УДК 51.681.3

С.Л. КРЫВЫЙ, А.В. ЧУГАЕНКО

ФОРМАЛЬНЫЕ МЕТОДЫ АНАЛИЗА ДИСКРЕТНЫХ СИСТЕМ С ИСПОЛЬЗОВАНИЕМ ЯЗЫКА СПЕЦИФИКАЦИЙ

Ключевые слова: *сети Петри, язык MSC, верификация.*

ВВЕДЕНИЕ

Современные компьютерные системы эволюционируют в сторону усложнения своей структуры, и поэтому в процессе их разработки и проектирования возникает потребность в использовании формальных методов верификации шагов проектирования, в обосновании принятых решений, проверке выполнимости спецификаций и т.п. Исследования в этой области привели к тому, что верификация компьютерных систем выделилась в отдельную проблему. За последнее двадцатилетие имеется существенный прогресс, обусловленный прежде всего возникновением и быстрым развитием метода проверки на модели (model checking). Благодаря этому методу стала возможной верификация реактивных систем, которая, в частности, сводится к проверке выполнимости спецификаций, описывающих ожидаемые поведения системы. При этом спецификации записываются в определенном формальном логическом языке, в то время как сама система выступает в качестве модели, на которой проверяется выполнимость ее спецификаций.

Настоящая статья относится к области верификации, в которой предлагается записывать спецификации программной системы в одном из языков темпоральной логики, а саму систему моделировать средствами языка MSC (Message Sequence Charts) [1], т.е. представлять в виде MSC-диаграмм. Исследование структурных свойств MSC-диаграмм, представляющих данную программную систему, выполняется путем конвертации MSC-диаграммы в сеть Петри (СП), а проверка выполнимости спецификации осуществляется построением транзиционной системы для полученной в результате конвертации СП. В частности, описывается алгоритм конвертации MSC-диаграмм в СП, который применим практически ко всему множеству операторов, включенных в стандарт этого языка (с использованием строгой последовательной композиции диаграмм). Устанавливается событийная эквивалентность исходных MSC-описаний и полученной по ним СП. Детальный анализ языка MSC показал, что конвертируемый вариант этого языка по выразительным свойствам эквивалентен ординарным СП (язык MSC может использоваться как надстройка над произвольным языком программирования, за счет которого достигается его алгоритмическая полнота). Данная статья примыкает к работам по конвертации конструкций языка MSC в СП [2, 3] и завершает цикл работ [4–8].

Все вышесказанное составляет единый технологический процесс исследования свойств исходной модели системы.

1. НЕОБХОДИМЫЕ ОПРЕДЕЛЕНИЯ И ПОНЯТИЯ

1.1. Краткие сведения по сетям Петри [9]. Определение 1. Сеть Петри — это упорядоченная пятерка (P,T,F,W,M_0), где P — непустое конечное множество, элементы которого называются местами сети; T — непустое конечное множество, элементы которого называются переходами сети; $F \subseteq P\times T \cup T\times P$ — отношение инцидентности между переходами и местами; W: $F \to \mathbb{N}\setminus\{0\}$ и $M_0 : P \to \mathbb{N}$ — две функции, называемые кратностью дуг и начальной разметкой соответственно. При этом выполняются следующие условия:

• $P \cap T = \varnothing$ (множества мест и переходов не пересекаются);

• $\forall x \in P \cup T \;\; \exists y \in P \cup T$: $xFy \vee yFx$ (любой элемент сети инцидентен хотя бы одному элементу другого типа).

Для удобства множества мест и переходов будем считать строго упорядоченными, т.е. $P = (p_1, \ldots, p_n)$, $n = |P|$, $T = (t_1, \ldots, t_m)$, $m = |T|$.

Определение 2. Сеть Петри, кратность всех дуг которой равна единице, называется ординарной.

Графически сеть Петри представляется в виде двудольного ориентированного графа с двумя типами вершин: местами и переходами. Места изображаются в виде кружков, а переходы — в виде прямоугольников. Дуги графа соединяют инцидентные вершины. Если кратность какой-либо дуги больше единицы, то ее обозначают числом, стоящим рядом с дугой, или (для небольшой кратности) заменяют соответствующим пучком дуг. Разметку изображают цифрой внутри кружка или (для небольших значений) соответствующим количеством точек внутри кружка. Точки, обозначающие разметку сети, также называют фишками.

Разметку сети N представляет функция $M: P \to \mathbb{N}$. Учитывая, что множество мест сети упорядочено, разметку можно задать вектором чисел $M = (m_1, \ldots, m_n)$, где $m_i = M(p_i)$, $i = 1, \ldots, n$.

Введем понятие функции инцидентности f: $P \times T \cup T \times P \to \mathbb{N}$:

$$f(x, y) = \begin{cases} n, & xFy \wedge (W(x, y) = n), \\ 0, & \neg(xFy). \end{cases}$$

В этом случае каждому переходу $t \in T$ можно сопоставить два вектора:

$$^{*}F(t) = (b_1, \ldots, b_n), \; b_i = f(p_i, t),$$

$$F(t)^{*} = (b_1, \ldots, b_n), \; b_i = f(t, p_i),$$

при этом переход t может сработать при разметке M тогда и только тогда, когда $M \geq {}^{*}F(t)$ и его срабатывание переведет разметку M в разметку M' по следующему правилу:

$$M' = M - {}^{*}F(t) + F(t)^{*}.$$

В случае, если при данной разметке в сети может сработать несколько переходов, то они срабатывают в произвольном порядке независимо один от другого.

Если переход t может сработать при разметке M и в результате получится разметка M', то говорят, что разметка M' непосредственно достижима из M путем срабатывания перехода t ($M \xrightarrow{t} M'$).

Отношение непосредственной достижимости можно обобщить. Считается, что разметка M' достижима из разметки M, если существует последовательность переходов $\tau = t_1 \ldots t_k$ такая, что $M \xrightarrow{t_1} M_1 \xrightarrow{t_2} \ldots \xrightarrow{t_k} M'$ ($M \xrightarrow{\tau} M'$).

Обозначим $R(N, M)$ множество разметок в сети N, достижимых из разметки M. Множество $R(N) = R(N, M_0)$ означает множество достижимых разметок сети N из начальной разметки M_0. Очевидно, что $M \in R(N, M)$ и $M_0 \in R(N)$.

Пусть T^{*} — множество всех последовательностей возможных переходов в сети N, тогда свободным языком сети N называют множество $L(N) = \{\tau \in T^{*} \mid \exists M \in R(N)$: $M_0 \xrightarrow{\tau} M\}$, т.е. множество всех возможных последовательностей срабатываний переходов сети N, начиная с начальной разметки.

Определение 3. Помеченная сеть Петри — это пара (N, Σ), где N — сеть Петри, а Σ: $T \to A$ — помечающая функция над некоторым алфавитом A. Если Σ — частичная функция, то переходы, на которых она не определена, помечают специальным символом λ и называют λ-переходами.

Помечающая функция естественным образом расширяется на последовательности срабатываний переходов:

$$\Sigma(\tau t)=\begin{cases}\Sigma(\tau)\Sigma(t), & \text{если } \Sigma(t) \text{ определено,}\\ \Sigma(\tau) & \text{в противном случае,}\end{cases}$$

где t — переход и τ — последовательность переходов.

Определение 4. Если $\tau \in T^*$ — последовательность срабатываний переходов сети N, а (N,Σ) — помеченная сеть, то $\Sigma(\tau)\in A^*$ называют помечающей последовательностью, где A^* — множество всех слов в алфавите A. Если $L(N)$ — свободный язык сети N, то множество $\{\Sigma(\tau)|\ \tau \in L(N)\}$ называют префиксным языком помеченной сети (N,Σ).

Место p в сети N называют ограниченным, если существует n такое, что $\forall M \in R(N)$: $M(p)\le n$. Сеть Петри N называется ограниченной, если все ее места ограничены. В случае, если $n=1$, то места СП называются безопасными, а сама сеть — безопасной.

Сеть N называется консервативной, если суммарное число фишек в сети остается неизменным при любых срабатываниях переходов, т.е. $\forall M_1, M_2 \in R(N)$: $\sum\limits_{p\in P} M_1(p)=\sum\limits_{p\in P} M_2(p)$.

Переход t в сети N называется потенциально живым при разметке M, если $\exists M' \in R(N,M)$: $M' \ge^* F(t)$. Если $M=M_0$, то t в сети N называется потенциально живым. Переход t — мертвый при разметке M, если он не является потенциально живым при M. Переход t — мертвый, если он мертвый при любой достижимой в сети разметке.

Переход t в сети N называется живым, если $\forall M \in R(N)\ \exists M' \in R(N,M)$: $M' \ge^* F(t)$. Переход t — потенциально мертвый, если $\exists M \in R(N)$: $\forall M' \in R(N,M)$: $M' \not\ge^* F(t)$. Разметка M в этом случае называется t-тупиковой. Если разметка является t-тупиковой для всех переходов, ее называют тупиковой. Сеть называется живой, если все ее переходы живые.

1.1.1. Покрывающее дерево СП. Покрывающее дерево СП — это конечный ориентированный ациклический граф, вершинами которого являются разметки СП, достижимые из начальной разметки и удовлетворяющие определенным условиям. Эти условия проверяются в процессе построения покрывающего дерева и описаны в виде следующего алгоритма.

1. Изначально дерево покрытия состоит из одного корня, который содержит разметку M_0 и не имеет дуг.

2. Пусть M — вершина дерева, еще не объявленная листом, из которой пока не исходит ни одна дуга. Для разметки M возможны следующие случаи:

а) на пути из корня дерева в вершину M уже существует вершина M' такая, что $M=M'$. В этом случае вершина M объявляется листом;

б) на пути из корня дерева в вершину M существуют вершины $M'_1,\dots,M'_n$ такие, что $M'_i<M$, $i=1,\dots,n$. В этом случае в разметке M каждый элемент, соответствующий месту p, для которого $\exists i$: $M'_i(p)<M(p)$, заменяется на ω и вершина объявляется ω-листом;

в) при разметке M не может сработать ни один из переходов сети, в этом случае вершина M объявляется листом.

В остальных случаях для каждого перехода t, который может сработать при этой разметке, создается новая вершина M' такая, что $M \xrightarrow{t} M'$, и M соединяется с ней дугой, помеченной символом t. Покрывающее дерево позволяет исследовать следующие свойства сети.

1. Сеть является ограниченной, если ее покрывающее дерево не содержит ω-листов.

2. Сеть является безопасной, если все вершины в ее дереве покрытия содержат разметки, состоящие лишь из нулей и единиц.

3. Если хотя бы в одной из вершин покрывающего дерева позиция в разметке, соответствующая месту p, содержит число, большее нуля, место p может получить фишку в процессе функционирования сети.

4. Если символ перехода t помечает хотя бы одну дугу покрывающего дерева, переход t является потенциально живым.

1.1.2. Уравнение состояния СП и инварианты СП. Популярным методом анализа сети является также метод вычисления инвариантов СП путем решения ее уравнения состояния.

Определение 5. Уравнением состояния СП для произвольной разметки M [10] называют диофантово уравнение вида $Ax = M - M_0$, где $A = |a_{i,j}|$ — матрица инцидентности этой сети, $a_{i,j} = f(t_j, p_i) - f(p_i, t_j)$.

В [10] показано, что если некая разметка M достижима из начальной, то соответствующее уравнение состояния $Ax = M - M_0$ имеет решение. Утверждение о том, что если уравнение состояния $Ax = M - M_0$ имеет решение, то разметка M достижима из начальной, в общем случае неверно, но если уравнение состояния несовместно, то разметка не достижима из начальной.

Определение 6. T-инвариантами СП называются положительные целочисленные решения системы $Ax = 0$; S-инвариантами — положительные целочисленные решения системы $A^t x = 0$, где A^t — транспонированная матрица инцидентности.

Вектор x является T-инвариантом тогда и только тогда, когда существует разметка M_0 и последовательность переходов σ, ведущая от M_0 к M_0 такая, что $\sigma = x$. Следовательно, неформально T-инварианты соответствуют совокупности срабатывающих переходов СП, сохраняющих разметку. Если хотя бы в одном из T-инвариантов переходу соответствует ненулевая координата, этот переход является потенциально живым в исходной СП.

Для решения уравнения состояния СП (построения множества ее инвариантов) может быть использован TSS-алгоритм построения усеченного множества решений системы линейных однородных диофантовых уравнений. Данный алгоритм подробно описан и обоснован в [11], а особенности его реализации — в [12].

1.1.3. Упрощение СП. Для ускорения анализа СП ее необходимо (если возможно) упростить, т.е. подвергнуть преобразованиям для сокращения количества мест, переходов или дуг при сохранении исследуемых свойств исходной сети. В частности, преобразование, показанное на рис.1,*а* сохраняет язык сети, а показанные на рис. 1,*б–ж* (введены в [10]), сохраняют свойства живости, ограниченности и безопасности.

1.2. Краткие сведения о языке MSC. Стандарт языка MSC приведен в [1], а его операционная семантика — в приложении к стандарту [13]. Ниже приведены выдержки из стандарта в определенном объеме.

MSC — это язык для описания взаимодействия между независимыми сущностями, которые обмениваются сообщениями. Он имеет как текстовое, так и графическое представление, что делает его удобным и для ручной, и для машинной обработки. Выполнение документа MSC представляет собой последовательность таких событий, как посылка или прием сообщения, выполнение действия и т.п. Такая последовательность событий, происходящих при выполнении диаграммы или документа MSC, называется трассой. Поскольку язык MSC содержит итерацию, трассы могут быть и бесконечными. Каждая трасса описывает определенный сценарий работы исходной системы. Можно говорить о статически (или структурно) возможных трассах, которые определяются только структурой MSC, и динамически возможных трассах, на которые оказывает влияние также и состояние моделируемой

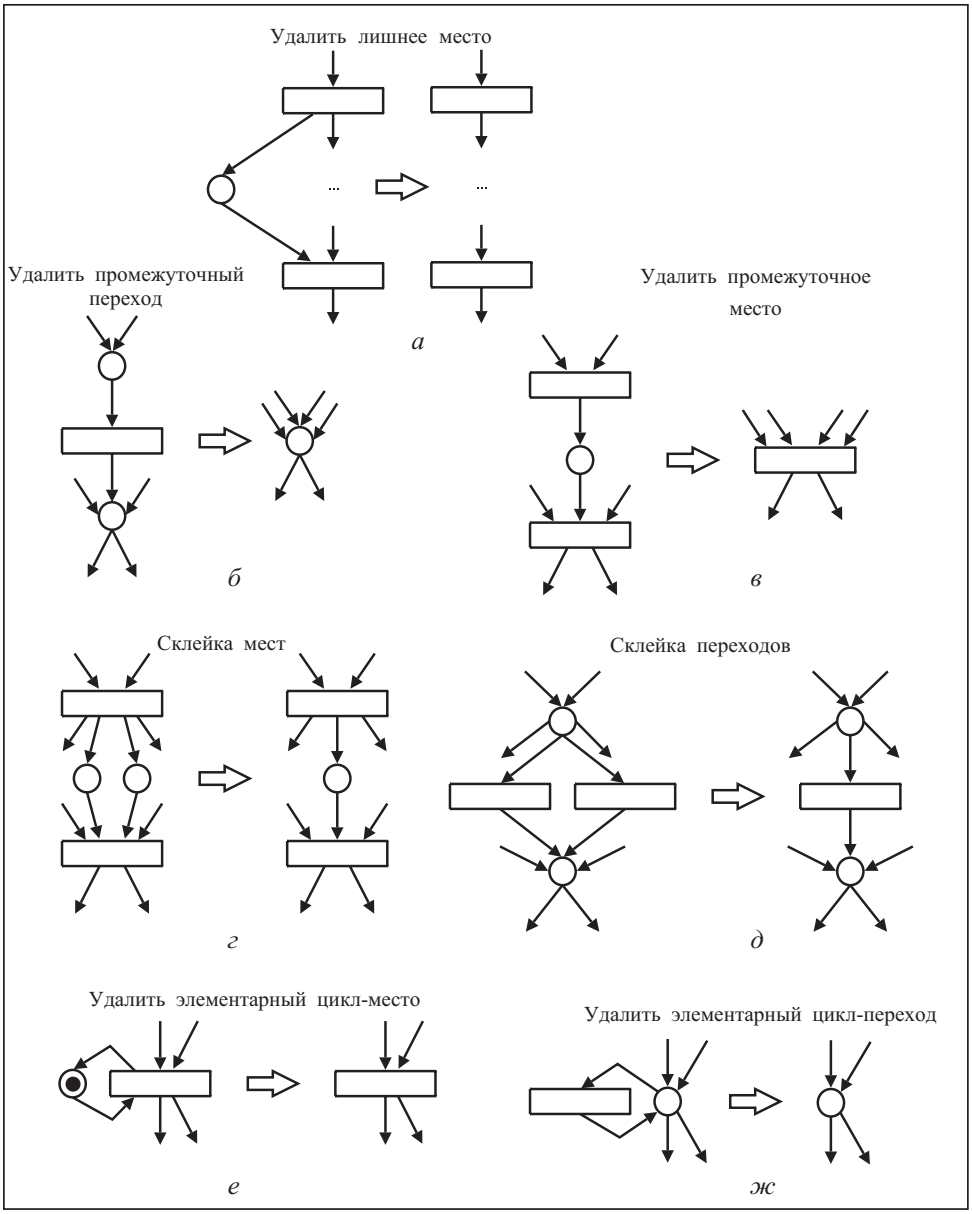

Рис. 1. Типы упрощения СП

системы через значения условий. Поскольку любая динамически возможная трасса должна быть также и статически возможной, то множество динамических трасс есть подмножеством множества статических. Приведенный в статье алгоритм рассматривает только множество статически возможных трасс, т.е. структурные свойства документа MSC.

Часто используемые элементы языка MSC описаны ниже (графический вариант). Полный список элементов MSC и их текстовой вариант приведены в [1].

Документ MSC (MSC document) (рис. 2, *а*), являющийся наиболее крупным объектом MSC, описывает поведение системы в целом. Он определяет присутствующие в системе сущности и включает в себя диаграммы MSC и HMSC (High-level Message Sequence Chart).

Диаграмма MSC (MSC diagram) (рис. 2, *в*) описывает некоторый фрагмент поведения системы и представляет собой сценарий взаимодействия между сущностями (см. ниже). Не существует каких-либо правил относительно того, что в данном

случае считать фрагментом поведения, поэтому разбиение исходной системы на диаграммы MSC выполняется разработчиком, исходя из удобства работы. Название «диаграмма MSC» запишем просто как MSC. Диаграмма однозначно определяется в пределах документа своим именем.

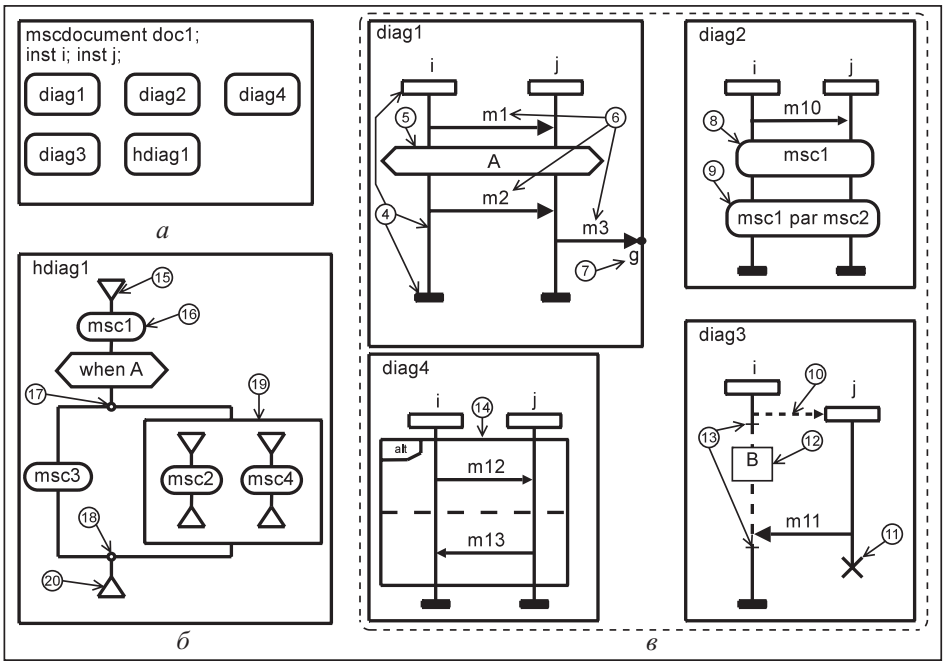

Рис. 2. Элементы языка MSC

Сущность (instance), обозначенная как (4) на рис. 2, представляет агента взаимодействия в документе MSC. Взаимодействие между сущностями описывается в диаграммах MSC, но сами сущности определяются на уровне документа MSC. Они считаются действующими независимо и имеющими независимое дискретное время, при этом сущность может существовать как на протяжении всего времени действия документа, так и динамически создаваться и уничтожаться (см. ниже). Сущность однозначно определяется в пределах документа своим именем, а также своим экземпляром, если он используется. Экземпляры сущностей (равно как и сообщений) используют для того, чтобы представить несколько агентов с одинаковым именем. С сущностью связывается ее ось (instance line), на которой располагаются связанные с этой сущностью (разделяемые этой сущностью) события и другие элементы MSC.

Сообщение (message) (рис. 2, (6)) — основной способ взаимодействия между сущностями. Сообщение имеет имя, а также может иметь имя экземпляра (instance) (чтобы различать сообщения с одинаковым именем) и прикрепленные данные. Сообщение представляется двумя событиями: посылкой сообщения и приемом сообщения, причем прием сообщения происходит строго после его посылки. Если сообщение состоит только из приема или только из посылки, оно называется неполным (incomplete). Посыл сообщений может происходить как между сущностями в пределах одной диаграммы (рис. 2, сообщения m1, m2), так и между разными диаграммами с помощью шлюзов (рис. 2, сообщение m3). В пределах диаграммы MSC сообщение однозначно определяется своим именем, экземпляром, приемным и передающим адресами. Адрес может быть именем сущности, шлюза или указывать на неполноту сообщения.

Шлюз (gate) (рис. 2, (7)) используется для обмена сообщениями между диаграммами MSC или для установки явного порядка между элементами, находящимися в разных диаграммах MSC.

Условие (condition) (рис. 2, (5)) используется для ограничения множества допустимых трасс. Условия бывают двух типов — установочные (setting conditions) и ограждающие (guarding conditions). Первые фиксируют некоторое глобальное состояние системы на момент своего выполнения, а вторые регулируют возможность выполнения трасс в зависимости от состояния системы. Следует заметить, что язык, на котором записывается состояние системы для условий, не определен и выбирается разработчиками. Свойством условий является также их синхронизирующее действие: если две сущности разделяют одно условие, то любое сообщение между ними, посылаемое до этого условия, должно быть принято также до условия. Например, сообщение m1 на рис. 2 посылается и принимается до условия *A*.

Создание сущности (instance creation) (рис. 2, (10)) является специальным вариантом сообщения, в результате посылки которого создается новая сущность.

Удаление сущности (instance stop) (рис. 2, (11)) останавливает ее выполнение. Никакое событие, разделяющее сущность, не может быть выполнено после ее удаления.

Действие (action) (рис. 2, (12)) — событие, связанное с выполнением какого-либо программного действия. Содержимое действия задается разработчиком в произвольной форме.

Область произвольного выполнения (coregion) (рис. 2, (13)) отмечает на оси сущности отрезок, в пределах которого события выполняются в произвольном порядке (действие *B* и прием сообщения m11 на рис. 2). В приложении к стандарту [13] рекомендуется интерпретировать область произвольного выполнения как параллельную композицию всех находящихся в ней событий.

Последовательность событий в диаграмме MSC определяется их явным и неявным упорядочением.

Неявное упорядочение (implicit ordering), определяющее порядок выполнения событий в пределах диаграммы, выполняется согласно следующим правилам:

— если два события разделяют одну и ту же сущность и не находятся в одной области произвольного выполнения, верхнее событие наступает строго раньше нижнего;

— прием сообщения происходит строго позже его посылки;

— создание сущности происходит строго позже события «создать сущность».

Явное упорядочение (explicit ordering) позволяет упорядочить события диаграммы MSC в дополнение к описаному выше неявному упорядочению (implicit ordering) и изображается пунктирной стрелкой, ведущей от предшествующего события непосредственно к последующему (на рисунке отсутствует).

Ссылка на диаграмму MSC (MSC reference) — конструкция, позволяющая ссылаться одной диаграмме MSC или HMSC на другую диаграмму. При этом управление передается диаграмме, на которую ссылаются, и возвращается по окончании выполнения этой диаграммы. Список сущностей, разделяемых ссылкой на MSC, должен совпадать со списком сущностей, присутствующих в диаграмме, на которую ссылаются (рис. 2, (8)). Из ссылок на диаграммы MSC могут составляться ссылочные выражения.

Ссылочное выражение (reference expression) — выражение над ссылками на MSC, полученное с помощью операторов alt, par, seq, loop, opt и exc. Размещение и использование ссылочных выражений совпадают с таковыми для ссылок на MSC (рис. 2, (9)). Ссылку на MSC можно рассматривать как элементарное ссылочное выражение. Операторы в ссылочном выражении согласно стандарту имеют следующее значение:

• alt — альтернативное выполнение, выполняется только один операнд;

• par — параллельное выполнение, операнды выполняются параллельно, в текущей семантике можно рассматривать как произвольное чередование событий из разных операндов;

• seq — последовательное выполнение, операнды выполняются последовательно;

• loop — итерация, операнд выполняется несколько раз, в текущей семантике итерация рассматривается как последовательное выполнение нескольких экземпляров операнда;

• opt — необязательное (опциональное) выполнение, операнд выполняется или пропускается, является модификацией оператора alt, для которой второй операнд пустой;

• exc — исключение, выполняется либо операнд, либо оставшаяся часть диаграммы, является модификацией оператора alt, для которой второй операнд представляет оставшуюся часть диаграммы.

Кроме этих операторов в ссылочном выражении используется слово empty, которое рассматривается как ссылка на пустую диаграмму.

Встроенное выражение (inline expression) аналогично ссылочному, но вместо ссылок на диаграммы в выражение включаются непосредственно их тела (таким образом, встроенное выражение можно рассматривать как ссылочное выражение над анонимными MSC) (рис. 2, (14)). При построении встроенных выражений используются те же операторы, что и в ссылочных выражениях.

Диаграмма HMSC (высокоуровневая MSC, HMSC) (рис. 2, *б*) описывает взаимодействие отдельных диаграмм MSC в рамках документа MSC. Она представляет собой ориентированный граф (направление ребер, идущих сверху вниз, обычно стрелками не обозначается). Типы вершин HMSC:

• старт (start, start symbol) — точка начала выполнения документа; в каждой диаграмме HMSC есть ровно одна такая вершина (рис. 2, (15));

• стоп (end, end symbol) — точка завершения выполнения документа (рис. 2, (20));

• ссылочное выражение (аналогично используемому в MSC, рис. 2, (16));

• условие (аналогично используемому в MSC);

• точка соединения (connection point) — в зависимости от направления ребер образует композицию альтернативного выполнения (рис. 2, (17)) либо точку объединения разных ветвей альтернативного выполнения (рис. 2, (18));

• параллельное выполнение (parallel frame) — конструкция, содержащая несколько HMSC, которые выполняются параллельно (рис. 2, (19)).

2. АЛГОРИТМ ПРЕОБРАЗОВАНИЯ MSC-ДИАГРАММ В СП

Алгоритм основан на следующих соображениях. Согласно семантике MSC ([13] с учетом расширения, приведенного в [5]) документ MSC можно рассматривать как множество событий, соединенных последовательной, параллельной, альтернативной композициями. Тип используемой композиции определяется управляющими конструкциями MSC. Таким образом, алгоритм конвертации MSC-диаграмм в СП состоит в симуляции этих управляющих конструкций с помощью СП. В результате получена СП, которая генерирует язык, являющийся событийно эквивалентным множеству трасс, генерируемых исходным документом MSC.

2.1. Обозначения и начальные допущения. Предположим, что входной документ MSC является синтаксически верным и удовлетворяет статическим требованиям, указанным в [1]. Тела встроенных выражений рассматриваются как анонимные диаграммы MSC (в соответствии с [13]). Согласно статическим требованиям [1] предположим, что все события документа MSC уникальны (при необходимости этого можно достичь соответствующим переименованием событий).

В отличие от подхода в [13] мы интерпретируем оператор seq как строгую последовательную композицию диаграмм MSC (т.е. если две диаграммы MSC соединены последовательно, то все события первой диаграммы будут выполнены до того, как начнет выполняться первое событие второй диаграммы). Полученная в результате выполнения представленного алгоритма СП представляет только структуру и статические свойства документа MSC. Таким образом, информация, содержащаяся в элементах типа условие, игнорируется.

При описании алгоритма и его доказательстве используются следующие обозначения:

1) события MSC (включая элемент типа условие), как и соответствующие им фрагменты СП, обозначаем ev_i;

2) последовательности и неупорядоченные группы событий MSC, как и соответствующие им фрагменты СП, обозначаем sev_i;

3) фрагменты СП, используемые для представления управляющих конструкций MSC, называем конструкциями;

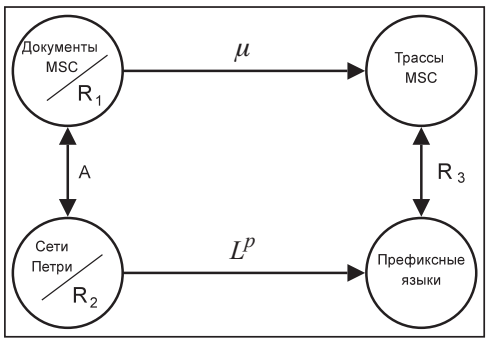

Рис. 3. Схема отношения эквивалентности документов MSC и сетей Петри

4) в полученной СП метки переходов будут размещаться рядом со знаком перехода, в то время как дополнительная служебная информация будет помещаться внутри значков перехода или места;

5) $L^p(a)$ — префиксный язык размеченной сети a, который назовем языком СП [9];

6) $\mu(b)$ — множество трасс, генерируемых документом MSC b.

В описании алгоритма используется изложенное ниже понятие событийной эквивалентности.

Рассмотрим схему отношений эквивалентности, используемых в алгоритме (рис. 3).

• Отношение эквивалентности R_1 индуцируется функцией μ и определено на множестве документов MSC: $pR_1q \Leftrightarrow \mu(p) = \mu(q)$. Отношение R_1 объединяет в один класс эквивалентности документы MSC, генерирующие одинаковые множества трасс.

• Отношение эквивалентности R_2 индуцируется функцией L^p и определено на множестве сетей Петри: $pR_2q \Leftrightarrow L^p(p) = L^p(q)$. Отношение R_2 объединяет в один класс эквивалентности сети Петри, имеющие одинаковый префиксный язык.

• Отношение R_3 устанавливает взаимно однозначное соответствие между множествами трасс документов MSC и префиксными языками сетей Петри, построенных с помощью рассматриваемого алгоритма. Отношение R_3 определено как $pR_3q \Leftrightarrow (\forall w_1 \in p\ \ \exists w_2 \in q\colon\ w_1 = r(w_2)) \wedge (\forall w_2 \in q\ \ \exists w_1 \in p\colon\ w_1 = r(w_2))$, где $r(w)$ — функция переименования.

Определение 7. Документ MSC a и сеть Петри b событийно эквивалентны, если выполняется условие $\mu(a) R_3 L^p(b)$.

2.2. Описание алгоритма. Построение СП из исходного документа MSC включает следующие шаги.

1. Каждая диаграмма MSC (включая анонимные диаграммы, находящиеся в телах встроенных выражений) переводится в конструкцию, показанную на рис. 4,*а*, где $sev_1, \ldots, sev_n$ — последовательности событий и конструкций, принадлежащих этой диаграмме. Если документ MSC содержит несколько ссылок на одну и ту же диаграмму, то каждая из них рассматривается в отдельности, а при необходимости создается несколько ее копий.

2. Для каждого события MSC ev_i строится $Prev(ev_i)$ — множество событий и конструкций (включая пропущенные сквозь шлюзы), которые непосредственно предшествуют данному событию (алгоритм построения множества $Prev(ev_i)$ приведен ниже). Затем ev_i переводится в конструкцию, показанную на рис. 4,*б*, где $n = |Prev(ev_i)|$, и входные места $p_1, \ldots, p_n$ соединяются с соответствующими событиями и конструкциями из $Prev(ev_i)$.

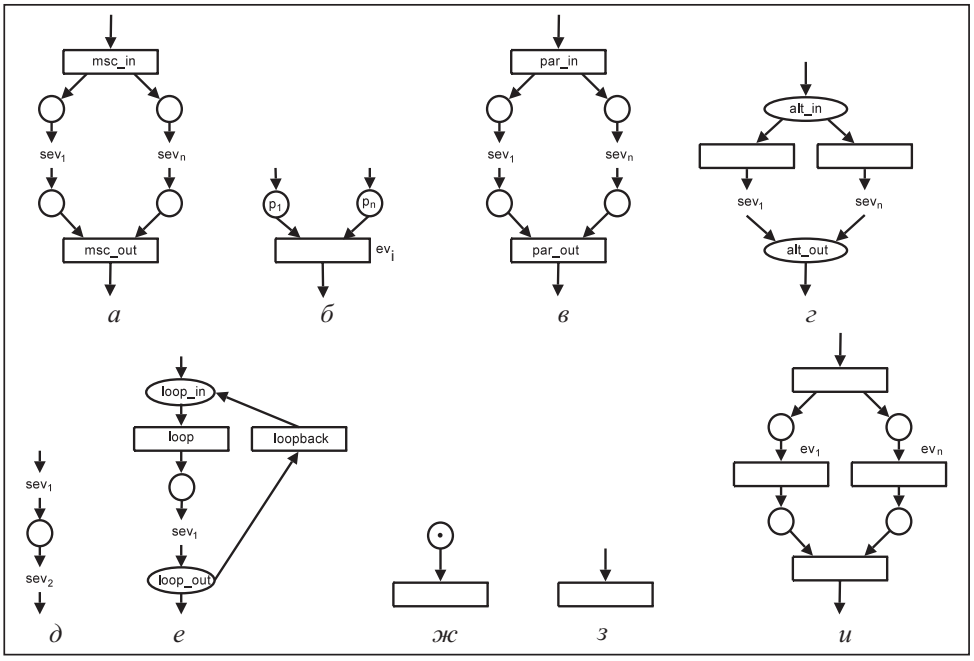

Рис. 4. Фрагменты СП, соответствующие конструкциям MSC

3. Параллельная композиция диаграмм MSC (представлена конструкцией par в ссылочном либо встроенном выражении MSC или конструкцией parallel frame в HMSC) переводится в конструкцию, показанную на рис. 4,*в*, где n — количество альтернатив в композиции.

4. Альтернативная композиция диаграмм MSC (представлена конструкциями alt, opt, exc в ссылочном либо встроенном выражении MSC или расщеплением линии в HMSC) переводится в конструкцию, показанную на рис. 4,*г*, где n — количество альтернатив в композиции.

5. Последовательная композиция диаграмм MSC (представлена конструкцией seq в ссылочном выражении MSC или линией в HMSC) переводится в конструкцию, показанную на рис. 4,*д*.

6. Итерация диаграммы MSC (представлена конструкцией loop в ссылочном или встроенном выражении MSC) рассматривается как вариант альтернативной композиции и переводится в конструкцию, показанную на рис. 4,*е*.

7. Последние элементы и конструкции диаграммы MSC (кроме события stop) соединяются с соответствующей конструкцией msc_out.

8. Последние элементы и конструкции блоков, связанных с ссылочными или встроенными выражениями (кроме события stop) соединяются с соответствующей конструкцией _out.

9. Элемент HMSC start переводится в конструкцию, показанную на рис. 4,*ж*, элемент HMSC stop переводится в конструкцию, показанную на рис. 4,*з*.

10. Элемент MSC coregion рассматривается как параллельная композиция (в соответствии с [13]) и переводится в конструкцию, показанную на рис. 4,*и*.

2.3. Построение множества $Prev(ev_i)$. Для каждого события ev_i строится множество $Prev(ev_i)$, которое состоит из следующих событий и конструкций.

1. События, имеющие общую с ev_i сущность и расположенные непосредственно перед ev_i по этой сущности.

2 Конструкция msc_in этой диаграммы, если ev_i находится в начале диаграммы и не является событием типа create in.

3. Событие message out, create out, call out или reply out, если ev_i принадлежит типам message in, create in, call in или reply in соответственно.

4. События, которые расположены непосредственно перед ev_i согласно явному упорядочению, выраженному конструкциями before или after.

Чтобы уменьшить размер результирующей СП, из множества $Prev(ev_i)$ можно исключить элементы, которые не удовлетворяют отношению доминирования относительно порядка событий и конструкций. Заметим, что эта операция не влияет на язык, генерируемый результирующей СП, а только уменьшает ее размер (соответствует упрощению СП, показанному на рис. 1,*а*.

Определение 8. Множеством $Prev(ev_i)^* = Prev(ev_i) \cup Prev(Prev(ev_i)) \cup \ldots$ будем называть транзитивное замыкание $Prev(ev_i)$ относительно предшествования.

3. ОБОСНОВАНИЕ АЛГОРИТМА КОНВЕРТАЦИИ

Пусть S_1 — документ MSC, а $S_2 = A(S_1)$ — сеть Петри, построенная по этому документу соответственно с вышеописанным алгоритмом. Покажем, что $r(\mu(S_1)) = L^p(S_2)$, доказав включения $r(\mu(S_1)) \supseteq L^p(S_2)$ и $r(\mu(S_1)) \subseteq L^p(S_2)$.

3.1. Доказательство включения $r(\mu(S_1)) \subseteq L^p(S_2)$. Определение 9. Документ MSC будем называть ациклическим, если в нем отсутствуют управляющие конструкции loop и его HMSC представляют ациклические графы.

Легко увидеть, что все трассы, соответствующие ациклическому документу MSC, конечны и их количество также конечно.

Теорема 1. Для ациклического документа MSC сеть Петри, построенная вышеприведенным алгоритмом, генерирует язык, включающий множество статических трасс, которые генерируются исходным документом MSC.

Доказательство теоремы выполним методом математической индукции по количеству событий n в исходном документе MSC.

Базис индукции. Документ MSC с одним событием изображен на рис. 5,*а*, где ev_1 — некоторое событие (на диаграмме изображено использован штриховым эллипсом). Множество трасс такого документа включает только одну трассу $\{ev_1\}$. СП, построенная для такого документа, показана на рис. 5,*б*, откуда видно, что ее язык состоит из одного слова (ev_1); таким образом, при $n = 1$ условие теоремы выполняется.

Шаг индукции. Предположим, что имеется документ MSC, состоящий из n событий, и соответствующая ему СП P_n, для которой выполняется условие теоремы. Рассмотрим новый документ MSC, полученный вследствие добавления нового события ev_{n+1} к исходному документу, и соответствующую ему СП P_{n+1}. Каждое событие ev_i, $i = \overline{1,n}$, из исходного документа находится в одном из трех возможных отношений к ev_{n+1}.

— Последовательное выполнение: $ev_i \ldots ev_{n+1}$ — в множестве трасс документа имеются слова вида $(\ldots ev_i \ldots ev_{n+1} \ldots)$ и нет слов вида $(\ldots ev_{n+1} \ldots ev_i \ldots)$.

— Параллельное выполнение (чередование): $ev_i \,||\, ev_{n+1}$ — в множестве трасс документа имеются слова как вида $(\ldots ev_i \ldots ev_{n+1} \ldots)$, так и вида $(\ldots ev_{n+1} \ldots ev_i \ldots)$.

— Альтернативное выполнение (конфликт): $ev_i \# ev_{n+1}$ — в множестве трасс документа имеются слова вида $(\ldots ev_i \ldots)$ или $(\ldots ev_{n+1} \ldots)$ и не имеется слов вида $(\ldots ev_i \ldots ev_{n+1} \ldots)$ или $(\ldots ev_{n+1} \ldots ev_i \ldots)$.

Покажем, что для каждого из перечисленных случаев язык сети P_{n+1} включает множество статических трасс, генерируемых исходным документом MSC.

1. Последовательное выполнение. Предположим, что имеет место ситуация $ev_i \ldots ev_{n+1}$. Рассмотрим возможные случаи последовательного выполнения событий ev_i и ev_{n+1} в документе MSC.

1^o. Как ev_i, так и ev_{n+1} принадлежат одной и той же диаграмме MSC, разделяют между собой одну сущность и не находятся в области произвольного выполнения. Кроме того, событие ev_i расположено непосредственно перед ev_{n+1} по этой сущности. Тогда $ev_i \in Prev(ev_{n+1})$ по построению $Prev(ev_{n+1})$ и этим событиям соответствует фрагмент СП, показанный на рис. 5,*в*. Таким образом, в $L^p(P_{n+1})$ бу-

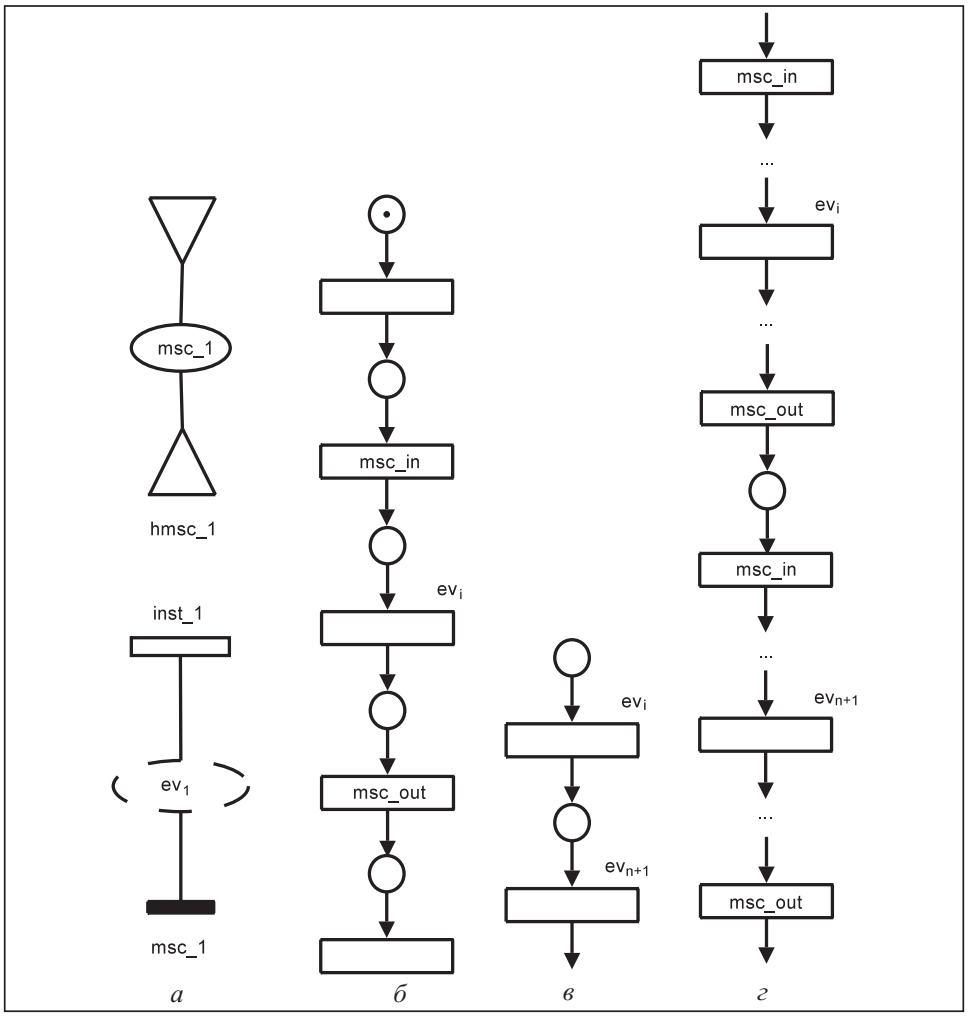

Рис. 5. Документ MSC с одним событием (*а*) и фрагменты СП для последовательной композиции (*б–г*)

дут присутствовать слова вида $(\dots ev_i ev_{n+1} \dots)$ и не будет слов вида $(\dots ev_{n+1} \dots ev_i \dots)$. В данном случае условие теоремы выполняется.

2^o. События ev_i и ev_{n+1} принадлежат разным диаграммам MSC, которые соединены последовательно. Соответствующий фрагмент СП показан на рис. 5,*г*. Таким образом, в $L^p(P_{n+1})$ будут присутствовать слова вида $(\dots ev_i \dots ev_{n+1} \dots)$ и не будет слов вида $(\dots ev_{n+1} \dots ev_i \dots)$. В данном случае условие теоремы выполняется.

3^o. Событие ev_i является посылкой сообщения, а ev_{n+1} — приемом этого сообщения. Тогда $ev_i \in Prev(ev_{n+1})$ по построению множества $Prev(ev_{n+1})$, и приходим к ситуации, описанной в п. 1^o.

4^o. Событие ev_i является событием типа создать сущность, а ev_{n+1} — созданием этой сущности. Тогда $ev_i \in Prev(ev_{n+1})$ по построению множества $Prev(ev_{n+1})$, и приходим к ситуации, описанной в п. 1^o.

5^o. События ev_i и ev_{n+1} упорядочены в соответствии с явным порядком, при этом ev_i непосредственно предшествует ev_{n+1}. Тогда $ev_i \in Prev(ev_{n+1})$ по построению множества $Prev(ev_{n+1})$, и приходим к ситуации, описанной в п. 1^o.

6^o. События ev_i и ev_{n+1} не упорядочены непосредственно, но существует последовательность $ev_1', \dots, ev_k'$ такая, что $ev_i \to ev_1' \to \dots \to ev_k' \to ev_{n+1}$, где знак $\to$ соответствует одному из отношений, описанных в пп. 1^o–3^o. Тогда в соответствии с выше-

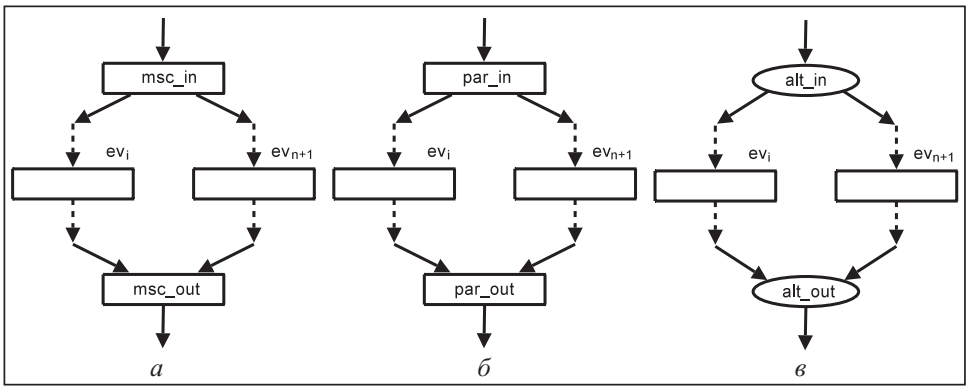

Рис. 6. Фрагменты СП для параллельной (*а*, *б*) и альтернативной (*в*) композиций

сказанным в $L^p(P_{n+1})$ будут присутствовать слова вида $(\dots ev_i \dots ev_{n+1} \dots)$ и не будет слов вида $(\dots ev_{n+1} \dots ev_i \dots)$. В данном случае условие теоремы выполняется.

2. Параллельное выполнение. Предположим, что имеет место ситуация $ev_i || ev_{n+1}$. Рассмотрим возможные случаи параллельного выполнения событий ev_i и ev_{n+1} в документе MSC.

1^o. Как ev_i, так и ev_{n+1} принадлежат одной и той же диаграмме MSC, но не разделяют между собой ни одной сущности. Кроме того, ev_i и ev_{n+1} не упорядочены последовательно каким-либо из описанных в п. 1 методов ($ev_i \notin Prev(ev_{n+1})^*$ и $ev_{n+1} \notin Prev(ev_i)^*$). Соответствующий фрагмент P_{n+1} показан на рис. 6,*а* (в нем отсутствуют пути между переходами ev_i и ev_{n+1}). Согласно принципу независимого срабатывания переходов в сети Петри язык $L^p(P_{n+1})$ будет содержать слова как вида $(\dots ev_i \dots ev_{n+1} \dots)$, так и вида $(\dots ev_{n+1} \dots ev_i \dots)$. В данном случае условие теоремы выполняется.

2^o. События ev_i и ev_{n+1} принадлежат разным диаграммам MSC, которые непосредственно соединены параллельной композицией. Кроме того, ev_i и ev_{n+1} не упорядочены последовательно каким-либо из описанных в п. 1 методов ($ev_i \notin Prev(ev_{n+1})^*$ и $ev_{n+1} \notin Prev(ev_i)^*$). Соответствующий фрагмент показан на рис. 6,*б* (в нем отсутствуют пути между переходами ev_i и ev_{n+1}). Согласно принципу независимого срабатывания переходов в сети Петри язык $L^p(P_{n+1})$ будет содержать слова как вида $(\dots ev_i \dots ev_{n+1} \dots)$, так и вида $(\dots ev_{n+1} \dots ev_i \dots)$. В данном случае условие теоремы выполняется.

3^o. События ev_i и ev_{n+1} принадлежат разным диаграммам MSC, которые косвенно соединены параллельной композицией (например, $ev_i \in$ msc1, $ev_{n+1} \in$ msc2 и документ MSC содержит следующее ссылочное выражение: msc1 par (msc3 seq msc2)). Кроме того, ev_i и ev_{n+1} не упорядочены последовательно каким-либо из описанных в п. 1 методов ($ev_i \notin Prev(ev_{n+1})^*$ и $ev_{n+1} \notin Prev(ev_i)^*$). В этом случае соответствующий фрагмент P_{n+1} такой же, как и для п. 2^o. Соответственно условие теоремы выполняется.

3. Альтернативное выполнение. Предположим, что имеет место ситуация $ev_i \# ev_{n+1}$. В документе MSC альтернативное выполнение событий ev_i и ev_{n+1} возможно только в случае, когда они принадлежат разным диаграммам MSC, непосредственно или косвенно соединенных альтернативной композицией. В обоих случаях соответствующий фрагмент P_{n+1} показан на рис. 6,*в*. Здесь $L^p(P_{n+1})$ содержит слова вида $(\dots ev_i \dots)$ или $(\dots ev_{n+1} \dots)$, но не содержит слов вида $(\dots ev_i \dots ev_{n+1} \dots)$ или $(\dots ev_{n+1} \dots ev_i \dots)$. В данном случае условие теоремы выполняется.

Как видим, для всех возможных композиций ev_i и ev_{n+1} каждой из множества

трасс документа MSC соответствует слово в языке, генерируемом построенной СП. Таким образом, по индукции имеем $r(\mu(S_1)) \subseteq L^p(S_2)$, что и требовалось доказать.

Теорема 2. Добавление конструкций итерации (управляющих конструкций loop или циклических ребер в HMSC) в документ MSC согласно вышеприведенному алгоритму не нарушает включения множества статических трасс, генерируемого исходным документом MSC в язык построенной по нему СП.

Доказательство. Итерации последовательности событий $(sev_1)^+$ в множестве трасс документа соответствуют слова вида $(\dots sev_1 \dots)$, $(\dots sev_1 sev_1 \dots)$, $(\dots sev_1 sev_1 sev_1 \dots)$ и т.д.

Пусть имеются документ MSC M и сеть Петри P такие, что множество статических трасс документа M включается в префиксный язык сети P. Рассмотрим документ M', полученный добавлением в M итерации, и соответствующую ему сеть P'.

Итерация в MSC может быть представлена либо конструкцией loop в ссылочном или встроенном выражении, либо циклической ссылкой в HMSC. Рассмотрим эти случаи.

Конструкция loop. Согласно разд. 1.2 встроенное выражение, использующее loop, можно рассматривать как ссылочное выражение над анонимной диаграммой MSC. Поэтому объединим доказательства для конструкции loop как для ссылочного, так и для встроенного выражений. Аргументом конструкции loop будет некоторое ссылочное выражение ref_1, которому соответствует последовательность событий sev_1. В новом документе M' оно заменяется на ссылочное выражение loop < 1, inf > (ref_1). Согласно алгоритму в P' ему будет соответствовать фрагмент, показанный на рис. 7,*а*. Как видим, в языке $L^p(P')$ будут присутствовать слова вида $(\dots sev_1 \dots)$, $(\dots sev_1 sev_1 \dots)$, $(\dots sev_1 sev_1 sev_1 \dots)$, ... Следовательно, при добавлении в документ MSC конструкции loop условие теоремы выполняется.

Циклическая ссылка в HMSC. Пусть в M имеется некоторая последовательность конструкций (путь) $ev_1 \to \dots \to ev_n$, к которой при построении M' добавляется циклическая ссылка $ev_n \to ev_1$. Данному пути соответствует последовательность событий sev_1. Рассмотрим сеть P', полученную согласно алгоритму. Данному пути в ней будет соответствовать конструкция, показанная на рис. 7,*б*. Как видим, в языке $L^p(P')$ будут присутствовать слова вида $(\dots sev_1 \dots)$, $(\dots sev_1 sev_1 \dots)$, $(\dots sev_1 sev_1 sev_1 \dots)$, ... Следовательно, при добавлении в HMSC циклической ссылки условие теоремы выполняется.

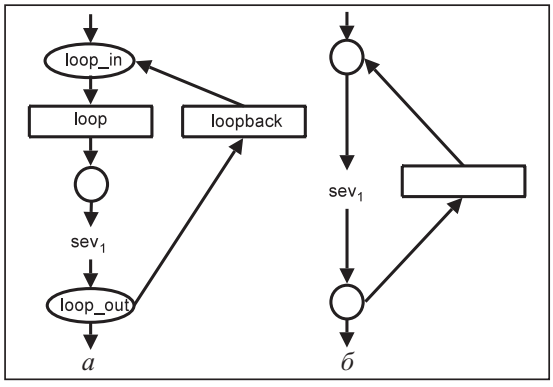

Рис. 7. Фрагменты СП для итерации

Таким образом, добавление итерации сохраняет включение множества трасс документа MSC в язык построенной по нему СП, что и требовалось доказать.

Из теорем 1 и 2 следует, что множество трасс, генерируемых документом MSC, включается в язык построенной по нему СП для любого документа MSC.

3.2. Доказательство включения $r(\mu(S_1)) \supseteq L^p(S_2)$. Теорема 3. Для документа MSC S_1 и построенной по этому документу СП $S_2 = A(S_1)$ выполняется включение $r(\mu(S_1)) \supseteq L^p(S_2)$, т.е. каждому слову языка построенной СП соответствует трасса в исходном документе.

Доказательство. Рассмотрим слова языка $L^p(S_2)$. По построению СП, множество символов, из которых состоят эти слова, включается в множество событий исходного документа (с точностью до переименования). Для любой пары символов ev_1, ev_2 возможны следующие варианты взаимного расположения. Рассмотрим их.

1^o. В языке присутствуют слова вида $(\ldots ev_1 \ldots ev_2 \ldots)$ и отсутствуют слова вида $(\ldots ev_2 \ldots ev_1 \ldots)$. Это имеет место лишь в случае, когда СП содержит фрагмент, показанный на рис. 8,*а*. Согласно алгоритму построения СП такой фрагмент может возникнуть только в случаях, показанных на рис. 5,*в* (в случае $ev_1 \in Prev(ev_2)^*$) и рис. 5,*г* (в случае, когда ev_i и ev_{n+1} принадлежат разным диаграммам MSC, которые соединены последовательно). Оба варианта приводят к появлению в множестве трасс исходного документа MSC последовательности событий $(\ldots ev_1 \ldots ev_2 \ldots)$. Таким образом, каждое слово из $L^p(S_2)$ имеет соответствующую ему трассу в $\mu(S_1)$.

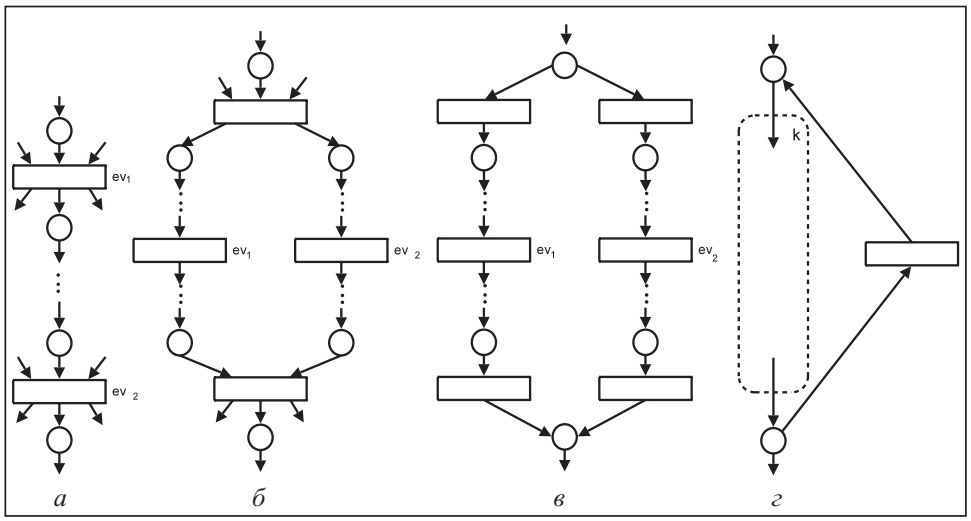

Рис. 8. Возможные фрагменты построенной СП

2^o. В языке присутствуют как слова вида $(\ldots ev_1 \ldots ev_2 \ldots)$, так и слова вида $(\ldots ev_2 \ldots ev_1 \ldots)$. Это имеет место лишь в случае, когда СП содержит фрагмент, показанный на рис. 8,*б*. Согласно алгоритму построения СП такой фрагмент может возникнуть только в случаях, показанных на рис. 6,*а* (независимое исполнение двух сущностей) и рис. 6,*б* (конструкция параллельной композиции). Согласно семантике MSC оба эти варианта приведут к появлению в множестве трасс исходного документа MSC последовательностей событий вида $(\ldots ev_1 \ldots ev_2 \ldots)$ и $(\ldots ev_2 \ldots ev_1 \ldots)$. Таким образом, каждое слово из $L^p(S_2)$ имеет соответствующую ему трассу в $\mu(S_1)$.

3^o. В языке присутствуют слова вида $(\ldots ev_1 \ldots)$ и $(\ldots ev_2 \ldots)$. Это имеет место лишь в случае, когда СП содержит фрагмент, показанный на рис. 8,*в*. Согласно алгоритму построения СП такой фрагмент может возникнуть только в случае, показанном на рис. 6,*в* (конструкция альтернативного выполнения). Согласно семантике MSC в множестве трасс исходного документа MSC будут присутствовать последовательности вида $(\ldots ev_1 \ldots)$ и $(\ldots ev_2 \ldots)$. Таким образом, каждое слово из $L^p(S_2)$ имеет соответствующую ему трассу в $\mu(S_1)$.

4^o. В языке присутствуют слова вида $(\ldots sev_1 \ldots)$, $(\ldots sev_1 sev_1 \ldots)$, $(\ldots sev_1 sev_1 sev_1 \ldots), \ldots$, где sev_1 — некоторая последовательность символов. Заметим, что согласно построению сеть Петри S_2 конечна. Таким образом, бесконечная последовательность в ее языке может быть получена только с помощью цикла, показанного на рис. 8,*г*, где k обозначает фрагмент сети, генерирующий sev_1 (циклы другого вида невозможны по построению СП). Подобный цикл имеет место в случаях, пока-

занных на рис. 7,*а* (конструкция loop) и рис. 7,*б* (циклическая ссылка в HMSC). Согласно семантике MSC в множестве трасс исходного документа MSC будут присутствовать последовательности вида $(\dots sev_1 \dots)$, $(\dots sev_1 sev_1 \dots)$, $(\dots sev_1 sev_1 sev_1 \dots)$, ... Таким образом, каждое слово из $L^p(S_2)$ имеет соответствующую ему трассу в $\mu(S_1)$.

В силу произвольности выбора ev_1 и ev_2 каждое слово из $L^p(S_2)$ имеет соответствующую ему трассу в $\mu(S_1)$, что и требовалось доказать. Таким образом, $r(\mu(S_1)) \subseteq L^p(S_2)$ и $r(\mu(S_1)) \supseteq L^p(S_2)$, т.е. $r(\mu(S_1)) = L^p(S_2)$ и построенная по описанному алгоритму СП событийно эквивалентна исходному документу MSC.

4. ПРИМЕР КОНВЕРТАЦИИ MSC-ДИАГРАММЫ В СП И ИССЛЕДОВАНИЯ СТРУКТУРНЫХ СВОЙСТВ

Проиллюстрируем работу алгоритма на примере одновременной конкурентной работы двух агентов с общим ресурсом. Поскольку полные модели реальных систем и соответствующие им сети Петри занимают большой объем, пример будет максимально упрощенным. (Более подробные примеры размещены по адресу http://avch.org.ua/samples/.)

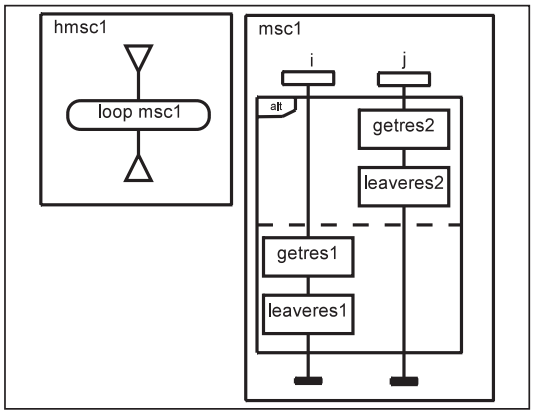

Рис. 9. Документ MSC, описывающий конкурентное взаимодействие с общим ресурсом

4.1. Исходная MSC-диаграмма. Исходный документ MSC показан на рис. 9. На диаграмме msc1 присутствуют два агента (представленных сущностями *i* и *j*), каждый из которых работает с общим ресурсом с помощью действий getres1, leaveres1 и getres2, leaveres2 соответственно. Предполагается, что в каждый момент времени с ресурсом может работать только один агент, поэтому соответствующие действия разнесены по разным альтернативам встроенного выражения типа alt. На диаграмме hmsc1 ссылка на msc1 сопровождается оператором итерации loop, поэтому диаграмма msc1 будет выполняться неоднократно.

4.2. Сеть Петри, полученная в результате конвертации. Построим сеть Петри для приведенной диаграммы с помощью рассмотренного алгоритма. Документ MSC содержит четыре события: getres1, leaveres1, getres2 и leaveres2. Множества *Prev*() для этих событий имеют вид

$$Prev(getres1) = \{alt\_in\},$$

$$Prev(leaveres1) = \{getres1\},$$

$$Prev(getres2) = \{alt\_in\},$$

$$Prev(leaveres2) = \{getres2\}.$$

С помощью описанного алгоритма получаем сеть, показанную на рис. 10,*а*; применив к ней правила упрощения сети Петри (см. рис. 1), получим сеть, показанную на рис. 10,*б*. Разметка сети показана рядом со знаками переходов.

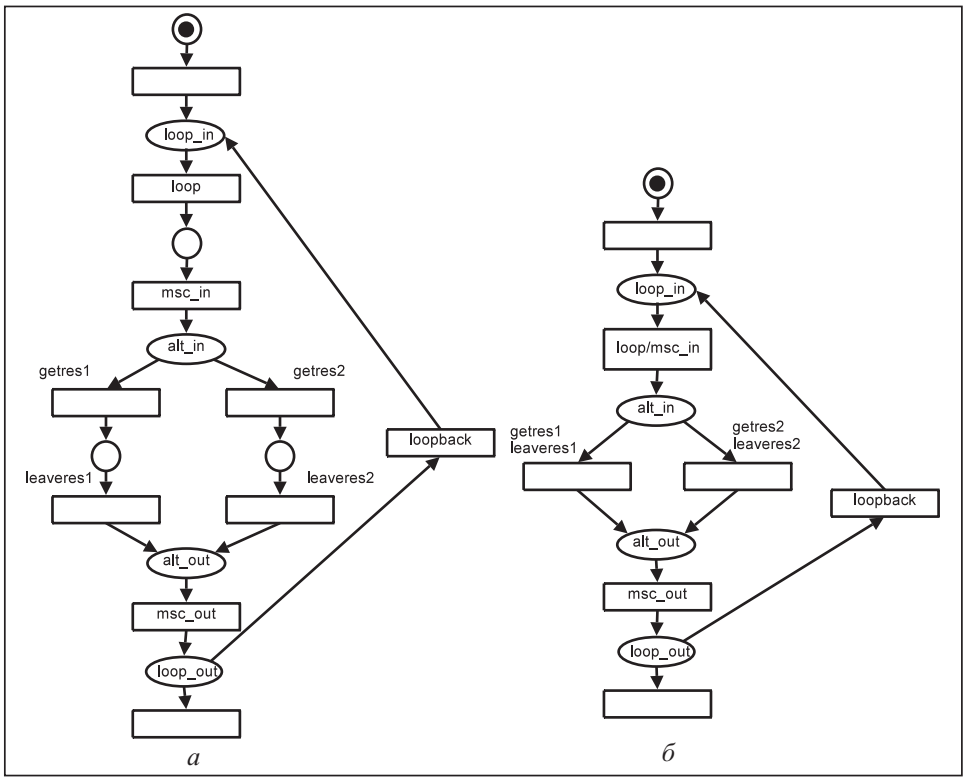

Рис. 10. Сеть Петри, полученная из исходного документа (*а*), и ее упрощенный вариант (*б*)

4.3. Дерево покрытия. Построим дерево покрытия для полученной (упрощенной) сети (рис. 11). Анализируя дерево покрытия, можно сделать следующие выводы:

1) сеть ограничена;
2) сеть безопасна;
3) в сети нет недостижимых мест;
4) все переходы сети потенциально живы.

Применительно к анализу исходного документа MSC это означает, что в документе нет «мертвых» участков (каждое событие сети может быть выполнено в рамках определенного сценария) и нет ситуации множественного захвата ресурса (поскольку сеть безопасна).

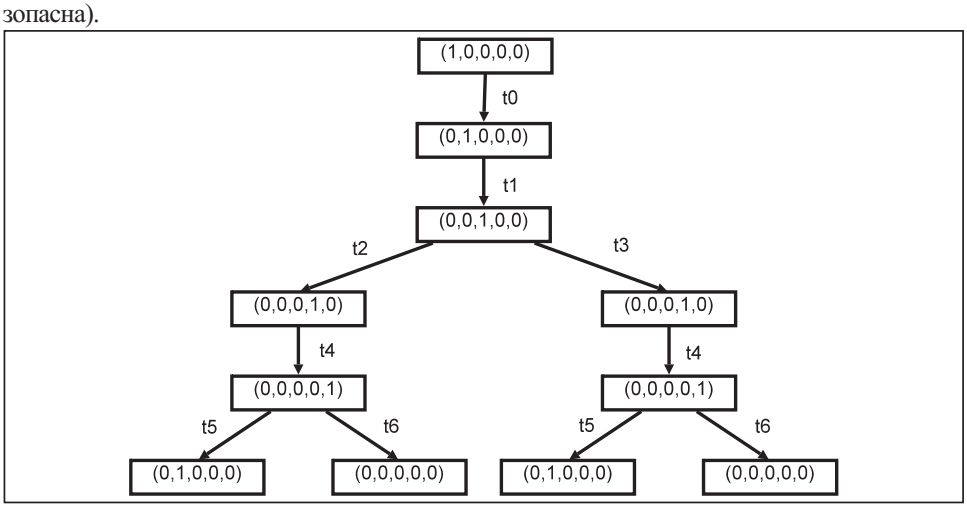

Рис. 11. Дерево покрытия для упрощенной сети

4.4. Уравнение состояния и его решение TSS-алгоритмом. Построим матрицу инцидентности для упрощенной сети

$$A = \begin{Bmatrix} -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & -1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & -1 & 0 \\ 0 & 0 & 0 & 0 & 1 & -1 & -1 \end{Bmatrix}.$$

Решив уравнение состояния системы $Ax = 0$ с помощью метода TSS, получим усеченное множество решений

$$\begin{pmatrix} 0 & 1 & 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 1 & 1 & 0 \end{pmatrix}.$$

Отсюда следует, что все переходы, кроме первого и последнего (соответствующих старту и остановке системы), являются живыми. Применительно к анализируемой системе это обозначает, что все четыре ее события могут исполняться неограниченное число раз.

ЗАКЛЮЧЕНИЕ

В настоящей статье описан алгоритм перевода документа MSC в событийно эквивалентную ему сеть Петри. Доказана событийная эквивалентность полученной СП исходному документу. Показан технологический процесс использования этого алгоритма для анализа свойств систем, выраженных на языке MSC, а также приведен пример, иллюстрирующий работу алгоритма.

СПИСОК ЛИТЕРАТУРЫ

1. ITU-TS Recommendation Z.120: Message Sequence Chart (MSC) / Edited by ITU-TS. — ITU-TS, Geneva, 2000. — 128 p.
2. Nakamura M., Kakuda Y., Kikuno T. Petri-net based detection method for non-deterministic feature interaction and its experimental evaluation // Feature Interaction in Telecommunication Networks. — 1997. — N 4. — P. 138–152.
3. Kluge O., Padberg J., Ehrig H. Model train control systems: From message sequence charts to petri nets // Technische Universitat Berlin, 2001 — 18 p.
4. Kryvyy S., Matvyeyeva L., Lopatina M. Automatic modeling and analysis of msc-specified systems // Fundamenta Informaticae. — 2005. — **67**, N 1–3. — P. 107–120.
5. Kryvyi S., Matvyeyeva L. Algorithm of translation of msc-specified system into petri net // Ibid. — 2007. — N 79. — P. 1–15.
6. Kryvyi S., Matvyeyeva L., Chugaenko A. Extension of algorithm of translation of msc-specified system into petri net // Proceedings of the CS&P'2007 Workshop, 2007. — P. 376–387.
7. Чугаенко А. О реализации алгоритма перевода набора msc-диаграмм в сеть Петри // Управляющие системы и машины. — 2007. — № 6. — С. 17–23.
8. Kryvyy S., Chugayenko O. Extended algorithm for translation of msc diagrams into petri nets // Fundamenta Informaticae. — 2008. — **2**, N 1. — P. 68–75.
9. Котов В.Е. Сети Петри. — М.: Наука, 1984. — 160 с.
10. Murata T. Petri nets: Properties, analysis and applications // Proceedings of the IEEE. — 1989. — **77**. — P. 541–580.
11. Крывый С.Л. Алгоритмы решения систем линейных диофантовых уравнений в целочисленных областях // Кибернетика и системный анализ. — 2006. — № 2. — С. 3–17.
12. Чугаенко А. О реализации *tss*-алгоритма // Управляющие системы и машины. — 2007. — № 4. — С. 14–18.
13. ITU-TS Recommendation Z.120. Annex B: Algebraic Semantics of Message Sequence Charts / Edited by ITU-TS. — ITU-TS, Geneva, 1998. — 73 p.

Поступила 19.08.2008