Моделирование структурированных дискретно-событийных систем сетями Петри

Александр А. Амбарцумян

Институт проблем управления РАН

Москва, Россия; (тел.: +7-495-334-87-89, e-mail: ambar@ipu.ru)

В работе излагается методология дискретно-событийного моделирования объекта и его требуемого поведения (спецификаций) при проектировании систем автоматизации, работающих в реальном времени. Методология основана на использовании для анализа функциональности и согласованности модели, названной структурированная дискретно-событийная система (СДС2). Определен состав модели СДС2, предложена техника анализа управляемости СДС2, разработан метод синтеза по СДС2-модели сети Петри, моделирующей процессы в объекте, разработаны техника анализа сети процесса и метод синтеза управляющей сети супервизора, обеспечивающей совместно с сетью процесса выполнение исходных спецификаций[1].

1.Введение. Постановка задачи.

Практически востребованные задачи анализа, проектирования и программирования систем автоматизации сложных технических систем мотивируют развитие логических исследований управления дискретными процессами, поведение которых определяется на множестве дискретных состояний (не обязательно конечном) и на потоке событий. Общее название этого направления, утвердившееся в зарубежной литературе последние 20 лет – динамические дискретно событийные системы ДДСС или чаще сокращенно дискретно-событийные системы ДСС (в англоязычном варианте: discrete event dynamic systems DEDS чаще упрощенно DES). В работе [1] первооткрывателей направления У. Уонхэм и Дж. Рамадж (Wonham, W. M и Ramadge, J. G), изложены основные идеи направления, обзор состояния и систематическое изложение можно найти в монографии [2].

ДСС это триплет $< G, K, S >$, где G это объект, K требования к его поведению, а S это супервизор - управляющая компонента ДСС, обеспечивающая поведение G, в соответствии с ограничениями (спецификацией) K. При этом S должен быть неблокирующим – не содержать тупиков и «живых циклов» (ловушек). Функционирование ДСС характеризуется множеством событий E и генерируемыми последовательностями s из этих событий $e \in E$. Множество генерируемых последовательностей называют языкам (например, язык генерируемый объектом G

[1] Работа выполнена при частичной поддержке российского фонда фундаментальных исследований (проект № 09-08-00559-а).

обозначается $L(G)$). В теории получены фундаментальные результаты, по основным этапам проектирования ДСС:

- анализу исходного представления поведения как ничем не ограниченного генератора языка - $L(G)$;
- специфицированию ограничений на требуемое поведение в форме языка $K \subseteq L(G)$ или в терминах недопустимых состояний (линейных ограничений на соотношение активности в позициях для сетей Петри);
- исследованию управляемости и разработке методов синтеза неблокирующего супервизора S.

Полученные результаты открывают новые возможности в проектировании логического управления на основе дискретно-событийного моделирования. Постановки вопроса о существовании и задачи синтеза супервизора дают аналитический ответ на основной вопрос логического управления: при заданных объекте и спецификации требований разрешима ли задача управления и если да, то каким образом. Более того, если ответ отрицательный, то исследован вопрос: при каких изменениях спецификаций (ограничений на поведение объекта) управление разрешимо. Потенциально открывшиеся возможности составляют прорыв в логическом управлении в части повышения его достоверности.

Вместе с тем практика реального проектирования САРВ, опирающаяся на инструментальные средств (например, на SCADA системы), не использует эти подходы.

Две основные трудности в практическом использовании дискретно-событийного моделирования (ДСС-моделирования).

Первая заключается в высокой размерности модели объекта управления - G (для реальных задач число состояний конечного автомата порядка несколько тысяч). Трудности с размерностью исходных данных усугубляется мультипликативной зависимостью сложности синтеза супервизора, от сложности объекта G и от сложности спецификации K (как правило, наблюдается «взрыв состояний»). Для преодоления этих затруднений разработаны методы модульного синтеза супервизора по G как единому целому [4 - 7]. Однако, сложность модульного синтеза и слабое соответствие структуры исходных спецификаций результату, делают предложенные методы пока малопривлекательными для практического применения. Сохранение исходной модульности для отдельных классов ДСС достаточно эффективно обеспечивается использованием сетей Петри (СП) как аппарата описания и синтеза супервизорного управления в ДСС (работы [8, 9, 10] и многие другие).

Вторая трудность – отсутствие структурного подобия синтезированного супервизора и исходных описаний объекта управления. Это вызвано тем, что основные результаты и методы теории ДСС сформулированы в понятийном базисе языков над множеством событий, в то время как в реальном проектировании используется построения, использующие в явном виде понятия состояния, условия, переходы, структура и т.п. Эти понятия лежат в основе принятых способов описания функционирования объекта и его компонент: диаграммы переходов, тактограммы, циклограммы, структурные схемы. Сложившийся понятийный базис нашел отражение в международных нормативных документах на проектирование и программирование систем промышленной автоматизации (см, например, Стандарта IEC 61131-3 для программируемых промышленных систем управления).

В настоящей работе ставиться задача: в рамках ДСС разработать методику анализа и проектирования супервизора, преодолевающую сформулированные выше трудности и ориентированную на сохранение структурного подобия результатов проектирования и исходных описаний поведения объекта управления. Предлагается данные о ДСС последовательно представлять в моделях: набор конечных автоматов (этап описания данных о структуре объекта), строки команд (событий) на исполнение операций оборудованием (этап формирование требований технологов на последовательности операций); структура взаимодействия оборудования при выполнении команд и операций – сети Петри (этап системного аналитика и реализации системы). Алгоритмы, составляющие методику должны обеспечивать преобразование моделей, так что бы исходная модульность задания ДСС сохранилась в результирующей управляемой сети Петри, а все преобразования не приводили к эффекту «взрыва состояний». С этой целью исходные структуры данных алгоритмами методики будут последовательно синтаксически преобразовываться в решения по управлению, ограничивающие поведение объекта не более, чем того требуют исходные спецификации (ограничения).

Структура работы. Во втором разделе мы приводим базовые понятия и результаты, которые используются в предлагаемой методике. В третьем разделе излагается методика структурированного дискретно-событийного моделирования (СДС2-моделирования), в четвертом разделе предлагаются: метод преобразования автоматной модели G в S_p - сеть Петри моделирующей процесс (в объекте G); излагаются результаты исследования особенностей сетей Петри, реализующей процессы в объекте; предлагается базовый метод синтеза супервизора S_c - управляющей сети Петри и приводится иллюстративный пример. В пятом разделе излагается метод синтеза сети Петри, реализующей супервизор

без ограничения на количество и тип входо-выходных последовательностей. В заключении кратко характеризуется вклад данной работы в проблему проектирования супервизорного управления для систем автоматизации.

2. Базовые понятия. Результаты, на которые опирается данная работа.

Настоящая работа основывается на модели ДСС, использующей разделение множества событий E на Euc и Ec (множества управляемых и неуправляемых событий - традиционное для теории DES разбиение) и Ew - множество ожидаемых событий. Эта модель предложена в работе [11]. События из Ew моделируют состояния (положение) исполнительных механизмов (актуаторов) или компонент объекта. Эти события нельзя блокировать супервизором как управляемые события из Ec. Однако их появление ожидаемо, как отклик на события из Ec, подтверждающий факт выполнения команд на актуаторы. Учет этих особенностей позволяет получить ряд преимуществ как в определении ДСС, так и формулировании условий управляемости и синтезе супервизора.

ДСС, с перечисленным структурированием, назовем *ДСС с углубленным разделением событий*. Ограничение свободного поведения $G = \langle G^1, G^2, \dots, G^n \rangle$ с целью выполнения K осуществляет супервизор S , встроенный в контур обратной связи в ДСС .Поведение G под контролем S обозначается - S/G, а соответствующий язык $L(S/G)$.

Поведение каждой компоненты $G = \langle G^1, G^2, \dots, G^n \rangle$ это язык $L(G^i)$, генерируемый конечным автоматом $G^i = \left\langle Q^i, E^i, \delta_i, \Gamma^i, Q_m^i, q_0^i \right\rangle$, определяемым множествами состояний, событий, функциями переходов и допустимых событий, множеством обязательно достижимых состояний и начальным состоянием соответственно.

Система, содержащая супервизор S так, что выполняется K, состав неуправляемой части которой задан ДСС с углубленным разделением событий, а требуемое поведение определенно языком спецификаций $K \subseteq E_d^*$, $(K \neq \varnothing)$ в работе [11] названа ***структурированной дискретно событийной системой (СДС2)***. (E_d^* - множество всевозможных слов из $e \in E_d = \{E_c \cup E_{uc}\}$ любой, но конечной длины).

Выполнение K предполагает, что проекция $P_{Ed}(L(S/G)) = K$, т.е. K будет совпадать с проекцией по E_d языка $L(S/G)$, генерируемого S/G - объектом G под контролем супервизора S. В работе [10] сформулированы свойства СДСС и разработана процедура, названная алгоритм эксперимента $\Re(s)$, которая является основным приемом

исследования совместного поведения *G* и *H*. При этом предполагается, что *G* обладает собственным контроллером, обеспечивающим генерацию управляющих событий, а супервизор лишь обеспечивает блокирование тех управляющих событий, появление которых противоречит спецификации. Для моделирования ДСС с пассивными актуаторами (а САРВ именно таковыми и являются) в теории ДСС используется модель с форсируемыми управляемыми событиями, в которой управляемые события генерирует (форсирует) супервизор [12]. Существование супервизора для структурированных ДСС с форсируемыми событиями исследованы в работе [13].

Первая часть СДСС-моделирования – анализ существования неблокирующего супервизора *S*, основывается на проверке сформулированных в работе [11] условий существования *S* и определении $Q=\{Q_1, Q_2, \ldots Q_r\}$- множество достижимых векторов-состояний компонент $G=\langle G^1, G^2, ..., G^n \rangle$.

С целью сохранения исходного структурирования ДСС в результате проектирования, синтез супервизора в предлагаемой методике, осуществляется на сетях Петри, при этом для последних используются общепринятые обозначения (см., на пример, [2], но в принципе, эти обозначения согласуются с более ранними из [17]) . Сеть Петри определяется через структуру и разметку. Структура это четверка $S = (P, T, Pre, Post)$, где *P* – множество из *m* позиций, *S* – множество из *n* переходов (множества конечны и не пусты). $Pre: P \times T \to N$ и $Post: P \times T \to N$ - входная и выходная функции инцидентности переходов, специфицируют дуги и их вес (кратность) в структуре двудольного графа *S*. Функции могут определяться в матричной форме и тогда $W = Post - Pre$ – матрица инцидентности (изменений) размерности $(m \times n)$. По функциям инцидентности определяются для всякого перехода t_i множества его входных $pre(t_i)$ и выходных $post(t_i)$ элементов. Аналогично для позиций. Вектор $\mu: P \to N$ – вектор разметки сети, сопоставляет каждой позиции целое положительное число (называют метки, фишки). Обозначают $\mu(p_j)$ количество меток в позиции p_j. Если $\mu(p_i) = k \in \mathrm{N}$, то говорят, что в позиции $p_i \in P$ имеется k фишек. Переход t_i допустим для S на разметке μ_j $(S, \mu_j) \Leftrightarrow \forall p \in pre(t_i): \mu(p) \geq Pre(p, t)$. Разметка является фундаментальным состоянием сети S, при этом, μ_0 - начальная разметка сети S . Сеть Петри (СП) это пара (S, μ_0). Динамика состояний (достижимых разметок) СП определяется срабатыванием (запуском) переходов. Переход t_j срабатывает, если он допустим в μ_i текущей разметке *S* и при срабатывании количество меток во всех его входных позициях, в соответствии с функцией *Pre* - уменьшается, а в выходных позициях, в соответствии с *Post* - увеличивается.

Заметим, что w_{ij} - элемент матрицы W показывает: какие изменения в позиции p_i вносит при срабатывании переход t_j. Пусть $y(t_j)$ вектор размерности m, у которого в позиции $y(j)$=1, а в остальных 0. Тогда выражением $\mu_k = \mu_i + Wy^T(t_i)$ в матричной форме представляет изменения маркировки сети, вызванные срабатыванием перехода t_j. По последовательности срабатывания переходов s:=$t_1,t_2,\ldots t_k$ построим вектор $y(m)$ (размерности m), значение которого в позиции $y(i)$ равно числу срабатываний перехода $y(i)$ в последовательности s. Такой вектор y называют вектор срабатывания или вектор Париха. Тогда в векторной форме маркировка μ_k, достижимая из μ_0 после последовательности s, определяется следующим образом $\mu_k = \mu_0 + Wy^T$. Часто используют способ задания сети Петри с матрицей инцидентности (различий) $S = (P, T, W, \mu_0)$. При моделировании ДСС переходы взвешиваются именами *событий*.

Следующие два понятия относятся к свойствам структуры сети Петри.

<u>*P*- инвариантом</u> (*инвариантом позиций*) называют целочисленный вектор $X = \{x_i\}, i = 1,2,\ldots n$, такой, что $x_i \in \{0,1\}$, являющийся решением линейной системы: $XW = 0$. Вектор X единичными значениями характеризует подмножество позиций, в которых при любой достижимой маркировке число фишек постоянно.

<u>*T*-инвариантом</u> (*инвариантом переходов*) называется целочисленный неотрицательный вектор $Y = \{y_i\}, i = 1,2,\ldots m$. если он является решением линейной системы: $WY^T = 0$. Если для сети S существует T-инвариант Y, то для сети допустима хотя бы одна последовательность срабатывания s, для которой Y является вектором Париха. T-инвариант определяет наличие циклов и подциклов в сети Петри.

В ДСС-моделировании сеть Петри, моделирующая G называют S_p- сетью процесса, а компоненту сети, моделирующую супервизор S называют управляющей сетью – S_c – (сетью контроллера).

3. Методика СДС2-моделирования объекта управления

При моделировании ДСС на сетях Петри, характеризующихся очень бедным набором базовых средств (позиция и переход), возникают объективные методологические трудности. Одна из них: как технологическую операцию представить (смоделировать) позициями и переходами. Базовым понятием технологических данных является технологическая операция, основной управляемой компонентой структуры оборудования

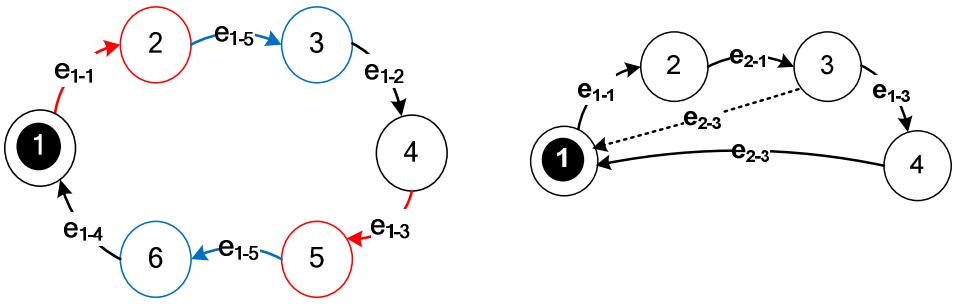
чаще всего является исполнительный механизм (актуатор), а различного рода диаграммы, циклограммы и/или блок-схемы описывают требуемое (допустимое, востребованное) поведение – порядок исполнения операций. Технологическая операция характеризуется событием (командой на актуатор), изменением состояния объекта и соответствующими откликами (реагированием). На наш взгляд целесообразно на этапе первичной формализации использовать модель конечный автомат, поскольку состояния, условия (события) их завершения и функции переходов и выходов достаточно прозрачно корреспондируются с упомянутыми выше традиционными для технологов инструментами (диаграммы, циклограммы и/или блок-схемы), а использование конечно-автоматных состояний наиболее близко структуре исходных сведений.

Исходя из вышесказанного, на первом этапе проектирования супервизора для ДСС в работе предлагается провести СДС2-моделирование объекта управления в соответствии с основными положениями работы [10]. СДС2-моделированием будем называть структурирование объекта управления в виде набора конечных автоматов и определение их совместное поведения как требование выполнить последовательность команд на эти автоматы.

3.1. Моделирование структуры объекта и функциональности компонент как ДСС, представленной набором конечных автоматов

Структуризация объекта начинается с выделения всех актуаторов (приводов и исполнительных механизмов); пусть это будет $G = \langle G^1, G^2, \dots, G^n \rangle$. Для каждого G^i определяются: множество событий, являющихся командами E_c^i, множество событий E_w^i, характеризующих траекторию движения этого конкретного механизма (множество реакций на E_c^i) и, наконец, определяется множество условий, внешних относительно G^i – E_{uc}. Общий алфавит для G^i – $E^i = \{E_w^i \cup E_c^i \cup E_{uc}^i\}$. Взаимосвязь команд ($E_c^i$), ожидаемых реакций и внешних условий (если это требуется), на практике представляется в виде диаграмм (циклограмм), что достаточно прозрачно формализуется конечным автоматом $G^i = \left\langle Q^i, E^i, \delta_i, \Gamma^i, Q_m^i, q_0^i \right\rangle$. Здесь и далее такие автоматы будем называть компонентными конечными автоматами (ККА) В итоге анализа объекта определяются: состав СДС2 $G = \langle G^1, G^2, \dots, G^n \rangle$, множества E^i событий, каждое из которых структурируется на $E^i = \{E^i{}_w \cup E^i{}_c \cup E^i{}_{uc}\}$, и множество E_{uc}^i – общих неуправляемых событий. Поясним сказанное на примере устройства перекладчика, состоящего из двух простых однотипных механизмов. Граф переходов ККА модели одного механизма представлен на рис. 1 а).

События e_{1-1}, e_{1-3} соответствуют командам на механизм, а события e_{14}, e_{15}, e_{12} положениям механизма (левому, промежуточному и правому соответственно). Механизм выполняет циклическую последовательность операций. Второй ККА, совпадающий с первым в точности до обозначений событий (индексы начинается с 2), опущен.



а) б)

Рисунок 1. а) ККА - модель механизма G^1; б) автомат спецификации H

Следующим этапом СДС2-моделирования является определение востребованного поведения набора компонент. Поведение представляется как последовательность команд из $E_c = \cup_1^n E_c^i$, заданной строками в языке $K \in E_d^*$. При этом все последовательности команд, которые должны быть отработаны ККА, представляются всевозможными простыми путями графа переходов конечного автомата $H = (Q^h, E_d, \delta_h, \Gamma^h, Q_m^h, q_0)$. Пусть в рассматриваем примере требуется чтобы механизмы (обозначим их G^1 G^2) последовательно выполняли операции: вначале первые - e_{1-1} , e_{2-1}, а затем вторые - e_{1-3} , e_{2-3} и эта последовательность циклически повторялась. Спецификация сформулированного совместного поведения 2-х механизмов представлена на рис. 1 б) графом переходов (пунктирную стрелку не учитывать).

Завершающим этапом СДС2-моделированрия является проверка управляемости на основе алгоритма эксперимента $\mathfrak{R}(s)$, который является основным инструментом при изучении совместного поведения G и H. Результатом эксперимента $\mathfrak{R}$ над строками s ∈ K (все строки допустимы для начального состояния q_0) может быть $\mathfrak{R}$(s) = True | False. Если после всех действий над символами s все компоненты G успешно осуществили свои переходы, то $\mathfrak{R}$(s) = True; если во время эксперимента новый символ $\boldsymbol{s(i)}$ не «сработал» в соответствующей компоненте, тогда $\mathfrak{R}(s)$ = *False*. В последнем случае подразумевается , что $\boldsymbol{H}$ не согласован с $\boldsymbol{G}$ и спецификация требуемого поведения (автомат $H = (Q^h, E_d, \delta_h, \Gamma^h, Q_m^h, q_0)$) должна быть изменена. В примере (рис. 1.б) заявленная

единственная последовательность допустима, но если пропустить одну хотя бы одну команду (пунктирная стрелка как раз демонстрирует такую последовательность), то эта последовательность покажет $\mathfrak{R}(s) = False$ при эксперименте и в целом такая спецификация не реализуема. В завершение анализа определим $Qvs=\{Q_1, Q_2, \ldots Q_r\}$- множество достижимых векторов-состояний компонент $G = \langle G^1, G^2, \ldots, G^n \rangle$, где $Q_j := \left\langle q^1_{j1}, q^2_{j2}, \ldots, q^n_{jn} \right\rangle$. Для рассматриваемого примера перекладчика $Qvs = \{q^1_1 q^2_1, q^1_2 q^2_1, q^1_1 q^2_2, q^1_1 q^2_2\}$.

3.2. Моделирование компонентных конечных автоматов логическими схемами последовательности строк

Синтез супервизора основан на анализ допустимых строк в языках $L(G^i)$ и $K \subseteq E_d^*$, заданных компонентными конечными автоматами $G = \langle G^1, G^2, \ldots, G^n \rangle$ и конечным автоматом $H = (Q^h, E_d, \delta_h, \Gamma^h, Q^h_m, q_0)$, определяющим спецификацию. Специфика промышленных объектов такова, что логика отдельных компонент объекта довольно прозрачна. Графы переходов, задающие автоматы G_i и H слабоветвящиеся, как правило, задают ограниченное количество циклически выполняемых последовательностей. Поэтому сами строки и их последовательности достаточно просто извлекаются из анализа последовательности ребер графа переходов. Такие последовательности довольно просто описываются на языке ЛСА - логических схем алгоритмов хорошо изученном в работах Лазарева В.Г. и Баранова С.И. [14, 15,16]. В данной работе мы используем упрощенный вариант ЛСА для описания последовательности строк событий только из соображений упрощения изложения метода синтеза сети Петри и простоты приведения иллюстративного материала. Напомним ([14, 15,16]), ЛСА это последовательность символов и исходящих $\uparrow^n$ и входящих $\downarrow^n$ стрелок с индексами, организующих различные последовательности выполнения строк. В СДСС-моделировании под логической структурой последовательности строк (ЛСПС) будем понимать представление графа переходов конечного автомата в виде строк символов состояний и событий этого графа.

Правила преобразования графа переходов в ЛСПС и обратно довольно просты и аналогичны описанным в [14,15] правилам перехода от граф-схем алгоритмов к ЛСА. Поэтому ограничимся их изложением на концептуальном уровне. Каждая строка ЛСПС соответствует простому пути графа переходов и составляется из имен состояний и событий, взвешивающих ребра, в порядке их следования в этом пути. Следование строк соответствует порядку обхода простых путей в графе переходов и представляется

стрелочками с индексами. Стрелочка вверх с индексом $\uparrow^n$ – исходящее ребро, заканчивающее простой путь (после события, взвешивающего это ребро), стрелочка вниз с индексом $\downarrow^n$ – входящее ребро (перед состоянием, в которое входит помеченное раннее стрелкой исходящее ребро). Индексы стрелок однозначно связывают начало и конец перехода между строками. Если после некоторого состояния имеет место ветвление (исходящих ребер более одного), то ветвление заканчивает строку, а события, соответствующие этим ребрам, с верхними стрелочками заключается в *фигурные* скобки. Список событий с верхними стрелочками в фигурных скобках будем называть ветвитель. На рис. 2 представлен граф переходов механизма G3, стрелочками помечены ребра графа, разрыв которых выделяет простые пути для построения строк ЛСПС. Для графа переходов G^3 ЛСПС: $\downarrow^1 q_1\ e_{3\text{-}1}\ q_2\ e_{3\text{-}}2\ q_3\ e_{3\text{-}3}\ \downarrow^4 q_4\{e_{3\text{-}8}\ \uparrow^2,\ e_{3\text{-}4}\ \uparrow^3\};\ \downarrow^2 q_{10}\ e_{3\text{-}2}\ q_{11}\ e_{3\text{-}9}\ \uparrow^1;\ \downarrow^3 q_5\ e_{3\text{-}5}\ q_6\ e_{3\text{-}6}\ q_7\ e_{3\text{-}7}\ q_8\ e_{3\text{-}5}\ q_9\ e_{3\text{-}3}\ \uparrow^4$.

Легко прослеживается связь строки ЛСПС и пути на графе: путь q1 → q4 это первая строка, заканчивающаяся фигурными скобками; остальные фрагменты графа представлены строками, стрелки которых указывают связь с первой строкой.

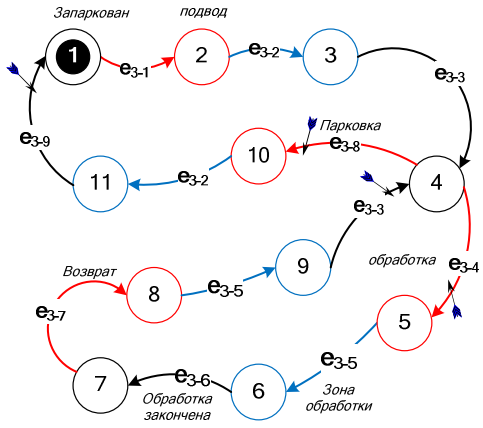


Рисунок 2 Граф переходов автомата G^3.

Иллюстрируем изложенные правила примером из раздела 3.1 2-х механизмов G^1 и G^2. Для G^1 (см. рис. 1 а) ЛСПС: $\downarrow^1 q_1\ e_{1\text{-}1}\ q_2\ e_{1\text{-}5}\ q_3\ e_{1\text{-}2}\ q_4\ e_{1\text{-}3}\ q_5\ e_{1\text{-}5}\ q_6\ e_{1\text{-}4}\ \uparrow^1$. ЛСПС G^2 имеет вид: $\downarrow^1 q_1\ e_{2\text{-}1}\ q_2\ e_{2\text{-}5}\ q_3\ e_{2\text{-}2}\ q_4\ e_{2\text{-}3}\ q_5\ e_{2\text{-}5}\ q_6\ e_{2\text{-}4}\ \uparrow^1$. Построение ЛСПС по графу спецификации аналогично. Напомним, что в автомате H не используются ожидаемые события. Тогда ЛСПС LS^H автомата H, представленного на рис. 1 б) имеет следующий вид: $\downarrow^1 q_1\ e_{1\text{-}1}\ q_2\ e_{2\text{-}1}\ q_3\ e_{1\text{-}3}\ q_4\ e_{2\text{-}3}\ \uparrow^1$.

3.3. Редукция логической структуры последовательности строк

Моделирование объекта как СДС2 включает все события, представляющие детально его поведение. Каждая операция механизма представляется строкой, начинающейся с управляемого события – команды, и последовательности всех последующих ожидаемых событий. Так, например, для G^1 операция «сдвиг детали» представляется строкой: $e_{1\text{-}1}\ q_2\ e_{1\text{-}5}\ q_3\ e_{1\text{-}2}\ q_4$. При моделировании компонент СДС2 $G = \langle G^1, G^2, \dots, G^n \rangle$ сетями Петри события E будем представлять переходами, а позиции сопоставлять состояниям процесса, соответствующим операциям. Поэтому при моделировании операции сетью Петри достаточно после события - команды $e_{1\text{-}1}$ перейти в позицию выполнения операции, а по событию $e_{1\text{-}2}$ (последнему из строки ожидаемых событий) перейти в позицию, соответствующую завершению операции. Таким образом все состояния ККА, соответствующие выполнению операции (в нашем примере это q_2 , q_3) моделируются одной позицией сети и в строке их достаточно представить последним состоянием ККА и событием, соответствующем завершению операции. Это преобразование определим в виде правила редукции строк ЛСПС. первым после события- команды

Правило редукции строк ЛСПС.

Подстрока из 4-х символов типа: <состояние> i_1, < ожидаемое событие> j_1, <состояние> i_2, < ожидаемое событие> j_2, в которой все события относятся к типу ***ожидаемых***

редуцируется в подстроку: <состояние> i_2, <ожидаемое событие> j_2 .

Применим правило редукции к ЛСПС G^1: $\downarrow^1 q_1\ e_{1\text{-}1}\ q_2\ e_{1\text{-}5}\ q_3\ e_{1\text{-}2}\ q_4\ e_{1\text{-}3}\ q_5\ e_{1\text{-}5}\ q_6\ e_{1\text{-}4}\ \uparrow^1$.

Подстрока из 4 символов, например $q_2\ e_{1\text{-}5}\ q_3\ e_{1\text{-}2} \Rightarrow \cancel{q_2\ e_{1\text{-}5}}\ q_3\ e_{1\text{-}2} \Rightarrow q_3\ e_{1\text{-}2}$

В соответствии с правилом редукции строка ЛСПС G^1 последовательно преобразуется $\downarrow^1 q_1\ e_{1\text{-}1}\ q_2\ e_{1\text{-}5}\ q_3\ e_{1\text{-}2}\ q_4\ e_{1\text{-}3}\ q_5\ e_{1\text{-}5}\ q_6\ e_{1\text{-}4}\ \uparrow^1 \Rightarrow \downarrow^1 q_1\ e_{1\text{-}1}\ q_3\ e_{1\text{-}2}\ q_4\ e_{1\text{-}3}\ q_6\ e_{1\text{-}4}\ \uparrow^1$. Далее ЛСПС автомата G^i , для краткости, будем обозначать LS^i, а ЛСПС автомата H - LS^H.

4.Синтез сети Петри моделирующей СДС2.

Как отмечалось в разделе 1, исходная модульность для отдельных классов ДСС достаточно эффективно сохраняется при использовании сетей Петри (СП) как аппарата описания и синтеза супервизорного управления и исследовалась рядом авторов (работы [8, 9, 10] и многие другие). Метод, предложенный в работе [8] основан на использовании анализа запрещенных состояний и в общем случае, синтез сети супервизора (как дополнение сети процесса) решается методами целочисленного программирования. Методы в работах [9, 10] основываются на матричном представлении СП и используют

технику инвариантов позиций. Эти методы дают хорошие решения для управления ресурсным взаимодействием компонент ДСС (выполнение ограничений на количество одновременно активных компонент или задействованных ресурсов), но эта техника неэффективна при задании спецификации в виде порядка срабатывания всех переходов. При разработке САРВ характерно задание спецификации именно в виде требуемой последовательности операций актуаторов, при этом операции циклически повторяются, что соответствует Т-инварианту в результирующей сети Петри.

В результате СДС2-моделирования определены: множество ККА $LS = \{LS^i\}$ и спецификация LS^H. Поскольку все эти компоненты – конечные автоматы, можно применить известные подходы к переходу от КА к СП (см. например, [17]). Однако, модель ДСС2 определяет иерархию автоматов, с пересекающимися алфавитами событий. С позиции управления выполнение команд объекта управления САРВ, моделируется множеством ККА, представленных $LS = \{LS^i\}$, а порядок следования между событий, различных ККА определяется в спецификации LS^H. Появление события в LS^Hтребует одновременного его выполнения в соответствующем ККА.

В настоящей работе предлагается задачу синтеза СП, моделирующей СДС2, разделить на задачу синтеза сети процесса S_p по $LS = \{LS^i\}$(моделирование на СП технологического объекта) и задачу синтеза сети контроллера – S_c. При этом синтез сети процесса S_p выполним, как и в работе [17] сопоставлением состояниям - позиций, а триплету: состояние – событие – следующее состояние, переход в позицию, соответствующему следующему состоянию. При синтезе сети контроллера – S_c предлагается использовать спецификацию требуемого поведения LS^H как описание Т-инвариантов будущей управляемой сети, и исходя из этих предпосылок, разрабатывается метод конструирования супервизор по S_p и LS^H.

4.0. Механизм взаимодействия модели СДС2 на сети Петри и объекта управления.

Поскольку конечной целью моделирования является построение управляющей СП, необходимо определить ее взаимодействие с внешним миром. Предлагается использовать некоторое сужение механизма определенного в [17].

Механизм взаимодействия (МВ - схематично представлен на рис. 3) содержит:

- вектор входо-выходных позиций (ВВП), сопоставляемых всем переходам S_p и переходам типа $\theta(t) = uc$ из S_c ;

- регистр соответствия факта наличия события и двоичного сигнала $\{0|1\}$.

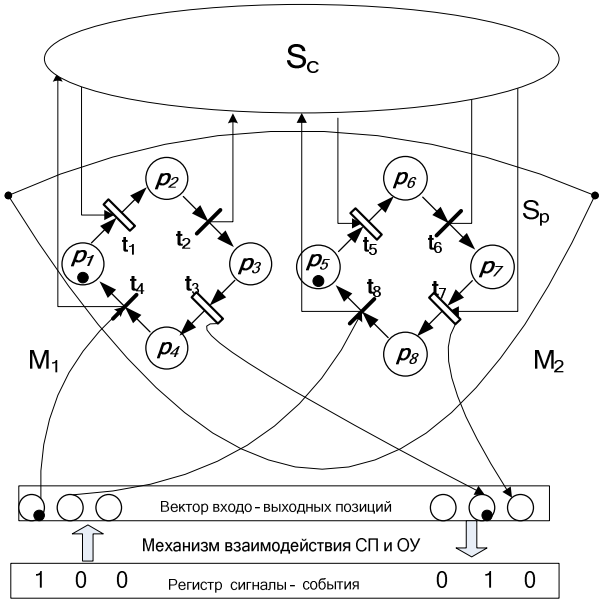


Рисунок 3. Механизм взаимодействия СП , ОУ и внешней среды.

Вектор и регистр одной размерности, кроме того позиции расположены в следующем порядке: слева сгруппированы все позиции сопоставленные неуправляемым и ожидаемым переходам, а справа – сопоставленные управляемым переходам. Поскольку в дальнейшем изложении МВ не участвует, мы ограничимся содержательным изложением его структуры связей с $\{S_p \cup S_c\}$ и функционирования. Все позиции вектора ВВП, соответствующие переходам типа *ис* и *w* включаются в их входные множества $(pre(t_i))$, а позиции, соответствующие управляемым переходам включаются в выходные множества $(post(t_i))$ этих переходов.

Сужение МВ заключается в ограничении порядка функционирования МВ в виде 3-х шагов.

1. На первом шаге, по факту наличия событий на объекте и во внешней среде (события типа *ис* и *w*), заполняется левая часть регистра, и проставляются метки в соответствующие позиции вектора входо-выходных позиций (ВВП).

2. На втором шаге просчитываются всевозможные срабатывания переходов до достижения устойчивой маркировки – нет готовых к срабатыванию переходов. Заметим, что при срабатывании управляемых переходов появляются метки в правой части вектора ВВП.

3. На третьем шаге в правой части регистра проставляются 1 и 0 в соответствии с наличием меток в векторе ВВП и эти метки из правой части убираются. Далее, при изменении факта наличия события типа *ис* и *w*, шаги повторяются в бесконечном цикле.

4.1. Синтез сети Петри, моделирующей технологический объект управления.

После редукции ЛСПС синтез СП заключается в замене символов состояний символами позиций, сопоставлении событиям переходов и в построении по строкам ЛСПС (в соответствии со следованием в них символов) сети Петри. Введем обозначение: если подстрока h встречается в ЛСПС LS^i , то это будем обозначать $h \leftarrow\!\shortmid LS^i$.

Задача этапа конструирование СП для механизмов решается преобразованием $LS^{i.}$ по следующим правилам.

1. Множество G преобразуется в объединение множества подсетей $G = \langle G^1, G^2, \ldots, G^n \rangle \Rightarrow S_p = \cup_1^n S^i$ при этом каждый $G^i = \left\langle Q^i, E^i, \delta_i, \Gamma^i, Q_m^i, q_0^i \right\rangle$ преобразуется в $S^i = \left\{ P^i, T^i, Pre^i, Post^i, \mu_0 \right\}$ следующим образом: $P^i = \{ p_k \left| q_k \leftarrow\!\shortmid LS^i \right. \}$; $T^i = \{ t_{k-l} \left| e_{k-l} \leftarrow\!\shortmid LS^i \right. \}$.

2. Структура переходов (матрицы $Pre^i, Post^i$) определяется через входные и выходные функции переходов и соответствующие подстроки LS^i: $pre(t_{k-l}) = \{ p_j \left| q_j e_{k-l} \leftarrow\!\shortmid LS^i \right. \}$; $post(t_{k-l}) = \{ p_j \left| e_{k-l} q_j \leftarrow\!\shortmid LS^i \right. \}$.

3. Вектор начальной маркировки сети S_p $\mu_{0p} = [\mu_0^1 \mu_0^2 \ldots \mu_0^n]$ составляется из блоков μ_0^i, соответствующих позициям компонентных подсетей S_p^i. В каждом блоке μ_0^i в разряде, соотнесенном с q_0^i содержится 1, а в остальных - 0.

Для механизма G^1 и G^2 сеть имеет вид, представленный на рис.3 . Конструируем по автомату H соответствующую ЛСПС LS^H и сделаем ее расширение: каждый управляемое событие e_{k-i} заменим парой $e_{k-i} e_{k-j}$ где e_{k-j} событие, завершающий операцию e_{k-i} в соответствующем LS^i. ЛСПС автомата H - LS^H приобретает следующий вид: $\downarrow^1 q_1\ e_{1\text{-}1}\ e_{1\text{-}2}\ q_2\ e_{2\text{-}1}\ e_{2\text{-}2}\ q_3\ e_{1\text{-}3}\ e_{1\text{-}4}\ q_4\ e_{2\text{-}3}\ e_{2\text{-}4} \uparrow^1$.

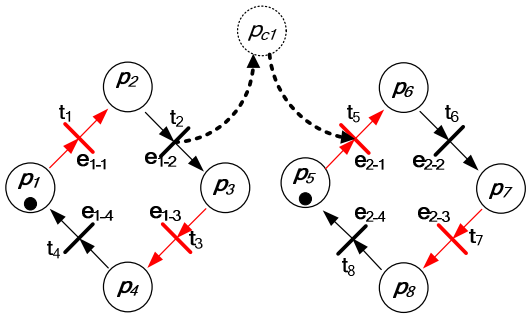


Рис. 3 Сеть Петри процесса для G^1 и G^2

4.2. Исследование свойств сетей Петри, моделирующих СДС2.

Введем необходимые определения. Далее в тексте все события e_{i-j} заменим именами переходов t_k и перенумеруем сплошной нумерацией по мере их следование в S_p, например как это показано на рис 3. Распространим на переходы сети S_p и S_c типы: *c* – управляемый, *uc* - неуправляемый, *w* – ожидаемый, в соответствии с типом событий соотнесенных с конкретными переходами.

Замечание 1. В соответствии с определением правил срабатывания переходов в СП всякий переход срабатывает, если во всех его входных позициях достаточно меток. Поскольку мы рассматриваем ДСС с форсируемыми управляемыми событиями, то задача сети S_p так организовать переходы, чтобы соответствующий t_k типа *c* срабатывал, когда он востребован по LS^H , при этом факт наличия событий типа *uc* и *w* в соответствующих фрагментах сети S_p проверяется и переходы срабатывают при наличии меток во входных позициях и факта наличия соответствующих событий на входе ДСС.

Определение1. Пусть сеть S_p моделируют объект (процесс) в СДС2 с форсируемыми управляющими событиями и в моделируемой СДС2 должна выполняться LS^H. Переходы t_j ,t_i сети S_p называются несвязанной парой (н-парой), если подстрока t_j,t_i входит в LS^H ($t_j,t_i \downarrow LS^H$) и в сети S_p не существует позиции p_k такой, что $t_j \in {}^\bullet p_k$, а $t_i \in p_k^\bullet$.

Следствие 1. Если t_j ,t_i н-пара и $t_j \in T^k$,а $t_i \in T^l$, то $k \neq l$.

Иными словами: переходы t_j и t_i принадлежат различным подсетям S^k и S^l сети S_p.

Определение 2. Пусть фрагмент *F* сети *S* , содержит единственный простой путь t_i …,t_j такой, что только для его начала t_i и конца t_j выполняется тип $\theta(t_i), \theta(t_j) \in \{c, u, uc\}$, тогда *F* называется структурой прямого взаимодействия (СПВ) переходов t_i и,t_j.

Лемма 1. Пусть сети процесса S_p и S_c, моделируют СДС2 с форсируемыми управляемыми событиями, так, что выполняется LS^H и переходы t_j,t_i является н-парой в S_p. Тогда в управляющей сети S_c (в супервизоре) существует СПВ из непосредственно связанных компонент $t_j \rightarrow p_c \rightarrow t_i$.

Доказательство необходимости (от противного). Допустим, выполняются условия леммы и подстрока $h := t_j,t_i$ входит в LS^H ($h \downarrow LS^H$), но в S_c - управляющей сети СДСС нет СПВ из непосредственно связанных компонент $t_j \rightarrow p_c \rightarrow t_i$ (далее просто СПВ $t_j p_c t_i$). Поскольку t_j,t_i являются н-парой, то такой СПВ нет и в S_p. Поскольку в объединенной сети должно выполняться основное свойства сетевой структуры (СП - двудольный граф), то

последовательность срабатывания t_j,t_i не выполняется в объединенной сети и не может входить в LS^H. Приходим к противоречию.

Доказательство достаточности. Допустим $u := rt_jt_iv$, $u \in LS^H$ и u является ее начальной подстрокой, переходы $t_j,,t_i$ является н-парой в сети процесса S_p, а сеть S_c конструируется последовательно и уже в ней выполняется подстрока r. Расширим матрицу инцидентности W_p дополнительной строкой p_c. Определим элементы дополнительной строки матрицы следующим образом: $W(p_c,t_j)$=+1, $W(p_c,t_i)$=-1, все остальные элементы строки равны 0. Тем самым мы преобразовали сеть S_p в S_p' путем расширения S_p СПВ $t_j\ p_c\ t_i$, так, как это показано на рис.3., и в этой сети становиться немедленно допустимой строка срабатывания rt_jt_i, содержащая подстроку h;= t_j,t_i, поскольку LS^H содержит все только допустимые строки срабатывания, то введение СПВ $t_j\ p_c\ t_i$ является достаточным преобразованием S_p. Что требовалось доказать.

Следствие 2. Поскольку LS^H и сети процесса S_p, моделируют СДСС с форсируемыми управляющими событиями, которые по определению может инициировать (разрешать) только супервизор, то из этого следует важное свойство СПВ $t_j\ p_c\ t_i$: первый переход в $t_jp_ct_i$ всегда соответствует ожидаемому или неуправляемому событию (не может быть взвешен управляемым событием), а второй переход в $t_jp_ct_i$ соответствует управляемому или неуправляемому событию (не может быть взвешен ожидаемым событием). Сформулированное следствие иллюстрирует таблица.

Таблица1. Допустимые сочетания типов переходов в н-парах и СПВ

1-е событие	*2-е событие*
w	*c*
w	*uc*
uc	*c*
uc	*uc*

Следствие 3. При преобразовании сети S_p в S_p' путем расширения S_p' СПВ $t_jp_ct_i$ P-инварианты сети S_p сохраняются и в сети S_p ′ (преобразование иллюстрирует рис.3, цепочка СПВ $t_jp_ct_i$ выделена пунктиром).

Однако множество достижимых состояний, несомненно, измениться. Более того, сеть из ограниченной превращается в неограниченную, поскольку в позиции p_c возможен неограниченный рост числа меток при бесконечном цикле в первой сети.

Лемма 2. Для всякой н-пары $t_j\ ,t_i$ скалярное произведение векторов – столбцов

$W(t_i)\bullet\ W(t_j)$= 0.

Доказательство. Все подсети S_P^k сети S_p автономны и не пересекаются ни по позициям, ни по переходам. Расположим позиции и переходы каждой S_P^k компактно в W_p (друг за другом в столбцах и строках), тем самым подматрицы W_P^k расположатся по диагонали матрицы инцидентности процесса W_p. Отсутствие в S_p позиции, связывающих переходы t_j ,t_i , свидетельствует о принадлежности t_j и ,t_i к различным S_P^i, и поэтому в пересечениях столбцов, соответствующих t_j и ,t_i со всеми строками W нет отличных от 0 значений, из чего следует, что скалярное произведение векторов – столбцов $W(t_i)\bullet\ W(t_j)= 0$. Что требовалось доказать.

Лемма 3. Пусть расширение сети S_p осуществляется множеством СПУ, сконструированных для всех н-пар из последовательности срабатывания *и*, соответствующей LS^H . Тогда существует изоморфное отображение M′ - множества достижимых маркировок сети S_p ′ на Q={Q_1, Q_2, ….Q_r} - множество достижимых векторов-состояний компонент $G = \langle G^1, G^2, ..., G^n \rangle$.

Доказательство. Существование изоморфного отображения между M′ - множеством достижимых маркировок сети S_p ′ и Q={Q_1, Q_2, ….Q_r} - множеством достижимых векторов-состояний компонент $G = \langle G^1, G^2, \dots, G^n \rangle$, где $Q_j := \langle q_{j1}^1, q_{j2}^2, ..., q_{jn}^n \rangle$, следует из того, что, во-первых, $G = \langle G^1, G^2, \dots, G^n \rangle$ и $H = (Q^h, E_d, \delta_h, \Gamma^h, Q_m^h, q_0)$ согласованны; во-вторых, $G = \langle G^1, G^2, \dots, G^n \rangle \Rightarrow S_\delta = \bigcup_{i=1}^{n} S^i$; множество позиций S_p конструируется как отображение множества состояний на множество позиций сети (фактически пере обозначение), и ,наконец, в-третьих, правила определения структуры переходов S_p, сформулированные через входные и выходные функции переходов и соответствующие подстроки LS^i, и поэтому соответствуют функциям переходов автоматов G^i. Из чего следует, что срабатывание переходов в сети $\langle Sp \cup Sc \rangle$ соответствует последовательности смены Q_i, Q_j, ….Q_r векторов-состояний компонент $G = \langle G^1, G^2, \dots, G^n \rangle$. Таким образом, сохранение отношения следования при отображении M′ на Q определяет его изоморфизм.

Теорема 1. Пусть $S = S_p \cup S_c$, а S_c определяется как расширение сети S_p множеством СПВ, сконструированных для всех н-пар из последовательностей срабатывания *и*, входящих LS^H . Тогда вектор Париха $Y(LS^H)$, соответствующий *и* является Т- инвариантом сети S., и, кроме того Р- инварианты сети S_p сохраняются в S.

Доказательство. Т- инвариантность вектора Париха $Y(LS^H)$ соответствующего *и* является следствием того, что $H = (Q^h, E_d, \delta_h, \Gamma^h, Q_m^h, q_0)$ задает связной автомат, поэтому LS^H ,

сконструированная по конечному автомату H описывает циклический процесс. Справедливость последнего утверждения теоремы: Р- инварианты сети S_p сохраняются в $S_p^{'}$ следует из справедливости утверждений леммы 1 , леммы 3 и следствия 1. Что и требовалось доказать.

Замечание 2. Из утверждений лемм, теоремы и сформулированных в настоящем разделе свойств следует, что если для всех возможных последовательностей строк, задаваемых например ЛСПС выявить все н-пары t_j t_i и каждую пару в сети процесса обеспечить цепочкой $t_j\, p_c\, t_i$ СПУ, то:

- во-первых, все структурные свойства исходной сети процесса сохраняются;
- во-вторых, все требуемые последовательности операций, определяемые спецификацией, будут выполняться.

4.2. Синтез управляющей сети по S_p и ЛСПС спецификации последовательностей команд.

Основная идея предлагаемого нами метода основывается на следующем свойстве циклических реактивных систем с форсируемыми событиями: каждая следующая операция объекта выполняются по событию, завершающему предшествующую операцию (подобно падению костяшек домино), или внешнему событию (например, при пуске или выборе). В этом случае задача супервизора - организовать передачу управления от механизма, выполнившему операцию, к механизму, операция которого по LS^H должна выполняться следующей. Метод синтеза управляющей сети Петри мы назвали методом домино. В этом разделе мы ограничимся разработкой ядра метода ограничивающего LS^H бесповторной [2] ЛСПС, содержащей одну последовательность.

Опишем процедуру проектирования управляющей сети S_c из СПУ t_j p_c t_i для ЛСПС, содержащей 1 бесповторную последовательность срабатывания переходов.

Процедура построения управляющей сети S_c (базовый алгоритм).

Обозначения, используемые в тексте процедуры. **Sintez** - имя процедуры; **W,n, u, r** – формальные параметры**;** $u := t_1 t_2 t_3 \ldots t_{r-1} t_r$ - строка ЛСПС, обрабатываемая процедурой (в u стрелочки заменены именами переходов, на которые эти стрелочки указывают в ЛСПС); r - длина строки u; $W(n \times m)$ – матрица инцидентности сети процесса S_p ; $W(k, \sim) = 0$ означает присвоение всем элементам строки k матрицы W значение 0; $u(i)$ i-тый символ

[2] В бесповторной последовательности срабатывания каждый переход упоминается не более 1 раза. Это ограничение далее будет снято.

строки *u.. W(u(i)), W(u(i+1))* - векторы- столбцы матрицы *W,* SProdct(*W(u(i)), W(u(i+1))*) – скалярное произведение векторов. Определение процедуры на языке Паскаль:

Sintez (W, Pc, n, u, r)

i=1

While i≤(r-1) do

Begin

If SProdct (W(u(i)), W(u(i+1)))≠ 0 go to M1;

n=n+1;

W(n, ~)=0;

W(n, u(i))=+1;

W(n, u(i+1))=—1;

Pc= {Pc}∪{p_n};

M1: i=i+1

End.

Применим процедуру **Sintez** к сети (см. рис. 3 а)) для 2 –х механизмов G^1 и G^2. Матрица инцидентности *W* этой сети представлена на рис. 3 б). В матрице *W* имена событий заменены именами соответствующих переходов следующим образом: ($e_{1\text{-}1}\rightarrow t_1$)…($e_{2\text{-}4}\rightarrow t_8$). Обрабатываемая строка u:=$t_1t_2t_5t_6t_3t_4t_7t_8t_1$; r=9.

Таблица 2. Матрица инцидентности W для СП механизмов G^1 и G^2 .

	t1	t2	t3	t4	t5	t6	t7	t8
1	-1	0	0	1	0	0	0	0
2	1	-1	0	0	0	0	0	0
3	0	1	-1	0	0	0	0	0
4	0	0	1	-1	0	0	0	0
5	0	0	0	0	-1	0	0	1
6	0	0	0	0	1	-1	0	0
7	0	0	0	0	0	1	-1	0
8	0	0	0	0	0	0	1	-1
P_{c1}	0	+1	0	0	-1	0	0	0
P_{c2}	0	0	-1	0	0	+1	0	0
P_{c3}	0	0	0	+1	0	0	-1	0
P_{c4}	-1	0	0	0	0	0	0	+1

Результатом работы процедуры **Sintez** является построение сети из 4-х управляющих позиций *($p_{c1},…p_{c4}$).* Соответствующая сеть представлена на рис. 4.

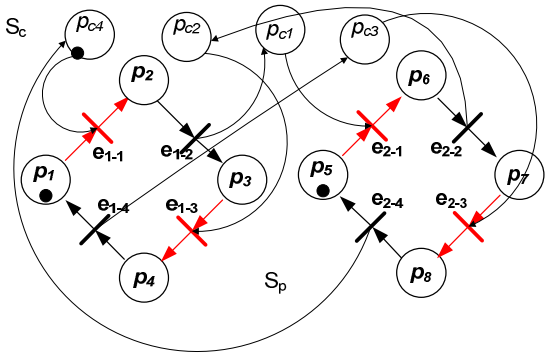


Рисунок 4. Сеть Петри с управляющими позициями $p_{c1}, p_{c2}, p_{c3}, p_{c4}$, реализующими управление, так, что выполняется последовательность u:=$t_1t_2t_5t_6t_3t_4t_7t_8t_1$;

Вычисление μ_0.

Вектор начальной маркировки состоит из начальной маркировки сети процесса и начальной маркировки сети контроллера $\mu_0^T = [\mu_{0P}^T \mu_{0C}^T]$. Первая часть определяется при построении и соответствует начальным состояниям ЛСПС (в позициях вектора μ_{0p}^t, соответствующих начальным состояниям в определении ЛСПС присваивается 1, а остальным 0).

Правило для начальной разметки μ_{0C}^T. Поскольку вектор Y является Т-инвариантом (задает цикл), то если из него исключить срабатывание последних в цикле переходов, то матричная операция W с модифицированным Y' определит начальную разметку для S_c.

Определим вектор Y'=Y(-1). В Y'уменьшено на 1 значение в позициях вектора Y, соответствующих переходам из н-пары, вторыми компонентами в которых является t_1 из ЛСПС (первый переход в ЛСПС). В нашем примере это t_8. Тогда μ_{0C}^T = -WY' , для позиций, относящихся к сети контроллера. Для нашего примера μ_{0C}^T=0001. В итоге, $\mu_0^T = 100010000001$.

4.3. Улучшение S_c путем минимизация каналов управления.

Процедура синтез вводит позицию, передающую управление, для каждой н-пары подобно каналу связи. Эти каналы «работают», только в момент активности соответствующей н-пары и после передачи управления в следующие переходы не участвуют в процессе активизации других переходов, пока в следующем цикле они не будут вновь востребованы. Из чего следует, что формирование для каждой н-пары нового канала в сети во многих случаях избыточно. Прозрачно определяются отношение совместимости

каналов (по отсутствию пересечения предшественников и последователей) и соответствующие правила. Эти правила подобны правилам совместимости состояний для конечного автомата. Применив эти правила и проделав традиционные преобразования, для нашего примера удалось сократить число позиций в два раза. Результат представлен на рис 5.

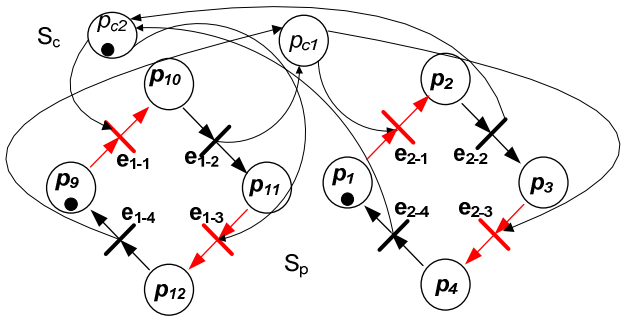


Рисунок 5. Преобразованная сеть Петри, выполняющая последовательность u:= $t_1t_2t_5t_6t_3t_4t_7t_8t_1$;

5. Синтез управляющей сети по ЛСПС, построенной по спецификации СДСС без ограничений на бесповторность и наличие ветвлений.

В настоящем разделе предложенная процедура синтеза S_c развивается, в направлении отказа от ограничений на бесповторность и количество строк в LS.

5.1. Формирование структуры прямого взаимодействия в зависимости от свойств н-пар в ЛСПС.

В спецификации требуемого поведения в общем случае допустимы многократные срабатывания механизмов и многовариантное поведение, регулируемое предшествующими срабатываниями механизмов в уже отработанных последовательностях или внешними (неуправляемыми) событиями. Последнее представляется ветвлениями в графе переходов автомата $H = (Q^h, E_d, \delta_h, \Gamma^h, Q^h_m, q_0)$, которые в ЛСПС представляются ветвителями.

Рассмотрим фрагмент перегрузочного устройства транспортной системы, состоящего из 4-х механизмов: М1, М2, М3, М4 (назначение опустим). Механизмы однотипные, каждый выполняет 2-е операции подобно механизмов из примера на рис. 3. Считаем, что СДСС-моделирование устройства прошло успешно и сконструированы следующие сети Петри по ККА.

$S^i = (P^i, T^i, W^i, \mu_0)$, где $P^i = \{p_j | j = k \div (k+3)\}$, $T^i = \{t_j | j = k \div (k+3)\}$, при этом для значения *i*=1,2,3,4 значение *k*=1,5,9,11 соответственно. $S_p = \bigcup_1^4 S^i$ W^i подобны матрицам из таблицы 2, они представлены в объединенной матрице инцидентности W_p в таблице 3. Последовательности выполнения операций представлены следующим множеством строк ЛСПИ $LS = \{\downarrow^1 t_{17}t_1t_2t_5t_6t_7t_8t_9t_{10}\{t_{19}\uparrow^2, t_{20}\uparrow^3\}; \downarrow^2 t_{13}t_{14}t_{11}t_{12}t_{15}t_{16}\uparrow^4; \downarrow^3 t_5t_6t_{11}t_{12}t_7t_8\uparrow^4; \downarrow^4 t_3t_4t_{18}\uparrow^1;\}$, построенным по графу переходов автомата *H*.

Переходы с нечетными индексами в диапазоне 1÷16 - управляемые переходы (переходы соответствующие управляемым событиям), следующие за ними переходы с четными индексами в этом же диапазоне - ожидаемые (переходы соответствующие ожидаемым событиям окончания соответствующей операции).

Неуправляемые переходы: t_{17} - подан полет с деталью на разгрузочную площадку; t_{18} - разгруженный полет снят с разгрузочной площадки; t_{18} - деталь без дефекта, t_{19} – деталь с дефектом.

Функционирование. После подачи полета на разгрузочную площадку (сработал неуправляемый переход t_{17}), механизм М1 фиксирует полет (t_1t_2), механизм М2 сдвигает деталь с полета на площадку перегрузки в контейнеры и возвращается в исходное положение ($t_5t_6t_7t_8t_9$), затем М3 поднимает площадку перегрузки на уровень контейнеров (t_9t_{10}). Затем, в зависимости от отсутствия или наличия дефекта, срабатывают в ветвителе соответствующие переходы $t_{19}\uparrow^2, t_{20}\uparrow^3$. Продолжение будет или в строке 2 механизмом М4 или в строке3 механизмом М2 в любом варианте подъемник М3 после передачи детали в контейнер возвращается в исходное положение ($t_{11}t_{12}$). После завершения перегрузки полет расфиксируется М1 (t_3t_4) и снимается с разгрузочной площадки (внешним устройством), что подтверждается соответствующим событием и срабатыванием перехода t_{18} , цикл завершен.

Многократность и вариантность срабатывания механизмов приводит к многообразию в сочетании событий в н-парах (срабатывание перехода будем называть событием). Возможны различные н-пары, в которых совпадают 1- е или вторые 2-е события. Рассмотрим, как это отразиться на структуре звеньев передачи управления.

Замечание 3. (Совпадение 1-го события). Наличие в LS двух или более н-пар с совпадающими первыми символами означает, что одно и то же событие, например t_8 в различных частях последовательности - спецификации влечет различные последствия: $t_8 - t_9$ и $t_8 - t_3$. Следовательно, необходимо в СПВ привлечь дополнительную

информацию (другие события или их последовательности, которые различаются, например, автоматами Fe_{ij}, Fe_{ik}). Соответствующая схема соединения с выходами автоматов Fe_{ij}, Fe_{ik} представлена рис. 6.

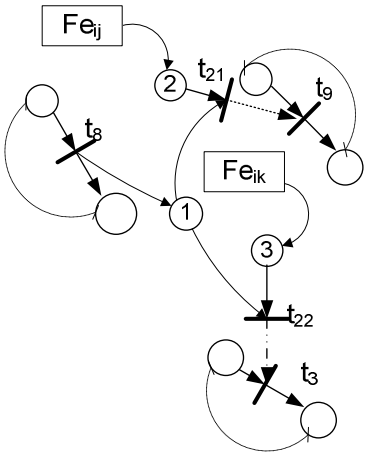


Рисунок 6 Схема подключения автоматов-агентов выбора направления в СПВ

На рис. 6 автомат-агент различия направлений взаимодействия представлен блоками Fe_{ij} и Fe_{ik}, и их выходными позициями 1 и 2 соответственно. Блоки также представляются сетями Петри, а логика их срабатывания определяет передачу управления по цепи $1 - t_{21} - t_9$ либо по цепи $1 - t_{22} - t_3$. Пунктирные стрелки указывают на совмещение переходов в один, если соответствующий переход не участвует в сложном сочетании, соответствующем рис. 7. Вопросы построения Fe_{ij} и Fe_{ik} обсудим позже, в разделе 5.3.

Замечание 4. Совпадение 2-го события. Наличие 2-х или более н-пар типа: $t_i - t_k, t_j - t_k, \dots, t_n - t_k$ означает, что переход t_k срабатывает всякий раз, когда срабатывает один из переходов $t_i, t_j, \dots t_n$. Таким образом в СПВ должна производиться сборка по функции “ИЛИ”. Для этого в процедуре конструирования СПВ всякая н-пара, например $t_i - t_k$ будет анализироваться на возможность «повтора 2-го символа». При обнаружении такой н-пары, например $t_j - t_k$, в СПВ вводиться позиция, в которой осуществляется сборка передачи управления в общий для всех н-пар переход t_k , в который из позиции – сборки вводится ребро $p_{or} - t_k$ (см. рис. 7).

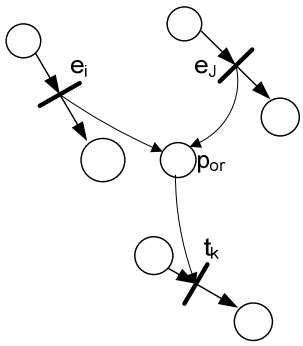


Рисунок 7 Соединение двух звеньев СПВ по схеме «ИЛИ»

Тезис. Из замечаний 3 и 4 следует, что в СПВ полезно выделить входную (см. рис. 6) и выходную (см. рис. 7) части и локальный анализа н-пары недостаточен для выбора типа входной и выходной частей СПВ. При анализе н-пары необходима информация о наличии совпадений по 1-му и 2-му событию с другими н-парами в LS. Выполним просмотр LS и результаты сведем в *MF (g×h)* – матрице запусков.

Таблица 3. Матрица *MF(g×h)* запусков для LS перегрузочного устройства транспортной системы

	t_1	t_3	t_5	t_7	t_9	t_{11}	t_{13}	t_{15}	ω_1	t_{17}	t_{18}
t_2	0	0	1	0	0	0	0	0	0	0	0
t_4	0	0	0	0	0	0	0	0	0	0	1
t_6	0	0	0	1	0	1	0	0	0	0	0
t_8	0	1	0	0	1	0	0	0	0	0	0
t_{10}	0	0	0	0	0	0	0	0	1	0	0
t_{12}	0	0	0	1	0	0	0	1	0	0	0
t_{14}	0	0	0	0	0	1	0	0	0	0	0
t_{16}	0	1	0	0	0	0	0	0	0	0	0
t_{17}	1	0	0	0	0	0	0	0	0	0	0
t_{18}	0	0	0	0	0	0	0	0	0	1	0
t_{19}	0	0	0	0	0	0	1	0	0	0	0
t_{20}	0	0	1	0	0	0	0	0	0	0	0

Матрица строится по событиям, образующим множество всех н-пар в *LS* и в соответствии с допустимым типажом. Для LS функционирования перегрузочного устройства транспортной системы, состоящего из 4-х механизмов: М1, М2, М3, М4 матрица *MF(g×h)* запусков представлена в таблице 3.

Столбцы *MF(g×h)* сопоставляются событиям, являющимися вторыми компонентами в н-парах: переходам t_i спостaвленным управляемым событиям $e_i \in E_c$; переходы t_i

сопоставленным событиям $e_i \in E_{uc}$и входящими в строки LS вне ветвителей ω_i; переменныу– ω_i, сопоставленным ветвителям. Напротив, строки *MF(g×h)* сопоставлены первым компонентам из н-пар – это события типа *w* и *uc*. Кроме того в число н-пар необходимо включить подстроки $t_i t_j$↵ *LS* такие, что $\exists(t_k t_j)$↵ *LS* и $t_k t_j$ является н-парой (у этих пар одинаковый второй символ, необходимость наделения их статусом н-пар поясним ниже).

Элемент $MF_{ij} = 1$, если $t_i t_j$↵ *LS* и $t_i t_j$ является н-парой, в противном случае $MF_{ij} = 0$. Легко видеть, что в строках матрицы *MF(g×h)* более одной 1, если 2 или более н-пар имеют совпадающими первые переходы, а в столбцах матрицы более одной 1, если 2 или более н-пар имеют совпадающими вторые переходы. Таким образом, матрица аккумулирует информацию о соотношении н-пар и ее достаточно для построения СПВ. Введем для каждого *i*-го ветвителя $\{e_{i1}\uparrow^k, \dots, e_{ir}\uparrow^r\}_i$ переменную ω_{i} как компоненту в н-пары, что позволяет формирование соответствующей сетевой конструкции выполнить также в процессе обследования матрицы *MF*. С этой целью введем в рассмотрение и множества $\Omega_i = \{t_j | e_j \leftarrow \omega_i\}$ (из переходов, соответствующих событиям ветвителя).

Для формирования *MF(g×h)* заменим в процедуре ***Sintez*** операторы расширяющие матрицу *W* на операторы заполнения матрицы *MF* для каждой н-пары (Приложение1).

5.2. Метод синтеза управляющей сети по ЛСПС общего вида

Метод синтеза управляющей сети S_c разрабатывается как циклическая процедура перебора и анализа всех н-пар и формирования для каждой н-пары структуры прямого взаимодействия. Основные шаги метода будем иллюстрировать синтезом сети S_c для перегрузочного устройства транспортной системы, изложенного в разделе 5.1.. На начало синтеза управляющей сети Петри S_c полагаем (как и разделе 3.4), что сеть S_p уже сконструирована по LS^i и матрица инцидентности сети процесса S_p - *W(n×m)* – определена для S_p. (Столбцы сопоставлены переходам, а строки сопоставлены позициям S_p). Исходная матрица *W* представлена в таблице 4 (выделена двойной линией). Кроме того в *W(n×m)*добавлены столбцы, сопоставленные событиям из E_{uc}, входящими в LS (столбцы $t_{17}, t_{18}, t_{19}, t_{20}$).

Таблица 4. Составная матрица изменений сети $S = S_p \cup S_c$ для M1,M2,M3, M4.

Составная матрица изменений сети Петри для модели объекта M1-M2-M3-M4

		t1	t2	t3	t4	t5	t6	t7	t8	t9	t10	t11	t12	t13	t14	t15	t16	t17	t18	t19	t20	t21	t22	t23	t24
	1	-1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	2	1	-1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	3	0	1	-1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
M1	4	0	0	1	-1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	5	0	0	0	0	-1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	6	0	0	0	0	1	-1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	7	0	0	0	0	0	1	-1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
M2	8	0	0	0	0	0	0	1	-1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	9	0	0	0	0	0	0	0	0	-1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
	10	0	0	0	0	0	0	0	0	1	-1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	11	0	0	0	0	0	0	0	0	0	1	-1	0	0	0	0	0	0	0	0	0	0	0	0	0
M3	12	0	0	0	0	0	0	0	0	0	0	1	-1	0	0	0	0	0	0	0	0	0	0	0	0
	13	0	0	0	0	0	0	0	0	0	0	0	0	-1	0	0	1	0	0	0	0	0	0	0	0
	14	0	0	0	0	0	0	0	0	0	0	0	0	1	-1	0	0	0	0	0	0	0	0	0	0
	15	0	0	0	0	0	0	0	0	0	0	0	0	0	1	-1	0	0	0	0	0	0	0	0	0
M4	16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	-1	0	0	0	0	0	0	0	0
	17	-1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
	18	0	0	-1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0
	19	0	1	0	0	-1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
	20	0	0	0	0	0	0	-1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1
	21	0	0	0	0	0	0	0	0	0	0	-1	0	0	1	0	0	0	0	0	0	0	1	0	0
	22	0	0	0	0	0	0	0	0	0	0	0	0	-1	0	0	0	0	0	1	0	0	0	0	0
^	23	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	-1	-1	0	0	0	0
	24	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-1	1	0	0	0	0	0	0
	25	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	-1	0	0	0	0	0	0
$	26	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-1	-1	0	0
&	27	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	-1	0	0	0
&	28	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	-1	0	0
$	29	0	0	0	0	0	0	0	1	-1	0	0	0	0	0	0	0	0	0	0	0	0	0	-1	0
&	30	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	-1	0
&	31	0	0	0	0	0	0	0	0	-1	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
$	33	0	0	0	0	0	0	0	0	0	0	0	1	0	0	-1	0	0	0	0	0	0	0	0	-1
&	34	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	-1
&	35	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-1	0	0	0	1	0	0	0	0	0
	0	17	0	18	0	19	0	20	0	-1	0	21	0	22	0	-1	0	24	25	0	0	0	0	0	0
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Для ветвителей в LS определены множества Ω_i (напомним, список событий в фигурных скобках мы назвали ветвителем). С этой целью введены в рассмотрение множества $\Omega_i = \{t_j | e_j \leftarrow \omega_i\}$ (из переходов, соответствующих событиям ветвителя), i=1,…k , где k количество ветвителей в LS. Матрица *MF(g×h)* запусков также определена и представлена в таблице 3 (раздел 5.1.).

Основные процедуры (шаги) метода представлены блок-схемой на рис. 8. Конструирование СПВ предлагается выполнять в два приема. На первой фазе (блоки 1,2,3) выполняется цикл по столбцам *MF(g×h)* и формируется для всех СПВ выходная компонента – это позиция и ребро в соответствующий переход (вторая компонента н-пары). Вариант конструкции определяется по типу столбцов и пересекающихся с ними по единицам строк, в соответствии с положением, изложенным в Замечании3 (см. рис. 7).

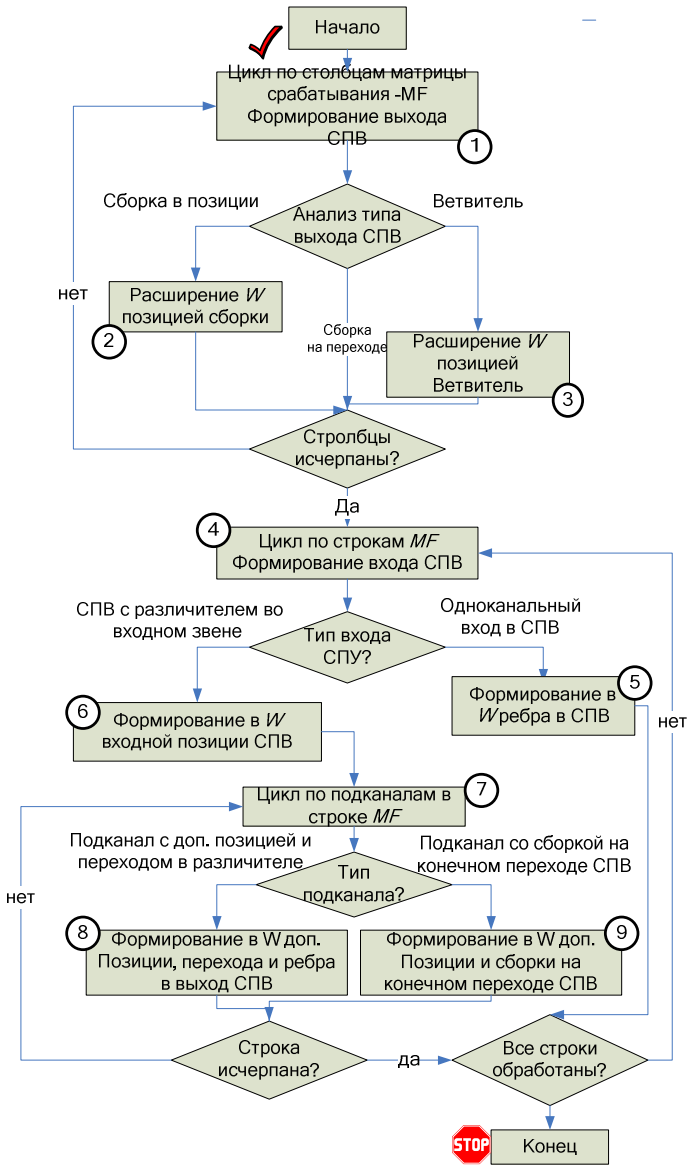


Рисунок 8. Блок-схема метода синтеза $\boldsymbol{S_c}$

Конструирование включает:

- введение новых позиций (строк) в матрице W – позиции $p_{17} - p_{25}$ (исключая p_{23}, которая введена в блоке 3 алгоритма для ветвителя);
- определение связи ребрами сети этих позиций с соответствующими переходами.

На второй фазе метода (цикл по строкам, блоки 4 – 9 на рис. 8) формируется входная часть СПВ по типу строк и пересекающихся с ними по единицам столбцов. При одноканальном входе в СПВ блок 5 формирует ребро. В случае СПВ для н-пар с совпадающим первым событием (переходом) необходимо формирование различителей. В нашем примере таких н-пар 3. В соответствии с типовой конструкцией, изложенной в Замечании 2 (см. рис. 6) , в блоках 6,7,8,9 алгоритма сформированы: p_{26}, p_{27}, p_{28} ; p_{29}, p_{30}, p_{31}; p_{33}, p_{34}, p_{35} и новые переходы t_{21}, t_{22}, t_{23}. Последние соединяются ребрами с

выходной частью соответствующего СПВ. Для нашего примера это отражено в соответствующем заполнении матрицы W.

Легко показывается истинность следующего утверждения относительно предложенного метода «домино» синтеза сети контроллера.

Утверждение. Пусть даны S_p,$W(n \times m)$ $MF(g \times h)$, LS и количество заполненных элементов $MF(g \times h)$ равно k. Тогда алгоритм «домино» по S_p,$W(n \times m)$ $MF(g \times h)$, LS синтезирует S_c с числом позиций $\leq (k+g)$ и при этом число операций $O(g \cdot h)$.

Доказательство. Количество позиций определяется в блоках алгоритма 2,3,8, и 9 исключительно при значении 1 соответствующего элемента матрицы $MF(g \times h)$ плюс дополнительно в блоке 6, число переходов к которому ≤ g. Число операций меньще или равно числу элементов матрицы $MF(g \times h)$, из чего следует справедливость второй части утверждения.

5.3 Формирование логики функций различителей в сложных структурах прямого взаимодействия.

Общим решением является конструирование автомат-агент выбора направления A по спецификации $H = (Q^h, E_d, \delta_h, \Gamma^h, Q_m^h, q_0)$. А – это автомат Мура с числом выходов равным числу н-пар с одинаковыми первыми символами. В нашем примере это два выхода e_8e_9 и e_8e_3 (для простоты, пусть индексы событий совпадают с индексами соответствующих им переходов). Можно предложить несколько вариантов конструирования А. Самый очевидный – преобразование H в автомат Мура с выходами e_ie_j и e_ie_k, равными 1 на состояниях, из которых исходят ребра, соответствующие e_i и 0 на всех остальных состояниях. (Это однозначно, поскольку каждой н-паре соответствует 1 ребро в графе переходов H). Более упрощенным подходом является формирование множества простых путей и соответствующих им *опорные множества входных слов*, ведущих в ребра, взвешенные различными вторыми событиями. Определение опорных множеств даны в работе (Амбарцумян, Томилин -2010). Следующим шагом является поиск событий в опорных множества, которые являются доминантами всех простых путей в соответствующем множестве. Если эти события для каждого опорного множества оригинальны, то это и является решением задачи различия направлений передачи прямого взаимодействия. Практически это всегда удается, за исключением ситуаций, когда нужен счет появления доминантного события. В этом случае нужно вернуться к определению соответствующего автомата.

В нашем примере для н-пар $t_6 - t_7$ и $t_6 - t_{11}$ доминатами являются соответственно переходы t_{20} и t_{17}, для $t_8 - t_3$ и $t_8 - t_9$ соответственно t_{10} и t_{17} для $t_{12} - t_7$ и $t_{12} - t_{15}$ соответственно t_{20} и t_{19}, что и отражено в переходах матрицы W.

На рис. 9 по матрице W построена соответствующая сеть Петри

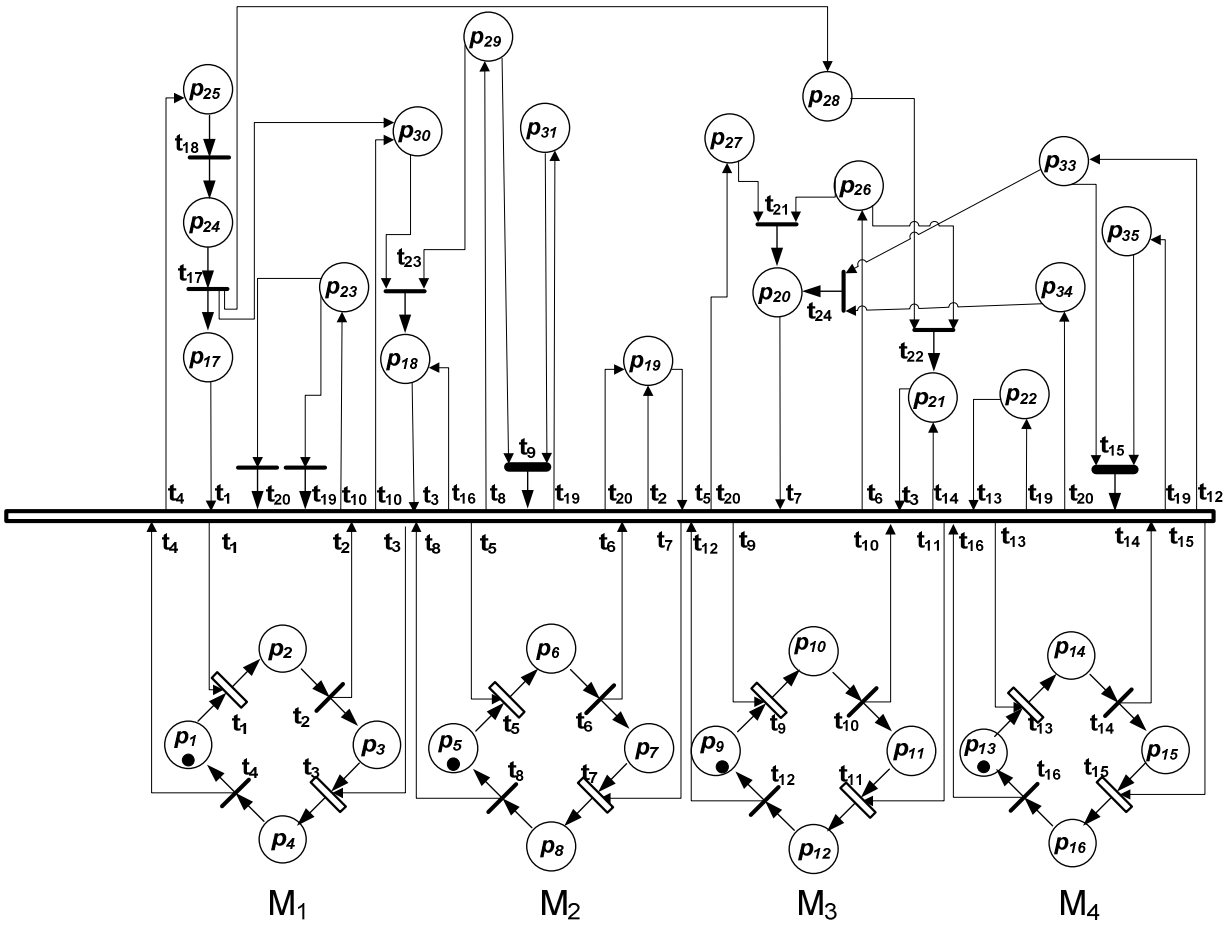


Рисунок 9. Сеть Петри к задаче управления фрагментом перегрузочного устройства транспортной системы, состоящего из 4-х механизмов: M1, M2, M3, M4

Определение вектора начальной маркировки произведем, в соответствии с правилом, изложенным в раздале 4.2. $\mu_0^T = [1000100010001000100000000000000000]$, у сети S_c только для p_{17} 1 в начальной маркировке.

С целью облегчения анализа рисунка 9, соединения между переходами сети процесса S_p и сети контроллера S_c изображены символически через «общую щину», но их логику легко восстановить по именам. По рисунку легко просматриваются основные особенности метода синтеза управляющей сети контроллера S_c :

- все управляемые переходы сети процесса S_p дополнены входными ребрами от S_c ;
- Структура сети процесса S_p осталась неизменной, но управляемой входными ребрами на управляемые события.

- Всякие изменения порядка работы компонент сети процесса S_p затронут изменениями только сеть контроллера S_c.

6. Заключение.

В работе разработана методология дискретно-событийного моделирования объекта и его требуемого поведения (спецификаций) при проектировании систем автоматизации, работающих в реальном времени.

Методология основана на использовании для анализа функциональности и согласованности модели: структурированная дискретно-событийная система (СДС2). Разработан метод синтеза по СДС2-модели сети Петри S_p, моделирующей процесссы в объекте, разработана техника анализа S_p и метод синтеза управляющей сети супервизора – S_c, обеспечивающей совместно с сетью S_p выполнение исходных спецификаций. Метод синтеза управляющей сети S_c основан на анализе ЛСПС - компактной модели последовательности срабатывания переходов в сети процесса. Предложенная модель СДС2 и процедуры работы с ней максимально учитывают особенности систем автоматизации, работающих в реальном масштабе времени, отмеченные во введении. Есть основание полагать (*утверждение* раздел 5.2), что в предложенном методе, основанном на линейном просмотре всех простых строк из ЛСПС, удается избежать «взрыва состояний» при синтезе супервизора.

Важным свойством управляющей сети Петри, синтезированной по предложенной методике, является то, что в ее структуре сохранена S_p - СП-модель актуаторов (исполнительных механизмов нижнего уровня). Сеть Петри S_c, управляющая интеграцией компонент нижнего уровня (супервизор), синтезирована как отдельная компонента, взаимодействующая с сетями нижнего уровня. Важно отметить, что сети S_c нет в структуре исходных данных. Напротив – эта сеть результат синтеза обратных связей в S_p на основе анализа возможностей модели процессов в объекте - S_p и требований спецификации *LS.* Таким образом, можно констатировать, что предложенная методология синтеза управляющей сети Петри, соблюдает важнейший концепт сетецентрического управления: *неизменность функциональности автономных компонент и выполнение на уровне сетевой интеграции обработки информации в интересах потребителей*.

Предлагаемая методика изложена применительно к механической системе. Однако нет никаких ограничений на применение сформулированных результатов к синтезу супервизорного управления бизнес-процессами в организационных системах. При этом строки из команд (директив на выполнение вариантов бизнес-процессов) языка

супервизора *K* можно интерпретировать как различные варианты достижения цели функционирования. Синтезированный супервизор – сценарий заданный сетью Петри; изменение сценария – смена матрицы, а в реальной организационной системе смена текущего плана мероприятий.

7. Литература.

1. Ramadge, J. G., & Wonham, W. M. Supervisory control of a class of discrete event processes. // SIAM Journal of Control and Optimization, – 1987., N 25, pp.206-230.
2. Cassandras, C.G., Lafortune, S. Introduction to discrete event systems. Dordrecht: Kluwer Academic Publishers. – 2008., p. 848.
3. Wonham, W. M., Ramadge, J. G. Modular supervisory control of discrete event systems. // Math Control Signals and Systems, - 1988., N 1, pp.13-30.
4. Yoo, T.-S., Lafortune, S. A general architecture for decentralized supervisory control of discrete event systems. // Discrete Event Dynamic Systems: Theory& Applications, - 2002. N 12(3), pp. 335-37
5. De Queiroz, M.H., Cury, J.E.R. Modular supervisory control of large scale discrete event systems. // Discrete Event Systems: Analysis and Control, Proc. WODES, 2000, pp. 103-110.
6. Akesson, K., Flordal, H., Fabian, M. Exploiting modularity for synthesis and verification of supervisors. // In Proc. Of the IFAC World Congress, 2002.
7. Gaudin, B., Marchand, H. Modular supervisory control of asynchronous and hierarchical finite state machines. // In European ControlConference, Cambridge, 2003
8. Holloway L. E. and Krogh B. H., Synthesis of feedback logic for a class of controlled Petri nets. // *IEEE Trans. Autom. Control*, - 1990. AC-35, N 5, pp. 514-523.
9. Yamalidou K, Moody J O, Lemmon M D, Antsaklis P J., Feedback control of Petri nets based on place invariants. // IEEE Transaction on Robotics and Automation, 1996. N 32(1), pp. 15-28.
10. Dideban A., Alla H., Reduction of Constraints for Controller Synthesis based on Safe Petri Nets. // Automatica, 2008. N 44(7), pp. 1697-1706.
11. Амбарцумян А.А. Супервизорное управление структурированными динамическими дискретно-событийными системами. // М. АиТ №8, 2009, с.с. 156-176.
12. Golaszewski, C. H., Ramadge, P. J. Control of discrete event processes with forced events. // Proceedings of the 28th Conference on Decision and Control, Los Angeles. 1987. pp. 247–251

13. Амбарцумян А.А. Томилин Е.Е. Метод прямого синтеза супервизора для структурированных дискретно событийных систем // М.: Автоматика и Телемеханика, - 2010, N8, в печати
14. ЛАЗАРЕВ В.Г., ПИЙЛЬ Е.И. Синтез управляющих автоматов. – М.: Энергоатомиздат, 1989
15. BARANOV S. Logic and System Design of Digital Systems. // Published jointly Tallinn University of Technology Press fnd SIB Publishers (Toronto), 2008, p.266 ISBN 978-9985-59-769-9
16 ЗАКРЕВСКИЙ А.Д., ПОТТОСИН Ю.В., ЧЕРЕМИСИНОВА Л.Д. Логические основы проектирования дискретных устройств., М.: Физматлит, 2007, 588 стр.
17. Petrson ,J. L. Petri Net theory and the modeling of systems., Prentice-Hall, Inc. Englewood Cliffs, N.J. 1981. Русск. Издание Питерсон Дж. Теория сетей Петри и моделирование систем. М., Мир, 1984 г. 264 с.

Приложение 1.

Процедура формирования матрицы срабатывания *MF*(*g*×*h*).

В общем виде ЛСПС, построенная по графу переходов конечного автомата, представляет собой конечное множество строк LS={v_i| i=1÷k}. Типичная строка v_i приведена в разделе 2 для графа переходов G^3: $\downarrow^1 q_1\ e_{3\text{-}1}\ q_2\ e_{3\text{-}2}\ q_3\ e_{3\text{-}3}\ \downarrow^4 q_4\{e_{3\text{-}8}\uparrow^2,\ e_{3\text{-}4}\uparrow^3\}$. В общем виде v_i:= $\downarrow^{m1} u_i \ldots \downarrow^{mk} u_j \ldots u_l\{e_{3\text{-}8}\uparrow^{r1},\ldots,\ e_{3\text{-}4}\uparrow^{rn}\}$.

Введем преобразования строк LS, исключающие стрелки.

Преобразование 1. Пусть в строку v_i входит $\downarrow^m e_r$ - входная стрелочка m и следующий за ней символ события e_r . Во всех строках LS исходящие стрелочки с индексом m заменим на e_r. Стрелочку $\downarrow^m$ из v_i исключить. (Внимание, преобразование на применяется, если следующий за входящей стрелочкой символ ветвитель –«{}».) Легко видеть , что Преобразование 1 не изменяет множество н-пар в *LS*. Действительно, удаление $\downarrow^m$ из строки не меняет количество н-пар, поскольку стрелочки в них не входят. Однако стрелочки организуют новые н-пары на границах между строками в *LS*. Поскольку, во-первых, порядок следования символов в н-паре между строками однозначно задается стрелками и, во-вторых, эта н-пара начинается перед соответствующей исходящей

стрелкой, то замена этих стрелочек на символы соответствующие «концам» сохраняет каждую н-пару, и, следовательно, не изменяет их количества.

Преобразование 2. (Для ветвителей). Если в скобках ветвителей не убрались исходящие стрелки, то перенести символы событий, записанных в паре с исходящей стрелочкой в начала соответствующих строк и убрать стрелочки. Затем из создавшихся в скобках н-пар образовать строки, исключить их из скобок и включить в множество строк *LS*.

Пустые скобки в конце строк заменить символом ω_i .

Преобразование 3. Вычеркнуть все символы состояний из строк.

После выполнения преобразований 1, 2 и 3 LS содержит только последовательности событий.

Проиллюстрируем эти преобразования на примере LS^H фрагмента перегрузочного устройства транспортной системы, состоящего из 4-х механизмов: М1, М2, М3, М4 (см. раздел 5.1.)

Последовательности выполнения операций представлены следующим множеством строк ЛСПИ $LS = \{\downarrow^1 t_{17}t_1t_2t_5t_6t_7t_8t_9t_{10}\{t_{19}\uparrow^2, t_{20}\uparrow^3\};$ $\downarrow^2 t_{13}t_{14}t_{11}t_{12}t_{15}t_{16}\uparrow^4; \downarrow^3 t_5t_6t_{11}t_{12}t_7t_8\uparrow^4; \downarrow^4 t_3t_4t_{18}\uparrow^1;\},$

построенным по графу переходов автомата *H*.

Шаг 1.Вводим переменные ω_i для ветвителей: $LS = \{\downarrow^1 t_{17}t_1t_2t_5t_6t_7t_8t_9t_{10}\omega_i;$ $\downarrow^2 t_{13}t_{14}t_{11}t_{12}t_{15}t_{16}\uparrow^4; \downarrow^3 t_5t_6t_{11}t_{12}t_7t_8\uparrow^4; \downarrow^4 t_3t_4t_{18}\uparrow^1;\}$, $\omega_1 = \{t_{19}\uparrow^2, t_{20}\uparrow^3\};$

Шаг 2. Исключаем входящие стрелочки с индексами 1,2,3,4: $LS = \{t_{18}t_{17}t_1t_2t_5t_6t_7t_8t_9t_{10}\omega_i;\ t_{19}t_{13}t_{14}t_{11}t_{12}t_{15}t_{16}\uparrow^4; t_{20}t_5t_6t_{11}t_{12}t_7t_8\uparrow^4; t_{16}t_3t_4t_{18}\uparrow^1;\}.$

$\omega_1 = \{t_{19}\uparrow^2, t_{20}\uparrow^3\}$

Шаг 3. Исключаем все исходящие стрелки, которым нет ответных входящих (стрелочки с индексами 1,2,3,4): $LS = \{t_{18}t_{17}t_1t_2t_5t_6t_7t_8t_9t_{10}\omega_i;\ t_{19}t_{13}t_{14}t_{11}t_{12}t_{15}t_{16};$ $t_{20}t_5t_6t_{11}t_{12}t_7t_8; t_{16}t_3t_4t_{18};\}.$

$\omega_1 = \{t_{19}, t_{20}\}.$

Переходы с нечетными индексами в диапазоне 1÷16 - управляемые переходы (переходы соответствующие управляемым событиям), следующие за ними переходы с четными индексами в этом же диапазоне - ожидаемые (переходы соответствующие ожидаемым событиям окончания соответствующей операции).

$S^i = \left(P^i, T^i, W^i, \mu_0\right)$, где $P^i = \{p_j | j = k \div (k+3)\}$, $T^i = \{t_j | j = k \div (k+3)\}$, при этом для значения i=1,2,3,4 значение k=1,5,9,11 соответственно. $S_p = \bigcup_1^4 S^i$ W^i подобны матрицам из таблицы 2, они представлены в объединенной матрице инцидентности W_p в таблице 3.

Процедура построения матрицы *MF(g×h)* .

Обозначения, используемые в тексте процедуры. **ConstrMF** - имя процедуры; ***W,n,m ,LS, k, MF,g, h, u, r, VRD***,– формальные параметры**;** *u*:= $t_1 t_2 t_3 \ldots t_{r-1} t_r$ - строка ЛСПС, обрабатываемая процедурой (в *u* стрелочки заменены именами переходов, на которые эти стрелочки указывают в ЛСПС); *r* - длина строки *u*; *W(n×m)* – матрица инцидентности сети процесса S_p ; *W(k, ~)* =0 означает присвоение всем элементам строки *k* матрицы *W* значение 0; *u(i) i*-тый символ строки *u.*. *W(u(i)), W(u(i+1))* - векторы- столбцы матрицы *W,* SProdct(*W(u(i)), W(u(i+1))*) – скалярное произведение векторов. Определение процедуры на языке Паскаль:

```
ConstrMF (W,n,m ,LS, k, MF,g, h, u, r, VRD)
MF:=0 Комментарий: Всем элементам MF присвоить 0;
Комментарий: Цикл по строкам LS;
j=1;
While j≤k  do
Begin
u:=LS(j);
r:=Long(u)
Комментарий: Цикл по строке u;
i=1
While i≤(r-1)  do
Begin
If  SProdct ( W(u(i)), W(u(i+1)))≠ 0 go to M1;
MF(u(i),u(i+1)=1
M1: i=i+1
End
j=j+1
End.
```

Приложение 2

Алгоритма синтеза управляющей сети на языке Паскаль.

Обозначения и понятия, используемые в тексте процедуры синтеза S_c.

На начало синтеза управляющей сети Петри S_c полагаем (как и разделе 3.4), что сеть S_p уже сконструирована по LS^i и матрица инцидентности сети процесса S_p - *W(n×m)* – определена для S_p. (Столбцы сопоставлены переходам, а строки сопоставлены позициям S_p). Кроме того строки *W(n×m)* помечены также событиями из E_{uc}, входящими в LS.

По ветвителям в LS определим множества ω_k (напомним, список событий в фигурных скобках мы назвали ветвителем). Множество $\omega_k = \{e_{k1}, \dots, e_{kr}\}$ включает события, входящие в ветвитель *k*.

Определим ряд функций, используемых в изложении алгоритма синтеза .

Функция $MF(r,°)$ - ее значение для *r* - переход t_i, сопоставленное строке *r* матрицы *MF*, функция $MF(°,r)$- ее значение для *r* - переход t_j, сопоставленное столбцу *r* матрицы *MF*.

$In(M(r,°)$ – функция, значение которой *k*= $In(M(r,°)$ равно номеру столбца k матрицы *W*, сопоставленному переходу t_k , соответствующему событию строки $MF(r,°)$.

Если *k*= $In(M(°,r))$, то это означает, что столбцу *k* матрицы *W* сопоставлен переход t_k, соответствующий событию столбца $MF(°,r)$. *NUC*(*MF*,*r*) – количество 1 в столбце *r* матрицы *MF*, *NUR*(*MF*,*r*) – количество 1 в строке *r* матрицы *MF*.

NPC(*MF,r*) – номер строки , содержащей 1(элемент отличный от 0) в столбце r.

Функция – процедура *TAKE*(*A*) её значением является элемент $e \in A$, при этом после очередного обращения к *TAKE*(*A*) $A := A \backslash e$. При *A*=∅ обращение к TAKE(*A*) прекращается.

Функция – процедура *TNUR(MF,r)* используется в цикле и находит в строке r матрицы *MF* очередной элемент равный 1(отличный от 0), при этом ее значение *k* равно номеру столбца, в котором *TNUR(MF,r)* очередной раз обнаружила этот элемент в строке r, если при очередном обращении единичного элемента не обнаружено, то *TNUR(MF,r)* =0. *TNUR(MF,r)* допустимо присвоение начального значения отличного от 0.

VRD(h)- вектор строка, элементы которого являются позиции p_d, в которых осуществляется «сборка переходов» перед управляемыми переходами сети S_c (номер соответствующе строки в матрице W). Если *VRD(i)*=0, то это означает, что в управляемом переходе t_i осуществляется сборка по «И».

Текст процедуры представлен на языке Паскаль.

Sintez_2 (W,MF, Pc,Tc, n, m, g, h)
Комментарий:Цикл по столбцам (i) матрицы MF;
i=1
While (i≤h) do
Begin
If ((NUC(MF,i)=1) AND (NUR(MF, NPC(MF,i))>1))THEN
Begin VRD(i) =0; Go to M1;End; *Комментарий: Обнаружен ветвитель;*
Else
Комментарий: Введение «позиций ИЛИ» канала;
n=n+1
VRD(i) =n
t_k***= In(M(°, i)***
W(n, t_k***)=-1***
IF (MF(°, r)≠ ω_k ***) THEN GO TO M1***
Комментарий: Столбец ω_k *– это позиция начала ветвления, переход в неё будет определен позже. Здесь ниже определяется связь с переходами собственно ветвителя;.*
IF (θ(MF(°, r)) ∈ {c, uc. w}) THEN GO TO M1
Else
W(n, t_k***)=0***
Комментарий: формирование ветвителя в сети;
While ω_k***≠∅ do***
Begin
t=TAKE(ω_k***)***
W(n,t)= -1
End
M1: i=i+1
End
Комментарий: Цикл по строкам (индекс i) матрицы MF;.
While (i≤g) do
Begin
If (NUR (MF, i)>1) Go to M2
k= TNUR(MF,r)
W(VRD(k), MF(i,°)) = −1; *Комментарий: завершение простого звена;*
Go to M3 Комментарий: переход на продолжение цикла по строкам;
M2: n=n+1
W(n, MF(i,°)) = −1; *Комментарий: переход в выходную позицию;*
v=n Комментарий: запомнить номер позиции - начало группового звена;
TNUR(MF,i):=h
While TNUR(MF,i)≠0
do
Begin
k= TNUR(MF,i)
n=n+1
IF (VRD(k)≠0) THEN GO TO M4
W(n, MF(k,°))=-1
W(v, MF(k,°))=-1
GO TO M3
M4: m=m+1 *Комментарий: добавлен в СП новый переход* t_m*;*

W(n, m)=-1
W(v, m)=-1
W(VRD(k), m)=+1 *Комментарий: переход t_m связали с 2 источниками и приемником;*
End
M3: i=i+1
End