Глава 11

Сети Петри, графы операций и графы переходов

11.1. Сети Петри и графы операций. Основные определения

Сетью Петри (СП) называется двудольный ориентированный граф $N = < P, T, * >$, где $P = \{p_i\}$, $T = \{t_i\}$ — конечные непустые множества вершин, называемые соответственно позициями (места) и переходами; * — отношение между вершинами, соответствующее дугам графа.

Позиции изображаются кружками, а переходы — черточками (полочки). Дуги соединяют кружки с черточками и черточки с кружками, но не однотипные вершины между собой.

Маркировкой сети Петри называется функция Ф, которая каждой позиции ставит в соответствие целое неотрицательное число. Маркировка характеризуется вектором $\Phi = < \Phi(p_1), \ldots, \Phi(p_n) >$, где и — число позиций сети Петри. В графическом изображении маркировке Ф соответствует размещение меток (точки, маркеры, фишки) в позициях сети. При этом число меток в позиции p_i равно $\Phi(p_i)$.

Различные маркировки сети Петри характеризуют состояния соответствующей ей динамической системы, причем динамика изменений состояний моделируется движением меток по позициям. Маркировка сети может изменяться при срабатывании ее переходов.

Если каждая из входных позиций перехода t_j содержит по меньшей мере одну метку, то переход t_j может сработать (возбужден). При срабатывании перехода из каждой его позиции удаляется одна метка, а в каждую выходную позицию добавляется одна метка.

Обычно в сетях Петри считается, что если при одной и той же маркировке возбуждено несколько переходов, то может сработать любой, но только один из них. Это ограничение не является принципиальным и может быть снято.

При применении сетей Петри для целей управления позициям сопоставляются операции (действия), а переходам — условия, при выполнении которых возбужденные переходы срабатывают, активизируя соответствующие операции [31]. При этом попадание меток в позицию ассоциируется с началом операции, а удаление метки — с ее окончанием. При использовании такого предположения считают, что любая операция не может быть повторно начата до ее завершения. Для описания таких процессов могут применяться только безопасные сети Петри, т. е. такие сети, в

которых при любой маркировке в каждой позиции не может быть более одной метки.

Так как при любом течении дискретного процесса должна быть возможность его возобновления, а любая из множества заданных операций должна быть выполнена, то сеть Петри в таких случаях должна быть живой, т. е. она не должна порождать такие маркировки, для которых другие маркировки недостижимы.

Безопасные и живые сети Петри называются правильными. Поэтому в качестве модели дискретных процессов в [31] было предложено использовать правильные сети Петри. При этом отметим, что проверка сети Петри на правильность является весьма трудоемкой [146].

Основное достоинство сетей Петри состоит в возможности отображения в виде одной компоненты взаимодействия нескольких параллельно-последовательных процессов, а их недостаток заключается в том, что они не описывают в явном виде поведение — динамику смены состояний. Сети Петри в некотором смысле аналогичны мостиковым контактным схемам, для которых описание их структуры отличается от описания их поведения.

Сложность анализа поведения сетей Петри состоит в том, что приходится одновременно следить за положением нескольких точек и запоминать эти ситуации. Поведение сети Петри в явном виде описывается с помощью графа достижимых маркировок, который в некотором смысле аналогичен эквивалентной параллельно-последовательной схеме (П-схема), построенной по заданной мостиковой схеме. Основное достоинство П-схем, определившее их широкое применение, состоит в том, что для каждой из них структура и поведение могут быть описаны одной и той же булевой формулой, что позволяет выполнить ее формальные преобразования с целью упрощения структуры без изменения поведения.

На рис. 11.1 приведена правильная сеть Петри, а на рис. 11.2 — ГДМ, соответствующий этой сети.

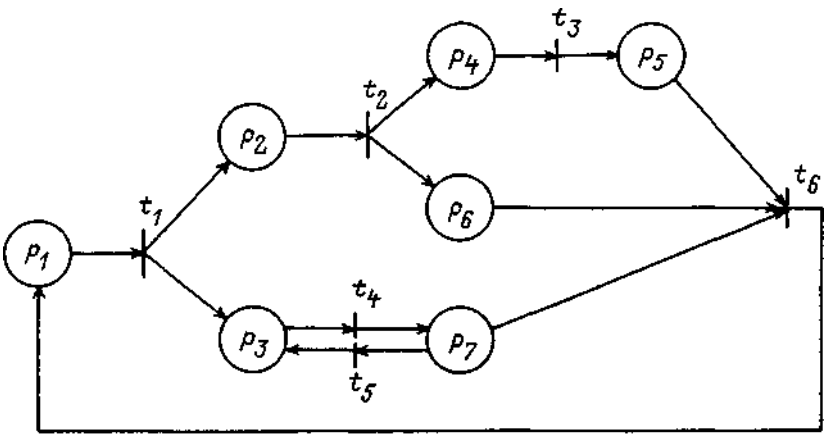

Рис. 11.1

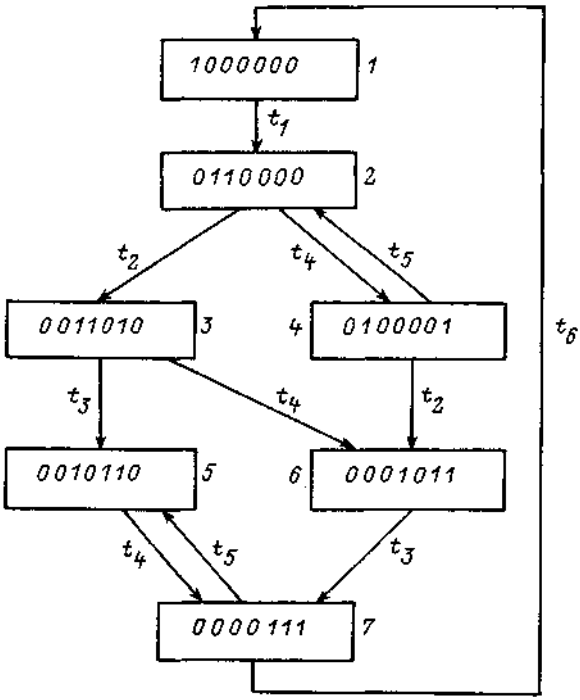

Рис. 11.2

При этом отметим, что ГДМ аналогичны предложенным для анализа асинхронных схем диаграммам переходов (диаграммы Маллера), которые, правда, дополнительно содержат информацию о том, какие компоненты векторов, записанные в каждой вершине, возбуждены в ней.

Несмотря на то что число вершин в ГДМ в общем случае не меньше, чем в соответствующей сети Петри, читать ГДМ существенно проще, так как, во-первых, в нем в каждой вершине в явном виде указаны значения всех компонент вектора, определяющие состояние системы, а во-вторых, он в явном виде описывает динамику изменений состояний.

Правильная сеть Петри называется автоматной (АСП), если все ее переходы имеют не более одной входящей и не более одной выходящей дуги, а в ее начальной маркировке имеется не более одной метки. На рис. 11.3 приведена автоматная сеть Петри, а на рис. 11.4 — соответствующий ГДМ.

Из рассмотрения этих графов следует, что они в некотором смысле изоморфны, и поэтому можно утверждать, что автоматные сети Петри не только задают структуру процесса, но и описывают его динамику. В этом смысле такие сети аналогичны П-схемам.

Если переходы в сети Петри пометить условиями от входных переменных, то такая сеть называется сетью Петри с входами. В такой сети переход срабатывает, если он возбужден и выполняется условие, помечающее этот переход. В таких сетях при изменении значений входных

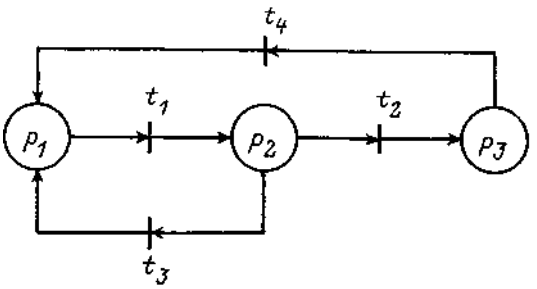

Рис. 11.3

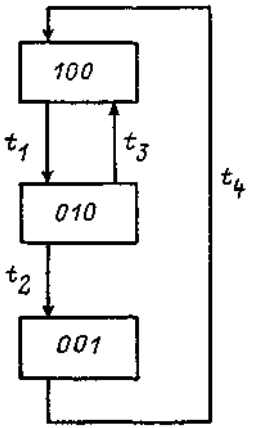

Рис. 11.4

переменных возникает переходный процесс, который завершается установлением равновесия, прежде чем произойдет следующее изменение на входе.

При использовании сетей Петри для целей управления в них должны быть помечены не только переходы, но и позиции. Сети Петри с помеченными переходами и позициями, помеченными значениями выходных переменных, называются сетями Петри с входами и выходами (СПВБ), или графами операций (ГО) [31]. Эти графы могут быть изоморфно реализованы с помощью диаграмм «Графсет» [59].

Для графов операций характерно, что позиции обычно помечаются не всеми значениями переменных, а лишь теми из них, которые изменяются в этой позиции. Это, с одной стороны, делает описания алгоритмов весьма компактными, а с другой — весьма сложными для понимания в случаях, когда значения выходных переменных связаны между собой семантически.

При этом отметим, что для произвольного графа операций, выполняющего в позициях в том числе и вычисления, всегда может быть построена эквивалентная ингибиторная сеть Петри [146], содержащая кроме «прямых» входов в переходы также и «инверсные» входы, и наоборот. Несмотря на то что эти модели равномощны между собой и равномощны с машинами Тьюринга [146], применение графов операций для рассматриваемых целей предпочтительнее, так как позволяет строить компактные и наглядные модели процессов.

Выделим в классе графов операций такой важный подкласс, как автоматные графы операций, которые являются помеченными автоматными сетями Петри. Такие графы операций эквивалентны подклассу графов переходов, а именно ГП автоматов Мура с двоичным кодированием состояний. Однако так как ГП содержат вершины только одного типа и петли, позволяющие отличать устойчивые вершины от неустойчивых, а кроме того, обычно не содержат умолчаний значений выходных переменных, то рекомендуется в большинстве случаев вместо автоматных графов операций применять графы переходов. Возможность использования для описания алгоритмов разных типов ГП с различными видами кодирования делает их применение по сравнению с автоматными графами операций еще более предпочтительным.

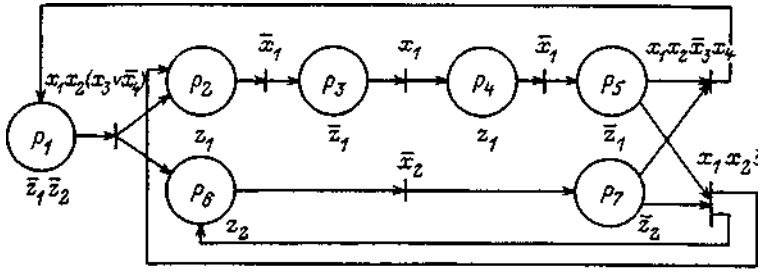

Рис. 11.5

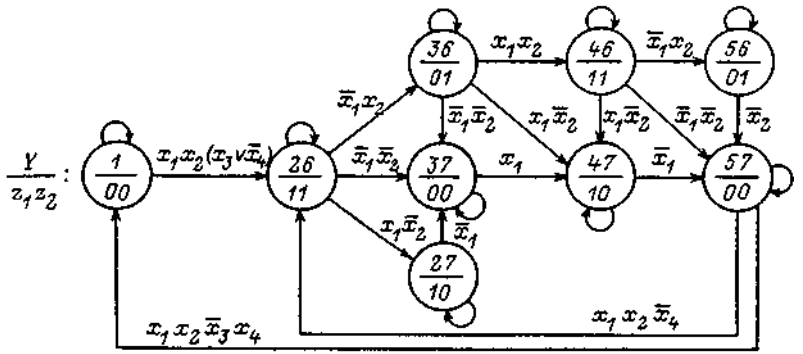

Рис. 11.6

Преимущество изобразительных возможностей неавтоматных графов операций по сравнению с графами переходов проявляется только при необходимости отобразить в одном графе синхронизацию параллельных процессов: если процессы запускаются переходом с одним входом и n выходами, то они должны завершаться переходом с n входами и одним выходом. Это имеет место, например, в версии языка «Графсет», используемой фирмой «Омрон» [236].

Необходимо отметить, что для каждого графа операций может быть построена система взаимосвязанных ГП, реализующая то же поведение (СВГП, так же как и графы операций, равномощны с машинами Тьюринга [163]). Более того, по графу операций всегда может быть построен один граф переходов, который (за счет увеличения числа вершин и дуг и параллелизма по входам и выходам) позволяет реализовать параллельные процессы. Число вершин в этом ГП равно числу состояний в ГО.

Для этого по графу операций строится граф, в котором в вершинах указываются номера позиций, активизируемых при срабатывании переходов. Эти вершины помечаются соответствующими значениями выходных переменных, формируемых в этих позициях, и связываются дугами, определяемыми переходами графа операций. Получаемый граф является ГП автомата Мура, описывающим то же поведение динамической системы, что и граф операций. Эта процедура в терминах [31] выглядит следующим образом: по быстрой сети Петри, являющейся основой ГП,

построить медленную сеть Петри и сопоставить ее маркировкам вершины графа переходов, а ее переходам — дуги этого графа.

На рис. 11.5 приведен граф операций, а на рис. 11.6 — ГП автомата с памятью, построенный по этому графу операций в предположении о возможности срабатывания в нем более одного перехода.

Необходимо отметить, что строить ГП по графу операций целесообразно лишь в том случае, когда значения выходных переменных семантически зависимы. При отсутствии зависимости между выходными переменными в параллельных процессах строить ГП нет необходимости.

Непосредственно по графу операций (рис. 11.5), рассматривая каждую переменную p и z как R-триггер, можно записать систему булевых формул, реализующую этот граф:

$$p_{11} = \overline{(p_2 \,\&\, p_6)} \,\&\, (x_1 \,\&\, x_2 \,\&\, \bar{x}_3 \,\&\, x_4 \,\&\, p_5 \,\&\, p_7 \vee p_1);$$
$$p_{21} = \bar{p}_3 \,\&\, (x_1 \,\&\, x_2 \,\&\, (x_3 \vee \bar{x}_4) \,\&\, p_1 \vee x_1 \,\&\, x_2 \,\&\, \bar{x}_4 \,\&\, p_5 \,\&\, p_7 \vee p_2);$$
$$p_{31} = \bar{p}_4 \,\&\, (\bar{x}_1 \,\&\, p_2 \vee p_3);$$
$$p_{41} = \bar{p}_5 \,\&\, (x_1 \,\&\, p_3 \vee p_4);$$
$$p_{51} = (p_1 \vee p_2 \,\&\, p_6) \,\&\, (\bar{x}_1 \,\&\, p_4 \vee p_5);$$
$$p_{61} = \bar{p}_7 \,\&\, (x_1 \,\&\, x_2 \,\&\, (x_3 \vee \bar{x}_4) \,\&\, p_1 \vee x_1 \,\&\, x_2 \,\&\, \bar{x}_4 \,\&\, p_5 \,\&\, p_7 \vee p_6);$$
$$p_7 = (p_1 \vee p_2 \,\&\, p_6) \,\&\, (\bar{x}_2 \,\&\, p_6 \vee p_7);$$
$$z_1 = (p_1 \vee p_3 \vee p_5) \,\&\, (p_2 \vee p_4 \vee z_1);$$
$$z_2 = (p_1 \vee p_7) \,\&\, (p_6 \vee z_2);$$
$$p_1 = p_{11}; \quad p_2 = p_{21}; \quad p_3 = p_{31};$$
$$p_4 = p_{41}; \quad p_5 = p_{51}; \quad p_6 = p_{61}.$$

В начальной позиции $p_1 = 1$; $p_2 = p_3 = p_4 = p_5 = p_6 = p_7 = 0$.

Использование переобозначений переменных снимает проблему [31] пропуска позиций, для которых пометки входных и выходных переходов не ортогональны.

11.2. Новые методы реализации графов операций

Рассмотренный в предыдущем разделе метод строит систему булевых формул непосредственно по графу операций. При этом каждая позиция и выходная переменная графа операций моделируются R-триггером. Это приводит к ограничению на класс реализуемых графов операций, т. е. не могут быть реализованы такие графы с контурами, которые содержат менее трех позиций, так как в противном случае условия установки и сброса триггеров аннулируются. Это ограничение может быть снято при использовании других методов построения программных моделей.

Реализуем граф операций на рис. 11.5 программой, применяя операторы условного и безусловного переходов, а также операторы присваивания:

```
  p1 = 1; p2=p3=p4=p5=p6=p7=z1=z2=0;
M:if(p1&x1&x2&(x3∨x̄4))  p11=0   p21=p61=1;
  if(p2&x̄1)             p21=0;  p31=1;
```

```
if(p3&x1)                  p31=0;      p41=1;
if(p4&x̄1)                  p41=0;      p51=1;
if(p5&p7&x1&x2&x̄3&x4)      p51=p71=0;  p11=1;
if(p5&p7&x1&x2&x̄4)         p51=p71=0;  p21=p61=1;
if(p6&x̄2)                  p61=0;      p71=1;
z1=p21∨p41; z2=p61;
p1=p11; p2=p21; p3=p31; p4=p41;
p5=p51; p6=p61; p7=p71;
goto M.
```

В этой программе используются переобозначения переменных, которые в данном случае не требуются ввиду ортогональности пометок переходов до и после каждой позиции, но решают проблему пропуска позиций в случае, если указанные пометки не ортогональны.

Другой метод реализации графа операций состоит в построении системы булевых формул на основе записи условий установки единичных значений позиций и сохранения этих значений по аналогии с тем, как это делается для графов переходов при двоичном кодировании состояний (при этом предполагается, что позиции графа операций (рис. 11.5) как бы содержат петли):

$$p_{11} = x_1 \,\&\, x_2 \,\&\, \bar{x}_3 \,\&\, x_4 \,\&\, p_5 \,\&\, p_7 \vee \overline{(x_1 \,\&\, x_2 \,\&\, (x_3 \vee \bar{x}_4))} \,\&\, p_1;$$
$$p_{21} = x_1 \,\&\, x_2 \,\&\, (x_3 \vee \bar{x}_4) \,\&\, p_1 \vee x_1 \,\&\, x_2 \,\&\, \bar{x}_4 \,\&\, p_5 \,\&\, p_7 \vee x_1 \,\&\, p_2;$$
$$p_{31} = \bar{x}_1 \,\&\, (p_2 \vee p_3);$$
$$p_{41} = x_1 \,\&\, (p_3 \vee p_4);$$
$$p_{51} = \bar{x}_1 \,\&\, p_4 \vee \overline{(x_1 \,\&\, x_2 \,\&\, \bar{x}_3 \,\&\, x_4)} \,\&\, p_5 \vee x_1 \,\&\, x_2 \,\&\, \bar{x}_3 \,\&\, x_4 \,\&\, p_5 \,\&\, \bar{p}_7;$$
$$p_{61} = x_1 \,\&\, x_2 \,\&\, \overline{(x_3 \vee \bar{x}_4)} \,\&\, p_1 \vee x_1 \,\&\, x_2 \,\&\, \bar{x}_4 \,\&\, p_5 \,\&\, p_7 \vee x_2 \,\&\, p_6;$$
$$p_7 = \bar{x}_2 \,\&\, p_6 \vee \overline{(x_1 \,\&\, x_2 \,\&\, \bar{x}_4)} \,\&\, p_7 \vee x_1 \,\&\, x_2 \,\&\, \bar{x}_4 \,\&\, p_7 \,\&\, \bar{p}_5;$$
$$z_1 = p_{21} \vee p_{41};\quad z_2 = p_{61};$$
$$p_1 = p_{11};\quad p_2 = p_{21};\quad p_3 = p_{31};$$
$$p_4 = p_{41};\quad p_5 = p_{51};\quad p_6 = p_{61}.$$

В начальной позиции $p_1 = 1$; $p_2 = p_3 = p_4 = p_5 = p_6 = p_7 = 0$.

Эта система формул при необходимости может быть минимизирована.

11.3. Области использования моделей описания параллельных процессов

В рамках предлагаемой технологии граф переходов в большинстве случаев строится таким образом, что число вершин в нем определяется числомприменяемыхвалгоритмекомбинаций всех выходныхпеременных *m*. При этом некоторые из этих комбинаций могут использоваться многократно. Граф переходов задает последовательное по состояниям и параллельное по выходам поведение системы.

Если число вершин *s* в графе переходов существенно меньше, чем два в степени *m*, а число дуг существенно меньше, чем их количество в полном

графе, то процесс алгоритмизации с помощью одного графа переходов эффективен.

Если имеют место «чистый» параллелизм (общие переменные отсутствуют) и семантическая независимость процессов, то целесообразно применять систему ГП (СГП), так как единый граф переходов в этом случае весьма громоздок.

Если параллельные процессы не имеют синхронизации между собой, то такой алгоритм наиболее целесообразно описать системой взаимосвязанных ГП.

Если параллельные процессы должны быть синхронизированы между собой, в том числе и после выполнения отдельных этапов, то вместо графа операций целесообразно использовать граф переходов, так как зависимость процессов не позволяет сильно увеличиться числу состояний автомата.

Если параллельные процессы, синхронизируемые только в их конце и начале, зависимы между собой семантически и если в качестве описания алгоритма использовать граф операций или СВГП, то для их анализа целесообразно строить весьма громоздкий граф переходов. Вопрос о том, по какой из этих моделей проводить программирование, зависит от требований, предъявляемых к программе.

Если независимые семантически параллельные процессы должны быть синхронизированы между собой лишь в их начале и конце, то в этом случае целесообразно применять граф операций или СВГП.

Если имеется несколько групп независимых процессов, каждая из которых описывается графом операций, то говорят, что используется система графов операций (СГО).

Для отображения иерархии процессов могут применяться системы взаимосвязанных ГП (вызываемые (гл. 8) и вложенные (разд. 12.4) графы переходов) и системы взаимосвязанных ГО (СВГО), построенных на базе сложных сетей Петри, в которых используются вершины-дублеры, заменяющие процессы, описываемые другими графами операций.

При этом отметим, что (в отличие от [31]) для СВГО нет необходимости обязательно выполнять замену дублеров соответствующими процессами с целью получения одного графа операций, а можно реализовать эту систему непосредственно, как это делается для СВГП.

Для реализации сложных алгоритмов логического управления бывает целесообразным совместное использование ГП, СГП, СВГП, ГО, СГО и СВГО.

11.4. Примеры реализации алгоритмов логического управления с параллелизмом

11.4.1. Алгоритмы с параллельными процессами и синхронизацией этапов

Рассмотрим в качестве примера алгоритм, задаваемый графом операций, в котором кроме синхронизации окончания параллельных процессов требуется также и синхронизация этапов (рис. 11.7).

Этот алгоритм может быть реализован также двумя системами взаимосвязанных графов переходов — иерархической и децентрализованной. Децентрализованная (распределенная) реализация алгоритма с помощью СВГП, состоящей из двух компонент, приведена на рис. 11.8.

Эта система может быть корректно реализована при использовании, например, конструкций `switch` лишь при введении дополнительной переменной Y_{II}, значения которой вместо переменной Y_I устанавливаются в конструкции `switch` (Y_I), и применении переобозначения переменных $Y_I = Y_{II}$, располагаемого после конструкции `switch` (Y_I).

Каждая из приведенных реализаций обладает тем недостатком, что при их анализе (чтении) приходится следить одновременно за движением нескольких точек, перемещающихся в одной или нескольких компонентах.

Если по исходному графу операций построить граф достижимых маркировок и дополнить его значениями всех входных переменных в каждой маркировке, то формируется ГП автомата Мура (рис. 11.9).

Реализация параллельных процессов этим автоматом, последовательным по состояниям, обеспечивается за счет его параллелизма по выходам.

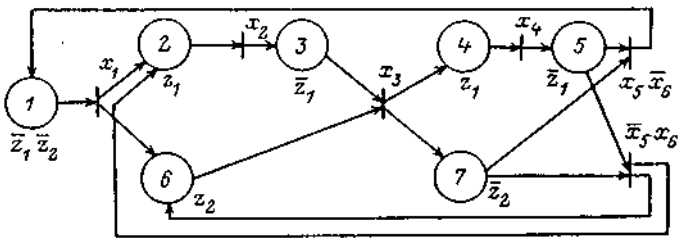

Рис. 11.7

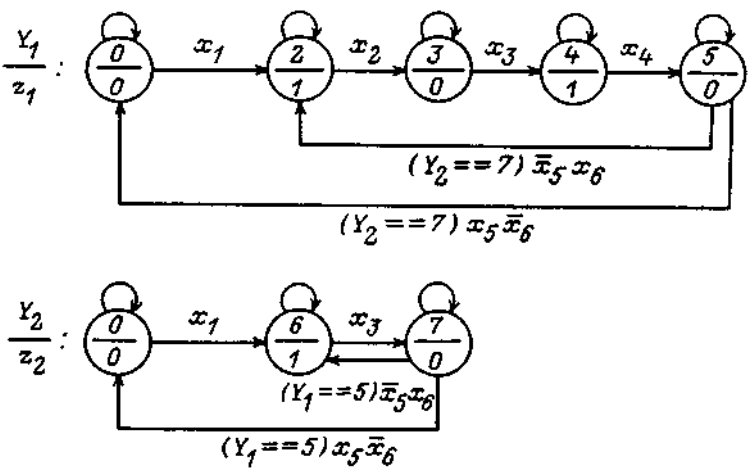

Рис. 11.8

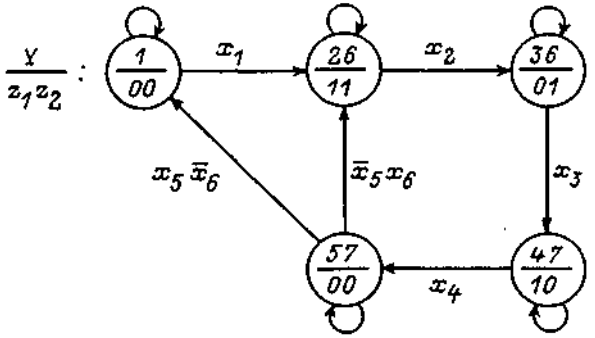

Рис. 11.9

Число состояний этого автомата меньше не только суммарного числа состояний в СВГП, но и числа позиций в графе операций. Получившийся ГП весьма прост, так как в рассматриваемом алгоритме процессы взаимосвязаны с помощью синхронизации некоторых этапов.

11.4.2. Событийные алгоритмы управления двумя клапанами

Рассмотрим на примере алгоритма управления двумя клапанами без памяти (клапаны Кл1 и Кл2) вопрос о выборе вида графовой модели для его описания.

Алгоритм 1. При импульсном нажатии кнопки ($x_1 = 1$) начинают открываться оба клапана ($z_1 = z_2 = 1$). После того как они откроются (x_2 & $x_3 = 1$; $x_2 = 1$ — открылся первый клапан; $x_3 = 1$ — открылся второй клапан), управляющие сигналы снимаются ($z_1 = z_2 = 0$).

Этот алгоритм, описывающий параллельные процессы, может быть реализован одним ГП (рис. 11.10), одним автоматным (рис. 11.11) и одним неавтоматным графом операций (рис. 11.12). При этом отметим, что в первых двух графах в каждой вершине задаются значения всех выходных переменных, а в третьем графе значения всех переменных задаются лишь в одной вершине. Реализация алгоритма с помощью ГП в этом случае наиболее целесообразна.

Алгоритм 2. При $x_1 = 1$ начинают открываться оба клапана ($z_1 = z_2 = 1$). После открытия любого из них (x_2 - 1 или x_3 - 1) соответствующий управляющий сигнал должен быть сброшен ($z_1 = 0$ или $z_2 = 0$).

Этот алгоритм не может быть корректно реализован одним графом операций (рис. 11.13), так как в нем могут формироваться противоположные значения одной и той же выходной переменной. Например, если в этом графе активными являются вершины p_1 и p_2, то после $x_2 = 1$ при $x_1 = x_3 = 0$ активными станут вершины p_0 и p_2, в первой из которых — $!z_2$, а во второй — z_2 .Также некорректен и граф операций на рис. 11.14.

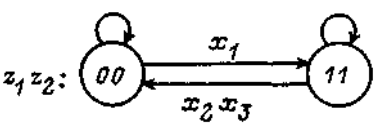

Рис. 11.10

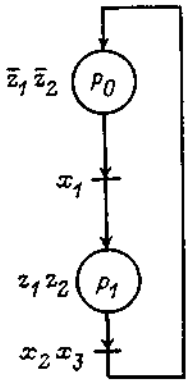

Рис. 11.11

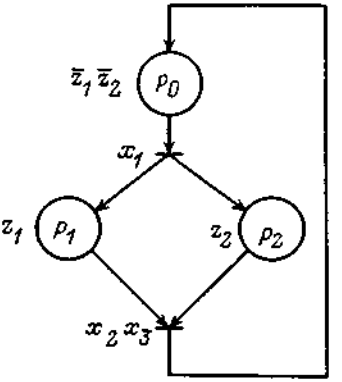

Рис. 11.12

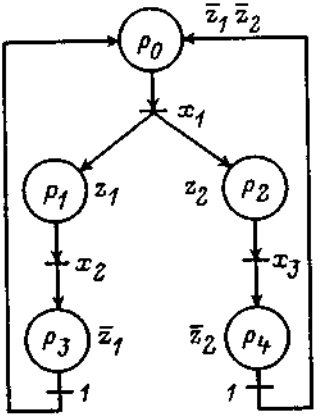

Рис. 11.13

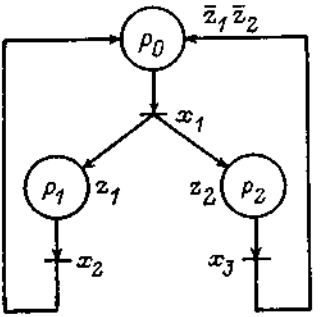

Рис. 11.14

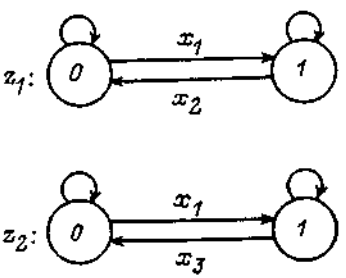

Рис. 11.15

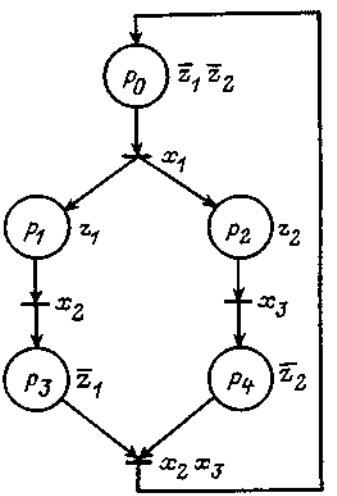

Рис. 11.16

Графы операций на рис. 11.13 и 11.14 иллюстрируют тот факт, что в неавтоматном графе операций параллельные процессы должны быть синхронизированы, чего не требуется в данном алгоритме.

Заданный алгоритм корректно реализуется СВГП (рис. 11.15).

Алгоритм 3. При $x_1 = 1$ начинают открываться оба клапана ($z_1 = z_2 = 1$). После открытия любого из них (x_2 - 1 или $x_3 = 1$) соответствующий управляющий сигнал должен быть сброшен ($z_1 = 0$ или $z_2 = 0$). После открытия обоих клапанов (x_2 & $x_3 = 1$ — условия синхронизации окончания параллельных процессов) осуществляется возврат в исходную позицию.

Этот алгоритм может быть реализован одним неавтоматным графом операций (рис. 11.16), децентрализованной СВГП из двух компонент (рис. 11.17) или иерархической СВГП из трех компонент (рис. 11.18). Для этого алгоритма может быть также построена СВГО (рис. 11.19).

. Система взаимосвязанных ГО может быть изоморфно реализована на языке «Графсет».

Алгоритм 4. При $x_1 = 1$ начинает открываться ($z_1 = 1$) первый клапан, а когда он откроется ($x_2 = 1$), управляющий сигнал снимается ($z_1 = 0$). При $x_1 = 0$, $x_4 = 1$ начинает открываться ($z_2 = 1$) второй клапан, а когда он откроется ($x_3 = 1$), его управляющий сигнал снимается ($z_2 = 0$).

Этот алгоритм реализуется СВГП, состоящей из двух компонент (рис. 11.20). В заданном алгоритме и его реализации процессы управления клапанами описаны независимо. Если они и на самом деле семантически независимы, то выполнять анализ их совместной работы нет необходимости. Однако если семантическая зависимость имеется или отсутствует уверенность в их независимости, то требуется выполнить анализ поведения СВГП, состоящий в построении по системе булевых формул, описывающей СВГП, одного графа переходов, задающего в явном виде поведение системы в целом (рис. 11.21).

Отметим, что связность СВГП по входам исключает лишь один переход из полного графа. Таким образом, система графов переходов реализует заданный алгоритм с учетом умолчаний о совместном поведе-

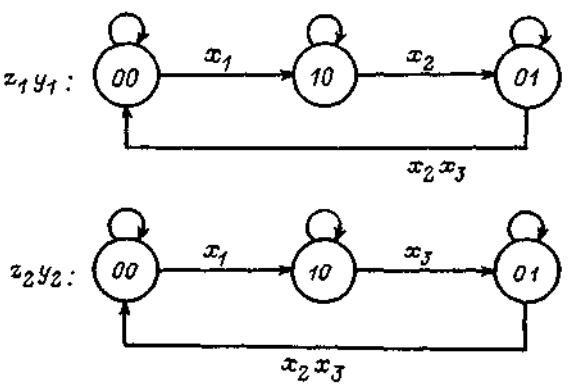

Рис. 11.17

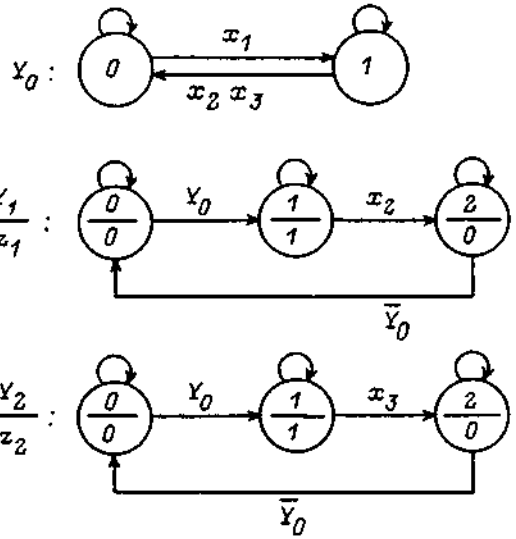

Рис. 11.18

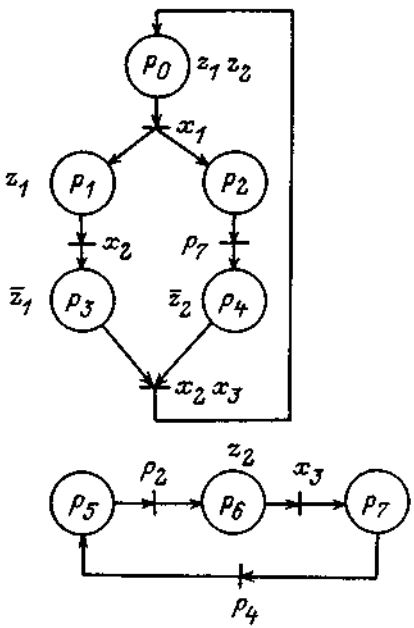

Рис. 11.19

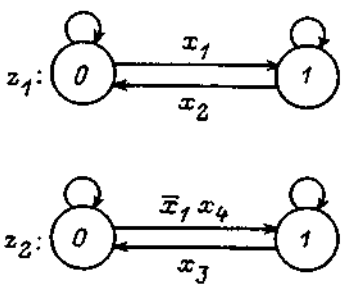

Рис. 11.20

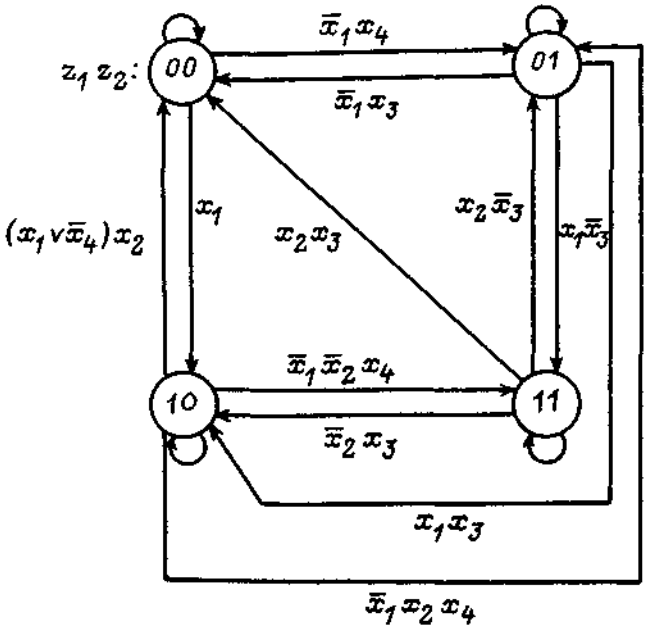

Рис. 11.21

нии клапанов. Без умолчаний словесное описание в этом случае становится чрезвычайно громоздким.

Алгоритм 5. В процессе управления каждым из клапанов по алгоритму *4* отсутствует возможность управлять другим клапаном.

Этот алгоритм может быть реализован одним ГП (рис. 11.22) или одним автоматным ГО (рис. 11.23). В автоматных графах операций процессы могут не синхронизироваться. Применение ГП в этом случае более целесообразно, так как его вершины задают состояния автомата в целом.

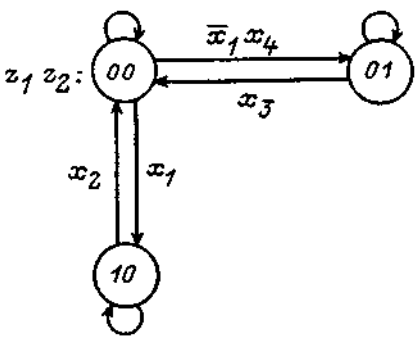

Рис. 11.22

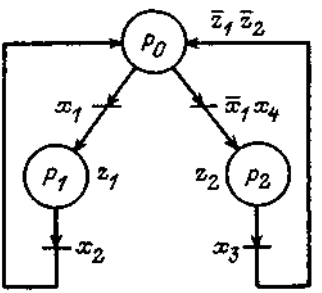

Рис. 11.23

11.4.3. Временные алгоритмы управления двумя клапанами

Покажем на примере управления двумя клапанами без памяти (клапаны Кл1 и Кл2), время срабатывания которых d_{1k} и d_{2k} неизвестно, но не превышает трех секунд, как «незначительное» изменение алгоритма влияет на выбор модели для его описания.

Алгоритм 6. Предположим, что первоначально требуется реализовать следующий алгоритм. При $x_1 = 1$ подаются сигналы на открытие первого клапана Кл1 ($z_1 = 1$) и первого элемента задержки ФЭ31 ($t_1 = 1$), время срабатывания которого равно $d_1 = 3$ с. По истечении этого времени ($T_1 = 1$) и факту открытия этого клапана ($x_2 = 1$) с последнего снимается управляющий сигнал ($z_1 = 0$) и запускается второй элемент задержки ФЭ32 ($t_2 = 1$), время срабатывания которого равно $d_2 = 2$ с. После срабатывания последнего ($T_2 = 1$) подается сигнал на открытие второго клапана Кл2 ($z_2 = 1$), после открытия которого ($x_3 = 1$) осуществляется сброс всех еще не сброшенных к моменту наступления этого события сигналов.

Этот алгоритм имеет причинно-следственный характер и поэтому может быть реализован одним автоматом (рис. 11.24), последовательным по состояниям и параллельным по выходам. При этом единичные значения переменных z_1 и t_1, должны существовать параллельно некоторое априори неизвестное время.

Алгоритм 7. Изменим заданный алгоритм, исключив зависимость срабатывания второго элемента задержки от факта срабатывания первого элемента. При $x_1 = 1$ подаются сигналы на срабатывание клапана Кл1 ($z_1 = 1$) и первого элемента задержки ФЭ31 (f, = 1). После срабатывания этого элемента задержки (T_1 - 1) снимается управляющий сигнал с первого клапана ($z_1 = 0$). После открытия первого клапана ($x_2 = 1$) запускается второй элемент задержки ФЭ32 ($t_2 = 1$), а после его срабатывания ($T_2 = 1$) начинает открываться клапан Кл2 ($z_2 = 1$). После открытия второго клапана ($x_3 = 1$) осуществляется сброс всех еще не сброшенных сигналов.

В этом алгоритме процесс управления вторым клапаном не зависит от факта срабатывания первого элемента задержки ФЭ31 (переменной T_1) и поэтому реализация одним автоматом становится неестественной и весьма сложной. Поэтому используем для описания алгоритма систему взаимо-

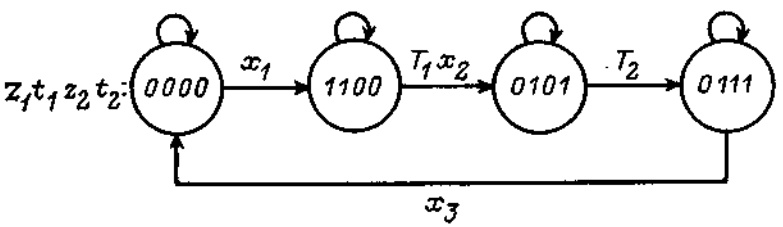

Рис. 11.24

связанных ГП из двух параллельно функционирующих автоматов, в первом из которых протекают параллельные процессы — открытие клапана Кл1 и запуск первого элемента задержки ФЭ31 (рис. 11.25). Поведение первого автомата зависит от значения переменной $d = d_{1k} + d_{2k} = d_2$. Если $d < 3$, то с возбужденного элемента ФЭ31 до его срабатывания (при $x_3 = 1$) будет снят управляющий-сигнал ($t_1 = 0$). Если $d >= 3$, то возбужденный элемент ФЭ31 сработает (при $x_3 = 0$).

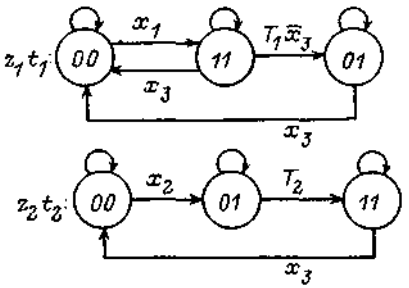

Рис. 11.25

Приведенный алгоритм может быть назван неполумодулярным, или неустойчивым. Это определение базируется на следующих определениях.

Сеть относится к классу полумодулярных, если в ней каждый возбужденный элемент обязательно срабатывает. Это обеспечивает независимость поведения сети от величин задержек элементов [13].

Сеть Петри называется устойчивой относительно некоторой маркировки, если в ней выполнение входных условий для некоторого перехода обязательно приводит к его срабатыванию [13].

Неустойчивая сеть Петри является также и неживой и поэтому не может быть применена для построения графа операций. Таким образом, показано, что заданный алгоритм реализуется СВГП, но не описывается одним графом операций.

Алгоритм 8. Изменим реализуемый алгоритм с параллельными процессами таким образом, чтобы он стал устойчивым, т. е. реализовался одним графом операций.

При $x_1 = 1$ подаются сигналы на срабатывание клапана Кл1 ($z_1 = 1$) и первого элемента задержки ФЭ31 ($t_1 = 1$). После срабатывания этого элемента задержки ($T_1 = 1$) снимается управляющий сигнал с первого клапана ($z_1 = 0$). После открытия первого клапана (x_2 - 1) запускается второй элемент задержки ФЭ32 ($t_2 = 1$), а после его срабатывания ($T_2 = 1$) начинает открываться клапан Кл2 ($z_2 = 1$)- После закрытия клапана Кл2 (x_3 - 1) осуществляется возврат в исходную позицию.

Этот алгоритм, так же как и предыдущий, нецелесообразно реализовывать одним автоматом (одним ГП), в то время как он эффективно описывается с помощью одного графа операций (рис. 11.26). Реализация этого графа на языке «Графсет», описываемом в гл. 12, приведена на рис. 11.27. Отметим, что в этом алгоритме синхронизация процессов не позволяет в отличие от предыдущего случая ни при каких значениях d_{1k} и d_{2k} завершить его менее чем за три секунды.

Если алгоритм *7*, реализованный СВГП, не мог быть описан одним графом операций, то алгоритм *8* может быть выполнен не только одним графом операций, но и СВГП (рис. 11.28). Сравнение графов на рис. 11.26 и 11.28 показывает, что в первом случае используется декомпозиция по процессам (режимам управления), а во втором — по объектам

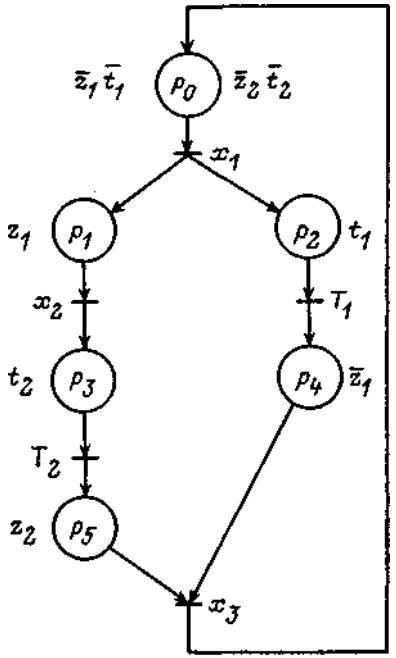

Рис. 11.26

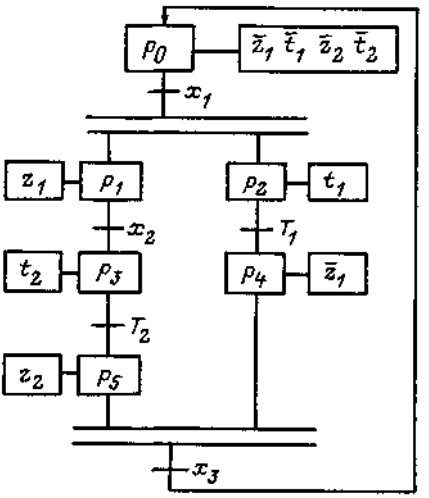

Рис. 11.27

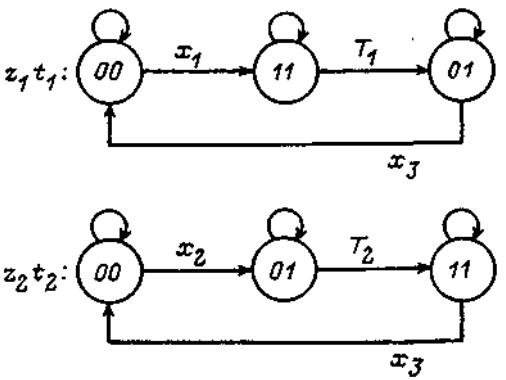

Рис. 11.28

управления. Во втором случае, кроме того, обеспечивается сокращение объема памяти, а при переходе к многозначному кодированию возможен изоморфизм между описанием и текстом программы.

11.5. Особенности реализации логико-вычислительных алгоритмов

До сих пор в настоящей главе рассматривались вопросы описания задач логического управления, при котором выходные переменные принимали два значения — 0 и 1. При реализации логико-вычислительных алгоритмов наряду с процессами, определяемыми сменой состояний (последовательной или параллельной) в отдельных вершинах (позициях, ситуациях, этапах и состояниях), могут протекать также последовательные или параллельные процесс

Процессы первого типа можно условно считать происходящими по вертикали, а процессы второго типа — происходящими по горизонтали. При этом единица в вершине графа рассматривается либо как факт запуска, либо как факт наличия процесса, а ноль — его сброс или отсутствие.

Простейшим примером процесса второго типа является реализация в вершине графа переходов функционального элемента задержки, например с помощью переменной, принимающей единичное значение один раз в секунду, — импульсной переменной ПЛК (разд. 14.3.2). При этом ФЭЗ может быть описан, например, с помощью ГСА, в котором понятие «состояние» не используется, а алгоритм в целом реализуется совокупностью ГП и ГСА, которые объединяются в единую компоненту на программном уровне.

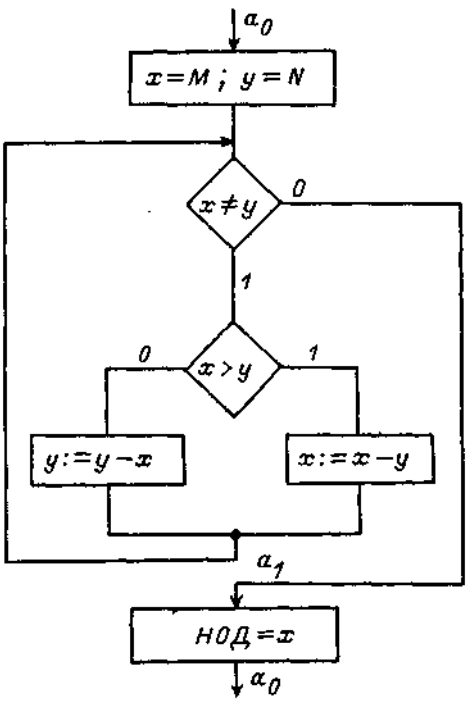

Рис. 11.29

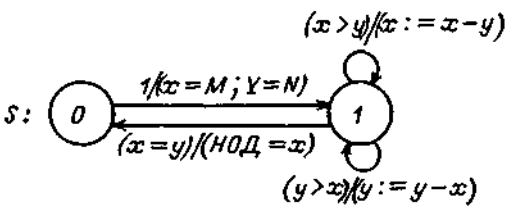

Рис. 11.30

В этом примере единица в вершине указывает на необходимость размещения процесса запуска ФЭЗ в соответствующей вершине графа переходов, а ноль — его сброса. Если этот граф реализовать СБФ, то единичное значение временной переменной указывает на необходимость запуска подпрограммы, обеспечивающей реализацию ФЭЗ, а нулевое значение этой переменной — на возврат подпрограммы в исходное состояние.

Аналогично могут быть реализованы и другие логико-вычислительные процессы. Если исходное описание выполнено с помощью ГСА, то его программная реализация может выполняться на основе графа переходов, эквивалентного заданной ГСА. На рис. 11.29 приведена ГСА, реализующая алгоритм определения наибольшего общего делителя двух целых положительных чисел (алгоритм Евклида), а на рис. 11.30 — эквивалентный, но более компактный ГП автомата Мили, изоморфно реализуемый конструкцией `switch`.