгоритмов системы $(A)_p$. Под расширенным исчислением предикатов в данном случае понимается исчисление предикатов с добавлением к нему двух аксиом, формализующих закон нейтральности пустого слова алгебры событий [9, 22].

Аксиома 1.

$$F(t) \approx (\exists_{\tau \leq t} F(\tau)) \& (\forall_{\tau_1 \leq t} e(\tau_1)). \quad (4.44)$$

Аксиома 2.

$$(\forall_{\tau < \tau_1 \leq t} F(\tau_1)) \approx (\forall_{\tau < \tau_1 \leq t} e(\tau_1)) \vee \tau < t \& F(t) \& (\forall_{\tau < \tau_1 \leq t} F(\tau_1)), \quad (4.45)$$

где $\approx$ - знак эквивалентности в расширенном исчислении предикатов,

$F(t)$ – предикатная формула.

Для построения новой системы $(A)_p$ введем понятие рода событий и ветвей алгоритма.

Все события β -го ранга $S_{\alpha,\beta}^{[i,j]}$, непосредственно определяющие некоторое событие $S_{\alpha,\beta}^i$, разделим на две принципиально разные группы событий. К одной группе отнесем все события, которые определяют зарождение, т.е. первичное появление события $S_{\alpha,\beta}^i$. Будем называть такие события событиями 1-го рода.

К другой группе отнесем все события $S_{\alpha,\beta}^{[i,j]}$, которые определяют возобновление события $S_{\alpha,\beta}^i$, т.е. повторное его появление. Будем называть такие события событиями 2-го рода.

Аналогично ветви алгоритма, описывающие события 1-го (2-го) рода, будем называть ветвями 1-го (2-го) рода.

В связи с изложенным выражение вида (4.37), описывающее в исчислении предикатов событие $S_{\alpha,\beta}^i$, может быть записано так:

$$S_{\alpha,\beta}^i(t) = \bigvee_{j=1}^{j=k_1} S_{\alpha,\beta}^{[i,j]}(t) \vee \bigvee_{j=1}^{j=k_2} S_{\alpha,\beta}^{[i,k_1+j]}(t), \quad (4.46)$$

где k_1 - число ветвей 1-го рода;

k_2 - число ветвей 2-го рода.

Выполнив подстановку вместо предикатов выражения (4.46) обозначаемые ими предикатные формулы, получим:

$$S_{\alpha,\beta}^i(t) = \bigvee_{j=1}^{j=k_1} (\exists \tau) \chi_{\alpha,\beta}^{[i,j]}(\tau) \& (\forall \tau_1)_{\tau < \tau_1 \leq t} e(\tau_1) \& S_{\alpha,\beta+1}^{[i,j]}(\tau - 1) \vee \\ \vee \bigvee_{j=1}^{j=k_2} (\exists \tau) \chi_{\alpha,\beta}^{[i,k_1+j]}(\tau) \& (\forall \tau_1)_{\tau < \tau_1 \leq t} e(\tau_1) \& S_{\alpha,\beta}^i(\tau - 1). \quad (4.47)$$

где $\chi_{\alpha,\beta}(\tau)$ - предикат формализующий частный входной сигнал.

На основании аксиом, формализующих закон нейтральности пустого слова (4.44), (4.45), нетрудно доказать, что выражение (4.47) в расширенном исчислении предикатов эквивалентно выражению:

$$S_{\alpha,\beta}^i(t) = \bigvee_{j=1}^{j=k_1} (\exists \tau) \chi_{\alpha,\beta}^{[i,j]}(\tau) \& (\forall \tau_1)_{\tau < \tau_1 \leq t} q_{\alpha,\beta}^i(\tau_1) \& S_{\alpha,\beta+1}^{[i,j]}(\tau - 1), \quad (4.48)$$

$$\text{где } q_{\alpha,\beta}^i(\tau_1) = \bigvee_{r=1}^{r=k_2} \chi_{\alpha,\beta}^{[i,k_1+r]}(\tau_1).$$

На основании изложенного при формализации словесных алгоритмов функционирования цифровых автоматов можно использовать систему $(A)_p$, построенную на основе предикатных выражений вида (4.48), каждое из которых представляет событие, реализуемое в автомате. При этом описания таких событий имеют одинаковую структуру, состоящую из упомянутых ранее трех частей (4.24): описание наступления события и его сохранения, и описание непосредственно предшествующего события.

4.2.3. Построение системы канонических уравнений по описанию алгоритма обработки информации, представленным на языке исчисления предикатов

Система канонических уравнений, являющаяся основой для построения структуры цифровых устройств обработки информации, функционирование которых представлено на начальном языке исчисления предикатов первого порядка, может быть получена из системы $(A)_p$ как способом, изложенным в [34], так и на основании аксиом (4.44) и (4.45).

На основании аксиом (4.44) и (4.45) выражение вида (4.48) преобразуется в каноническое уравнение по следующему алгоритму [37].

1. Применить один раз аксиому 2 (4.45) к предикату $(\forall \tau_1)_{\tau < \tau_1 \leq t} q_{\alpha,\beta}^i(\tau_1)$ выражения (4.48). В результате чего получим:

$$S_{\alpha,\beta}^i(t) = \bigvee_{j=1}^{j=k_1} (\exists \tau) \chi_{\alpha,\beta}^{[i,j]}(\tau) \& (\forall \tau_1)_{\tau < \tau_1 \leq t} e(\tau_1) \& S_{\alpha,\beta+1}^{[i,j]}(\tau - 1) \vee \\ \vee \bigvee_{j=1}^{j=k_1} (\exists \tau) \chi_{\alpha,\beta}^{[i,j]}(\tau) \& \tau < t \& q_{\alpha,\beta}^i(t) \& (\forall \tau_1)_{\tau < \tau_1 \leq t} q_{\alpha,\beta}^i(\tau_1) \& S_{\alpha,\beta+1}^i(\tau - 1) \quad (4.49)$$

Перейти к выполнению пункта 2.

2. Учитывая, что $q_{\alpha,\beta}^i(t)$ не содержит свободно анонимной переменной τ и формулу

$$\tau \leq t \ \& \ \tau < t \equiv \tau < t,$$

записать выражение (4.49) в следующей эквивалентной форме:

$$\begin{aligned} S_{\alpha,\beta}^i(t) = & \bigvee_{j=1}^{j=k_1} (\exists \tau) \chi_{\alpha,\beta}^{[i,j]}(\tau) \ \& \ (\forall \tau_1) e(\tau_1) \ \& \ S_{\alpha,\beta+1}^{[i,j]}(\tau-1) \vee q_{\alpha,\beta}^i(t) \ \& \\ & \& \ [\bigvee_{j=1}^{j=k_1} (\exists \tau) \chi_{\alpha,\beta}^{[i,j]}(\tau) \ \& \ (\forall \tau_1) q_{\alpha,\beta}^i(\tau_1) \ \& \ S_{\alpha,\beta+1}^{[i,j]}(\tau-1)]. \end{aligned} \quad (4.50)$$

Перейти к выполнению требований пункта 3.

3. Выражение (4.50) путем применения к первому дизъюнктивному члену аксиомы 1 (4.44) и сокращенного обозначения на основании (4.44) соответствующей связной группы предикатов (обозначена квадратными скобками) преобразовать к виду:

$$S_{\alpha,\beta}^i(t) = \bigvee_{j=1}^{j=k_1} \chi_{\alpha,\beta}^{[i,j]}(t) \ \& \ S_{\alpha,\beta+1}^{[i,j]}(t-1) \vee q_{\alpha,\beta}^i(t) \ \& \ S_{\alpha,\beta}^i(t-1). \quad (4.51)$$

На этом работа алгоритма заканчивается.

4.2.4. Пример синтеза абстрактного цифрового автомата с использованием языка исчисления предикатов первого порядка

Требуется построить цифровой автомат Мура, заданный следующим словесным алгоритмом.

1. На вход автомата могут подаваться восемь элементарных входных сигналов:

$$[X] = x_0, x_1, x_2, x_3, x_4, x_5, x_6, x_7$$

2. Цифровой автомат должен функционировать циклически. Начало цикла должно определять событие S_0 , отмеченное выходным сигналом y_0 .

Событие S_0 должно иметь место при поступлении на вход автомата элементарного входного сигнала x_0 или x_7 независимо от наличия в этот момент времени других элементарных входных сигналов, а также в том случае, если после зарождения события S_0 другие входные сигналы на вход автомата еще не подавались.

3. В цифровом автомате должно иметь место событие S_1 , отмеченное выходным сигналом y_1 , если в интервале времени между поступлением, соответственно, сигналов x_5 и x_6 (не включая моменты времени поступления сигналов x_5 и x_6) на вход автомата было подано четное количество сигналов

x_2 , которое (это четное количество сигналов x_2) может чередоваться с произвольным количеством сигналов x_1 , при условии, что для всех моментов времени этого интервала другие сигналы, кроме быть может сигнала x_4 , на вход автомата не поступают.

Все сигналы, кроме сигналов x_0 и x_7 , поданные на вход автомата после поступления сигнала x_6 на событие S_1 не влияют.

Одновременно с сигналами x_5 и x_6 на вход автомата может поступать, не влияя на событие S_1 , только сигнал x_3 .

Прежде чем производить формализацию этого словесного алгоритма введем для сокращения записи следующие обозначения входных сигналов:

$$e_{2,4}(\tau) = \bar{x}_0(\tau) \& \bar{x}_1(\tau) \& \bar{x}_3(\tau) \& \bar{x}_5(\tau) \& \bar{x}_6(\tau) \& \bar{x}_7(\tau),$$

$$e_{1,4}(\tau) = \bar{x}_0(\tau) \& \bar{x}_2(\tau) \& \bar{x}_3(\tau) \& \bar{x}_5(\tau) \& \bar{x}_6(\tau) \& \bar{x}_7(\tau),$$

$$e_{3,5}(\tau) = \bar{x}_0(\tau) \& \bar{x}_1(\tau) \& \bar{x}_2(\tau) \& \bar{x}_4(\tau) \& \bar{x}_6(\tau) \& \bar{x}_7(\tau),$$

$$e_{3,6}(\tau) = \bar{x}_0(\tau) \& \bar{x}_1(\tau) \& \bar{x}_2(\tau) \& \bar{x}_4(\tau) \& \bar{x}_5(\tau) \& \bar{x}_7(\tau),$$

$$e_4(\tau) = \bar{x}_0(\tau) \& \bar{x}_1(\tau) \& \bar{x}_2(\tau) \& \bar{x}_3(\tau) \& \bar{x}_5(\tau) \& \bar{x}_6(\tau) \& \bar{x}_7(\tau).$$

Для упрощения процесса формализации предикатных формул, описывающих исходные события, воспользуемся временными схемами их формирования (рис. 4.1).

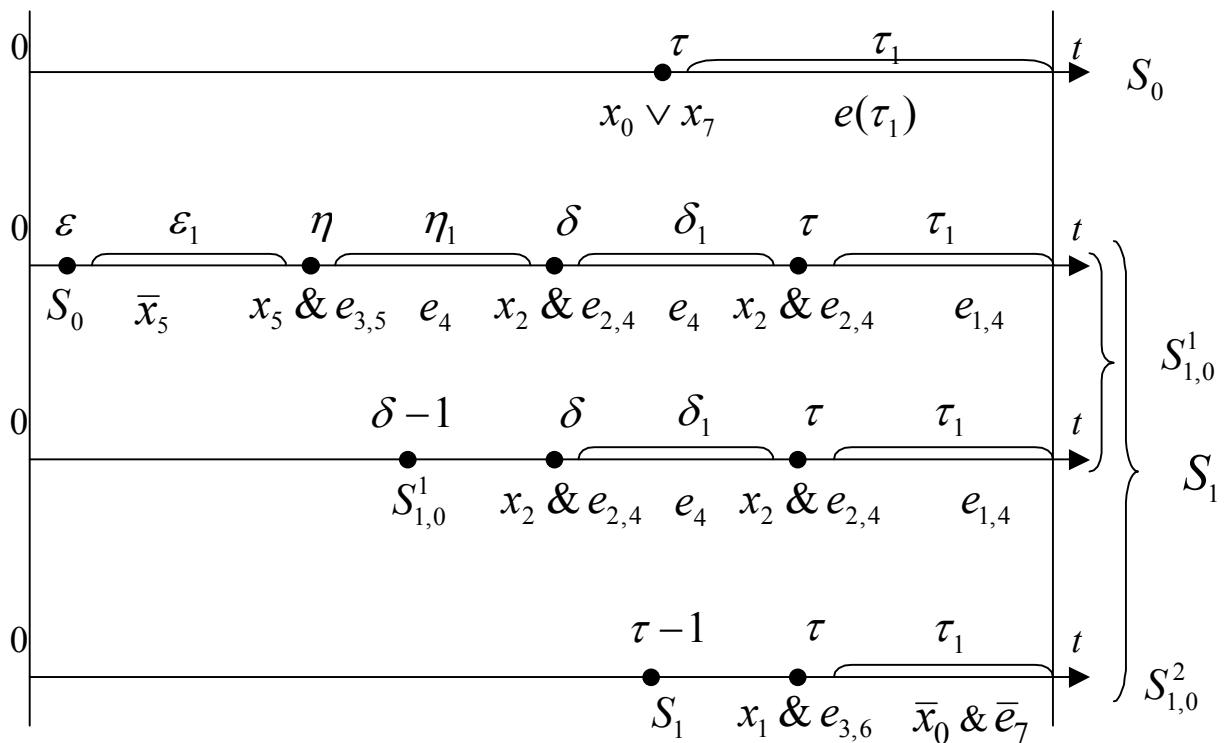


Рис. 4.1. Временные схемы формирования событий S_0 и S_1

Примечание. При построении временных схем формирования событий временные интервалы, в течении которых действуют входные сигналы, обозначаются буквами с индексами, а моменты времени поступления входных сигналов буквами без индексов.

Выполняя формализацию словесного алгоритма с учетом (4.48) и временных схем формирования событий (рис. 4.1), получим систему $(A)_p$, представляющую заданные события S_0 и S_1 , которая имеет вид:

$$\begin{aligned}
 S_0^{y_0}(t) &= (\exists \tau) [x_0(\tau) \vee x_7(\tau)] \& (\forall \tau_1) e(\tau_1), \\
 &\quad \tau \leq t \qquad \qquad \qquad \tau < \tau_1 \leq t \\
 S_1^{y_1}(t) &= (\exists \tau) x_2(\tau \& e_{2,4}(\tau) \& (\forall \tau_1) e_{1,4}(\tau_1) \& (\exists \delta) x_2(\delta) \& \\
 &\quad \tau \leq t \qquad \qquad \qquad \tau < \tau_1 \leq t \qquad \qquad \qquad \delta < t \\
 &\& e_{2,4}(\delta) \& (\forall \delta_1) e(\delta_1) \& (\exists \eta) x_5(\eta) \& e_{3,5}(\eta) \& \\
 &\quad \delta < \delta_1 \leq t \qquad \qquad \qquad \eta < \delta \\
 &\& (\forall \eta_1) e_4(\eta_1) \& (\exists \varepsilon) S_0(\varepsilon) \& (\forall \varepsilon_1) \bar{x}_5(\varepsilon_1) \vee \\
 &\quad \eta < \eta_1 < \delta \qquad \qquad \qquad \varepsilon < \eta \qquad \qquad \qquad \varepsilon < \varepsilon_1 < \eta \\
 &\vee (\exists \tau) x_2(\tau \& e_{2,4}(\tau) \& (\forall \tau_1) e_{1,4}(\tau_1) \& \\
 &\quad \tau \leq t \qquad \qquad \qquad \tau < \tau_1 \leq t \\
 &\& (\exists \delta) x_2(\delta) \& e_{2,4}(\delta) \& (\forall \delta_1) e(\delta_1) \& S_1(\delta - 1) \vee \\
 &\quad \delta < \tau \qquad \qquad \qquad \delta < \delta_1 < \tau \\
 &\vee (\exists \tau) x_6(\tau) \& e_{3,6}(\tau) \& (\forall \tau_1) \overline{x_0(\tau_1)} \& \overline{x_7(\tau_1)} \& S_1(\tau - 1). \\
 &\quad \tau \leq t \qquad \qquad \qquad \tau < \tau_1 \leq t
 \end{aligned} \tag{4.52}$$

Выполняя преобразование уравнения системы (4.52), определяющего событие S_1 , за счет введения сокращенного обозначения связных групп предикатных выражений, получим систему рекурсивных предикатных выражений, каждое из которых имеет одинаковую структуру, соответствующую предикатному выражению (4.48) и состоящую из упомянутых ранее 3-х частей.

$$\begin{aligned}
S_1(t) &= S_{1,0}^1 \vee S_{1,0}^2, \\
S_{1,0}^1(t) &= (\exists \tau)_{\tau \leq t} x_2(\tau) \& e_{2,4}(\tau) \& (\forall \tau_1)_{\tau < \tau_1 \leq t} e_{1,4}(\tau_1) \& S_{1,1}^1(\tau - 1), \\
S_{1,1}^1(t) &= (\exists \delta)_{\delta \leq \tau} x_2(\delta) \& e_{2,4}(\delta) \& (\forall \delta_1)_{\delta < \delta_1 < \tau} e(\delta_1) \& \\
&\quad \& [S_{1,2}^1(\delta - 1) \vee S_{1,0}^1(\delta - 1)], \\
S_{1,2}^1(t) &= (\exists \eta)_{\eta < \delta} x_5(\eta) \& e_{3,5}(\eta) \& (\forall \eta_1)_{\eta < \eta_1 < \delta} e_4(\eta_1) \& S_{1,3}^1(\eta - 1), \\
S_{1,3}^1(t) &= (\exists \varepsilon)_{\varepsilon < \eta} S_0(\varepsilon) \& (\forall \varepsilon_1)_{\varepsilon < \varepsilon_1 < \eta} \bar{x}_5(\varepsilon_1), \\
S_{1,0}^2(t) &= (\exists \tau)_{\tau \leq t} x_6(\tau) \& e_{3,6}(\tau) \& (\forall \tau_1)_{\tau < \tau_1 \leq t} \overline{x_0(\tau_1)} \& \overline{x_7(\tau_1)} \& S_1(\tau - 1).
\end{aligned} \tag{4.53}$$

Выполняя спуск кванторов в уравнениях системы (4.53) на основании аксиом (4.44) и (4.45), получим следующую систему бескванторных канонических уравнений, которая имеет вид:

$$\begin{aligned}
S_0(t) &= x_0(t) \vee x_7(t), \\
S_1(t) &= S_{1,0}^1 \vee S_{1,0}^2, \\
S_{1,0}^1(t) &= x_2(t) \& e_{2,4}(t) \& S_{1,1}^1(t - 1) \vee e_{1,4}(t) \& S_{1,0}^1(t - 1), \\
S_{1,1}^1(t) &= x_2(t) \& e_{2,4}(t) \& [S_{1,2}^1(t - 1) \vee S_{1,0}^1(t - 1)] \vee \\
&\quad \vee e_4(t) \& S_{1,1}^1(t - 1), \\
S_{1,2}^1(t) &= x_5(t) \& e_{3,5}(t) \& S_{1,3}^1(t - 1) \vee e(t) \& S_{1,2}^1(t - 1), \\
S_{1,3}^1(t) &= S_0(t) \vee \bar{x}_5(t) \& S_{1,3}^1(t - 1), \quad S_0(t) \Rightarrow S_{1,3}^1, \\
S_{1,0}^2(t) &= x_6(t) \& e_{3,6}(t) \& S_{1,0}^1(t - 1) \vee \bar{x_0}(t) \& \bar{x_7}(t) \& S_{1,0}^2.
\end{aligned} \tag{4.54}$$

Для наглядности при проверке правильности формального представленных событий можно использовать графы НДА, реализующие эти события. Для нашего примера граф НДА, построенный на основе системы (4.54) и реализующего событие S_1 , имеет вид (рис.4.2).

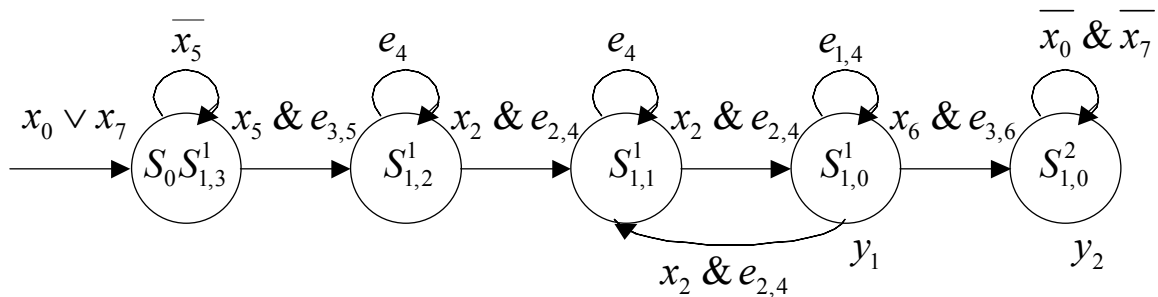


Рис.4.2. Граф НДА, реализующего событие S_1

В заключение рассмотрим представление события S_1 на языке РВАС и построение на его основе системы канонических уравнений. Для этого воспользуемся временными схемами формирования события S_1 (рис.4.1), на основании которого событие S_1 на языке РВАС может быть представлено следующим выражением:

$$S_1 = S_0 \{\bar{x}_5\} (x_5 e_{3,5}) \{e_4\} (x_2 e_{2,4}) \{e_4\} (x_2 e_{2,4}) \{e_{1,4}\} \{ (x_2 e_{2,4}) \{e_4\} (x_2 e_{2,4}) \{e_{1,4}\} \vee (x_6 e_{3,6}) \{(\bar{x}_0 \bar{x}_7)\} \} \quad (4.55)$$

Выполнив в выражении (4.55) раскрытие итерационных скобок, в которых представлено описание условия сохранения события S_1 , и некоторые предварительные преобразования, получим описание события S_1 в следующем виде:

$$S_1 = [S_0 \{\bar{x}_5\} (x_5 e_{3,5}) \{e_4\} (x_2 e_{2,4}) \vee S_1 (x_2 e_{2,4}) \{e_4\}] (x_2 e_{2,4}) \{e_{1,4}\} \vee S_1 (x_6 e_{3,6}) \{(\bar{x}_0 \bar{x}_7)\} = S_{1,0}^1 \vee S_{1,0}^2. \quad (4.56)$$

Полученное регулярное выражение (4.56) на основании методики, представленной в п. 4.1.2, преобразуется в следующую каноническую систему регулярных выражений для всех событий, реализуемых в исходном алгоритме для события S_1 :

$$\begin{aligned} S_{1,0}^1 &= S_{1,1}^1 (x_2 e_{2,4}) \{e_{1,4}\} = S_{1,1}^1 (x_2 e_{2,4}) \vee S_{1,0}^1 e_{1,4}, \\ S_{1,1}^1 &= S_{1,2}^1 (x_2 e_{2,4}) \{e_4\} \vee S_1 (x_2 e_{2,4}) \{e_4\} = (S_{1,2}^1 \vee S_1) e_4, \\ S_{1,2}^1 &= S_0 \{\bar{x}_5\} (x_5 e_{3,5}) \{e_4\} = S_{1,3}^1 (x_5 e_{3,5}) \{e_4\} = \\ &= S_{1,3}^1 (x_5 e_{3,5}) \vee S_{1,2}^1 e_4, \\ S_{1,3}^1 &= S_0 \{\bar{x}_5\} = S_0 \vee S_{1,3}^1 \bar{x}_5, \quad S_0 \Rightarrow S_{1,3}^1, \\ S_{1,0}^2 &= S_1 (x_6 e_{3,6}) \{(\bar{x}_0 \bar{x}_7)\} = S_1 (x_6 e_{3,6}) \vee S_{1,0}^2 \{(\bar{x}_0 \bar{x}_7)\}. \end{aligned} \quad (5.57)$$

Система уравнений (4.57) полностью соответствует канонической системе уравнений (4.54), полученной по описанию события S_1 на языке исчисления предикатов.

4.3. Язык граф-схем алгоритмов (ГСА)

4.3.1. Общие сведения о языке ГСА

Граф-схемы алгоритмов нашли широкое применение для описания функционирования управляющих микропрограммных устройств в различных отраслях техники, в том числе и в вычислительной технике. Это объясняется их хорошей обозримостью, наглядностью и простотой конструкции, а также

возможностью преобразований и формального перехода к таблицам перехода и к аналитическому представлению алгоритмов в виде СКУ.

Язык ГСА совместно с языком логических схем алгоритмов (ЛСА) иногда называют одним общим названием язык операторных схем алгоритмов (ОСА).

Язык ГСА - графический язык, поэтому символы, применяемые в нем, имеют определенное геометрическое начертание определяемое ГОСТом. В языке ГСА применяются четыре основных символа для обозначения начальной, конечной, операторных и логических (условных) вершин. Для указания связи между прерванными линиями внутри одного листа или между разными листами могут быть использованы два дополнительных символа. Любой алгоритм должен начинаться и заканчиваться символами начальной и конечной вершин. Начальная вершина A_0 имеет только одну выходящую линию, а конечная A_k - только входящие линии. Операторной вершине сопоставляется вполне определенный оператор A_i , символизирующий определенные действия. На первом этапе проектирования алгоритма функционирования цифрового автомата используют содержательные ГСА, когда внутри операторных вершин записывается, содержательное обозначение управляющих действий. Например, для алгоритма выполнения каких-либо операций в операционном устройстве ЭВМ внутри операторных вершин ГСА управления выполнением таких операций записываются микрооперации в виде оператора присваивания или совокупности таких операторов. Внутри условных вершин в содержательный ГСА записывается некоторое логическое выражение, принимающее значение 1 или 0, или "Истина" или "Ложно".

При кодировании содержательной ГСА внутри операторных вершин записываются символы из множества выходных сигналов структурного алфавита $[Y]$, которые в МПА иницируют выполнение соответствующих микроопераций в операционном автомате, а внутри условных вершин - из множества входных сигналов структурного алфавита $[X]$, которые для МПА принято называть осведомительными сигналами.

Операторная вершина имеет одну входящую и одну выходящую линии, причем входящая линия может быть образована слиянием нескольких линий. Условная вершина имеет одну входящую линию и две выходящих. Входящая линия также может быть образована слиянием нескольких линий. Выходящие линии помечаются цифрами 1 и 0 или словами "Да" и "Нет".

Запись алгоритма функционирования цифрового автомата на языке ГСА сводится к начертанию графических символов указанных типов и соединению их в определенном порядке между собой с помощью линий. При этом должны соблюдаться следующие условия:

- 1) Входы и выходы вершин соединяются друг с другом с помощью линий, направленных от выхода к входу.
- 2) Каждый выход соединен точно с одним входом.

- 3) Любой вход соединяется по крайней мере с одним выходом.
- 4) Для любой вершины графа существует по крайней мере один путь из этой вершины к конечной вершине.
- 5) Один из выходов условной вершины может соединяться с ее входом, что не допустимо для операторной вершины. В случае соединения выхода условной вершины с ее входом в цепь обратной связи вводится пустая операторная вершина, отмеченная пустым выходным сигналом y_e .

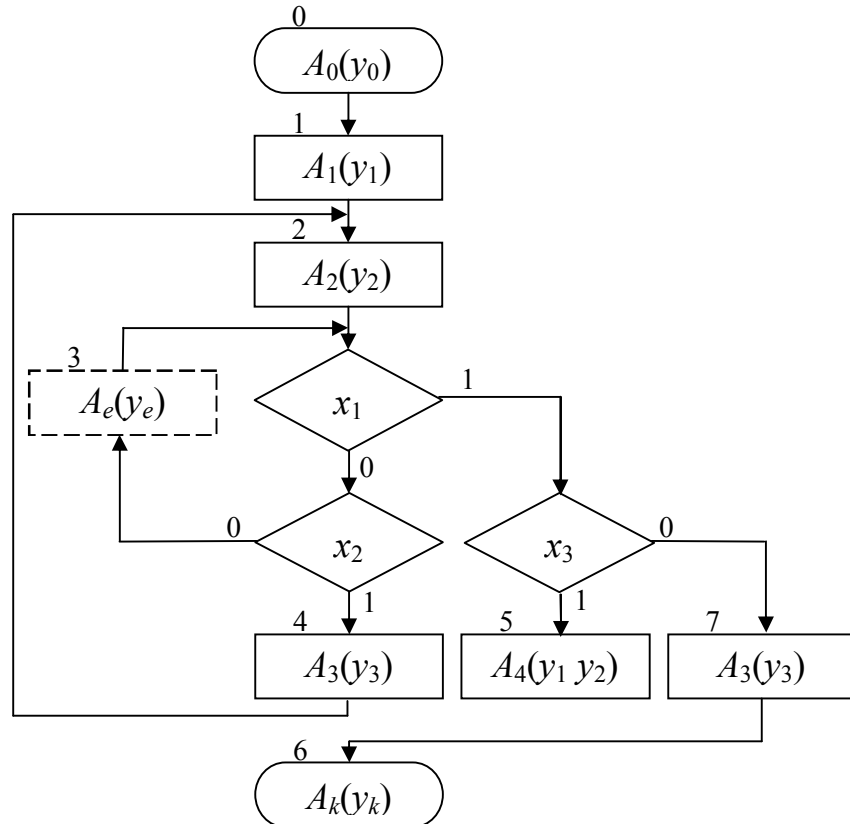


Рис. 4.3. Граф-схема алгоритма произвольного автомата

Рассмотрим выполнение алгоритма по ГСА, представленной на рис.4.3. Выполнение алгоритма начинается с оператора A_0 . На следующих двух шагах выполняется последовательно операторы A_1 и A_2 . На четвертом шаге может выполняться или один из трех операторов A_3 , A_4 и A_k , или выполняется пустой оператор A_e в зависимости от значений логических условий x_1 , x_2 и x_3 . Если $x_1 = x_2 = 0$, то независимо от значения сигнала x_3 будет выполняться пустой оператор A_e , т.е. автомат будет находиться в режиме ожидания, выдавать пустой сигнал y_e , до тех пор, пока или x_1 или x_2 не станут равными единице. Если $\bar{x}_1 = x_2 = 1$, то выполняется оператор A_3 , если $x_1 = 1$, то выполняется или A_4 при $x_3 = 1$, или A_3 при $x_3 = 0$. Предположим, что после выполнения A_2 имеет место $\bar{x}_1 = x_2 = 1$. Тогда независимо от значения x_3 будут последовательно выполняться A_2 и A_3 столько раз, пока после очередного выполнения оператора A_2 и проверки

логических условий логическое условие x_1 не станет равным единице. После этого выполняется оператор A_4 или как было отмечено выше. Выполнение алгоритма закончится оператором A_k .

4.3.2. Построение таблиц переходов и СКУ по ГСА

СКУ в принципе можно построить сразу непосредственно по ГСА, поставив в соответствие каждой операторной вершине событие СКУ, учитывая при этом, что одинаковым операторам, стоящим в разных местах ГСА, должны соответствовать разные события. Однако для облегчения процесса построения СКУ целесообразно сначала по ГСА построить прямую таблицу переходов для событий, которая будет соответствовать прямой таблице переходов автомата, т.к. рассматриваемый вид ГСА не допускает параллельных ветвей алгоритма. По прямой таблице переходов строится в дальнейшем СКУ. Тем более, что во многих случаях прямая таблица переходов автомата используется для структурного синтеза автомата и самостоятельно. Таким образом, построенная по ГСА СКУ будет детерминированной всегда и каждому событию СКУ будет соответствовать свое вполне определенное состояние автомата.

Алгоритм построения СКУ для автомата Мура будет включать следующие этапы:

1) Разметить (пронумеровать) все вершины ГСА, предварительно введя пустые операторные вершины, если в ГСА имеются циклы из условных вершин. Пустые операторные вершины размещают при выходе из условной вершины, определяющей начало пути в цикле из условных вершин.

2) Каждому оператору ГСА поставить в соответствие вполне определенное событие (состояние автомата), присвоив ему индекс, соответствующий номеру вершины ГСА. При этом необходимо иметь в виду, что одинаковым операторам, стоящим в разных местах ГСА, должны соответствовать разные события (состояния автомата). Это обеспечивается сквозной нумерацией вершин исходной ГСА. Начальной вершине ГСА будет соответствовать начальное событие S_0 (начальное состояние автомата a_0). Часто начальную и конечную вершины ГСА совмещают, тогда автомат после выполнения программы алгоритма переходит в начальное состояние.

3) Заготовить прямую таблицу переходов, как это выполнялось в гл.2, и отметить каждую строку в столбце 2 обозначением одного из исходных событий (состояний автомата), начиная с начального события. Осуществить просмотр всех путей перехода от исходного события (состояния автомата) в момент времени (t) к событиям (состояниям автомата) в конце пути в момент времени $(t+1)$. Каждый такой путь должен соответствовать шагу алгоритма и состоять только из логических условий (условных вершин).

Каждому пути из логических условий от одного оператора к другому сопоставить конъюнкцию (сочетание) входных сигналов, т.е. частный входной сигнал автомата $X_{i,j}$. Причем в эту конъюнкцию элементарный

входной сигнал войдет без отрицания, если выход из условной вершины отмечен 1, и с отрицанием - 0.

Для каждого исходного события $S_i(t)$ (состояния автомата) для всех путей, ведущих из него записать в третьем столбце выражения для частных входных сигналов $X_{i,j}$, а в четвертом столбце записать соответствующие им обозначения событий $S_j(t+1)$ перехода (состояний автомата).

Примечание. Частные входные сигналы $X_{i,j}$, определяемые по ГСА, соответствуют условиям так называемого универсального распределения сдвигов, когда логические условия могут изменяться в любом такте работы автомата. Другие случаи распределения сдвигов рассматриваются, например в [23-26].

4) Отметив каждое событие (состояние автомата) соответствующими выходными сигналами, получим прямую таблицу переходов автомата Мура, которую необходимо проверить на удовлетворение условиям автоматности: для каждого исходного события (состояния автомата) любое попарное произведение частных входных сигналов для всех путей из каждого из исходных событий должно равняться нулю, а логическая сумма всех частных входных сигналов должна равняться единице.

5) Построение SKU осуществляется путем отыскания в прямой таблице переходов всех путей, которые оканчиваются одинаковыми событиями $S_j(t+1)$ и составляются для них конъюнкции из обозначений событий $S_i(t)$, стоящих вначале пути перехода, с соответствующими частными входными сигналами $X_{i,j}$, вызывающими этот переход. Полученные конъюнкции $X_{i,j}(t) \& S_i(t)$ объединяют знаком дизъюнкции. Такую операцию выполняют для всех событий $S_j(t+1)$, которые отмечают соответствующими выходными сигналами.

Пример 4.5. Построить прямую таблицу переходов и SKU по ГСА, представленной на рис. 4.3, для автомата Мура. Введя сквозную нумерацию операторных вершин и следуя предложенному алгоритму, получим искомую прямую таблицу переходов автомата Мура (табл. 4.5.).

Таблица 4.5.

	Исходное событие (состояние) $S_i(t)$	Частный входной сигнал $X_{i,j}(t)$	Событие (состояние перехода), отмеченное совокупностью выходных сигналов $S_j(t+1)(Y_j)$
1	2	3	4
1	S_0	1	$S_1(y_1)$
2	S_1	1	$S_2(y_2)$
3	S_2	$\bar{x}_1 \bar{x}_2$	$S_3(y_e)$
		$\bar{x}_1 x_2$	$S_4(y_3)$
		$x_1 \bar{x}_3$	$S_7(y_3)$
		$x_1 x_3$	$S_5(y_1 y_2)$
4	S_3	$\bar{x}_1 \bar{x}_2$	$S_3(y_e)$
		$\bar{x}_1 x_2$	$S_4(y_3)$
		$x_1 \bar{x}_3$	$S_7(y_3)$
		$x_1 x_3$	$S_5(y_1 y_2)$
5	S_4	1	$S_2(y_2)$
6	S_5	1	$S_6(y_k)$
7	S_7	1	$S_6(y_k)$

По табл. 4.5. в соответствии с пунктом 5 алгоритма получим следующую СКУ для автомата Мура:

$$S_1^{y_1}(t+1) = S_0,$$

$$S_2^{y_2}(t+1) = S_1 \vee S_4,$$

$$S_3^{y_e}(t+1) = \bar{x}_1 \bar{x}_2 (S_2 \vee S_3),$$

$$S_4^{y_3}(t+1) = \bar{x}_1 x_2 (S_2 \vee S_3),$$

$$S_5^{y_1 y_2}(t+1) = x_1 x_3 (S_2 \vee S_3),$$

$$S_6^{y_k}(t+1) = S_5 \vee S_7,$$

$$S_7^{y_3}(t+1) = x_1 \bar{x}_3 (S_2 \vee S_3).$$

В том случае, если требуется построить автомат на основе модели автомата Мили с использованием прямой таблицы переходов, необходимо в таблице переходов автомата Мура объединить те события (состояния)

автомата Мура, переходы из которых полностью совпадают. Для рассмотренного выше примера это соответствует группам событий (состояний): $S_2, S_3; S_1, S_4$ и S_5, S_7 . Делая необходимые подстановки и замены переменных вида: $S_0 = a_0; S_1 \vee S_4 = a_1; S_2 \vee S_3 = a_2; S_5 \vee S_7 = a_3; S_6 = a_4$, получим прямую таблицу переходов автомата Мили, эквивалентного автомату Мура (табл.4.6), дополненную столбцом 5, отмечающим переходы выходными сигналами.

Таблица 4.6.

Прямой таблице переходов (табл. 4.6) будут соответствовать следующие СКУ (4.58) и СВФ (4.59) автомата Мили:

	Исходное состояние $a_i(t)$	Частный входной сигнал $X_{i,j}(t)$	Состояние перехода $a_j(t+1)$	Комбинация выходных сигналов $Y_{i,j}(t)$
1	2	3	4	5
1	a_0	1	a_1	y_1
2	a_1	1	a_2	y_2
3	a_2	$\bar{x}_1 \bar{x}_2$	a_2	y_e
		$\bar{x}_1 x_2$	a_1	y_3
		$x_1 \bar{x}_3$	a_3	y_3
		$x_1 x_3$	a_3	$y_1 y_2$
4	a_3	1	a_4	y_k

$$\begin{aligned}
a_1(t+1) &= a_0 \vee \bar{x}_1 x_2 a_2, \\
a_2(t+1) &= a_1 \vee \bar{x}_1 \bar{x}_2 a_2, \\
a_3(t+1) &= x_1 a_2, \\
a_4(t+1) &= a_3.
\end{aligned}
\tag{4.58}$$

$$\begin{aligned}
y_1(t) &= a_0 \vee x_1 x_3 a_2, \\
y_2(t) &= a_1 \vee x_1 x_3 a_2, \\
y_3(t) &= (\bar{x}_1 x_2 \vee x_1 \bar{x}_3) a_2, \\
y_e(t) &= \bar{x}_1 \bar{x}_2 a_2, \\
y_k(t) &= a_3.
\end{aligned}
\tag{4.59}$$

4.4. Логические схемы алгоритмов (ЛСА)

4.4.1. Общие сведения о языке ЛСА

Язык ЛСА является аналитической интерпретацией языка ГСА и может быть использован для более компактной формы записи алгоритма функционирования автомата. ЛСА были впервые предложены в 1952-53 г.г. А.А. Ляпуновым для целей программирования. В дальнейшем они стали широко использоваться в качестве начального языка задания алгоритмов функционирования управляющих устройств [24, 26, 45]. В ЛСА операторы записываются в строчку слева направо. При этом последовательность выполнения операторов будет определяться порядком их записи. Порядок выполнения операторов может быть строго фиксированным или зависящим от некоторых логических условий. В последнем случае применяют логическую переменную $x_m \uparrow^i$, со стоящей от нее началом стрелки $\uparrow^i$ с индексом $i = 1, 2, \dots$. Конец стрелки $\downarrow^i$ с этим же индексом стоит слева от того члена ЛСА, который должен выполняться, если логическая переменная x_m принимает нулевое значение.

Таким образом, логической схемой алгоритма называют конечную строчку, составленную из символов операторов $A_0, A_1, \dots, A_k$, логических переменных (логических условий) $x_1 \uparrow^i, x_2 \uparrow^j, \dots, x_L \uparrow^r$ и концов стрелок $\downarrow^i, \downarrow^j, \dots, \downarrow^r$, таких, что для каждого начала стрелки, например, с индексом i найдется один и только один конец стрелки с тем же индексом.

Пример. Строчка $A_1 x_m \uparrow^i A_2 \dots \downarrow^i A_n$ означает, что после выполнения оператора A_1 в зависимости от значения логического условия x_m может быть выполнен оператор A_2 , стоящий непосредственно справа от $x_m \uparrow^i$, если $x_m = 1$, или - оператор A_n , стоящий непосредственно справа от конца стрелки $\downarrow^i$ с тем же индексом, что и начало стрелки, если $x_m = 0$.

В некоторых случаях целесообразно выделять логические условия, которые всегда принимает нулевое (ложное) значение, т.е. тождественно-ложные логические условия. Принято обозначать такие условия через ω (оператор ω). После оператора ω всегда производится переход по стрелке, которая стоит справа от него.

Приведем в качестве примера запись ЛСА, эквивалентную ГСА, рассмотренной в п. 4.3.1. Учитывая основные конструкции языка ЛСА, эта запись будет иметь вид:

$$A_0(y_0)A_2(y_1)\downarrow^3 A_2(y_2)\downarrow^2 \bar{x}_1 \uparrow^1 x_2 \uparrow^2 A_3(y_3)\omega \uparrow^3 \downarrow^1 x_3 \uparrow^4 A_4(y_1 y_2)\omega \uparrow^5 \downarrow^4 A_3(y_3)\downarrow^5 A_k(y_k) \quad (4.60)$$

4.4.2. Построение таблиц переходов и СКУ по ЛСА

Методика построения таблицы переходов и СКУ по ЛСА в принципе мало отличается от аналогичной методики для ГСА. Поэтому отметим только уточнения по выполнению отдельных этапов приведенного выше алгоритма [2].

1) По пункту 1. В том случае, если в ЛСА имеются циклы из логических условий, то в ЛСА вводится пустой оператор $A_e(y_e)$, отмеченный пустым выходным сигналом. Этот оператор помещается в ЛСА путем замены символа конца стрелки $\downarrow^i$, помещенной вначале петли из логических условий, на следующую группу символов ЛСА: $\omega \uparrow^k \downarrow^i A_e(y_e) \downarrow^k$. Например, для фрагмента ЛСА: $A_p \downarrow^2 x_1 \uparrow^1 x_2 \uparrow^2 A_n \dots \downarrow^1 A_m$, получим следующую скорректированную ЛСА: $A_p \omega \uparrow^k \downarrow^2 A_e(y_e) \downarrow^k x_1 \uparrow^1 x_2 \uparrow^2 A_n \dots \downarrow^1 A_m$.

2) По пункту 3. Замечания относительно всех путей из логических условий при переходе от одного оператора к другому и определения выражения для частного входного сигнала $X_{i,j}$. В конъюнкцию из логических переменных, т.е. в сигнал $X_{i,j}$, логическая переменная, например, x_m войдет с отрицанием или без отрицания в зависимости от следующего:

а) без отрицания, если $x_m = 1$ (логическое условие истинно), тогда проверяется следующая по порядку записи логическая переменная (логическое условие), например x_α , если она присутствует в строке;

б) с отрицанием, если $x_m = 0$ (логическое условие ложно), тогда проверяется по стрелке, стоящей справа от $x_m \uparrow^e$, следующая логическая переменная (логическое условие), например x_p , стоящая справа от конца стрелки $\downarrow^e$.

Пример 4.6. Найдем все пути из логических условий от оператора A_0 для фрагмента ЛСА:

$$A_0 x_m \uparrow^e x_\alpha \uparrow^k A_n \downarrow^k A_\gamma \dots \downarrow^e x_p \uparrow^c A_\varepsilon \dots \downarrow^c A_k.$$

Учитывая отмеченные замечания по определению выражения для частного входного сигнала $X_{i,j}$, для заданного фрагмента ЛСА получим:

первый путь	- $X_{i,j} = x_m x_\alpha$	к оператору перехода A_n ;
второй путь	- $X_{i,j} = \bar{x}_m x_p$	к оператору перехода A_ε ;
третий путь	- $X_{i,j} = \bar{x}_m \bar{x}_p$	к оператору перехода A_k ;
четвертый путь	- $X_{i,j} = x_m \bar{x}_\alpha$	к оператору перехода A_γ .

Если в ЛСА имеется безусловный переход, например для фрагмента ЛСА $A_\alpha \omega \uparrow^k \dots \downarrow^k A_p$, то путь от исходного оператора A_α до конечного в конце пути определится концом стрелки безусловного перехода. В данном случае $X_{i,j} = 1$.

3) Пункты 2, 4 и 5 алгоритма полностью совпадают.

Пример 4.7. Построить прямую таблицу переходов автомата Мура и соответствующую СКУ для ЛСА, эквивалентной ГСА, представленной на рис. 4.3.

Учитывая условия по введению в ЛСА пустого оператора, там где имеются петли из логических переменных (логических условий), получим следующую скорректированную ЛСА:

$$\begin{array}{ccccccc}
 A_0(y_0)A_1(y_1) \downarrow^3 & A_2(y_2) \omega \uparrow^k \downarrow^2 & A_e(y_e) \downarrow^k \bar{x}_1 \uparrow^1 & x_2 \uparrow^2 & A_3(y_3) \omega \uparrow^3 & & \\
 0 & 1 & 2 & & 3 & 4 & \\
 \downarrow^1 x_3 \uparrow^4 & A_4(y_1 y_2) \omega \uparrow^5 \uparrow^4 & A_3(y_3) \downarrow^5 & A_k(y_k) & & & \\
 & 5 & 7 & 6 & & &
 \end{array} \quad (4.61)$$

Разметив операторы ЛСА (4.61) таким же образом, как и в ГСА на рис. 4.3., дальнейший ход решения поставленной задачи с учетом замечаний по определению всех путей из исходных операторов и соответствующих выражений для $X_{i,j}$ полностью совпадает с решением аналогичной задачи на основе ГСА.

4.5. Язык операторных схем алгоритмов с параллельными ветвями (ОСАП)

4.5.1. Общие сведения о языке ОСАП

Возможность формального описания параллельно выполняемых во времени ветвей алгоритма значительно способствует увеличению выразительных средств любого начального языка. Однако параллельные алгоритмы обладают большим разнообразием по сравнению с обычными, что существенно затрудняет осмысливание и правильный учет всех факторов, связанных с параллельно протекающими процессами и их взаимодействием.

Предлагаемый в данном разделе язык ОСАП базируется на расширении синтаксиса известных языков ГСА и ЛСА, а также на использовании модели НДА и стандартного языка СКУ для описания всех частных событий. Основные идеи рассматриваемого языка ОСАП изложены в работах автора в [2, 16-19]. Выбор такого базиса для языка ОСАП вытекает из следующих предпосылок. Использование модели НДА позволяет описывать параллельные ветви алгоритма в виде частных СКУ независимо друг от друга на основе обычных ГСА или ЛСА, а взаимодействие параллельных ветвей и их синхронизацию можно учитывать при построении общей НД СКУ путем надлежащей формализации событий, определяющих выход алгоритмического процесса за вершину объединения ветвей и коррекции частных событий СКУ, полученных для каждой параллельной ветви.

Применение НД СКУ для формализации взаимодействия параллельных ветвей, а при необходимости и детерминированной СКУ позволяет снять многие ограничения на виды и условия взаимодействия параллельных ветвей, отмеченные в ряде работ. Например, в [43] конечные члены каждой параллельной ветви формализуемого алгоритма должны оканчиваться одновременно; в [39] алгоритмы не должны содержать циклы; в [40] ограничиваются рассмотрением параллельных ветвей, представленных двумя частными видами ГСА; в [41] предлагаемый язык характеризуется ограниченными возможностями по организации взаимодействия параллельных ветвей.

Предлагаемый язык ОСАП, обладая сравнительно широкими выразительными возможностями, позволяет также использовать многие результаты по синтезу цифровых автоматов, полученные до настоящего времени, например, в [26, 42] и др.

Дальнейшее изложение основных положений языка ОСАП будем вести на основе использования языка ГСА в связи с его большей наглядностью по сравнению с языком ЛСА. Учитывая это обстоятельство будем сокращенно обозначать язык ОСАП на основе языка ГСА через ГСАП.

4.5.2. Основные конструкции, вводимые в язык ГСАП

Для описания параллельных ветвей алгоритма и их взаимодействия в язык ГСА вводятся дополнительные вершины, указывающие места разветвления и объединения параллельных ветвей: разветвители и соединители.

Разветвителю в ГСАП будет соответствовать разветвительная вершина (рис. 4.4) обозначаемая буквой F_{α} , которая имеет один вход и n выходов, α - номер вершины.

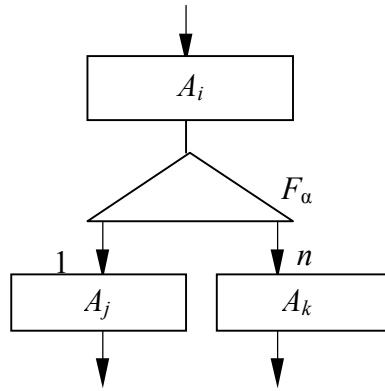


Рис. 4.4. Фрагмент ГСАП с разветвительной вершиной

Соединителю в ГСАП будет соответствовать соединительная вершина (рис. 4.5) обозначаемая буквой J_β , имеющая столько входящих линий, сколько параллельных ветвей она объединяет, β - номер вершины.

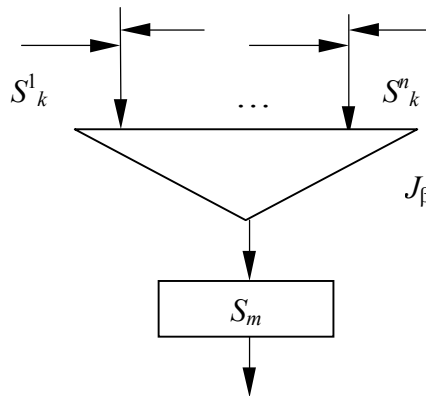


Рис. 4.5. Фрагмент ГСАП с соединительной вершиной

Каждая входящая линия может быть образована слиянием нескольких линий, но все компоненты такой линии должны принадлежать одной и той же параллельной ветви. Выходящая линия из соединителя может быть только одна.

Объединение параллельных ветвей в соединителе осуществляется с учетом определенных условий, т.е. при наступлении определенных ситуаций в объединяемых параллельных ветвях. В самом общем виде условия объединения могут быть заданы в виде булевой функции над обозначениями частных событий, имеющих место в объединяемых параллельных ветвях:

$$S_B = f_B(\tilde{S}_k^1, \dots, \tilde{S}_k^n, \tilde{S}_\alpha^1, \dots, \tilde{S}_\beta^n), \quad (4.62)$$

где $\tilde{S}_k^1, \dots, \tilde{S}_k^n$ - сокращенное обозначение событий, отмечающих окончание алгоритмического процесса в соответствующих ветвях с 1-ой по n -ой;

$\tilde{S}_\alpha^1, \dots, \tilde{S}_\beta^n$ - сокращенное обозначение частных событий, реализуемых в ветвях с 1-ой по n -ой;

$\sim$ - знак, означающий, что переменная может быть взята с отрицанием или без отрицания.

В соответствии с выражением (4.62) алгоритмический процесс распространяется за вершину соединения, если будет истинным выражение (4.62) т.е. имеет место соотношение:

$$S_m(t+1) = S_B(t), \quad (4.63)$$

где S_m - первое событие, реализуемое в последовательной части алгоритма после соединителя.

Практический интерес представляет ряд частных случаев задания условий для объединения параллельных ветвей, которые формализуются с использованием соединителей трех типов: конъюнктивного соединителя, дизъюнктивного соединителя и конъюнктивно-дизъюнктивного соединителя.

Конъюнктивный соединитель $J(\&)$.

При использовании такого соединителя алгоритмический процесс распространяется за вершину соединителя в том случае, если имеет определенное сочетание (конъюнкция) событий, отмечающих окончание алгоритмического процесса в объединяемых ветвях взятых с отрицанием или без отрицания. При этом возможны два частных случая. Первый случай соответствует алгоритму, когда алгоритмический процесс распространяется за вершину соединителя при условии окончания алгоритмического процесса во всех объединяемых ветвях одновременно (в один и тот же момент времени). Для такого соединителя имеет место соотношение:

$$S_B = \tilde{S}_k^1 \cdots \tilde{S}_k^i \cdots \tilde{S}_k^n, \quad (4.64)$$

Второй случай соответствует условию окончания алгоритмических процессов во всех объединяемых ветвях, но не обязательно в один и тот же момент времени. В этом случае факт окончания алгоритмического процесса в каждой из параллельных ветвей должен запоминаться, т.е. этот факт должен найти отражение при формализации условий сохранения событий, отмечающих окончание алгоритмического процесса в каждой из параллельных ветвей. Тогда для любой i -ой ветви событие S_k^1 должно иметь вид:

$$S_k^i(t+1) = S_k^i(0) \vee S_k^i \cdot \bar{S}_B, \quad (4.65)$$

где $S_k^i(0)$ - сокращенное обозначение события, определяющего зарождение события S_k^i (его первоначальное появление).

Из соотношения (4.65) следует, что событие S_k^i после своего зарождения сохраняется до тех пор, пока не будет истинным событие S_B .

Дизъюнктивный соединитель $J(v)$.

Для таких соединителей алгоритмический процесс распространяется за соединитель при окончании его в любой из соединяемых ветвей, т.е. имеем:

$$S_B = S_k^1 \vee \dots \vee S_k^i \vee \dots \vee S_k^n, \quad (4.66)$$

Из выражения (4.66) следует, что если по инициативе какой-либо ветви процесс распространился за вершину соединения, а в это время в другой ветви процесс достигает соединителя, то возможна неоднозначность выполнения алгоритма. Если это недопустимо, то должны быть предусмотрены меры устранения такой неоднозначности. Для этой цели выполняют коррекцию алгоритма таким образом, чтобы при достижении алгоритмическим процессом в любой ветви соединителя, он распространялся бы за соединитель с одновременной ликвидацией алгоритмических процессов во всех параллельных ветвях. Это обеспечивается коррекцией описания частных событий во всех параллельных ветвях с учетом того, что все частные события в любой i -ой ветви должны быть несовместимы с событиями, отмечающими окончание алгоритмического процесса во всех других ветвях. В общем случае коррекцию частных событий в любой i -ой ветви можно представить таким образом:

$$S_\alpha^i(t+1) = S_\alpha^i(0) \cdot \bar{S}_B, \quad (4.67)$$

где $S_\alpha^i(0)$ - сокращенное обозначение события S_α^i без учета коррекции.

Соединитель комбинированный $J(\&,v)$.

Для комбинированного соединителя функция S_B имеет вид:

$$S_B = V \tilde{S}_k^1 \dots \tilde{S}_k^i \dots \bar{S}_k^n, \quad (4.68)$$

Из (4.68) следует, что S_B есть дизъюнктивная нормальная форма от переменных типа S_k^i .

Для такого типа соединителя однозначность выполнения алгоритма также требует коррекции всех частных событий во всех параллельных ветвях, как и для дизъюнктивного соединителя. Это означает, что для всех частных событий всех параллельных ветвей правая часть их описания должна быть умножена на $\bar{S}_B$ в соответствии с выражением (4.67).

В тех случаях, когда выход за соединитель $J(\&,v)$ в соответствии с алгоритмом может осуществляться не обязательно при одновременном появлении событий типа S_k^i , то эти события после их зарождения следует сохранять до тех пор, пока не будет истинным комбинационное событие S_B . В связи с этим в описаниях событий типа S_k^i должны быть предусмотрены условия сохранения событий S_k^i в соответствии с выражением (4.65).

4.5.3. Построение таблиц переходов и СКУ для цифровых автоматов, заданных на языке ГСАП

Цифровые автоматы, заданные на языке ГСАП, являются недетерминированными, т.к. отдельные параллельные ветви алгоритма управления имеют общую вершину, от которой выполняется разветвление алгоритмического процесса. В связи с этим прямая таблица переходов (ПТП), соответствующая такому автомату, также будет недетерминированной, хотя она и будет представлять собой совокупность нескольких детерминированных ПТП, к числу которых относятся: детерминированные ПТП, соответствующие отдельным параллельным ветвям исходной ГСАП с общей начальной вершиной и детерминированная ПТП, которая соответствует последовательной части алгоритма управления, состоящей из начальной части до разветвительной вершины и конечной части после соединительной вершины.

Недетерминированная ПТП для автомата, заданного на языке ГСАП, будет строиться в два этапа. Сначала строят независимо друг от друга ПТП для отдельных параллельных ветвей и последовательной части алгоритма управления также как и для обычных ГСА, а затем выполняют необходимую коррекцию частных событий, реализуемых алгоритмом управления. Коррекция частных событий выполняется с учетом условий выхода алгоритмического процесса за вершину соединения параллельных ветвей.

Исходя из основных положений языка ГСАП алгоритм построения детерминированной ПТП для управляющего автомата, заданного на языке ГСАП, будет включать следующие этапы:

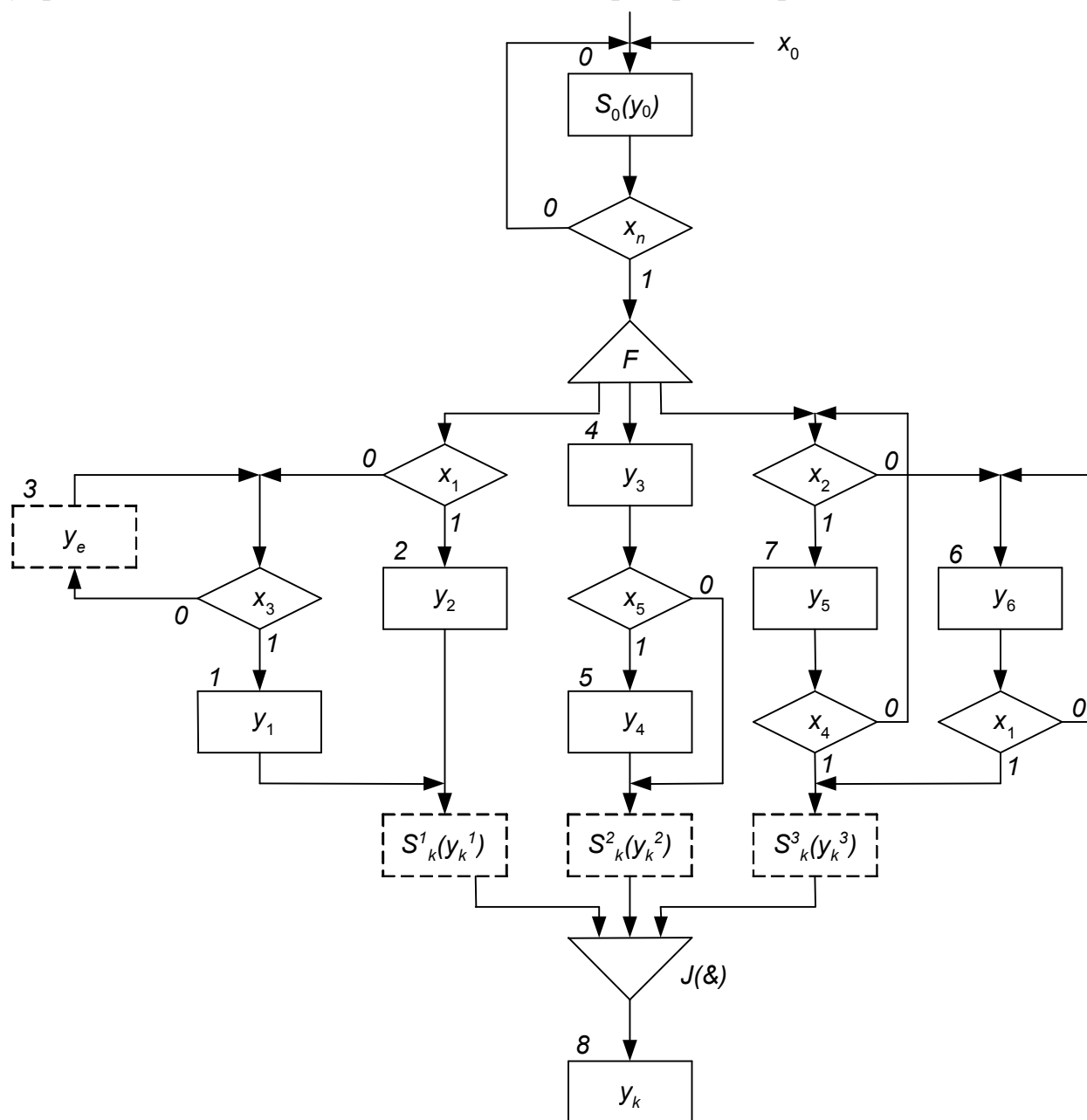
- 1) Построение недетерминированной ПТП управляющего автомата, состоящей из совокупности отдельных детерминированных ПТП для всех частных событий, реализуемых в параллельных ветвях и последовательной части алгоритма управления. Построение таких ПТП выполняется также как и для обычных ГСА без учета связи между отдельными ветвями.

- 2) Выполнение коррекции всех частных событий, полученных на предыдущем этапе, в соответствии с заданными условиями выхода алгоритмического процесса за вершину объединения параллельных ветвей. В результате получим общую скорректированную недетерминированную ПТП управляющего автомата.

- 3) Выполнение детерминизации недетерминированной ПТП, полученной на предыдущем этапе на основе алгоритма детерминизации, предложенного в главе 2.

П р и м е р 4.8. Для управляющего автомата, заданного на языке ГСАП (рис. 4.6), построить недетерминированную и детерминированную ПТП и соответствующие им СКУ автомата Мура при условии, что выход алгоритмического процесса за вершину объединения параллельных ветвей осуществляется после окончания алгоритмических процессов во всех ветвях, но не обязательно одновременно. Для заданной исходной ГСАП для

простоты примера начальная и конечная последовательные части алгоритма управления включают только по одной операторной вершине.



. 4.6.

и конъюнктивным соединителем, x_n - пусковой сигнал

В соответствии с заданными условиями выхода алгоритмического процесса за вершину объединения в исходную ГСАП вводятся промежуточные вершины S^1_k , S^2_k , S^3_k и соответствующие им события, отмеченные сигналами y^1_k , y^2_k , y^3_k , которые свидетельствуют об окончании алгоритмических процессов в соответствующих параллельных ветвях. Вводится также в первой ветви пустая операторная вершина в петлю из логического условия x_3 , отмеченную пустым выходным сигналом y_e .

Выполнив сквозную нумерацию операторных вершин и учитывая заданное условие объединения ветвей $S_B = S_k^1 S_k^2 S_k^3$, получим недетерминированную ПТП для всех частных событий, реализуемых в управляющем автомате (табл. 4.7)

Таблица 4.7.

Шаг алгоритма	Совокупность Исходных событий $R_i(t)$	Частный входной сигнал $X_{i,j}(t)$	Событие перехода $S_j(t+1)(Y_j)$	Примечание
1	S_0	$x_n x_1$	$S_2(y_2)$	Для 1 –ой ветви
		$x_n \bar{x}_1 x_3$	$S_1(y_1)$	
		$x_n \bar{x}_1 \bar{x}_3$	$S_3(y_e)$	
		x_n	$S_4(y_3)$	Для 2 –ой ветви
		$x_n x_2$	$S_7(y_5)$	Для 3 –ей ветви
		$x_n \bar{x}_2$	$S_6(y_6)$	
2	S_1	1	$S_k^1(y_k^1)$	Для 1 –ой ветви
3	S_2	1	$S_k^1(y_k^1)$	
4	S_3	x_3	$S_1(y_1)$	
		$\bar{x}_3$	$S_3(y_e)$	
5	S_4	x_5	$S_5(y_4)$	Для 2 –ой ветви
		$\bar{x}_5$	$S_k^2(y_k^2)$	
6	S_5	1	$S_k^2(y_k^2)$	

Продолжение таблицы 4.7

Шаг алгоритма	Совокупность Исходных событий $R_i(t)$	Частный входной сигнал $x_{i,j}(t)$	Событие перехода $S_j(t+1)(y_j)$	Примечание
7	S_7	x_4	$S_k^3(y_k^3)$	Для 3 –ей ветви
		$\bar{x}_4 x_2$	$S_7(y_5)$	
		$\bar{x}_4 \bar{x}_2$	$S_6(y_6)$	
8	S_6	$\bar{x}_1$	$S_6(y_6)$	
		x_1	$S_k^3(y_k^3)$	
9	$S_B = S_k^1 S_k^2 S_k^3$	1	$S_8(y_k)$	Выход за соединитель
10	$S_k^1 \bar{S}_B$	1	$S_k^1(y_k^1)$	Сохранение событий S_k^1, S_k^2, S_k^3
11	$S_k^2 \bar{S}_B$	1	$S_k^2(y_k^2)$	
12	$S_k^3 \bar{S}_B$	1	$S_k^3(y_k^3)$	
13	S_8	1	$S_0(y_0)$	Возврат в исходное событие

В таблице 4.7 строки с 1-ой по 8-ую представляют собой детерминированные ПТП для трех параллельных ветвей, которые строятся также как и ПТП для простых ГСА, строки с 10-ой по 12-ую в соответствии с заданным типом соединителя формализуют условия сохранения событий типа S_k^i (коррекция событий типа S_k^i), а строка 9-ая в соответствии с заданным типом соединителя формализует выход алгоритмического процесса за вершину объединения параллельных ветвей.

Недетерминированная скорректированная СКУ, соответствующая ПТП (табл.4.7), будет иметь вид:

$$\begin{aligned}
S_1^{y_1}(t+1) &= S_0 x_n \bar{x}_1 x_3 \vee S_3 x_3, \\
S_2^{y_2}(t+1) &= S_0 x_n x_1, \\
S_3^{y_e}(t+1) &= S_0 x_n \bar{x}_1 \bar{x}_3 \vee S_3 \bar{x}_3, \\
S_k^1 y_k^1(t+1) &= (S_0 \vee S_3) \vee S_k^1 \bar{S}_B, \\
S_4^{y_3}(t+1) &= S_0 x_n, \\
S_5^{y_4}(t+1) &= S_4 x_5, \\
S_k^2 y_k^2(t+1) &= S_5 \vee S_4 \bar{x}_5 \vee S_k^2 \bar{S}_B, \\
S_6^{y_6}(t+1) &= S_0 x_n \bar{x}_2 \vee S_7 \bar{x}_4 \bar{x}_2 \vee S_6 \bar{x}_1, \\
S_7^{y_5}(t+1) &= S_0 x_n x_2 \vee S_7 \bar{x}_4 x_2, \\
S_k^3 y_k^3(t+1) &= S_7 x_4 \vee S_6 x_1 \vee S_k^3 \bar{S}_B, \\
S_8^{y_k}(t+1) &= S_B = S_k^1 S_k^2 S_k^3, \\
S_0^{y_0}(t+1) &= x_0 \vee S_8 \vee S_0 \bar{x}_n.
\end{aligned} \tag{4.69}$$

На основании табл.4.7 и алгоритма детерминизации (глава 2) ПТП детерминированного автомата будет представлена табл.4.8.

Таблица 4.8.

Шаг алгоритма	Сочетание частных исходных событий полное событие $a_m(t)(Y_m)$	Сочетание частных входных сигналов на переходе $X(a_m, a_s)(t)$	Сочетание частных событий в момент времени $(t+1)$ $a_s(t+1)$
1	$\frac{S_0}{a_0(y_0)}$	$\bar{x}_n$	S_0 / a_0
		$x_n x_1 x_2$	$S_2 S_4 S_7 / a_1$
		$x_n x_1 \bar{x}_2$	$S_2 S_4 S_6 / a_2$
		$x_n \bar{x}_1 x_2 x_3$	$S_1 S_4 S_7 / a_3$
		$x_n \bar{x}_1 \bar{x}_2 x_3$	$S_1 S_4 S_6 / a_4$
		$x_n \bar{x}_1 x_2 \bar{x}_3$	$S_3 S_4 S_7 / a_5$
		$x_n \bar{x}_1 \bar{x}_2 \bar{x}_3$	$S_3 S_4 S_6 / a_6$

Шаг алго- рит- ма	Сочетание частных исходных событий полное событие $a_m(t)(Y_m)$	Сочетание частных входных сигналов на переходе $X(a_m, a_s)(t)$	Сочетание частных событий в момент времени $(t+1)$ $a_s(t+1)$
2	$\frac{S_2 S_4 S_7}{a_1(y_2 y_3 y_5)}$	$x_4 x_5$	$S_k^1 S_5 S_k^3 / a_7$
		$x_4 \bar{x}_5$	$S_k^1 S_k^2 S_k^3 / a_8$
		$\bar{x}_4 x_2 x_5$	$S_k^1 S_5 S_7 / a_9$
		$\bar{x}_4 x_2 \bar{x}_5$	$S_k^1 S_k^2 S_7 / a_{10}$
		$\bar{x}_4 \bar{x}_2 x_5$	$S_k^1 S_5 S_6 / a_{11}$
		$\bar{x}_4 \bar{x}_2 \bar{x}_5$	$S_k^1 S_k^2 S_6 / a_{12}$
3	$\frac{S_2 S_4 S_6}{a_2(y_2 y_3 y_6)}$	$\bar{x}_1 \bar{x}_5$	$S_k^1 S_k^2 S_6 / a_{12}$
		$\bar{x}_1 x_5$	$S_k^1 S_5 S_6 / a_{11}$
		$x_1 \bar{x}_5$	$S_k^1 S_k^2 S_k^3 / a_8$
		$x_1 x_5$	$S_k^1 S_5 S_k^3 / a_7$
4	$\frac{S_1 S_4 S_7}{a_3(y_1 y_3 y_5)}$	$x_4 x_5$	$S_k^1 S_5 S_k^3 / a_7$
		$x_4 \bar{x}_5$	$S_k^1 S_k^2 S_k^3 / a_8$
		$\bar{x}_4 x_2 x_5$	$S_k^1 S_5 S_7 / a_9$
		$\bar{x}_4 x_2 \bar{x}_5$	$S_k^1 S_k^2 S_7 / a_{10}$
		$\bar{x}_4 \bar{x}_2 x_5$	$S_k^1 S_5 S_6 / a_{11}$
		$\bar{x}_4 \bar{x}_2 \bar{x}_5$	$S_k^1 S_k^2 S_6 / a_{12}$
5	$\frac{S_1 S_4 S_6}{a_4(y_1 y_3 y_6)}$	$\bar{x}_1 \bar{x}_5$	$S_k^1 S_k^2 S_6 / a_{12}$
		$\bar{x}_1 x_5$	$S_k^1 S_5 S_6 / a_{11}$
		$x_1 \bar{x}_5$	$S_k^1 S_k^2 S_k^3 / a_8$
		$x_1 x_5$	$S_k^1 S_5 S_k^3 / a_7$

Шаг алго- рит- ма	Сочетание частных исходных событий полное событие $a_m(t)(Y_m)$	Сочетание частных входных сигналов на переходе $X(a_m, a_s)(t)$	Сочетание частных событий в момент времени $(t+1)$ $a_s(t+1)$
6	$\frac{S_3 S_4 S_7}{a_5(y_e y_3 y_5)}$	$x_4 \bar{x}_3 \bar{x}_5$	$S_3 S_k^2 S_k^3 / a_{13}$
		$x_4 \bar{x}_3 x_5$	$S_3 S_5 S_k^3 / a_{14}$
		$x_4 x_3 \bar{x}_5$	$S_1 S_k^2 S_k^3 / a_{15}$
		$x_4 x_3 x_5$	$S_1 S_5 S_k^3 / a_{16}$
		$\bar{x}_4 x_2 \bar{x}_3 \bar{x}_5$	$S_3 S_k^2 S_7 / a_{17}$
		$\bar{x}_4 x_2 \bar{x}_3 x_5$	$S_3 S_5 S_7 / a_{18}$
		$\bar{x}_4 x_2 x_3 \bar{x}_5$	$S_1 S_k^2 S_7 / a_{19}$
		$\bar{x}_4 x_2 x_3 x_5$	$S_1 S_5 S_7 / a_{20}$
		$\bar{x}_4 \bar{x}_2 \bar{x}_3 \bar{x}_5$	$S_3 S_k^2 S_6 / a_{21}$
		$\bar{x}_4 \bar{x}_2 \bar{x}_3 x_5$	$S_3 S_5 S_6 / a_{22}$
		$\bar{x}_4 \bar{x}_2 x_3 \bar{x}_5$	$S_1 S_k^2 S_6 / a_{23}$
		$\bar{x}_4 \bar{x}_2 x_3 x_5$	$S_1 S_5 S_6 / a_{24}$
7	$\frac{S_3 S_4 S_6}{a_6(y_e y_3 y_6)}$	$\bar{x}_1 \bar{x}_3 \bar{x}_5$	$S_3 S_k^2 S_6 / a_{21}$
		$\bar{x}_1 \bar{x}_3 x_5$	$S_3 S_5 S_6 / a_{22}$
		$\bar{x}_1 x_3 \bar{x}_5$	$S_1 S_k^2 S_6 / a_{23}$
		$\bar{x}_1 x_3 x_5$	$S_1 S_5 S_6 / a_{24}$
		$x_1 \bar{x}_3 \bar{x}_5$	$S_3 S_k^2 S_k^3 / a_{13}$
		$x_1 \bar{x}_3 x_5$	$S_3 S_5 S_k^3 / a_{14}$
		$x_1 x_3 \bar{x}_5$	$S_1 S_k^2 S_k^3 / a_{15}$
		$x_1 x_3 x_5$	$S_1 S_5 S_k^3 / a_{16}$
8	$S_k^1 S_5 S_k^3$	1	$S_k^1 S_k^2 S_k^3 / a_8$
9	$S_k^1 S_k^2 S_k^3$	1	S_8 / a_{25}

Продолжение таблицы 4.8

Шаг алгоритма	Сочетание частных исходных событий полное событие $a_m(t)(Y_m)$	Сочетание частных входных сигналов на переходе $X(a_m, a_s)(t)$	Сочетание частных событий в момент времени $(t+1)$ $a_s(t+1)$
10	$\frac{S_k^1 S_5 S_7}{a_9(y_k^1 y_4 y_5)}$	x_4	$S_k^1 S_k^2 S_k^3 / a_8$
		$\bar{x}_4 x_2$	$S_k^1 S_k^2 S_7 / a_{10}$
		$\bar{x}_4 \bar{x}_2$	$S_k^1 S_k^2 S_6 / a_{12}$
11	$\frac{S_k^1 S_k^2 S_7}{a_{10}(y_k^1 y_k^2 y_5)}$	x_4	$S_k^1 S_k^2 S_k^3 / a_8$
		$\bar{x}_4 x_2$	$S_k^1 S_k^2 S_7 / a_{10}$
		$\bar{x}_4 \bar{x}_2$	$S_k^1 S_k^2 S_6 / a_{12}$
12	$\frac{S_k^1 S_5 S_6}{a_{11}(y_k^1 y_4 y_6)}$	$\bar{x}_1$	$S_k^1 S_k^2 S_6 / a_{12}$
		x_1	$S_k^1 S_k^2 S_k^3 / a_8$
13	$\frac{S_k^1 S_k^2 S_6}{a_{12}(y_k^1 y_k^2 y_6)}$	$\bar{x}_1$	$S_k^1 S_k^2 S_6 / a_{12}$
		x_1	$S_k^1 S_k^2 S_k^3 / a_8$
14	$\frac{S_3 S_k^2 S_k^3}{a_{13}(y_e y_k^2 y_6)}$	x_3	$S_1 S_k^2 S_k^3 / a_{15}$
		$\bar{x}_3$	$S_3 S_k^2 S_k^3 / a_{13}$
15	$\frac{S_3 S_5 S_k^3}{a_{14}(y_e y_4 y_k^3)}$	x_3	$S_1 S_k^2 S_k^3 / a_{15}$
		$\bar{x}_3$	$S_3 S_k^2 S_k^3 / a_{13}$
16	$\frac{S_1 S_k^2 S_k^3}{a_{15}(y_1 y_k^2 y_k^3)}$	1	$S_k^1 S_k^2 S_k^3 / a_8$
17	$\frac{S_1 S_5 S_k^3}{a_{16}(y_1 y_4 y_k^3)}$	1	$S_k^1 S_k^2 S_k^3 / a_8$
18	$\frac{S_3 S_k^2 S_7}{a_{17}(y_e y_k^2 y_5)}$	$x_4 x_3$	$S_1 S_k^2 S_k^3 / a_{15}$
		$x_4 \bar{x}_3$	$S_3 S_k^2 S_k^3 / a_{13}$
		$\bar{x}_4 x_2 x_3$	$S_1 S_k^2 S_7 / a_{19}$
		$\bar{x}_4 x_2 \bar{x}_3$	$S_3 S_k^2 S_7 / a_{17}$
		$\bar{x}_4 \bar{x}_2 x_3$	$S_1 S_k^2 S_6 / a_{23}$
		$\bar{x}_4 \bar{x}_2 \bar{x}_3$	$S_3 S_k^2 S_6 / a_{21}$

Продолжение таблицы 4.8

Шаг алгоритма	Сочетание частных исходных событий полное событие	Сочетание частных входных сигналов на	Сочетание частных событий в момент
---------------	--	---------------------------------------	------------------------------------

ма	$a_m(t)(Y_m)$	переходе $X(a_m, a_s)(t)$	времени $(t+1)$ $a_s(t+1)$
19	$\frac{S_3 S_5 S_7}{a_{18}(y_e y_4 y_5)}$	$x_4 x_3$	$S_1 S_k^2 S_k^3 / a_{15}$
		$x_4 \bar{x}_3$	$S_3 S_k^2 S_k^3 / a_{13}$
		$\bar{x}_4 x_2 x_3$	$S_1 S_k^2 S_7 / a_{19}$
		$\bar{x}_4 x_2 \bar{x}_3$	$S_3 S_k^2 S_7 / a_{17}$
		$\bar{x}_4 \bar{x}_2 x_3$	$S_1 S_k^2 S_6 / a_{23}$
		$\bar{x}_4 \bar{x}_2 \bar{x}_3$	$S_3 S_k^2 S_6 / a_{21}$
20	$\frac{S_1 S_k^2 S_7}{a_{19}(y_e y_k^2 y_5)}$	x_4	$S_k^1 S_k^2 S_k^3 / a_8$
		$\bar{x}_4 x_2$	$S_k^1 S_k^2 S_7 / a_{10}$
		$\bar{x}_4 \bar{x}_2$	$S_k^1 S_k^2 S_6 / a_{12}$
21	$\frac{S_1 S_5 S_7}{a_{20}(y_1 y_4 y_5)}$	x_4	$S_k^1 S_k^2 S_k^3 / a_8$
		$\bar{x}_4 x_2$	$S_k^1 S_k^2 S_7 / a_{10}$
		$\bar{x}_4 \bar{x}_2$	$S_k^1 S_k^2 S_6 / a_{12}$
22	$\frac{S_3 S_k^2 S_6}{a_{21}(y_e y_k^2 y_6)}$	$\bar{x}_1 \bar{x}_3$	$S_3 S_k^2 S_6 / a_{21}$
		$\bar{x}_1 x_3$	$S_1 S_k^2 S_6 / a_{23}$
		$x_1 \bar{x}_3$	$S_3 S_k^2 S_k^3 / a_{13}$
		$x_1 x_3$	$S_1 S_k^2 S_k^3 / a_{15}$
23	$\frac{S_3 S_5 S_6}{a_{22}(y_e y_4 y_6)}$	$\bar{x}_1 \bar{x}_3$	$S_3 S_k^2 S_6 / a_{21}$
		$\bar{x}_1 x_3$	$S_1 S_k^2 S_6 / a_{23}$
		$x_1 \bar{x}_3$	$S_3 S_k^2 S_k^3 / a_{13}$
		$x_1 x_3$	$S_1 S_k^2 S_k^3 / a_{15}$
24	$\frac{S_1 S_k^2 S_6}{a_{23}(y_1 y_k^2 y_6)}$	$\bar{x}_1$	$S_k^1 S_k^2 S_6 / a_{12}$
		x_1	$S_k^1 S_k^2 S_k^3 / a_8$

Шаг алгоритма	Сочетание частных исходных событий полное событие $a_m(t)(Y_m)$	Сочетание частных входных сигналов на переходе $X(a_m, a_s)(t)$	Сочетание частных событий в момент времени $(t+1)$ $a_s(t+1)$
25	$\frac{S_1 S_5 S_7}{a_{24}(y_1 y_4 y_6)}$	$\bar{x}_1$	$S_k^1 S_k^2 S_6 / a_{12}$
		x_1	$S_k^1 S_k^2 S_k^3 / a_8$
26	$\frac{S_8}{a_{25}(y_k)}$	1	S_0 / a_0

Как видно из ПТП (табл. 4.8) каждое состояние детерминированного автомата, кроме начального и заключительного, представлено совокупностью из трех событий, каждое из которых реализуется одной из параллельных ветвей алгоритма управления.

Искомая детерминированная СКУ управляющего автомата Мура, полученная на основании ПТП (табл. 4.8), представлена системой уравнений (4.70):

$$\begin{aligned}
a_0^{y_0}(t+1) &= x_0 \vee a_{25} \vee a_0 \bar{x}_n, \\
a_1^{y_2 y_3 y_5}(t+1) &= a_0 x_n x_1 x_2, \\
a_2^{y_2 y_3 y_6}(t+1) &= a_0 x_n x_1 \bar{x}_2, \\
a_3^{y_1 y_3 y_5}(t+1) &= a_0 x_n \bar{x}_1 x_2 x_3, \\
a_4^{y_1 y_3 y_6}(t+1) &= a_0 x_n \bar{x}_1 \bar{x}_2 x_3, \\
a_5^{y_e y_3 y_5}(t+1) &= a_0 x_n \bar{x}_1 \bar{x}_2 x_3, \\
a_6^{y_e y_3 y_6}(t+1) &= a_0 x_n \bar{x}_1 \bar{x}_2 \bar{x}_3, \\
a_7^{y_k^1 y_4 y_k^3}(t+1) &= (a_1 \vee a_3) x_4 x_5 \vee (a_2 \vee a_4) x_1 x_5, \\
a_8^{y_k^1 y_k^2 y_k^3}(t+1) &= (a_1 \vee a_3) x_4 \bar{x}_5 \vee (a_2 \vee a_4) x_1 \bar{x}_5 \vee (a_9 \vee a_{10} \vee a_{19} \vee a_{20}) \bar{x}_4 \vee \\
&\vee (a_{11} \vee a_{21} \vee a_{22} \vee a_{24}) x_1 \vee (a_7 \vee a_{15} \vee a_{16}), \\
a_9^{y_k^1 y_4 y_5}(t+1) &= (a_1 \vee a_3) x_2 \bar{x}_4 x_5, \\
a_{10}^{y_k^1 y_k^2 y_5}(t+1) &= (a_1 \vee a_3) x_2 \bar{x}_4 \bar{x}_5 \vee (a_2 \vee a_9 \vee a_{10} \vee a_{19} \vee a_{20}) x_2 \bar{x}_4, \\
a_{11}^{y_k^1 y_4 y_6}(t+1) &= (a_1 \vee a_3) \bar{x}_2 \bar{x}_4 x_5 \vee (a_2 \vee a_4) \bar{x}_1 x_5,
\end{aligned} \tag{4.70}$$

$$\begin{aligned}
a_{12}^{y_k^1 y_k^2 y_6} (t+1) &= (a_1 \vee a_3) \bar{x}_2 \bar{x}_4 \bar{x}_5 \vee (a_2 \vee a_4) \bar{x}_1 \bar{x}_5 \vee \\
&\vee (a_2 \vee a_9 \vee a_{10} \vee a_{19} \vee a_{20}) x_2 \bar{x}_4 \vee (a_{11} \vee a_{12} \vee a_{23} \vee a_{24}) \bar{x}_1, \\
a_{13}^{y_e y_k^2 y_k^3} (t+1) &= a_5 \bar{x}_3 x_4 \bar{x}_5 \vee a_6 x_1 \bar{x}_3 \bar{x}_5 \vee (a_{13} \vee a_{14}) \bar{x}_3 \vee (a_{17} \vee a_{18}) \bar{x}_3 x_4 \\
&\vee (a_{21} \vee a_{22}) x_1 \bar{x}_3, \\
a_{14}^{y_e y_4 y_k^3} (t+1) &= a_5 \bar{x}_3 x_4 x_5 \vee a_6 x_1 \bar{x}_3 x_5, \\
a_{15}^{y_1 y_k^2 y_k^3} (t+1) &= a_5 x_3 x_4 \bar{x}_5 \vee a_6 x_1 x_3 \bar{x}_5 \vee (a_{13} \vee a_{14}) x_3 \vee (a_{17} \vee a_{18}) x_3 x_4 \\
&\vee (a_{21} \vee a_{22}) x_1 x_3, \\
a_{16}^{y_1 y_4 y_k^3} (t+1) &= (a_5 x_4 \vee a_6 x_1) x_3 x_5, \\
a_{17}^{y_e y_k^2 y_5} (t+1) &= a_5 x_2 \bar{x}_3 \bar{x}_4 \bar{x}_5 \vee (a_{17} \vee a_{18}) x_2 \bar{x}_3 \bar{x}_4, \\
a_{18}^{y_e y_4 y_5} (t+1) &= a_5 x_2 \bar{x}_3 \bar{x}_4 x_5, \\
a_{19}^{y_1 y_k^2 y_5} (t+1) &= a_5 x_2 x_3 \bar{x}_4 \bar{x}_5 \vee (a_{17} \vee a_{18}) x_2 x_3 \bar{x}_4, \\
a_{20}^{y_1 y_4 y_5} (t+1) &= a_5 x_2 x_3 \bar{x}_4 x_5, \\
a_{21}^{y_e y_k^2 y_6} (t+1) &= a_5 \bar{x}_2 \bar{x}_3 \bar{x}_4 \bar{x}_5 \vee a_6 \bar{x}_1 \bar{x}_3 \bar{x}_5 \vee (a_{17} \vee a_{18}) \bar{x}_2 x_3 \bar{x}_4 \vee (a_{21} \vee a_{22}) \bar{x}_1 x_3, \\
a_{22}^{y_e y_4 y_6} (t+1) &= a_5 \bar{x}_2 \bar{x}_3 \bar{x}_4 x_5 \vee a_6 \bar{x}_1 \bar{x}_3 x_5, \\
a_{23}^{y_1 y_k^1 y_6} (t+1) &= a_5 \bar{x}_2 x_3 \bar{x}_4 \bar{x}_5 \vee a_6 \bar{x}_1 x_3 \bar{x}_5 \vee (a_{17} \vee a_{18}) \bar{x}_2 x_3 \bar{x}_4 \vee (a_{21} \vee a_{22}) \bar{x}_1 x_3, \\
a_{24}^{y_1 y_4 y_6} (t+1) &= a_5 \bar{x}_2 x_3 \bar{x}_4 x_5 \vee a_6 \bar{x}_1 x_3 x_5, \\
a_{25}^{y_k} (t+1) &= a_8.
\end{aligned}$$

Пр и м е р 4.9. Для управляющего автомата, заданного на языке ГСАП (рис.4.6) построить недетерминированную ПТП и соответствующую ей недетерминированную СКУ автомата Мура при условии, что в качестве соединителя параллельных ветвей используется комбинированный соединитель, для которого $S_B = S_k^1 \vee S_k^2 S_k^3$.

Для данного типа комбинированного соединителя НД ПТП с учетом требований однозначности управляющего алгоритма будет представлена табл.4.9, которая отличается от НД ПТП предыдущего примера только в следующем: в строках с 2-ой по 8-ую все обозначения частных событий в колонке 2 умножаются на $\bar{S}_B$; условия сохранения для событий типа S_k^i предусматриваются только для событий S_k^2 и S_k^3 ; событие S_k^1 не требуется сохранять после его появления, т.к. для истинного S_k^1 одновременно будет истинным и S_B .

Таблица 4.9.

Шаг	Совокупность	Частный	Событие	Примечание
-----	--------------	---------	---------	------------

алго- ритма	Исходных событий $R_i(t)$	входной сигнал $x_{i,j}(t)$	перехода $S_j(t+1)(y_j)$	
1	S_0	$\bar{x}_n$	$S_0(y_0)$	Для 1 –ой ветви
		$x_n x_1$	$S_2(y_2)$	
		$x_n \bar{x}_1 x_3$	$S_1(y_1)$	
		$x_n \bar{x}_1 \bar{x}_3$	$S_3(y_e)$	
		x_n	$S_4(y_3)$	Для 2 –ой
		$x_n x_2$	$S_7(y_5)$	Для 3 –ей ветви
		$x_n \bar{x}_2$	$S_6(y_6)$	
2	$S_2(\bar{S}_B)$	1	$S_k^1(y_k^1)$	Для 1 –ой ветви с учетом коррекции
3	$S_2(\bar{S}_B)$	1	$S_k^1(y_k^1)$	
4	$S_3(\bar{S}_B)$	x_3	$S_1(y_1)$	
		$\bar{x}_3$	$S_3(y_e)$	
5	$S_4(\bar{S}_B)$	x_5	$S_5(y_4)$	Для 2 –ой ветви с учетом коррекции
		$\bar{x}_5$	$S_k^2(y_k^2)$	
6	$S_5(\bar{S}_B)$	1	$S_k^2(y_k^2)$	
7	$S_7(\bar{S}_B)$	x_4	$S_k^3(y_k^3)$	Для 3 –ей ветви с учетом коррекции
		$\bar{x}_4 x_2$	$S_7(y_5)$	
		$\bar{x}_4 \bar{x}_2$	$S_6(y_6)$	
8	$S_6(\bar{S}_B)$	x_1	$S_6(y_6)$	
		$\bar{x}_1$	$S_k^3(y_k^3)$	

Шаг алго- ритма	Совокупность Исходных событий $R_i(t)$	Частный входной сигнал $x_{i,j}(t)$	Событие перехода $S_j(t+1)(y_j)$	Примечание
9	$S_B = S_k^1 \vee S_k^2 S_k^3$	1	$S_8(y_k)$	Выход за соединитель
10	$S_k^2 \bar{S}_B$	1	$S_k^2(y_k^2)$	Сохранение событий $S_k^2 S_k^3$
11	$S_k^3 \bar{S}_B$	1	$S_k^3(y_k^3)$	
12	S_8	1	$S_0(y_0)$	Возврат в исходное событие

Искомая НД СКУ управляющего автомата Мура, построенная по НД ПТП (табл.4.9) представлена системой уравнений (4.71):

$$\begin{aligned}
 S_1^{y_1}(t+1) &= S_0 x_n \bar{x}_1 x_3 \vee S_3(\bar{S}_B) x_3, \\
 S_2^{y_2}(t+1) &= S_0 x_n x_1, \\
 S_3^{y_e}(t+1) &= S_0 x_n \bar{x}_1 \bar{x}_3 \vee S_3(\bar{S}_B) \bar{x}_3, \\
 S_k^{y_k^1}(t+1) &= (S_0 \vee S_3)(\bar{S}_B), \\
 S_4^{y_3}(t+1) &= S_0 x_n, \\
 S_5^{y_4}(t+1) &= S_4 x_5, \\
 S_k^{y_k^2}(t+1) &= S_5(\bar{S}_B) \vee S_4(\bar{S}_B) \bar{x}_5 \vee S_k^2(\bar{S}_B), \\
 S_6^{y_6}(t+1) &= S_0 x_n \bar{x}_2 \vee S_7(\bar{S}_B) \bar{x}_4 \bar{x}_2 \vee S_6(\bar{S}_B) \bar{x}_1, \\
 S_7^{y_5}(t+1) &= S_0 x_n x_2 \vee S_7(\bar{S}_B) \bar{x}_4 x_2, \\
 S_k^{y_k^3}(t+1) &= S_7(\bar{S}_B) x_4 \vee S_6(\bar{S}_B) x_1 \vee S_k^3(\bar{S}_B), \\
 S_8^{y_k}(t+1) &= S_B = S_k^1 \vee S_k^2 S_k^3, \\
 S_0^{y_0}(t+1) &= x_0 \vee S_8 \vee S_0 \bar{x}_n.
 \end{aligned} \tag{4.71}$$

Контрольные вопросы к главе 4.

1. Как определяется событие на языке РВАС?
2. Какие события называются регулярными и какова их характерная особенность?
3. Как формируется правило раскрытия итерационных скобок?
4. На чем основана методика преобразования систем событий, представленных на языке РВАС, в стандартную систему канонических уравнений?
5. Как может быть представлено описание произвольного регулярного выражения алгебры событий на языке исчисления предикатов первого порядка с ограниченными кванторами? Какие составные части включает это описание?
6. Какое событие понимается как событие, непосредственно предшествующее исходному событию?
7. Как выражается закон нейтральности пустого слова и как этот закон формализуется аксиомами в исчислении предикатов первого порядка?
8. Каким образом можно преобразовать описание событий, представленных на языке исчисления предикатов первого порядка, в систему канонических уравнений в бескванторной предикатной форме?
9. В чем заключается методика построения систем канонических уравнений для событий, реализуемых в алгоритмах управления, представленных на языках ГСА и ЛСА? Что понимается под шагом работы алгоритма управления, как определяются частные входные сигналы, как нумеруются операторы алгоритма управления, как решаются вопросы с циклами из условных вершин (логических условий)?
10. Какие достоинства языка ГСАП по сравнению с параллельными языками?
11. Как формализуются на языке ГСАП условия выхода параллельных процессов за вершину объединения ветвей?
12. В каких случаях и как осуществляется коррекция событий, реализуемых в параллельных ветвях алгоритма управления, представленного на языке ГСАП?
13. Из каких основных этапов состоит алгоритм построения детерминированной ПТП для управляющего алгоритма, заданного на языке ГСАП?

Глава 5. Структурный синтез систем микропрограммного управления (МПУ), заданных моделью НДА

Если исходный алгоритм работы управляющего автомата (УА) представлен моделью НДА, то после его детерминизации структуру УА можно построить на основе использования известных классических методов структурного синтеза детерминированных конечных автоматов. В соответствии с этим методом выполняют два основных этапа структурного синтеза УА: кодирование внутренних состояний УА, представленных в процессе детерминизации в виде сочетаний частных S -событий НДА, и построение функций возбуждения и функций выходов для принятой системы элементов памяти УА. Использование такого классического метода структурного синтеза УА приводит к тому, что исходная информация об алгоритме управления, представленная в компактной форме в виде НД SKU для S -событий, дважды преобразуется: первый раз при выполнении детерминизации и второй раз при кодировании внутренних состояний УА, полученных в результате детерминизации. Такое двойное преобразование исходной НД SKU приводит к следующим негативным явлениям:

а) «Теряется» исходная информация о реализуемом алгоритме управления, представленном в результате преобразований в виде SKU для Q -событий, что затрудняет впоследствии выполнение контрольных и диагностических процедур на различных этапах как при проектировании УА, так и при его изготовлении и эксплуатации.

б) Функции возбуждения элементов памяти, представленные в виде SKU для Q -событий, сильно усложняются по сравнению с исходной НД SKU для S -событий. Это объясняется тем обстоятельством, что сложность SKU для Q -событий, полученных после детерминизации исходного алгоритма управления, значительно увеличивается по сравнению со сложностью исходной SKU для S -событий. Например, для УА, заданного ГСАП (рис.4.6), сложность булевых функций, определяющих SKU для Q -событий, примерно в 5 раз выше сложности булевых функций, представляющих исходную SKU для S -событий, если выполнить оценку сложности булевых функций по Квайну.

Учитывая отмеченные выше недостатки использования классического метода синтеза для построения структуры УА, заданного моделью НДА, в данном разделе будут рассмотрены другие подходы и методы построения структур УА, заданных моделью НДА, позволяющие в той или иной степени не только избежать отмеченных выше недостатков использования классического метода синтеза структур УА, заданных моделью НДА, но и развить новые более эффективные методы, которые вытекают (следуют) из достоинств представления управляющих алгоритмов моделью НДА. К числу

таких методов структурного синтеза УА, заданных моделью НДА, относятся методы, базирующиеся на использовании унитарного способа кодирования частных S -событий, входящих в исходную НД СКУ, а также методы, основанные на разбиении всех частных S -событий, реализуемых в УА, на группы несовместимых событий, каждая из которых может быть реализована отдельным подавтоматом. Такой подход к методу синтеза структуры УА позволяет реализовать операционные устройства, выполняющие распределенную параллельную обработку информации.

В заключение отметим, что рассматриваемые в данном разделе методы преобразования алгоритмов управления, заданных моделью НДА, могут быть использованы не только для построения структуры систем управления преобразованием информации на микропрограммном уровне, но и могут быть использованы для построения высокопроизводительных систем управления преобразованием информации для различных распределенных и параллельных вычислительных систем.

5.1. Структурная реализация систем МПУ на основе разбиения частных событий на группы несовместимых событий

5.1.1. Преобразование структуры управляющего алгоритма, представленного моделью НДА, для построения распределенной системы МПУ параллельной обработки

В том случае, если исходный управляющий алгоритм, заданный моделью НДА, может быть разбит на группы несовместимых событий, то его обобщенную структуру можно представить на основе языка ГСАП в виде графа, включающего последовательную и параллельную компоненты. При этом последовательная компонента будет состоять из двух частей: начальной и заключительной (рис. 5.1).

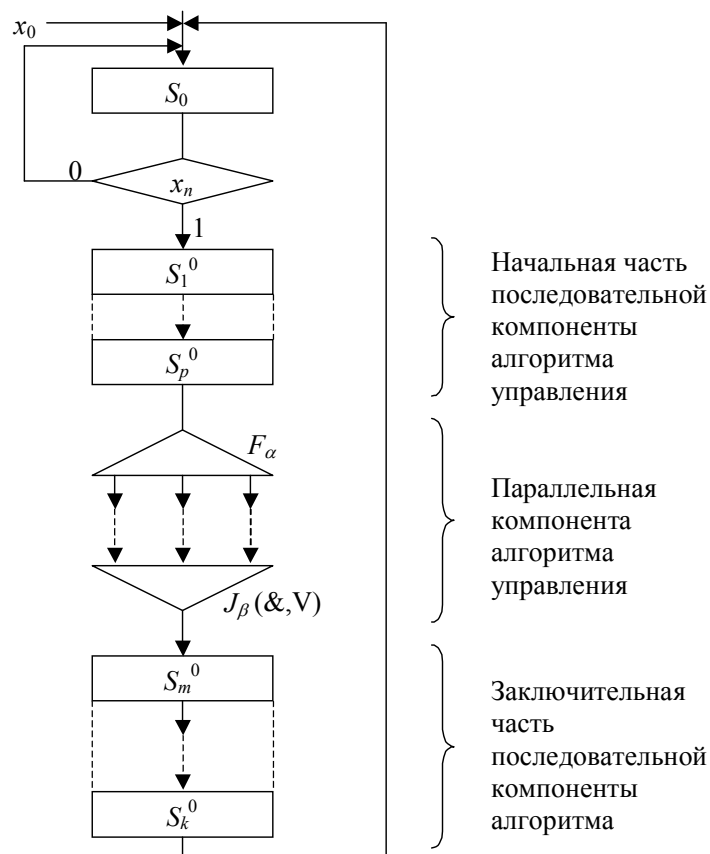


Рис. 5.1. Обобщенная структура ГСАП управляющего алгоритма с параллельными ветвями

На рис. 5.1 введены следующие обозначения:

S_0 - начальное событие;

x_0 - сигнал приведения УА в начальное состояние;

x_n - пусковой сигнал,

S_1^0 - первое событие начальной части последовательной компоненты алгоритма управления;

событие,

S_p^0 - событие, определяющее начало распараллеливания алгоритмического процесса;

S_m^0 - событие, свидетельствующее о выходе алгоритмического процесса за вершину объединения параллельных ветвей (первое событие заключительной части последовательной компоненты алгоритма управления);

S_k^0 - заключительное событие алгоритма управления.

Из графического представления управляющего алгоритма (рис. 5.1) следует, что все частные S -события, представляющие управляющий алгоритм, в явном виде разбиты на группы несовместимых событий. Поэтому структуру УА можно представить в виде композиции из нескольких подавтоматов (п/А), каждый из которых реализует одну из групп несовместимых событий.

Практически можно отметить два варианта исходных управляющих алгоритмов, представленных моделью НДА, для которых все частные события можно разбить на группы несовместимых событий. Для первого варианта все частные события, представленные в исходном управляющем алгоритме, в явном виде разбиты на группы несовместимых частных событий, что соответствует представлению исходного управляющего алгоритма на языке ГСАП.

Для второго варианта, в отличие от первого, частные события, представленные в исходном управляющем алгоритме в явном виде не разбиты на группы несовместимых событий, и для того чтобы разбить их на такие группы необходимо выполнить над моделью НДА управляющего алгоритма некоторые процедуры. К числу таких процедур относятся: детерминизация НДА, представляющего исходный управляющий алгоритм и построение матрицы совместимости частных событий и матрицы включения, на основе которых могут быть получены группы несовместимых частных событий. Такие процедуры над моделью НДА управляющего алгоритма будут рассмотрены в последующем разделе.

В данном разделе рассмотрим методику преобразования исходного управляющего алгоритма, представленного в виде графа рис. 5.1, в соответствии с которой структуру УА можно представить в виде композиции из нескольких п/А, каждый из которых реализует одну из групп несовместимых частных событий.

В дальнейшем будем рассматривать структуру УА, состоящей из n параллельно работающих п/А, названных в пособии рабочими п/А, каждый из которых реализует одну из параллельных ветвей алгоритма управления и одного п/А, названного главным п/А, который реализует последовательную компоненту алгоритма управления и осуществляет взаимодействие его параллельных ветвей.

В связи с тем, что параллельные ветви алгоритма управления реализуются отдельными п/А независимо друг от друга, то в каждую i -ую ветвь алгоритма управления вводится дополнительная операторная вершина (событие S_0^i), которая будет играть роль начальной вершины в i -ой ветви. Тогда ГСА для любого i -го рабочего п/А будет иметь вид (рис. 5.2).

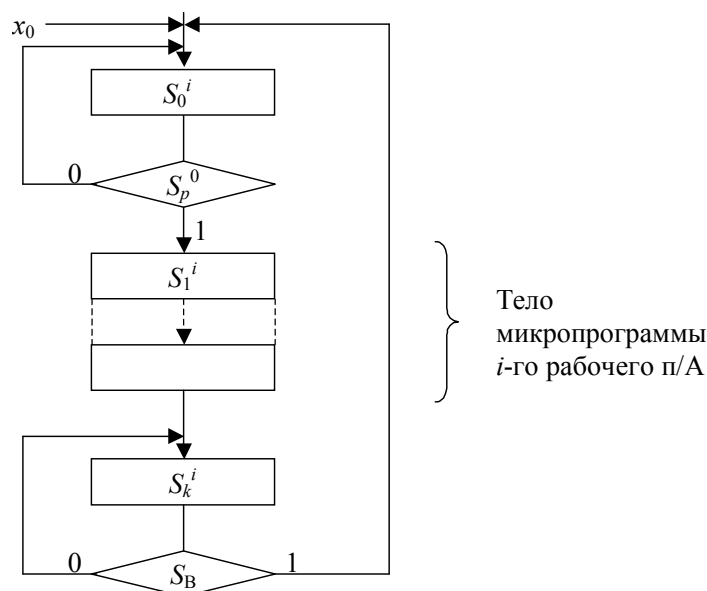


Рис. 5.2. ГСА i -го рабочего п/А, реализующего i -ую ветвь алгоритма управления

Основные события, реализуемые i -м рабочим п/А, определяющие начало и конец работы алгоритма управления для i -ой ветви, определяются следующим образом:

$$\begin{aligned}
 S_0^i(t+1) &= x_0 \vee S_k^i S_B \vee S_0^i \bar{S}_p^0, \\
 S_1^i(t+1) &= (S_0^i S_p^0) S_{1,3}^i \vee S_1^i S_{1,c}^i, \\
 S_k^i(t+1) &= S_{k,3}^i \vee S_k^i \bar{S}_B
 \end{aligned} \tag{5.1}$$

где S_k^i - заключительное событие для i -го рабочего п/А;

S_B - комбинационное событие, определяющее условие выхода алгоритмического процесса за вершину объединения параллельных ветвей;

S_1^i - первое событие микропрограммы работы i -го п/А;

$S_{1,3}^i$ и $S_{1,c}^i$ - события, определяющие зарождение и сохранение события S_1^i соответственно;

$S_{k,3}^i$ - событие, определяющее зарождение события S_k^i .

ГСА для главного п/А, реализующего последовательную часть алгоритма управления, будет иметь следующий вид (рис. 5.3).



Рис. 5.3. ГСА главного п/А, реализующего начальную и заключительную части последовательной компоненты алгоритма управления

Основные события, реализуемые главным п/А, определяющие начало и конец работы алгоритма управления для его последовательной компоненты, будут иметь вид:

$$\begin{aligned}
 S_0(t+1) &= x_0 \vee S_k^0 \vee \bar{x}_n S_0, \\
 S_1^0(t+1) &= (S_0 x_n) S_{1,3}^0 \vee S_1^0 S_{1,c}^0, \\
 S_r^0(t+1) &= (S_p^0 \vee S_r) \bar{S}_B, \\
 S_m^0(t+1) &= (S_r S_B) S_{m,3}^0 \vee S_m^0 S_{m,c}^0
 \end{aligned} \tag{5.2}$$

где S_r^0 - событие, символизирующее ожидание условия выхода алгоритмического процесса за вершину объединения параллельных ветвей;

$S_{m,3}^0$ и $S_{m,c}^0$ - события, определяющие зарождение и сохранение событий S_m^0 соответственно;

y_e - пустой выходной сигнал.

В том случае, когда выполнение функций главного п/А возлагаются на один из рабочих п/А, то ГСА для такого п/А примет следующий вид (рис. 5.4).

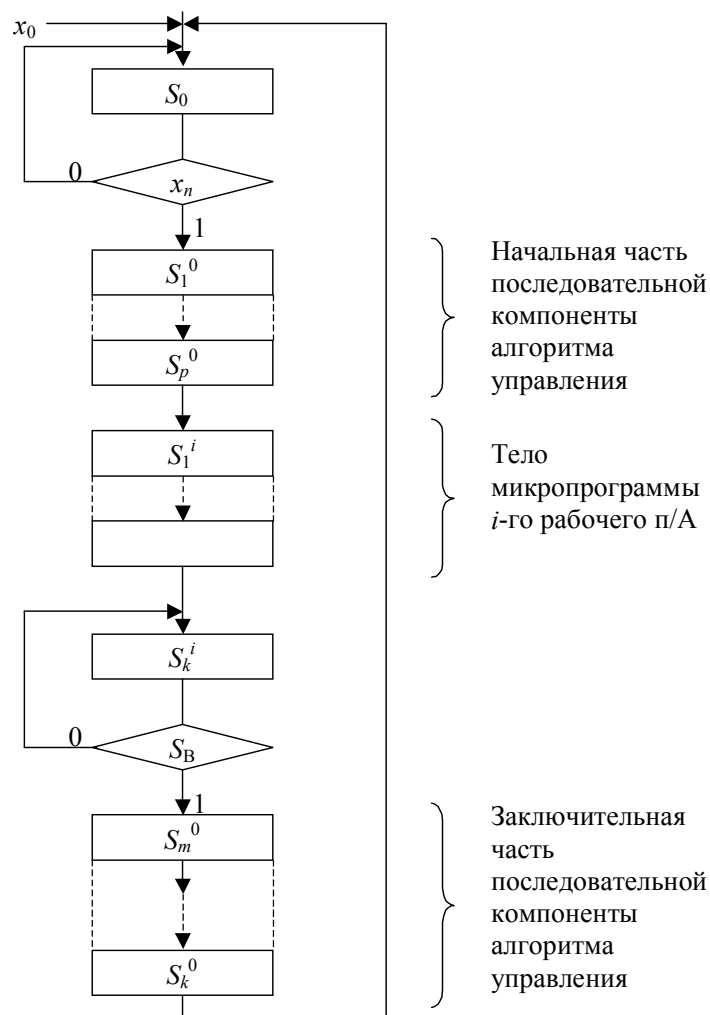


Рис. 5.4. ГСА рабочего i -го п/А, выполняющего функции главного п/А

Построив по исходной НД СКУ и по ГСА (рис. 5.2, 5.3, 5.4) скорректированные СКУ для S -событий для всех п/А, каждый из которых реализует одну из групп несовместимых событий, представляющих исходный управляющий алгоритм, можно приступить к непосредственно структурному синтезу п/А. Пример реализации таких подавтоматов системы микропрограммного управления (МПУ) будет рассмотрен в последующем разделе 5.2.3 .

5.1.2. Методика разбиения частных событий на группы несовместимых событий для управляющего алгоритма, заданного моделью НДА

В том случае, если из исходного управляющего алгоритма, заданного моделью НДА, не следует в явном виде разбиение частных событий, представляющих управляющий алгоритм, на группы несовместимых событий и требуется преобразовать такой алгоритм к виду, соответствующему обобщенной структуре с параллельными ветвями (рис.5.1), то для решения этой задачи необходимо использовать специальные процедуры, позволяющие разбить частные события на группы несовместимых событий. К числу таких процедур относятся процедура детерминизации исходного управляющего алгоритма и процедура построения матрицы включения частных событий в группы несовместимых событий, которая строится на основе использования вспомогательной матрицы совместимости частных S -событий.

Как следует из главы 2 в результате детерминизации исходного управляющего алгоритма, представленного в виде НД СКУ для частных S -событий, получают все полные a -события, определенные в виде сочетаний частных S -событий, одновременное существование которых для исходного управляющего алгоритма возможно.

По полученным a -событиям, учитывая вхождение в них частных S -событий, все множество последних разбивают на группы, в каждой из которых все частные события должны быть несовместимыми. Отметим, что под несовместимыми частными событиями понимаются события, которые не входят совместно ни в одно из полных a -событий. Число таких групп несовместимых частных событий должно быть не менее максимального числа частных событий, входящих в отдельные полные a -события.

При решении задачи распределения частных событий на группы несовместимых событий необходимо руководствоваться двумя условиями:

$$\begin{aligned} m_i \cap m_j &= \emptyset, \quad i \neq j, \quad i, j = \overline{1, N} \\ m_1 \cup m_2 \cup \dots \cup m_N &= n \end{aligned} \quad (5.3)$$

где m_i - подмножество частных событий, входящих в i -ю группу несовместимых частных событий,

N - число всех групп несовместимых частных событий;

n - число всех частных S -событий, представленных в исходном управляющем алгоритме.

Первое условие (5.3) обеспечивает требование несовместимости частных событий, входящих в любые пары из N групп. Второе условие обеспечивает включение каждого из всех n частных событий в какую-либо из N групп.

Такой способ размещения частных событий, входящих в исходный управляющий алгоритм, аналогичен способу размещения кодов

микроопераций в полях операционной части микрокоманд, который основан на построении матрицы включения при использовании вспомогательной матрицы совместимости микроопераций [2, 21, 47].

В нашем случае вспомогательная матрица совместимости частных S -событий, строится на основе учета вхождения частных событий в полные a -события и будет иметь вид (рис. 5), где $\tau_{\alpha,\beta}=0$, если S_α и S_β несовместимы; $\tau_{\alpha,\beta}=1$, если S_α и S_β совместимы.

	S_1	S_2		S_β		S_n
S_1	0	$\tau_{1,2}$	...	$\tau_{1,\beta}$	...	$\tau_{1,n}$
S_2	$\tau_{2,1}$	0	...	$\tau_{2,\beta}$	...	$\tau_{2,n}$
S_β	$\tau_{\alpha,1}$	$\tau_{\alpha,2}$	...	$\tau_{\alpha,\beta}$	...	$\tau_{\alpha,n}$
S_n	$\tau_{n,1}$	$\tau_{n,2}$	...	$\tau_{n,\beta}$	...	0

Рис. 5.5. Матрица совместимости частных событий

В матрице включения (рис. 5.6) после завершения ее построения каждая i -я строка будет содержать подмножество m_i частных несовместимых событий, размещаемых в одной группе, а каждый столбец матрицы должен включать только одно частное событие, это следует из условий (5.3), где $P_{i,\beta}=1$, если $S_\beta \subset m_i$; $P_{i,\beta}=0$, если $S_\beta \not\subset m_i$.

	S_1	S_2		S_β		S_n
m_1	$P_{1,1}$	$P_{1,2}$	...	$P_{1,\beta}$	...	$P_{1,n}$
m_2	$P_{2,1}$	$P_{2,2}$	...	$P_{2,\beta}$	...	$P_{2,n}$
m_i	$P_{i,1}$	$P_{i,2}$	...	$P_{i,\beta}$	...	$P_{i,n}$
m_H	$P_{H,1}$	$P_{H,2}$	...	$P_{H,\beta}$	...	$P_{H,n}$

Рис. 5.6. Матрица включения частных событий в группы несовместимых событий

Матрица включения строится постепенно по шагам, число которых равно числу всех частных событий n . При этом для облегчения решения задачи на каждом шаге используется вспомогательная матрица совместимости частных событий (см. рис. 5.5).

На каждом шаге построения матрицы включения для очередного частного события, например, инициируемого S_β , отыскивается такое подмножество частных событий, для которого рассматриваемое частное событие S_β ни с одним из частных событий этого подмножества не совместимо. Если такое подмножество частных событий найдено, то в него включается

рассматриваемое частное событие S_β , если оно не найдено, то образуется новое подмножество частных событий, в которое включается в качестве первого элемента частное событие S_β .

Для первого шага построения матрицы включения для S_1 имеем только пустое подмножество частных событий, поэтому для S_1 назначают подмножество $m_1=[S_\beta, 0, \dots, 0]$.

Для второго шага для S_2 определяют результат операции пересечения m_1 со второй строкой матрицы совместимости, обозначенной буквой S_2 . Если $m_1 \cap S_2 = \emptyset$, то S_2 войдет в m_1 , в противном случае образуется новое подмножество $m_2=[0, S_2, 0, \dots, 0]$.

Для третьего шага для S_3 определяют результат операции пересечения m_1 и m_2 с третьей строкой матрицы совместимости $m_1 \cap S_3$ и $m_2 \cap S_3$. Если $m_1 \cap S_3 = \emptyset$ или $m_2 \cap S_3 = \emptyset$, то S_3 войдет в m_1 (или m_2 соответственно). Если результаты операции пересечения в обоих случаях не равны $\emptyset$, то образуется новое подмножество m_3 ; если же эти результаты в обоих случаях равны $\emptyset$, то возможно включение в одно из подмножеств m_1 или m_2 . Последним шагом для S_n завершится построение матрицы включения, и может быть коррекция (перераспределение) состава подмножеств несовместимых частных событий m_i , если в процессе построения матрицы включения выявилась возможность отнести очередное частное событие S_β в одно из нескольких подмножеств.

Коррекция состава подмножеств m_i позволит в дальнейшем при структурном синтезе полнее использовать кодовые группы для частных событий, входящих в подмножество m_i с учетом того, что с точки зрения минимизации оборудования наиболее выгодным является разбиение всех частных событий на равные по числу событий в подмножествах m_i .

Проиллюстрируем методику разбиения частных событий на группы несовместимых событий на примере, когда исходный управляющий алгоритм представлен НД СКУ для S -событий.

Пример 5.1. Выполнить разбиение частных событий на группы несовместимых событий для управляющего алгоритма, заданного моделью НДА на основе НД СКУ для S -событий.

В качестве примера такого управляющего алгоритма для иллюстрации методики разбиения частных событий на группы несовместимых событий будем использовать НД СКУ (4-69) из раздела 4.5.3, условно предполагая, что она получена не по ГСАП, а на основе какого-то другого начального языка, для которого управляющий алгоритм в явном виде не выявляет параллельные ветви алгоритма.

Используя результаты детерминизации исходной НД СКУ, представленные в табл. 4.8, (раздел 4.5.3), построим вспомогательную матрицу совместимости частных событий, которая будет иметь следующий вид:

	S_1	S_2	S_3	S_4	S_5	S_6	S_7	S_k^1	S_k^2	S_k^3
S_1	0	0	0	1	1	1	1	0	1	1

S_2	0	0	0	1	0	1	1	0	0	0
S_3	0	0	0	1	1	1	1	0	1	1
S_4	1	1	1	0	0	1	1	0	0	0
S_5	1	0	1	0	0	1	1	1	0	1
S_6	1	1	1	1	1	0	0	1	1	0
S_7	1	1	1	1	1	0	0	1	1	0
S_k^1	0	0	0	0	1	1	1	0	1	1
S_k^2	1	0	1	0	0	1	1	1	0	1
S_k^3	1	0	1	0	1	0	0	1	1	0

Рис. 5.7. Матрица совместимости частных событий
исходного управляющего алгоритма

На рис. 5.7 не представлены события S_0 и S_8 , которые для данного исходного управляющего алгоритма не входят ни в одну из совокупностей из трех частных событий, определяющих a -событие.

Построение матрицы включения частных событий в группы несовместимых событий будет содержать следующие шаги:

1) Для частного события S_1 подмножество частных несовместимых событий пусто, поэтому для S_1 назначаем первое подмножество m_1 , включающее пока один элемент

$$m_1 = \begin{array}{c} S_1 \quad S_2 \quad S_3 \quad S_4 \quad S_5 \quad S_6 \quad S_7 \quad S_k^1 \quad S_k^2 \quad S_k^3 \\ 1 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \end{array}$$

2) Поскольку $S_2 \cap m_1 = \emptyset$ (здесь и в дальнейшем под первой буквой операции пересечения понимается номер строки матрицы совместимости, отмеченный буквой S_2), то S_2 включается в m_1

$$m_1 = \begin{array}{c} S_1 \quad S_2 \quad S_3 \quad S_4 \quad S_5 \quad S_6 \quad S_7 \quad S_k^1 \quad S_k^2 \quad S_k^3 \\ 1 \quad 1 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \end{array}$$

3) Поскольку $S_3 \cap m_1 = \emptyset$, то S_3 включается в m_1

$$m_1 = \begin{array}{c} S_1 \quad S_2 \quad S_3 \quad S_4 \quad S_5 \quad S_6 \quad S_7 \quad S_k^1 \quad S_k^2 \quad S_k^3 \\ 1 \quad 1 \quad 1 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \end{array}$$

4) Поскольку $S_4 \cap m_1 \neq \emptyset$, то S_4 не включается в m_1 и образуется подмножество m_2 , в которое войдет S_4 в качестве первого элемента

$$\begin{array}{l} m_1 = \begin{array}{c} S_1 \quad S_2 \quad S_3 \quad S_4 \quad S_5 \quad S_6 \quad S_7 \quad S_k^1 \quad S_k^2 \quad S_k^3 \\ 1 \quad 1 \quad 1 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \end{array} \\ m_2 = \begin{array}{c} S_1 \quad S_2 \quad S_3 \quad S_4 \quad S_5 \quad S_6 \quad S_7 \quad S_k^1 \quad S_k^2 \quad S_k^3 \\ 0 \quad 0 \quad 0 \quad 1 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \end{array} \end{array}$$

5) Поскольку $S_5 \cap m_1 \neq \emptyset$, а $S_5 \cap m_2 \neq \emptyset$, то S_5 не войдет в m_1 , а войдет в m_2

$$\begin{array}{c} S_1 \quad S_2 \quad S_3 \quad S_4 \quad S_5 \quad S_6 \quad S_7 \quad S_k^1 \quad S_k^2 \quad S_k^3 \end{array}$$

$$\begin{array}{l} m_1 = \\ m_2 = \end{array} \begin{array}{ccccccccccc} 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \end{array}$$

6) Поскольку $S_6 \cap m_1 \neq \emptyset$ и $S_6 \cap m_2 \neq \emptyset$, то S_6 не войдет ни в m_1 , ни в m_2 , поэтому образуется новое подмножество m_3 , в которое войдет S_6 в качестве первого элемента

$$\begin{array}{l} m_1 = \\ m_2 = \\ m_3 = \end{array} \begin{array}{ccccccccccc} S_1 & S_2 & S_3 & S_4 & S_5 & S_6 & S_7 & S_k^1 & S_k^2 & S_k^3 \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \end{array}$$

7) Поскольку $S_7 \cap m_1 \neq \emptyset$ и $S_7 \cap m_2 \neq \emptyset$, $S_7 \cap m_3 = \emptyset$, то S_7 не войдет ни в m_1 , ни в m_2 , а войдет m_3

$$\begin{array}{l} m_1 = \\ m_2 = \\ m_3 = \end{array} \begin{array}{ccccccccccc} S_1 & S_2 & S_3 & S_4 & S_5 & S_6 & S_7 & S_k^1 & S_k^2 & S_k^3 \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \end{array}$$

8) Поскольку $S_k^1 \cap m_1 = \emptyset$ и $S_k^1 \cap m_2 \neq \emptyset$, $S_k^1 \cap m_3 \neq \emptyset$, то S_k^1 войдет в m_1

	S_1	S_2	S_3	S_4	S_5	S_6	S_7	S_k^1	S_k^2	S_k^3
$m_1 =$	1	1	1	0	0	0	0	1	0	0
$m_2 =$	0	0	0	1	1	0	0	0	0	0
$m_3 =$	0	0	0	0	0	1	1	0	0	0

9) Поскольку $S_k^2 \cap m_1 \neq \emptyset$, $S_k^2 \cap m_3 \neq \emptyset$, а $S_k^2 \cap m_2 = \emptyset$, то S_k^2 не войдет ни в m_1 , ни в m_3 , а войдет m_2

	S_1	S_2	S_3	S_4	S_5	S_6	S_7	S_k^1	S_k^2	S_k^3
$m_1 =$	1	1	1	0	0	0	0	1	0	0
$m_2 =$	0	0	0	1	1	0	0	0	1	0
$m_3 =$	0	0	0	0	0	1	1	0	0	0

10) Поскольку $S_k^3 \cap m_1 \neq \emptyset$, $S_k^3 \cap m_2 \neq \emptyset$, а $S_k^3 \cap m_3 = \emptyset$, то S_k^3 не войдет ни в m_1 , ни в m_2 , а войдет m_3

	S_1	S_2	S_3	S_4	S_5	S_6	S_7	S_k^1	S_k^2	S_k^3
$m_1 =$	1	1	1	0	0	0	0	1	0	0
$m_2 =$	0	0	0	1	1	0	0	0	1	0
$m_3 =$	0	0	0	0	0	1	1	0	0	1

11) Поскольку события S_0 и S_8 не совместимы ни с одним из частных событий исходного управляющего алгоритма (в полные события a_0 и a_{25} включаются только по одному событию S_0 и S_8 соответственно), то они могут войти в любое из подмножеств m_1 , m_2 , m_3 , либо образовать новое подмножество m_4 .

Таким образом в результате построения матрицы включения получили следующие группы несовместимых частных событий:

$$\begin{aligned}
 m_1 &= [S_1, S_2, S_3, S_k^1]; \\
 m_2 &= [S_4, S_5, S_k^2]; \\
 m_3 &= [S_6, S_7, S_k^3]; \\
 m_4 &= [S_0, S_8];
 \end{aligned}
 \tag{5.4}$$

Полученное в рассмотренном примере разбиение частных событий на группы несовместимых частных событий полностью соответствует размещению этих событий на ГСАП, представленным на рис. 4.6.

5.1.3. Пример построения структуры распределенной системы МПУ по управляющему алгоритму с взаимодействующими параллельными ветвями

Как было показано в разделе 5.2.1, если управляющий алгоритм содержит параллельные взаимодействующие ветви, то для структурной реализации такого алгоритма может быть предложена структура распределенной системы управления, состоящая из нескольких параллельно работающих подавтоматов, каждый из которых реализует одну из групп несовместимых частных событий и главного подавтомата, реализующего последовательную часть алгоритма управления и взаимодействие рабочих подавтоматов.

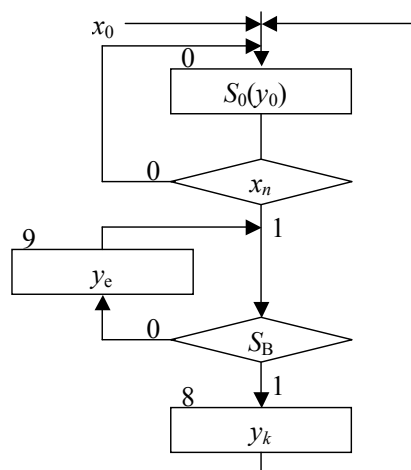
В качестве исходного управляющего алгоритма для примера построения структуры распределенной системы МПУ примем управляющий алгоритм, заданный на языке ГСАП (рис.4.6) с тремя параллельными ветвями и с конъюнктивным соединителем, для которого выход алгоритмического процесса за соединитель возможен только в том случае, когда закончится алгоритмический процесс во всех ветвях, но не обязательно одновременно. Особенностью данного примера управляющего алгоритма является то, что начальная и заключительная компоненты последовательной части алгоритма управления содержат только по одному событию: S_0 и S_8 (см. ГСАП на рис. 4.6).

Для нашего примера для исходного управляющего алгоритма имеет место 4 группы несовместимых частных S -событий (5.4):

Будем рассматривать структуру МПУ, состоящей из главного п/А и 3-х рабочих п/А. Тогда в соответствии с изложенным исходная ГСАП (рис.4.6) будет представлена следующими частными ГСА и соответствующими им СКУ для частных S -событий:

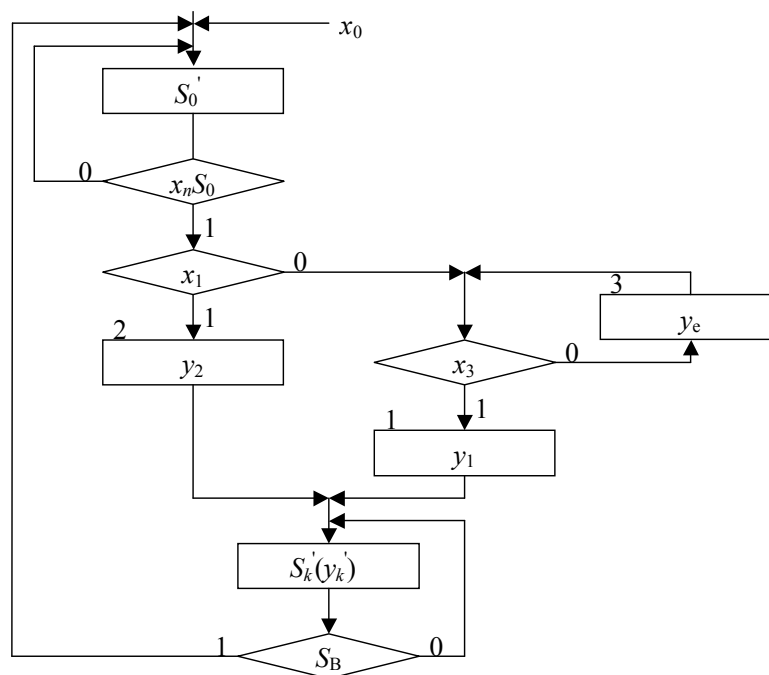
Для главного п/А

$$\begin{aligned} S_0(t+1) &= x_0 \vee S_8 \vee S_0 \bar{x}_n, \\ S_9(t+1) &= (S_0 x_n \vee S_9) \bar{S}_B, \\ S_8(t+1) &= S_9 S_B, \\ S_B &= S_k^1 S_k^2 S_k^3. \end{aligned} \tag{5.5}$$



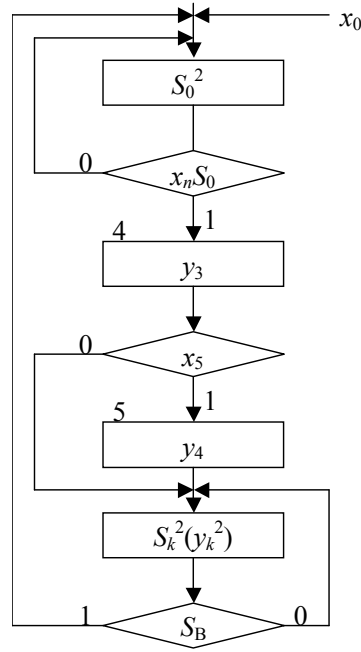
Для первого п/А

$$\begin{aligned}
 S_0^1(t+1) &= x_0 \vee S_k' S_B \vee S_0'(\overline{x_n S_0}), \\
 S_1(t+1) &= S_0'(x_n S_0) \bar{x}_1 x_3 \vee S_3 x_3, \\
 S_2(t+1) &= S_0'(x_n S_0) x_1, \\
 S_3(t+1) &= S_0'(x_n S_0) \bar{x}_1 \bar{x}_3 \vee S_3 \bar{x}_3, \\
 S_k'(t+1) &= S_1 \vee S_2 \vee S_k' \bar{S}_B.
 \end{aligned}
 \tag{5.6}$$



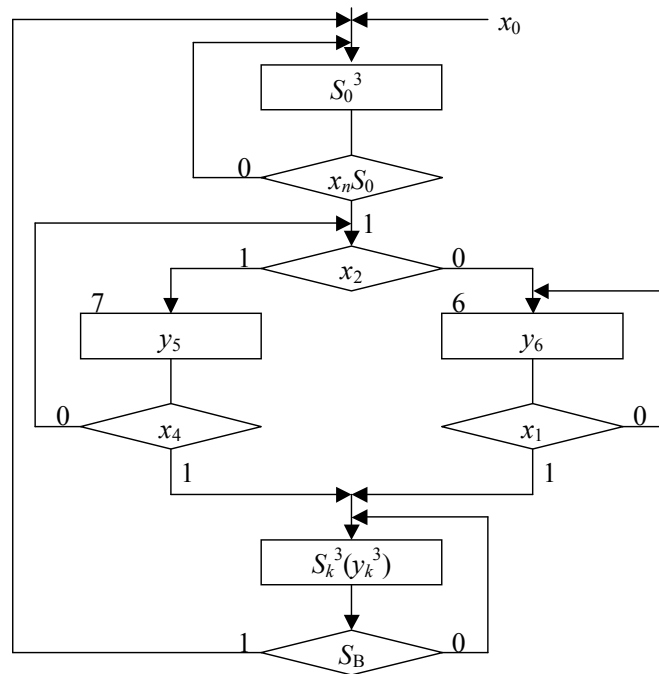
Для второго п/А

$$\begin{aligned}
S_0^2(t+1) &= x_0 \vee S_k^2 S_B \vee S_0^2(\overline{x_n S_0}), \\
S_4(t+1) &= S_0^2(x_n S_0), \\
S_5(t+1) &= S_4 x_5, \\
S_k^2(t+1) &= S_4 \bar{x}_5 \vee S_5 \vee S_k^2 \bar{S}_B
\end{aligned}
\tag{5.7}$$



Для третьего п/А

$$\begin{aligned}
S_0^3(t+1) &= x_0 \vee S_k^3 S_B \vee S_0^3(\overline{x_n S_0}), \\
S_6(t+1) &= S_0^3(x_n S_0) \bar{x}_2 \vee S_6 \bar{x}_1 \vee S_7 \bar{x}_4 \bar{x}_2, \\
S_7(t+1) &= S_0^3(x_n S_0) x_2 \vee S_7 \bar{x}_4 x_2, \\
S_k^3(t+1) &= S_6 x_1 \vee S_7 x_4 \vee S_k^3 \bar{S}_B
\end{aligned}
\tag{5.8}$$



Для построения структуры всех п/А по их СКУ необходимо выполнить следующие процедуры:

а) Кодирование событий (состояний) в каждой из групп m_1 , m_2 , m_3 и m_4 несовместимых событий, реализуемых одним из подавтоматов.

m_1	$Q_1 Q_2 Q_3$
S_0^1	0 0 0
S_1	0 0 1
S_2	0 1 0
S_3	0 1 1
S_k^1	1 0 0

m_2	$Q_4 Q_5$
S_0^2	0 0
S_4	0 1
S_5	1 0
S_k^2	1 1

m_3	$Q_6 Q_7$
S_0^3	0 0
S_6	0 1
S_7	1 0
S_k^3	1 1

m_4	$Q_8 Q_9$
S_0^4	0 0
S_8	1 0
S_9	0 1

Внутри каждой группы несовместимых событий примем произвольное максимальное кодирование.

б) Построение СКУ для Q -событий для каждого п/А.

Будем предполагать, что в качестве элементов памяти приняты Д-триггеры. Тогда по таблице кодирования и по СКУ для каждого из п/А получим СКУ для Q -событий.

$$\begin{aligned}
Q_1(t+1) &= S_k^1(t+1) = S_1 \vee S_2 \vee S_k^1 \bar{S}_B, \\
Q_2(t+1) &= (S_2 \vee S_3)(t+1) = S_0^1(x_n S_0) x_1 \vee S_0^1(x_n S_0) \bar{x}_1 \bar{x}_3 \vee S_3 \bar{x}_3, \\
Q_3(t+1) &= (S_1 \vee S_3)(t+1) = S_0^1(x_n S_0) \bar{x}_1 x_3 \vee S_3 x_3 \vee S_0^1(x_n S_0) \bar{x}_1 \bar{x}_3 \vee S_3 \bar{x}_3, \\
Q_4(t+1) &= (S_5 \vee S_k^2)(t+1) = S_4 x_5 \vee S_4 \bar{x}_5 \vee S_5 \vee S_k^2 \bar{S}_B, \\
Q_5(t+1) &= (S_4 \vee S_k^2)(t+1) = S_0^2(x_n S_0) \vee S_4 \bar{x}_5 \vee S_5 \vee S_k^2 \bar{S}_B, \\
Q_6(t+1) &= (S_6 \vee S_k^3)(t+1) = S_0^3(x_n S_0) \bar{x}_2 \vee S_6 \bar{x}_1 \vee S_6 x_1 \vee S_7 x_4 \vee S_k^3 \bar{S}_B, \\
Q_7(t+1) &= (S_7 \vee S_k^3)(t+1) = S_0^3(x_n S_0) x_2 \vee S_7 \bar{x}_4 x_2 \vee S_6 x_1 \vee S_7 x_4 \vee S_k^3 \bar{S}_B, \\
Q_8(t+1) &= S_8(t+1) = S_9 S_B, \\
Q_9(t+1) &= S_9(t+1) = (x_n S_0 \vee S_9) \bar{S}_B.
\end{aligned} \tag{5.9}$$

В полученных уравнениях (5.9) СКУ для Q -событий выполним подстановку в правую часть кодов для всех событий в соответствии с таблицами кодирования. Тогда получим СКУ для Q -событий, в которых правая часть будет зависеть от Q -событий. Далее для иллюстрации процедур будем рассматривать все преобразования только для 1-го п/А. Для всех других п/А все преобразования будут выполняться аналогично. Учитывая отмеченное, получим для 1-го п/А СКУ для Q -событий:

$$\begin{aligned}
Q_1(t+1) &= \bar{Q}_1 \bar{Q}_2 Q_3 \vee \bar{Q}_1 Q_2 \bar{Q}_3 \vee Q_1 \bar{Q}_2 \bar{Q}_3 \bar{S}_B, \\
Q_2(t+1) &= \bar{Q}_1 \bar{Q}_2 \bar{Q}_3 (x_n S_0) (x_1 \vee \bar{x}_3) \vee \bar{Q}_1 Q_2 Q_3 \bar{x}_3, \\
Q_3(t+1) &= \bar{Q}_1 \bar{Q}_2 \bar{Q}_3 (x_n S_0) x_1 \vee \bar{Q}_1 Q_2 Q_3.
\end{aligned} \tag{5.10}$$

в) Минимизацию систем уравнений типа (5.10) СКУ для Q -событий для всех п/А с учетом неиспользованных кодовых групп.

Для 1-го п/А в результате минимизации получим следующую систему СКУ для Q -событий.

$$\begin{aligned}
Q_1(t+1) &= Q_2 \bar{Q}_3 \vee Q_1 \bar{S}_B \vee \bar{Q}_2 Q_3, \\
Q_2(t+1) &= Q_2 Q_3 \bar{x}_3 \vee \bar{Q}_1 \bar{Q}_2 \bar{Q}_3 (x_n S_0) (x_1 \vee \bar{x}_3), \\
Q_3(t+1) &= \bar{Q}_1 \bar{Q}_2 (x_n S_0) \bar{x}_1 \vee Q_2 Q_3.
\end{aligned} \tag{5.11}$$

Так как в качестве элементарных автоматов приняты Д-триггеры, то система уравнений (5.11) является функциями возбуждения элементов памяти, по которой не трудно построить структурную схему для рабочего п/А, реализующую его функции перехода.

В том случае, когда при построении структуры распределенной системы МПУ функции главного п/А возлагают на один из рабочих п/А, то тогда строится ГСА и соответствующая ему СКУ для такого совмещенного п/А. Если в качестве такого рабочего п/А принять 3-й п/А, то ГСА и СКУ совмещенного п/А будут иметь вид:

$$S_0(t+1) = x_0 \vee S_8 \vee S_0 \bar{x}_n,$$

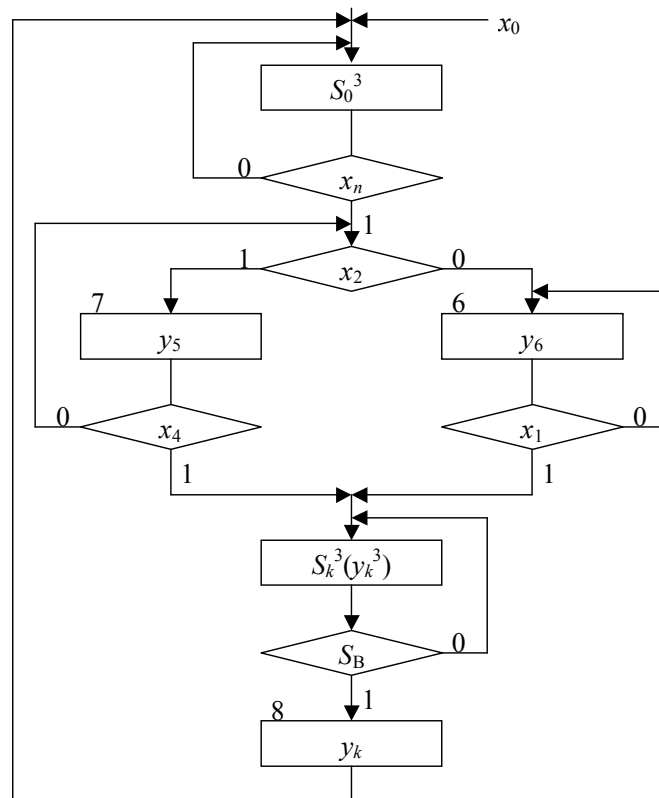
$$S_6(t+1) = (S_0 x_n) \bar{x}_2 \vee S_0 \bar{x}_1 \vee S_7 \bar{x}_1 \bar{x}_2,$$

$$S_7(t+1) = S_7 \bar{x}_4 x_2 \vee (S_0 x_n) x_2,$$

$$S_k^3(t+1) = S_6 x_1 \vee S_7 x_4 \vee S_k^3 \bar{S}_B,$$

$$S_8(t+1) = S_k^3 S_B,$$

$$S_B(t+1) = S_k^1 S_k^2 S_k^3(t)$$



5.2. Структурная реализация систем МПУ, заданных моделью НДА, на основе использования унитарного кодирования частных событий

Для некоторых вариантов управляющих алгоритмов, представленных моделью НДА, частные события, входящие в состав НД СКУ, не могут быть разложены на группы несовместимых событий, поэтому такой управляющий алгоритм невозможно реализовать на основе структуры, состоящей из параллельно работающих п/А, т.е. не может быть построена управляющая система для распределенной параллельной обработки.

Отличительной особенностью управляющих алгоритмов такого типа является то, что каждое частное событие, представленное в алгоритме, совместимо со всеми другими частными событиями, входящими в той или иной комбинации в полные события, определяемыми при детерминизации управляющего алгоритма. Для таких управляющих алгоритмов, заданных моделью НДА, чтобы избежать недостатков классического метода синтеза структур МПУ, можно предложить методику построения структуры МПУ, базирующуюся на использовании унитарного кодирования частных событий. Особенно целесообразно использование такой методики для построения структуры систем МПУ, для которых в управляющем алгоритме должны параллельно выполняться различные микрооперации и имеется достаточно много повторяющихся комбинаций из них. Такие комбинации микроопераций, выполняемые в одно и то же время, принято называть **нанокомандой**.

В дальнейшем будем рассматривать два возможных варианта организации систем МПУ на основе использования унитарного кодирования частных событий. Для первого варианта, соответствующего одноуровневой и двухуровневой организации системы МПУ, функции переходов и функции выходов системы МПУ реализуются на основе использования комбинационных схем. Для второго варианта, соответствующего двухуровневой организации функции выходов системы МПУ, реализуются на основе использования нанопамяти, в которую записывается все многообразие нанокоманд, предусмотренных в управляющем алгоритме.

5.2.1. Одноуровневая организация структуры системы МПУ для унитарного кодирования частных событий

Если исходный управляющий алгоритм задан недетерминированной СКУ, то один из вариантов одноуровневой организации структур системы МПУ можно представить следующей схемой (рис. 5.8).

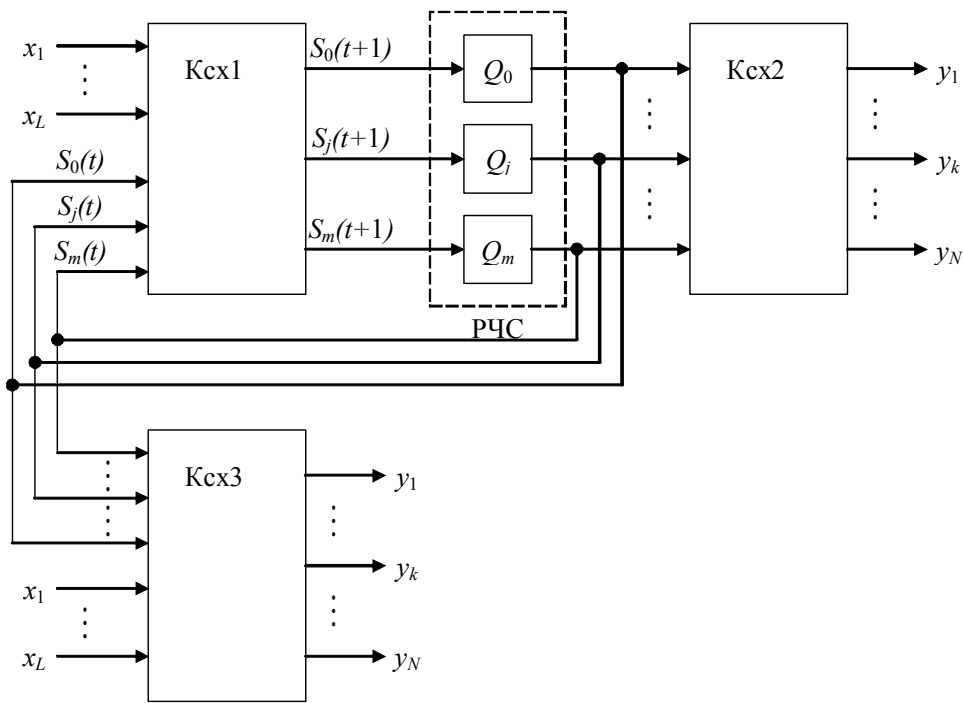


Рис. 5.8. Структурная схема одноуровневой системы МПУ, представленная на основе использования НД СКУ в виде совмещенного С-автомата

На рис. 5.8 представлены следующие блоки системы МПУ:

Ксх1 - многовыходная комбинационная схема реализует функции переходов в системе МПУ (последовательность выполнения микрокоманд):

$$S_j^{Y_j}(t+1) = f_j(x_1, x_2, \dots, x_L, S_0, S_1, \dots, S_m), \quad j = \overline{0, m} \quad (5.12)$$

Ксх2 - многовыходная комбинационная схема реализует функции выходов модели автомата Мура:

$$y_k(t) = \bigvee S_j^{Y_j}(t), \quad k = \overline{1, N};$$

$$(\forall S_j)(Y_j \subset y_k) \quad (5.13)$$

Ксх3 - многовыходная комбинационная схема реализует функции выходов модели автомата Мили:

$$y_k(t) = \bigvee S_j^{Y_j}(t+1), \quad k = \overline{1, N} \quad (5.14)$$

где $[x_1, x_2, \dots, x_L]$ - элементарные входные сигналы (осведомительные сигналы от операционного автомата (ОА));

$[y_1, \dots, y_k, \dots, y_N]$ - элементарные выходные сигналы, инициирующие выполнение микроопераций в ОА;

$[S_0, S_1, \dots, S_m]$ - частные события, реализуемые в алгоритме управления;

Y_j - совокупность управляющих сигналов, отмечающих событие S_j ;

РЧС - регистр частных событий, организованный на двухступенчатых Д-триггерах; для унитарного кодирования событий каждому событию S_j

соответствует свой триггер, для которых функции возбуждения имеют вид:
 $q_j(t) = Q_j(t+1) = S_j(t+1)$.

Отличительной особенностью структуры системы МПУ представленной на рис. 5.8, по сравнению с вариантом структуры систем МПУ, построенной на основе использования детерминированной СКУ, является ее значительная простота.

В тех случаях, когда исходный управляющий алгоритм будет иметь также и такие сложные управляющие конструкции как обращение к микроподпрограммам, сложные циклы и простые линейные части, то представленная на рис. 5.8 структура системы МПУ может быть использована в качестве составной части общей системы МПУ в качестве блока реализации многоальтернативных переходов.

5.2.2. Двухуровневая организация структуры системы МПУ для унитарного кодирования частных событий

Двухуровневую организацию структуры системы МПУ для унитарного кодирования частных событий целесообразно использовать, когда совокупности параллельно выполняемых микроопераций (нанокоманд) часто повторяются в различных микрокомандах управляющего алгоритма. В этом случае количество адресов нанокоманд может существенно сократиться, что приведет к упрощению общей структуры системы МПУ. Рассмотрим один из возможных вариантов двухуровневой организации системы МПУ, когда функции переходов, определяющие последовательность выполнения микрокоманд, реализованы с использованием многовыходной комбинационной схемы, которая определяет первый уровень структуры системы МПУ, а второй уровень представлен нанопамятью со схемой формирования адреса нанокоманд. Такая организация структуры системы МПУ представлена на рис. 5.9.

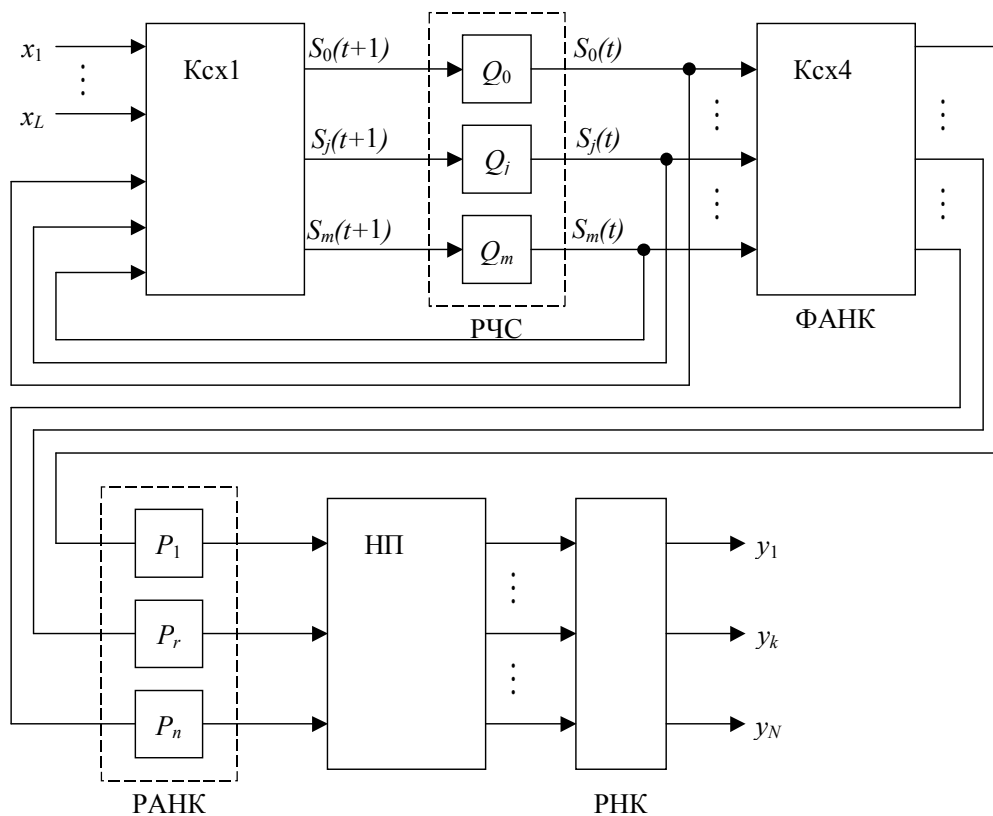


Рис. 5.9. Структурная схема двухуровневой системы МПУ для унитарного кодирования частных событий

Структура системы МПУ (рис. 5.9) включает следующие три основных блока: блок определения всех частных событий исходной НД СКУ, состоящий из многовыходной комбинационной схемы (Ксх1) и регистра частных событий (РЧС); блок формирования адреса наноконанд (ФАНК) с регистром адреса наноконанд (РАНК) и блок нанопамати (НП) с выходным регистром наноконанд (РНК).

Блок формирования всех частных событий строится на основе НД СКУ модели автомата Мура, формализующих последовательность выполнения микрокоманд (функции переходов) управляющего алгоритма. Регистр частных событий состоит из двухступенчатых Д-триггеров, каждому из которых соответствует свое частное событие.

Блок формирования наноконанд (ФАНК) реализуется на основе комбинационной схемы (Ксх4), выполняющей функции шифратора. Для построения этого блока используется таблица соответствия, которая формируется по результатам детерминизации исходного управляющего алгоритма. В соответствии с алгоритмом детерминизации определяются все сочетания одновременно существующих частных событий, каждому из которых ставится в соответствие полное событие или вполне определенное состояние системы МПУ, отмеченное совокупностью выходных управляющих сигналов.

Поскольку для автомата Мура существует однозначное соответствие между состоянием автомата и сочетаниями отмечающих их управляющих сигналов, то каждому полному событию можно поставить в соответствие адрес нанопамяти, по каждому из которых в нанопамть можно записать код совокупности управляющих сигналов, которыми отмечаются частные события, входящие в полное событие. При этом каждому k -му управляющему сигналу y_k будет соответствовать свой разряд слова нанопамяти. Таким образом таблица соответствия, используемая для построения блока ФАНК, будет иметь следующий вид (рис. 5.10).

Полное событие	Сочетание частных событий, реализуемых управляющим алгоритмом одновременно	Код адреса наноккоманды	Код наноккоманды
a_i	$S_0 S_1 S_2 \dots S_j \dots S_{m-1} S_m$	$p_1 \dots p_r \dots p_n$	$y_1 y_2 \dots y_k \dots y_N$

Рис. 5.10. Таблица соответствия, построенная по результатам детерминизации управляющего алгоритма

При построении таблицы соответствия (рис. 5.10) необходимо иметь ввиду, что, если некоторые полные события отмечаются одинаковыми совокупностями выходных сигналов, то соответствующие им адреса нанопамяти должны иметь одинаковый код.

Блок нанопамяти строится на основе использования ПЗУ, число ячеек которого не должно быть меньше числа всех возможных сочетаний выходных сигналов, определяемых в процессе детерминизации управляющего алгоритма. Количество разрядов в слове нанопамяти для унитарного кодирования определяется числом всех управляющих сигналов исходного алгоритма управления, каждый из которых инициирует выполнение определенной микрооперации.

Двухуровневая реализация системы МПУ дает большой эффект при высокой частоте появления одинаковых адресов наноккоманд для различных микрокоманд. Такая организация управления позволяет:

- сохранить возможность параллельного выполнения микроопераций и эффективно использовать память ПЗУ, т.к. исключаются адреса с одинаковыми комбинациями управляющих сигналов;
- структура блока, реализующего функции переходов управляющего алгоритма, отличается значительной простотой, т.к. основой его построения является НД СКУ, которая представляет самое компактное описание алгоритма управления.

Пример 5.2. Построить блок формирования адреса наноккоманды для системы МПУ, управляющий алгоритм которой представлен

недетерминированным направленным графом модели автомата Мура (рис. 1.1).

В результате детерминизации исходного управляющего алгоритма получены следующие полные события и соответствующие им совокупности управляющих сигналов:

$$a_0 = S_0(y_0),$$

$$a_1 = S_1S_2(y_1y_2),$$

$$a_2 = S_1S_4(y_1y_2y_4),$$

$$a_3 = S_2S_3S_4(y_2y_3y_4),$$

$$a_4 = S_1S_3S_5(y_1y_3y_4),$$

$$a_5 = S_3(y_3y_4),$$

$$a_6 = S_1S_3(y_1y_3y_4),$$

$$a_7 = S_3S_4(y_2y_3y_4),$$

$$a_8 = S_2S_3(y_2y_3y_4),$$

$$a_9 = S_4(y_2y_4),$$

$$a_{10} = S_3S_4S_5(y_1y_2y_3y_4),$$

$$a_{11} = S_3S_5(y_1y_3y_4),$$

$$a_{12} = S_1S_3S_4(y_1y_2y_3y_4).$$

Таблица соответствия, построенная по результатам детерминизации исходного управляющего алгоритма, будет иметь следующий вид:

Таблица 5.1

Полное событие	Сочетание частных событий, реализуемых управляющим алгоритмом одновременно						Код адреса нано-команды			Код нанокomанды				
a_i	S_0	S_1	S_2	S_3	S_4	S_5	p_1	p_2	p_3	y_0	y_1	y_2	y_3	y_4
1	2						3			4				
a_0	1	0	0	0	0	0	0	0	0	1	0	0	0	0
a_1	0	1	1	0	0	0	0	0	1	0	1	1	0	0
a_2	0	1	0	0	1	0	0	1	0	0	1	1	0	1
a_3	0	0	1	1	1	0	0	1	1	0	0	1	1	1
a_4	0	1	0	1	0	1	1	0	0	0	1	0	1	1
a_5	0	0	0	1	0	0	1	0	1	0	0	0	1	1
a_6	0	1	0	1	0	0	1	0	0	0	1	0	1	1
a_7	0	0	0	1	1	0	0	1	1	0	0	1	1	1

Продолжение таблицы 5.1

Полное событие	Сочетание частных событий, реализуемых управляющим алгоритмом одновременно						Код адреса нано-команды			Код нанокomанды				
a_8	0	0	1	1	0	0	0	1	1	0	0	1	1	1
a_9	0	0	0	0	1	0	1	1	0	0	0	1	0	1
a_{10}	0	0	0	1	1	1	1	1	1	0	1	1	1	1
a_{11}	0	0	0	1	0	1	1	0	0	0	1	0	1	1

a_{12}	0	1	0	1	1	0	1	1	1	0	1	1	1	1
----------	---	---	---	---	---	---	---	---	---	---	---	---	---	---

При кодировании адресов НК, как было отмечено выше, необходимо учитывать наличие в таблице истинности полных событий, которые отмечены одинаковыми выходными сигналами. Для нашего примера это три группы полных событий:

$$(a_3, a_7, a_8), (a_4, a_6, a_{11}), (a_{10}, a_{12}).$$

Для каждой группы таких полных событий предусматривается свой индивидуальный код адреса НК. Для произвольного кодирования адресов НК из таблицы соответствия следует, что адрес НК можно сформировать комбинационной схемой, реализующей три не полностью определенных булевых функций: p_1, p_2, p_3 .

Для нашего примера функции p_1, p_2 и p_3 не определены на 51-ом наборе значений переменных $S_0, S_1, \dots, S_5$. Используя диаграммы Вейча для минимизации указанных не полностью определенных функций, получим следующие представления их в ДНФ:

$$\begin{aligned} p_1 &= S_5 \vee S_1 S_3 \vee \bar{S}_1 \bar{S}_3 S_4 \vee \bar{S}_2 S_3 \bar{S}_4, \\ p_2 &= S_4 \vee S_2 S_3 \vee \bar{S}_5, \\ p_3 &= S_2 \vee \bar{S}_1 S_3 \bar{S}_5 \vee S_3 S_4. \end{aligned} \quad (5.15)$$

Блок формирования адреса НК может быть построен на ЛЭ, например, ПЛМ, на основе использования многовыходных булевых функций типа (5.15).

В том случае, если исходный управляющий алгоритм содержит, как было отмечено в п. 5.3.2. и другие сложные управляющие конструкции, то структуру системы МПУ для таких управляющих алгоритмов целесообразно строить на основе использования типовых блоков микропрограммного управления (БМУ) с двухуровневой реализацией системы МПУ. Для таких блоков БМУ программа работы управляющего алгоритма будет представлена в управляющей памяти (УП), а блок формирования адреса следующей микрокоманды (ФАМК) расширяется за счет организации в нем многоальтернативных переходов. В основу построения блока, обеспечивающего многоальтернативные переходы с высокой скоростью, может быть использован принцип построения блока формирования частных событий, рис. 5.9. Если адреса микрокоманд будут представлены в УП в каждой микрокоманде [2], то обобщенная схема структуры системы МПУ будет иметь следующий вид, рис. 5.11. Подобная структурная организация двухуровневой системы МПУ используется, например, в микропроцессоре МС 68010 фирмы «Моторола» [48].

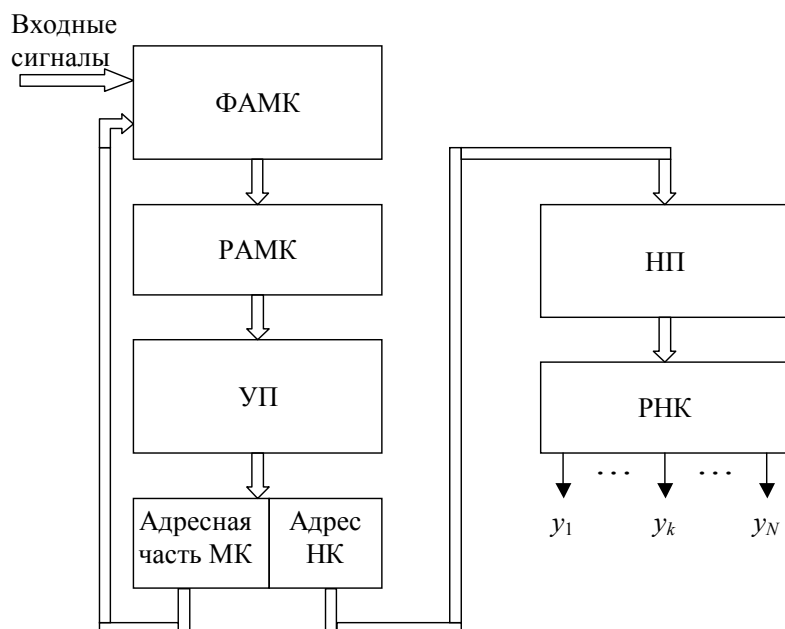


Рис. 5.11. Обобщенная схема двухуровневой системы МПУ

В том случае, если в пределах неизменяемых микропрограмм имеется однозначное соответствие между адресом микрокоманды и адресом НК, то адрес нанокоманды можно сформировать путем преобразования адреса микрокоманды в соответствующий адрес НК с помощью комбинационной схемы ФАНК. Такой вариант формирования адреса НК позволит еще более уменьшить длину слов управляющей памяти. Структурная организация системы МПУ в соответствии со схемой рис. 5.11 позволяет при сохранении возможности параллельного выполнения микроопераций вынести основную управляющую память (УП) за пределы кристалла микропроцессора, что в свою очередь позволяет экономить площадь кристалла до 40 %, на которой могут быть размещены дополнительные схемы. Кроме того повышается универсальность системы МПУ путем возможной модификации УП. Такая реализация основной УП нашла применение, например, в ЭВМ типа LSI 11 фирмы DEC, Т-88000 фирмы «Тосиба» и др. [48].

5.3. Некоторые операции композиции систем канонических уравнений и их структурная интерпретация

В данном разделе рассматриваются операции над СКУ, которые позволяют выполнить ее преобразование с целью удовлетворения определенным структурным требованиям, предъявляемым к проектируемой системе управления. Среди таких операций рассматриваются операции суммирования, конкатенации и объединения СКУ [11]

Суммирование СКУ. Эта операция сводится к получению из двух или более СКУ общей СКУ с целью построения нового автомата, который будет эквивалентен двум или более параллельно работающим автоматам, построенным по исходным СКУ. Таким образом, операция суммирования в структурной интерпретации эквивалентна замене параллельного соединения нескольких автоматов одним автоматом. При этом, если каждый из исходных автоматов представлен детерминированной СКУ, то в результате операции суммирования будет получена недетерминированная СКУ, так как в объединенной СКУ будет объединено несколько начальных событий в одно событие.

Для выполнения операции суммирования с последующей детерминизацией объединенной СКУ необходимо:

- переобозначить события (состояния) исходных СКУ, например, введением для событий дополнительного верхнего индекса, указывающего на принадлежность события к СКУ с номером i , где $i = \overline{1, l}$, l - число исходных автоматов;
- образовать новую объединенную СКУ путем объединения уравнений всех исходных СКУ в одну группу;
- образовать начальное событие для результирующей объединенной СКУ путем объединения множеств начальных событий исходных СКУ в группу (совокупность), представляющую собой конъюнкцию начальных событий исходных СКУ;
- выполнить детерминизацию объединенной СКУ, если это необходимо для дальнейшего синтеза объединенного автомата.

Для варианта суммирования СКУ, когда исходные автоматы заданы в детерминированной форме, при выполнении операции детерминизации объединенной СКУ будут возникать сочетания из l событий (состояний). В каждое такое сочетание из l событий будут входить только по одному из событий (состояний) каждой из исходных детерминированных СКУ, а общее число таких сочетаний, полных событий, объединенной детерминированной СКУ будет не более величины $M = M_1 \dots M_i \dots M_l$, где M_i - число событий (состояний) i -го автомата. В том случае, если в каждом из автоматов все состояния, в том числе и начальное достижимы из любого состояния, то общее число полных событий объединенной детерминированной СКУ будет равно M . Тогда для рассматриваемого варианта исходных автоматов все сочетания из обозначений событий (состояний), входящих в исходные СКУ, можно заранее перечислить, т.е. в этом случае для решения задачи определения всех возможных сочетаний событий, одновременное существование которых в объединенном автомате возможно, не потребуются использование алгоритма детерминизации, рассмотренного ранее. В то же время описание полных событий детерминированной СКУ объединенного автомата можно получить в этом случае путем выполнения операции конъюнкции правых частей тех уравнений исходных СКУ, формализующих события, сокращенные обозначения которых входят в рассматриваемое полное событие.

В том случае, если в исходных детерминированных СКУ не удовлетворяются условия достижимости состояний, указанные выше или если исходные СКУ заданы в недетерминированной форме, то детерминизацию объединенной СКУ можно выполнить только с использованием алгоритма детерминизации, рассмотренного в главе 2.

Получение СВФ объединенного автомата выполняется по-разному в зависимости от того, одинаковый или разный смысл имеют выходные сигналы объединяемых автоматов. Если все автоматы работают на один и тот же операционный автомат, то все сигналы имеют одинаковый смысл и СВФ объединенного автомата получают следующим образом. Осуществляют дизъюнкцию правых частей уравнений СВФ объединяемых автоматов для одинаковых выходных сигналов. Далее, если необходимо выразить уравнения СВФ через полные события, то процедура их построения будет та же, что была рассмотрена в главе 2.

Если же действие одинаково обозначенных выходных сигналов в разных автоматах на операционный автомат разное, то необходимо выходные сигналы в исходных уравнениях СВФ переобозначить, например, путем введения верхних индексов.

Конкатенация СКУ. Конкатенация двух СКУ позволяет получить автомат, последовательно реализующий два алгоритма. Для выполнения такой операции необходимо отождествить конечное событие первой СКУ $S_{k,1}$ с начальным событием второй $S_{0,2}$. Под конечным событием здесь понимается событие первой СКУ,

от которого происходит переход ко второму алгоритму. В качестве конечного события первой SKU могут быть приняты как тупиковое, так и не тупиковое событие.

Если в качестве конечного события первой SKU принимается тупиковое событие, т.е. такое, которое не встречается в качестве предшествующего события ни в одном из уравнений SKU, то результирующую SKU можно получить переобозначением конечного события первой SKU обозначением начального события второй SKU. Кроме того, если вторая SKU является недетерминированной и имеет несколько начальных событий, то предварительно должна быть выполнена операция стягивания их в одно начальное событие.

Если в качестве конечного события первой SKU принимается не тупиковое событие, то переход от него к алгоритму, представленному второй SKU, будет возможен только при определенных значениях входных сигналов. Поэтому в результирующей SKU необходимо сформировать событие $S_0 = S_{k,1} \vee S_{0,2}$,

представляющее собой дизъюнкцию конечного события первой SKU и начального события второй SKU. В описании этого события и событий, выводимых из него, необходимо выполнить операцию замены обозначений $S_{k,1}$ и $S_{0,2}$ на обозначение S_0 .

Если SKU первого уровня имеет несколько конечных событий, после каждого из которых реализуется свой алгоритм, описываемый соответствующей SKU второго уровня, то результирующую SKU получим последовательным выполнением операций конкатенации SKU первого уровня с каждой из SKU второго уровня. В полученной результирующей SKU будет, как и в исходной SKU первого уровня, несколько конечных (финальных) событий, которые при необходимости стягиваются в один полюс путем введения нового конечного события, образованного дизъюнкцией правых частей уравнений для конечных событий всех исходных SKU второго уровня. При таком объединении нескольких SKU невозможно объединить их общие фрагменты и одинаковые события. Перед объединением все одинаковые события из разных SKU должны быть переобозначены (например, введением верхнего индекса).

Операция объединения нескольких SKU с расширением входного алфавита. Эта операция позволяет объединить фрагменты нескольких SKU, реализуемых одним автоматом, в общую объединенную SKU с расширением алфавита входных сигналов сигналами, кодирующими выбор той или иной SKU. Такая операция объединения сохраняет детерминированность SKU и позволяет выполнить один из возможных алгоритмов, задаваемых исходными SKU, в зависимости от кода, поступающего на вход автомата (код операции). В этом случае можно не переобозначать одинаковые события из различных SKU.

Перед объединением все исходные SKU должны быть приведены к двухполюснику путем стягивания начальных и конечных (финальных) событий. Затем каждое из уравнений исходных SKU умножается (своей правой частью) на конъюнкцию входных сигналов кода операции (частный входной сигнал кода операции). Эта конъюнкция задает код, при котором будет выполняться соответствующая SKU. Далее выполняется непосредственно операция объединения SKU путем дизъюнкции правых частей уравнений SKU для одинаковых событий и объединения всех уравнений исходных SKU в одну группу.

Таким образом в зависимости от структуры объединяемых SKU рассматриваемая операция во многих случаях позволяет достичь упрощения объединенной SKU по сравнению с общей структурой необъединенных SKU.

5.4. Контроль правильности построения функций возбуждения элементов памяти систем МПУ

В связи с тем, что операции детерминизации и кодирования НДА являются обратными друг другу, то этим обстоятельством можно воспользоваться для организации контроля правильности построения функций возбуждения элементов памяти систем МПУ. Такая процедура может быть организована на основе выполнения операции детерминизации НД SKU для Q-событий, полученной в результате кодирования, с последующей минимизацией выражений для исходных событий, представляющих управляющий алгоритм.

Если после детерминизации системы уравнений для Q-событий получим SKU, которая будет соответствовать исходной SKU до ее кодирования, то

это будет свидетельствовать о правильности системы уравнений для Q -событий, построенной по результатам кодирования исходной СКУ.

Для выполнения операции детерминизации СКУ для Q -событий с целью ее контроля необходимо алгоритм детерминизации, рассмотренный в главе 2, дополнить пунктами, вытекающими из специфики формирования полных событий (состояний автомата) и возможности отсутствия полноты входных сигналов, вызывающих переходы в Q -события. Это связано с тем, что полные события при их кодировании представляются в виде конъюнкции переменных Q_τ , часть из которых может быть взята с отрицанием.

Таким образом, в алгоритм детерминизации необходимо ввести следующие дополнения и разъяснения:

1) Если исходная система уравнений для Q -событий состоит из R уравнений, то любое полное событие, которое получается в результате детерминизации такой системы должно быть представлено конъюнкцией (сочетанием) из R букв: $\tilde{Q}_1 \dots \tilde{Q}_\tau \dots \tilde{Q}_R$, некоторые из которых могут быть как с отрицанием, так и без отрицания. Откуда следует, что, если на очередном шаге алгоритма детерминизации будут выводимы только некоторые из R событий, то сочетание этих событий необходимо дополнить отрицаниями тех Q -событий, которые не вошли в полученное сочетание.

В том случае, если на каком-то шаге алгоритма детерминизации не будет выводимо ни одно из Q -событий, то получим полное событие, равное конъюнкции отрицаний из всех Q -событий. Такое событие, ранее было названо пустым событием. В нашем случае это полное событие может быть и не пустым, ему соответствует код $\overline{Q}_1 \dots \overline{Q}_R$, принятый при кодировании некоторого полного события.

2) Начало работы алгоритма детерминизации определяется начальным полным событием, поэтому его код должен быть зафиксирован в первой строке (первый шаг алгоритма детерминизации) прямой таблицы переходов детерминизации СКУ для Q -событий.

3) Если при определении сочетаний частных входных сигналов на переходе $X(a_m, a_S)$ их совокупность на каком-либо шаге алгоритма детерминизации не удовлетворяет условиям полноты, то эта совокупность входных сигналов должна быть дополнена в соответствии с рекомендациями пункта 3 алгоритма детерминизации (глава 2). При этом необходимо иметь в виду, что для дополненных входных сигналов события переходов и соответствующие им состояния переходов (полные события) не будут неопределенными, так как выполняемая операция над СКУ для Q -событий относится к реальному, вполне определенному автомату.

Пример 5.2. Выполнить операцию детерминизации системы уравнений для Q -событий (5.16), полученной в результате кодирования СКУ (3.11).

В соответствии с системой уравнений (5.16) построим вспомогательную прямую таблицу переходов для Q -событий (табл.5.2).

$$\begin{aligned}
Q_1(t+1) &= \bar{Q}_1 \bar{Q}_2 \bar{Q}_3 \vee x_1 \bar{x}_2 \vee Q_2 Q_3, \\
Q_2(t+1) &= \bar{x}_1 Q_2 Q_3 \vee x_2 \bar{x}_3 Q_2 \bar{Q}_3, \\
Q_3(t+1) &= \bar{x}_2 \bar{Q}_3 \vee \bar{x}_4 Q_1 \vee \bar{Q}_2 \bar{Q}_3.
\end{aligned}
\tag{5.16}$$

Таблица 5.2

Шаг алгоритма	Сочетания непосредственно-предшествующих Q -событий в момент времени (t) $R_i = \tilde{Q}_{i,1} \dots \tilde{Q}_{i,R}(t)$	Частные входные сигналы на переходе $X_{i,j}(t)$	Q -событие перехода в момент времени $(t+1)$ $Q_j(t+1)$
1	2	3	4
1	$\bar{Q}_1 \bar{Q}_2 \bar{Q}_3$	1	Q_1
2	$Q_2 Q_3$	$x_1 \bar{x}_2$ $\bar{x}_1$	Q_1 Q_2
3	$Q_2 \bar{Q}_3$	$x_2 \bar{x}_3$	Q_2
4	$\bar{Q}_3$	$\bar{x}_2$	Q_3
5	Q_1	$\bar{x}_4$	Q_3
6	$\bar{Q}_2 \bar{Q}_3$	1	Q_3

На основании алгоритма детерминизации, представленного в главе 2, используя вспомогательную прямую таблицу переходов для Q -событий (табл.5.2) и учитывая требования к формированию полных событий и входных сигналов, получим прямую таблицу переходов детерминированного автомата (табл. 5.3).

Таблица 5.3

Шаг алгоритма	Сочетание всех Q -событий в момент времени (t) полное событие $a_m(t)$	Подмножества частных входных сигналов на переходе Сочетание (конъюнкция) частных входных сигналов на переходе $X(a_m, a_s)(t)$	Сочетание всех Q -событий в момент времени ($t+1$) полное событие $a_s(t+1)$
1	Сочетание всех Q -событий, соответствующее начальному полному событию $\overline{Q_1} \overline{Q_2} \overline{Q_3}$ a_0	$\overline{[x_1 \bar{x}_2]}, [\bar{x}_1]$ $x_1 \bar{x}_2$ $(x_1 x_2)^*$ $\bar{x}_1$	$\overline{Q_1} \overline{Q_2} \overline{Q_3} / a_3$ $\overline{Q_1} \overline{Q_2} \overline{Q_3} / a_5$ $\overline{Q_1} \overline{Q_2} \overline{Q_3} / a_1$
2	$\overline{Q_1} \overline{Q_2} \overline{Q_3}$ a_3	$\overline{[\bar{x}_4]}, [\bar{x}_2]$ $\bar{x}_4 \bar{x}_2$ $(\bar{x}_4 x_2)^*$ x_4	$\overline{Q_1} \overline{Q_2} \overline{Q_3} / a_6$ $\overline{Q_1} \overline{Q_2} \overline{Q_3} / a_6$ $\overline{Q_1} \overline{Q_2} \overline{Q_3} / a_6$
3	$\overline{Q_1} \overline{Q_2} \overline{Q_3}$ a_5	1	$\overline{Q_1} \overline{Q_2} \overline{Q_3} / a_2$
4	$\overline{Q_1} \overline{Q_2} \overline{Q_3}$ a_1	$\overline{[\bar{x}_2]}, [x_2 \bar{x}_3]$ $x_2 \bar{x}_3$ $(x_2 x_3)^*$ $\bar{x}_2$	$\overline{Q_1} \overline{Q_2} \overline{Q_3} / a_1$ $\overline{Q_1} \overline{Q_2} \overline{Q_3} / a_5$ $\overline{Q_1} \overline{Q_2} \overline{Q_3} / a_6$
5	$\overline{Q_1} \overline{Q_2} \overline{Q_3}$ a_2	$\overline{[\bar{x}_4]}$ $\bar{x}_4$ $(x_4)^*$	$\overline{Q_1} \overline{Q_2} \overline{Q_3} / a_6$ $\overline{Q_1} \overline{Q_2} \overline{Q_3} / a_5$

Дадим некоторые пояснения к табл.5.3. Частные входные сигналы, отмеченные звездочкой, являются дополнительными сигналами, обеспечивающими полноту переходов на соответствующем шаге алгоритма работы автомата, а обозначение сочетаний всех Q -событий принято таким же, что и при кодировании СКУ (3.11). По табл.5.3 нетрудно построить СКУ, которая будет полностью соответствовать (до обозначений) исходной СКУ (3.11).

Контрольные вопросы к главе 5.

1. Какова характерная особенность несовместимых событий, как строится матрица несовместимости событий?
2. Как разбить события, представляющие управляющий алгоритм, на группы несовместимых событий, каковы основные условия такого разбиения?
3. В чем заключаются основные недостатки «классического» способа структурного синтеза систем управления заданных моделью НДА?
4. Как доопределить состав событий, любой i -ой ветви параллельного алгоритма управления, чтобы каждая такая ветвь была реализована отдельным подавтоматом?
5. Как доопределить состав событий последовательной компоненты параллельного алгоритма управления для реализации ее главным подавтоматом? Как осуществляется синхронизация работы подавтоматов, реализующих параллельный алгоритм?
6. Как изменится состав событий любой i -ой ветви параллельного алгоритма управления, чтобы, наряду с реализацией i -ой ветви, рабочий подавтомат выполнял бы также и функции главного подавтомата?
7. В каких случаях целесообразно использовать одноуровневую и двухуровневую организацию структур системы управления для унитарного кодирования частных событий?
8. В чем заключается достоинство реализации функций выходов управляющего алгоритма, заданного моделью НДА, когда для этих целей используется нанопамять? Как определяются в этом случае адреса нанопамати?
9. В чем заключаются операции суммирования, конкатенации и объединения над СКУ и для какой цели они используются?
10. Каким образом можно организовать контроль правильности функций, определяющих переключение элементов памяти систем МПУ?

Глава 6. Формализация алгоритмов управления взаимодействующими процессами для организации параллельной обработки информации

6.1. Общие сведения о процессах и их взаимодействиях

При рассмотрении вопросов формального описания алгоритмов управления взаимодействующими процессами будем использовать некоторые основные понятия и определения, связанные с механизмом управления процессами и средствами их взаимодействия, которые приняты в теории и практике, как системного программного обеспечения при решении вопросов координации взаимодействующих процессов, так и прикладного программного обеспечения при решении задач организации параллельных вычислений.

К числу таких основных понятий и определений относятся понятия:

- процесса и основных типов их взаимодействия;
- разделяемых и критических ресурсов;
- критического интервала (участка) и конфликтных ситуаций, возникающих при использовании общих ресурсов;
- основных базовых функций управления взаимодействием параллельными процессами и механизмах организации взаимодействия процессов для решения задач их координации.

Понятие процесса

В литературе по аппаратному и программному обеспечению вычислительных систем (ВС) существует множество неформальных определений процесса. В данном разделе мы используем термин процесс полагая, что интуитивно понимаем его значения, связывая понятие процесса с программой, выполняемой на компьютере. В [61] дано такое неформальное определение процесса: «Последовательный процесс (иногда называемый «ЗАДАЧА») есть работа, производимая последовательным процессом при выполнении программы с ее данными с логической точки зрения каждый процесс имеет свой собственный процессор и программу. В действительности два разных процесса могут разделять одну и ту же программу или один и тот же процессор. Таким образом, процесс не эквивалентен программе, а также это не то же самое что процессор; это пара <процессор, программа> при выполнении».

В нашей работе под понятием процесса будем понимать его неформальное определение, как «программа во время выполнения» [63]. В то

же время мы не будем вдаваться в такие подразделения видов работ в вычислительных системах, для которых, кроме крупных единиц работы определяющих процессы (задачи), определяются и более мелкие единицы работ, для обозначения которых используются термины «поток», «нить» [64].

Последовательные процессы (в дальнейшем просто процессы), которые могут существовать одновременно, называются параллельными. В том случае, если работа таких процессов будет зависеть друг от друга, то говорят, что они взаимодействуют друг с другом, т.е. должны периодически синхронизироваться и выполнять функции обмена данными.

Основные типы (способы) взаимодействия процессов

В большинстве случаев процессы, определяемые программой, взаимодействуют друг с другом в основном двумя способами:

- путем обмена данными между процессами (связь процессов);
- синхронизацией процессов (согласование выполнения процессов).

Как отмечено в [65], эти два способа взаимодействия процессов не являются взаимоисключающими. Действительно, для обмена данными между процессами необходимо выполнить их синхронизацию, а с другой – для синхронизации процессов они обязательно обмениваются данными (хотя бы элементарными сигналами).

Разделяемые и критические ресурсы вычислительных систем

Для современных средств вычислительной техники по сравнению с первыми ВС понятие ресурса стало более универсальным и общим [66]. Ресурсом стали называть всякий объект ВС, который может разделяться внутри ВС. К ресурсам ВС относятся как аппаратные средства (процессоры, память, устройства ввода/ вывода и т.д.), так и информационные и программные средства.

Для повышения эффективности ВС необходимо, чтобы ресурсы ВС использовались несколькими процессами, т.е. они должны быть разделяемыми. Таким образом, разделяемые ресурсы – это ресурсы, которые могут использоваться несколькими процессами.

Под критическим ресурсом понимается ресурс, который допускает обслуживание только одного пользователя за один раз.

Следует отметить, что кроме физических ресурсов (память, печать и др.), которые являются критическими, критическими ресурсами могут быть и разделяемые переменные, значения которых могут менять несколько процессов. Например, пусть два процесса А и В разделяют переменную СЧЕТЧИК. Если А и В пытаются увеличить счетчик на единицу одновременно, то окончательное значение переменной СЧЕТЧИК может

оказаться неверным. Поэтому СЧЕТЧИК в данном случае следует рассматривать как критический ресурс.

Основные базовые функции управления взаимодействующими процессами

Если несколько процессов хотят пользоваться критическим ресурсом в режиме разделения, им следует синхронизировать свои действия таким образом, чтобы этот ресурс всегда находился в распоряжении только одного из них. Для такого режима работы процессов необходимо, чтобы система управления процессами обеспечивала функцию «**взаимоисключение**» для процессов, которые могут пользоваться критическим ресурсом.

Если один процесс пользуется в данный момент критическим ресурсом, то все остальные процессы, которым нужен этот ресурс, временно получают отказ и должны ждать, пока он не освободиться. Для реализации такого режима работы необходимо, чтобы система управления процессами обеспечивала функцию «**ожидания**».

Таким образом, к основным базовым функциям управления взаимодействующими процессами относятся: функция «**взаимоисключение**» процессов и функция «**ожидание**».

Критический интервал (или критический участок)

Критический интервал – это группа операций (или событий) процесса, которые непосредственно организуют обращение к разделяемому критическому ресурсу. Критические интервалы должны обладать важнейшим свойством – они должны быть **взаимоисключающими**. Это означает, что в каждый момент времени не более чем один процесс может быть занят выполнением своего критического, относительно некоторого ресурса, участка. В том случае, если критические интервалы для взаимодействующих процессов формализуются с использованием систем канонических уравнений, то события, обеспечивающие вход процессов в свои критические интервалы, должны быть **несовместимыми**.

Конфликтные ситуации

Конфликтные ситуации могут возникнуть в ВС при монопольном захвате и использовании общих ресурсов для нескольких параллельных процессов. К числу конфликтных ситуаций относятся: взаимоблокировка и взаимоотталкивание.

Взаимоблокировка может возникнуть вследствие конкуренции процессов при захвате разделяемых ресурсов. Простейшим примером возникновения взаимоблокировки является ситуация, когда два взаимодействующих процесса разделяют два ресурса, причем каждому из них для нормального функционирования требуется сразу оба ресурса [67]. В

этом случае, если каждый из процессов захватит по одному ресурсу и будет ожидать освобождения второго, захваченного другим процессом, то ни один из них не сможет продолжить свою работу. Таким образом, взаимоблокировка проявляется в том, что ни один из взаимодействующих процессов не в состоянии продолжить свою работу.

Взаимоотталкивание может возникнуть, когда один процесс (или группа процессов) не может продолжить свою работу вследствие постоянной занятости требуемых ресурсов, которые попеременно захватываются другими процессами [67]. Ситуация, связанная с взаимоотталкиванием, похожа на ситуацию взаимоблокировки (в обоих случаях конфликтные ситуации возникают как результат конкуренции процессов из-за ресурсов), но в отличие от случая взаимоблокировки блокируются лишь некоторые из взаимодействующих процессов, другие же процессы могут продолжать нормальную работу. Таким образом, взаимоотталкивание проявляется в том, что один или несколько процессов из взаимодействующих процессов не в состоянии продолжать свою работу, тогда как другие процессы продолжают корректно функционировать.

Механизмы взаимодействия (или средства взаимодействия) процессов, используемые для решения задач их координации

Для организации взаимодействия процессов, основными типами которой является синхронизация и обмен сообщениями, используются следующие механизмы [68]:

- Механизм семафорной техники, который обеспечивает взаимоисключающий доступ нескольких процессов к разделяемым ресурсам.
- Организация специальных областей памяти (разделяемая память), в которые несколько процессов могут записать и считать информацию, осуществляя, таким образом, взаимодействие.
- Механизм обмена сообщениями между процессами на уровне специальных сигналов.
- Механизм организации очередей процессов, ожидающих обслуживания
- Механизм, обеспечивающий задержку выполнения процессов до наступления некоторого события.

В заключении отметим, что проблему обмена информацией и синхронизации процессов решают в современных ВС в основном программным способом с использованием различных механизмов. Такое решение вопросов управления межпроцессным взаимодействием имеет следующие недостатки: недостаточная надежность ВС и их защита от возможного несанкционированного доступа [62,69]; требуется значительный объем оперативной памяти и сравнительно большое время для реализации таких взаимодействий, что уменьшает производительность ВС. Эти недостатки особенно усугубляются, когда многопроцессорные ВС используются для управления технологическими процессами и объектами в реальном масштабе времени.

В связи с этим возникает задача реализации таких функций управления процессами на аппаратном или микропрограммном уровне в виде аппаратной поддержки операционных систем, что в свою очередь ставит задачу формального описания алгоритмов управления взаимодействующими процессами [62, 81-83]. Для решения этих задач в последующих разделах учебного пособия предлагается один из возможных вариантов формализации алгоритмов управления взаимодействующими процессами, основанный на использовании методов теории недетерминированных автоматов.

6.2. Формализация простейших базовых структур управления (управляющих конструкций) взаимодействующими процессами

Для формализации алгоритмов управления взаимодействующими процессами будем использовать модели НДА, представленные в виде ГСАП и НДСКУ. В данном разделе рассмотрим формализацию некоторых простейших базовых структур управления, на основе которых можно формализовать более сложные управляющие конструкции, в том числе и базовые функции управления взаимодействующими процессами, к числу которых, как было отмечено ранее, относятся две функции: функция «ожидание» и функция «взаимоисключение».

Функция «ожидание» включает в себя ожидание некоторого события, которое будет причиной (условием) прекращения существования другого события или наоборот будет причиной (условием) появления другого события.

Функция «взаимоисключение» обеспечивает (гарантирует) однозначность параллельных алгоритмов. Формализация этой функции основывается на понятиях несовместимости событий; принадлежащих разным параллельным ветвям (процессам) алгоритма управления процессами и обеспечивающих вход в свои критические интервалы.

Формализация простейших базовых структур управления основывается на использовании представления любого события S_j модели НДА в виде формулы, входящей в систему рекуррентных предикатных выражений вида (1.12):

Варьируя составом выражений для R_i и R_j , входящих в формулы определяющие событие R_j , можно влиять на условие зарождения и сохранения события S_j . Учитывая эти обстоятельства можно выполнять описание событий, которые будут использованы для формализации простейших управляющих конструкций. К числу простейших управляющих конструкций, которые будут использованы в дальнейшем, относятся: управление длительностью сохранения события и его зарождения (первоначального появления), представление несовместимости событий, принадлежащим различным параллельным ветвям и условий их реализации.

Сохранение события S_j

Если событие S_j после его зарождения необходимо сохранять до наступления или исчезновения другого события, например S_p , то описание события S_j будет иметь вид:

$$S_j(t+1) = S_{j,3} \vee S_j \bar{S}_p \quad \text{или} \quad S_j(t+1) = S_{j,3} \vee S_j S_p \quad (6.1)$$

где $S_{j,3}$ - сокращенное обозначение описания условия зарождения

события S_j .

Задержка появления (или зарождения) события S_j

Если появление события S_j необходимо задержать до появления или исчезновения некоторого события S_n , то описание события S_j можно представить в следующем виде:

$$S_j(t+1) = S_{j,3} \cdot S_n \vee S_j \cdot S_{j,c} \quad \text{или} \quad S_j(t+1) = S_{j,3} \bar{S}_n \vee S_j S_{j,c} \quad (6.2)$$

где $S_{j,c}$ - сокращенное обозначение описания условий сохранения события S_j .

Описание событий в соответствии с формулами (6.1) и (6.2) является основой для формализации функции «ожидание», являющейся одной из базовых функций управления взаимодействующими процессами.

Формализация несовместимости двух событий S_i и S_j при условии их одновременного зарождения

Если события S_i и S_j , принадлежащие двум параллельным ветвям алгоритма управления, несовместимы друг с другом и если не допускается их одновременного зарождения, то описание таких событий будет иметь вид:

$$S_i(t+1) = S_{i,3} \bar{S}_j \vee S_i S_{i,c}, \quad (6.3)$$

$$S_j(t+1) = S_{j,3} \bar{S}_i \vee S_j S_{j,c}$$

Из уравнений (6.3) следует, что одновременное появление событий $S_{i,3}$ и $S_{j,3}$, определяющих зарождение событий S_i и S_j соответственно, приведет к тупиковому результату.

Формализация несовместимости двух событий S_i и S_j , когда события, определяющие их зарождение, могут появляться одновременно

Если для двух несовместимых событий S_i и S_j , принадлежащих двум параллельным ветвям алгоритма управления, условия их зарождения могут появиться в один и тот же момент времени, то описанию таких событий будут соответствовать уравнения (6.3), но с учетом относительного приоритета одного события над другим. Тогда описание таких событий, предотвращающее конфликтную ситуацию, при условии более высокого приоритета, например, события S_i над событием S_j , будет иметь вид:

$$S_i(t+1) = S_{i,3} \bar{S}_j \vee S_i S_{i,c} \quad (6.4)$$

$$S_j(t+1) = S_{j,3} \bar{S}_i \bar{S}_{i,3} \vee S_j S_{j,c},$$

где введение в первую часть второго уравнения отрицания события $S_{i,3}$, учитывающего условие приоритетности события S_i над событием S_j , обеспечивает отсутствие конфликтной ситуации.

Реализация несовместимых событий, принадлежащих различным параллельным ветвям алгоритма управления

Из системы уравнений (6.4) следует, что из двух событий S_i и S_j в любой момент времени может существовать только одно из них, т.е. такие события могут быть реализованы только последовательно.

Поэтому, если по условиям алгоритма управления требуется, чтобы после реализации алгоритмического процесса (или его части), например, в первой ветви по инициативе события S_i , имеющего более высокий приоритет, был бы реализован также и алгоритмический процесс (или его часть) во второй ветви по инициативе события S_j , то необходимо событие $S_{j,3}$ после его зарождения сохранить. Эти условия учитываются в описании события $S_{j,3}$ следующим образом:

$$S_{j,3}(t+1) = [S_{j,3}(0) \vee S_j] \bar{S}_j. \quad (6.5a)$$

То же самое выполняется и для описания события $S_{i,3}$, если событие S_i может зародиться после события S_j . Тогда имеем:

$$S_{i,3}(t+1) = [S_{i,3}(0) \vee S_{i,3}] \bar{S}_i, \quad (6.5б)$$

где $S_{i,3}(0)$, $S_{j,3}(0)$ - сокращенное обозначение условий первоначального появления событий $S_{i,3}$ и $S_{j,3}$ соответственно.

Из системы уравнений (6.5a) и (6.5б) следует, что события $S_{j,3}$ и $S_{i,3}$ сохраняются до появления событий S_j и S_i соответственно.

Описания событий в соответствии с формулами (6.4), (6.5a) и (6.5б) являются основой для формализации функции «**взаимоисключение**» процессов, являющейся второй базовой функцией управления взаимодействующими процессами.

6.3.Формализация функций взаимoisключения критических интервалов (участков), обеспечивающих доступ к общим разделяемым данным (общему ресурсу) для двух процессов

В том случае, если параллельным процессам в соответствии с алгоритмом их работы требуется обратиться к общим разделяемым данным (критическим ресурсом), то программа их работы должна включать специальные участки, содержащие операторы, обеспечивающие доступ к разделяемым данным. Эти участки, называемые критическими, должны обладать важнейшим свойством – они должны быть взаимоисключаемыми. В технической литературе такую группу операторов, обеспечивающих функцию взаимоисключения критических участков, часто называют **примитивами взаимоисключения** [62].

Взаимоисключение критических участков должно обеспечивать выполнение такого алгоритма взаимодействия параллельных процессов, когда один из рассматриваемых процессов выполняет обращение к разделяемым данным, то все остальные процессы, которым требуется обращение к тем же разделяемым данным в то же самое время, должны ждать. Только после завершения рассматриваемым процессом обращения к разделяемым данным, будет разрешен доступ к разделяемым данным одному из ожидающих процессов. Какому именно процессу будет разрешен доступ к разделяемым данным определяется принятыми условиями приоритетности процессов.

Следует обратить внимание, что критический участок не является последовательностью операторов программы, он является только последовательностью **действий**, которые выполняются этими операторами. На это обстоятельство особое внимание было обращено в [70]. Исходя из этого, можно утверждать, что несколько процессов, которые могут иметь место при выполнении одной и той же программы, могут выполнять критические интервалы, базирующиеся на одной и той же последовательности операторов программы.

В данном разделе с методической точки зрения рассматривается взаимодействие параллельных процессов при обращении к разделяемым данным только для двух процессов, а в последующем разделе, базируясь на полученные результаты, делается обобщение для $n > 2$ процессов.

Действительно, как мы убедимся в дальнейшем, структура основных уравнений, определяющих входы процессов в свои критические участки, остается одной и той же для любого числа конкурирующих процессов. Отметим, что программная реализация механизма взаимоисключения для двух процессов была впервые предложена голландским математиком Деккером, а для $n > 2$ процессов программную реализацию механизма взаимоисключения впервые предложил Дейкстра [62]. Им же был предложен механизм семафоров для реализации примитивов взаимоисключения.

Для того, чтобы избежать конфликтных ситуаций при взаимодействии параллельных процессов во время организации доступа к разделяемым данным, на критические участки налагаются следующие основные требования [62, 70]:

1. В любой момент времени только один процесс может находиться внутри критического интервала.
2. Ни один процесс не может оставаться внутри критического интервала бесконечно долго.
3. Ни один процесс не должен ждать бесконечно долго входа в критический интервал.

Первое требование, предъявляемое к критическим участкам должно обеспечиваться взаимоисключением (несовместимостью) событий, определяющих как входы процессов в свои критические участки при организации доступа к разделяемым данным, так и нахождение процессов в своих критических участках с учетом принятой дисциплины приоритетности процессов.

Второе требование, предъявляемое к критическим участкам, должно обеспечиваться таким описанием условий сохранения событий, определяющих входы процессов в свои критические интервалы, для которых эти события будут существовать, пока не закончится процедура одноразового обращения к разделяемым данным. Учитывая эти замечания относительно обеспечения первого и второго требований, предъявляемых к критическим участкам, а также результат формализации несовместимых событий (6.4), система уравнений для событий, определяющих входы процессов в свои критические участки при обращении к разделяемым данным будут иметь следующий вид:

$$\begin{aligned} S_k^1(t+1) &= S_1 \bar{S}_k^2 \vee S_k^1 \bar{S}_p^1, \\ S_k^2(t+1) &= S_2 \bar{S}_k^1 \bar{S}_1 \vee S_k^2 \bar{S}_p^2, \end{aligned} \quad (6.6)$$

где S_1 и S_2 - сокращенное обозначение событий, определяющих прием заявок первого и второго процессов на обслуживание для обращения к разделяемым данным (событие пассивного ожидания обращения к разделяемым данным);

S_k^1 и S_k^2 - сокращенное обозначение событий, определяющих входы первого и второго процессов в свои критические участки; эти события при их истинности свидетельствуют также о том, что соответствующие процессы находятся внутри своих критических участков;

S_p^1 и S_p^2 - сокращенное обозначение событий, обеспечивающих выход первого и второго процессов из критического участка после окончания процедуры обращения к разделяемым данным.

Третье требование, предъявляемое к критическим участкам, должно обеспечиваться такой формализацией событий, определяющих прием заявок на обслуживание при обращении к разделяемым данным, когда заявка какого-либо процесса на обслуживание воспринимается только в том случае,

когда данный процесс не находится в своем критическом участке. Тем самым должны исключаться повторные обращения к разделяемым данным для процесса, имеющего наивысший приоритет, несмотря на наличие заявки на обслуживание другого процесса, имеющего более низкий приоритет. Учитывая эти замечания, а также результат формализации несовместимых событий (6а) и (6б), система уравнений для событий, определяющих прием заявок двух процессов на обслуживание при обращении к разделяемым данным, будет иметь следующий вид:

$$\begin{aligned} S_1(t+1) &= (S_{1,3} \vee S_1) \bar{S}_k^1, \\ S_2(t+1) &= (S_{2,3} \vee S_2) \bar{S}_k^2, \end{aligned} \quad (6.7)$$

где $S_{1,3}$ и $S_{2,3}$ - сокращенное обозначение событий, свидетельствующих о поступлении заявок первого и второго процессов на обслуживание для обращения к разделяемым данным.

В заключение обратим внимание, что полученные в результате формализации функции взаимоисключения критических участков (интервалов), представленные системами уравнений (6.6) и (6.7) являются основой для формализации алгоритмов управления параллельными процессами при решении многих задач межпроцессного взаимодействия. Решение некоторых из таких задач будет рассмотрено в последующих разделах.

6.4. Формализация алгоритма управления взаимодействующими параллельными процессами при обращении к разделяемым данным (общему ресурсу)

Рассматриваемый алгоритм управления взаимодействующими параллельными процессами является важнейшей составной частью решения многих «классических» задач, посвященных проблемам межпроцессного взаимодействия при использовании разделяемых ресурсов. К числу таких «классических» задач, варианты решения которых будут рассматриваться в последующих разделах, относятся задачи: «производители-потребители», «читатели-писатели» и «обедающие философы».

Решение задачи формализации рассматриваемого алгоритма управления взаимодействующими параллельными процессами при обращении к разделяемым данным основывается на результатах формализации функции взаимоисключения критических участков (раздел 6.3).

Напомним, что основное требование к алгоритму взаимодействия параллельными процессами при организации доступа к разделяемым данным заключается в том, чтобы во время обращения одного процесса к разделяемым данным всем другим процессам это было бы запрещено. Для того, чтобы избежать конфликтных ситуаций при таком взаимодействии

параллельных процессов необходимо, чтобы алгоритм взаимодействия удовлетворял бы требованиям, предъявляемым к критическим участкам (раздел 6.3).

Так же как в разделе 6.3 с методической точки зрения будем сначала рассматривать алгоритм управления только для двух процессов [2].

6.4.1. Формализация алгоритма управления взаимодействующими параллельными процессами при обращении к разделяемым данным для $n=2$

Представим для наглядности общую структуру рассматриваемого алгоритма управления параллельными процессами в виде графа НДА (рис.6.1), на котором для простоты опущена некоторая информация о логических условиях для отдельных переходов.

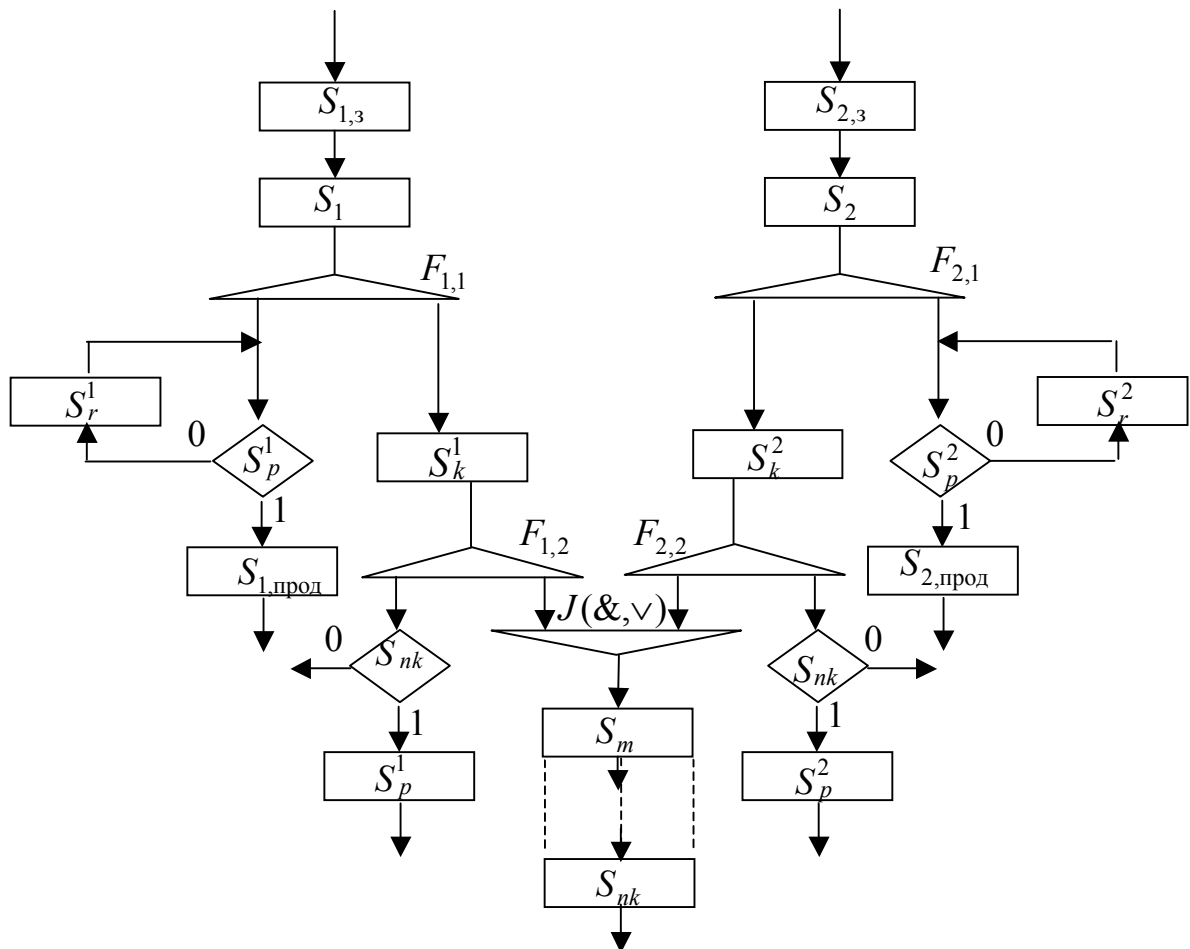


Рис.6.1. Фрагмент графа НДА алгоритма управления взаимодействующими параллельными процессами при обращении к разделяемым данным (общему критическому ресурсу) для 2-х процессов

Учитывая рассматриваемые требования к алгоритму взаимодействия параллельными процессами при обращении к разделяемым данным и результаты формализации взаимоисключения критических участков, представленных в виде уравнений (6.6) и (6.7), систему канонических уравнений, формализующих рассматриваемый алгоритм управления для 2-х процессов можно представить, состоящей из следующих 3-х частей:

Для 1-ой ветви:

Для 2-ой ветви:

$$S_1(t+1) = (S_{1,3} \vee S_1) \bar{S}_k^1,$$

$$S_2(t+1) = (S_{2,3} \vee S_2) \bar{S}_k^2,$$

$$S_k^1(t+1) = S_1 \bar{S}_k^2 \vee S_k^1 \bar{S}_p^1,$$

$$S_k^2(t+1) = S_2 \bar{S}_k^1 \bar{S}_1 \vee S_k^2 \bar{S}_p^2,$$

$$S_p^1(t+1) = S_{nk} S_k^1,$$

$$(6.8a) \quad S_p^2(t+1) = S_{nk} S_k^2,$$

$$(6.8b)$$

$$S_r^1(t+1) = (S_1 \vee S_r^1) S_p^1,$$

$$S_r^2(t+1) = (S_2 \vee S_r^2) \bar{S}_p^2,$$

$$S_{1,прод}(t+1) = S_r^1 S_p^1,$$

$$S_{1,прод}(t+1) = S_r^2 S_p^2.$$

Для последовательной компоненты алгоритма управления:

$$S_m(t+1) = S_k^1 S_1 \vee S_k^2 S_2,$$

(6.8в)

$$S_{nk}(t+1) = S_m,$$

где переменные $(S_{1,3}$ и $S_{2,3})$, $(S_1$ и $S_2)$ и $(S_p^1$ и $S_p^2)$ имеют тот же смысл, что и в разделе 6.3 при рассмотрении вопросов взаимоисключения критических участков;

S_m - сокращенное обозначение события, обеспечивающего начало процедуры реализации обращения к разделяемым данным (критическому ресурсу);

S_{nk} - сокращенное обозначение события, являющееся заключительным в процедуре реализации обращения к разделяемым данным;

$(S_r^1$ и $S_r^2)$ - сокращенное обозначение событий, символизирующих ожидание условия выхода первого и второго процессов из критического участка после окончания процедуры обращения к разделяемым данным;

$(S_{1,прод}$ и $S_{2,прод})$ - сокращенное обозначение событий, инициирующих продолжение работы первого и второго процессов после окончания процедуры обращения к разделяемым данным.

Событие S_m будет истинным тогда, когда в непосредственно предшествующий момент времени будут истинными условия выхода алгоритмического процесса за комбинированный соединитель $J(\&,v)$. Истинность этих условий определяется наличием принятой заявки на обслуживание процесса и нахождения этого процесса в своем критическом интервале $(S_k^i S_i = 1)$. Учитывая это обстоятельство формальное представление события S_m будет соответствовать выражению (6.8в).

Для простоты рассмотрения вопросов реализации процедуры обращения к разделяемым данным будем считать, что для выполнения этой процедуры необходимо только два оператора (два события S_m и S_{nk}).

Проследим работу рассматриваемого алгоритма управления параллельными процессами по ГСАП (рис.6.1) и системе уравнений (6.8а, 6.8б, 6.8в). Для определенности был принят приоритет первого процесса выше приоритета второго процесса.

Будем рассматривать самый наихудший вариант межпроцессного взаимодействия, когда оба процесса одновременно достигли операторов, определяющих заявки на доступ к разделяемым данным (события $S_{1,3}=S_{2,3}=1$). Тогда воспринимается для реализации заявка того процесса, который не находится в данный момент времени в критическом участке. В том случае, если оба процесса не находятся в критическом участке, то воспринимаются обе заявки ($S_1=S_2=1$), т.е. оба процесса будут находиться в режиме пассивного ожидания доступа к разделяемым данным. При этом будет открыт доступ к разделяемым данным для того процесса, который имеет наивысший приоритет

Так как для нашего предположения наивысшим приоритетом обладает первый процесс, то будет истинным событие $S_k^1 (S_k^1=1)$. Одновременно с зарождением события S_k^1 первый процесс приостанавливает свою работу и ждет окончания процедуры обращения к разделяемым данным. Это обеспечивается зарождением события S_r^1 и его сохранением ($S_r^1=1$) до тех пор, пока не закончится процедура обращения к разделяемым данным и первый процесс выйдет из критического участка при $S_p^1=1$. При $S_r^1 S_p^1=1$ первый процесс, приостановленный на период обращения к разделяемым данным, продолжит свою работу ($S_{1,прод}=1$).

В том случае, когда были восприняты заявки на обслуживание от обоих процессов, то после выхода первого процесса из критического участка ($S_k^1=0$) и в связи с тем, что событие S_1 к этому времени также будет равно нулю, зародится событие S_k^2 , определяющее вход второго процесса в критический участок. После этого работа алгоритма управления доступом второго процесса к разделяемым данным будет такой же, как и работа алгоритма управления для первого процесса.

Для большей наглядности при изучении работы алгоритма управления взаимодействующими параллельными процессами при обращении к разделяемым данным можно воспользоваться временными диаграммами, отражающими организацию синхронизации процессов, которая реализуется событиями, представленными в управляющем алгоритме.

S_1	S_1	S_1					S_1	S_1	S_1	S_1	S_1	S_1
S_k^1		S_k^1	S_k^1	S_k^1	S_k^1							S_k^1

S_m			S_m					S_m				
S_{nk}				S_{nk}					S_{nk}			
$S_p^1(S_p^2)$					S_p^1					S_p^2		
S_r^1		S_r^1	S_r^1	S_r^1	S_r^1			S_r^1	S_r^1	S_r^1	S_r^1	S_r^1
$S_{1, прод}$						$S_{1,n}$						
S_2	S_2	S_2	S_2	S_2	S_2	S_2	S_2					S_2
S_k^2							S_k^2	S_k^2	S_k^2	S_k^2		
S_r^2		S_r^2	S_r^2	S_r^2	S_r^2	S_r^2	S_r^2	S_r^2	S_r^2	S_r^2		
$S_{2, прод}$											$S_{2,n}$	

Рис.6.2. Временные диаграммы работы алгоритма управления двумя параллельными процессами при обращении к разделяемым данным

На рис.6.2 представлены временные диаграммы для всех событий, реализуемых в алгоритме управления двумя параллельными процессами при обращении к разделяемым данным. Временные диаграммы построены по уравнениям (6.8а, 6.8б, 6.8в), представляющих систему НД СКУ для наиболее худшего варианта межпроцессного взаимодействия, когда заявки на обслуживание восприняты от обоих процессов одновременно. Причем после восприятия заявок они опять поступили на обслуживание.

Такие временные диаграммы для разных комбинаций заявок на обслуживание позволят осуществить проверку правильности работы алгоритма управления, представленного системой НД СКУ для различных комбинаций поступлений заявок на обслуживание. Действительно, т.к. для формализации алгоритма управления взаимодействующими параллельными процессами используется модель НДА, то на временных диаграммах можно наблюдать взаимное временное расположение отдельных событий, их возможное одновременное существование и моменты их зарождения и сохранения.

Для построения временных диаграмм работы алгоритма управления межпроцессного взаимодействия для большего числа процессов можно использовать систему моделирования СОМПА (78).

6.4.2. Формализация алгоритма управления взаимодействующими параллельными процессами при обращении к разделяемым данным (общему ресурсу) для n -процессов

Рассматриваемая в данном разделе методика формализации алгоритма управления взаимодействующими параллельными процессами для

организации обращения к разделяемым данным, когда число процессов $n > 2$, базируется на следующих предпосылках:

- Все основные требования к алгоритму управления параллельными процессами не должны зависеть от количества процессов, участвующих в межпроцессных взаимодействиях.
- Алгоритм управления взаимодействующими параллельными процессами должен строиться на основе использования результатов формализации функции взаимоисключения критических участков и на основе учета принятой приоритетности процессов по доступу к разделяемым данным.
- Система уравнений, формализующих алгоритм управления в виде НД СКУ для всех событий, реализуемых в алгоритме управления, должна иметь одинаковую структуру для обслуживания любого i -го процесса.

Напомним основные требования к организации межпроцессного взаимодействия при обращении к разделяемым данным.

Если несколько параллельных процессов, более 2-х, хотят пользоваться некоторым критическим ресурсом, под которым понимается ресурс, допускающий обслуживание только одного пользователя за один раз в режиме разделения, то им следует, во избежание конфликтных ситуаций, синхронизировать свои действия таким образом, чтобы этот ресурс всегда находился в распоряжении не более чем одного из них. Все другие процессы должны переводиться в режим ожидания обращения к общему ресурсу.

Учитывая основные требования к организации межпроцессного взаимодействия при обращении к разделяемым данным и основные подходы для решения задачи формализации алгоритма управления процессами (для $n > 2$), граф НДА, представляющий этот алгоритм (рис.6.3), будет иметь ту же структуру, что и граф НДА для 2-х процессов (рис.6.1).

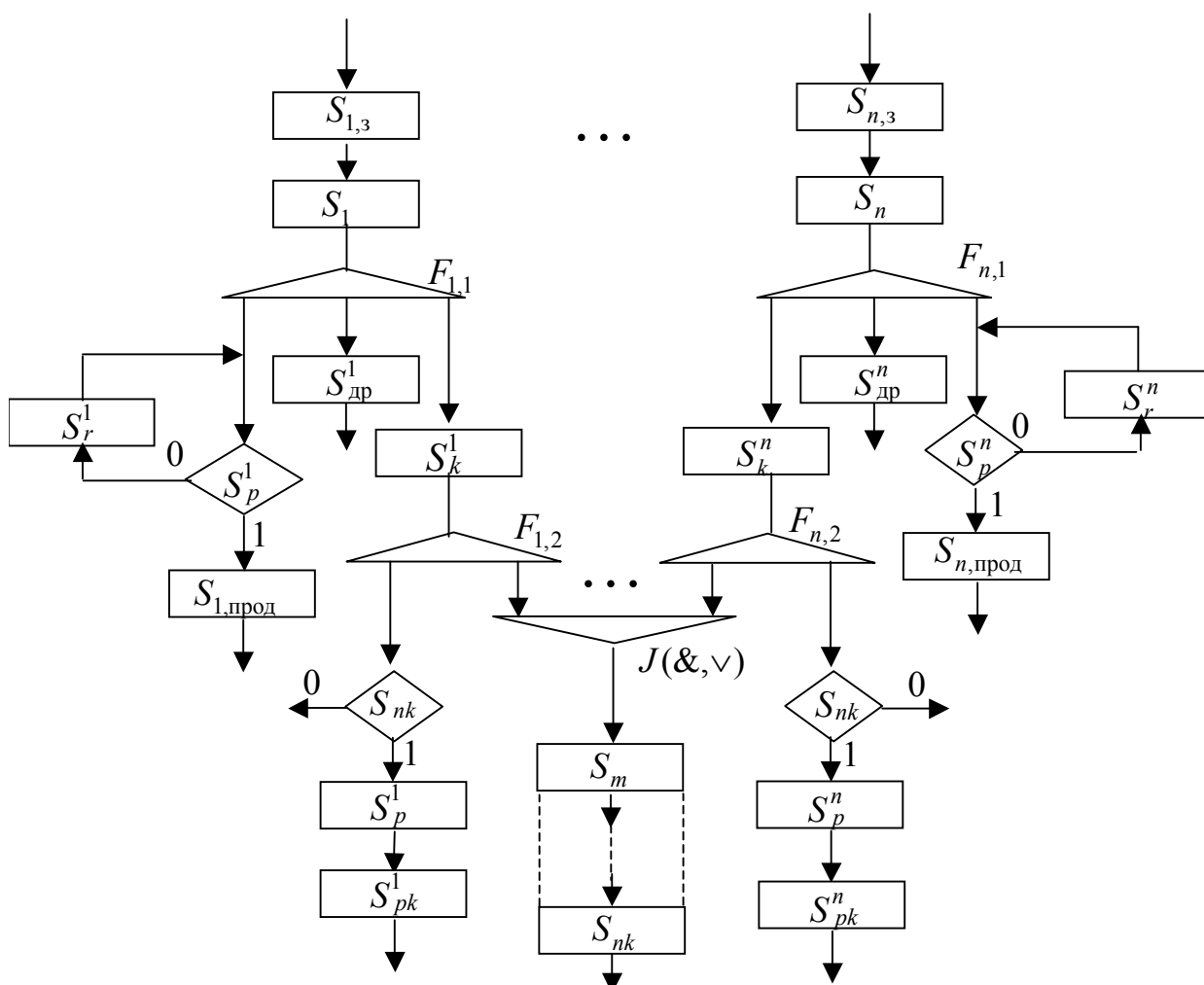


Рис.6.3. Фрагмент графа НДА алгоритма управления взаимодействующими параллельными процессами при обращении к разделяемым данным для $n > 2$ процессов

В соответствии с графом НДА (рис.6.3) и учитывая системы канонических уравнений (6.8а), (6.8б) и (6.8в), формализующих алгоритм управления межпроцессного взаимодействия для 2-х процессов, система канонических уравнений, формализующая алгоритм управления межпроцессного взаимодействия для $n > 2$ процессов для обслуживания любого i -го процесса и для последовательной части алгоритма управления, будут иметь следующий вид [71]:

$$\begin{aligned}
S_i(t+1) &= (S_{i,3} \vee S_i) \bar{S}_k^i, \\
S_k^i(t+1) &= S_i S_{\text{вз}}^i S_{\text{пр}}^i \vee S_k^i \bar{S}_p^i, \\
S_p^i(t+1) &= S_{nk} S_k^i, \\
S_r^i(t+1) &= (S_i \vee S_r^i) \bar{S}_p^i, \\
S_{i,\text{прод}}(t+1) &= S_r^i S_p^i, \\
S_m(t+1) &= S_k^1 S_1 \vee S_k^2 S_2 \vee \dots \vee S_k^n S_n, \\
S_{nk}(t+1) &= S_m,
\end{aligned} \tag{6.9}$$

где переменные $S_{i,3}, S_i, S_k^i, S_p^i, S_{nk}, S_r^i, S_m, S_{i,\text{прод}}$ имеют тот же смысл, что и аналогичные переменные в системах уравнений (6.8а), (6.8б), (6.8в);

$S_{\text{вз}}^i$ - комбинационное событие, обеспечивающее взаимоисключение критических интервалов на основе несовместимости события S_k^i с другими событиями из общего их числа n . Т.к. для всех событий типа S_k^i должно выполняться условие: $S_k^1 \bar{S}_k^2 \dots \bar{S}_k^n \vee \bar{S}_k^1 S_k^2 \bar{S}_k^3 \dots \bar{S}_k^n \vee \dots \vee \bar{S}_k^1 \bar{S}_k^2 \dots \bar{S}_k^{n-1} S_k^n = 1$, то событие $S_{\text{вз}}^i$ определится следующей формулой:

$$S_{\text{вз}}^i = \bigwedge_{(\forall \alpha)(\alpha \neq i)} \bar{S}_k^\alpha, i = \overline{1, n}, \tag{6.10}$$

$S_{\text{пр}}^i$ - сокращенное обозначение события, обеспечивающего заданное приоритетное обслуживание i -го процесса при обращении к разделяемым данным.

В качестве примера в данном разделе принята циклическая дисциплина обслуживания (рис.6.4).

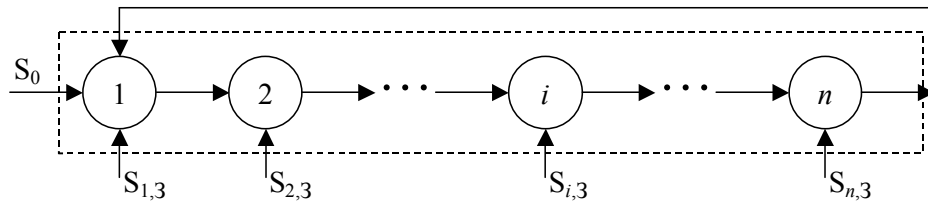


Рис.6.4. Схема циклической дисциплины обслуживания

Для такой дисциплины $S_{\text{пр}}^i$ может быть представлена следующим выражением:

$$S_{\text{пр}}^i(t+1) = S_{\text{пр}}^i(0) \vee S_{i,3} [S_{pk}^i \Lambda_{(\forall \alpha)(\alpha \neq i)} \bar{S}_{\alpha,3} \vee S_{pk}^{i \oplus 1} \Lambda_{(\forall \alpha)(i \oplus 1 < \alpha \leq i-1)} \bar{S}_{\alpha,3} \vee S_{pk}^{i \oplus 2} \Lambda_{(\forall \alpha)(i \oplus 2 < \alpha \leq i-1)} \bar{S}_{\alpha,3} \vee \dots \vee S_{pk}^{i \oplus (n-1)} \Lambda_{(\forall \alpha)(i \oplus n-1 < \alpha \leq i-1)} \bar{S}_{\alpha,3}], \quad (6.11)$$

где $\oplus$ - знак сложения по модулю n ;

S_{pk}^i - сокращенное обозначение события, свидетельствующее о факте выполненного обслуживания i -го процесса при обращении к разделяемым данным, которое определится в данном варианте как задержка события S_p^i на один такт:

$$S_{pk}^i(t+1) = S_p^i. \quad (6.12)$$

П р и м е ч а н и е: Отметим некоторые особенности в определении индексов событий S_{pk} и $S_{\alpha,3}$ в выражении (6.11). Если при определении верхнего индекса события S_{pk} в результате сложения по модулю n получили нуль, то в качестве индекса принимается не нуль, а величина n . Если при определении индекса α события $S_{\alpha,3}$ получилось противоречивое неравенство, например, $3 < \alpha \leq 3$, то соответствующее событие $S_{\alpha,3}$ не существует, тогда

$$\bar{S}_{\alpha,3} = 1.$$

$S_{\text{пр}}^i(0)$ - сокращенное обозначение комбинационного события, определяющего начальный приоритет обслуживания i -го процесса.

$$S_{\text{пр}}^i(0) = S_0 x_n S_{i,3} \Lambda_{(\forall \alpha)(\alpha < i)} \bar{S}_{\alpha,3}, \quad (6.13)$$

где S_0 - начальное событие системы управления обращением к разделяемым данным;

$$S_0(t+1) = x_0 \vee S_{pk}^n \vee S_0 \bar{x}_n, \quad (6.14)$$

где x_n - сигнал инициализации системы управления;

x_0 - сигнал приведения системы управления в начальное состояние.

Введение события $S_{\text{пр}}^i(0)$ в условия зарождения события S_k^i , обеспечивающего вход i -го процесса в критический участок, объясняется необходимостью обеспечения функции взаимоисключения на начальном этапе работы алгоритма управления параллельными процессами, когда в один и тот же момент времени могли бы зародиться сразу несколько событий типа S_k^i .

В случае, если в многопроцессорной ВС необходимо использовать время ожидания обращения к разделяемым данным в каком-либо i -м процессе для решения других задач (реализуемых подпроцессом), не требующих обращения к разделяемым данным, то событие, инициирующее выполнение такого подпроцесса можно представить в следующем виде (рис.6.3):

$$S_{др}^i(t+1) = S_i S_{д,3}^i \bar{S}_k^i \vee S_{др}^i \bar{S}_p^i, \quad (6.15)$$

где $S_{д,3}^i$ - сокращенное обозначение события, определяющего заявку на реализацию подпроцесса в i -м процессе.

Пример 6.1. Построить систему уравнений для событий $S_{пр}^i$, определяющих приоритетность доступа к разделяемым критическим ресурсам для 3-х параллельных процессов при циклической дисциплине обслуживания, и событий $S_{вз}^i$, обеспечивающих взаимоисключение критических интервалов.

На основании уравнений (6.10), (6.11) и (6.12) и учитывая при этом примечания относительно определения индексов событий, представленных в них, получим следующие представления искомых событий для $i=1,2,3$:

$$S_{пр}^1(t+1) = S_0 x_n S_{1,3} \vee S_{1,3} [S_{pk}^1 \bar{S}_{2,3} \bar{S}_{3,3} \vee S_{pk}^2 \bar{S}_{3,3} \vee S_{pk}^3],$$

$$S_{пр}^2(t+1) = S_0 x_n S_{2,3} \bar{S}_{1,3} \vee S_{2,3} [S_{pk}^2 \bar{S}_{1,3} \bar{S}_{3,3} \vee S_{pk}^3 \bar{S}_{1,3} \vee S_{pk}^1],$$

$$S_{пр}^3(t+1) = S_0 x_n S_{3,3} \bar{S}_{1,3} \bar{S}_{2,3} \vee S_{3,3} [S_{pk}^3 \bar{S}_{1,3} \bar{S}_{2,3} \vee S_{pk}^1 \bar{S}_{2,3} \vee S_{pk}^2],$$

$$S_{вз}^1 = \bar{S}_k^2 \bar{S}_k^3, \quad S_{вз}^2 = \bar{S}_k^1 \bar{S}_k^3, \quad S_{вз}^3 = \bar{S}_k^1 \bar{S}_k^2.$$

6.5. Формализация алгоритма управления взаимодействующими параллельными процессами в задаче «производители-потребители»

Задача «производители-потребители» является одной из классических задач синхронизации двух асинхронных процессов: процесс производитель (например, процессор) вырабатывает символы (слова), которые должны быть переданы другому процессу – потребителю (например, печать). Так как интенсивность воспроизведения и потребления символов (слов) различная, то требуется обеспечить передачу символов (слов) с согласованием их интенсивностей [67,70,72]. Процессы взаимодействуют через некоторую обобщенную область памяти – согласующий буфер сообщений. В эту область памяти процесс-производитель помещает очередное сообщение, а процесс-потребитель считывает очередное сообщение. В общем случае буфер способен хранить несколько сообщений. Необходимо согласовать работу двух процессов при одностороннем обмене сообщениями по мере развития процессов таким образом, чтобы удовлетворить следующие требования:

- Выполнялись требования задачи взаимного исключения по отношению к критическому ресурсу – буферу сообщений.
- Учитывалось состояние буфера сообщений, характеризующего возможность и невозможность отправки (принятия) очередного сообщения, а именно: процесс-производитель при попытке поместить очередное сообщение в полностью заполненный буфер должен быть полностью заблокирован; попытка процесса-потребителя чтения из пустого буфера также должна быть заблокирована.

Рассмотрим вначале простой вариант, когда процессы взаимодействуют через согласующий буфер в одно слово. Для этого варианта имеет место следующее:

- производитель производит по одному слову (за один раз), а потребитель использует их по одному (за один раз);
- согласующий буфер имеет размер в одно слово.

Для такого простейшего случая взаимодействия – взаимное исключение процессов специально не предусматривается (не формализуется), т.к. в этом случае оно обеспечивается синхронизацией процессов. Фрагмент графа НДА алгоритма взаимодействия двух параллельных процессов представлен на рис.6.5.

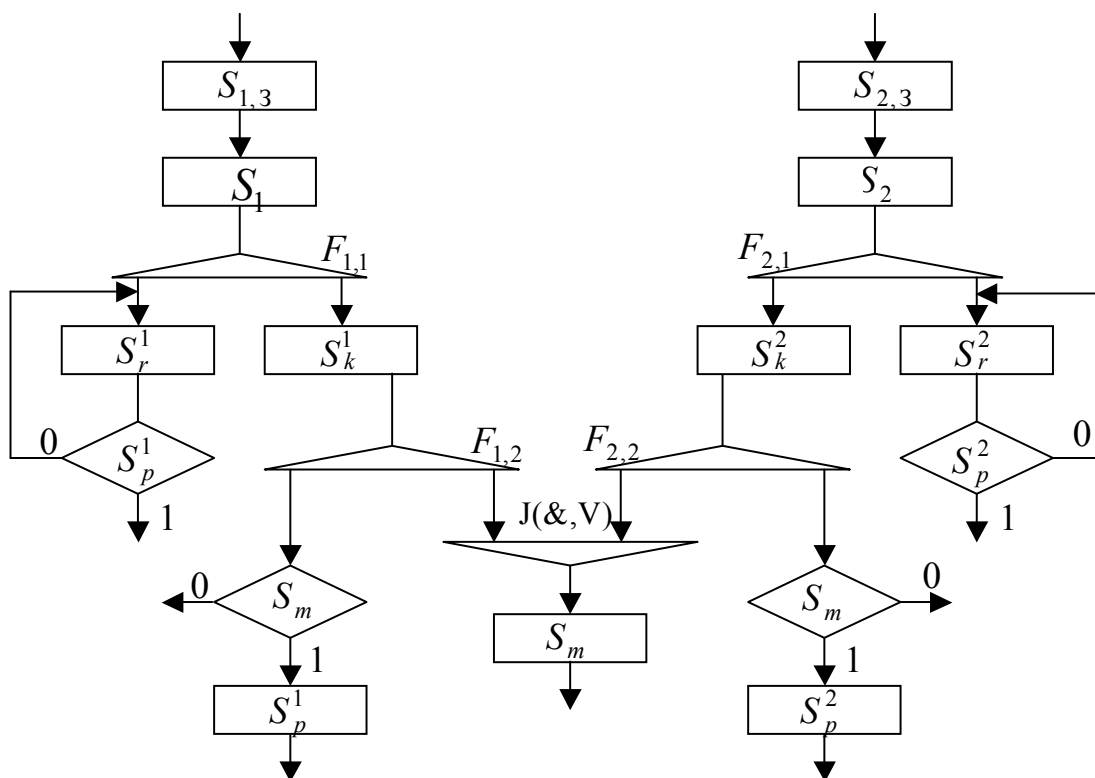


Рис.6.5. Фрагмент графа НДА алгоритма управления параллельными процессами задачи «производитель-потребитель» с согласующим буфером в одно слово

Общая структура графа НДА (рис.6.5) почти ничем не отличается от общей структуры графа НДА, представляющего алгоритм управления двумя параллельными процессами при обращении к разделяемому критическому ресурсу (рис.6.1). Для рассматриваемого алгоритма управления процедура обращения к разделяемому критическому ресурсу (согласующему буферу в одно слово) на графе НДА (рис.6.5) представлена не только общей частью (событие S_m), но и двумя непересекающимися операциями: запись в буфер (событие S_p^1) и чтение из буфера (событие S_p^2). Эти же события (S_p^1 и S_p^2) при их истинности осуществляют реализацию выхода из своих критических участков процесса производителя и процесса потребителя соответственно.

Формальное описание рассматриваемого алгоритма управления процессами будет также в основном соответствовать системам уравнений (6.8а), (6.8б) и (6.8в), но с некоторой корректировкой, учитывающей отмеченные выше конкретные требования к алгоритму управления межпроцессного взаимодействия в задаче «производители-потребители» с учетом наличия согласующего буфера в одно слово. Такое формальное описание, представленное системами канонических уравнений НД СКУ для событий, реализуемых алгоритмом управления, имеют следующий вид:

Для процесса - производителя:

Для процесса - потребителя:

$$\begin{aligned} S_1(t+1) &= (S_{1,3} \vee S_1) \bar{S}_2^1, & S_2(t+1) &= (S_{1,3} \vee S_2) \bar{S}_k^2, \\ S_k^1(t+1) &= S_1 S_{\text{БО}} \vee S_k^1 \bar{S}_p^1, & S_p^1(t+1) &= S_2 \bar{S}_{\text{БО}} \vee S_k^2 \bar{S}_p^2, \\ S_p^1(t+1) &= S_m S_k^1, & S_p^2(t+1) &= S_p^1 S_k^2, \\ S_r(t+1) &= (S_1 \vee S_r^1) \bar{S}_p^2, & S_r^2(t+1) &= (S_2 \vee S_r^2) \bar{S}_p^2, \end{aligned} \quad (6.16\text{б})$$

при $t = 0 / S_{\text{БО}} = 1$.

Для последовательной компоненты алгоритма управления:

$$S_m(t+1) = S_k^1 S_1 \vee S_k^2 S_2, \quad (6.16\text{в})$$

где S_m - сокращенное обозначение события, определяющего обращение к согласующему буферу;

S_p^1 и S_p^2 - сокращенные обозначения событий, определяющих операции записи и чтения из буфера, а после выполнения которых обеспечивающих выход из критических участков процесса производителя и процесса потребителя соответственно;

$S_{\text{БО}}$ - сокращенное обозначение события, определяющего состояние буфера (при $S_{\text{БО}} = 1$ буфер пуст);

$S_{1,3}$ и $S_{2,3}$ - сокращенные обозначения событий, определяющих заявку на запись и чтение из буфера соответственно;

S_1 и S_2 - сокращенные обозначения событий, определяющих прием заявок на запись и чтение соответственно;

S_r^1 и S_r^2 - сокращенные обозначения событий, символизирующих ожидание процессами производителя и потребителя окончания операции записи и чтения из буфера соответственно.

Рассмотрим теперь вариант формализации алгоритма управления параллельными процессами в задаче «производители-потребители», когда процессы взаимодействуют через согласующий буфер на число слов $N > 1$. Для этого алгоритма должно быть обеспечено:

- взаимное исключение процессов при обращении к критическому ресурсу – согласующему буферу;
- предотвращение чтения из пустого буфера и записи в полный буфер.

Для определения состояния буфера используется счетчик буфера (СчБ), в который перед операциями записи (чтения) в буфер заносится константа N , определяющая максимальное число слов, записываемых в буфер. При $t=0$ / СчБ := N .

Во время операций записи (чтения) в буфер в СчБ выполняются следующие операции:

- при каждой записи одного слова из СчБ вычитается единица (СчБ := СчБ - 1), поэтому при СчБ = 0 буфер будет полон, а его состояние будем характеризовать событием $S_{\text{БП}}$ (при $S_{\text{БП}} = 1$ буфер полон);
- при каждом чтении из буфера одного слова в СчБ посылается единица (СчБ := СчБ + 1), поэтому при СчБ = N буфер будет пуст, а его состояние будем характеризовать событием $S_{\text{БО}}$ (при $S_{\text{БО}} = 1$ буфер пуст).

Введение двух событий $S_{\text{БП}}$ и $S_{\text{БО}}$ позволяет охарактеризовать промежуточное состояние буфера, когда он может быть и не пустым и не полным ($S_{\text{БП}} = S_{\text{БО}} = 0$). Это может иметь место, когда при очередной записи буфер был заполнен не полностью, т.е. число слов в буфере после очередной записи будет меньше N .

Учитывая рассмотренные выше требования к алгоритму управления межпроцессного взаимодействия для двух параллельных процессов через согласующий буфер с числом слов $N > 1$, а также методику формализации функции взаимного исключения критических участков (раздел 6.3), граф НДА, представляющий этот алгоритм (рис.6.6) будет иметь в основном такую же структуру, что и граф НДА (рис.6.5), представляющий алгоритм управления процессами для варианта, когда согласующий буфер имеет объем в одно слово.

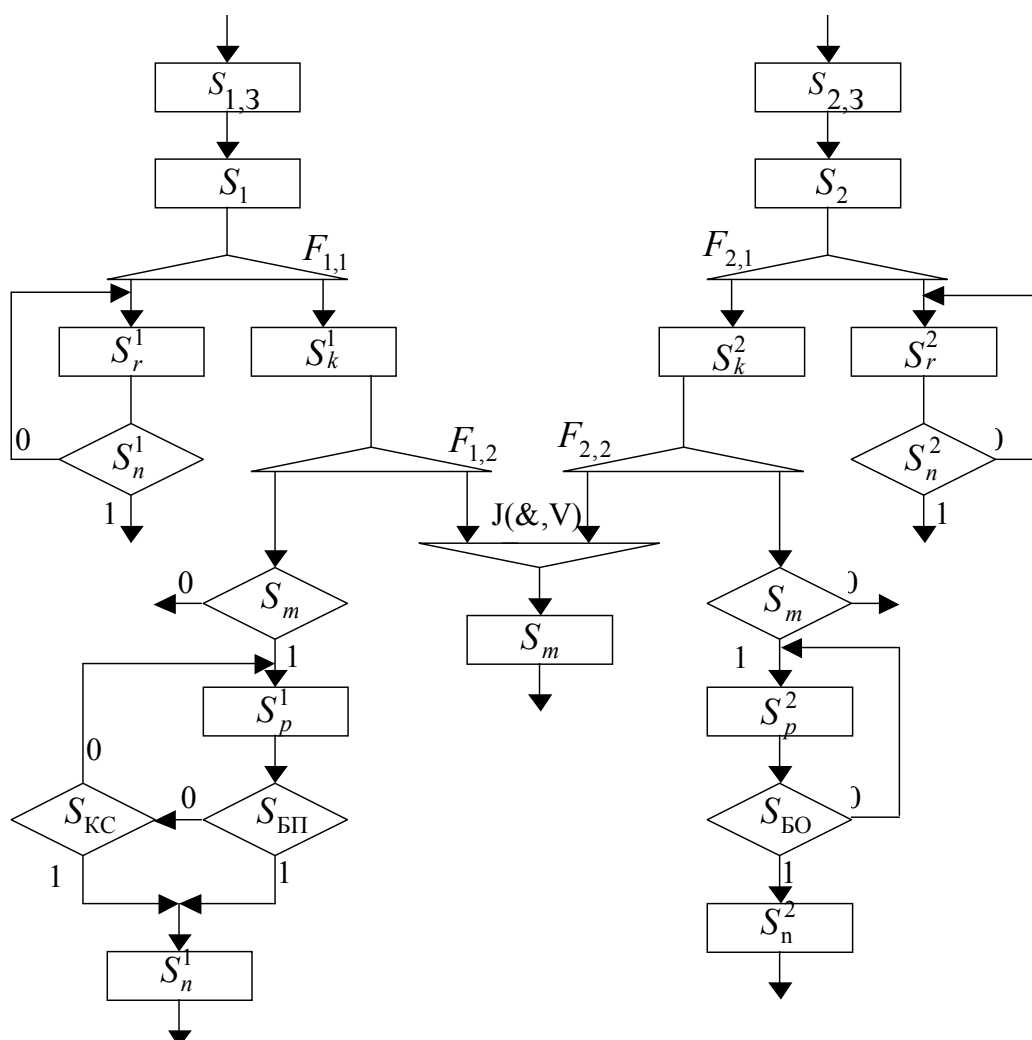


Рис. 6.6. Фрагмент графа НДА алгоритма управления параллельными процессами задачи «производители-потребители» с согласующим буфером в $N > 1$ слов

На графе НДА (рис.6.6) по сравнению с графом НДА (рис.6.5) дополнительно представлены необходимые переходы, связанные с организацией записи и чтения из буфера и выходом взаимодействующих процессов из своих критических участков. Следует также отметить, что на графе НДА (рис.6.6), с целью упрощения его схемы, опущены некоторые логические вершины, определяющие события (S_1 и S_2) и (S_k^1 и S_k^2). Несмотря на указанные упрощения, граф НДА (рис.6.6) в достаточной степени дает представление об общей структуре алгоритма управления взаимодействующими процессами.

Такой граф НДА (рис.6.6), как следует из дальнейшего изложения, совместно с системами канонических уравнений НД СКУ, описывающих все представимые в управляющем алгоритме события, дает полную картину межпроцессного взаимодействия двух параллельных процессов задачи «производители-потребители» для варианта использования согласующего буфера с числом слов $N > 1$.

Перейдем теперь к определению формального описания основных событий, реализуемых в рассматриваемом алгоритме управления процессами и представляемых в дальнейшем в виде системы канонических уравнений НД СКУ. В начале рассмотрим формализацию самых главных событий, определяющих входы процессов в свои критические участки и обозначенные в предыдущих разделах символом S_k^i (где i - номер процесса).

Для нашего варианта задачи «производители-потребители» используются только два взаимодействующих процесса. Для них указанные события, для удобства дальнейшего анализа, представим, на основании системы уравнений (6.9), в следующем общем виде:

$$\begin{aligned} S_k^1(t+1) &= S_{вз}^1 S_{ко}^1 \vee S_k^1 \bar{S}_n^1, \\ S_k^2(t+1) &= S_{вз}^2 S_{ко}^2 \vee S_k^2 \bar{S}_n^2, \end{aligned} \quad (6.17)$$

где S_n^1 и S_n^2 - сокращенные обозначения событий, свидетельствующих о факте окончания процедуры записи и чтения из буфера и обеспечивающих выход процессов производителя и потребителя из своих критических участков соответственно.

Выражения для событий $S_{вз}^1$ и $S_{вз}^2$ определяются, исходя из соотношения (6.10), следующим образом:

$$S_{вз}^1 = \bar{S}_k^2, \quad S_{вз}^2 = \bar{S}_k^1. \quad (6.18)$$

Представление условий зарождения событий S_k^1 и S_k^2 в (6.17), в отличие от представления их в системе уравнений (6.18), в виде объединенных событий $S_{ко}^1 = (S_1 S_{пр}^1)$ и $S_{ко}^2 = (S_2 S_{пр}^2)$ объясняется тем, что события $S_{пр}^1$ и $S_{пр}^2$, входящие в них, зависят не только от состояния согласующего буфера, но также и от событий S_1 и S_2 .

Для получения описаний событий $S_{ко}^1$ и $S_{ко}^2$ воспользуемся прямой таблицей переходов (ПТП), определяющей эти события. ПТП (табл.6.1) строится на основе требований к рассматриваемому алгоритму взаимодействия двух процессов.

Таблица 6.1

$S_{\text{БО}}(t)$	$S_{\text{БП}}(t)$	$S_1(t)$	$S_2(t)$	$S_{\text{КО}}^1(t+1)$	$S_{\text{КО}}^2(t+1)$	Комментарий
0	0	0	0	0	0	Буфер не пустой и не полный, возможна или запись, или чтение из буфера: $S_{\text{КО}}^1=1$ при $S_1=1$; $S_{\text{КО}}^2=1$ при $S_2=1$ и $S_1=0$
0	0	0	1	0	1	
0	0	1	0	1	0	
0	0	1	1	1	0	
0	1	0	0	0	0	Буфер полон (не пустой), возможно только чтение из буфера: $S_{\text{КО}}^2=1$ при $S_2=1$
0	1	0	1	0	1	
0	1	1	0	0	0	
0	1	1	1	0	1	
1	0	0	0	0	0	Буфер пустой («не полный»), возможна только запись в буфер: $S_{\text{КО}}^1=1$ при $S_1=1$
1	0	0	1	0	0	
1	0	1	0	1	0	
1	0	1	1	1	0	
1	1	0	0	-	-	Невозможные комбинации состояний буфера, $S_{\text{КО}}^1$ и $S_{\text{КО}}^2$ - не определены
1	1	0	1	-	-	
1	1	1	0	-	-	
1	1	1	1	-	-	

Исходя из ПТП (табл.6.1) минимальные выражения для событий $S_{\text{КО}}^1$ и $S_{\text{КО}}^2$, с учетом их неопределенных значений, примут следующий вид:

$$\begin{aligned}
 S_{\text{КО}}^1(t+1) &= S_1 \bar{S}_{\text{БП}}, \\
 S_{\text{КО}}^2(t+1) &= S_2 (S_{\text{БП}} \vee \bar{S}_1 \bar{S}_{\text{БО}}).
 \end{aligned}
 \tag{6.19}$$

Назначение и формальное описание остальных основных событий, представленных на графе НДА (рис.6.6), исходя из требований алгоритма управления процессами, определяются следующим образом:

- События S_1 и S_2 - определяют прием заявок процессов на запись и чтение из согласующего буфера; они определяются также как в системах уравнений (6.16а) и (6.16б).

- Событие S_m - определяет начало процедуры обращения к критическому ресурсу (согласующему буферу); выражение для события S_m определяется в соответствии с уравнением (6.16в).

- События S_p^1 и S_p^2 - определяют непересекающиеся операции записи и чтения из буфера; выражения для них имеют вид:

$$\begin{aligned} S_p^1(t+1) &= S_m S_k^1 \vee S_p^1 (\overline{S_{\text{БП}} \vee S_{\text{КС}}}), \\ S_p^2(t+1) &= S_m S_k^2 \vee S_p^2 \bar{S}_{\text{БО}}. \end{aligned} \quad (6.20)$$

Из выражения (6.20) следует, что запись в буфер заканчивается только тогда, когда буфер будет полон ($S_{\text{БП}} = 1$) или когда закончится поступление записываемой информации в буфер, хотя буфер может быть еще не полон (при $S_{\text{КС}} = 1$); чтение из буфера осуществляется до тех пор, пока буфер полностью не очистится (при $S_{\text{БО}} = 1$).

- События S_n^1 и S_n^2 , обеспечивающие выход взаимодействующих процессов из соответствующих критических участков, определяются следующим образом:

$$\begin{aligned} S_n^1(t+1) &= S_p^1 (S_{\text{БП}} \vee S_{\text{КС}}), \\ S_n^2(t+1) &= S_p^2 S_{\text{БО}}. \end{aligned} \quad (6.21)$$

- События S_r^1 и S_r^2 - символизируют ожидание взаимодействующими процессами окончания процедуры записи и чтения из буфера, чтобы продолжить свою работу, прерванную обращением к согласующему буферу; выражения для них имеют вид:

$$\begin{aligned} S_r^1(t+1) &= (S_1 \vee S_r^1) \bar{S}_n^1, \\ S_r^2(t+1) &= (S_2 \vee S_r^2) \bar{S}_n^2. \end{aligned} \quad (6.22)$$

Таким образом, общая сводная система канонических уравнений НД СКУ, представляющая алгоритм управления взаимодействующими процессами задачи «производители-потребители», когда процессы взаимодействуют через согласующий буфер с числом слов $N > 1$, будет представлена следующими выражениями:

Для процесса - производителя:

Для процесса - потребителя:

$$\begin{aligned}
 S_1(t+1) &= (S_{1,3} \vee S_1) \bar{S}_k S_2(t+1) = (S_{2,3} \vee S_2) \bar{S}_k^2, \\
 S_k^1(t+1) &= S_1 \bar{S}_k^2 \bar{S}_{\text{БП}} \vee S_k^2 \bar{S}_n^1(t+1) = S_2 \bar{S}_k^1 (S_{\text{БП}} \vee \bar{S}_1 \bar{S}_{\text{БО}}) \vee S_k^2 \bar{S}_n^2, \\
 S_p^1(t+1) &= S_m S_k^1 \vee S_p^1 \overline{(S_k^2(t+1) \vee S_{\text{БП}}^2 \vee S_{\text{КО}}^2 \vee S_{\text{МК}}^2 \vee S_{\text{БП}}^2 \vee S_{\text{КО}}^2 \vee S_{\text{МК}}^2)} \bar{S}_{\text{БО}}, \quad (6.23\text{а}) \\
 S_r^1(t+1) &= (S_1 \vee S_r^1) \bar{S}_n^1 S_r^2(t+1) = (S_2 \vee S_r^2) \bar{S}_n^2, \\
 S_n^1(t+1) &= S_p^1 (S_{\text{БП}} \vee S_{\text{КО}}^2)(t+1) = S_p^2 S_{\text{БО}}.
 \end{aligned}$$

Для последовательной компоненты алгоритма управления:

$$S_m(t+1) = S_k^1 S_1 \vee S_k^2 S_2 \quad (6.23\text{в})$$

6.6. Формализация алгоритмов управления взаимодействующими параллельными процессами в задаче «читатели-писатели»

Задача «читатели-писатели», наряду с задачей «производители-потребители», является еще одним примером классической задачи синхронизации двух классов процессов, которые имеют доступ к некоторому общему разделяемому ресурсу (РР), в качестве которого может быть, например, общая область памяти. Процессы первого класса (писатели) должны иметь монопольный доступ к ресурсу. Процессы второго класса (читатели) могут обращаться к данному ресурсу параллельно с любым числом процессов этого же класса, так как осуществляют чтение без изменения содержимого памяти. В качестве примера взаимодействия таких процессов может служить система продажи авиабилетов, позволяющая организовывать продажу билетов и изменять таким образом списки пассажиров и число свободных мест одновременно несколькими кассирами. Если в такой системе не будут предусмотрены правила, регламентирующие доступ к базам данных нескольких программ, то вполне возможна продажа билетов на одно и то же место нескольким пассажирам. Для исключения таких событий не должно допускаться параллельное исполнение двух процессов, которые читают и изменяют значение одного и того же объекта [73].

Словесная формулировка рассматриваемого варианта алгоритма взаимодействия параллельных процессов задачи «читатели-писатели» характеризуется следующими особенностями [61,70]. Алгоритм взаимодействия процессов должен гарантировать, чтобы в любой момент времени только один читатель или один писатель могли войти в критический

интервал (участок). Для того, чтобы не монополизировать доступ к РР читателей, когда их будет слишком много и они могли бы заблокировать запись, должен быть предусмотрен счетчик читателей (СчЧ), в который заносится некоторая константа E . Занесение этой константы в СчЧ должно выполняться при пустом счетчике после окончания работы читателя.

Читатель получает доступ к данным только в том случае, если в настоящий момент нет как работающего, так и ожидающего писателя. Первое условие должно обеспечиваться взаимоисключением критических интервалов обращения читателей и писателей к РР. Второе условие предотвращает бесконечное откладывание реализации процессов писателей из-за наплыва читателей. После окончания работы читателя СчЧ уменьшается на единицу и, если он будет не пустой, то работа читателей может быть продолжена до тех пор, пока СчЧ не станет пустым. Вычитание единицы из СчЧ осуществляется для непустого счетчика после окончания работы читателя.

Писатель получает монопольный доступ к РР, если в предшествующий момент времени не было ни чтения, ни записи в РР. В дальнейшем писатель может получить доступ к РР только в том случае, если в настоящий момент времени нет работающего читателя и СчЧ пуст или когда, независимо от состояния СчЧ, нет как ожидающего, так и работающего читателя. Так как доступ писателя к РР для обоих вариантов возможен только при условии отсутствия работающего читателя, то это условие должно обеспечиваться взаимоисключением критических интервалов обращения читателей и писателей к РР. Когда писатель заканчивает работу и СчЧ не пуст, то предпочтение отдается ожидающим читателям, а не ожидающим писателям, что предотвращает бесконечное откладывание процессов читателей из-за наплыва писателей. Если после окончания работы писателя СчЧ пуст, то в него заносится константа E и предпочтение также отдается ожидающим читателям, а не ожидающим писателям. Это обстоятельство обеспечивается, как это будет в дальнейшем показано, надлежащей формализацией события, характеризующего ожидающего писателя [74].

Учитывая основные требования к алгоритму взаимодействия процессов в задаче «читатели-писатели», его формальное описание начнем с определения событий, характеризующих **ожидающих и работающих писателей и читателей**. При этом будем базироваться на методике формализации алгоритмов взаимодействия параллельных процессов при обращении к РР в [2, 71]. Представим также для наглядности алгоритм управления параллельными процессами в виде графа НДА (рис.6.7). Представленный на рис.6.7 граф НДА, несмотря на отсутствие в нем некоторых логических вершин, которые опущены для простоты схемы графа, в достаточной степени дает представление об общей структуре алгоритма управления процессами.

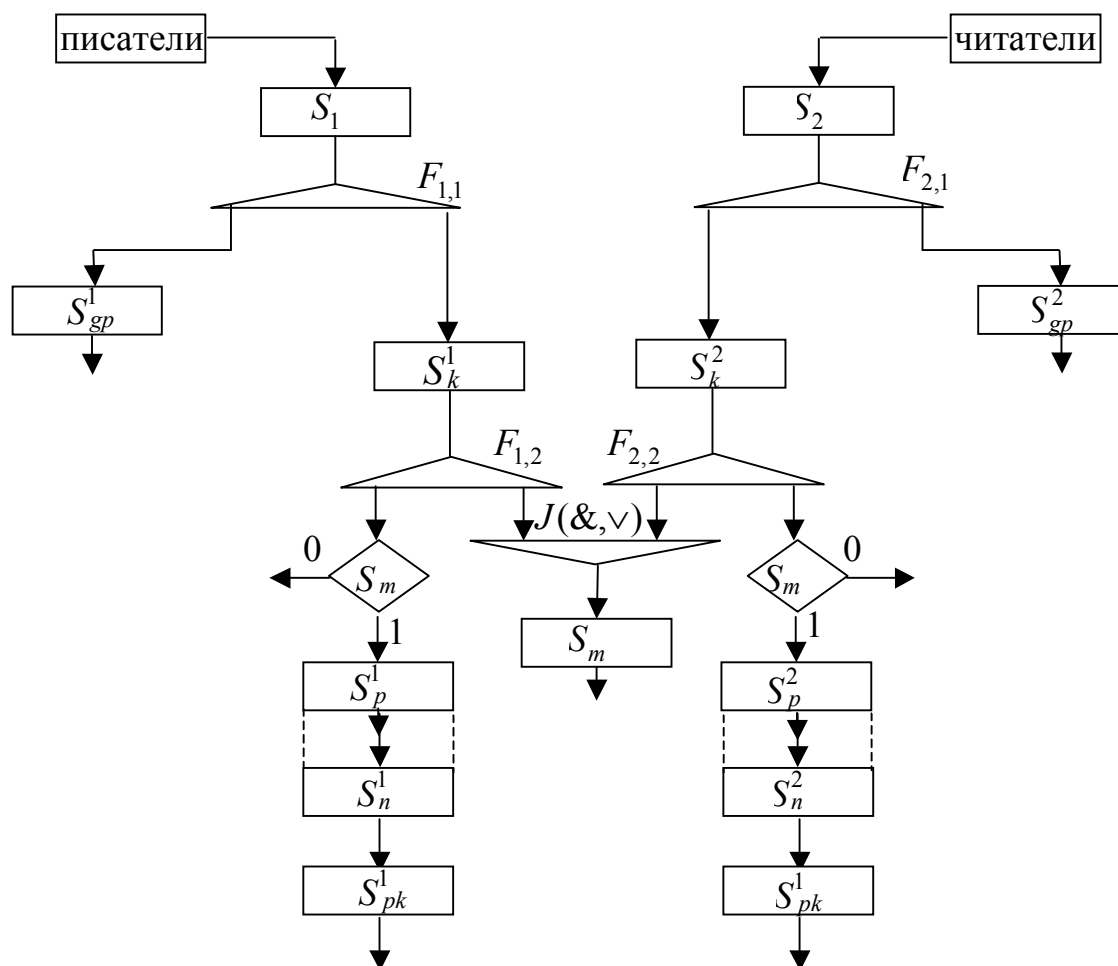


Рис.6.7. Фрагмент графа НДА алгоритма управления процессами в задаче «читатели-писатели»

Работающего писателя и читателя будем характеризовать событиями, которые определяют не только вход в критический интервал писателей и читателей, но и нахождение их в критическом интервале. Обозначим эти события буквами S_k^1 и S_k^2 соответственно. Т.е. если $S_k^1 = 1$, то ожидающий писатель переходит в разряд работающих писателей. Тоже самое относится и к событию S_k^2 , т.е. при $S_k^2 = 1$ ожидающий читатель переходит в разряд работающих читателей.

События типа S_k^1 и S_k^2 в некоторых наших работах [71] называли событиями активного ожидания обращения к РР, т.к. фактически непосредственная запись или чтение в РР будет выполняться лишь после зарождения событий S_k^1 и S_k^2 (спустя несколько тактов после их зарождения).

Ожидающего писателя и читателя будем характеризовать событиями S_1 и S_2 соответственно, которые могут зародиться только при наличии заявки на запись и чтение из РР и при условии, если свободен вход в критический

интервал писателя и читателя, т.е. при условиях $S_k^1 = 0$ и $S_k^2 = 0$. События S_1 и S_2 будут сохраняться до тех пор, пока условия $S_k^1 = 0$ и $S_k^2 = 0$ не изменятся. Повторное зарождение событий S_1 и S_2 для очередного писателя и читателя возможно после окончания работы предыдущего писателя и читателя соответственно. Исходя из рассмотренных условий события S_1 и S_2 могут быть представлены следующими уравнениями:

$$\left. \begin{aligned} S_1(t+1) &= S_{1,3}(\bar{S}_k^1 \vee S_n^1) \vee S_1 \bar{S}_k^1 \\ S_2(t+1) &= S_{2,3}(\bar{S}_k^2 \vee S_n^2) \vee S_2 \bar{S}_k^2 \end{aligned} \right\}, \quad (6.24)$$

где $S_{1,3}$ и $S_{2,3}$ - события, определяющие заявку на запись и чтение;

S_n^1 и S_n^2 - события являются заключительными в процедуре записи или чтения при обращении к РР писателей и читателей и обеспечивающими выход писателей и читателей из критических участков соответственно.

Условия зарождения событий S_k^1 и S_k^2 , определяющих входы в критические интервалы писателей и читателей, должны в соответствии с алгоритмом взаимодействия процессов удовлетворять требованиям как по взаимоисключению, так и по приоритетности взаимодействующих процессов при обращении к РР. В связи с этим уравнения, определяющие события S_k^1 и S_k^2 представим в следующем общем виде, который соответствует формальному представлению их в системе (6.9):

$$\left. \begin{aligned} S_k^1(t+1) &= S_1 S_{пр}^1 S_{вз}^1 \vee S_k^1 \bar{S}_n^1 \\ S_k^2(t+1) &= S_2 S_{пр}^2 S_{вз}^2 \vee S_k^2 \bar{S}_n^2 \end{aligned} \right\} \quad (6.25)$$

Первая часть уравнений (6.25), определяющая зарождение событий S_k^1 и S_k^2 , представлена конъюнкцией из трех событий:

S_1 и S_2 - события, определяющие ожидающего писателя и читателя;

$S_{пр}^1$ и $S_{пр}^2$ - события, обеспечивающие условия приоритетности процессов записи и чтения в соответствии с требованиями алгоритма взаимодействия процессов;

$S_{вз}^1$ и $S_{вз}^2$ - события, обеспечивающие взаимоисключение процессов записи и чтения на основе несовместимости событий S_k^1 и S_k^2 , т.е. для них должно выполняться условие: $S_k^1 \bar{S}_k^2 \vee \bar{S}_k^1 S_k^2 = 1$, откуда $S_{вз}^1 = \bar{S}_k^2$, а $S_{вз}^2 = \bar{S}_k^1$.

Вторая часть уравнений (6.25), определяющая условие сохранения событий S_k^1 и S_k^2 , обеспечивает занятие критических интервалов до тех пор, пока не закончится процесс записи (при $S_n^1 = 1$) или чтения (при $S_n^2 = 1$).

Исходя из требований алгоритма взаимодействия процессов, приоритетность процессов записи и чтения зависит от состояния СЧ, от

наличия ожидающих писателей и читателей (события S_1 и S_2) и от значений событий, свидетельствующих о завершении процессов записи и чтения, т.е. имеем:

$$\begin{aligned} S_{\text{пр}}^1 &= f_1(S_1, S_2, S_{\text{чо}}, S_{pk}^1, S_{pk}^2) \\ S_{\text{пр}}^2 &= f_2(S_1, S_2, S_{\text{чо}}, S_{pk}^1, S_{pk}^2), \end{aligned}$$

где S_{pk}^1 и S_{pk}^2 - события, свидетельствующие о факте окончания процедуры записи и чтения соответственно, эти события фактически повторяют события S_n^1 и S_n^2 , но со сдвигом на один такт.

В связи с тем, что события $S_{\text{пр}}^1$ и $S_{\text{пр}}^2$ зависят от событий S_1 и S_2 , то целесообразно рассматривать формализацию условий для зарождения событий S_k^1 и S_k^2 , состоящих из двух событий ($S_1 S_{\text{пр}}^1$) и соответственно ($S_2 S_{\text{пр}}^2$), как индивидуальных событий, обозначив их символами $S_{k,0}^1$ и $S_{k,0}^2$. Тогда уравнения (6.25) примут вид:

$$\left. \begin{aligned} S_k^1(t+1) &= S_{k,0}^1 S_{\text{вз}}^1 \vee S_k^1 \bar{S}_n^1 \\ S_k^2(t+1) &= S_{k,0}^2 S_{\text{вз}}^2 \vee S_k^2 \bar{S}_n^2 \end{aligned} \right\} \quad (6.26)$$

Для получения уравнений, формализующих события $S_{k,0}^1$ и $S_{k,0}^2$ с учетом пяти исходных событий ($S_1, S_2, S_{\text{чо}}, S_{pk}^1, S_{pk}^2$), воспользуемся построенной прямой таблицей переходов (ПТП) для событий $S_{k,0}^1$ и $S_{k,0}^2$ (табл.6.2).

При построении такой ПТП учитывались следующие обстоятельства:

- так как события S_{pk}^1 и S_{pk}^2 не могут существовать в один и тот же момент времени, то для комбинации их значений $S_{pk}^1 = S_{pk}^2 = 1$ значения событий $S_{k,0}^1$ и $S_{k,0}^2$ будут неопределенными;
- так как события S_k^1 и S_k^2 по требованию алгоритма взаимодействия процессов должны быть несовместимыми, то в каждой строке ПТП значения для $S_{k,0}^1$ и $S_{k,0}^2$ имеют противоположные значения;
- с целью уменьшения размерности ПТП в ней опущены строки, для которых $S_1 = S_2 = 0$ и $S_{pk}^1 = S_{pk}^2 = 1$.

С учетом указанных замечаний и в соответствии со словесной формулировкой алгоритма взаимодействия процессов ПТП для событий $S_{k,0}^1$ и $S_{k,0}^2$ будет иметь следующий вид (табл. 6.2).

Таблица 6.2

$S_{\text{ч0}}(t)$	$S_1(t)$	$S_2(t)$	$S_{pk}^1(t)$	$S_{pk}^2(t)$	$S_{k,0}^1(t+1)$	$S_{k,0}^2(t+1)$	Комментарий
0	0	1	0	0	0	1	СчЧ пустой, есть ожидающий читатель и нет
0	0	1	0	1	0	1	ожидающего писателя, поэтому
0	0	1	1	0	0	1	$S_{k,0}^2 = 1$ (чтение)
0	1	0	0	0	1	0	СчЧ пустой, есть ожидающий писатель
0	1	0	0	1	1	0	и нет ожидающего читателя, поэтому $S_{k,0}^1 = 1$ (запись)
0	1	0	1	0	1	0	
0	1	1	0	0	1	0	СчЧ пустой, есть ожидающий писатель
0	1	1	0	1	1	0	и ожидающий читатель, поэтому $S_{k,0}^1 = 1$
0	1	1	1	0	1	0	(монопольный режим для писателей)
1	0	1	0	0	0	1	СчЧ не пустой, есть ожидающий читатель
1	0	1	0	1	0	1	и нет ожидающего писателя, поэтому
1	0	1	1	0	0	1	$S_{k,0}^2 = 1$ (чтение)
1	1	0	0	0	1	0	СчЧ не пустой, есть ожидающий писатель
1	1	0	0	1	1	0	и нет ожидающего читателя, поэтому
1	1	0	1	0	1	0	$S_{k,0}^1 = 1$ (запись)

Продолжение таблицы 6.2

$S_{\text{чо}}(t)$	$S_1(t)$	$S_2(t)$	$S_{pk}^1(t)$	$S_{pk}^2(t)$	$S_{k,0}^1(t+1)$	$S_{k,0}^2(t+1)$	Комментарий
1	1	1	0	0	1	0	СчЧ не пустой, есть ожидающий писатель и читатель: -для 1-ой строки в предшествующий момент времени не было ни чтения, ни записи, поэтому $S_{k,0}^1 = 1$
1	1	1	0	1	0	1	(монопольный режим записи); -для 2-ой строки в предшествующий момент времени было чтение, поэтому $S_{k,0}^2 = 1$ (чтение продолжается);
1	1	1	1	0	0	1	-для 3-ей строки в предшествующий момент времени была запись, поэтому она запрещается и $S_{k,0}^2 = 1$ (чтение), т.к. не допускается бесконечная запись.

Для получения минимальных выражений для правых частей описания событий $S_{k,0}^1$ и $S_{k,0}^2$ воспользуемся представлением их диаграммами Вейча.

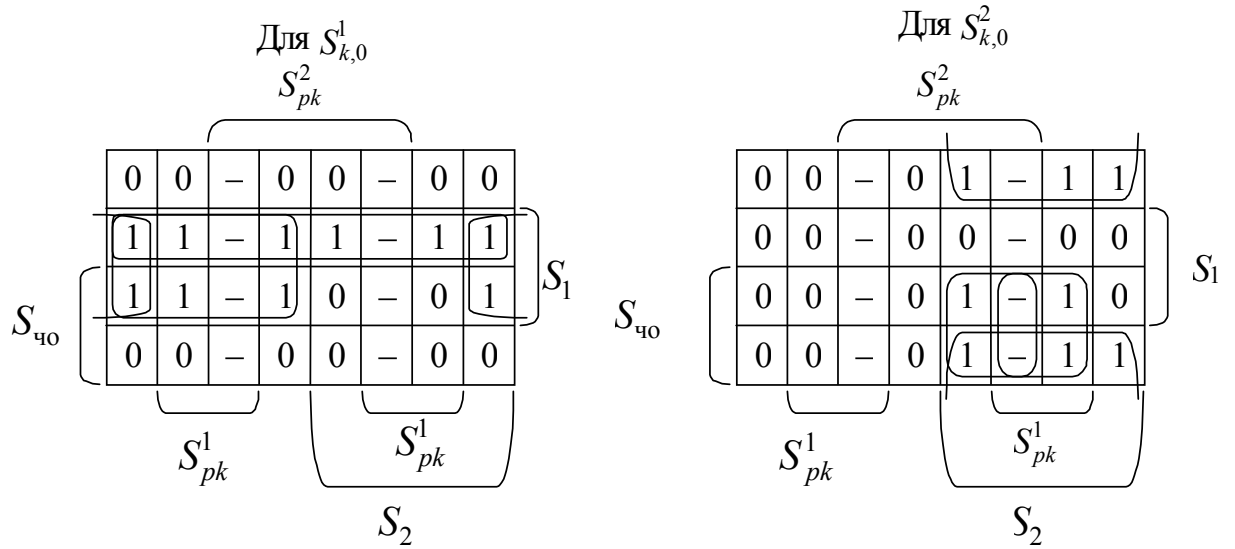


Рис.6.9. Диаграммы Вейча, представляющие правые части описаний для событий $S_{k,0}^1$ и $S_{k,0}^2$

Из диаграмм Вейча (рис.6.9) минимальные выражения для событий $S_{k,0}^1$ и $S_{k,0}^2$ будут иметь вид:

$$\left. \begin{aligned} S_{k,0}^1(t+1) &= S_1(\bar{S}_{чо} \vee \bar{S}_2 \vee \bar{S}_{pk}^1 \bar{S}_{pk}^2) \\ S_{k,0}^2(t+1) &= S_2[S_{чо}(S_{pk}^1 \vee S_{pk}^2) \vee \bar{S}_1] \end{aligned} \right\} \quad (6.27)$$

Учитывая полученные описания [выражения (6.24 - 6.27)], для главных событий, формализующих алгоритм управления взаимодействием процессов в задаче «читатели-писатели», общая сводная система канонических уравнений НД СКУ, представляющая рассматриваемый алгоритм управления процессами, будет иметь вид:

$$\begin{aligned} S_1(t+1) &= S_{1,3}(\bar{S}_k^1 \vee S_n^1) \vee S_1 \bar{S}_k^1, \\ S_2(t+1) &= S_{2,3}(\bar{S}_k^2 \vee S_n^2) \vee S_2 \bar{S}_k^2, \\ S_k^1(t+1) &= S_1 \bar{S}_k^2 (\bar{S}_2 \vee \bar{S}_{чо} \vee \bar{S}_{pk}^1 \bar{S}_{pk}^2) \vee S_k^1 \bar{S}_n^1, \\ S_k^2(t+1) &= S_2 \bar{S}_k^1 [\bar{S}_1 \vee S_{чо}(S_{pk}^1 \vee S_{pk}^2)] \vee S_k^2 \bar{S}_n^2, \\ S_m(t+1) &= S_k^1 S_1 \vee S_k^2 S_2, \\ S_p^1(t+1) &= S_m S_k^1, \\ S_p^2(t+1) &= S_m S_k^2, \\ S_n^1(t+1) &= S_p^1, \quad S_{pk}^1(t+1) = S_n^1, \\ S_n^2(t+1) &= S_p^2, \quad S_{pk}^2(t+1) = S_n^2, \end{aligned} \quad (6.28)$$

где S_m - событие, определяющее обращение к РР, как для записи, так и для чтения;

S_p^1 и S_p^2 - события, определяющие начало процедуры записи и чтения соответственно.

П р и м е ч а н и е: для рассматриваемого варианта алгоритма управления процессами условно принято, что для выполнения процедуры записи и чтения необходимы только по два оператора: для записи - события S_p^1 и S_n^1 и для чтения – события S_p^2 и S_n^2 .

При $t=0$ или при $\overline{S_{\text{чо}}} S_n^1 = 1$ выполняется операция $\text{СчЧ} := E$, при $S_{\text{чо}} S_n^2 = 1$ - операция $\text{СчЧ} := \text{СчЧ} - 1$. Представленные на графе (рис.6.7) события S_{gp}^1 и S_{gp}^2 могут инициировать выполнение некоторых подпроцессов за время ожидания работы писателей и читателей [71];

$$S_{gp}^1(t+1) = S_1 S_{g,3}^1 \bar{S}_k^1 \vee S_{gp}^1 \bar{S}_n^1,$$

$$S_{gp}^2(t+1) = S_2 S_{g,3}^2 \bar{S}_k^2 \vee S_{gp}^2 \bar{S}_n^2,$$

где $S_{g,3}^1$ и $S_{g,3}^2$ - события, определяющие заявки на реализацию подпрограммы для процессов писателей и читателей соответственно.

Проверку правильности работы алгоритма управления процессами можно выполнить на основе моделирования системы канонических уравнений НД СКУ (6.28), описывающих все представленные в управляющем алгоритме события для различных комбинаций заявок на запись и чтение из разделяемого критического ресурса.

По результатам моделирования можно построить временные диаграммы, в каждом такте которых, будут определены события, одновременное существование которых в данном такте возможно.

В качестве примера на рис.6.10 представлены временные диаграммы для следующих начальных условий: при $t=0/ \text{СчЧ}:=2$, $S_{\text{чо}}=1$, $S_{1,3}=S_{2,3}=1$.

S_1	S_1^1	S_1^1														
S_2	S_2^1	S_2^1	S_2^1	S_2^1	S_2^1	S_2^1	S_2^1				S_2^2	S_2^2				S_2^3
$S_{\text{чо}}$	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0
$S_k^1(S_k^2)$		S_k^1	S_k^1	S_k^1	S_k^1		S_k^2	S_k^2	S_k^2	S_k^2		S_k^2	S_k^2	S_k^2	S_k^2	
S_m			S_m					S_m					S_m			

$S_p^1(S_p^2)$				S_p^1					S_p^2					S_p^2		
$S_n^1(S_n^2)$					S_n^1					S_n^2					S_n^2	
$S_{pk}^1(S_{pk}^2)$						S_{pk}^1					S_{pk}^2					S_{pk}^2
Такты	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16

Рис. 6.10. Временные диаграммы работы алгоритма управления
взаимодействующими параллельными процессами
в задаче «читатели-писатели» (такты 1-16)

S_1	S_1^2	S_1^2	S_1^2	S_1^2	S_1^2	S_1^2				S_1^3	S_1^3				S_1^4	S_1^4
S_2	S_2^3														S_2^4	S_2^4
$S_{\text{чО}}$	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1
$S_k^1(S_k^2)$	S_k^2	S_k^2	S_k^2	S_k^2		S_k^1	S_k^1	S_k^1	S_k^1		S_k^1	S_k^1	S_k^1	S_k^1		S_k^2
S_m		S_m					S_m					S_m				
$S_p^1(S_p^2)$			S_p^2					S_p^1					S_p^1			
$S_n^1(S_n^2)$				S_n^2					S_n^1					S_n^1		
$S_{pk}^1(S_{pk}^2)$					S_{pk}^2					S_{pk}^1					S_{pk}^1	
Такты	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32

Рис.6.10. Временные диаграммы работы алгоритма управления
взаимодействующими параллельными процессами
в задаче «читатели-писатели» (такты 17-32)

Проследим работу управляющего алгоритма по временным диаграммам для некоторых частных случаев задания заявок на запись и чтение в процессе работы алгоритма по тактам:

- В 1-м такте фиксируются первые ожидающий писатель ($S_1^1 = 1$) и ожидающий читатель ($S_2^1 = 1$).

П р и м е ч а н и е: верхним индексом у событий S_1 и S_2 будем определять порядковые номера писателей и читателей соответственно.

- Во 2-м такте зарождается событие S_k^1 , т.к. $\overline{S_{pk}^1} \vee S_{pk}^2 = 1$ и ожидающий писатель переходит в следующем такте в ранг работающих писателей (открытый вход в критический участок писателей - $S_k^1 = 1$).

- С 3-его такта начинается процедура обращения к РР ($S_m = 1$) и запрещается восприятие возможной повторной заявки на обращение к РР для записи ($S_1^2 = 0$).

- В 4-м и 5-м тактах выполняется запись в РР.
- В 6-м такте подтверждается факт окончания записи в РР ($S_{pk}^1 = 1$) и осуществляется выход первого писателя из своего критического участка ($S_k^1 = 0$).

- В 7-м такте зарождается событие S_k^2 , т.к. $S_{\text{чО}} S_{pk}^1 = 1$ и первый ожидающий читатель (S_2^1) переходит в ранг работающих читателей, если даже был бы в наличии второй ожидающий писатель (т.е. при $S_1^2 = 1$).

- С 8-ого по 10-й такты выполняется обращение к РР с последующим чтением из РР.
- В 11-ом такте в СчЧ выполняется операция вычитания единицы (т.к. в предыдущем такте имело место соотношение $S_n^2 S_{чО} = 1$) и осуществляется выход первого читателя из своего критического участка ($S_k^2 = 0$).
- С 12-ого по 15-й такты осуществляется вход второго ожидающего читателя (S_2^2) в свой критический участок ($S_k^2 = 1$) с последующим обращением ($S_m = 1$) к РР для выполнения чтения.
- В 16-ом такте выполняются: операция вычитания единицы из СчЧ, в результате чего событие $S_{чО}$ становится равным единице ($S_{чО} = 1$); подтверждается факт окончания чтения из РР ($S_{pk}^2 = 1$) и осуществляется выход второго читателя из своего критического участка.
- С 17-ого по 20-й такты осуществляется вход уже третьего ожидающего читателя (S_3^2) в критический участок ($S_k^2 = 1$) с последующим обращением ($S_m = 1$) к РР для выполнения чтения, т.к. при зарождении события S_k^2 в предшествующий момент времени (16 такт) отсутствовал ожидающий писатель ($S_1^2 = 0$).
- В 21-м такте подтверждается факт окончания чтения из РР ($S_{pk}^2 = 1$) и осуществляется выход третьего читателя из своего критического участка $S_k^2 = 0$.
- С 22-ого по 25-й такты осуществляется вход второго писателя (S_1^2) в критический участок ($S_k^1 = 1$) с последующим обращением ($S_m = 1$) к РР для выполнения записи, т.к. $S_{чО} = 0$.
- В 26-м такте в СчЧ заносится константа $E=2$, подтверждается факт окончания записи в РР ($S_{pk}^1 = 1$) и осуществляется выход второго писателя из своего критического участка ($S_k^1 = 0$).
- С 27-ого по 30-й такты осуществляется вход третьего писателя (S_1^3) в критический участок ($S_k^1 = 1$) с последующим обращением ($S_m = 1$) к РР для выполнения записи, т.к. отсутствовал четвертый ожидающий читатель и т.д.

6.7. Формализация алгоритма управления взаимодействующими параллельными процессами в задаче «обедающие философы»

Задача «обедающие философы» является примером классической задачи, иллюстрирующей работу параллельных процессов, совместно использующих пересекающиеся группы ресурсов. Эта задача рассматривается во многих источниках, например, в [75, 76], где условие задачи формулируется следующим образом.

Пять философов, решая свои философские проблемы, время от времени, почувствовав голод, заходят в общую столовую, чтобы подкрепиться. В столовой за круглым столом расставлены пять стульев, каждый из которых занимает определенный философ. В центре стола находится большое блюдо спагетти, а между двумя соседними стульями на столе лежит по одной вилке. Для того чтобы покушать спагетти, философ должен обязательно взять две вилки, расположенные непосредственно слева и справа от него и наполнить с их помощью свою тарелку спагетти из общего блюда. После этого философ может приступить к трапезе, используя те же обе вилки. Окончив трапезу, он кладет вилки на свои места и выходит из столовой.

Тупиковая ситуация может возникнуть в том случае, если все пять философов для утоления голода сядут на свои стулья одновременно и каждый из них возьмет одновременно по одной, например, правой вилке. В этом случае никто из философов не сможет приступить к трапезе, т.к. любому из них для этого необходимы две вилки. Такую конфликтную ситуацию принято называть взаимоблокировкой процессов.

Одно из возможных решений рассматриваемой задачи, которое может позволить избежать взаимоблокировки процессов, основывается на том, чтобы не допускать пребывания в столовой всех пяти философов. В частности, исходя из условий задачи, когда количество ресурсов (вилок) ограничено, трапезой в столовой могут одновременно заниматься лишь два философа (или один). Однако такое обслуживание не позволит избежать другой конфликтной ситуации, называемой взаимоотталкиванием процессов, если не предусмотреть специальных мер, позволяющих избежать такой ситуации. Для рассматриваемой задачи «обедающие философы» взаимоотталкивание процессов может возникнуть в том случае, если два философа по договоренности между собой поочередно занимают вилки слева и справа от третьего философа так, что он никогда не сможет воспользоваться обеими вилками. Это и есть ситуация взаимоотталкивания. Для иллюстрации действующих в задаче «обедающие философы» процессов и возможных связей между ними и ресурсами (вилками) воспользуемся диаграммой, представленной на рис.6.11.

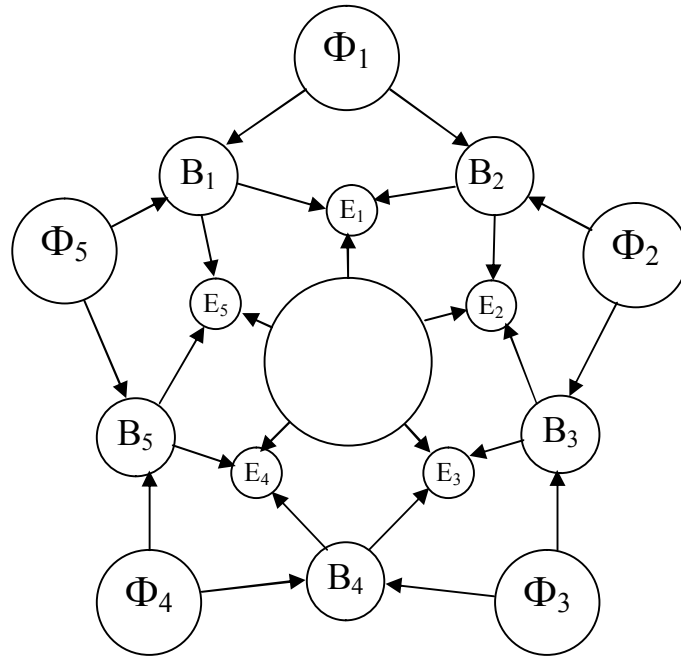


Рис.6.11. Диаграмма возможных связей действующих лиц-философов и вилок в задаче «обедающие философы»

На рис.6.11 введены следующие обозначения: $\Phi_1, \dots, \Phi_5$ – философы; $B_1, \dots, B_5$ – вилки (ресурсы); $E_1, \dots, E_5$ – тарелки философов; БС – общее блюдо спагетти.

Для формализации алгоритма управления взаимодействующими процессами, позволяющим избежать тупиковые ситуации, определим следующий состав событий, которые реализуются в алгоритме управления:

S_3^i – событие, свидетельствующее о том, что i -философ почувствовал потребность в утолении голода (заявка i -го философа на обслуживание);

S_i – событие, свидетельствующее о том, что заявка на обслуживание i -го философа принята (событие, характеризующее i -го философа, как философа ожидающего трапезу);

S_k^i – событие, свидетельствующее о том, что ожидающий i -й философ перешел в состав обедающих философов (событие, обеспечивающее вход i -го процесса в критический интервал);

S_{Ei} – событие, свидетельствующее об активном выполнении i -м философом процедуры трапезы;

S_{ik} – событие, свидетельствующее о том, что i -й философ закончил трапезу и положил вилки на стол (освободил ресурсы);

S_{Bi} – событие, свидетельствующее о том, что i -я вилка взята философом;

S_{pk}^i – событие, свидетельствующее о факте окончания i -го процесса.

В соответствии с введенными выше обозначениями событий алгоритм управления взаимодействующими процессами в задаче «обедающие философы» может быть представлен для наглядности следующим фрагментом графа НДА (рис.6.12).

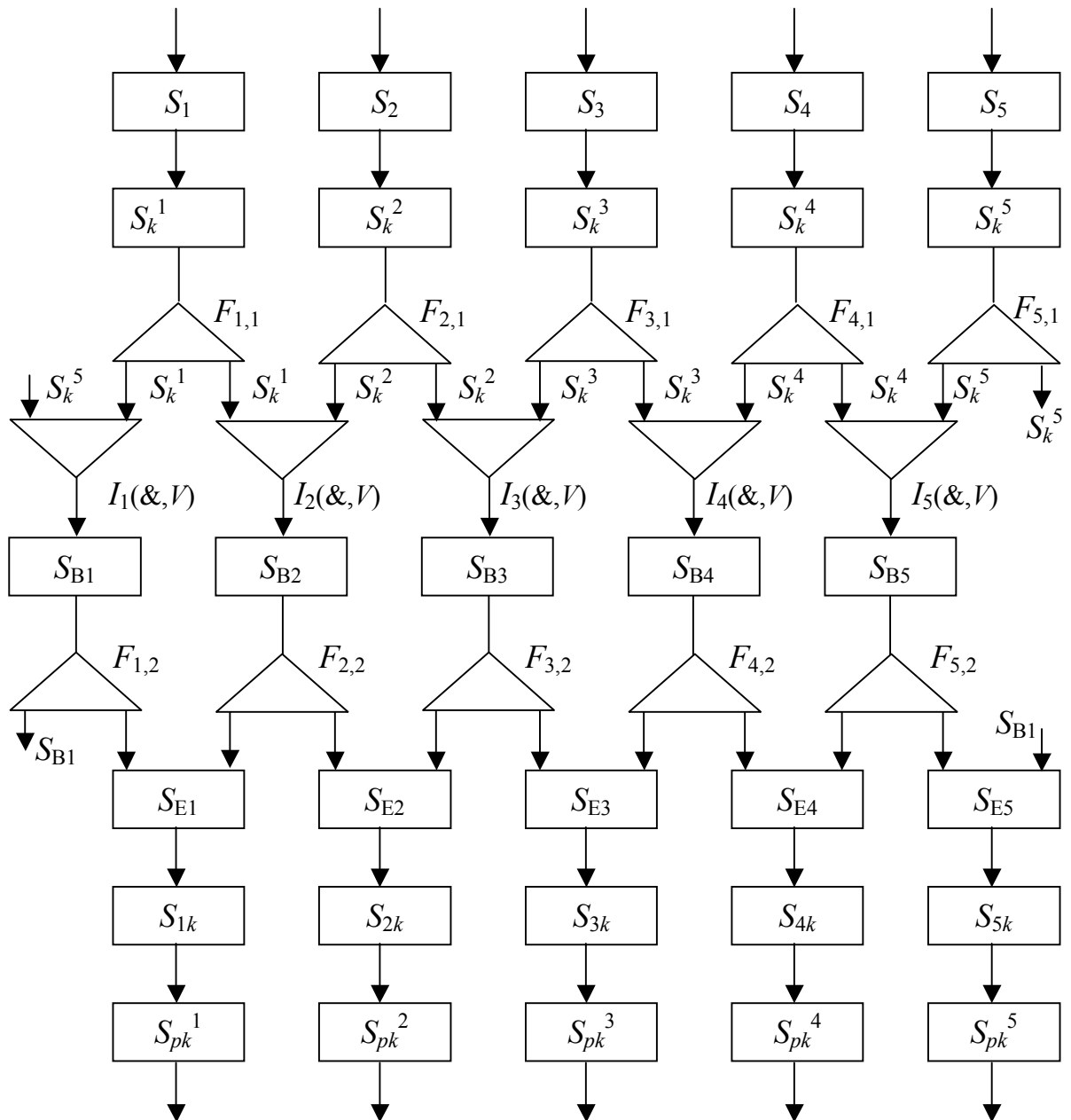


Рис.6.12. Фрагмент графа НДА алгоритма управления процессами в задаче «обедающие философы»

Граф НДА, представленный на рис. 6.12, несмотря на то, что в нем для простоты опущены логические вершины, позволяет, наряду с системой уравнений для событий, реализуемых алгоритмом, более наглядно представить работу алгоритма управления параллельными процессами, совместно использующих пересекающие группы ресурсов.

Учитывая основные требования алгоритма взаимодействия процессами в задаче «обедающие философы», а также введенный выше состав событий,

реализуемых в алгоритме управления, формальное описание такого алгоритма с учетом методики, рассмотренной в [2, 71], можно представить системой канонических уравнений вида (6.29).

В системе (6.29) правая часть всех уравнений рассматривается в момент времени t , но в данной записи системы время t для простоты опущено.

$$\begin{aligned}
S_i(t+1) &= (S_3^i \vee S_i) \bar{S}_k^i, \\
S_k^i(t+1) &= S_i S_{B3}^i S_{np}^i \vee S_k^i \bar{S}_{Ei}, \\
S_{Ei}(t+1) &= S_{Bi} S_{B(i \oplus 1)} S_k^i \vee \overline{S_{Ei} S_{ik}}, \\
S_{B1}(t+1) &= S_1 S_k^1 \vee S_5 S_k^5 \vee S_{B1} (S_k^1 \bar{S}_{E1} \vee S_k^5 \bar{S}_{E5}), \\
S_{B2}(t+1) &= S_2 S_k^2 \vee S_1 S_k^1 \vee S_{B2} (S_k^2 \bar{S}_{E2} \vee S_k^1 \bar{S}_{E1}), \\
S_{B3}(t+1) &= S_3 S_k^3 \vee S_2 S_k^2 \vee S_{B3} (S_k^3 \bar{S}_{E3} \vee S_k^2 \bar{S}_{E2}), \\
S_{B4}(t+1) &= S_4 S_k^4 \vee S_3 S_k^3 \vee S_{B4} (S_k^4 \bar{S}_{E4} \vee S_k^3 \bar{S}_{E3}), \\
S_{B5}(t+1) &= S_5 S_k^5 \vee S_4 S_k^4 \vee S_{B5} (S_k^5 \bar{S}_{E5} \vee S_k^4 \bar{S}_{E4}), \\
S_{ik}(t+1) &= S_{Ei}, \\
S_{pk}^i(t+1) &= S_{ik},
\end{aligned} \tag{6.29}$$

где $\oplus$ - знак сложения по модулю 5.

Рассмотрим структуру отдельных уравнений системы (6.29).

Первое уравнение системы (6.29) обеспечивает восприятие заявки на обслуживание i -го философа, когда он будет представлен в ранге философа ожидающего трапезу. Заявка будет воспринята и сохранена только в том случае, если будет свободен вход в критический интервал для i -го процесса ($S_k^i = 0$).

Второе уравнение системы (6.29) обеспечивает при $S_k^i = 1$ переход i -го философа, ожидающего трапезу, в ранг обедающих философов, когда будет обеспечен вход в критический интервал для i -го процесса. Условие зарождения события S_k^i представлено конъюнкцией из 3-х событий: S_i, S_{B3}^i, S_{np}^i .

Событие S_{B3}^i обеспечивает функцию взаимоисключения процессов при обслуживании философов. Функция S_{B3}^i определяется несовместимостью события S_k^i с событиями, формализующими входы в критические интервалы, имеющими номера соседние с i -м номером. Для рассматриваемой задачи для любого i -го процесса функция S_{B3}^i определится следующим образом:

$$S_{вз}^1 = \bar{S}_k^2 \bar{S}_k^5; \quad S_{вз}^2 = \bar{S}_k^1 \bar{S}_k^3; \quad S_{вз}^3 = \bar{S}_k^2 \bar{S}_k^4; \\ S_{вз}^4 = \bar{S}_k^3 \bar{S}_k^5; \quad S_{вз}^5 = \bar{S}_k^1 \bar{S}_k^4.$$

Событие S_{np}^i обеспечивает условие приоритетности обслуживания процессов в соответствии с требованиями взаимодействия процессов, исключающих тупиковые ситуации.

В качестве примера в рассматриваемой задаче будем базироваться на циклическую дисциплину обслуживания. Схематически такая последовательность обслуживания процессов представлена на рис.6.14. Здесь в соответствии с условиями задачи одновременно могут активно выполнять трапезу только два философа, к числу которых относятся пары: $(\Phi_1$ и $\Phi_3)$, $(\Phi_2$ и $\Phi_4)$, $(\Phi_3$ и $\Phi_5)$, $(\Phi_1$ и $\Phi_4)$ и $(\Phi_2$ и $\Phi_5)$.

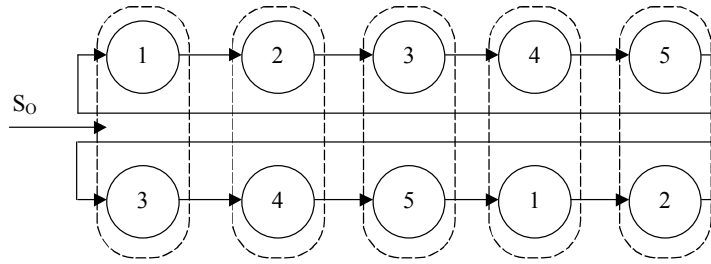


Рис.6.13. Схема циклической дисциплины обслуживания процессов, принятая в задаче «обедающие философы»

Как видно из рис.6.13 в одном цикле каждый философ может быть обслужен дважды, если будут соответствующие заявки на обслуживание. В тоже время предусматривается обслуживание и по одному философу, если на начало цикла имеет место только одна заявка на обслуживание или только две заявки, но имеющие соседние номера. Условием для такого обслуживания является истинность следующего события:

$$S_y = \bigvee_{i=1}^5 S_3^i \bigvee_{(\forall S_3^j)(j \neq i)} \bar{S}_3^j, i = \overline{1,5}. \quad (6.30)$$

Функцию приоритетности процессов для рассматриваемой дисциплины обслуживания можно представить следующим уравнением:

$$S_{np}^i(t+1) = S_{np}^i(0) \vee f_i(S_3^1, \dots, S_3^5, S_{pk}^1, \dots, S_{pk}^5), \quad (6.31)$$

где $S_{np}^i(0)$ - начальный приоритет обслуживания i -го процесса.

Исходя из принятой циклической дисциплины обслуживания выражения (6.31) для любого i -го процесса примут следующий вид:

$$\begin{aligned}
S_{np}^1(t+1) &= S_{np}^1(0) \vee S_3^1(S_{pk}^5 \vee \bar{S}_3^5(S_{pk}^4 \vee \bar{S}_3^4(S_{pk}^3 \vee \bar{S}_3^3(S_{pk}^2 \vee \bar{S}_3^2 S_{pk}^1)))) \\
S_{np}^2(t+1) &= S_{np}^2(0) \vee S_3^2(S_{pk}^1 \vee \bar{S}_3^1(S_{pk}^5 \vee \bar{S}_3^5(S_{pk}^4 \vee \bar{S}_3^4(S_{pk}^3 \vee \bar{S}_3^3 S_{pk}^2)))) \\
S_{np}^3(t+1) &= S_{np}^3(0) \vee S_3^3(S_{pk}^2 \vee \bar{S}_3^2(S_{pk}^1 \vee \bar{S}_3^1(S_{pk}^5 \vee \bar{S}_3^5(S_{pk}^4 \vee \bar{S}_3^4 S_{pk}^3)))) \quad (6.32) \\
S_{np}^4(t+1) &= S_{np}^4(0) \vee S_3^4(S_{pk}^3 \vee \bar{S}_3^3(S_{pk}^2 \vee \bar{S}_3^2(S_{pk}^1 \vee \bar{S}_3^1(S_{pk}^5 \vee \bar{S}_3^5 S_{pk}^4)))) \\
S_{np}^5(t+1) &= S_{np}^5(0) \vee S_3^5(S_{pk}^4 \vee \bar{S}_3^4(S_{pk}^3 \vee \bar{S}_3^3(S_{pk}^2 \vee \bar{S}_3^2(S_{pk}^1 \vee \bar{S}_3^1 S_{pk}^5))))
\end{aligned}$$

Учитывая принятую систему обслуживания по одному или по два процесса в зависимости от комбинации заявок выражение для $S_{np}^i(0)$ будет включать две составляющие:

$$S_{np}^i(0) = S_{np,1}^i(0) \vee S_{np,2}^i(0), \quad (6.33)$$

где $S_{np,1}^i(0)$ - начальный приоритет обслуживания i -го процесса при условии, что $S_y=1$. Тогда система уравнений для $S_{np,1}^i(0)$ будет иметь вид:

$$\begin{aligned}
S_{np,1}^1(0) &= S_0 x_n S_3^1 S_y, \\
S_{np,1}^2(0) &= S_0 x_n S_3^2 \bar{S}_3^1 S_y, \\
S_{np,1}^3(0) &= S_0 x_n S_3^3 \bar{S}_3^1 \bar{S}_3^2 S_y, \\
S_{np,1}^4(0) &= S_0 x_n S_3^4 \bar{S}_3^1 \bar{S}_3^2 \bar{S}_3^3 S_y, \\
S_{np,1}^5(0) &= S_0 x_n S_3^5 \bar{S}_3^1 \bar{S}_3^2 \bar{S}_3^3 \bar{S}_3^4 S_y,
\end{aligned} \quad (6.34)$$

где S_0 - начальное событие системы управления процессами;

x_n – сигнал инициализации системы управления.

Так как i -й философ в цикле обслуживания может обслужиться дважды, то выражение для начального приоритета соответствующего i -го процесса при обслуживании по паре философов будет состоять из двух событий:

$$S_{np,2}^i = S_{np,2}^{i,k}(0) \vee S_{np,2}^{i,m}(0), \quad (6.35)$$

где для любого i -го процесса значение k и m определится исходя из следующей таблицы:

i	k	m
1	3	4
2	4	5
3	1	5
4	1	2
5	2	3

События в правой части выражения (6.35) в соответствии с принятой дисциплиной обслуживания имеют вид:

$$\begin{aligned}
S_{np,2}^{1,3}(0) &= S_0 x_n S_3^1 S_3^3, \\
S_{np,2}^{2,4}(0) &= S_0 x_n S_3^2 S_3^4 (\bar{S}_3^1 \vee \bar{S}_3^3), \\
S_{np,2}^{3,5}(0) &= S_0 x_n S_3^3 S_3^5 (\bar{S}_3^2 \vee \bar{S}_3^4) \bar{S}_3^1, \\
S_{np,2}^{4,1}(0) &= S_0 x_n S_3^4 S_3^1 (\bar{S}_3^2 \bar{S}_3^3), \\
S_{np,2}^{5,2}(0) &= S_0 x_n S_3^5 S_3^2 (\bar{S}_3^3 \bar{S}_3^4).
\end{aligned} \tag{6.36}$$

Выполнив подстановку в уравнения (6.35) выражений из (6.36), получим систему уравнений для событий, определяющих приоритеты для всех i -х процессов на начало работы алгоритма управления процессами для циклической дисциплины обслуживания пар процессов:

$$\begin{aligned}
S_{np,2}^1(0) &= S_0 x_n S_3^1 (S_3^3 \vee S_3^4 \bar{S}_3^2), \\
S_{np,2}^2(0) &= S_0 x_n S_3^2 [S_3^4 (\bar{S}_3^1 \vee \bar{S}_3^3) \vee \bar{S}_3^3 (S_3^4 \vee S_3^5)], \\
S_{np,2}^3(0) &= S_0 x_n S_3^3 [S_3^1 \vee S_3^5 (\bar{S}_3^2 \vee \bar{S}_3^4)], \\
S_{np,2}^4(0) &= S_0 x_n S_3^4 [S_3^1 (\bar{S}_3^2 \vee \bar{S}_3^3) \vee S_3^2 (\bar{S}_3^1 \vee \bar{S}_3^3)], \\
S_{np,2}^5(0) &= S_0 x_n S_3^5 [S_3^2 \bar{S}_3^3 \bar{S}_3^4 \vee S_3^3 (\bar{S}_3^2 \vee \bar{S}_3^4) \bar{S}_3^1].
\end{aligned} \tag{6.37}$$

Третье уравнение системы (6.29) представляет событие (S_{Ei}) , определяющее активное выполнение i -м философом трапезы. Это событие зарождается при условии, что i -й философ владеет двумя вилками ($S_{Bi} = S_{B(i \oplus I)} = 1$) и вход в критический интервал занят i -м процессом ($S_k^i = 1$).

Уравнения системы (6.29) с четвертого по восьмое представляют события $(S_{B1}, \dots, S_{B5})$, свидетельствующие о том, что в любой момент времени i -й философ может занять две вилки (ресурсы) с соседними номерами (i -й и $(I \oplus 1)$ -й). Этим номерам вилок (ресурсов) соответствуют события, условия зарождения которых определяют выражение $S_i S_k^i = 1$.

Отметим некоторые общие особенности систем уравнений для событий, представленных в данном разделе.

Для конкретных практических условий время существования событий системы (6.29) можно регулировать за счет коррекции второй составляющей выражений, определяющих соответствующее событие.

В том случае, если после какого-либо такта цикла обслуживания философов в соответствии с заявками обслужился только один философ, то в соответствии с уравнениями (6.32), независимо от комбинаций заявок, обслуживание в последующих тактах будет выполняться только по одному философу. Для повышения эффективности системы управления обслуживанием, если после какого-нибудь такта обслуживания имеет место

комбинация заявок не удовлетворяющая условию $S_y = 1$, то следует произвести повторную инициализацию системы управления, для которой будут работать уравнения системы (6.37).

Контрольные вопросы к главе 6.

1. Какие разделяемые ресурсы называют критическими?
2. Как описать событие, если его зарождение необходимо задержать до появления другого события?
3. Как описать событие, если его истинность необходимо сохранить до появления другого события?
4. Что понимается под критическим интервалом (или критическим участком)? Каковы основные требования к критическим участкам при организации доступа к разделяемым данным параллельных процессов?
5. В чем отличительная особенность формализации несовместимости событий при их возможном одновременном и неодновременном зарождении?
6. Из каких составных частей должно состоять описание событий, определяющих входы процессов в свои критические интервалы? Каким требованиям к критическим интервалам должны удовлетворять описания таких событий?
7. Из каких составных частей должно состоять описание событий, определяющих прием заявок на обслуживание процессов при обращении к разделяемым данным? Каким требованиям к критическим интервалам должны удовлетворять описания таких событий?
8. Как формализуется событие, определяющее начало процедуры обращения к критическому ресурсу (захват разделяемого ресурса)?
9. Для какой цели состояния согласующего буфера, играющего роль критического ресурса в задаче «производители-потребители», характеризуют двумя событиями (буфер полон и буфер пустой)? Как это обстоятельство используется при формализации события, определяющего зарождение события входа в критический интервал?
10. Как формализуются события, определяющие ожидание процессами окончания процедуры обращения к критическому ресурсу?
11. В чем заключается особенность алгоритма взаимодействия процессов в задаче «читатели-писатели», обеспечивающая предотвращение бесконечного откладывания реализации процессов как писателей из-за наплыва читателей, так и читателей из-за наплыва писателей?
12. Как можно формализовать события, характеризующие ожидающих и работающих писателей и читателей? В чем основная разница в описаниях таких событий?
13. В чем заключаются характерные тупиковые ситуации, которые могут возникнуть при реализации задачи «обедающие философы»?

14. Какие компоненты входят в описание событий, характеризующих любого i -го философа, ожидающего трапезу, и обедающего философа?

15. В чем заключается особенность формализации события, определяющего захват разделяемого ресурса в задаче «обедающие философы», в отличие от формализации события, определяющего тоже захват разделяемого ресурса, но только одного?

16. В чем проявляется общность описания основных событий, определяющих бесконфликтность межпроцессного взаимодействия при решении задач такого взаимодействия, связанных с организацией обращения процессов к разделяемым ресурсам?

Глава 7. Проектирование алгоритмов для задач идентификации языков

Задачи идентификации языков, реализуемые цифровым автоматом-распознавателем, имеют очень широкое применение. В частности, идентификация цепочек символов входит как составная часть во многие задачи, связанные с редактированием текстов, поиском данных и символьной обработкой. Многие программы редактирования текстов разрешают пользователю задавать различные типы замен в тексте. Например, пользователю необходимо заменить какое-то слово другим словом во всем тексте или его части. Чтобы выполнить такое действие, программа редактирования текста должна суметь найти вхождение слова и определить его местоположение. Некоторые искусные редактирующие программы разрешают пользователю в качестве множества заменяемых цепочек символов указывать *регулярное множество*. Например, пользователь может поставить задачу: «Заменить $[R]$ в тексте M пустой цепочкой», имея в виду, что в M следует стереть пару квадратных скобок и символы между ними, представляющие выражение R .

Подобные задачи решаются в антивирусных программах. Например, для обнаружения «простых» вирусов необходимо найти последовательность байт (сигнатуру вирусов) в зараженном файле. При обнаружении «полиморфных» вирусов обнаружение сигнатуры входит как одна из составляющих технологии их поиска.

В типичной задаче идентификации цепочек-образов - задаются входная последовательность M (например, символов или пронумерованные фрагменты графических файлов и т.п.) и множество цепочек-образов $[p_1, p_2, \dots, p_n]$. Требуется найти либо хотя бы одно вхождение какой-то цепочки-образа, либо m из n ($1 \leq m < n$), либо вхождения всех n цепочек-образов в M . При этом множество цепочек-образов часто является регулярным множеством, которое можно задать регулярным выражением алгебры событий. Ниже будут показаны приемы решения такого рода задач, основанные на использовании языка регулярных выражений алгебры событий (РВАС) и модели недетерминированного конечного автомата (НДА).

Модель конечного детерминированного цифрового автомата (ДА)-распознавателя, для которого будут разрабатываться алгоритмы решения задач идентификации цепочек-образов, будем рассматривать как устройство, которое в каждый дискретный момент времени находится в одном из конечного множества состояний, и входной ленты (входная последовательность) состоящей из клеток, которая просматривается слева направо блоком чтения этого устройства. Каждый шаг алгоритма распознавания ДА состоит из перехода в новое состояние, определяемое его текущим состоянием и читаемым входным сигналом, и сдвига блока чтения на одну клетку вправо. Доказано, что любой язык представим регулярным

выражением тогда и только тогда, когда он допускается некоторым конечным автоматом.

Если решается задача только обнаружения наличия цепочки - образа, то достаточно модели **одностороннего автомата**, в которой на каждом шаге алгоритма блок чтения может перемещаться на одну клетку только слева направо. Если же решается задача обнаружения местоположения цепочки - образа, то необходимо использовать модель **двухстороннего автомата**, в которой блок чтения может перемещаться или слева направо или справа налево. Лента, которую читает автомат, должна содержать два специальных символа: «начало» - z_t (top) и «конец» - z_b (bottom), при достижении которых передвижение головки соответственно влево и вправо невозможно, о чем сообщается выработкой специальных сигналов **"достижение начала"(y_t)/"достижение конца"(y_b)**.

Важным обобщением понятия ДА является НДА. Для каждого события (состояния) и каждого входного сигнала функция переходов НДА имеет нуль или более вариантов выбора следующего шага алгоритма. При этом необходимо иметь в виду, что для построения систем распознавания языков, в определении НДА должен быть указан состав заключительных событий (состояний), определяющих все допустимые слова языка (вместо состава выходных сигналов НДА, используемых при построении управляющих систем). Аналитический язык задания НДА, получивший название языка систем канонических уравнений и систем выходных функций (СКУ и СВФ) совместно с алгоритмом перехода от РВАС к СКУ и СВФ, позволяют снять имеющиеся ограничения языка РВАС при формализации ниже перечисленных задач идентификации. Следует отметить, что автор языка предложил (глава 4) простой и эффективный способ перехода от РВАС к СКУ и СВФ.

Кроме того, хотя алгоритм идентификации задачи, например, типа "1" (смотри ниже) практически напрямую формализуется на языке РВАС, переход к его программной или аппаратной реализации неочевиден. В то же время алгоритм, записанный на языке СКУ и СВФ, представляет готовое решение для программной или аппаратной реализации.

7.1. Формализация задач распознавания цепочек - образов

7.1.1. Формулировка задач распознавания цепочек-образов

Рассмотрим более детально формулировку задач распознавания цепочек-образов. Пусть дана цепочка-последовательность следующего вида $M = z_t z_{\alpha,1} z_{\beta,2} \cdots z_{m,j} \cdots z_{\gamma,n} z_b$ (где $z_{m,j}$ любой символ из алфавита $[Z]$, у которого введен дополнительный второй индекс j , определяющий порядковый номер этого символа в последовательности M) и цепочка—образ p , заданная регулярным выражением в том же алфавите. Требуется найти такой наименьший индекс j символа входной последовательности M , что для

некоторого i подцепочка $z_{k,i} z_{v,i+1} \dots z_{m,j}$ цепочки M принадлежит языку, представленному выражением p [58].

В зависимости от того, какая цель преследуется в результате поиска можно выделить следующие классы задач.

Обнаружение факта наличия цепочки, и только. Например, наличие в тексте слова или предложения; обнаружение сигнатуры вируса при поиске в файле и т.п. В этом случае достаточно реализовать алгоритм поиска с использованием модели одностороннего автомата, когда при обнаружении цепочки вырабатывается выходной сигнал, индицирующий положительный результат поиска.

Обнаружение факта наличия цепочки и локализация места ее расположения. Примером этого может быть необходимость нахождения фразы в тексте и замены ее, или обнаружение вируса в файле и "лечение" его. Для таких случаев необходимо реализация алгоритма поиска и локализации двухсторонним автоматом, когда при обнаружении цепочки для локализации ее места положения блок чтения автомата начинает перемещаться справа налево (при этом запоминаются позиции ленты) до момента обнаружения начала цепочки. При программной реализации, если этот поиск реализуется как один процесс, после того как обнаружена «цепочка – образ» блок чтения начинает перемещаться в обратном направлении для поиска начала цепочки. Если запомнить при этом позицию конца цепочки, то затем блок чтения можно сразу переместить вправо на позицию, следующую за ее концом. Если же операционная система позволяет параллельно выполнить несколько процессов, то тогда можно запустить еще один процесс для локализации цепочки, в то время как родительский продолжает просматривать входную последовательность. Алгоритм работы автомата в прямом направлении реализуется на основе РВАС, а алгоритм работы в обратном на основе «зеркальной» РВАС.

Необходимо отметить, что задачи распознавания образов не сводятся только к поиску одной единственной цепочки во входной последовательности. Приведем формулировку таких задач, которые встречаются на практике.

1) **Поиск любой из нескольких заданных цепочек**, т.е. хотя бы одна из цепочек есть в последовательности. Эту задачу можно свести к поочередному поиску каждой цепочки в последовательности. Тогда число просмотров будет равно числу искомых цепочек. Но в этом случае время поиска возрастает пропорционально количеству цепочек и их длине, а в некоторых случаях, например «поиск на лету» (без возможности повторных просмотров входной последовательности) он в принципе становится невозможным.

2) **Поиск наличия всех заданных цепочек**, т.е. в последовательности должны быть все искомые цепочки.

3) **Поиск заданного количества искомых цепочек** т.е. например, ищутся 5 цепочек, а результат поиска может быть положительным, если будут найдены хотя бы 3 из них. Здесь дополнительная трудность при поиске состоит в том, что заранее невозможно определить порядок, в котором они будут находиться в последовательности. Если же искать их многократным просмотром последовательности, то будут существенные потери времени.

4) **Поиск нескольких цепочек, у которых префикс и суффикс одинаковы, а корни различны.** Если префикс обозначить через A , суффикс через C , а корни через B_1, B_2, B_3 , то на языке РВАС алгоритм поиска запишется как: $r(y) = A \& (B_1 \vee B_2 \vee B_3) \& C$. Можно, конечно, выполнить преобразование, раскрыв скобки, но тогда суффиксы и префиксы уйдут в каждую из трех цепочек, что увеличит размерность алгоритма.

5) **Поиск цепочек, когда в последовательности некоторые символы являются «прозрачными»** (т.е. их как бы нет). Например, игнорирование знаков препинания в тексте или определенных байтовых комбинаций в двоичных файлах.

6) **Когда поиск в последовательности начинается по сигналу «извне»**, а до того времени на каждом шаге осуществляется просто просмотр очередного символа последовательности. Например, когда поиском управляет внешняя среда или когда нужно найти цепочку после пропуска определенного количества символов входной последовательности.

Отметим, что начальное состояние S_0 НДА, которое назовем **маркером начала поиска**, имеет очень существенное значение. Маркер позволяет динамически управлять процессом поиска без изменения самого алгоритма поиска (программная или аппаратная реализация не меняется), что обеспечивает гибкость при реализации алгоритма. Например, не меняя алгоритма, а управляя только маркером можно получить решение следующих подзадач, которые могут являться составной частью задач 1-6.

1) Не начинается ли последовательность с искомой цепочки, т.е. самый первый символ последовательности совпадает с первым символом цепочки. Такая задача решается установкой маркера начала поиска ($S_0=1$) перед приемом первого символа последовательности и сбросом его ($S_0=0$) после приема первого символа.

2) Классическая задача поиска, когда последовательность может содержать цепочки в любом месте между "top" и "bottom". Эта задача решается установкой маркера ($S_0=1$) перед приемом первого символа последовательности, без последующего сброса (поиск искомых цепочек начинается с каждого очередного символа).

3) Поиск в последовательности начинается после обнаружения в ней определенного символа или символов (точки, отмечающей конец предложения и т.п.). Эта задача решается установкой маркера после поступления этого символа, без последующего сброса.

4) Поиск в последовательности начинается после поступления сигнала начать поиск «извне» или наступлении определенного события "внутри" (например, прочитано определенное количество символов в последовательности и т.п.)

Рассмотрим, как можно решить задачи 1-5 и подзадачи 1-4. Будем рассматривать задачи идентификации только на обнаружение, т.к. решение задач локализации полностью идентично, только с «зеркальным» отражением.

7.1.2. Разработка алгоритмов решения задач распознавания цепочек-образов

Решение задачи типа "1". Алгоритм решения этой задачи (без учета поступления символа z_i) можно непосредственно записать на языке РВАС. Например, необходимо определить есть ли во входной последовательности любая из трех следующих цепочек:

1) $z_3 z_2 z_1 \dots z_1 z_3$; 2) $z_2 \dots z_2 z_1 z_3$; 3) $z_4 \dots z_4 z_2 \dots z_2 z_3 z_1 \dots z_1$,

где $z_i \dots z_i$ повторение символа z_i любое число раз, но не менее одного. При обнаружении любой цепочки должен быть выработан выходной сигнал y .

Тогда на языке РВАС алгоритм поиска запишется:

$$r(y) = s_0(z_3 z_2 z_1 \{z_1\} z_3 \vee z_2 \{z_2\} z_1 z_3 \vee z_4 \{z_4\} z_2 \{z_2\} z_3 z_1 \{z_1\})$$

При переходе от РВАС к СКУ и СВФ в соответствии с алгоритмом приведенным ранее, преобразуем выражение r в следующую систему из трех уравнений, где S_{k1} , S_{k2} , S_{k3} события, появляющиеся при обнаружении цепочек 1, 2, 3 соответственно:

$$S_{k1}(y) = S_0 z_3 z_2 z_1 \{z_1\} z_3$$

$$S_{k2}(y) = S_0 z_2 \{z_2\} z_1 z_3$$

$$S_{k3}(y) = S_0 z_4 \{z_4\} z_2 \{z_2\} z_3 z_1 \{z_1\}.$$

Дальнейшие преобразования дадут следующие СКУ и СВФ:

СКУ:

$$S_1 = S_2 z_1 \vee S_1 z_1$$

$$S_2 = S_3 z_2$$

$$S_3 = S_0 z_3$$

$$S_{k1} = S_1 z_3$$

$$S_4 = S_5 z_1$$

$$S_5 = S_0 z_2 \vee S_5 z_2$$

$$S_{k2} = S_4 z_3$$

$$S_6 = S_7 z_3$$

$$S_7 = S_8 z_2 \vee S_7 z_2$$

$$S_8 = S_0 z_4 \vee S_8 z_4$$

$$S_{k3} = S_6 z_1 \vee S_{k3} z_1$$

СВФ:

$$y = S_{k1} \vee S_{k2} \vee S_{k3}$$

Для поиска цепочек в любом месте последовательности необходимо устанавливать маркер начала поиска перед приемом каждого следующего символа. Поэтому в СКУ нужно добавить уравнение: $S_0 = 1$, а чтобы учесть поступление символа z_b необходимо добавить еще одно уравнение в СКУ и одно в СВФ:

СКУ:

$$S_0 = 1$$

$$S_b = S_0 z_b \vee S_b;$$

СВФ:

$$y_b = S_b.$$

При практической реализации алгоритма необязательно повторять переходы из S_b в S_b . Для программной реализации достаточно будет остановить выполнение программы, а при аппаратной реализации заблокировать прохождение синхросигнала на элементы памяти.

Решение задачи типа "2". Получить алгоритм решения этой задачи возможно следующим образом:

- на языке РВАС записывается алгоритм поиска любой цепочки, как и в предыдущей задаче;
- по РВАС строится СКУ и СВФ;
- СКУ изменяется (корректируется) так, что событие, возникающее при обнаружении какой-либо цепочки, становится равным 1 и остается таким при каждом последующем поступлении входного символа (тактовом сигнале).
- поиск завершается успешно, если конъюнкция всех таких событий становится равной 1 (все цепочки найдены) и неуспешно, если до этого поступит символ z_b .

Используем тот же пример что и в задаче 1, только теперь все искомые цепочки должны присутствовать во входной последовательности.

СКУ:

$$S_0 = 1$$

$$S_1 = S_2 z_1 \vee S_1 z_1$$

$$\begin{aligned}
S_2 &= S_3 z_2 \\
S_3 &= S_0 z_3 \\
S_{k1} &= S_1 z_3 \vee S_{k1} \\
S_4 &= S_5 z_1 \\
S_5 &= S_0 z_2 \vee S_5 z_2 \\
S_{k2} &= S_4 z_3 \vee S_{k2} \\
S_6 &= S_7 z_3 \\
S_7 &= S_8 z_2 \vee S_7 z_2 \\
S_8 &= S_0 z_4 \vee S_8 z_4 \\
S_{k3} &= S_6 z_1 \vee S_{k3} z_1 \vee S_{k3} \\
S_b &= S_0 z_b \vee S_b \\
\text{СВФ:} \\
y &= S_{k1} \& S_{k2} \& S_{k3} \\
y_b &= S_b.
\end{aligned}$$

Решение задачи типа "3". При разработке алгоритма решения этой задачи наряду с замечаниями к задаче "2" нужно еще учесть и то, что заранее неизвестен порядок, в котором появятся цепочки во входной последовательности. Используем тот же пример что и в задаче 1, только теперь необходимо найти хотя бы две цепочки из трех.

При переходе от РВАС к СКУ и СВФ проделаем тоже, что и в задаче 2, но по иному выполним коррекцию СВФ.

$$\begin{aligned}
\text{СКУ:} \\
S_0 &= 1 \\
S_1 &= S_2 z_1 \vee S_1 z_1 \\
S_2 &= S_3 z_2 \\
S_3 &= S_0 z_3 \\
S_{k1} &= S_1 z_3 \vee S_{k1} \\
S_4 &= S_5 z_1 \\
S_5 &= S_0 z_2 \vee S_5 z_2 \\
S_{k2} &= S_4 z_3 \vee S_{k2} \\
S_6 &= S_7 \& z_3 \\
S_7 &= S_8 z_2 \vee S_7 z_2 \\
S_8 &= S_0 z_4 \vee S_8 z_4 \\
S_{k3} &= S_6 z_1 \vee S_{k3} z_1 \vee S_{k3} \\
S_b &= S_0 z_b \vee S_b \\
\text{СВФ:} \\
y &= S_{\underline{k1}} S_{\underline{k2}} \vee S_{\underline{k1}} S_{\underline{k3}} \vee S_{\underline{k2}} S_{\underline{k3}} \\
y_b &= S_b.
\end{aligned}$$

Решение задачи типа "4". Рассмотрим решение такой задачи, когда префикс будет задан регулярным выражением $z_1z_1z_2z_3$, суффикс - $z_2\{z_1\}z_3$, а корни будут из задачи 1. Необходимо найти во входной последовательности любую цепочку из трех. Тогда алгоритм поиска на языке РВАС запишется:

$$r(y) = S_0z_1z_1z_2z_3(z_3z_2z_1\{z_1\}z_3 \vee z_2\{z_2\}z_1z_3 \vee z_4\{z_4\}z_2\{z_2\}z_3z_1\{z_1\})z_2\{z_1\}z_3.$$

Разобьем выражение r на три части:

$$S_p = S_0z_1z_1z_2z_3 - \text{для префикса,}$$

$$S_{k1} = S_p(z_3z_2z_1\{z_1\}z_3) - \text{для корня 1,}$$

$$S_{k2} = S_p(z_2\{z_2\}z_1z_3) - \text{для корня 2}$$

$$S_{k3} = S_p(z_4\{z_4\}z_2\{z_2\}z_3z_1\{z_1\}) - \text{для корня 3}$$

$$r(y) = S_rz_2\{z_1\}z_3 - \text{для суффикса.}$$

Для каждого уравнения перейдем от РВАС к СКУ и объединим их. Получим:

СКУ:

$$S_0 = 1$$

Для префикса (маркер начала поиска S_0):

$$S_p = S_9z_3$$

$$S_9 = S_{10}z_2$$

$$S_{10} = S_{11}z_1$$

$$S_{11} = S_0z_1$$

Для корней (маркер начала поиска S_p):

$$S_1 = S_2z_1 \vee S_{11}z_1$$

$$S_2 = S_3z_2$$

$$S_3 = S_pz_3$$

$$S_{k1} = S_1z_3$$

$$S_4 = S_5z_1$$

$$S_5 = S_pz_2 \vee S_5z_2$$

$$S_{k2} = S_4z_3$$

$$S_6 = S_7z_3$$

$$S_7 = S_8z_2 \vee S_7z_2$$

$$S_8 = S_pz_4 \vee S_8z_4$$

$$S_{k3} = S_6z_1 \vee S_{k3}z_1$$

Для суффикса (маркеры начала поиска S_{k1} , S_{k2} , S_{k3}):

$$r = S_{12}z_3$$

$$S_{12} = (S_{k1} \vee S_{k2} \vee S_{k3})z_2 \vee S_{12}z_1$$

Для входного символа "*bottom*":

$$S_b = S_0z_b \vee S_b;$$

СВФ:

$$y = r$$

$$y_b = S_b.$$

Решение задачи типа "5" Рассмотрим решение такой задачи на примере, когда искомая цепочка будет $z_1 z_1 z_2 z_3$, в алфавите $Z = \{z_1, z_2, z_3, z_4, z_5, z_6, z_7, z_8, z_9, z_{10}\}$, а игнорируемый символ при поиске z_{10} . Тогда цепочка в последовательности $z_1 z_{10} z_{10} z_1 z_2 z_{10} z_3$ должна быть обнаружена.

Получить алгоритм решения этой задачи возможно следующим образом:

- на языке РВАС записывается алгоритм поиска цепочки;
- по РВАС строится СКУ и СВФ;
- СКУ изменяется (корректируется) так, что алгоритм не меняет своего состояния при поступлении на вход z_{10} (символ игнорируется и как бы «прозрачен» для цепочки).

Алгоритм поиска на языке РВАС запишется:

$$r(y) = S_0 z_1 z_1 z_2 z_3.$$

Перейдем от РВАС к СКУ и СВФ и получим:

СКУ:

$$S_0 = 1$$

$$r = S_1 z_3$$

$$S_1 = S_2 z_2$$

$$S_2 = S_3 z_1$$

$$S_3 = S_0 z_1$$

$$S_b = S_0 z_b \vee S_b$$

СВФ:

$$y = r$$

$$y_b = S_b.$$

Корректируем СКУ и получаем:

$$S_0 = 1$$

$$r = S_1 z_3 \vee r \& z_{10}$$

$$S_1 = S_2 z_2 \vee S_1 \& z_{10}$$

$$S_2 = S_3 z_1 \vee S_2 \& z_{10}$$

$$S_3 = S_0 z_1 \vee S_3 \& z_{10}$$

$$S_b = S_0 z_b \vee S_b.$$

Решение задачи типа "6". При решении этой задачи алгоритм разрабатывается в том же порядке как и в задачах 1-5. Но в СКУ добавляется $S_0 = 0$. Поэтому все символы в последовательности игнорируются до тех пока по сигналу «извне» не установится маркер начала поиска ($S_0 = 1$).

В заключение следует отметить, что хотя разработка алгоритмов решения задач поиска велась для однопроцессорной системы (последовательная реализация алгоритма), по своей сути полученные алгоритмы уже являются параллельными. Но если задача поиска имеет

большую размерность (длинные искомые цепочки, их большое количество, много файлов большой длины, внутри которых осуществляется поиск), и есть возможность использования многопроцессорной системы, то имеет смысл с точки зрения повышения производительности системы поиска произвести дальнейшее распараллеливание алгоритма.

7.2. Разработка алгоритмов решения задач распознавания цепочек-образов для многопроцессорной реализации

7.2.1. Общие сведения о методике проектирования параллельных алгоритмов

Одна из важных тенденций развития современной вычислительной техники состоит в расширенном внедрении многопроцессорных архитектур. Микропроцессоры позволили получить процессорное время по номинальной цене, а их малые габариты делают вполне обоснованной постановку вопроса об их комплексировании в единой системе в количестве десятков, сотен и более. Одним из ключевых факторов способствующих использованию таких систем будет наличие параллельных алгоритмов.

Один из возможных подходов к разработке параллельного алгоритма для решения какой-либо задачи, в том числе задач идентификации базируется на методике, приведенной в [59] и состоит из следующих этапов проектирования:

- разбиение исходной задачи на элементарные задачи;
- определение всех необходимых взаимодействий (коммуникаций) между элементарными задачами;
- объединение элементарных задач (агломерация) в соответствии с определенной стратегией, учитывающей в том числе число процессоров в системе;
- распределение полученных при агломерации задач на заданное число процессоров;
- оценка полученного алгоритма (или алгоритмов, если их получено несколько) на масштабируемость (при увеличении объема исходных данных и/или изменении числа процессоров в мультипроцессорной системе).

Разбиение задачи на элементарные подзадачи.

На этом этапе проектирования определяется максимально возможный параллелизм при решении задачи. Для этого производится разбиение данных (исходных, промежуточных), используемых в задаче, а также операций выполняемых над ними на элементарные части (не допускающие дальнейшего разбиения) без учета числа процессоров в многопроцессорной системе.

Рассмотрим **разбиение по данным**. Как правило, разбиение данных дает и разбиение операций, выполняемых над ними. В результате такого разбиения получается комплекс элементарных задач (ЭЗ), выполнение которых дает решение исходной задачи. При этом нужно учитывать, что если какие-то ЭЗ требуют данных от других ЭЗ, то потребуется организации связей (коммуникаций) между ними. На коммуникации затрачивается время многопроцессорной системы, и это время может занять существенный процент от общего времени решения задачи. Идеальным можно считать случай, когда коммуникации между ЭЗ отсутствуют. Существует хорошее правило при разбиении данных - начинать разбиение с наибольших структур данных или с тех, которые используются наиболее часто в задаче.

Рассмотрим **разбиение операций**. В этом случае исходная задача разбивается на элементарные операции (функции) в идеальном случае не зависящие друг от друга.

При разбиении задачи необходимо рассмотреть все возможные варианты. Затем, поскольку разбиение по данным и разбиение по операциям являются интуитивными, полученные результаты необходимо сверить со следующими рекомендациями, позволяющими оценить качество разбиения.

1) Если полученное число ЭЗ меньше или равно числу процессоров в системе, то на последующих этапах проектирования не будет обеспечена гибкость в принятии решений.

2) Если полученное разбиение приводит к чрезмерному количеству операций и/или к чрезмерным требованиям по затратам памяти, то результирующий параллельный алгоритм масштабируем (не может быть применим при увеличении входной размерности данных).

3) Если полученные задачи неодинакового размера, то при дальнейшем проектировании будет трудно обеспечить равномерную загрузку процессоров (балансировка системы).

4) Нужно, чтобы число ЭЗ было пропорционально размеру исходной задачи. В идеальном случае увеличение размерности исходной задачи увеличивает число ЭЗ значительно быстрее, чем размер самой ЭЗ. Если это не соблюдается, то параллельный алгоритм будет не пригоден при росте размерности задачи и увеличении числа процессоров в системе.

Определение необходимых взаимодействий (коммуникаций) между элементарными задачами.

На втором этапе проектирования определяются необходимые взаимодействия (коммуникации) между ЭЗ. ЭЗ выполняются параллельно, но, как правило, являются взаимозависимыми. Элементарная задача в какой-то момент времени может потребовать данные другой ЭЗ. Для этого необходима пересылка данных между ними. Эти информационные потоки и определяются на этапе проектирования коммуникаций.

Для проектирования коммуникаций определяются каналы от источника (кто посылает) к приемнику (кто принимает) и определяются сообщения,

которые пересылаются по этим каналам. При разбиении данных определение коммуникаций между ЭЗ является сложной задачей, так как существует достаточно много вариантов, из которых надо выбрать наиболее оптимальный. Наоборот, при разбиении по операциям определение коммуникаций достаточно простая задача, так как коммуникации соответствуют потоку данных между ЭЗ.

В общем случае можно произвести классификацию коммуникаций по следующим признакам: - локальная или глобальная; -структурированная (регулярная) или неструктурированная (нерегулярная); - статическая или динамическая; - синхронная или асинхронная.

При *локальной* коммуникации каждая ЭЗ взаимодействует с небольшим числом задач (ее «соседями»), а при *глобальной* со многими задачами (в пределе с каждой). При *структурированной* коммуникации каждая ЭЗ и ее соседи образуют регулярную структуру (например «дерево» или «сетка»), а при *неструктурированной* нерегулярную структуру (произвольный граф). При *статической* коммуникации взаимосвязи определяются «раз и навсегда» и не меняются при выполнении алгоритма, а при *динамической* могут меняться. При *синхронной* коммуникации есть синхронизирующие сигналы между источником и приемником, а при *асинхронной* нет. Результаты определения коммуникаций также необходимо сверить со следующими рекомендациями для оценки их качества.

1) Если не все элементарные задачи выполняют одинаковые коммуникации, то такая несогласованность дает немасштабируемый алгоритм.

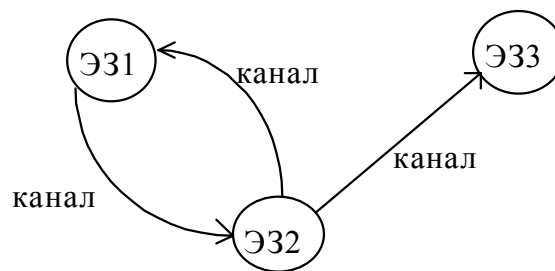
2) Если каждая элементарная задача связана с большим числом соседей, то следует исследовать возможность преобразования глобальных коммуникаций в локальные.

3) Если все коммуникации не могут выполняться параллельно (одновременно), то алгоритм не масштабируем и не эффективен.

Выбор модели аппаратной и программной реализации параллельных вычислений.

Если алгоритм решения задачи разрабатывается не под конкретную многопроцессорную систему, а с возможностью выбора её архитектуры, то на этом этапе необходимо определить наиболее подходящую *модель аппаратной реализации* из числа следующих: - сетка процессоров (транспьютер); - объединение процессоров через сеть (локальную, глобальную); - взаимодействие процессоров через общую память.

Затем необходимо определить наиболее подходящую *модель программной реализации* параллельных вычислений (задачи и каналы, пересылка сообщений с использованием стандартных библиотечных утилит, разделяемая память с использованием блокировок и семафоров).



Задачи и каналы

Пересылка сообщений с использованием стандартных библиотечных утилит во многом аналогична модели «задачи и каналы», но для отправки и приема сообщений используются вызовы библиотечных утилит, т.е. есть существуют определенные ограничения на пересылаемые и получаемые сообщения по каналам.

В модели разделяемой памяти с использованием блокировок и семафоров задачи используют общее адресное пространство, в котором они читают и записывают данные асинхронно. Механизм блокировок и семафоров используется для контроля доступа к общей памяти.

Объединение элементарных задач.

На этом этапе проектирования производится объединение (укрупнение) ЭЗ. Комплекс ЭЗ полученных после этапа разбиения и этапа определения коммуникаций, представляет абстрактный алгоритм в том смысле, что он не может быть реализован на любой многопроцессорной системе. Поэтому рассматриваются решения по объединению ЭЗ с целью получения параллельного алгоритма, который эффективно выполнялся бы на некотором классе многопроцессорных систем. Нужно рассмотреть, где полезно скомбинировать задачи, полученные на этапе разбиения **так, чтобы обеспечить в результате наименьшее число задач, каждая из которых имеет наибольший размер.**

При агломерации получаемое число задач сокращается (в идеальном случае одна задача на процессор), но еще может быть больше числа процессоров в системе. В этом случае проектирование предусматривает следующий шаг - распределение задач по процессорам. Соответственно на этапе агломерации нужно стремиться, чтобы в результате была одна задача на процессор.

Как правило, три конфликтующих цели преследуются при агломерации и размножении данных и/или операций:

- сокращение стоимости коммуникаций за счет объединения взаимосвязанных ЭЗ (локализация коммуникаций);
- сохранение гибкости по отношению к масштабируемости (изменению размерности задачи) и распределению задач по процессорам;

- сокращении стоимости инженерной разработки программного обеспечения.

Рассмотрим каждую из них.

Локализация коммуникаций. На этапе разбиения алгоритма на ЭЗ усилия сосредотачиваются на получении как можно большего числа задач. Это позволяет рассмотреть широкий круг возможностей для параллельного выполнения. Однако, следует заметить, что получение большого числа задач при разбиении не обязательно дает эффективный параллельный алгоритм. Самое негативное влияние на время выполнения параллельного алгоритма оказывают коммуникации. На большинстве параллельных систем приходится останавливать вычисления, чтобы послать или принять сообщения необходимые для их продолжения. Улучшить скорость вычисления можно пересылкой меньшего объема данных между задачами, выполняющимися параллельно. Менее очевидно, но сократить время на коммуникации можно и при использовании меньшего числа сообщений при тех же объемах данных. Это объясняется тем, что каждая коммуникация требует время не только пропорциональное объему пересылаемых данных, но имеет фиксированные начальные затраты времени по ее установлению. В дополнение к коммуникационным временным затратам нужно учитывать и стоимость создания задачи на процессоре (выделение памяти и т.д.).

Исходя из вышесказанного, объединение взаимодействующих задач в одну устраняет необходимость их внешних коммуникаций, так как полученная задача будет выполняться на одном процессоре.

Можно отметить, что при разбиении по данным эффективен подход увеличения локальности за счет увеличения размерности «куска», обрабатываемых регулярных данных (векторы, матрицы и т.п.). Но если данные нерегулярны, то задача агломерации становится достаточно трудной. Можно попытаться использовать размножение данных и/или операций для сокращения коммуникаций и/или для сокращения времени вычисления.

Сохранение гибкости. Хорошо разработанный параллельный алгоритм должен быть гибким по отношению к числу процессоров в системе, что повышает ее надежность в случае отказа одного или нескольких процессоров. Гибкость также полезна и при создании программного кода для отдельного процессора. Если задачи часто блокируются в ожидании данных от других процессоров, это может быть использовано для назначения нескольких таких задач на один процессор, с тем, что пока одна задача ожидает данных, другая может выполняться. Тогда время коммуникации одной задачи перекрывается временем вычисления другой и таким образом процессор не простаивает. Третья полезность создания большего числа задач чем процессоров состоит в обеспечении лучших возможностей при распределении задач по процессорам (балансировка вычислений). Хорошее правило состоит в том, что число задач должно быть на порядок больше числа процессоров.

Оптимальное число задач обычно определяется комбинацией эмпирического изучения и аналитического моделирования.

Сокращение стоимости программного обеспечения. До этого момента выбор стратегии агрегации определялся только улучшением скорости вычислений и гибкостью параллельного алгоритма. Если же на этом этапе удастся получить идентичные задачи (несколько или все) то при технической реализации алгоритма (программировании) можно получить большую экономию как по времени, так и по трудовым затратам при разработке машинных кодов задач.

Результаты этапа разработки коммуникаций необходимо сверить со следующими контрольными вопросами.

1) Если затраты времени на коммуникации не уменьшаются при увеличении локальности, то надо искать другую стратегию распараллеливания задачи.

2) Если агрегация размножает данные, то нужно проверить, что это не ухудшает масштабируемость алгоритма, ограничивая размер задачи или число используемых процессоров.

3) Производит ли агрегация задачи с одинаковыми затратами на вычисления и коммуникации? Если получается одна задача на процессор, то их размерности должны быть примерно одинаковы.

4) Масштабируется ли число задач, получаемых при агрегации, с ростом входной размерности обрабатываемых данных. Если нет, то трудно использовать такой алгоритм с ростом размерности задачи и числа процессоров.

5) Можно ли еще увеличивать агрегацию без нарушения балансировки загрузки процессоров, увеличения инженерной стоимости программирования или сокращения масштабируемости. При прочих равных условиях, алгоритм создающий несколько больших задач проще и эффективнее того, который создает много задач.

Распределение задач по процессорам.

На заключительной стадии проектирования параллельного алгоритма нужно определить какая задача на каком процессоре будет выполняться. В целом задача распределения остается трудной задачей, которая должна быть точно разрешена при проектировании параллельного алгоритма. Цель распределения состоит в том чтобы минимизировать время выполнения алгоритма (сумма времен вычисления, простоя и коммуникаций) и затраты памяти.

Используются две стратегии для достижения этой цели:

- размещение задач, которые могут выполняться параллельно на разных процессорах;

- размещение задач, которые часто взаимодействуют, на одном и том же процессоре, чтобы уменьшить коммуникационную стоимость.

Ясно, что эти стратегии конфликтуют и проектирование должно разрешить эти противоречия. Кроме того, на распределение задач по процессорам могут влиять доступные ресурсы одного процессора, ограничивая число задач которые могут быть одновременно размещены на нем.

Такая задача распределения известна как *NP*-задача. Это означает, что задача трудно разрешима, поскольку для нее нет алгоритма решения с полиномиальной зависимостью от входной размерности (число задач и число процессоров). Поэтому при решении таких задач используются эвристические методы.

Так при разбиении по данным используется статический или динамический загрузочно-балансирующий алгоритм, обеспечивающий равномерную загрузку процессоров.

При функциональной декомпозиции используется алгоритм составления расписаний для задач. Есть набор задач - централизованный (в одном месте) или распределенный (в нескольких местах) куда новые задачи помещаются и откуда берутся для размещения по процессорам.

Наиболее сложным вопросом при составлении расписания является используемая стратегия при размещении задач по процессорам. В основном выбираемая стратегия представляет компромисс между конфликтными требованиями по независимости задач (чтобы сократить стоимость коммуникаций) и полном знании о ходе вычислений (чтобы улучшить балансировку загрузки процессоров). Рассмотрим централизованный (управляющий/исполнители), иерархический (главный управляющий/управляющий/исполнители) и децентрализованный подходы.

Централизованный подход. При таком подходе управляющий отвечает за проблему размещения задач по процессорам. Каждый исполнитель постоянно обращается к управляющему с запросами (например, с сообщением об окончании выполнения очередной задачи), получает новую задачу от него и исполняет ее. В свою очередь исполнители могут послать новые задачи управляющему, которые затем могут быть размещены по другим исполнителям. Эффективность такой стратегии зависит от числа исполнителей, относительной стоимости получения задач от управляющего. Результаты могут быть улучшены кешированием задач у исполнителей так, чтобы исполнители связывались с управляющим только тогда, когда нет локальных (своих) задач.

Иерархический подход. Исполнители делятся на отдельные группы, у каждой из которых свой управляющий. Исполнители группы связываются только со своим управляющим. Управляющие периодически связываются с главным управляющим и другими управляющими, чтобы обеспечить балансировку загрузки процессоров, за которые они отвечают.

Децентрализованный подход. В полностью децентрализованной схеме нет центрального управляющего (все исполнители), и отдельный набор задач назначается на каждый процессор. Если процессор выполнил все свои задачи, он требует задач от других процессоров. В результате набор задач становится распределенной структурой данных, которая доступна различным задачам асинхронно. При этом определяется какая-либо процедура доступа между процессорами. Например, исполнитель может требовать задания только от небольшого числа соседей или выбирать другие исполнители по “заданному” случайному закону.

Гибридный централизованно - децентрализованный подход. В гибридной централизованной/децентрализованной схеме запросы от процессоров-исполнителей посылаются центральному управляющему который размещает их по исполнителям, используя круговой "обход-отбор". Нужно отметить, что хотя в этом случае при большом числе процессоров центральный управляющий по-прежнему будет узким местом в системе, доступ к нему будет требоваться менее часто чем к управляющему в централизованной системе, и следовательно, это более масштабируемая конструкция.

Обнаружение завершения. Составление расписаний требует механизма для обнаружения момента времени когда весь алгоритм будет завершен, в противном случае простаивающие исполнители никогда не остановятся, требуя задания от других исполнителей Эта задача легко решается в централизованной схеме, где управляющий определяет, что все исполнители закончили работу. Но это более сложная задача в децентрализованной схеме поскольку нет общей информации, что процессоры закончили и требования на новые задания будут «бродить» по системе.

Результаты этапа распределения задач необходимо сверить со следующими контрольными вопросами.

1) При использовании централизованной загрузочно-балансной схемы, нужно проверить, в какой степени управляющий является узким местом в системе.

2) Есть ли достаточное число задач при использовании вероятностных или циклических методов балансировки загрузки. Для этого обычно требуется по крайней мере не менее 10 задач на процессор.

В результате такого подхода к проектированию будет получено, как правило, несколько алгоритмов. Затем нужно провести анализ, чтобы выбрать один из альтернативных алгоритмов, для которого будет выполнен этап технической реализации. При этом нужно оценить **эффективность параллельной реализации** алгоритма как отношения времени его выполнения при последовательной реализации к времени выполнения при параллельной реализации, а также изменение его эффективности с **ростом числа процессоров при неизменной размерности исходных данных** и

изменение эффективности с *ростом размерности исходных данных при неизменности числа процессоров*.

7.2.2. Пример разработки параллельных алгоритмов для решения задач распознавания цепочек-образов

Рассмотрим задачу, в которой требуется за один просмотр каждой последовательности символов идентифицировать наличие хотя бы одной из искомым цепочек-образов [78]. Выполнение задачи завершается при обнаружении любой цепочки в любой последовательности. Количество последовательностей символов (S) равно 10. Длины последовательностей символов равны 10(2),20(2),30(4),40(2), т.е. две последовательности по 10 символов, две последовательности по 20 символов и т.д. Число процессоров в системе (P) равно 6. Входной алфавит $Z=[z_1, \dots, z_9]$. Искомые цепочки-образы:

$$z_1z_2z_2 \dots z_2z_1; \quad z_3 z_2z_1; \quad z_3z_2z_3 \dots z_2z_3z_1.$$

Последовательности символов и результаты поиска цепочек в них – локализованы (в одном месте). Необходимо выполнить проектирование на событийном уровне параллельного алгоритма решения этой задачи и его исследование.

Вначале разработаем алгоритм поиска цепочек в одной последовательности символов для однопроцессорной системы. Алгоритм можно записать регулярным выражением на языке РВАС:

$$r(y)=S_0(z_1z_2\{z_2\}z_1 \vee z_3z_2z_1 \vee z_3\{z_2z_3\}z_1).$$

Разобьем на части:

$$S_{k1}(y)=S_0z_1z_2\{z_2\}z_1$$

$$S_{k2}(y)=S_0z_3z_2z_1$$

$$S_{k3}(y)=S_0z_3\{z_2z_3\}z_1$$

Перейдем от РВАС к СКУ и СВФ:

СКУ:

$$S_{k1}=S_1z_1$$

$$S_1=S_2z_2 \vee S_{k1}z_2$$

$$S_2=S_0z_1$$

$$S_{k2}=S_3z_1$$

$$S_3=S_4z_2$$

$$S_4=S_0z_3$$

$$S_{k3}=S_5z_1$$

$$S_5=S_0z_3 \vee S_6z_3$$

$$S_6=S_5z_2$$

$$S_f=(S_{k1} \vee S_{k2} \vee S_{k3} \vee S_1 \vee S_2 \vee S_3 \vee S_4 \vee S_5 \vee S_6 \vee S_f) z_b$$

СВФ:

$$y = S_{k1} \vee S_{k2} \vee S_{k3}$$

$$y_f = S_f$$

Далее выполним разработку параллельного алгоритма в следующей последовательности.

На первом шаге проектирования параллельного алгоритма производим разбиение алгоритма по данным и функциям поиска цепочек.

Разбиение по данным. Поскольку задача поиска цепочек в заданной последовательности символов в своей основе последовательна, то разбить последовательность символов на части не представляется возможным. Поэтому данные могут быть разбиты только по числу входных последовательностей (P_i) символов. В нашем случае будет 10 задач (рис.7.1).

П1 106	П2 106	П3 206	П4 206	П5 306
П6 306	П7 306	П8 306	П9 406	П10 406

Рис. 7.1. Разбиение по данным

На следующем более низком уровне декомпозиции можно разбить каждую из последовательностей на блоки одинаковой длины (за исключением возможно последнего блока). И, казалось бы, в предельном случае минимальный размер блока может достигнуть 1 символа. На самом деле его длина не может быть выбрана любой, т. к. при выборе произвольной длины блока возможен случай, при котором часть цепочки окажется в одном блоке, а другая часть - в другом блоке. Тогда при поиске в каждом из этих блоков цепочка не будет обнаружена. Если выбирать длину блока в зависимости от максимальной длины искомой цепочки-образа можно избежать этого. Максимальная же длина цепочки-образа является постоянной для конкретного алгоритма поиска и поэтому при определении её длины не возникает трудностей. Необходимо учесть еще и то, что брать блоки данных из последовательности нужно с перекрытием, то есть начало следующего блока должно захватывать часть предыдущего, как показано на рис 7.2

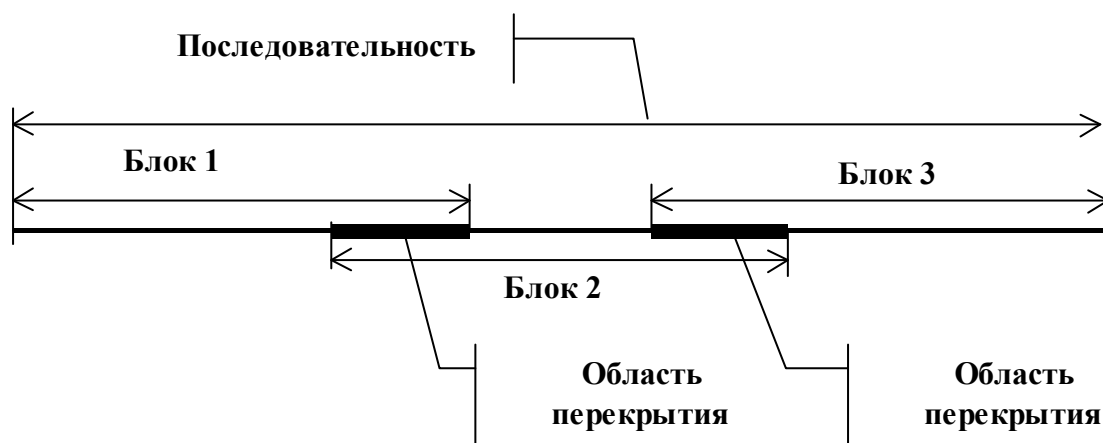


Рис.7.2

Если цепочки-образы не содержат итераций, то при таком разбиении на блоки поиск может быть выполнен параллельно во всех блоках. Если же цепочка-образ не имеет фиксированной длины (при её задании используется итерация), то алгоритм просмотра блоков последовательности усложняется, и сужаются возможности параллельной обработки. Длина области перекрытия определится в этом случае количеством символов в цепочке предшествующих внешней итерации. Причем если при поиске в блоке эта часть цепочки обнаружена, то в дальнейшем необходимо просматривать последовательно все следующие за этим блоки до тех пор, пока не будет обнаружено, что искомая цепочка прервалась или цепочка найдена.

Исходя из вышесказанного, длину области перекрытия можно определить следующим образом. Пусть максимальная длина цепочки (или её части предшествующей внешней итерации) равна N , тогда длина перекрытия L должна быть равна $L=N-1$. Единица вычитается для того, чтобы цепочка, если она есть, не вошла в последующий смежный блок. При таком условии невозможен случай, когда часть одной и той же цепочки лежала бы в двух смежных блоках. Естественно, что и длина перекрытия не может быть больше выбранной длины блока.

Такое деление на блоки соответствует условиям реальных задач, поскольку обычно длина искомой цепочки намного меньше длины файла, внутри которого она ищется.

Кроме того, при разбиении последовательности на блоки нужно учесть следующее. Если среди искомых есть цепочки, длины которых меньше N , то при поиске возможен случай, когда такие цепочки целиком будут лежать в двух смежных блоках в области перекрытия. Тогда при поиске они будут учтены дважды. Для решения этой проблемы можно предложить следующее. Если в блоке обнаружена такая цепочка, то нужно проверить - не лежит ли последний символ этой цепочки внутри области перекрытия. Если последний символ окажется в пределах области перекрытия, то полученный результат поиска можно игнорировать, т.к. эта цепочка будет найдена в следующем блоке.

На этом декомпозиция данных завершается, т.к. разбиение символа, как элемента данных, невозможно. Символ, являясь входным сигналом на очередном шаге алгоритма, неделим. Операции, полученные при такой декомпозиции, с входными данными очевидны, а именно, в зависимости от значения очередного входного сигнала происходит изменение состояния алгоритма поиска.

Разбиение по функциям, используемым в алгоритме, выполняем с учетом их иерархии. В алгоритме можно выделить следующие функции:

- 1) функция поиска одной цепочки-образа;
- 2) функция вычисления уравнения НДСКУ;

3) функции, соответствующие операциям при вычислении правой части уравнения НДСКУ.

Разбиение алгоритма по функциям поиска каждой цепочки приведено на рис.7.3.

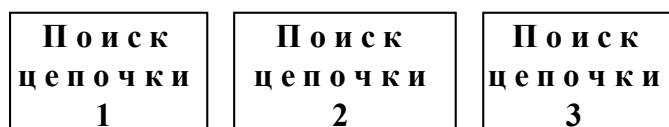


Рис. 7.3. Разбиение по функциям поиска цепочек

На следующем уровне функцию поиска отдельной цепочки можно разбить по функциям вычисления уравнений НДСКУ. Например, поиск цепочки 1 состоит в вычислении следующих уравнений: $S_{k1}=S_1z_1$; $S_1=S_2z_2 \vee S_1z_2$; $S_2=S_0z_1$ и соответствующее разбиение приведено на рис.7.4.



Рис.7.4. Разбиение по функциям вычисления уравнений

Далее, на следующем уровне иерархии, возможно разбиение вычислений правых частей уравнений. Например, уравнение для S_1 можно разбить на функции: конъюнкции, дизъюнкции и присвоения (рис.7.5).

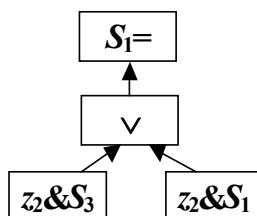


Рис.7.5. Разбиение правой части для S_1

В конечном итоге полное функциональное разбиение будет представлять собой дерево (рис.7.6), корнем которого будет определение вхождения какой-либо цепочки во входную последовательность, а далее соответственно по ярусам – нахождение конкретной цепочки, затем вычисление уравнений НДСКУ для этой цепочки и затем вычисление их правых частей. На рисунке не приведено разбиение по функциям при вычислении уравнения S_f , т.к. при практической реализации алгоритма достаточно проверять входной сигнал на равенство "*bottom*", и в случае обнаружения конца последовательности завершать алгоритм поиска. Из рассмотрения этого дерева хорошо видно, что на нижнем уровне выполняется операция конъюнкции, затем дизъюнкции и затем присвоения. Таким образом, в результате разбиения алгоритма по функциям получаются следующие элементарные задачи:

- 1) конъюнкция $S_i \& z_j$, где S_i - одно из событий НД СКУ, а z_j - входной сигнал;
- 2) дизъюнкция полученных конъюнкций;
- 3) присвоение.

Так как эти операции являются элементарными (неделимыми), то дальнейшее функциональное разбиение невозможно и декомпозицию алгоритма, как по данным, так и по функциям можно считать завершенной.

Анализируя полученные ЭЗ можно утверждать что:

- 1) число ЭЗ достаточно велико, и это позволит в дальнейшем оптимально распределять ЭЗ между процессорами;
- 2) с ростом числа одновременно обрабатываемых последовательностей возможны слишком большие затраты памяти (первый уровень декомпозиции данных), и это может привести к ограничению числа разбиений;
- 3) размеры ЭЗ практически равны, и это позволит в дальнейшем равномерно загрузить процессоры;
- 4) при увеличении размерности входных данных возрастает число ЭЗ, а размер ЭЗ остается неизменным, и это обеспечивает масштабируемость алгоритма поиска.

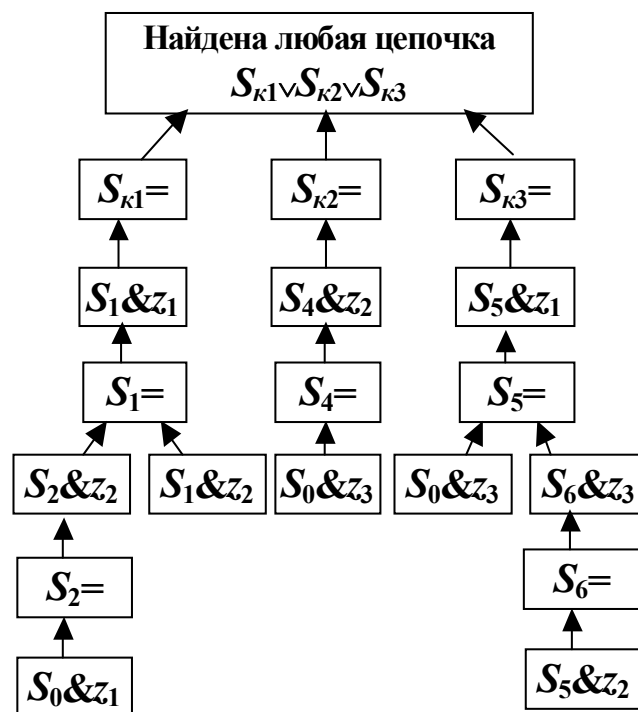


Рис.7.6. Полное функциональное разбиение

На втором шаге проектирования параллельного алгоритма определим необходимые коммуникации между ЭЗ, учитывая что:

- 1) входные последовательности и результаты поиска в каждой из них будут храниться в одном месте (оперативная память, файл на диске, и т.п.);

2) необходимо определить во входных последовательностях любую из трех цепочек, и поэтому при обнаружении какой либо цепочки в любой последовательности алгоритм завершается.

Определим необходимые коммуникации при разбиении алгоритма по данным (рис.7.8). При разбиении входной последовательности на блоки коммуникации будут аналогичными (при отсутствии итераций).

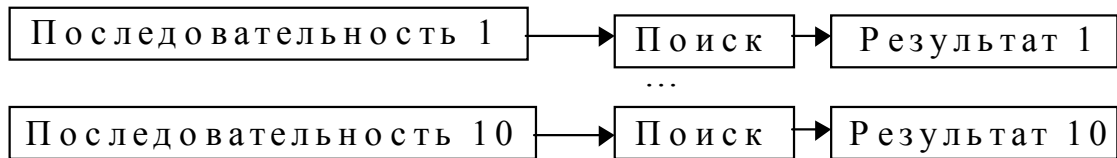


Рис.7.8. Коммуникации при разбиении алгоритма по данным

Определим необходимые коммуникации при разбиении алгоритма по функциям. При разбиении алгоритма по функциям поиска каждой цепочки коммуникации приведены на рис.7.9:

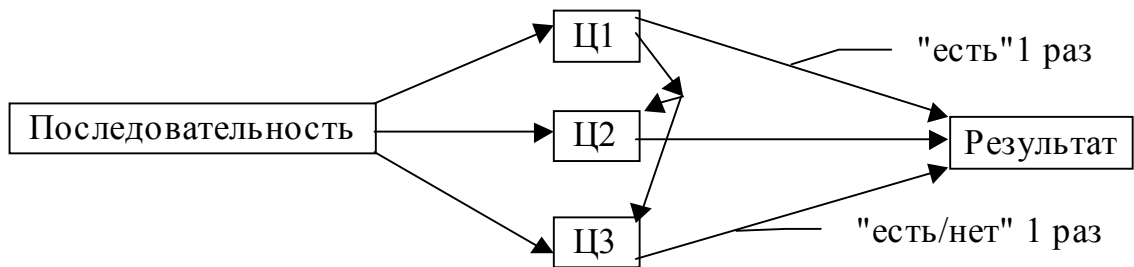


Рис.7.9. Коммуникации при разбиении по функциям поиска цепочки

На рис.7.9, из-за недостатка места, показана только пересылка сообщений («цепочка есть») от задачи, обнаруживающей первую цепочку к двум другим. Аналогичные сообщения пересылает каждая задача каждой.

Теперь рассмотрим коммуникации при комбинировании разбиения по данным и разбиения по функциям. Результат установления необходимых коммуникаций представлен на рис.7.10. Для максимального же разбиения по данным, и максимального разбиения по функциям необходимые коммуникации, представлены на рис.7.11.

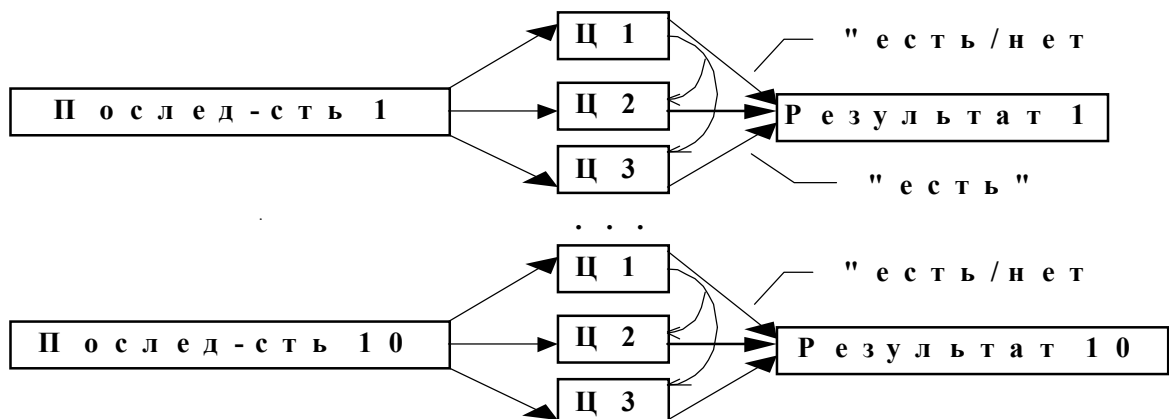


Рис.7.10. Коммуникации для разбиения по данным и функционального разбиения по цепочкам

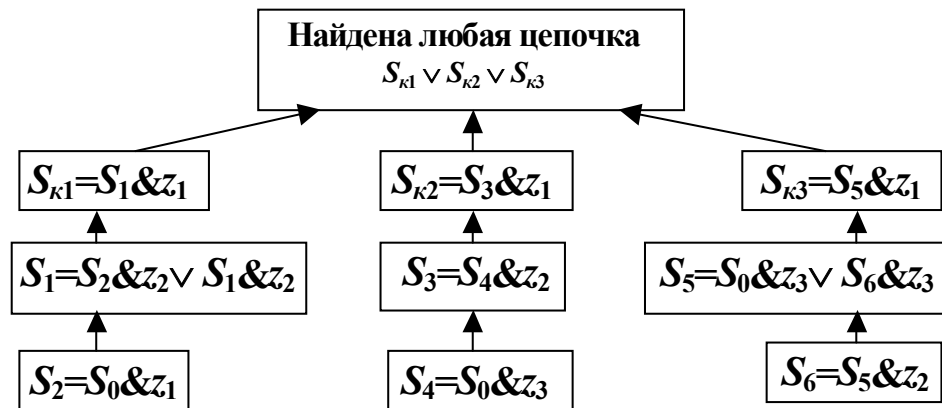


Рис. 7.11. Коммуникации для максимального разбиения по данным и максимального разбиения по функциям

Полученная при декомпозиции в качестве элементарной задачи конъюнкция требует для своего выполнения два входных параметра. Этими входами являются событие s_i и входной сигнал z_j , которые и определяют коммуникации в алгоритме. Поэтому более детальное рассмотрение коммуникаций по событиям и по входному сигналу в мультипроцессорной системе приводит к следующим выводам.

Вывод 1. В предельном случае (наихудшем) входной сигнал может потребоваться сразу для всех ЭЗ, поэтому для него разумно организовать глобальную коммуникацию. В то же время, используемое событие в каждой ЭЗ своё и поэтому для передачи событий лучше подойдет локальная коммуникация.

Вывод 2. И для входных сигналов и для событий коммуникация получается структурированной. Для входных сигналов это будет дерево, где корнем является входной сигнал, а ветвями - все ЭЗ. Для событий структура коммуникаций будет представлять собой дерево, в корне которого будут события, определяющие наличие какой-либо цепочки во входной последовательности, далее события, определяющие наличие конкретной цепочки, и наконец, события определяющие текущее состояние поиска этой цепочки.

Вывод 3. Т.к. при выполнении алгоритма поиска цепочки-образы не могут быть изменены, то коммуникации будут статическими.

Вывод 4. Если организовать мультипроцессорную систему, на которой будет выполняться алгоритм, таким образом, что интервалы времени между поступлениями очередных входных сигналов будут равны, то возможна использование синхронных коммуникаций. В противном случае необходимы асинхронные коммуникации.

Итоговая оценка полученных коммуникаций между ЭЗ дает следующие результаты:

- 1) система коммуникаций является структурированной и упорядоченной, что обеспечит в дальнейшем масштабируемость алгоритма;
- 2) количество коммуникаций для одной ЭЗ мало, что позволит добиться лучшей параллельности выполнения алгоритма;
- 3) коммуникации нижнего уровня независимы и поэтому могут выполняться параллельно;
- 4) некоторые коммуникации верхнего уровня не могут выполняться параллельно, но в связи с тем, что их количество в процентном отношении мало, они допустимы в алгоритме;
- 5) все ЭЗ нижнего уровня выполняются параллельно, т.к. не зависят друг от друга, а также от ЭЗ верхнего уровня.

На третьем шаге проектирования параллельного алгоритма произведем объединение ЭЗ. Рассмотрим вначале возможные варианты объединения ЭЗ, которые были получены при разбиении алгоритма по функциям. Как видно из рис. 7.11, можно провести объединение по вертикали или по горизонтали.

Для вертикального способа объединяются в одну задачу отдельные ветви дерева. Таким образом, в каждой ЭЗ осуществляется поиск одной конкретной цепочки. Результаты поиска цепочек объединяются в корне дерева. ЭЗ при этом получаются независимыми, то есть могут работать параллельно. Алгоритм будет достаточно масштабируемым, особенно хорошие результаты получатся, когда количество групп будет равно количеству процессоров. Но у такого объединения есть существенный недостаток. Дело в том, что ЭЗ нижнего уровня представляет собой конъюнкцию, а конъюнкция является одной из самых быстрых операций в ЭВМ. Следовательно, выигрыш в производительности при таком объединении будет незначительным из-за того, что время, затрачиваемое на пересылку входного сигнала в каждую из ЭЗ, будет значительно превосходить время обработки этого сигнала (конъюнкции).

Для горизонтального способа в одну ЭЗ объединяется нижний уровень дерева, т.е. все возможные конъюнкции. Это соответствует поиску подцепочки. Вышележащие уровни объединяются в другую ЭЗ, которая будет получать информацию об обнаружении подцепочки, о принадлежности этой подцепочки конкретному образу, анализировать ее и по завершении просмотра входных последовательностей выдавать окончательный результат поиска. При таком объединении ЭЗ нижнего уровня выполняются последовательно, что сужает параллелизм. Основным достоинством такого объединения является возможность разбиения ЭЗ нижнего уровня по данным. Т.е. одна ЭЗ будет представлять собой поиск всех возможных цепочек в первой последовательности, другая - во второй и так далее.

Рассмотрим теперь варианты объединений ЭЗ, которые были получены при разбиении алгоритма по данным. На этапе декомпозиции конечным результатом разбиения по данным (теоретически) получился один символ входной последовательности. Естественно, что при объединении символы можно объединить в блоки. Наиболее очевидным будет объединение в блок целиком входной последовательности. Заметим, что при решении практических задач поиска, этими последовательностями, например, являются файлы. И как отмечалось ранее, параллельная обработка многих файлов (сотен и более) потребует слишком много ресурсов оперативной памяти. Другой возможный вариант – это объединять входные сигналы по принадлежности цепочкам-образам. В этом случае необходимо реализовать фильтр, который анализировал бы входные сигналы и в зависимости от их принадлежности какой-либо цепочке, передавал бы их соответствующему процессу поиска. Этот метод дает хорошие результаты при «вертикальном» функциональном объединении, когда каждая цепочка ищется отдельным процессом. При таком способе экономится время на пересылку входного сигнала процессам, хотя при большом количестве цепочек он не будет эффективным. Кроме того, этот способ достаточно сложен в практической реализации. Наконец, можно объединять входные данные в последовательности некоторой определенной длины. Этот способ подходит для «горизонтального» функционального объединения. В этом случае входную последовательность можно разбить на блоки по числу процессоров, каждый из которых способен отыскивать любую из цепочек-образов. Тогда каждый процессор будет обрабатывать свой блок данных, независимо от других, а итоги передавать в процессор, который производит обработку результатов поиска.

Проведя анализ возможных вариантов объединения, выберем для дальнейшего проектирования параллельного алгоритма «горизонтальное» функциональное объединение и объединение данных в блоки, количество которых соответствует числу процессоров в системе.

На четвертом шаге проектирования параллельного алгоритма произведем выбор модели аппаратной реализации многопроцессорной системы. Модель мультипроцессорной системы будет определяться выбранным вариантом агломерации ЭЗ и требуемыми при этом коммуникациями между ними. Из предлагаемых ранее трех моделей произведем выбор по следующим соображениям. Сетка процессоров не подходит из-за отсутствия в алгоритме «регулярных» коммуникаций (от соседа к соседу). Общая шина не подходит из-за ее общего недостатка – «узости» при коммуникациях. Выбираем модель с общей памятью. Во-первых, она подходит по источникам и приемникам данных для решаемой задачи, а именно исходные данные хранятся в одном источнике, результаты помещаются в один источник. Во-вторых, каждый процессор может работать независимо от других, за исключением времени обмена с общей памятью. В-

третьих, наличие КЭШа достаточного размера между общей памятью и процессором позволяет сократить временные издержки на коммуникации.

На пятом шаге проектирования параллельного алгоритма работы многопроцессорной системы произведем его разработку в самом общем виде (эскизный проект). Учитывая полученные на предыдущих шагах результаты, можно предложить централизованную мультипроцессорную систему «Управляющий – исполнители» с общей памятью (рис.7.12).

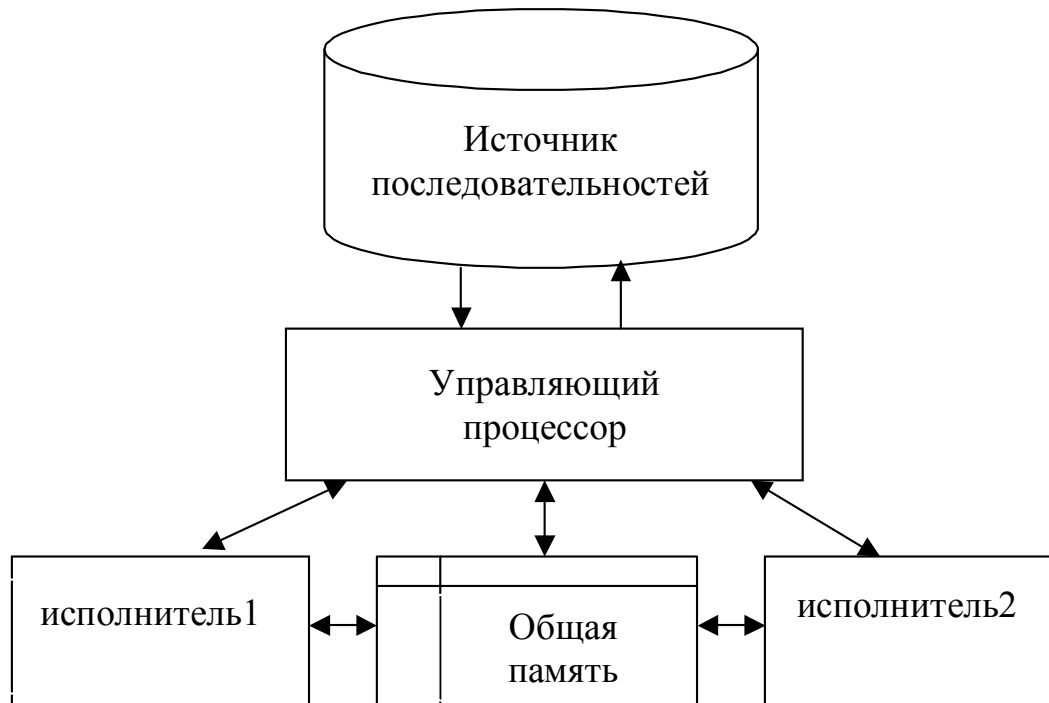


Рис.7.12. Архитектура мультипроцессорной системы

Для этой системы предлагается следующий алгоритм.

Для «управляющего» процессора:

- 1) выполнить начальную инициализацию системы;
- 2) определить через интерфейс пользователя источник последовательностей (наименование каталогов, файлов и т.д.);
- 3) открыть очередную входную последовательность (файл);
- 4) в соответствии с максимальной длиной цепочки - образа определить размер блока, на которые разбивается последовательность;
- 5) поочередно через общую память передать блоки процессорам «исполнителям», либо если позволяет размер общей памяти переписать всю последовательность в память и передать «исполнителям» только начальный адрес блоков;
- 6) анализировать получаемые от «исполнителей» сообщения, в том числе сигнал «конец работы»;
- 7) решать задачу верхнего уровня (устанавливать событие «найдена цепочка»);

8) Завершать обработку последовательности, для чего учитывать количество поступивших сигналов «конец работы». Когда их число станет равным числу занятых «исполнителей» (это число в принципе может быть меньше общего числа «исполнителей»), переходить к обработке следующей последовательности (см. п. 3).

Для процессора «исполнителя»:

1) читать очередной символ и осуществлять переходы в алгоритме;
2) если обрабатываемый блок закончился, и нет итерации, передать «управляющему» сигнал завершения обработки блока, иначе запросить блок, следующий в обрабатываемой последовательности за предыдущим.

3) если возникло состояние алгоритма «цепочка найдена», то передать сообщение «управляющему» об обнаружении цепочки (возможен вариант алгоритма, когда информация о найденных цепочках будет храниться у «исполнителя» до завершения обработки всех блоков, а затем пакетом передаваться «управляющему»);

Этот алгоритм можно усовершенствовать. При большом количестве цепочек-образов и достаточном количестве «исполнителей» возможно разделение алгоритма работы «управляющего» процессора на алгоритмы двух процессоров: «управляющего» и «анализатора». На «управляющего» будут возлагаться функции загрузки блоков последовательности, а на «анализатор» - прием сообщений от «исполнителей» и их обработка. По окончании обработки всех последовательностей «управляющий» получает от «анализатора» результаты. Добавление «анализатора» в мультипроцессорную систему целесообразно в том случае, когда объем оперативной памяти, как общей, так и процессоров мал по сравнению с длинами обрабатываемых последовательностей. В этом случае «управляющий» процессор будет тратить много времени на распределение блоков между рабочими процессорами. Архитектура такой системы приведена на рис.7.13.

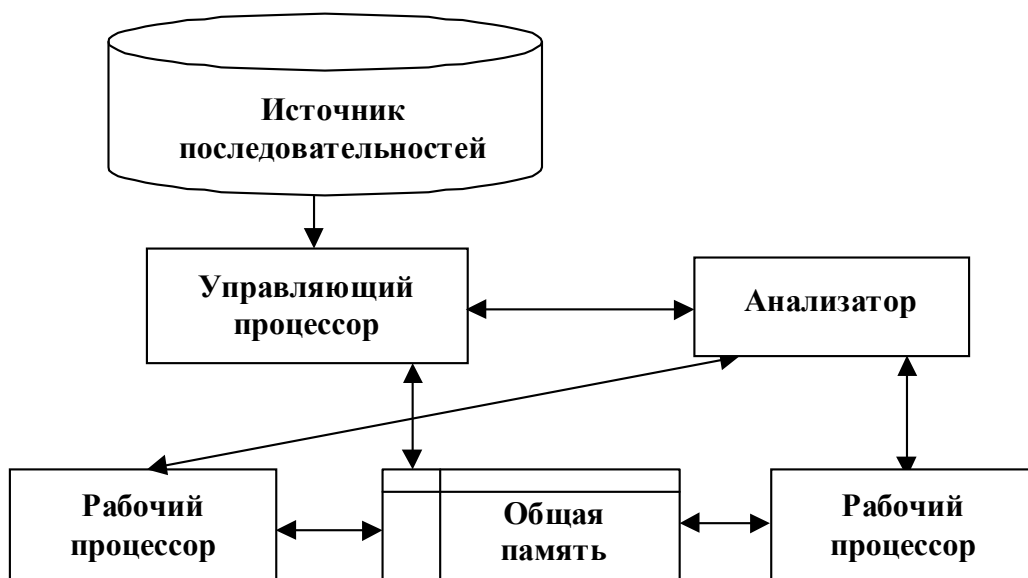


Рис.7.13 Мультипроцессорная система с анализатором

В предлагаемой мультипроцессорной системе при начале обработки очередной последовательности каждый «исполнитель» должен обратиться к «управляющему» с требованием данных. «Управляющий» распределяет имеющиеся блоки по «исполнителям». Каждый «исполнитель», обработав полученный блок, обращается к «управляющему» с запросом на получение следующего. Если у «управляющего» есть необработанные блоки, он передает очередной «исполнителю». В противном случае «исполнителю» передается сообщение об отсутствии необработанных блоков, после чего «исполнитель» передает «анализатору» сигнал об окончании работы. «Анализатор» получив сигналы окончания от всех «исполнителей», передает результат поиска «управляющему».

Произведем анализ потоков информации в системе. В ней будет существовать два основных потока информации: передача от «управляющего» к «исполнителям» входных данных и передача информационных сообщений между «управляющим», «анализатором» и «исполнителями». Эти два потока различаются по требуемой скорости и поэтому разумно организовать их передачу различными способами. Входные данные имеют относительно большой размер и должны передаваться как можно быстрее. Кроме того, блоки передаются последовательно, то есть из файла сначала считывается первый блок, который передается первому «исполнителю», затем второй блок, который передается второму «исполнителю» и так далее. Информационные сообщения в системе возникают значительно реже и скорость их передачи менее критична.

Если число «исполнителей» достаточно большое и производится поиск большого количества цепочек-образов, т.е. высока вероятность обнаружения какой-либо цепочки, то «анализатор» может оказаться узким местом в системе из-за большого числа пересылаемых ему сообщений. В этом случае возможно увеличение числа «анализаторов» в системе по методу «вертикального» функционального объединения. Каждый «анализатор» будет принимать только «свои» сообщения, отобранные по определенному признаку. Например, это могут быть определенные виды цепочек-образов или сообщения от определенной группы «исполнителей».

С учетом выше изложенного, предлагается следующий эскизный проект мультипроцессорной системы. В ее составе будут: процессор – «управляющий»; процессор – «анализатор»; два процессора – «исполнителя». Максимальная длина цепочки - 3 байта. Длина области перекрытия будет равна 2 байтам. Возможные типы сообщений в системе приведены ниже в таблице.

№	Источник	Приемник	Сообщение	Параметры
1	Рабочий	Управляющий	Свободен	Нет
2	Управляющий	Рабочий	Начать работу	Базовый адрес, размер блока

3	Рабочий	Анализатор	Найдена цепочка	Номер цепочки, относительный адрес конца цепочки, базовый адрес
4	Управляющий	Рабочий	Работы нет	Нет
5	Рабочий	Анализатор	Работу закончил	Нет
6	Анализатор	Управляющий	Поиск завершен	Пакет результатов поиска

Алгоритмы работы процессоров в системе представлены ниже.

Для «управляющего» процессора:

- 1) произвести инициализацию системы;
- 2) обеспечить интерфейс пользователя и получить через него указатель на последовательность для поиска;
- 3) открыть последовательность и определить её длину (L);
- 4) определить предварительную длину блока S для обработки «рабочим» ($S=L/2+2$);
- 5) если $S \leq 2$ (длина блока равна или меньше длины области перекрытия), то $S=L$ (при $L=0$, прекратить выполнение алгоритма);
- 6) если блок не помещается в оперативную память «исполнителя», то перейти к пункту 7;
- 7) установить базовый адрес - $B=0$;
- 8) указатель позиции файла (F) перевести на $B+S$. Если свободен первый «исполнитель», то передать ему текущий блок и затем сообщение типа 2 (с параметрами B и S). В противном случае, если свободен второй «исполнитель», то передать текущий блок и затем сообщение типа 2 (с параметрами B и S) второму «исполнителю». Если оба «исполнителя» заняты перейти в режим ожидания сообщений типа 1 (перейти к пункту 8);
- 9) если файл закончился, перейти к пункту 10, в противном случае $B=F-3$, $K=L-B$ (размер оставшейся части файла); если $K < S$, то $S=K$ перейти к блоку 9.
- 10) передавать на все запросы типа 1 от «исполнителей» ответы типа 4; ожидать прихода сообщения типа 6; после его получения вывести результаты поиска через интерфейс пользователя.
- 11) конец алгоритма.

Для процессора «исполнителя»:

- 1) передать сообщение типа 1;
- 2) ожидать сообщение типа 2 или 4; после получения сообщения типа 4 перейти к пункту 7; после получения сообщения типа 2 перейти к пункту 3;
- 3) установить указатель адреса на начало оперативной памяти;

4) считать очередной символ из оперативной памяти и выполнить вычисление НД SKU;

5) если появилось состояние «найдена цепочка», проверить, не лежит ли она в области перекрытия (цепочка лежит в области перекрытия, если относительный адрес ее окончания (G) меньше либо равен 2); если лежит, то перейти к пункту 4, иначе передать сообщение типа 3 (в качестве параметров относительного и базового адресов передать B и G соответственно);

6) перейти к пункту 4;

7) передать сообщение типа 5;

8) конец.

Для процессора «анализатора»:

1) установить счетчик числа «исполнителей» равным 2;

2) ожидать сообщения типа 3 или 5;

3) если получено сообщение типа 5, то перейти к пункту 6;

4) если получено сообщение типа 3, то вычислить абсолютный адрес конца цепочки: $A=B+G$ и записать полученный результат вместе с номером цепочки-образа в стек сообщений;

5) перейти к пункту 2;

6) вычесть из счетчика числа «исполнителей» единицу и если счетчик равен 0, то перейти к пункту 7, иначе перейти к пункту 2;

7) передать сообщение типа 6;

8) конец.

На шестом шаге произведем оценку эффективности многопроцессорной системы. При оценке эффективности многопроцессорной системы относительно однопроцессорной, нужно вначале определить оптимальное число процессоров в системе, т.к. увеличение числа процессоров в системе не дает увеличение скорости работы в такое же количество раз. Для этого выберем критерии, по которым будет определяться число процессоров в системе. Из анализа результатов, полученных при установлении коммуникаций, видно, что скорость работы системы будет в большей степени определяться скоростью передачи входной последовательности от места её хранения (физического носителя) до «управляющего» процессора. Эта скорость будет намного меньше, чем скорость передачи информации между процессорами системы. Поэтому определение эффективного числа процессоров будет вестись исходя из этого положения. Оптимальным количеством процессоров будет то, при котором первый "исполнитель" заканчивает работу в тот момент, когда «управляющий» заканчивает передачу данных последнему «исполнителю». В этом случае ни один из процессоров не будет простаивать. Если же: количество процессоров будет меньше, то простаивать будет «управляющий». Но это время может быть использовано «управляющим» процессором для выполнения других функций, кроме решения задач

обслуживания «исполнителей». Например, для удаления обнаруженных цепочек-образов, обеспечения интерфейса пользователя, и т.д. Если же количество процессоров будет больше оптимального, то будет простаивать часть процессоров «исполнителей», так как до них задания от «управляющего» не придут. Это будет происходить из-за того, что первый «исполнитель» вновь передаст запрос на получение следующего блока раньше, чем процессор сможет загрузить все «исполнители». Таким образом, увеличение числа процессоров не будет повышать скорость работы системы.

Увеличение размера входных данных в общем случае не влияет на величину оптимального количества процессоров. Если длина блока данных не изменяется, а увеличивается только их количество, то в алгоритме работы системы ничего не изменится (увеличится лишь общее время обработки). Если же увеличивается размер блока, то увеличивается время обработки этого блока процессором «исполнителем», но и увеличивается суммарное время передачи блока «исполнителю». Если обеспечить в системе, что эти два времени будут равны, то она будет работать без простоев.

Оценим эффективность мультипроцессорной системы при оптимальном количестве процессоров. Время работы однопроцессорной системы T_0 :

$T_0 = (T_{\text{счит}} + T_{\text{обrab}}) * n$, где $T_{\text{счит}}$, $T_{\text{обrab}}$ – время считывания и обработки одного байта, а n – число байт в последовательности.

Представим графически работу каждого процессора «исполнителя» в мультипроцессорной системе во времени без учета взаимодействия с «управляющим» и «анализатором» (рис.7.14).

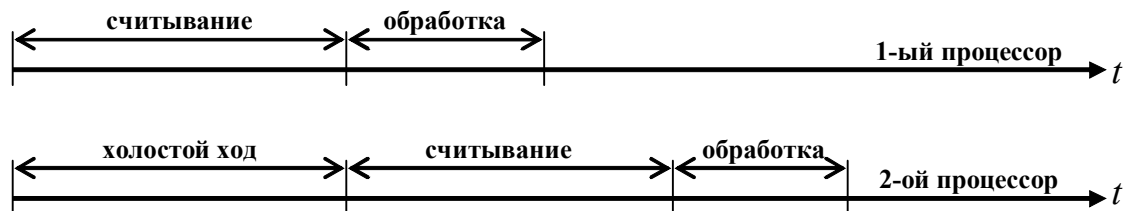


Рис.7.14 Загрузка работой во времени процессоров «исполнителей»

Тогда приближенно время работы многопроцессорной системы T_m :

$$T_m = T_{\text{счит}} * n + T_{\text{обrab}} * n/2$$

Эффективность выполнения алгоритма при параллельной реализации по сравнению с последовательной будет приблизительно равна:

$$E = \frac{T_0}{T_m} = \frac{2}{\frac{T_{\text{счит}}}{T_{\text{счит}} + T_{\text{обrab}}} + 1}$$

Следует отметить, что полученное выражение показывает максимальную эффективность в результате того, что ранее были введены

некоторые допущения. В действительности эффективность будет ниже величины полученной при помощи этой формулы. Из формулы также видно, что если $T_{\text{обrab}}$ намного меньше $T_{\text{чит}}$, то эффективность системы равна 1, т.е. такой алгоритм работы многопроцессорной системы будет неэффективен.

В заключение сделаем замечания по практической **аппаратной реализации алгоритма**. Аппаратная реализация отлаженного алгоритма в этом случае чрезвычайно проста, если в качестве элементарного автомата памяти использовать D триггер. При этом количество триггеров в схеме соответствует числу уравнений НД СКУ плюс количество начальных состояний в алгоритме. Каждое уравнение будет представлять собой функцию возбуждения соответствующего ему триггера. Таким образом будет получено описание схемы цифрового устройства на входном языке, которое может быть использовано в системах автоматизированного проектирования (CAD/CAM).

Контрольные вопросы к главе 7.

1. Каковы алгоритмы поиска цепочек символов во входной последовательности при решении следующих задач:

а) поиск наличия любой из нескольких заданных цепочек символов в произвольной входной последовательности символов?

б) поиск наличия всех заданных цепочек символов в произвольной входной последовательности символов?

в) поиск наличия заданного количества любых из нескольких заданных цепочек символов в произвольной входной последовательности символов?

г) поиск наличия нескольких заданных цепочек символов, у которых префикс и суффикс одинаковы, а корни различны в произвольной входной последовательности символов?

д) поиск наличия заданной цепочки с использованием маскирования некоторых её символов в произвольной входной последовательности символов?

е) поиск наличия заданной цепочки символов в произвольной входной последовательности символов, когда этот поиск должен быть начат по сигналу извне?

2. Как можно динамически управлять алгоритмом поиска с использованием маркера начала поиска?

3. Какие этапы содержит предложенная методика проектирования параллельных алгоритмов?

4. Почему локализация коммуникаций позволяет сократить время выполнения алгоритма?

5. Какие стратегии используются при распределении задач по процессорам?

6. Предложите способы обнаружения момента времени завершения алгоритма работы многопроцессорной системы для централизованных и децентрализованных схем управления ею.

7. Что такое масштабируемость алгоритма решения задачи на многопроцессорной системе?

ЗАКЛЮЧЕНИЕ

Отметим наиболее важные достоинства способа представления алгоритмов управления на основе использования модели НДА и ее аналитического представления на языке НД СКУ и методов его использования для синтеза систем логического управления на разных уровнях преобразования информации.

1) Математическая модель алгоритмов логического управления, представленная на языке НД СКУ, совместно с графическим представлением ее на языке ГСАП, является достаточно простой и универсальной, позволяющей ее использовать для широкого круга применений по формализации алгоритмов управления объектами и процессами, начиная от простейших устройств управления выполнением арифметических операций в процессоре и кончая управлением процессами и ресурсами в вычислительных системах и сетях. При этом обеспечивается сравнительная легкость представления информации об алгоритме управления в ЭВМ с помощью списочных структур для целей автоматизации проектирования и моделирования.

2) Способ обладает компактностью представления алгоритмов управления, в том числе алгоритмов с взаимодействующими параллельными ветвями с любой степенью параллелизма, что в значительной степени, позволяет снизить ограничения на размерность проектируемых и моделируемых систем управления, понимая под размерностью произведение числа событий на число входных сигналов.

3) Наиболее естественно (по шагам) отражается алгоритм работы управляющего устройства, что позволяет осуществить непосредственный переход к различным вариантам структурных схем проектируемого устройства. В частности, если каждому событию поставить в соответствие *D*-триггер, то правые части уравнений СКУ будут функциями возбуждения этих триггеров.

4) Способ допускает относительно легкий переход к СКУ для систем управления, заданных на различных начальных языках, что позволяет с единых позиций подойти к синтезу таких систем независимо от первоначального формального описания их на одном из начальных языков.

5) Математический аппарат эквивалентных преобразований НДА, представленный в учебном пособии, может успешно сочетаться с хорошо разработанным аппаратом математической логики и многими важными положениями теории цифровых автоматов для выполнения различных эквивалентных преобразований СКУ и СВФ на различных этапах синтеза систем логического управления.

6) В связи с тем, что операции детерминизации и кодирования НДА являются операциями обратными друг другу, представляется возможность

организации некоторых эффективных процедур контроля и верификации, получаемых в процессе синтеза различных систем канонических уравнений.

7) Представление состояний (полных событий) в виде совокупности частных событий, одновременное существование которых в системе управления возможно, позволяет использовать для структурного синтеза систем управления другой подход к кодированию состояний системы управления, когда код каждого состояния представляется в виде композиции кодов групп несовместимых частных событий. Такое кодирование приводит к значительному упрощению как структуры системы управления, так и процедур их диагностики.

8) Язык НД СКУ может быть успешно использован для формализации алгоритмов управления взаимодействующими асинхронными параллельными процессами с решением задач обмена данными между процессами и их синхронизации. Это перспективное направление использования языка НД СКУ может быть значительно расширено и дополнено с учетом решения задач структурной реализации и моделирования автоматных недетерминированных и детерминированных моделей систем параллельной и распределенной обработки информации.

9) В пособии были рассмотрены также вопросы построения структур микропрограммного управления на основе использования языка НД СКУ для систем параллельной обработки. Такие структуры отличаются высокой производительностью и простотой. Используя их формальное представление на языке НД СКУ можно успешно решать задачи автоматизации отдельных этапов их структурного синтеза.

10) Из анализа работ, посвященных теории синтаксического анализа, перевода и компиляции [3, 6, 7, 78], в которых авторы используют модель НДА, следует, что представленные в данном учебном пособии результаты в области теории НДА и их эквивалентных преобразований, могут быть успешно использованы также и при проектировании различных синтаксических анализаторов и компиляторов.

Учитывая отмеченные выше достоинства представления алгоритмов управления на языке НД СКУ можно отметить, что этот язык позволяет в некоторой степени приблизить решение задачи, поставленной Ч.Хоаром в [55], по поиску как можно более простой математической теории, позволяющей:

а) описывать широкий круг применений по управлению процессами и дискретному моделированию событий, начиная с торговых автоматов и кончая операционными системами с разделяемыми ресурсами;

б) обеспечивать эффективную реализацию систем управления от простейших автоматов до мультипроцессорных устройств и сетей взаимодействующих микропроцессоров;

в) обеспечивать комплексное решение вопросов в деле спецификации, разработки, реализации, верификации и анализа сложных систем управления,

в том числе управления процессами и ресурсами в вычислительных системах и сетях.

ЛИТЕРАТУРА

1. Глушков В.М. Синтез цифровых автоматов. – М.: Физматгиз, 1962. – 476 с.
2. Вашкевич Н.П. Синтез микропрограммных управляющих автоматов: Учебное пособие – Пенза: политехн. ин-т, 1990. – 115 с.
3. Льюис Ф., Розенкранц Д., Стирнз Р., Теоретические основы проектирования компиляторов. – М.: Мир, 1979, 654 с.
4. Рабин М.О., Скотт Д., Конечные автоматы и задачи их разрешения // Кибернетический сборник, №4 – М.: ИЛ, 1962.
5. Трахтенброт Б.А., Барздин Я.И. Конечные автоматы (поведение и синтез). – М.: Наука, 1970 – 400 с.
6. Ахо А., Ульман Дж., Теория синтаксического анализа, перевода и компиляции. т. 1, М.: Мир, 1978 – 612 с.
7. Рейуорд-Смит В. Дж., Теория формальных языков. Вводный курс. – М.: Радио и связь, 1988., 129 с.
8. Кузнецов О.П., Андельсон-Вельский Г.М. Дискретная математика для инженеров. – М.: Энергия, 1980, 341 с.
9. Зубков В.А., Вашкевич Н.П., Синтез абстрактных цифровых автоматов с использованием языка рекурсивных предикатов // Вычислительная техника: Учебные записки. – Пенза: Пенз. политехн. ин-т, 1969 – Вып. 3. – С. 3–12.
10. Вашкевич Н.П., Пучков В.Г., Проектирование алгоритмов и структур цифровых устройств: Учебное пособие. – Пенза: Пенз. политехн. ин-т, 1978. – 110 с.
11. Вашкевич Н.П., Пучков В.Г., Синтез цифровых управляющих автоматов на основе языка систем канонических уравнений: Учебн. пособие – Пенза: Пенз. политехн. ин-т, 1980 – 100 с.
12. Зубков В.А., Вашкевич Н.П. Алгоритм синтеза цифровых автоматов Мура с использованием языка исчисления предикатов // Вычислительная техника: Учебные записки. – Вып. 3. Пенза: Пенз. политехн. ин-т, 1969 – С. 25–36.
13. Айзерман М.А., Гусев Л.А., Розоноэр Л.И., Смирнова И.М., Таль А.А. Логика. Автоматы. Алгоритмы. – М.: Гос. изд. физмат. литературы, 1963, 556 с.
14. Кузнецов О.П. О сравнительной теории алгоритмических языков логического управления. В кн.: «Теория дискретных управляющих устройств». – М.: Наука, 1982, 252 с.
15. Клини С.К. Представление событий в нервных сетях и конечных автоматах, в сб. «Автоматы». – М.: ИЛ, 1956.

16. Вашкевич Н.П. Об одном способе синтеза цифровых автоматов по граф-схеме алгоритма с параллельными ветвями // Вычислительная техника: Межвуз. сб. науч. тр., Вып. 1,2. – Пенза: Пенз. политехн. ин-т, 1973.
17. Вашкевич Н.П. Способ синтеза цифровых управляющих автоматов, заданных на языке логических схем алгоритмов (ЛСА) с параллельными ветвями // Специализированные и комбинированные вычислительные устройства: Межвуз. сб. науч. тр. – Вып. 2, Рязань; Рязан. радиотехн. ин-т, 1975.
18. Вашкевич Н.П., Пучков В.Г. Описание алгоритмов функционирования управляющих автоматов с параллельными ветвями на языках ЛСА и ГСА // Вопросы радиоэлектроники. – Сер. ЭВТ. – Вып. 13, 1977.
19. Вашкевич Н.П., Пучков В.Г. Построение системы канонических уравнений для управляющих автоматов, заданных на языках ЛСА и ГСА с параллельными ветвями // Вопросы радиоэлектроники. Сер. ЭВТ. – Вып. 13, 1977.
20. Вашкевич Н.П. Особенности одного способа представления алгоритмов функционирования цифровых управляющих автоматов. Специализированные и комбинированные вычислительные устройства: Межвуз. сб. науч. тр. – Вып. 2. – Рязань, Рязан. радиотехн. ин-т, 1975.
21. Вашкевич Н.П., Вашкевич С.Н. Недетерминированные автоматы и их использование для синтеза систем уравнений. Часть 1. Эквивалентные преобразования недетерминированных автоматов: Учеб. пособие. – Пенза: изд-во Пенз. гос. техн. ун-та, 1996, 88 с.
22. Зубков В.А., Вашкевич Н.П. Описание регулярных выражений алгебры событий рекурсивными предикатами. Сб. Вычислительная техника. Приволжское книжное издательство. Саратов – Пенза, 1968.
23. Баранов С.И. Синтез микропрограммных автоматов. – Л.: Энергия, 1979. – 231 с.
24. Дьяченко В.Ф., Лазарев В.Г., Саввин Г.Г., Управление на сетях связи. – М.: Наука, 1967.
25. Лазарев В.Г., Пийль Е.И., Турута Е.Н. Построение программируемых управляющих устройств. – М.: Энергоатомиздат, 1984. – 192 с.
26. Лазарев В.Г., Пийль Е.И. Синтез управляющих автоматов. – М.: Энергоатомиздат, 1989, 328 с.
27. Гилл А. Введение в теорию конечных автоматов. – М.: Наука, 1966, 272 с.
28. Вашкевич Н.П. Минимизация систем канонических уравнений, представляющих цифровой автомат, с учетом распределения сдвигов. Сб. Вопросы радиоэлектроники, ЭВТ, Вып. 10, 1980.
29. Клини С.К. Введение в метаматематику. – М.: ИЛ, 1957.

30. Кутепов В.П. О тождественных преобразованиях регулярных выражений. В сб. «Цифровая вычислительная техника и программирование». Вып. 2. – М.: Изд. Сов. радио, 1967.
31. Вавилов Е.Н., Портной Г.П. Синтез схем электронных цифровых машин. – М.: Изд. Сов. радио, 1963, 440 с.
32. Вашкевич Н.П. Особенности одного способа представления алгоритмов функционирования цифровых управляющих автоматов. Специализированные устройства. Межвуз. сб., Вып. 2., Рязань, Рязанский радиотехн. ин-т, 1975.
33. Котов В.Е. Языки параллельного программирования. В кн.: «Алгоритмы, математическое обеспечение и архитектура микропроцессорных вычислительных систем». – М.: Наука, - 1982.
34. Кобринский Н.Е., Трахтенброт Б.А. Введение в теорию конечных автоматов. – М.: Физматгиз, 1962.
35. Черч А. Введение в математическую логику. Том 1. – М.: ИЛ, 1960.
36. Новиков П.С. Элементы математической логики. – М.: Физматгиз, 1959.
37. Вашкевич С.Н., Филатов А.Ю. Структурный синтез управляющих автоматов, заданных НДА: Материалы международной науч.-техн. конф. «Новые информационные технологии и системы». – Пенза, 1994 – с. 123.
38. Котов В.Е. Формальные модели параллельных вычислений. В кн.: «Алгоритмы, математическое обеспечение и архитектура микропроцессорных вычислительных систем». – М.: Наука, - 1982.
39. Барашенков В.В. Временное совмещение алгоритмов. В сб. «Вычислительная техника», Вып. 2. – Л.: Энергия, 1972.
40. Моисеева Г.Н. О синтезе автоматов по графам алгоритмов с параллельными ветвями. В сб. «Вычислительная техника и вопросы кибернетики», Вып. 9, изд. МГУ, 1979.
41. Аперидические автоматы / Под ред. Варшавского В.Н. – М.: Наука, 1976, 232 с.
42. Гаврилов С.А., Девятков В.В., Пупырев Е.И. Логическое проектирование дискретных автоматов. – М.: Наука, 1977.
43. Исьянов В.М. О логических схемах алгоритмов второго рода. В сб. «Синтез дискретных автоматов и управляющих устройств». – М.: Наука, 1968.
44. Миллер Р. Теория переключательных схем. Том II. – М.: Наука, 1971, 304 с.
45. Янов Ю.И. О логических схемах алгоритмов. В сб.: «Проблемы кибернетики». – М.: Наука, 1958.

46. Миклошко И. Синтез параллельных алгоритмов. В кн.: «Алгоритмы, математическое обеспечение и архитектура многопроцессорных вычислительных систем». – М.: Наука. - 1982.
47. Майоров С.А., Новиков Г.И. Структура электронных вычислительных машин. – Л.: Машиностроение, 1979, 383 с.
48. Мотоока Т., Томата С., Таната Х. и др., Компьютеры на СБИС. – В 2-х кн. – кн. 1. – М.: Мир, 1988, 387 с.
49. Вашкевич Н.П. О кодировании недетерминированных автоматов. Материалы Межреспубликанской конференции «Повышение эффективности средств обработки информации на базе математического и машинного моделирования». – Тамбов, 1993, С. 8–9.
50. Вашкевич Н.П., Вашкевич С.Н. Формализация алгоритмов управления с взаимодействующими параллельными ветвями. Межвуз. сб. науч. тр. – Вып. 18. – Пенза: изд-во Пенз. гос. техн. ун-та, 1993. 152 с.
51. Вашкевич Н.П. Синтез автоматных моделей управления параллельными взаимодействующими процессами. Материалы Международной научно-технич. конференции «Новые информационные технологии и системы». – Пенза, 1994.
52. Вашкевич Н.П., Вашкевич С.Н. Формализация взаимодействия параллельных ветвей алгоритмов управления. Материалы Межреспубликанской конференции «Повышение эффективности средств обработки информации на базе математического и машинного моделирования» – Тамбов, 1993. С. 259–260.
53. Вашкевич Н.П. Аппаратно-программная реализация микропрограммных устройств управления с параллельными ветвями. Сб. Тезисов II Всесоюзной НТ конференции «Микропроцессорные системы автоматизации». – Новосибирск, 1990.
54. Вашкевич Н.П. Микропрограммные системы управления для параллельных алгоритмов. – Вопросы радиоэлектроники, серия ЭВТ, вып. 13, 1990.
55. Хоар Ч. Взаимодействующие последовательные процессы. – М.: МИР, 1989, 264 с.
56. Ульман Дж. Д. Вычислительные аспекты СБИС. – М.: Радио и связь, 1990, 480 с.
57. Грис Д. Конструирование компиляторов для цифровых вычислительных машин. – М.: МИР, 1975, 456 с.
58. Ахо А., Хопкрофт Дж., Ульман Дж. Построение и анализ вычислительных алгоритмов. – М.: МИР, 1979. – 535 с.
59. Foster I. an «Designing and Building Parallel Programs»: Конспект лекций. Электронный адрес – www.mcs.anl.gov.
60. Вашкевич Н.П. Формализация алгоритма взаимодействия параллельных процессов в задаче «читатели-писатели». Новые

информационные технологии и системы: Труды V Международной науч.-техн. конф. – Пенза, ПГУ, 2002, с. 204–208.

61. Шоу А. Логическое проектирование операционных систем: Пер. с англ. – М.: МИР, 1981, 360 с.

62. Дейтел Г. Введение в операционные системы: в 2-х т., Т.1, пер. с англ. – М.: МИР, 1987, 359 с.

63. Исии Т. Симояма И., Икоуэ Х. и др. Мехатроника. – М.: МИР, 1988. – 314 с.

64. Олифер В.Г., Олифер Н.А. Сетевые операционные системы. – СПб.: Питер, 2001. – 544 с.

65. Калинин А.Г., Мацкевич И.В. Универсальные языки программирования. – М.: Радио и связь. 1991, 400 с.

66. Гордеев А.В. Молчанов А.Ю. Системное программное обеспечение. – СПб.: Питер, 2001. – 736 с.

67. Артемьев М.Ю. Программное обеспечение управляющих систем электросвязи. – М.: Радио и связь, 1990, – 272 с.

68. Робачевский А. Операционная система UNIX. – СПб.: БХВ–Петербург. 2002. – 528 с.

69. Аджиев В. Мифы безопасности ПО: Уроки знаменитых катастроф // Открытые системы № 6, 1998, с. 34.–50.

70. Кейлингерт П. Элементы операционных систем: Пер. с. англ. – М.: МИР, 1985, 295 с.

71. Вашкевич Н.П. Автоматный подход к формализации алгоритмов управления взаимодействием параллельными процессами при обращении к общим переменным (общему ресурсу) для n-процессов. Межвузовский сборник научных трудов «Вычислительные системы и технологии», выпуск 1(27). – Пенза: изд-во Пенз. гос. ун-та, 2002, с. 3–10.

72. Танненбаум Э. Современные операционные системы. 2-е изд. – СПб.: Питер, 2002. – 1040 с.

73. Ульман Д. Основы систем баз данных. – М.: Финансы и статистика, 1983, 334 с.

74. Вашкевич Н.П. Формализация алгоритма взаимодействия параллельных процессов в задаче «читатели-писатели» // Новые информационные технологии и системы: Труды V Международной научно-технической конференции. – Пенза, ПГУ, 2002, с. 204–208.

75. Соловьев Г.Н., Никитин В.Д. Операционные системы ЭВМ: Учебное пособие. – М.: В.шк., 1989. 255 с.

76. Вальковский В.А., Котов В.Е., Марчук А.Г., Миренков Н.Н. Элементы параллельного программирования. / Под ред. Котова В.Е.– М.: Радио и связь, 1983 – 240 с.

77. Вальковский В.А., Котов В.Е. Автоматическое построение параллельных программ. В кн.: «Алгоритмы. Математическое обеспечение и

архитектура многопроцессорных вычислительных систем». – М.: Наука. – 1982.

78. Вашкевич Н.П., Калиниченко Е.И. Проектирование параллельных алгоритмов в задачах идентификации. Учебное пособие. – Пенза: изд-во Пенз. гос. ун-та, 2000, 72 с.

79. Вашкевич Н.П., Калиниченко Е.И. Автоматная модель распределенных вычислительных систем. // Новые информационные технологии и системы. Труды VI Международной научно-технической конференции. – Пенза, ПГУ, 2004.

80. Оллонгрен А. Определение языков программирования интерпретирующими автоматами. – М.: Мир, 1977. 228 с.

81. Ганькин А.Л., Сидоров Ю.В., Соловьев Г.Н. Исследования и разработки в области аппаратной реализации алгоритмов операционных систем // Зарубежная электроника, 1984, №2, с. 34–50.

82. Назаров С.В. Операционные системы специализированных вычислительных комплексов: Теория построения и системного проектирования. – М.: Машиностроение, 1989, 400 с.

83. Майерс Г. Архитектура современных ЭВМ. В 2-х кн. Пер. с англ. – М.: МИР, 1985, кн. 1 – 364 с., кн. 2 – 312 с.

84. Вашкевич Н.П. Преобразование структуры управляющих алгоритмов, заданных моделью НДА для построения распределенных управляющих систем параллельной обработки. / Актуальные проблемы науки и образования : Труды международного юбилейного симпозиума. В 2-х томах, Т.2. – Пенза: информационно-издательский центр ПГУ, 2003, с.375-377.

Оглавление

Предисловие

Глава 1. Основные понятия и определения из теории недетерминированных автоматов	
1.1. Определение и понятия недетерминированного автомата	
1.2. Общие сведения о выборе языков для представления недетерминированных автоматов	
1.3. Аналитическое представление недетерминированных автоматов на стандартном языке	
1.4. Расширение выразительных возможностей языка НД СКУ и его отличительные особенности	
1.5. Иерархия входных сигналов и событий, реализуемых в системах логического управления	
Глава 2. Детерминизация недетерминированных автоматов	
2.1. Алгоритм детерминизации НДА	
2.2. Пример детерминизации НДА с построением СКУ и СВФ для автоматов Мура и Мили	
Глава 3. Минимизация систем канонических уравнений (СКУ)	
3.1. Минимизация СКУ на основе определения эквивалентного разбиения событий	
3.1.1. Некоторые понятия и определения	
3.1.2. Определение эквивалентного разбиения событий СКУ на основе использования таблицы пар	
3.1.3. Определение минимальной прямой таблицы переходов и минимальной СКУ для автомата Мили и Мура	
3.2. Минимизация СКУ на основе учета распределения сдвигов	
3.2.1. Методика минимизации СКУ	
3.2.2. Пример минимизации СКУ	
Глава 4. Начальные языки, используемые для представления управляющих алгоритмов параллельной обработки информации	
4.1. Язык регулярных выражений алгебры событий (РВАС) и построение СКУ на его основе	
4.1.1. Основные понятия и определения	

языка РВАС	
4.1.2. Построение СКУ по регулярному выражению алгебры событий и ее детерминизация	
4.2. Язык исчисления предикатов первого порядка	
4.2.1. Описание регулярных выражений алгебры событий рекурсивными предикатами	
4.2.2. Формализация словесных алгоритмов, определяющих реализуемые в цифровом автомате события, на основе использования языка исчисления предикатов первого порядка	
4.2.3. Построение СКУ по описанию алгоритма обработки информации, представленным на языке исчисления предикатов первого порядка	
4.2.4. Пример синтеза абстрактного цифрового автомата с использованием языка исчисления предикатов первого порядка	
4.3. Язык граф-схем алгоритмов (ГСА)	
4.3.1. Общие сведения о языке ГСА	
4.3.2. Построение таблиц переходов и СКУ по ГСА	
4.4. Логические схемы алгоритмов (ЛСА)	
4.4.1. Общие сведения о языке ЛСА	
4.4.2. Построение таблиц переходов и СКУ по ЛСА	
4.5. Язык операторных схем алгоритмов с параллельными ветвями (ОСАП)	
4.5.1. Общие сведения о языке ОСАП	
4.5.2. Основные конструкции, вводимые в языке ГСАП	
4.5.3. Построение таблиц переходов и СКУ для цифровых автоматов, заданных на языке ГСАП	
Глава 5. Структурный синтез систем микропрограммного управления (МПУ), заданных моделью НДА	
5.1. Структурная реализация систем МПУ на основе разбиения частных событий на группы несовместимых событий	
5.1.1. Преобразование структуры управляющего алгоритма, представленного моделью НДА, для построения распределенной системы МПУ параллельной обработки	
5.1.2. Методика разбиения частных событий на группы несовместимых событий для управляющего алгоритма, заданного моделью НДА	

5.1.3. Пример построения структуры распределенной системы МПУ по управляющему алгоритму с взаимодействующими параллельными ветвями	
5.2. Структурная реализация систем МПУ, заданных моделью НДА, на основе использования унитарного кодирования частных событий	
5.2.1. Одноуровневая организация структуры системы МПУ для унитарного кодирования частных событий	
5.2.2. Двухуровневая организация структуры МПУ для унитарного кодирования частных событий	
5.3. Некоторые операции композиции систем канонических уравнений и их структурная интерпретация	
5.4. Контроль правильности построения функций возбуждения элементов памяти систем МПУ	
Глава 6. Формализация алгоритмов управления взаимодействующими процессами для организации параллельной обработки информации	
6.1. Общие сведения о процессах и их взаимодействиях	
6.2. Формализация простейших базовых структур управления взаимодействующими процессами	
6.3. Формализация функций взаимоисключения критических интервалов (участков), обеспечивающих доступ к общим разделяемым данным (общему ресурсу) для двух процессов	
6.4. Формализация алгоритма управления взаимодействующими параллельными процессами при обращении к разделяемым данным (общему ресурсу)	
6.4.1. Формализация алгоритма управления взаимодействующими параллельными процессами при обращении к разделяемым данным для двух процессов	
6.4.2. Формализация алгоритма управления взаимодействующими параллельными процессами при обращении к разделяемым данным (общему ресурсу) для n процессов	
6.5. Формализация алгоритма управления взаимодействующими параллельными процессами в задаче «производители-потребители»	
6.6. Формализация алгоритмов управления взаимодействующими параллельными процессами в задаче «читатели-писатели»	

6.7. Формализация алгоритмов управления взаимодействующими параллельными процессами в задаче «обедающие философы»	
Глава 7. Проектирование алгоритмов для задач идентификации языков	
7.1. Формализация задач распознавания цепочек-образов	
7.1.1. Формулировка задач распознавания цепочек-образов	
7.1.2. Разработка алгоритмов решения задач распознавания цепочек-образов	
7.2. Разработка алгоритмов решения задач распознавания цепочек-образов для многопроцессорной реализации	
7.2.1. Общие сведения о методике проектирования параллельных алгоритмов	
7.2.2. Пример разработки параллельных алгоритмов для задач распознавания цепочек-образов	