УДК 681.5.62–5+681.5.681.3

МЕТОД ФОРМИРОВАНИЯ СУБОПТИМАЛЬНЫХ РАЗБИЕНИЙ ПАРАЛЛЕЛЬНЫХ УПРАВЛЯЮЩИХ АЛГОРИТМОВ

Э.И. Ватутин
Курский государственный технический университет
Россия, 305040, Курск, ул. 50 лет Октября, 94
E-mail: evatutin@mail.kurskline.ru

И.В. Зотов
Курский государственный технический университет
Россия, 305040, Курск, ул. 50 лет Октября, 94
E-mail: zotov@kursknet.ru

Ключевые слова: параллельный алгоритм логического управления, разбиение, оптимизация
Key words: parallel logic control algorithm, separation, optimization

В работе содержится дальнейшее развитие ранее разработанного авторами метода формирования субоптимальных разбиений параллельных алгоритмов логического управления на совокупность параллельно работающих последовательных подалгоритмов (блоков), реализуемых параллельной системой логического управления. Подробно рассматриваются все этапы метода. Изложение сопровождается примерами. Дается краткое описание инструментальной программной системы, реализующей изложенный метод.

A METHOD FOR THE CONSTRUCTION OF SUBOPTIMAL SEPARATIONS OF PARALLEL CONTROL ALGORITHMS / E.I. Vatutin (Kursk State Technical University, 94 50 Let Oktyabrya, Kursk 305040, Russia, E-mail: evatutin@mail.kurskline.ru), I.V. Zotov (Kursk State Technical University, 94 50 Let Oktyabrya, Kursk 305040, Russia, E-mail: zotov@kursknet.ru). The work addresses a refinement of the method for suboptimal separation of parallel logic control algorithms into a set of concurrent sequential subalgorithms (modules) to be implemented by a parallel logic control system, the method presented earlier by the authors. All the stages of the method are detailed. The consideration is accompanied with examples. A development software implementing the method is briefly described.

1. Введение

Построение систем логического управления (СЛУ) остается одной из важных проблем современной науки, техники и технологии в силу продолжающегося активного внедрения дискретных процессов управления в различные сферы деятельности человека. Проблемы анализа и синтеза СЛУ на протяжении нескольких десятков лет остаются в центре внимания отечественных и зарубежных ученых. Их решению посвящено значительное число теоретических исследований, получен ряд важных для науки и практики результатов.

Анализ показывает, что традиционные подходы к построению СЛУ в условиях быстро растущей сложности современных объектов управления, распараллеливания задач управления, появления особых требований к гибкости СЛУ (таких, как, например, оперативная настраиваемость и наращиваемость) сталкиваются с серьезными трудностями. Однако успехи современной микроэлектронной технологии позволяют по-новому подойти к этой задаче с учетом возможности реализации полнофункциональной СЛУ в масштабах нескольких специализированных СБИС. Подобные системы из СБИС (мультимикроконтроллеры) обладают быстродействием СЛУ с «жесткой» логикой и в то же время могут быть оперативно настроены (или перенастроены) на требуемый алгоритм управления подобно микропроцессорам. Они способны эффективно решать задачи параллельного логического управления и выполнять управляющие алгоритмы теоретически неограниченной сложности за счет их разбиения на параллельно работающие последовательные подалгоритмы.

В работе рассматривается задача выбора подобных разбиений с учетом заданных технологических и структурных ограничений мультимикроконтроллера. Для ее решения предлагается дальнейшее развитие ранее разработанного авторами параллельно-последовательного метода формирования субоптимальных разбиений, основанного на серии эквивалентных структурных преобразований исходного управляющего алгоритма [1,2].

2. Постановка задачи

Задача формирования (выбора) оптимального разбиения заключается в оптимальном представлении исходного параллельного алгоритма логического управления в виде множества взаимосвязанных последовательных подалгоритмов (блоков) ограниченной сложности. Каждый такой подалгоритм в общем случае представляет собой совокупность нескольких несвязанных между собой фрагментов (ветвей) исходного алгоритма, а все множество подалгоритмов в совокупности с соответствующими связями дает результат решения задачи – сеть взаимосвязанных подалгоритмов.

К получаемой сети подалгоритмов предъявляется ряд требований, к наиболее значимым из которых относятся:

- минимальное число подалгоритмов;
- ограниченная сложность подалгоритмов (выраженная максимально допустимым числом вершин, микроопераций и логических условий);
- минимальная сложность сети межблочных связей;
- минимальная интенсивность межблочных взаимодействий.

Формализованное представление задачи выбора разбиения имеет следующий вид.

Пусть H – число подалгоритмов разбиения; $W_{\max}$ – емкость участка памяти микропрограмм микроконтроллера, выделяемого для размещения микрокоманд подалгоритма (без учета дополнительных команд, обеспечивающих межмикроконтроллерное взаимодействие); $n_{\text{ЛУ}}$ – число выводов на корпусе СБИС микроконтроллера, используемых для приема сигналов логических условий от объекта управления. Требуется получить разбиение множества вершин A^0 исходного

управляющего алгоритма $Sep\left(A^0\right)=\left\{A_1, A_2, ..., A_H\right\}$, удовлетворяющее следующим условиям:

$$
\begin{aligned}
&\bigcup_{i=1}^{H} A_i = A^0, \quad A_i \neq \varnothing, \quad A_i \bigcap A_j = \varnothing, \quad i, j = \overline{1,H}, \quad i \neq j; \\
&\neg\left(a_i \omega a_j\right) \forall a_i, a_j \in A_k, \quad i \neq j, \quad k = \overline{1,H}; \\
&W\left(A_i\right) \leq W_{\max}, \quad \left|X\left(A_i\right)\right| \leq n_{\text{ЛУ}}, \quad i = \overline{1,H},
\end{aligned}
\tag{1}
$$

где ω – символ отношения параллельности вершин, $W\left(A_i\right)=\sum_{a_j \in A_i} W\left(a_j\right)$ – суммарный вес вершин в составе *i*-го подалгоритма, $X\left(A_i\right)=\bigcup_{a_j \in A_i} X\left(a_j\right)$ – множество логических условий, входящих в вершины *i*-го подалгоритма, такое что

$$
\begin{aligned}
&H \to \min; \\
&Z_1 = \sum_{i=1}^{H-1} \sum_{j=i+1}^{H} \alpha\left(A_i, A_j\right) \to \min; \\
&Z_2 = \sum_{i=1}^{H-1} \sum_{j=i+1}^{H} \delta\left(A_i, A_j\right) \to \min; \\
&Z_3 = \max_{i,j=\overline{1,H},\ i \neq j} \left\{E\left(A_i\right) - E\left(A_j\right)\right\} \to \min,
\end{aligned}
$$

где Z_1 – сложность сети межблочных связей, порождаемая разбиением $Sep\left(A^0\right)$; $\alpha\left(A_i, A_j\right)$ – коэффициент связи подалгоритмов (он равен 1, если подалгоритмы связаны, и 0 в противном случае); Z_2 – суммарное число межблочных взаимодействий; $\delta\left(A_i, A_j\right)$ – интенсивность взаимодействия подалгоритмов; Z_3 – дискриминант алгоритмов по сложности; $E\left(A_i\right)$ – функция, характеризующая сложность подалгоритма.

3. Состав и представление исходных данных

Исходный параллельный алгоритм $G_0=\left\langle A^0, U^0\right\rangle$, для которого требуется найти субоптимальное разбиение $Sep\left(A^0\right)$, представляется в виде параллельной граф-схемы (ПГС). Вершинам $a_i \in A^0$ ПГС G_0 сопоставляются следующие параметры:

- тип вершины;
- множество логических условий $X\left(a_i\right)$;
- множество микроопераций $Y\left(a_i\right)$;
- вес (число микрокоманд) $W\left(a_i\right)$.

В составе ПГС различаются типы вершин, представленные на рис.1.

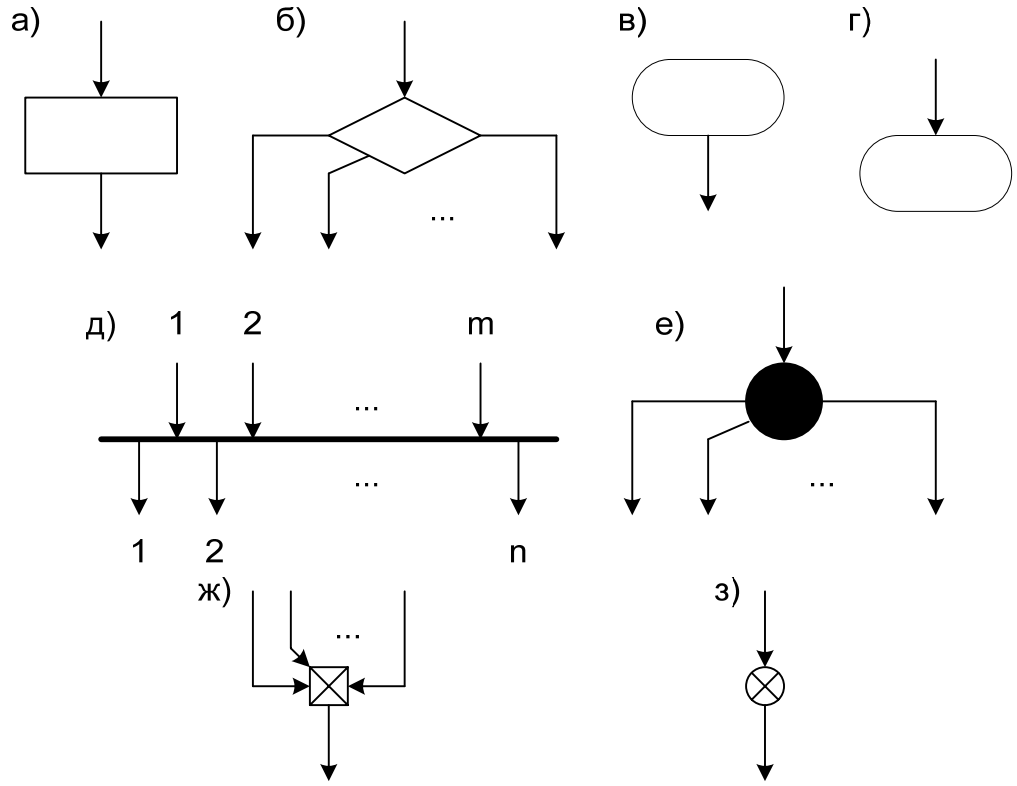

Рис. 1. Типы вершин параллельных алгоритмов логического управления:
(а) – операторная вершина;
(б) – условная вершина;
(в) – начальная вершина;
(г) – конечная вершина;
(д) – вершина распараллеливания/синхронизации;
(е) – вершины выбора направления перехода (зависимого условия);
(ж) – вершина объединения альтернативных дуг;
(з) – фиктивная операторная вершина.

Фиктивные операторные вершины используются с целью включения в «пустые» ветви (дуги). Вершины объединения альтернативных дуг, также являющиеся фиктивными, предназначены для идентификации завершения альтернативных ветвлений и используются при построении матрицы отношений и системы *R*-выражений. Вершины зависимых условий служат для реализации в алгоритмах управления параллельных альтернативных ветвлений и параллельных циклов.

В зависимости от типа вершина может не обладать теми или иными признаками. Так, например, для условных вершин $Y = \varnothing$, для вершин синхронизации $X = \varnothing$, $Y = \varnothing$, $W = 0$, и т.д.

Дугам $u_i = \left(a_j, a_k\right) \in U^0$ сопоставляются следующие параметры:

- вероятность реализации дуги β;
- среднее число передач абстрактной активности (интенсивность) δ.

Предполагается, что параметры дуг для исходного алгоритма известны или могут быть найдены с заданной точностью до построения разбиения.

Исходный алгоритм считается корректным, т.е. он не содержит недостижимых вершин (требование живости), а также вершин, которые могут быть по-

вторно активизированы в процессе исполнения (требование безопасности); содержит ровно одну начальную и одну конечную вершины. Кроме того полагается, что каждый циклический фрагмент алгоритма содержит ровно один вход и один выход.

4. Этапы метода формирования разбиения

4.1. Разрыв циклов

Условием корректной работы некоторых из последующих этапов метода (построение системы *R*-выражений, матрицы отношений) является ациклический характер разбиваемого алгоритма. Однако реальные алгоритмы управления в общем случае могут содержать циклические фрагменты произвольной сложности. Таким образом, для их корректной обработки необходимо проведение ряда эквивалентных структурных преобразований, направленных на условную «ликвидацию» циклических фрагментов.

В [2] предлагается проводить замену циклов гипотетическими альтернативными ветвлениями ($\overline{\omega}$-преобразование) согласно правилу, проиллюстрированному на рис.2.

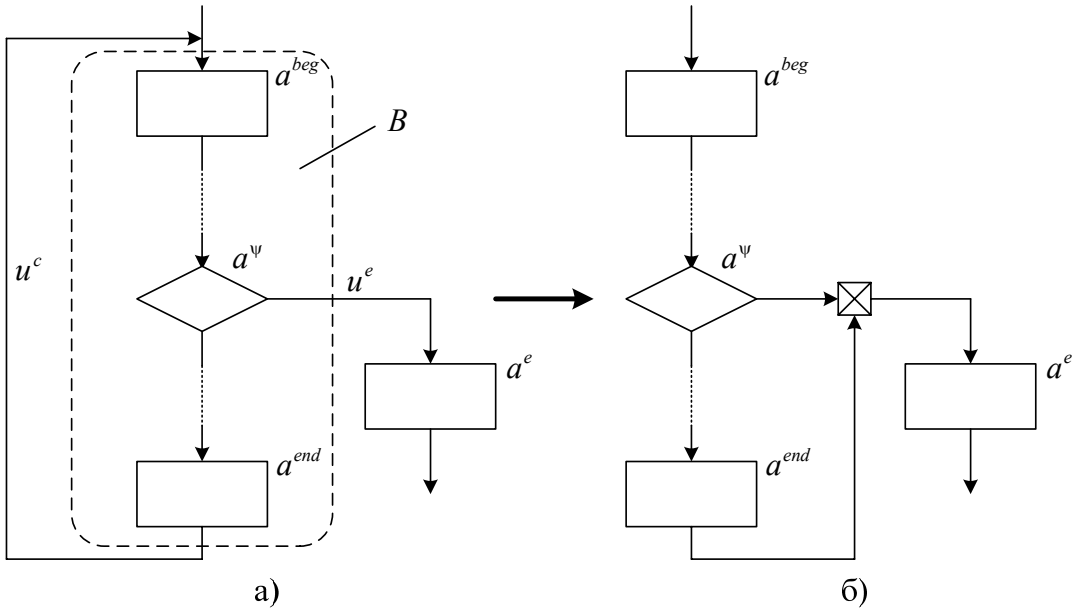

Рис. 2. Иллюстрация к правилу разрыва циклов

Замена циклов альтернативными ветвлениями нарушает структуру исходного алгоритма и отношения между вершинами. Однако, $\overline{\omega}$-преобразование сохраняет отношение параллельности (см. [2]), что является ключевым вопросом при распределении вершин по подалгоритмам с учетом ограничения (1).

Идентификация и разрыв циклов проводится в следующем порядке:

- выделение замыкающих дуг u^c, начальных a^{beg} и конечных a^{end} вершин циклов;
- построение тел циклов B (множеств вершин, входящих в состав циклов);

- выделение вершин выхода из цикла a^{Ψ}, вершин a^{e}, следующих непосредственно за циклом, и дуг выхода из цикла u^{e};
- добавление фиктивных вершин и дуг, перенастройка замыкающих дуг u^{c}.

Выделение замыкающих дуг является наиболее важным этапом, определяющим корректность проведения последующих преобразований. Идентификация замыкающих дуг проводится так, как описано в [3].

В результате проведения $\overline{\omega}$-преобразования получается граф-схема алгоритма G^{+}, не содержащая циклических фрагментов.

4.2. Объединение линейных участков

В алгоритмах управления довольно часто встречаются линейные (без разветвлений) последовательности операторных вершин (участки). Пример алгоритма управления, содержащего линейные участки, приведен на рис.3 (для наглядности линейные участки обведены штриховкой).

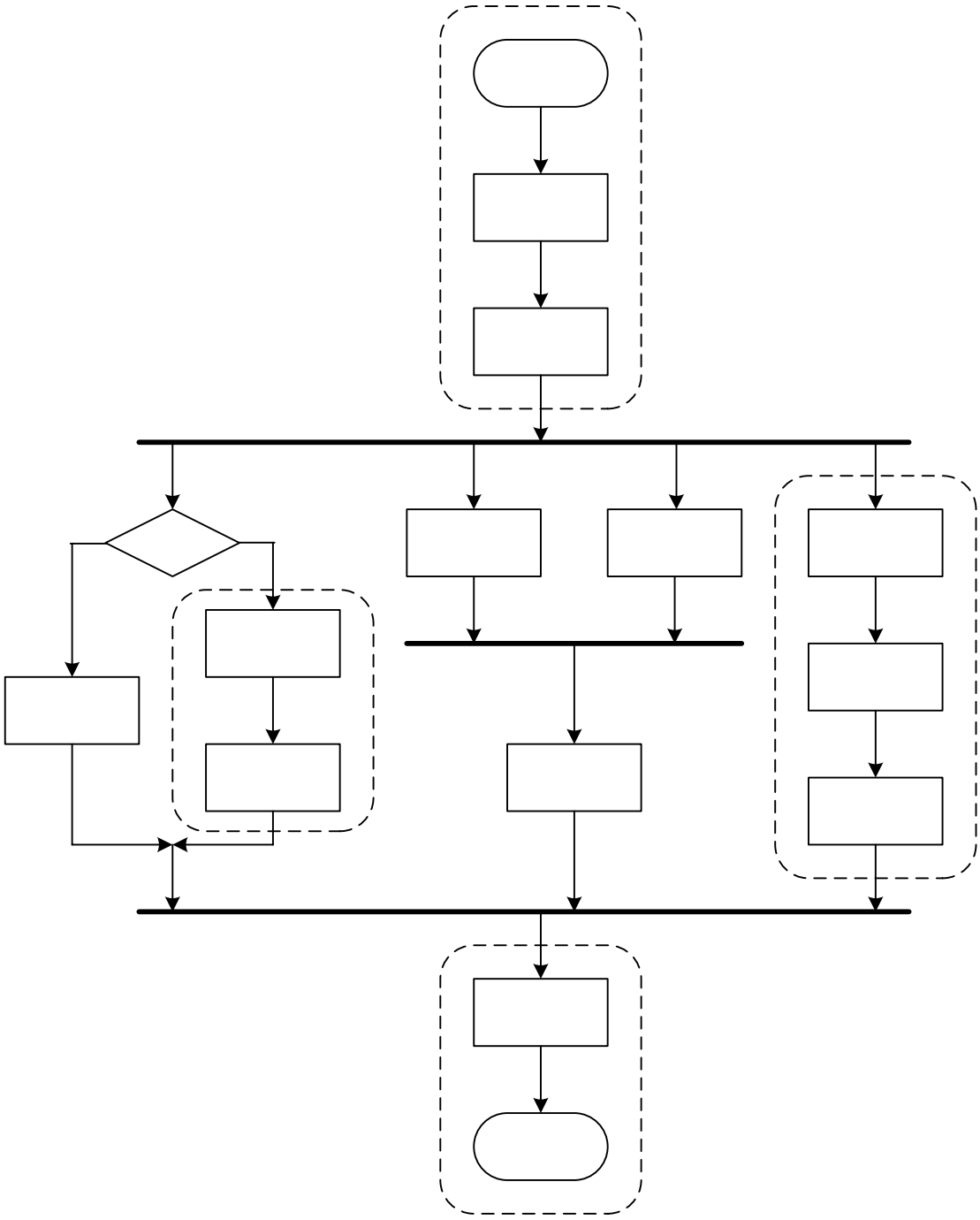

Рис. 3. Пример алгоритма, содержащего линейные участки

В состав линейного участка помимо операторных вершин может быть включена одна и только одна из перечисленных типов вершин: условная, объединения альтернативных дуг, начальная или конечная (рис. 1).

С целью повышения скорости нахождения разбиений проводится замена линейных участков обобщенными вершинами, причем все дальнейшие преобразования выполняются над алгоритмом, содержащим обобщенные вершины. Для алгоритма, изображенного на рис. 3, в результате объединения линейных участков получается алгоритм на рис. 4.

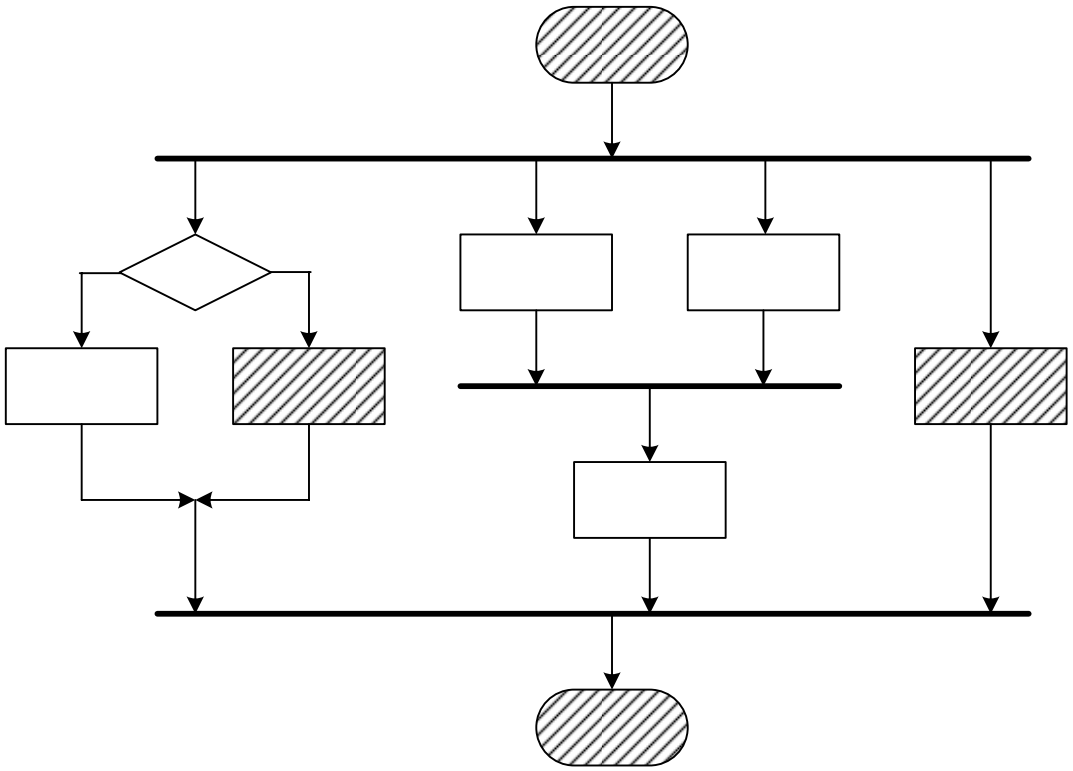

Рис. 4. Алгоритм рис. 3 с обобщенными вершинами (выделены штриховкой)

При образовании обобщенных вершин их параметры (тип, логические условия, микрооперации, вес) определяются на основе параметров вершин исходного алгоритма: тип вершины определяется составом вершин линейного участка (обобщенной вершине назначается тип операторной, если она состоит только из операторных вершин, в противном случае ее тип определяется типом «неоператорной» вершины: альтернативной, объединения альтернативных дуг, начальной или конечной, вошедшей в ее состав); $W = \sum_{a_i \in L} W(a_i)$, $X = \bigcup_{a_i \in L} X(a_i)$, $Y = \bigcup_{a_i \in L} Y(a_i)$, где L – линейный участок.

Алгоритм выделения линейных участков и образования обобщенных вершин заключается в итеративном применении следующей последовательности шагов:

- выделение очередной еще не рассмотренной операторной вершины;
- построение части линейного участка «вверх» (в направлении «против хода» выполнения алгоритма) до нахождения «неоператорной» вершины;
- построение части линейного участка «вниз» (по ходу выполнения алгоритма) до нахождения «неоператорной» вершины;
- замена найденного линейного участка обобщенной вершиной, пометка операторных вершин линейного участка как уже рассмотренных.

В результате проведения преобразования формируется граф G^{L+}, содержащий в общем случае меньшее количество вершин и дуг, чем исходный, и таблица соответствия вершин T^L, в которой хранится информация о линейных участках и соответствующих им обобщенных вершинах. Эта информация используется далее при построении подалгоритмов разбиения (переход к исходным вер-

шинам), а также в отладочных целях в программной системе (генерация протокола преобразования).

В общем случае возможно наличие алгоритмов, для которых более оптимальное разбиение достигается при рассмотрении линейных фрагментов состоящими из отдельных вершин, нежели как одной обобщенной вершины. Данная ситуация вызвана особенностью построения подалгоритмов разбиения (вершина целиком включается в состав одного из подалгоритмов без проведения каких-либо дополнительных действий). Также возможна ситуация, при которой обобщенная вершина в силу нарушения технологических ограничений не сможет «целиком» войти в состав какого-либо из подалгоритмов разбиения, что выльется в невозможность построения разбиения. Подобные ситуации, однако, маловероятны и механизм обобщенных вершин не оказывает существенного влияния на качество разбиений для большинства параллельных алгоритмов, но учет этих тонкостей в общем случае необходим. С этой целью при построении разбиений целесообразно запрещать либо разрешать проведение объединения линейных участков.

Экспериментальные исследования показывают, что уменьшение времени, затрачиваемого на построение разбиения при использовании механизма обобщенных вершин, составляет около 20%. Наибольший выигрыш заметен на этапах построения матрицы отношений (за счет сокращения размера матрицы отношений) и при построении множества смежных сечений (за счет уменьшения количества сечений и количества преобразований над *R*-выражениями).

4.3. Добавление фиктивных вершин

Для корректного проведения дальнейших преобразований (построение матрицы отношений, построение множества сечений) в исходном алгоритме управления не должно содержаться «пустых» дуг [2]. Под пустыми дугами в данном случае понимаются ветви альтернативных ветвлений или параллельные участки между вершинами синхронизации, не содержащие операторных вершин.

Наличие пустых дуг недопустимо из-за существенного усложнения процедуры разбиения алгоритма, поэтому необходимо добавление фиктивных операторных вершин по одной в каждую из них. Для фрагмента алгоритма, приведенного на рис. 5, результат добавления фиктивных операторных вершин приведен на рис. 6.

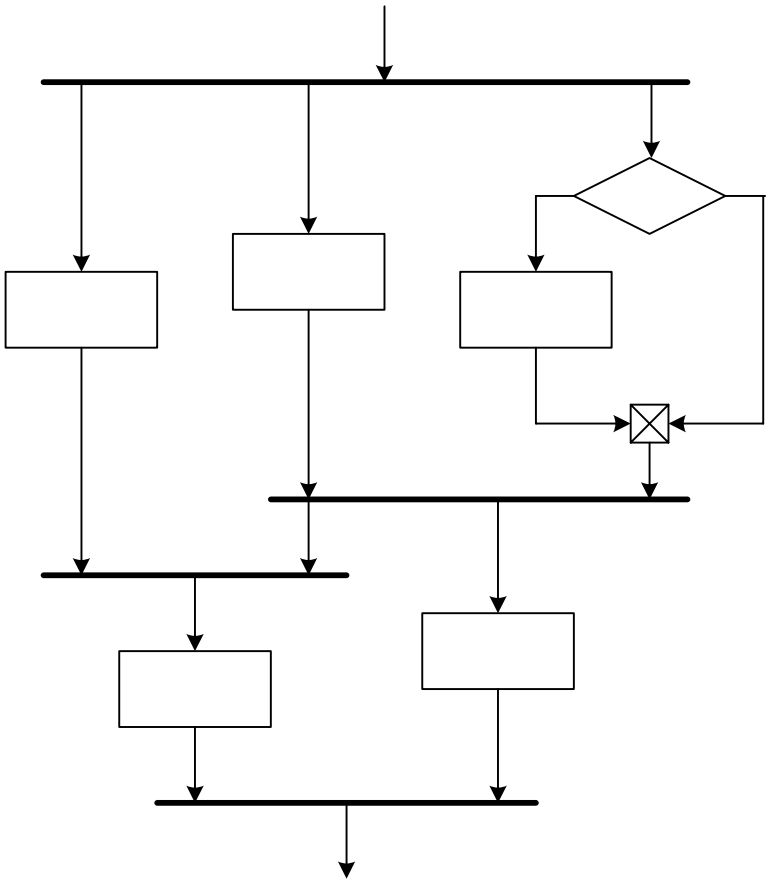

Рис. 5. Фрагмент алгоритма с пустыми дугами

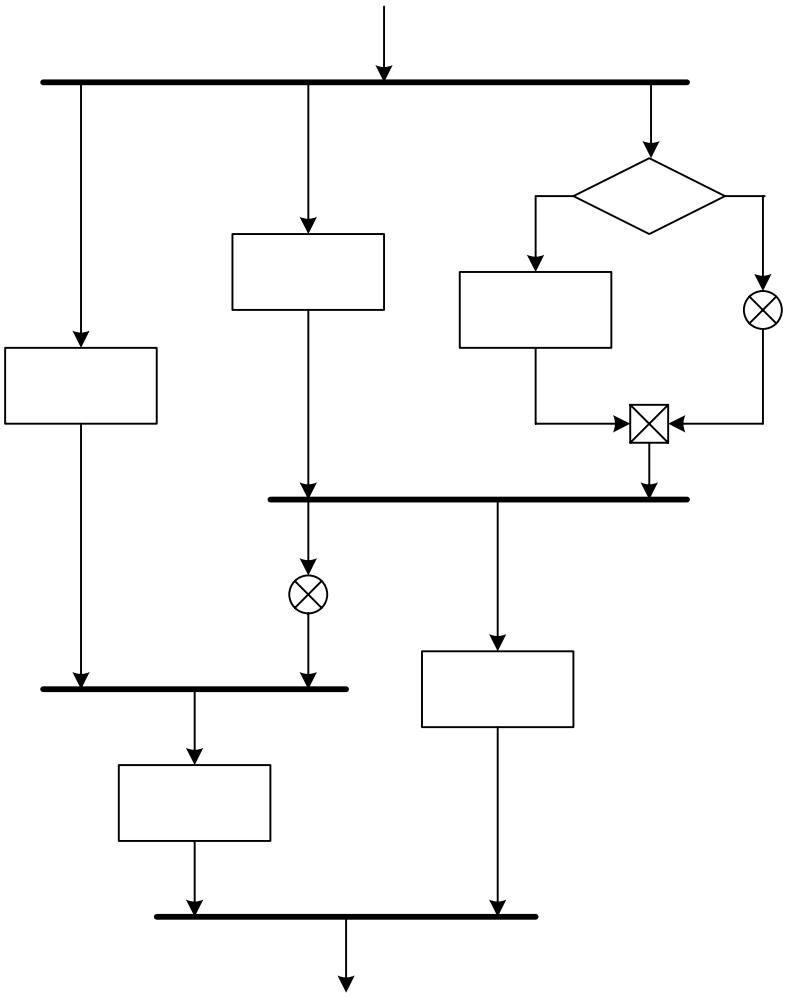

Рис. 6. Фрагмент алгоритма после «ликвидации» пустых дуг

Алгоритм поиска и устранения пустых дуг достаточно прост и заключается в сканировании дуг алгоритма управления и проверки типов вершин, коинцидентных рассматриваемой дуге.

В результате ликвидации пустых дуг получается граф $\tilde{G}$, удовлетворяющий всем ограничениям и пригодный для дальнейшего построения матрицы отношений *M* и системы *R*-выражений Ξ.

4.4. Построение матрицы отношений

При реализации некоторых этапов (построение множества сечений, синтез блоков разбиения) возникает необходимость в классификации отношений между вершинами. В рамках рассматриваемого метода интерес представляют следующие отношения [2,3]:

- следования (ν) – вершина a_i следует за вершиной a_j ($a_j \nu a_i$), если в графе *G* существует путь $L = \left\{a_i, a_{k_1}, a_{k_2}, ..., a_{k_n}, a_j\right\}$, соединяющий эти вершины;
- связи (φ) – вершины a_i и a_j находятся в отношении связи ($a_j \varphi a_i$), если в графе существует маршрут $P = \left\{a_i, a_{k_1}, a_{k_2}, ..., a_{k_n}, a_j\right\}$, соединяющий эти вершины;

- параллельности (ω) – вершины a_i и a_j параллельны ($a_j \omega a_i$), если они входят в состав различных параллельных ветвей алгоритма;
- альтернативы (ψ) – вершины a_i и a_j находятся в отношении альтернативы ($a_j \psi a_i$), если они принадлежат к разным ветвям одного и того же альтернативного ветвления.

Наиболее просто распределение отношений между вершинами описывает структура, называемая матрицей отношений. Она представляет собой квадратную матрицу M размерности $N \times N$ (N – число вершин графа G), каждый из элементов m_{ij} которой определяется отношениями между вершинами a_i и a_j. В качестве примера, иллюстрирующего распределение отношений, рассмотрим алгоритм рис. 7. Для упрощения вершины объединения альтернативных дуг не показаны, вершины синхронизации не включены в состав матрицы отношений.

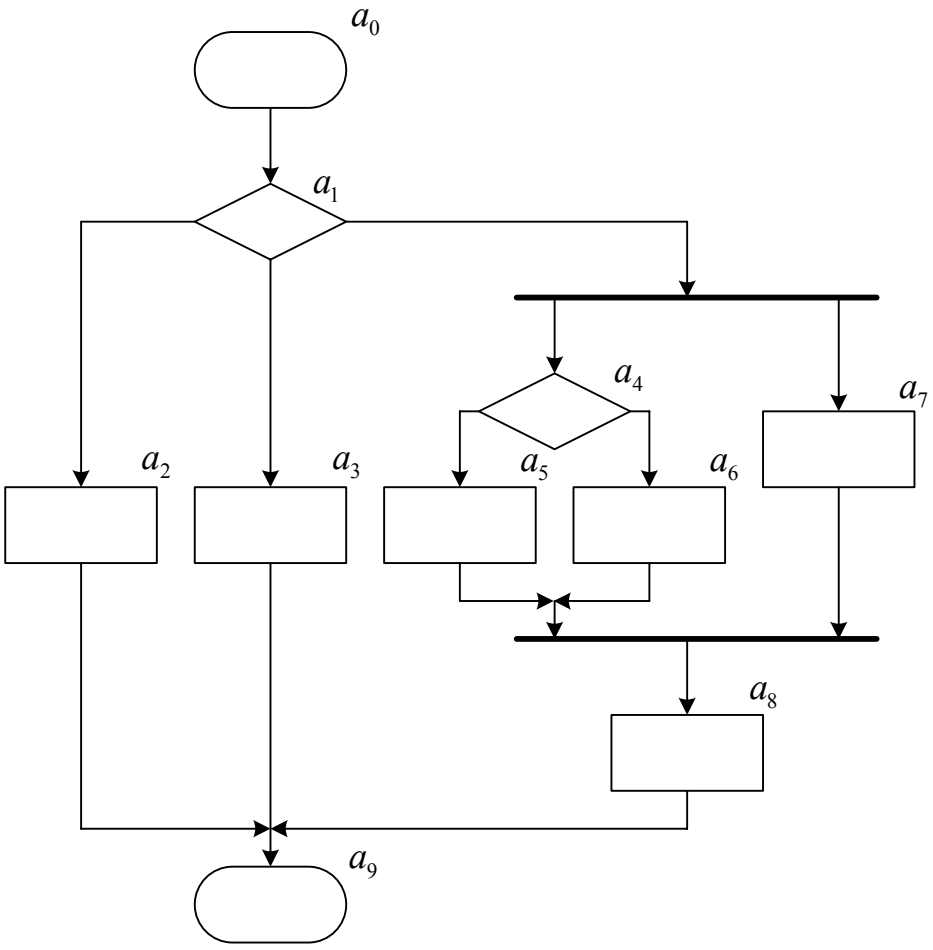

Рис. 7. Пример параллельного алгоритма

Для приведенного алгоритма матрица отношений имеет вид:

$$M=\begin{pmatrix} - & \varphi & \varphi & \varphi & \varphi & \varphi & \varphi & \varphi & \varphi & \varphi \\ \nu,\varphi & - & \varphi & \varphi & \varphi & \varphi & \varphi & \varphi & \varphi & \varphi \\ \nu,\varphi & \nu,\varphi & - & \psi & \psi & \psi & \psi & \psi & \psi & \varphi \\ \nu,\varphi & \nu,\varphi & \psi & - & \psi & \psi & \psi & \psi & \psi & \varphi \\ \nu,\varphi & \nu,\varphi & \psi & \psi & - & \varphi & \varphi & \omega & \varphi & \varphi \\ \nu,\varphi & \nu,\varphi & \psi & \psi & \nu,\varphi & - & \psi & \omega & \varphi & \varphi \\ \nu,\varphi & \nu,\varphi & \psi & \psi & \nu,\varphi & \psi & - & \omega & \varphi & \varphi \\ \nu,\varphi & \nu,\varphi & \psi & \psi & \omega & \omega & \omega & - & \varphi & \varphi \\ \nu,\varphi & \nu,\varphi & \psi & \psi & \nu,\varphi & \nu,\varphi & \nu,\varphi & \nu,\varphi & - & \varphi \\ \nu,\varphi & \nu,\varphi & \nu,\varphi & \nu,\varphi & \nu,\varphi & \nu,\varphi & \nu,\varphi & \nu,\varphi & \nu,\varphi & - \end{pmatrix}$$

Алгоритм построения матрицы отношений заключается в последовательном выяснении отношений в следующем порядке: отношение следования, отношение связи, отношение альтернативы, отношение параллельности на основании присущих им специфических свойств.

4.4.1. Отношение следования. Определение отношения следования основано на его транзитивности: если $a_i \nu a_j$ и $a_j \nu a_k$, то $a_i \nu a_k$. В качестве начальных значений задается отношение следования для вершин, соединенных дугой, после чего в матрице осуществляется поиск таких элементов m_{ij}, m_{jk} и m_{ik}, что

(2) $$\nu \in m_{ij},\ \nu \in m_{jk},\ \nu \notin m_{ik},$$

и производится включение отношения ν в множество отношений элемента m_{ik}. Процесс поиска и включения продолжается до тех пор, пока возможно нахождение элементов матрицы, удовлетворяющих (2).

4.4.2. Отношение связи. Вершины a_i и a_j находятся в отношении связи, если $a_i \nu a_j$ или $a_j \nu a_i$. Выяснение отношения сводится к поиску в матрице элементов m_{ij}, таких что $\nu \in m_{ij}$, и включению отношения φ в состав элементов m_{ij} и m_{ji}.

4.4.3. Отношение альтернативы. Выяснение отношения альтернативы сводится к поиску альтернативных ветвлений A_i^{ψ}, выделению альтернативных ветвей ветвлений $W_j \in A_i^{\psi}$ и заданию отношения альтернативы для вершин, входящих в состав различных ветвей в рамках одного ветвления: если $a_{k_1} \in W_{j_1}$, $a_{k_2} \in W_{j_2}$, причем $W_{j_1} \in A_i^{\psi}$ и $W_{j_2} \in A_i^{\psi}$, то $a_{k_1} \psi a_{k_2}$. Выделение альтернативных ветвей производится путем нахождения условной вершины и заполнения множеств вершин ветвей «вниз» (по ходу алгоритма) до тех пор, пока они не сойдутся в одной вершине. Для однозначного определения окончания альтернативного ветвления в граф-схему исходного алгоритма должны быть введены фиктивные вершины, получившие название вершин объединения альтернативных дуг (рис. 1ж).

4.4.4. Отношение параллельности. Отношения φ, ψ, ω образуют универсальное отношение $\varphi \cup \psi \cup \omega = A \times A$, причем $\varphi \cap \psi = \varnothing$ и $\psi \cap \omega = \varnothing$, откуда $a_i \omega a_j \Leftrightarrow \neg\left(a_i \varphi a_j\right) \& \neg\left(a_i \psi a_j\right)$ [2]. Т.е. для определения отношения ω необхо-

димо найти такие элементы m_{ij} матрицы отношений, для которых $\varphi \notin m_{ij}$, $\psi \notin m_{ij}$, и положить $m_{ij} = \omega$.

4.5. Построение системы *R*-выражений

Основу данного подхода к построению разбиения составляет формирование множества сечений параллельного ациклического алгоритма. Для этого главным образом используется структура данных, получившая название системы *R*-выражений Ξ, элементами которой являются выражения S_i вида $R_1^i \to R_2^i$, где R_1^i, R_2^i – конструктивные множества соответственно вершин-предшественников и вершин-последователей [2]. В качестве примера рассмотрим алгоритм рис. 8.

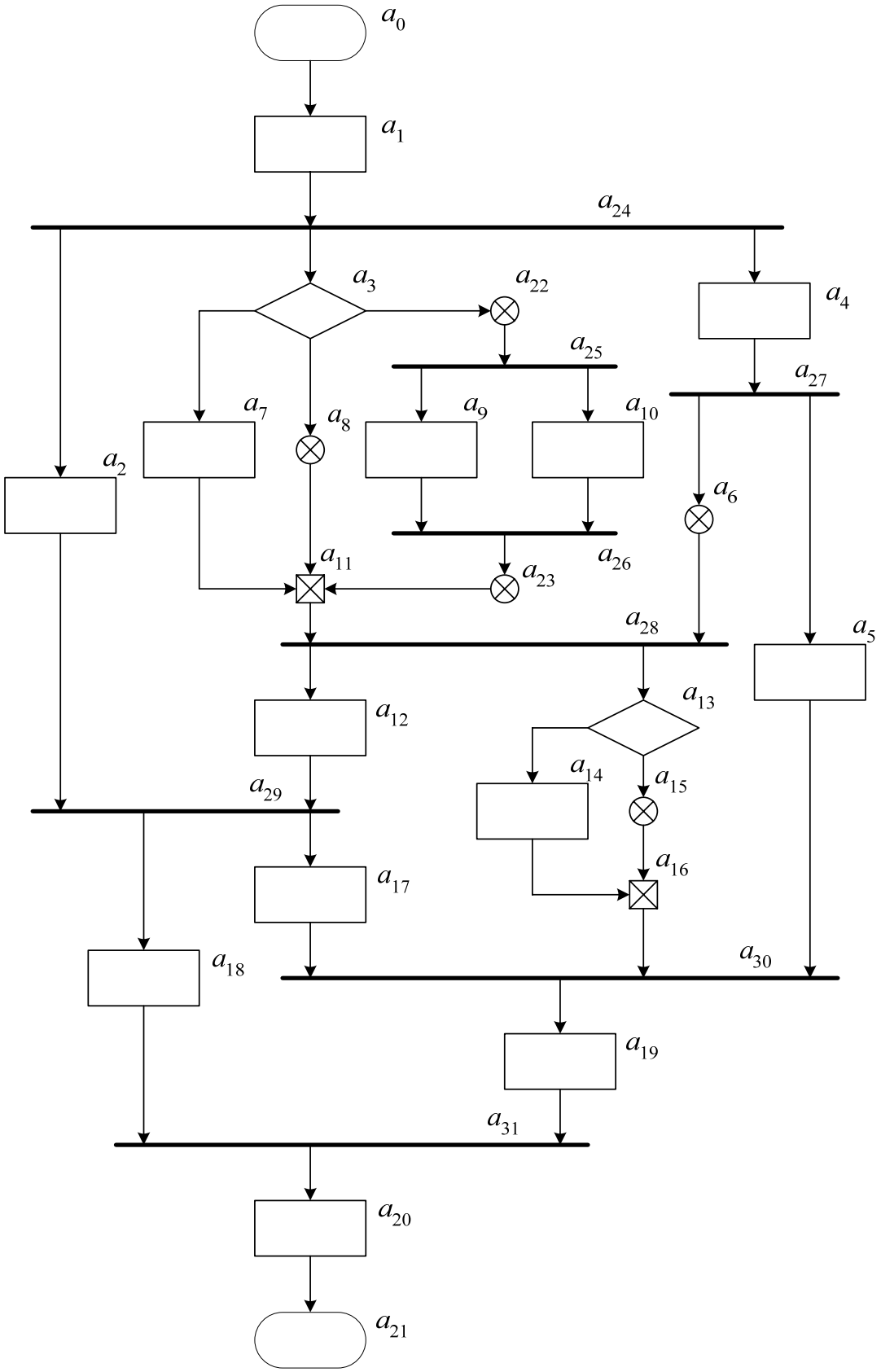

Рис. 8. Ациклический управляющий алгоритм

Система R-выражений, построенная по граф-схеме алгоритма управления (рис. 8), имеет вид:

$$
\begin{aligned}
& a_0 \to a_1 ; \\
& a_1 \to a_2 \bullet a_3 \bullet a_4 ; \\
& a_3 \to a_7 \mid a_8 \mid a_{22} ; \\
& a_{22} \to a_9 \bullet a_{10} ; \\
& a_9 \bullet a_{10} \to a_{23} ; \\
& a_7 \mid a_8 \mid a_{23} \to a_{11} ; \\
& a_4 \to a_5 \bullet a_6 ; \\
& a_6 \bullet a_{11} \to a_{12} \bullet a_{13} ; \\
& a_{13} \to a_{14} \mid a_{15} ; \\
& a_{14} \mid a_{15} \to a_{16} ; \\
& a_2 \bullet a_{12} \to a_{17} \bullet a_{18} ; \\
& a_5 \bullet a_{16} \bullet a_{17} \to a_{19} ; \\
& a_{18} \bullet a_{19} \to a_{20} ; \\
& a_{20} \to a_{21} .
\end{aligned}
\qquad (3)
$$

(при записи *R*-выражений в конструктивной форме символ «•» обозначает параллельность, «|» – альтернативу).

Хранение и обработка *R*-выражений могут быть реализованы как операции над деревьями [4]; совокупность *R*-выражений, образующих систему Ξ, является базовым набором для последующих этапов (выделение базового сечения, построение множества сечений). Синтез корректной системы выражений целиком определяет возможность получения множества смежных сечений. Важной особенностью является ациклический характер граф-схемы алгоритма, для которой требуется построение системы *R*-выражений.

Метод формирования системы *R*-выражений по граф-схеме алгоритма управления, предложенный в [2], труднореализуем на практике ввиду сложности построения конструктивного *R*-выражения из множества вершин, поэтому для синтеза системы Ξ предложен более простой метод, позволяющий свести построение к выделению набора простейших *R*-выражений. Под простейшими понимаются *R*-выражения, которые могут быть записаны без использования скобок. Алгоритм построения системы Ξ заключается в последовательном синтезе *R*-выражений в результате рассмотрения следующих характерным фрагментов алгоритмов управления:

- вершин синхронизации;
- условных вершин;
- вершин объединения альтернативных дуг;
- неиспользованных дуг.

Рассмотрим перечисленные этапы подробнее.

4.5.1. Вершины синхронизации. Вершины синхронизации не входят непосредственно в состав выражений системы Ξ, однако они учитываются в структуре получаемых *R*-выражений в виде отношения параллельности. В общем случае (рис. 9) вершина синхронизации имеет N вершин-предшественников и M вершин-последователей, $N \neq M$. Вершины-предшественники и вершины-последователи не обязательно должны иметь операторный тип.

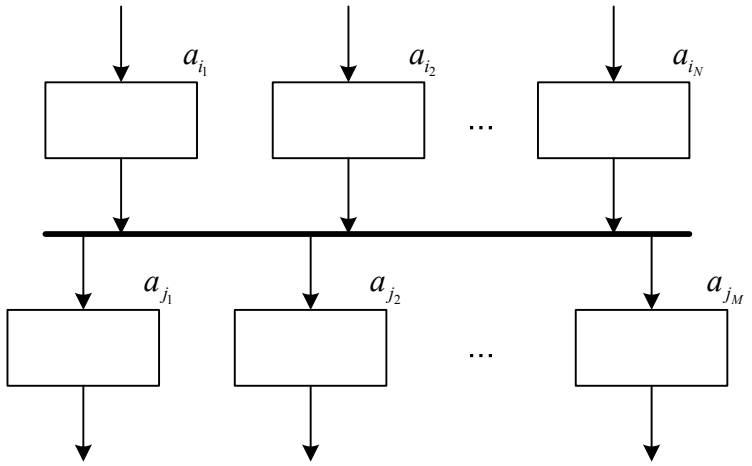

Рис. 9. Построение выражений системы Ξ по вершине синхронизации

Выражение системы Ξ, соответствующее изображенному фрагменту алгоритма, запишется в виде

$$a_{i_1} \bullet a_{i_2} \bullet ... \bullet a_{i_N} \rightarrow a_{j_1} \bullet a_{j_2} \bullet ... \bullet a_{j_M} .$$

4.5.2. Условные вершины. Условные вершины, являющиеся начальными вершинами альтернативных ветвлений, имеют $M > 1$ вершин последователей (рис. 10).

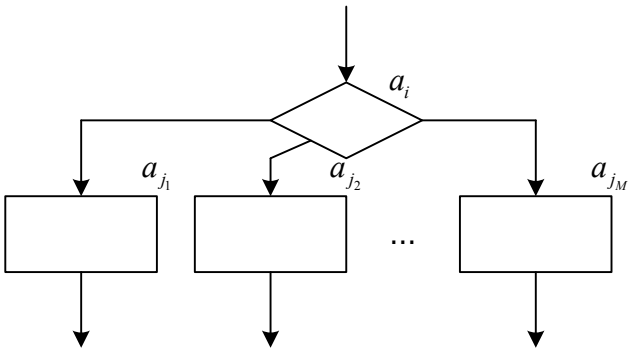

Рис. 10. Построение выражений системы Ξ по условной вершине

Выражение системы Ξ, соответствующее изображенному фрагменту алгоритма, запишется в виде

$$a_i \rightarrow a_{j_1} \mid a_{j_2} \mid ... \mid a_{j_M} .$$

4.5.3. Вершины объединения альтернативных дуг. Вершины объединения альтернативных дуг, являющиеся завершающими вершинами альтернативных ветвлений, обладают свойством фиктивности и должны быть введены в исходный алгоритм управления в места, где происходит объединение двух и более альтернативных ветвей (рис. 11).

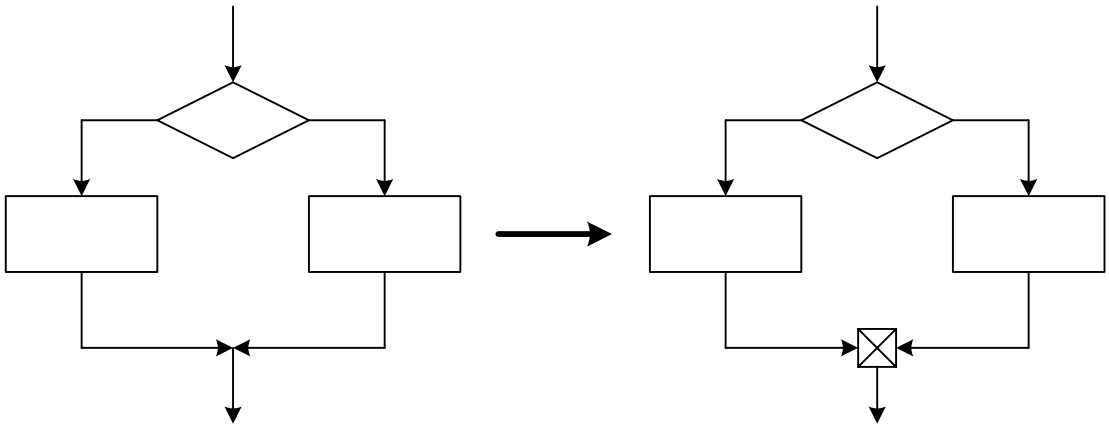

Рис. 11. Включение вершин объединения альтернативных дуг

Вершины объединения альтернативных дуг необходимы для построения некоторых выражений системы Ξ. В общем случае они имеют $N > 1$ предшественников (рис. 12).

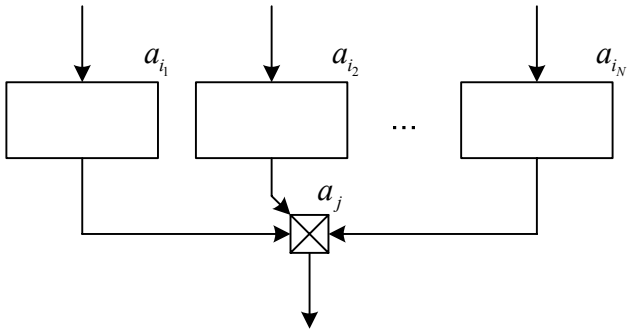

Рис. 12. Построение выражений системы Ξ по вершине объединения альтернативных дуг

Выражение системы Ξ, соответствующее изображенному фрагменту алгоритма, запишется в виде

$$a_{i_1} \mid a_{i_2} \mid ... \mid a_{i_N} \to a_j$$

4.5.4. Неиспользованные дуги. При выполнении перечисленных выше действий необходимо вести учет использованных дуг, т.к. в общем случае в алгоритме управления возможно наличие линейных участков или альтернативных ветвлений, следующих одно за другим (условная вершина, следующая непосредственно за вершиной объединения альтернативных дуг), которые не были рассмотрены и не нашли отражения в системе Ξ. Поэтому на завершающем шаге построения системы выражений необходимо рассмотрение неиспользованных дуг и синтез для них *R*-выражений вида

$$a_i \to a_j .$$

В результате перечисленных шагов получается система выражений Ξ, в которой отражена структура алгоритма управления, учтены все вершины и все дуги.

4.6. Выделение базового сечения

Построение множества сечений начинается с выделения базового сечения. Под сечением Ω_i понимается группа вершин алгоритма управления, не связанных между собой отношением связи. Для определения базового сечения введем понятие ω-мощности. Под ω-мощностью $\left|\Omega_i\right|^{\omega}$ понимается максимальное число попарно параллельных вершин, входящих в состав сечения Ω_i [2]. Базовым сечением $\Omega_{\max}$ называется одно из сечений алгоритма, обладающее максимальной ω-мощностью. ω-мощность базового сечения $\Omega_{\max}$ является важным числовым параметром и характеризует минимальное число блоков разбиения алгоритма управления.

Наиболее простым способом является выделение всех возможных сечений и выбор из них максимального с точки зрения ω-мощности. Однако такой способ не является приемлемым в силу того, что он требует построения и анализа значительного числа сечений, являющегося в общем случае экспоненциальной функцией от числа вершин алгоритма управления. Поэтому для выделения базового сечения воспользуемся алгоритмом, описанным в [2] и уточненным в [4,5], который основан на выполнении серии операций преобразования и обеспечивающий сведение исходной системы Ξ к системе Ξ^*, содержащей единственное (базовое) сечение.

Прежде чем перейти к непосредственному рассмотрению алгоритма выделения базового сечения введем понятие конструктивного включения. Под отношением конструктивного включения ($\left[\subseteq\right]$) будем подразумевать такое отношение между *R*-выражениями $R_i\left[\subseteq\right]R_j$, при котором $R_i = \xi_k$, где ξ_k – некоторое конструктивное подмножество множества R_j, т.е. подмножество вершин, заключенных в единые скобки.

Пусть, для примера, $R_j = a_1 \mid \left(a_2 \bullet \left(a_3 \mid a_4\right) \bullet a_5\right)$, тогда конструктивными подмножествами для заданного выражения являются:

$$\xi_1 = a_3;$$
$$\xi_2 = a_4;$$
$$\xi_3 = a_3 \mid a_4;$$
$$\xi_4 = a_2;$$
$$\xi_5 = a_5;$$
$$\xi_6 = a_2 \bullet \left(a_3 \mid a_4\right);$$
$$\xi_7 = \left(a_3 \mid a_4\right) \bullet a_5;$$
$$\xi_8 = a_2 \bullet a_5;$$
$$\xi_9 = a_2 \bullet \left(a_3 \mid a_4\right) \bullet a_5;$$
$$\xi_{10} = a_1;$$
$$\xi_{11} = R_j = a_1 \mid \left(a_2 \bullet \left(a_3 \mid a_4\right) \bullet a_5\right).$$

Определим базовые операции алгоритма.

4.6.1. Правило *и*-поглощения. Если в системе выражений Ξ можно найти два различных выражения S_i, S_j таких, что $R_1^j\left[\subseteq\right]R_2^i$, $\left|R_2^j\right|^{\omega} \geq \left|R_1^j\right|^{\omega}$ и конечная

вершина алгоритма управления не входит в состав R_2^j, то необходимо исключить из системы выражение S_j и произвести замену конструктивного подмножества R_1^j на R_2^j в составе R_2^i.

4.6.2. Правило *d*-поглощения. Если в системе выражений Ξ можно найти два различных выражения S_i, S_j таких, что $R_2^i \left[\subseteq\right] R_1^j$, $\left|R_1^i\right|^{\omega} \geq \left|R_2^i\right|^{\omega}$ и начальная вершина алгоритма управления не входит в состав R_1^i, то необходимо исключить из системы выражение S_i и произвести замену конструктивного подмножества R_2^i на R_1^i в составе R_1^j.

4.6.3. Правило ψ-перегруппировки (*p*→*a*-форма). Если в системе выражений Ξ можно найти два различных выражения S_i, S_j, для которых $R_1^j \subseteq R_2^i$, но $R_1^j \left[\not\subseteq\right] R_2^i$ и R_2^i включает конструктивное подмножество вида $\xi_k = (a_{11} \mid a_{12} \mid ... \mid a_{1n}) \bullet (a_{21} \mid a_{22} \mid ... \mid a_{2n}) \bullet ... \bullet (a_{m1} \mid a_{m2} \mid ... \mid a_{mn})$, то подмножество ξ_k должно быть преобразовано к виду $\xi_k^* = (a_{11} \bullet a_{21} \bullet ... \bullet a_{m1}) \mid (a_{12} \bullet a_{22} \bullet ... \bullet a_{m2}) \mid ... \mid (a_{1n} \bullet a_{2n} \bullet ... \bullet a_{mn})$ (перегруппировано) в составе R_2^i.

На основании введенных правил алгоритм выделения базового сечения можно представить в следующем виде.

1) Выбрать в системе Ξ выражения S_i, S_j, удовлетворяющие условиям проведения подстановки (*u*- или *d*-), выполнить подстановку. Если выбор выражений невозможен, перейти к п.2, в противном случае повторить действия п.1.
2) Выбрать в системе Ξ выражения S_i, S_j, удовлетворяющие условиям проведения перегруппировки, произвести перегруппировку. Если выбор выражений невозможен, перейти к п.3, в противном случае к п.1.
3) Конец алгоритма.

В результате выполнения алгоритма образуется система выражений Ξ^* вида

$$\begin{matrix} a_0 \to R_\gamma \\ R_\gamma \to a_k \end{matrix},$$

в которой R_γ является искомым базовым сечением. Наличие в результирующей системе Ξ^* более чем двух выражений говорит о некорректном формате исходной системы Ξ, который может быть связан с ошибкой в записи исходного алгоритма управления, наличием в нем циклов, неликвидированных пустых дуг и т.д.

Применение правил *u*- и *d*-поглощения уменьшает количество выражений системы Ξ и способствует образованию *R*-выражений, обладающих большей ω-мощностью, что в конечном счете приводит к образованию одного наиболее ω-мощного *R*-выражения – базового сечения. Операция ψ-перегруппировки, не изменяющая количество и ω-мощность выражений, является вспомогательной и служит для устранения неоднозначностей при записи *R*-выражений для алгоритмов управления, содержащих параллельные альтернативные ветвления и/или параллельные циклы. Для образования корректного перегруппированного выражения ξ_k^* необходимо проведение ряда проверок по матрице отношений и структуре выражения R_1^j, описанных в [4,5].

Рассмотрим процесс выделения базового сечения на примере системы Ξ (3), соответствующей алгоритму управления рис. 8. Ниже приведен листинг протокола преобразований, полученный с использованием разработанной программной системы.

Исходная система выражений:

0: a1 -> a2*a3*a4
1: a22 -> a9*a10
2: a9*a10 -> a23
3: a7|a8|a23 -> a11
4: a3 -> a7|a8|a22
5: a6*a11 -> a12*a13
6: a4 -> a5*a6
7: a2*a12 -> a17*a18
8: a14|a15 -> a16
9: a13 -> a14|a15
10: a5*a16*a17 -> a19
11: a18*a19 -> a20
12: a0 -> a1
13: a20 -> a21

u-поглощение 0, 4:

0: a1 -> a2*a4*(a7|a8|a22)
1: a22 -> a9*a10
2: a9*a10 -> a23
3: a7|a8|a23 -> a11
4: a6*a11 -> a12*a13
5: a4 -> a5*a6
6: a2*a12 -> a17*a18
7: a14|a15 -> a16
8: a13 -> a14|a15
9: a5*a16*a17 -> a19
10: a18*a19 -> a20
11: a0 -> a1
12: a20 -> a21

u-поглощение 0, 1:

0: a1 -> a2*a4*(a7|a8|(a9*a10))
1: a9*a10 -> a23
2: a7|a8|a23 -> a11
3: a6*a11 -> a12*a13
4: a4 -> a5*a6
5: a2*a12 -> a17*a18
6: a14|a15 -> a16
7: a13 -> a14|a15
8: a5*a16*a17 -> a19
9: a18*a19 -> a20
10: a0 -> a1
11: a20 -> a21

u-поглощение 0, 4:

0: a1 -> a2*a5*a6*(a7|a8|(a9*a10))
1: a9*a10 -> a23
2: a7|a8|a23 -> a11
3: a6*a11 -> a12*a13
4: a2*a12 -> a17*a18
5: a14|a15 -> a16

6: a13 -> a14|a15
7: a5*a16*a17 -> a19
8: a18*a19 -> a20
9: a0 -> a1
10: a20 -> a21

d-поглощение 1, 2:

0: a1 -> a2*a5*a6*(a7|a8|(a9*a10))
1: a7|a8|(a9*a10) -> a11
2: a6*a11 -> a12*a13
3: a2*a12 -> a17*a18
4: a14|a15 -> a16
5: a13 -> a14|a15
6: a5*a16*a17 -> a19
7: a18*a19 -> a20
8: a0 -> a1
9: a20 -> a21

d-поглощение 1, 2:

0: a1 -> a2*a5*a6*(a7|a8|(a9*a10))
1: a6*(a7|a8|(a9*a10)) -> a12*a13
2: a2*a12 -> a17*a18
3: a14|a15 -> a16
4: a13 -> a14|a15
5: a5*a16*a17 -> a19
6: a18*a19 -> a20
7: a0 -> a1
8: a20 -> a21

u-поглощение 1, 4:

0: a1 -> a2*a5*a6*(a7|a8|(a9*a10))
1: a6*(a7|a8|(a9*a10)) -> a12*(a14|a15)
2: a2*a12 -> a17*a18
3: a14|a15 -> a16
4: a5*a16*a17 -> a19
5: a18*a19 -> a20
6: a0 -> a1
7: a20 -> a21

u-поглощение 1, 3:

0: a1 -> a2*a5*a6*(a7|a8|(a9*a10))
1: a6*(a7|a8|(a9*a10)) -> a12*a16
2: a2*a12 -> a17*a18
3: a5*a16*a17 -> a19
4: a18*a19 -> a20
5: a0 -> a1
6: a20 -> a21

d-поглощение 3, 4:

0: a1 -> a2*a5*a6*(a7|a8|(a9*a10))
1: a6*(a7|a8|(a9*a10)) -> a12*a16
2: a2*a12 -> a17*a18
3: a5*a16*a17*a18 -> a20
4: a0 -> a1
5: a20 -> a21

d-поглощение 2, 3:

0: a1 -> a2*a5*a6*(a7|a8|(a9*a10))
1: a6*(a7|a8|(a9*a10)) -> a12*a16
2: a2*a5*a12*a16 -> a20
3: a0 -> a1
4: a20 -> a21

d-поглощение 1, 2:
0: a1 -> a2*a5*a6*(a7|a8|(a9*a10))
1: a2*a5*a6*(a7|a8|(a9*a10)) -> a20
2: a0 -> a1
3: a20 -> a21

d-поглощение 1, 3:
0: a1 -> a2*a5*a6*(a7|a8|(a9*a10))
1: a0 -> a1
2: a2*a5*a6*(a7|a8|(a9*a10)) -> a21

u-поглощение 1, 0:
0: a0 -> a2*a5*a6*(a7|a8|(a9*a10))
1: a2*a5*a6*(a7|a8|(a9*a10)) -> a21

Базовое сечение: a2*a5*a6*(a7|a8|(a9*a10))
w-мощность: 5

4.7. Построение множества смежных сечений

Непосредственно после выделения базового сечения начитается построение множества смежных сечений. Процесс перебора сечений заключается в последовательном получении так называемых *u*- и *d*-сечений, начиная от базового. Под *u*-сечением Ω_{i-1}^{u} для Ω_i понимается сечение, образованное из вершин, содержащихся в Ω_i либо связанных дугой с вершинами из Ω_i, причем начало дуги приходится на вершину, входящую в состав *u*-сечения. Под *d*-сечением Ω_{i+1}^{d} для Ω_i понимается сечение, образованное из вершин, содержащихся в Ω_i либо связанных дугой с вершинами из Ω_i, причем конец дуги приходится на вершину, входящую в состав *d*-сечения. Процесс построения сечений в каждом из направлений (вверх и вниз) завершается, когда начальная либо конечная вершина входит в состав вновь сформированного сечения. Пример сечения и построенных для него *u*- и *d*-сечения приведен на рис. 13.

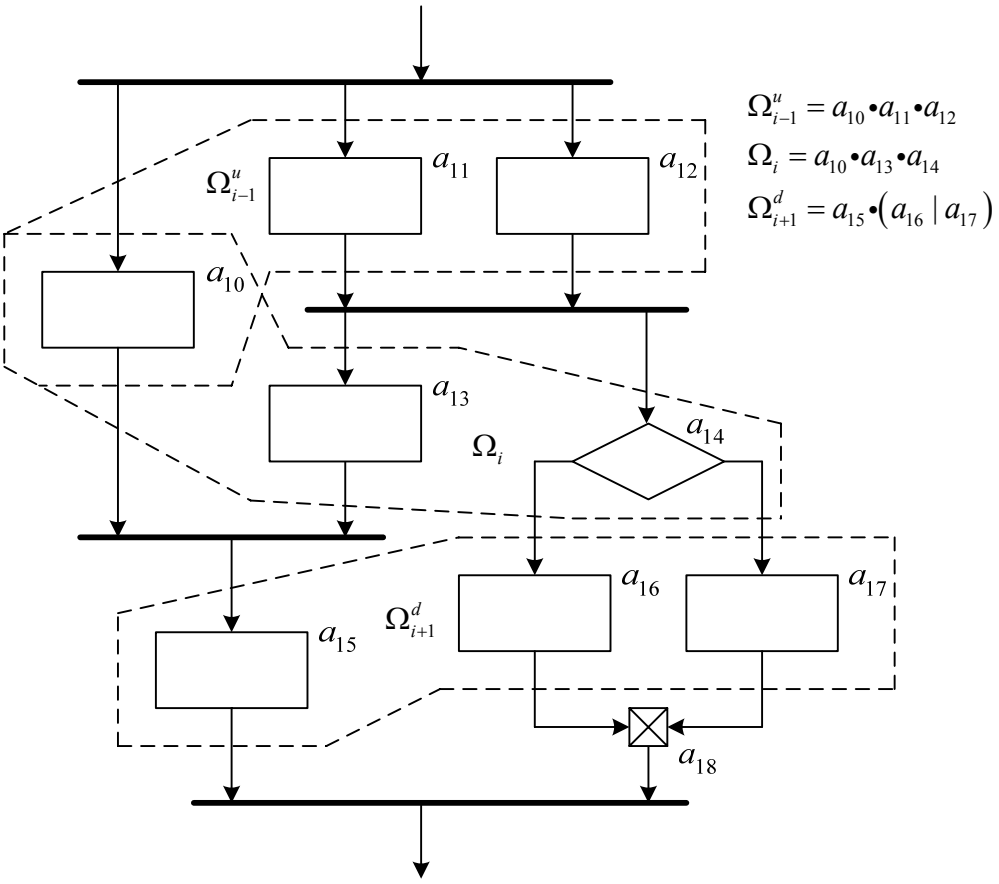

Рис. 13. Иллюстрация к построению *u*- и *d*-сечений

Для построения *u*- и *d*-сечений используется система выражений Ξ. Построение *u*-сечения для сечения Ω заключается в выделении множества выражения системы Ξ, для которых $R_2^i \subseteq \Omega$, и проведении подстановки выражений R_1^i вместо R_2^i. Формирование *d*-сечения заключается в выделении множества выражений системы Ξ, для которых $R_1^i \subseteq \Omega$, и проведении подстановки выражений R_2^i вместо R_1^i. По сути процедуры построения *u*- и *d*-сечений схожи и отличаются лишь направленностью действия. В процессе проведения подстановок возможна необходимость проведения ψ-перегруппировок сечения Ω [4] (*a*→*p*-перегруппировки в случае построения *u*-сечения, *p*→*a*-перегруппировки в случае построения *d*-сечения).

Рассмотрим процесс построения множества смежных сечений на примере алгоритма управления рис. 8. Ниже приведен листинг протокола преобразований, полученный с использованием разработанной программной системы.

Базовое сечение: a2*a5*a6*(a7|a8|(a9*a10))

Перебор u-сечений:

Множество подходящих выражений: {1,6}
u-сечение: a2*a4*(a7|a8|a22)

Множество подходящих выражений: {4}

u-сечение: a2*a3*a4

Множество подходящих выражений: {0}
u-сечение: a1

Множество подходящих выражений: {12}
u-сечение: a0

Перебор d-сечений:

Множество подходящих выражений: {2}
d-сечение: a2*a5*a6*(a7|a8|a23)

Множество подходящих выражений: {3}
d-сечение: a2*a5*a6*a11

Множество подходящих выражений: {5}
d-сечение: a2*a5*a12*a13

Множество подходящих выражений: {7,9}
d-сечение: a5*a17*a18*(a14|a15)

Множество подходящих выражений: {8}
d-сечение: a5*a16*a17*a18

Множество подходящих выражений: {10}
d-сечение: a18*a19

Множество подходящих выражений: {11}
d-сечение: a20

Множество подходящих выражений: {13}
d-сечение: a21

Сечения (в конструктивной форме):

a0
a1
a2*a3*a4
a2*a4*(a7|a8|a22)
a2*a5*a6*(a7|a8|(a9*a10))
a2*a5*a6*(a7|a8|a23)
a2*a5*a6*a11
a2*a5*a12*a13
a5*a17*a18*(a14|a15)
a5*a16*a17*a18
a18*a19
a20
a21

В результате применения процедур получения *u*- и *d*-сечений формируется множество смежных сечений алгоритма управления

$$\Theta = \{\Omega_0, \Omega_1, ..., \Omega_{\max}, ..., \Omega_{n-1}, \Omega_n\},$$

которое используется далее при построении блоков разбиения.

Одним из вариантов реализации операций над *R*-выражениями, предложенным в [4], является использование деревьев для представления *R*-выражений.

Операции над *R*-выражениями (проверки отношений, поглощения, перегруппировки) допускают разбиение на ряд элементарных операций (проверка отношения нестрогого включения, сравнение поддеревьев, проверка принадлежности вершины сечению, вычисление ω-мощности, раскрытие скобок, проверка структуры выражения при перегруппировках, получение перегруппированного подвыражения, подстановка деревьев). Состоятельность данного подхода подтверждается экспериментальными исследованиями.

4.8. Построение скелетного графа

При синтезе блоков разбиения важная роль отводится учету и минимизации межблочных взаимодействий, что способствует минимизации сложности сети межблочных связей и времени межмодульного обмена в системе логического управления [2]. Определение вероятности передачи управления между различными блоками и среднего числа передач активности осуществляется с использованием параметров β и δ дуг исходного алгоритма управления (см. выше). Считается, что эти параметры известны или могут быть вычислены с заданной точностью.

Особенности постановки задачи, к которым в первую очередь следует отнести построение разбиения без учета распределения по блокам вершин синхронизации и реализации механизмов синхронизации, делают затруднительной оценку числовых параметров получаемой сети межблочных взаимодействий непосредственно по граф-схеме алгоритма управления, поэтому возникает необходимость в построении особой структуры данных (графа), которая позволит просто и эффективно проводить численную оценку интересующих параметров. Такой структурой данных является скелетный граф $S = \langle A_S, U_S \rangle$ – ориентированный взвешенной граф, вершины которого однотипны и соотнесены вершинам исходного алгоритма управления G_0 за исключением вершин синхронизации, а дуги в основном образованы дугами алгоритма управления G_0 за исключением дуг, инцидентных вершинам синхронизации. Дуги графа взвешены параметрами β и δ. Скелетный граф *S* строится по исходному алгоритму управления G_0, т.к. дальнейшие преобразования (добавление фиктивных вершин, разрыв циклов и т.д.) искажают его. Пример алгоритма управления и соответствующего ему скелетного графа приведен на рис. 14.

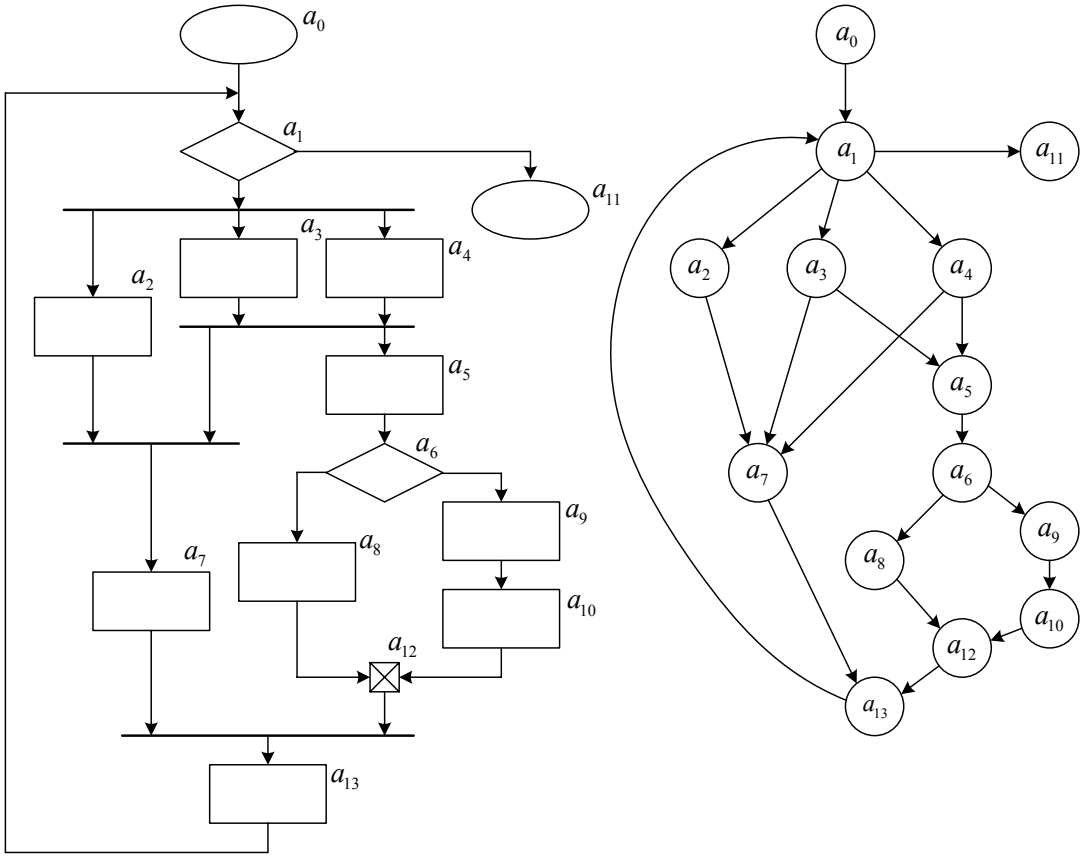

Рис. 14. Алгоритм и соответствующий ему скелетный граф

Алгоритм построения скелетного графа фактически направлен на получение графа без вершин синхронизации, что избавляет от проведения дополнительных проверок при его использовании. Однако простым удалением вершин синхронизации ограничиться нельзя, т.к. нарушится связность графа и будут потеряны некоторые из направлений передачи управления. Чтобы не допустить искажения графа удалению вершин синхронизации предшествует процедура выделения множеств вершин-предшественников и вершин-последователей для каждой из вершин синхронизации и образование из вершин этих множеств двудольного подграфа путем добавления дуг. Для корректно заданного алгоритма параметры дуг, подходящих к вершине синхронизации, совпадают, поэтому добавленные дуги взвешиваются набором параметров любой из удаленных дуг, инцидентных вершине синхронизации. В общем случае вершины синхронизации с N предшественниками и M последователями (рис. 9) подлежат замене на двудольный подграф следующим образом (рис. 15).

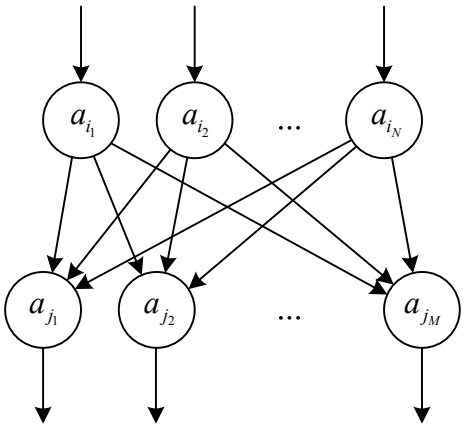

Рис. 15. Замена вершин синхронизации при построении скелетного графа

Скелетный граф используется на заключающем этапе метода при построении блоков разбиения для оценки приращений сложности сети межблочных связей и интенсивности межблочных взаимодействий.

4.9. Построение блоков разбиения

Последним и наиболее важным в рамках рассматриваемого метода является этап построения блоков разбиения (подалгоритмов) исходного алгоритма управления. Синтез подалгоритмов из субсечений проводится параллельно на основании весовой функции, учитывающей размещение микроопераций, логических условий и приращения интенсивности межблочных взаимодействий, а также технологические и функциональные ограничения. Процесс размещения вершин сечений в блоках разбиения начинается с базового сечения и последовательно повторяется в направлении начальной и конечной вершин для *u*- и *d*-сечений.

Прежде всего для сечения Ω_i формируется его разбиение на субсечения $\rho_1, \rho_2, ..., \rho_N$ таким образом, чтобы параллельные вершины были размещены в различных субсечениях. При этом соблюдается ортогональность разбиения, т.е. одна и та же вершина не может войти в состав нескольких субсечений. Разбиение строится путем анализа матрицы отношений и выделения групп вершин, не связанных отношением параллельности. В общем случае возможно получение нескольких вариантов разбиения сечения на множество субсечений, выбирается любой из них.

Для каждого субсечения определяется возможность и оптимальность включения в блоки путем заполнения и анализа таблицы включений. Таблица включений представляет собой квадратную матрицу размерности $N \times M$, где N в данном случае – количество субсечений, M – число блоков разбиения, элементами которой являются значения весовой функции, рассчитанные для каждого элемента таблицы T_{ij} при условии включения *i*-го субсечения в *j*-й блок. Весовая функция вычисляется по следующей формуле:

$$t\left(\rho_i, A_j\right) = \begin{cases} \text{"}-\text{"}, \exists a_m \in \rho_i, a_n \in A_j : a_m \omega a_n \\ \text{"}+\text{"}, \left(W\left(\rho_i\right) + W\left(A_j\right) > W_{\max}\right) \vee \left(\left|X\left(\rho_i\right) \cup X\left(A_j\right)\right| > n_{ЛУ}\right) \\ K_1^Y \left|Y\left(\rho_i\right) \cap Y\left(A_j\right)\right| - K_2^Y \left|Y\left(\rho_i\right) \setminus Y\left(A_j\right)\right| + \\ + K_1^X \left|X\left(\rho_i\right) \cap X\left(A_j\right)\right| - K_2^X \left|X\left(\rho_i\right) \setminus X\left(A_j\right)\right| - \\ -\left[K_1^Z \Delta Z_1 + K_2^Z \Delta Z_2 + K_W \Delta W\right] \text{ в противном случае} \end{cases},$$

где ΔZ_1, ΔZ_2 – приращения соответственно сложности сети межблочных взаимодействий и числа межблочных взаимодействий; ΔW – дискриминант алгоритмов по сложности; K_1^X, K_2^X, K_1^Y, K_2^Y, K_W, K_1^Z, K_2^Z – коэффициенты значимости соответствующих параметров.

Наибольшую сложность представляет вычисление приращений ΔZ_1 и ΔZ_2. Для этого используется граф межблочных взаимодействий, который представляет собой ориентированный взвешенный граф, вершинам которого сопоставлены блоки разбиения, а дугам – межблочные связи. Каждой дуге графа сопоставляются числовые коэффициенты: α_{ij} – вероятность взаимодействия подалгоритмов A_i и A_j в направлении от A_i к A_j; $\tilde{\alpha}_{ij}$ – среднее количество передач активности между подалгоритмами. Пересчет коэффициентов происходит всякий раз после распределения подмножеств по блокам. Пример разбиения алгоритма и соответствующий ему граф межблочных взаимодействий приведен на рис. 16.

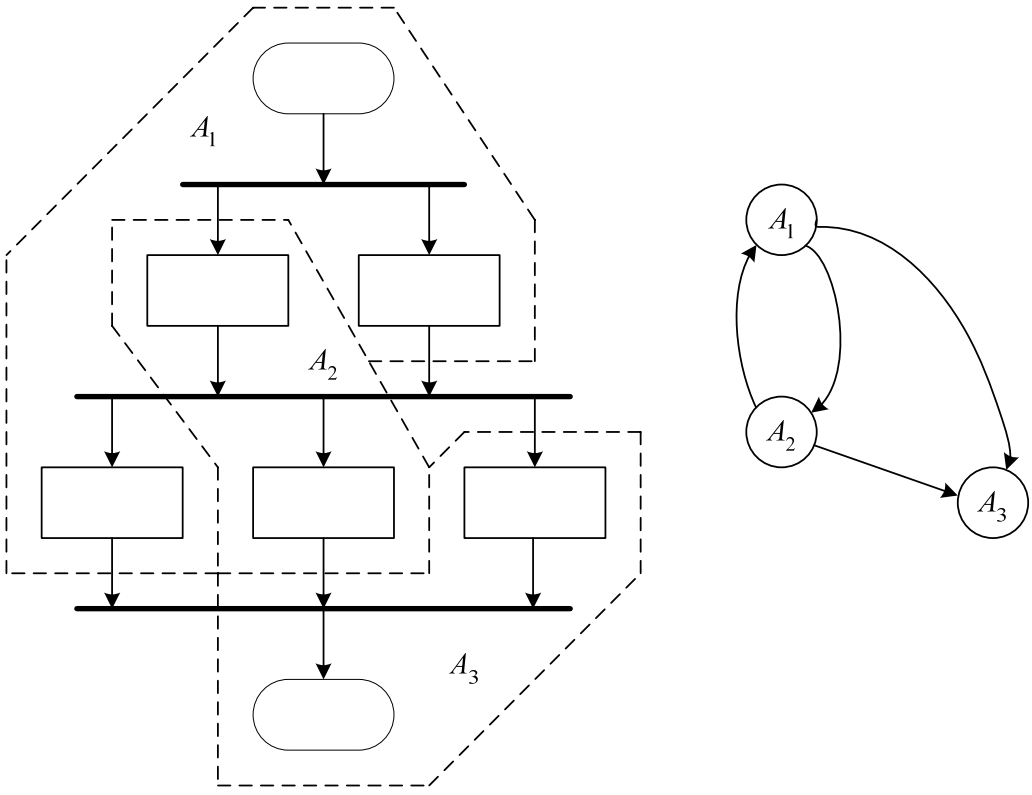

Рис. 16. Разбиение алгоритма и граф межблочных взаимодействий

Для определения значений коэффициентов введем понятие несовместимых дуг. С содержательной точки зрения несовместимыми дугами называются дуги,

которые могут быть реализованы только в различные периоды выполнения алгоритма управления. Для определения несовместимых дуг необходимо выполнение следующих условий:

- дуги должны находиться в различных ветвях альтернативного ветвления;
- дуги не должны входить в состав одного и того же цикла.

Для определения весов дуг графа межблочных взаимодействий прежде всего необходимо для каждой пары блоков A_i и A_j сформировать множество дуг $D\left(A_i, A_j\right) \subset U^0$, связывающих блоки, после чего выделить из множества дуг все максимальные по включению подмножества совместимых дуг $\left\{D^1_{\text{совм}}, D^2_{\text{совм}}, ..., D^N_{\text{совм}}\right\}$ и определить для них значения $\alpha'_k\left(D^k_{\text{совм}}\right) = \sum\limits_{v_l \in D^k_{\text{совм}}} \beta\left(v_l\right)$, $\tilde{\alpha}'_k\left(D^k_{\text{совм}}\right) = \sum\limits_{v_l \in D^k_{\text{совм}}} \delta\left(v_l\right)$. Тогда коэффициенты дуг графа межблочных взаимодействий могут быть определены как $\alpha_{ij}\left(A_i, A_j\right) = \max\limits_{k=\overline{1,N}}\left(a'_k\right)$, $\tilde{\alpha}_{ij}\left(A_i, A_j\right) = \max\limits_{k=\overline{1,N}}\left(\tilde{a}'_k\right)$.

После вычисления весов дуг графа межблочных взаимодействий можно определить значение приращений ΔZ_1 и ΔZ_2:

$$\Delta Z_1\left(\rho_i, A_j\right) = \sum_{h=\overline{1,H},\, h\neq j}^{H} \left[\chi\left(\alpha\left(\rho_i \cup A_j, A_h\right) - a_{jh}\right) + \chi\left(\alpha\left(A_h, \rho_i \cup A_j\right) - a_{hj}\right)\right];$$

$$\chi\left(f\right) = \begin{cases} f, & f > 0, \\ 0, & f \le 0; \end{cases}$$

$$\Delta Z_2\left(\rho_i, A_j\right) = \sum_{h=\overline{1,H},\, h\neq j}^{H} \left[\tilde{\alpha}\left(\rho_i \cup A_j, A_h\right) + \tilde{\alpha}\left(A_h, \rho_i \cup A_j\right)\right].$$

Приращение алгоритмов по сложности определяется как

$$\Delta W\left(\rho_i, A_j\right) = W\left(\rho_i \cup A_j\right) - \min_{k=\overline{1,H},\, k\neq j}\left(W\left(A_k\right)\right).$$

Анализ таблицы включений заключается в выборе такого соответствия $\Psi\left(\Omega_n\right) = \begin{bmatrix} \rho_1 & \rho_2 & ... & \rho_N \\ A_1 & A_2 & ... & A_M \end{bmatrix}$, что $\sum\limits_{\left(\rho_i, A_j\right) \in \Psi} t\left(\rho_i, A_j\right) \to \max$. Поиск оптимального соответствия проводится путем последовательного просмотра таблицы включений и выбора совокупности оптимальных включений. Включение $t\left(\rho_i, A_j\right)^+$ считается оптимальным, если оно не является недопустимым ($t\left(\rho_i, A_j\right) \notin \{"+","-"\}$) и $t\left(\rho_i, A_j\right)^+ \ge t\left(\rho_i, A_k\right)$, $k = \overline{1, M}$. Не исключен случай, когда оптимальными могут быть одновременно включения $\left(\rho_i, A_j\right)^+$ и $\left(\rho_k, A_j\right)^+$, $k \neq i$, т.е. некоторые подмножества целесообразно включить в один и тот же блок. В этом случае имеет место конфликт по включению, и для выбора оптимального включения необходим дополнительный критерий:

$$\varsigma\left(\rho_i, A_j\right) = t\left(\rho_i, A_j\right) - \max_{k\neq j}\left(t\left(\rho_i, A_k\right) \notin \{"+","-"\}\right).$$

В случае совпадения значений критерия $\varsigma\left(\rho_i, A_k\right) = \varsigma\left(\rho_j, A_k\right)$ в блок включается любое из подмножеств.

Если какое-либо субсечение невозможно включить ни в один из блоков разбиения, то оно образует собой новый блок.

Процесс размещения сечений в блоках разбиения повторяется последовательно для всех сечений, в результате чего формируется искомое разбиение алгоритма.

5. Программная реализация метода

Рассмотренный метод формирования субоптимальных разбиений направлен на автоматизацию процесса получения разбиений. Поэтому актуальна задача разработки реализующей его программной системы. С ее помощью должно быть возможно не только формирование разбиений, но и сравнение качества полученных разбиений с другими методами [3,6]. Ввод и вывод информации при этом должен осуществляться в удобной для пользователя текстовой и/или графической форме.

В настоящее время разработан прототип программной системы, реализующей все этапы рассматриваемого метода. Рис. 17 показывает фрагмент кода системы в стадии разработки в среде Delphi 7.

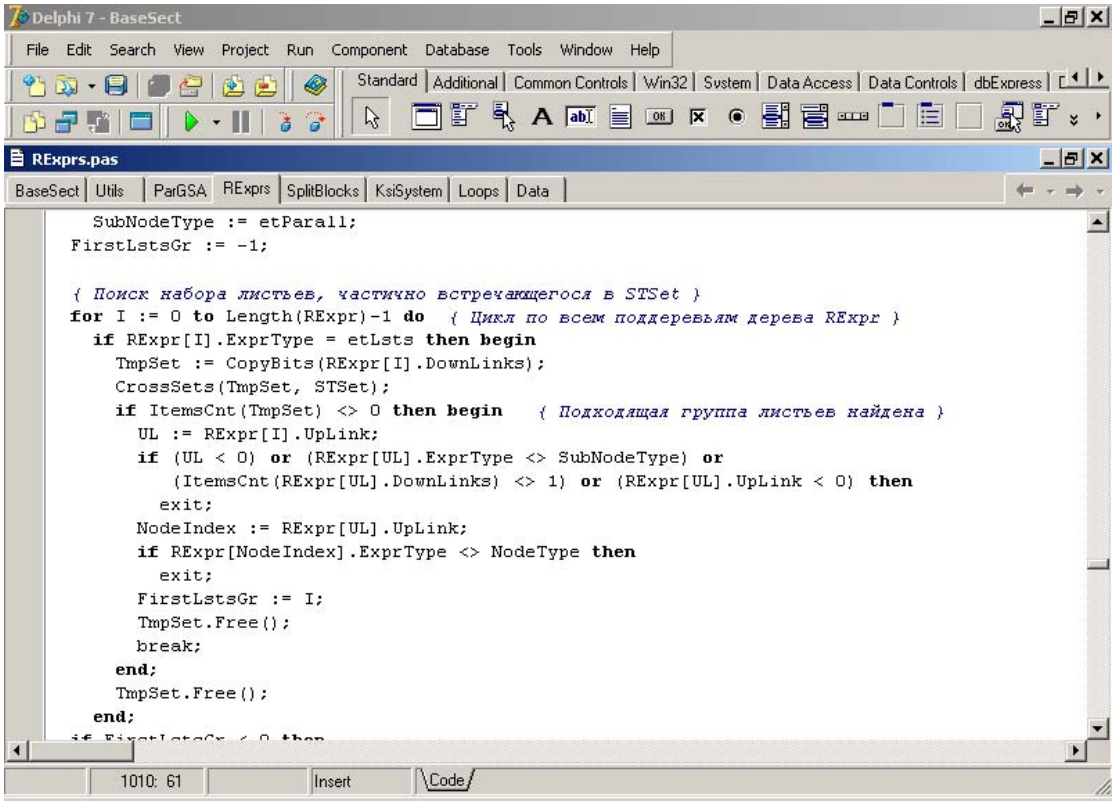

Рис. 17. Разработка программы построение разбиения

На данный момент программа представляет собой отлаженный рабочий вариант, оформленный в виде в виде консольного приложения и находящийся в стадии заключительного тестирования. Обмен данными с программой производится при помощи текстовых файлов стандартного формата (рис. 18).

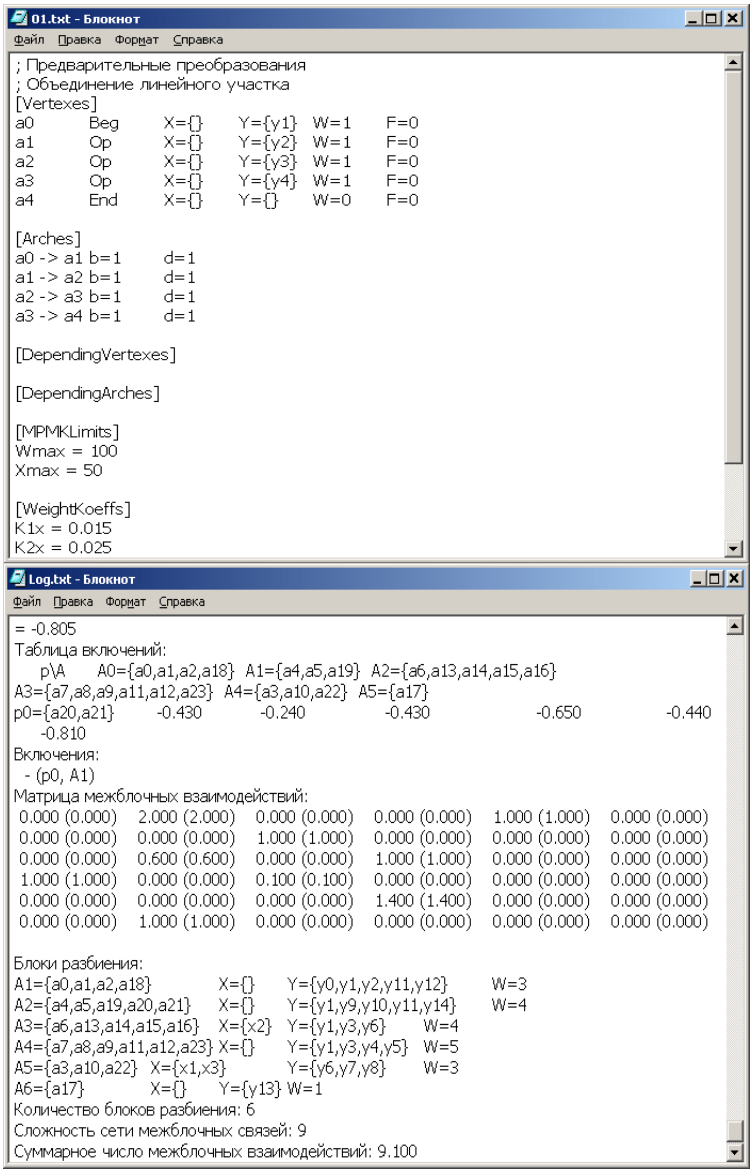

Рис. 18. Примеры входного и выходного файлов программной системы

Ввод больших алгоритмов управления в текстовом виде затруднителен, поэтому в настоящее время в стадии разработки находится графическая среда (рис. 19), которая позволит упростить процесс ввода исходных данных и анализ результатов.

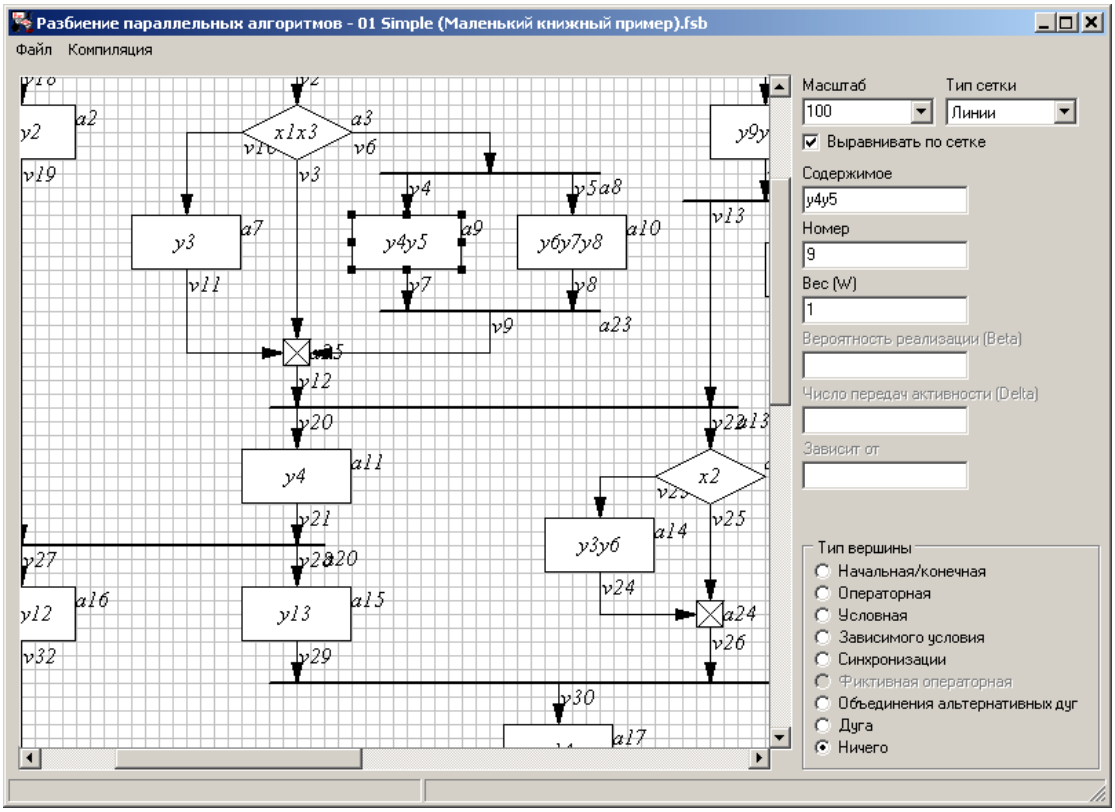

Рис. 19. Графическая оболочка для ввода алгоритмов управления

6. Заключение

Представленный в работе метод обеспечивает глубокую формализацию процесса получения разбиений параллельных алгоритмов управления, что позволило построить на его основе программную систему. В плане дальнейших исследований авторов экспериментальная оценка степени близости качества разбиений к оптимальному и анализ ее зависимости от весовых коэффициентов, определяющих значимость оптимизируемых показателей.

Работа выполнена при финансовой поддержке гранта Минобразования «Столетовские гранты – 2003».

Список литературы

1. Зотов И.В., Колосков В.А., Титов В.С. Выбор оптимальных разбиений алгоритмов при проектировании микроконтроллерных сетей // Автоматика и вычислительная техника. 1997. №5. С. 51–62.
2. Организация и синтез микропрограммных мультимикроконтроллеров / Зотов И.В. и др. Курск: ГУИПП «Курск», 1999. 368 с.
3. Баранов С.И., Журавина Л.Н., Песчанский В.А. Метод представления параллельных граф-схем алгоритмов совокупностями последовательных граф-схем // Автоматика и вычислительная техника. 1984. №5. С. 74–81.
4. Поиск базового сечения в задаче разбиения параллельных алгоритмов / Ватутин Э.И., Зотов И.В.; КГТУ. Курск, 2003. 30 с. деп. в ВИНИТИ 24.11.03 № 2036-В2003.

5. Ватутин Э.И., Зотов И.В., Титов В.С. Построение множества сечений в задаче оптимального разбиения параллельных управляющих алгоритмов // Известия ТулГУ. Вычислительная техника. Информационные технологии. Системы управления. 2003. Т. 1, Вып. 2. С. 70–77
6. Закревский А.Д. Параллельные алгоритмы логического управления. Минск. ИТК НАН Б. 1999. 202 с.