Российская академия наук
Сибирское отделение
Институт систем информатики
им. А. П. Ершова

Е. В. Окунишникова

ПРЕДСТАВЛЕНИЕ ВРЕМЕННЫХ КОНСТРУКЦИЙ ESTELLE В РАЗЛИЧНЫХ МОДЕЛЯХ ВРЕМЕННЫХ СЕТЕЙ ПЕТРИ

Препринт
70

Новосибирск 1999

Настоящая работа посвящена моделированию Estelle-таймеров в терминах временного механизма, предложенного Йенсеном для раскрашенных сетей. Рассматривается два способа представления Estelle-переходов с задержками: с использованием временной пометки на входных дугах переходов сети и без нее. Кроме того, предложен метод организации фазы управления, позволяющий не использовать в сети приоритеты при моделировании Estelle-переходов с задержками.

Siberian Division of the Russian Academy of Sciences
A. P. Ershov Institute of Informatics Systems

E. V. Okunishnikova

PRESENTATION OF THE ESTELLE TIME CONSTRUCTIONS USING DIFFERENT MODELS OF TIMED PETRI NETS

Preprint
70

Novosibirsk 1999

This work is devoted to modeling Estelle timers in terms of time mechanism presented by Jensen for coloured Petri nets. In the paper two methods are considered to represent Estelle transitions with delay-clauses. One of them uses the time inscriptions on the input arcs of the transitions, and the second does not. Moreover, the method of organization of the management phase is presented which allows us not to use the priorities when Estelle transitions with delay-clauses are modeled.

ВВЕДЕНИЕ

В настоящее время не вызывает сомнений важность использования формальных методов для спецификации и верификации распределенных систем таких, как, например, коммуникационные протоколы. Для описания последних были созданы языки выполнимых спецификаций Estelle, Lotos и SDL, принятые в качестве стандартов [1, 2, 3, 4]. Однако методы анализа для этих языков разработаны не столь хорошо, как для формальных моделей конечных автоматов или сетей Петри и их обобщений. Поэтому зачастую для верификации выполнимые спецификации транслируются в модели, для которых разработаны эффективные методы анализа и/или существуют средства автоматической верификации.

Для Estelle-спецификаций предложен метод автоматического построения конечно-автоматных моделей посредством исчерпывающей симуляции [6]. Известны также методы трансляции Estelle-спецификаций в сети Петри, причем используются как ординарные [7], так и сети Петри высокого уровня (так называемые нумерические) [8]. Методы трансляции SDL-спецификаций в обобщенные сети Петри, такие как SDL- и Pr/T-сети (от английского — predicate/transition), описаны в работах [9, 10, 11].

В ИСИ СО РАН реализована экспериментальная система EPV, которая позволяет автоматически транслировать Estelle-спецификации в модифицированные иерархические раскрашенные сети, обогащенные временным механизмом и приоритетами, и проводить их симуляцию [12, 13, 14]. Ведется также работа по отображению в раскрашенные сети SDL-спецификаций [15].

В работе [12] в качестве базовой сетевой модели использовались раскрашенные сети, предложенные Йенсеном [17]. Поскольку на момент разработки алгоритма трансляции Estelle-спецификаций для раскрашенных сетей не было определено собственного временного расширения, в сети Йенсена был перенесен временной механизм, предложенный Мерлином [20]. Выбор механизма был сделан на основании схожести временных ограничений, накладываемых на поведение сети и спецификации в языке Estelle.

Однако позднее для раскрашенных сетей был разработан временной механизм, основанный на понятиях глобальных часов и временных штампов фишек [18]. Этот механизм реализован в системе Design/CPN, которая позволяет строить, симулировать и анализировать раскрашен-

ные сети со временем или без него.

В данной работе описывается способ моделирования задержек Estelle-переходов в терминах временного механизма Йенсена. Предложенный способ позволяет переносить сети, полученные при трансляции Estelle-спецификаций, в среду системы Design/CPN с целью использования реализованных там методов анализа. Данная работа состоит из четырех разделов. Первый раздел посвящен базовым понятиям языка Estelle. Второй раздел содержит описание раскрашенных сетей Петри, а третий — их расширения двумя разными временными механизмами. В четвертом разделе представлены два способа моделирования задержек Estelle-переходов: в терминах временных механизмов Мерлина и Йенсена. Причем для последнего рассмотрена возможность не использовать временные пометки на входных дугах переходов.

1. ОБЗОР ЯЗЫКА ESTELLE

Язык формальных описаний Estelle [1, 2, 4] основан на модели расширенного конечного автомата. Estelle-спецификация описывает иерархически структурированную систему недетерминированных компонент, взаимодействующих с помощью сообщений (*взаимодействий*) по двунаправленным каналам между портами (*точками взаимодействия*). Каждая компонента есть экземпляр модуля. Иерархия компонент и структура связей может изменяться в процессе функционирования системы.

1.1. Понятие модуля

Модуль определяется заголовком и телом, связанным с заголовком. Тело модуля может включать в себя описания других модулей, называемых *наследниками* данного модуля, которые, в свою очередь, могут содержать описания модулей. Охватывающий модуль называется *родителем* модулей, непосредственно описанных в его теле. Модуль, охватывающий все прочие модули системы, называется *спецификацией* системы. В процессе выполнения Estelle-спецификации одновременно может существовать несколько *экземпляров* модуля, которые создаются как статически — при инициализации всей системы, так и динамически.

С внешней точки зрения экземпляр модуля представляет собой “черный ящик”. Взаимодействие с ним происходит через конечное число точек взаимодействия и экспортируемые переменные, доступ к которым

имеет только экземпляр родительского модуля. Все экземпляры одного модуля обладают одинаковыми внешними признаками, характеристики которых определяются заголовком модуля, где описаны тип и класс данного модуля, точки взаимодействия, экспортируемые переменные и формальные параметры.

Внутреннее поведение модуля определяется его телом, содержащим следующие разделы: деклараций, инициализации и описания переходов. Для модуля может быть описано несколько разных тел, т. е. поведение различных экземпляров одного и того же модуля может различаться. Выбор определенного тела для экземпляра модуля происходит при его создании.

В *разделе деклараций* содержатся описания констант, типов, переменных, процедур и функций, а также описания специфических объектов Estelle таких, как: каналы, модули и модульные переменные, управляющие (локальные) состояния и точки взаимодействия модуля. Описание канала определяет две роли — по одной на каждое окончание канала, которые должны играть модули при взаимодействии по этому каналу. С каждой ролью связан набор примитивов взаимодействий, которые может передавать модуль, играющий данную роль. Взаимодействие может сопровождаться списком формальных параметров. Модульные переменные определяют экземпляры модулей и позволяют различать их.

С каждой точкой взаимодействия ассоциирована неограниченная FIFO-очередь, в которую поступают все сообщения, полученные экземпляром модуля через эту точку взаимодействия. Несколько точек взаимодействия могут делить между собой одну и ту же очередь. Описание точки взаимодействия определяет, каналом какого типа она может быть связана с другой точкой, роль, которую играет модуль при взаимодействиях через данную точку, и дисциплину — индивидуальную или общую — очереди, ассоциированной с данной точкой. Описание канала дается в родительском модуле. Там же определяется, с какой точкой взаимодействия какого экземпляра модуля связана данная точка.

Раздел инициализации определяет значения переменных модуля, с которыми каждый созданный экземпляр данного модуля начинает свое выполнение. Инициализация модульных переменных приводит к созданию экземпляров модулей-наследников. В разделе инициализации определяются также связи между точками взаимодействия модулей-наследников.

И наконец, *раздел описания переходов* описывает действия модуля в

терминах системы состояний. Контрольные состояния системы состоят из управляющего состояния экземпляра модуля, очередей всех точек взаимодействия, значений переменных, текущего множества наследников модуля (если есть) и текущей структуры связей между ними. Начальное состояние определяется разделом инициализации. Изменение контрольных состояний происходит при выполнении действий, описанных в переходах.

В модуле может быть не описано ни одного перехода. Такой модуль называется *неактивным*. После инициализации неактивный модуль не выполняет ни одного действия, тогда как его наследники могут их выполнять. Модуль с непустым набором переходов называется *активным*. Каждый переход характеризуется *условием возможности* и *действием*. Условие возможности перехода включает текущее управляющее состояние (приставка *from*), входное взаимодействие (приставка *when*), предикат возможности (приставка *provided*), приоритет перехода (приставка *priority*) и условие задержки (приставка *delay*). Переход может осуществиться, если выполнены все части его условия возможности.

Действие перехода содержит оператор *to* и блок перехода, состоящий из операторов языка Паскаль с некоторыми ограничениями (например, не используются стандартные операторы *write* и *read*) и специальных операторов Estelle. Выполнение перехода рассматривается как атомарное действие. Начавшееся выполнение перехода не может быть прервано.

1.2. Принцип структуризации и шаг выполнения

Согласно концепциям Estelle, описываемая система определяется как множество взаимодействующих модулей. Текстуальная вложенность описаний модулей определяет иерархию модулей. Поведение каждого экземпляра модуля в спецификации зависит от его положения в иерархии модулей и класса, приписанного модулю.

Каждый модуль может иметь класс *системный процесс, системная активность, процесс* или *активность*. Системные процессы и активности называют также *системными модулями*. Множество экземпляров модулей, описания которых заключены в описании системного модуля, называется *подсистемой*. Классификация модулей подчиняется следующим требованиям:

– каждый активный модуль должен иметь определенный класс;
– системный модуль не может быть вложен в активный;
– каждый модуль класса процесс или активность должен быть вло-

жен в системный модуль;

– (системный) процесс может включать в себя описания процессов и активностей, а (системная) активность — только описания активностей.

Таким образом, существует единственный уровень системных модулей, и модули, охватывающие системные, — неактивны. Допускается существование нескольких экземпляров одного и того же системного модуля. Количество системных модулей и структура связей между ними описываются в охватывающем неактивном модуле и, следовательно, не меняются в течение всего времени выполнения спецификации.

Поведение системных модулей полностью асинхронно. Внутри подсистем оно синхронизируется экземплярами родительских модулей. Каждый экземпляр модуля предлагает один из своих готовых к выполнению переходов модулю-родителю. Переходы модуля-родителя имеют приоритет над переходами наследников: если модуль-родитель предлагает переход к выполнению, то он будет выполняться на следующем такте. Выполнение перехода родителя исключает выполнение переходов всех (не только непосредственных) наследников. Классы процесс и активность определяют два возможных способа выполнения: параллельное и недетерминированное. Если родитель не предлагает переходов к выполнению и имеет класс процесс, то на следующем такте будут параллельно выполнены все предложенные наследниками переходы. Если родитель — активность, то на следующем такте выполняется только один из предложенных наследниками переходов. Выбор перехода для выполнения осуществляется недетерминированно. Такт вычисления в подсистеме завершается, когда выполнены все переходы, предложенные к выполнению. Такты чередуются с фазами управления, в ходе которых после завершения очередного шага вычислений осуществляется проверка условий возможности всех переходов.

Ниже приведена Estelle-спецификация `example`, в которой описаны канал и два типа модулей: `sender` и `receiver`. Каждый из модулей имеет две точки взаимодействия, с которыми ассоциированы общая очередь. Модуль `sender` передает сообщения, нумеруя их натуральными числами. Отправление следующего сообщения происходит после того, как получено подтверждение удачной передачи предыдущего сообщения. Модуль `receiver` принимает сообщения от модуля `sender` и сразу передает подтверждение, содержащее номер принятого сообщения. Раздел инициализации спецификации `example` определяет систему, которая содержит по одному экземпляру `sender` и `receiver`. Точка взаимодей-

ствия s1 модуля sender соединена каналом с точкой взаимодействия r1 модуля receiver, а точка s2 — с точкой r2.

```
specification example;
channel CONN(ROLE1,ROLE2);
by ROLE1, ROLE2: mess(i:integer);

module sender systemprocess;
 ip s1:CONN(ROLE2) common queue;
    s2:CONN(ROLE1) common queue;
end;

body send for sender;
  state  wait, deliver;
  var j : integer;
    initialize to deliver
     begin  j:=0  end;

trans (* transmission of a new message *)
      from deliver to wait
      begin  output s2.mess(j) end;
trans (* reception of acknowledgement *)
      when s1.mess(i)
      provided (i=j)
      from wait to deliver
      begin j:=j+1 end;
end;

module receiver systemprocess;
 ip r1:CONN(ROLE1) common queue;
    r2:CONN(ROLE2) common queue;
end;

body reception for receiver;
var  j:integer;
initialize  begin j:=0 end;

trans (* reception of message and transmission *)
      (* of aknowledgement *)
```

```
     when r2.mess(i)
     provided (i=j)
     begin output r1.mess(j); j:=j+1 end;
end;

modvar  ms: sender;
        mr: receiver;

initialize  begin
   init  ms with send;
   init mr with reception;
   connect ms.s1 to mr.r1;
   connect ms.s2 to mr.r2;
end;
end.
```

1.3. Концепция времени в Estelle

Вычислительная модель Estelle описана в терминах, независимых от времени. Одним из принципиальных предположений является то, что о времени выполнения перехода ничего неизвестно, так как скорость выполнения полностью зависит от реализации. Утверждается только, что вычисления происходят не моментально, а на их выполнение затрачивается некоторое время. При этом ход времени одинаков для всех экземпляров модулей. Если в системе существует несколько переходов, выполнение которых было отложено, то по завершении очередного такта вычислений задержки этих переходов уменьшатся на одну и ту же величину.

Для соотнесения процесса вычислений с течением времени используется приставка *delay*, которая может входить в условие возможности некоторых переходов. Данная приставка служит для индикации того, что выполнение перехода (если он возможен) должно быть отложено. Задержка определяется двумя временными значениями: минимальным временем, на которое должно быть отложено выполнение перехода, и максимально возможным временем задержки. Иногда считают, что с каждым переходом, имеющим приставку *delay*, связан таймер, включение и выключение которого происходит во время очередной фазы управления после проверки условия возможности перехода.

Приставка *delay* имеет три синтаксические формы: *delay(d1)*,

delay(d1,d2) и *delay(d1,*)*. Форма *delay(d1)* семантически эквивалентна *delay(d1,d1)*, а символ $*$ служит для обозначения бесконечности. Приставка *delay(d1,d2)* указывает на то, что выполнение перехода должно быть задержано не менее чем на *d1* единиц времени и не может быть задержано более чем на *d2* единиц времени с момента, как его выполнение стало возможно. Если за время, на которое отложено выполнение некоторого перехода, произошло выполнение другого перехода, которое, возможно, изменило значения переменных и локальное состояние модуля, но не нарушило условие возможности отложенного перехода, то задержка продолжает отсчитываться.

2. СЕТИ ПЕТРИ

2.1. Ординарные сети Петри

Напомним, что обыкновенную или *ординарную* сеть Петри можно определить как размеченный ориентированный граф с вершинами двух типов: *местами* и *переходами*, соединенными дугами таким образом, что каждая дуга соединяет вершины различных типов [5]. Переходы изображаются, как правило, прямоугольниками или барьерами, места — окружностями. Места помечаются целыми неотрицательными числами (*разметка* места). В графическом представлении сети разметка изображаются соответствующим числом точек (*фишек*) в месте. Места, из которых в переход ведут дуги, называются *входными* местами для данного перехода. Места, в которые ведут дуги от данного перехода, называются его *выходными* местами. По аналогии с местами дуги, ведущие от места к переходу, будут называться *входными*, а ведущие от перехода к месту — *выходными*.

Сеть Петри функционирует, переходя от разметки к разметке. Функционирование начинается при заданной начальной разметке. Смена разметок происходит в результате срабатывания одного из переходов. Переход может сработать при некоторой разметке, если все входные места перехода содержат хотя бы по одной фишке. Срабатывание перехода изымает по фишке из каждого входного места перехода и помещает по фишке в каждое его выходное место.

Таким образом, сеть Петри моделирует некоторую систему и динамику ее функционирования. При этом места и находящиеся в них фишки представляют состояние моделируемой системы, а переходы — изменение ее состояний.

2.2. Раскрашенные сети Петри

Неиерархические раскрашенные сети Петри являются расширением ординарных сетей Петри, предложенным Йенсеном [16, 17]. Поскольку в данной работе мы не будем рассматривать иерархические конструкции для раскрашенных сетей, в дальнейшем будем употреблять название раскрашенная сеть Петри, не уточняя, что речь идет о неиерархической сети. Раскрашенная сеть состоит из трех частей: структуры сети (которая, по существу, не отличается от структуры ординарной сети), деклараций и пометки сети.

Декларации состоят из описания множеств цветов (типов) и объявления переменных, каждая из которых принимает значения из некоторого множества цветов. Декларации также могут содержать определение операций и функций. Кроме того, определение множества цветов зачастую неявно вводит набор операций и функций, которые могут быть применены к элементам этого множества.

Пометка сети приписывается месту, переходу либо дуге. Каждое место имеет три разных типа пометок: имя места, множество цветов и инициализирующее выражение. Имя не имеет формального значения и служит для идентификации. Множество цветов определяет тип фишек, которые могут находиться в месте, т. е. любая фишка, находящаяся в месте, должна иметь цвет, который является элементом данного множества цветов. Инициализирующее выражение определяет начальную разметку места. Переходы имеют два типа пометок: имена и спусковые функции, а дуги — один тип: выражения. Спусковая функция перехода является логическим выражением, которое должно быть выполнено до того, как переход сможет сработать. Выражения на дугах могут содержать переменные, константы, функции и операции, определенные в декларациях.

Пример раскрашенной сети приведен на рис. 1. Сеть состоит из перехода *trans* и четырех мест: *State*, *counter*, i и j. Все места за исключением i являются как входными, так и выходными местами перехода *trans*. Место i является только входным местом. Декларации сети ограничены пунктирной линией и содержат определения трех множеств цветов и декларации переменных, входящих в выражения на дугах. Рядом с каждым местом приведены имя места, связанное с ним множество цветов и начальная разметка места. Над переходом указаны его имя и спусковая функция.

Далее в работе в большинстве случаев декларации опускаются. Мно-

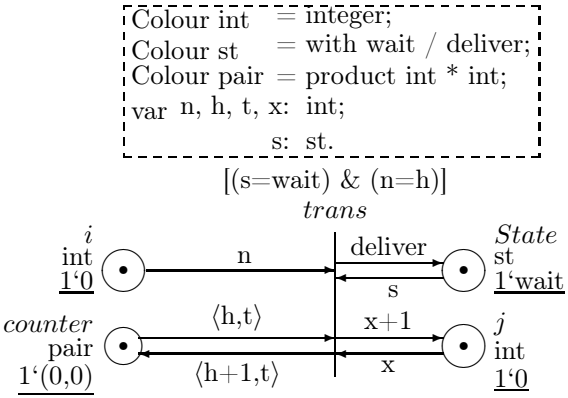

Рис. 1. Пример раскрашенной сети

жества цветов, приписанные местам, описываются в тексте; для переменных, входящих в выражение на дуге, предполагается, что они принимают значения из множества цветов, приписанного месту, с которым связана дуга.

Функционирование раскрашенной сети отличается от функционирования обыкновенной сети тем, что возможность срабатывания перехода зависит не только от наличия фишек во входных местах перехода, но и от их значений. Чтобы говорить о срабатывании перехода, необходимо определить значения переменных перехода, т. е. всех переменных в спусковой функции перехода и на связанных с ним дугах. Выбор значений переменных, при которых выполнена спусковая функция перехода, называется *связыванием*. Значение выражения на дуге при выбранных значениях переменных определяет фишку (мультимножество фишек в общем случае), которая может быть “перемещена” по этой дуге. Другими словами, значение выражения на входной дуге определяет, сколько и каких фишек должно содержаться в соответствующем входном месте перехода, чтобы переход мог сработать при выбранных значениях переменных. Выражения на выходных дугах определяют, сколько и каких фишек будет помещено в выходные места перехода, когда он сработает.

Переход раскрашенной сети *возможен*, если можно выбрать такие значения переменных перехода, что в каждом входном месте перехода имеется фишка, определенная значением выражения на соответствую-

щей дуге. Возможный переход может сработать. Срабатывание перехода изымает фишки из его входных мест и добавляет в выходные места. Количество и цвет изымаемых/добавляемых фишек определяются выражениями на соответствующих дугах.

В сети на рис. 1 переход *trans* может сработать при начальной разметке. Все входные места перехода содержат по фишке. Значения $s = wait$, $n = 0$, $h = 0$, $t = 0$, $x = 0$ определяют связывание. После срабатывания перехода *trans* место i станет пустым, место *State* будет содержать фишку со значением *deliver*, место *counter* — фишку со значением $\langle 1, 0 \rangle$, а место j — фишку со значением 1.

3. ВРЕМЕННЫЕ РАСШИРЕНИЯ СЕТЕЙ ПЕТРИ

Для моделирования систем, поведение которых явно зависит от времени, предложено множество временных расширений сетей Петри, сильно отличающихся между собой по существу и сложности. Общим для них является наличие внешнего времени, которое наряду с разметкой сети определяет возможность срабатывания переходов.

Ниже описываются два способа введения временного механизма в модель сетей Петри. Первый предложен Мерлином для обыкновенных сетей Петри, но, поскольку нашей конечной целью является моделирование таймеров в языке Estelle, мы переносим его в модель раскрашенных сетей. Второй механизм предложен Йенсеном для раскрашенных сетей. Наш интерес к этому механизму обусловлен тем, что он реализован в системе Design/CPN, которая позволяет создавать, симулировать и анализировать раскрашенные сети (со временем или без него).

Поскольку в обоих случаях результатом являются "временные раскрашенные сети", для уточнения, о какой модели идет речь, будем использовать имя автора, предложившего временную концепцию.

3.1. Временные сети Мерлина

Основой для временных сетей Мерлина [20] служат обыкновенные сети Петри. Временной механизм связан с переходами: каждому переходу сопоставлена пара неотрицательных чисел d_{min} и d_{max}. Переход, каждое из входных мест которого содержит хотя бы одну фишку, считается возможным. В отличие от ординарных сетей не всякий возможный переход имеет право сработать.

Если переход стал возможен в момент времени τ, то срабатывание

этого перехода произойдет в некоторый момент времени из интервала $[\tau + d_{min},\ \tau + d_{max}]$, если условие его возможности не будет нарушено до наступления времени $\tau + d_{max}$ в результате срабатывания другого перехода. Если произошло любое, даже моментальное, нарушение условия возможности перехода, то переход реинициируется, и время его задержки будет заново отсчитываться, начиная с момента восстановления условия его возможности. Возможный переход, имеющий право сработать, т. е. переход, который ожидает не менее d_{min} единиц времени с момента, когда он стал возможен, называется *реализуемым*, а интервал $[d_{min}, d_{max}]$ — *интервалом срабатывания* перехода. Если интервал срабатывания перехода не указан, то подразумевается интервал $[0, \infty]$, т.е. временные ограничения для выполнения перехода отсутствуют.

Если переход оставался возможным в течение всего интервала срабатывания и не сработал, то срабатывание форсируется и произойдет в момент времени $\tau + d_{max}$. Срабатывание перехода происходит мгновенно.

Таким образом, значение d_{min} определяет минимальное время, в течение которого переход должен оставаться возможным, прежде чем он сможет сработать, а значение d_{max} определяет максимальное время, в течение которого переход может оставаться возможным и не сработать.

Описанный временной механизм легко переносится в модель раскрашенных сетей. У переходов сети появляется новый тип пометки — интервал срабатывания. Те же правила, что и раньше, определяют, может ли переход сработать: если во входных местах перехода содержится набор фишек, необходимый для некоторого связывания, то переход возможен. Само срабатывание происходит с задержкой, значение которой выбирается из интервала срабатывания.

Обычно рассматриваются временные сети, в которых не допускается кратной возможности переходов, т. е. в любой момент времени ни один переход в сети не имеет такого числа фишек в своих входных местах, которого было бы достаточно для двух или более одновременных срабатываний. Такие сети носят название *T-безопасных* временных сетей Петри. Мы также можем ограничить рассмотрение T-безопасными сетями, поскольку в результате трансляции Estelle-спецификаций получаются сети, переходы которых не имеют возможности кратного срабатывания [12].

3.2. Временные сети Йенсена

В работе [16], где подробно описывались иерархические конструкции, собственный временной механизм для раскрашенных сетей предложен не был. Однако автор утверждал, что любой из предложенных на момент написания статьи временных механизмов корректно вписывается в модель раскрашенных сетей.

В работе [18] уже предложен собственный временной механизм. Для представления времени вводится понятие *глобальных часов*. Значение часов представляет текущее время в модели или *модельное время*. Начальное модельное время, т. е. значение часов, при котором сеть начинает функционировать, задается при описании сети. Множества цветов (все или часть) получают признак *timed*. Фишка, принимающая значения из множества, обладающего признаком *timed*, в дополнение к основному цвету несет временное значение. Это значение, также называемое *временным штампом*, определяет момент времени, раньше которого фишка не может использоваться при срабатывании какого-либо перехода. Фишки, принимающие значения из множеств, не имеющих признака *timed*, временного штампа не несут. Последнее означает, что фишка готова к использованию с момента своего создания.

Временной штамп новой фишки вычисляется как текущее модельное время плюс задержка, величина которой определяется выражением $@+D$. Это выражение называется также *временной пометкой* и может быть связано с переходом и/или с выходной дугой. Если временная пометка связана только с переходом, то во все выходные места перехода помещаются фишки с одинаковыми временными штампами. Временная пометка на дуге определяет значение временного штампа только для фишки, помещаемой в место, в которое ведет данная дуга. Если присутствуют оба типа пометок, то значения задержек на переходе и на дуге суммируются.

С помощью временной пометки могут задаваться различные типы задержек. Значение задержки может быть константным, зависеть от связывания, при котором произошло срабатывание, выбираться случайным образом из заданного интервала и т. п. В системе Design/CPN реализовано несколько стандартных статистических функций, которые можно использовать во временной пометке сети [21].

Как и во временных сетях Мерлина, временной механизм влияет на правило выбора переходов, которые имеют право сработать. Будем называть раскрашенную сеть, которая получается из временной удале-

нием всех временных конструкций, *базовой* для временной сети. Во временных раскрашенных сетях переход называется возможным по цвету при выбранном связывании, если он возможен при этом связывании в базовой сети. Однако чтобы возможный по цвету переход мог сработать, временные штампы фишек, необходимых для выбранного связывания, должны быть не больше, чем текущее модельное время. Переходы, для которых выполнено последнее условие, будем называть *готовыми*, как это делает Йенсен, или реализуемыми по аналогии с временными сетями Мерлина. Срабатывание перехода происходит мгновенно.

Текущее модельное время не изменяется до тех пор, пока остаются готовые переходы. Когда в сети остаются только возможные по цвету переходы, происходит изменение значения глобальных часов. При этом новым значением часов будет ближайший момент времени, в который какой-либо из возможных переходов становится готовым.

При описанном подходе фишка становится доступна одновременно для всех переходов, для которых место, содержащее данную фишку, является входным, что создает ряд неудобств при моделировании остановки таймеров до истечения отсчитываемого интервала времени. В работе [18] для моделирования остановки таймеров предложено использовать временную пометку на входных дугах переходов. Если переход возможен по цвету в текущем модельном времени τ, временная пометка на входных дугах определяет новое значение τ_i^* для каждого входного места p_i. Переход может сработать, если временные штампы фишек, изымаемых при срабатывании из места p_i, не больше τ_i^*, хотя могут превышать значение τ. Таким образом, один из переходов может использовать фишку раньше, чем она станет доступна остальным переходам. Однако данный механизм описан только теоретически и не реализован в системе Design/CPN.

В работе [21] предлагается другой подход. Чтобы обеспечить доступ к фишкам, временной штамп которых больше, чем текущее модельное время, на входных дугах перехода используется временная пометка $@\,ignore$. Такая пометка означает, что при определении готовности перехода временные штампы фишек в соответствующем входном месте игнорируются.

4. МОДЕЛИРОВАНИЕ ESTELLE-ТАЙМЕРОВ

Напомним основные принципы трансляции Estelle-спецификаций в раскрашенные сети, расширенные временным механизмом Мерлина и

приоритетами [12]. Сеть, моделирующая спецификацию, является иерархической и строится с помощью поэтапного уточнения. На каждом этапе строится новый уровень иерархии сети. На первом этапе отображается основная структура системы. Второй этап — трансляция тел модулей. На последующих этапах происходит трансляция действий Estelle-переходов (E-переходов).

В результирующей сети каждый E-переход представляется отдельной сетью, которая состоит из последовательно связанных фрагментов. Каждый из фрагментов моделирует одну из приставок *provided* или *delay* условия E-перехода или один из операторов блока E-перехода. Фрагмент может быть отдельным переходом сети или более сложной подсетью. Цепочку фрагментов, представляющих блок E-перехода, ограничивают два служебных перехода *start* и *end*. Рассмотрим пример. На рис. 2 схематично изображена сеть для следующего E-перехода:

```
trans
     from S1
     when U_to_S.DT
     provided P
       to S2
       begin
            . . .
       end;
```

Заметим, что моделирование приставок *from* и *when* и оператора *to* происходит с помощью выражений на дугах, а не самостоятельных фрагментов, как для приставки *provided*.

Сеть для E-перехода может содержать иерархические конструкции, если блок E-перехода содержал вызов процедуры или функции. Однако структура сети, представляющей блок E-перехода, не влияет на способ моделирования таймеров. Поэтому мы по-прежнему можем не рассматривать иерархические конструкции в раскрашенных сетях и считать, что блок E-перехода моделируется некоторой сетью, заключенной между переходами *start* и *end*. Фрагмент сети, соответствующий приставке *delay*, помещается между фрагментом для приставки *provided* и переходом *start*, так как включение таймера происходит, когда условие возможности E-перехода проверено и выполняется. Структура фрагмента зависит от выбора временного механизма.

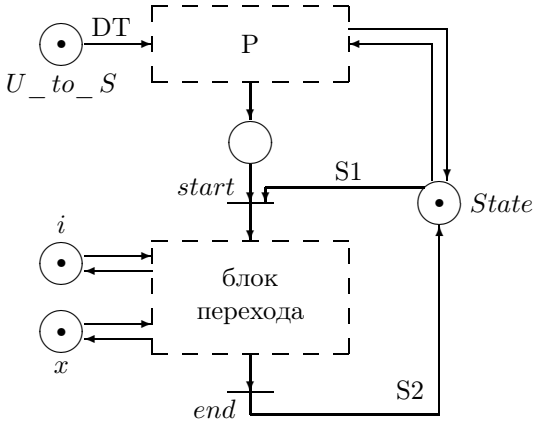

Рис. 2. Схема сети, моделирующей E-переход

4.1. Моделирование Estelle-таймеров в терминах временного механизма Мерлина

При разработке алгоритма трансляции Estelle-спецификаций в раскрашенные сети [12] было решено, что временной механизм Мерлина наиболее удобен для моделирования приставок *delay*. Включение таймеров задержки E-переходов, которые стали возможны после некоторого такта вычислений, и выключение таймеров E-переходов, условие возможности которых было нарушено в результате выполнения такта вычислений, должно происходить во время фазы управления, до того, как начнется следующий шаг вычислений. Поэтому потребовалось дополнительно ввести в модель раскрашенных сетей приоритеты для переходов [5, 12]. Введение приоритетов происходит по той же схеме, что и введение временного механизма Мерлина. У переходов раскрашенной сети появляется новый тип пометки — приоритет. Готовность перехода к срабатыванию определяется на основании трех параметров. Переход должен быть реализуем, т. е. быть возможным по цвету и оставаться таким не менее $d1$ единиц времени, и его приоритет должен быть не меньше, чем у остальных реализуемых переходов.

Для моделирования приставки *delay(d1,d2)* используется подсеть, изображенная на рис. 3. Можно сказать, что подсеть, моделирующая E-переход с задержкой, состоит из трех частей: первая часть представляет *provided*, вторая реализует механизм задержки E-перехода, а третья — блок E-перехода.

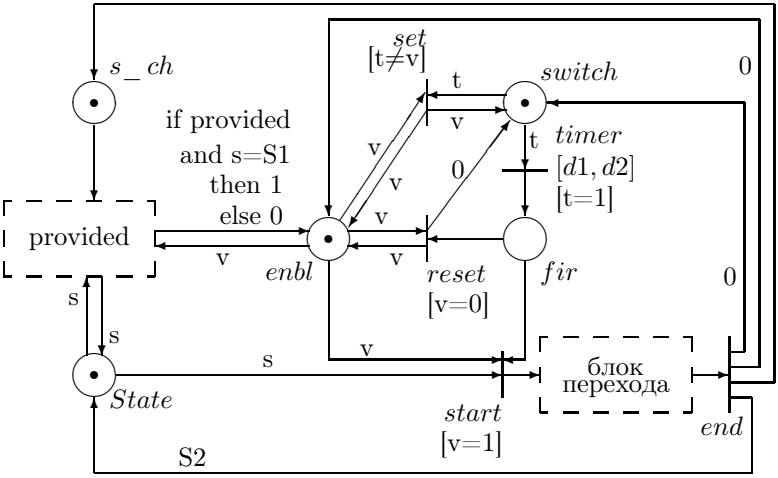

Рис. 3. Моделирование приставки *delay(d1, d2)*

Фишки в местах *switch* и *enbl* принимают значения 0 или 1. Наличие фишки 0 в месте *switch* означает, что таймер задержки для рассматриваемого E-перехода выключен, а в месте *enbl* — что предикат возможности для E-перехода ложен. Начальная разметка как места *switch*, так и *enbl* — фишка 0. Места fir и s_ch могут содержать бесцветные фишки. Место fir изначально пусто, а s_ch содержит одну фишку.

Время задержки E-перехода отсчитывается переходом *timer* с интервалом срабатывания $[d1, d2]$, все прочие переходы имеют интервал срабатывания $[0, 0]$. Переход *timer* имеет единственное входное место *switch* и не зависит от "внешних" мест, таких как место *State*, которые доступны переходам сети, относящимся к другим E-переходам. Благодаря этому, условие возможности перехода *timer* может нарушить только срабатывание перехода *set*. Переход *set* возможен, если значения фишек в местах *enbl* и *switch* различаются. Следовательно, когда переход *timer* возможен и место *switch* содержит 1, переход *set* может сработать, если и только если место *enbl* содержит 0. Последнее означает, что в результате выполнения какого-то другого E-перехода, условие возможности рассматриваемого E-перехода перестало выполняться, и его таймер должен быть выключен.

Если E-переход некоторого модуля имеет приставку *delay*, то первый переход подсети для приставки *provided* этого E-перехода получает максимальный приоритет. Интервал срабатывания $[0, 0]$ в этом случае не дает нужного результата, так как в сети может существовать переход $timer$, ожидающий $d2$ единиц времени. Чтобы проверка истинности приставки *provided* происходила только после завершения очередного шага вычислений, для каждого E-перехода с задержкой создается место s_ch — входное для первого перехода подсети для приставки *provided*. Кроме того, место s_ch становится выходным для служебных переходов *end* всех подсетей, которые моделируют E-переходы, принадлежащие тому же модулю, что и рассматриваемый. Таким образом, в начале моделирования спецификации происходит проверка истинности приставок *provided* всех E-переходов с задержками, которая использует фишки, содержащиеся в местах s_ch при начальной разметке. В дальнейшем проверка происходит только после того, как завершается выполнение некоторого E-перехода.

В результате проверки истинности приставки *provided*, из места s_ch изымается бесцветная фишка, а фишка, находившаяся в месте *enbl*, заменяется на фишку, значение которой совпадает со значением *provided*. Все остальные фишки, значения которых использовались подсетью для приставки *provided*, возвращаются в соответствующие места без изменений.

Рассмотрим работу сети, моделирующей приставку *delay(d1,d2)*. Пусть E-переход впервые стал возможен. Тогда после проверки истинности приставки *provided* место *enbl* содержит фишку со значением 1, а место *switch* — со значением 0. Возможно срабатывание перехода *set*, при котором фишка в месте *switch* заменяется на фишку со значением 1, а в место *enbl* фишка возвращается без изменения. С этого момента становится возможным переход $timer$. Срабатывание последнего добавляет фишку в место fir. Если в этот момент место *enbl* все еще содержит фишку 1, то срабатывание перехода *start* начнет выполнение E-перехода.

Если рассматриваемый E-переход перестал быть возможен в результате выполнения другого E-перехода, то в месте *enbl* фишка со значением 1 заменяется фишкой 0, что делает невозможным срабатывание перехода *start*. После этого необходимо выключить таймер E-перехода и восстановить разметку подсети, моделирующей приставку *delay*. Если срабатывание перехода $timer$ еще не произошло, возможен переход *set*.

Его срабатывание замещает фишку 1 в месте *switch* на фишку 0, что делает невозможным срабатывание перехода *timer*. В противном случае, срабатывание перехода *reset* изымает бесцветную фишку из места fir и помещает фишку 0 в место *switch*, восстанавливая исходную разметку подсети для приставки *delay*.

Моделирование приставок *delay(d1)* и *delay(d1,*)* отличается от моделирования приставки *delay(d1,d2)* выбором интервала срабатывания перехода *timer*. Приставке *delay(d1)* соответствует интервал $[d1, d1]$, а приставке *delay(d1,*)* — $[d1, \infty]$.

4.2. Моделирование Estelle-таймеров в терминах временного механизма Йенсена

Как было сказано выше, выбор другого временного механизма влияет на структуру подсети, моделирующей задержку E-перехода. Общая структура подсети для E-перехода с задержкой сохраняется. На рис. 4 изображена подсеть для приставки *delay(d1,d2)*, использующая временной механизм Йенсена. Чтобы не загромождать рисунок, сеть не содержит мест, не участвующих в моделировании задержки.

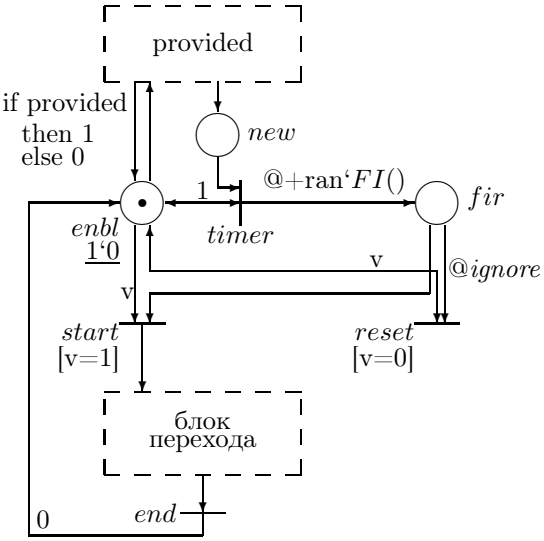

Рис. 4. Моделирование приставки *delay(d1,d2)*

Временные штампы несут только фишки в месте fir. При определении временного штампа фишки используется специальное множество цветов:

color FI = int with $d1$ .. $d2$ declare ran;

Для множества FI (от английского термина *firing interval* — интервал срабатывания) определена функция ran‘$FI()$, которая возвращает произвольное целое значение между $d1$ и $d2$. Фишки во всех остальных местах не несут временных штампов.

Значение фишки в месте *enbl* по-прежнему совпадает со значением приставки *provided*. Место fir может содержать фишку, не имеющую цвета, но несущую временной штамп. Переход *timer* может сработать, когда место *enbl* содержит 1. При этом нет необходимости изолировать переход *timer* от внешних мест, так как нет задержки срабатывания перехода, которая начала бы отсчитываться заново после перевычисления *provided*. Однако необходимо гарантировать, что срабатывание перехода *timer* происходит только один раз, когда E-переход становится возможным. Для этой цели в сети на рис. 4 присутствует место *new*. В это место фишка помещается одновременно с заменой фишки 0 в месте *enbl* на фишку 1.

Срабатывание перехода *timer* помещает фишку в место fir. Таким образом, временной штамп фишки в месте fir определяется как текущее модельное время плюс некоторое значение из интервала [$d1$, $d2$]. Фишка станет доступна переходам сети не ранее, чем через $d1$, и не позднее, чем через $d2$ единиц времени с момента своего создания. Если нарушения условия возможности E-перехода не происходит, переход *start* срабатывает сразу, как только фишка в месте fir становится доступной.

Если произошло нарушение условия возможности рассматриваемого E-перехода в результате выполнения какого-то другого E-перехода, фишку из места fir необходимо удалить. На рис. 4 изображена сеть, в которой фишка из места fir удаляется с помощью временной пометки @*ignore*. Данная пометка приписывается входной дуге перехода *reset*, который срабатывает сразу после нарушения условия возможности E-перехода. Таким образом, в месте fir находится всегда только одна фишка, помещенная туда при последнем включении таймера E-перехода.

Рассмотрим также способ моделирования приставки *delay(d1,d2)*, который не использует временных пометок на входных дугах перехо-

дов [1]. При таком подходе никакой переход не может извлечь фишку из места раньше, чем текущее модельное время станет равно ее временному штампу. Таким образом, условие возможности E-перехода может несколько раз нарушиться и восстановиться за время, пока фишка, помещенная в место fir первой, остается недоступной. В результате в месте fir окажется несколько фишек. Срабатывание перехода *start* должно быть связано с последней фишкой, которая станет доступной позже всех среди фишек, находящихся в месте fir. Чтобы иметь возможность определить, является ли ставшая доступной фишка последней, предлагается использовать дополнительное место *counter*, куда будет заноситься информация о числе фишек в fir. Множеством цветов для места *counter* является множество целых чисел. Начальная разметка — фишка 0.

Сначала рассмотрим функционирование сети для приставки *delay* подробно для каждого из возможных сценариев нарушения/восстановления условия возможности E-перехода. Предположим, что E-переход впервые стал возможен в некоторый момент времени. В сети этому событию будет соответствовать срабатывание всех переходов, относящихся к подсети для приставки *provided*. При этом бесцветная фишка помещается в место *new*, а фишка 0 в месте *enbl* заменяется на фишку 1. После этого становится реализуемым переход *timer*. Его срабатывание изымает фишку из места *new*, помещает в место fir фишку с некоторым временным штампом и заменяет фишку в месте *counter* на фишку, значение которой на единицу больше, чем у находившейся там фишки (0 на 1, если E-переход впервые стал возможен).

На рис. 5 показано, как место *counter* соединено с переходами *timer* и *start*. Пусть условие возможности E-перехода не было нарушено. Тогда в момент времени, когда фишка в месте fir станет доступной, все входные места перехода *start* содержат фишки, и его спусковая функция выполнена, т. е. переход *start* готов и может сработать. Заметим, что условие [v=1 & n=1] выполняется в двух случаях. Во-первых, когда условие возможности E-перехода, однажды выполненное, более не нарушалось в течение всего времени задержки. Во-вторых, когда условие возможности E-перехода неоднократно нарушалось и восстанавливалось, и из места fir удалены все фишки кроме последней.

[1] Помимо того, что моделирование таймера E-перехода без использования временных пометок на входных дугах переходов является интересной задачей, приведенный способ позволяет моделировать E-переходы с задержками и в тех версиях системы Design/CPN, где такие пометки не реализованы.

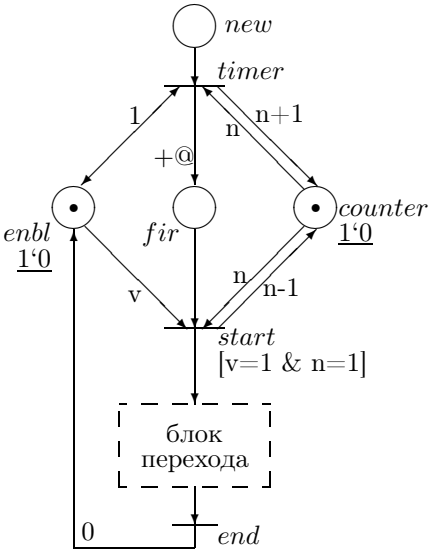

Рис. 5. Введение места для подсчета числа установок таймера

Рассмотрим, как происходит удаление фишек, которые накопились в месте fir. Если в итоге мы имеем в месте *enbl* фишку 0, т. е. E-переход невозможен, то переход сети *start* также невозможен. Фишки в месте fir становятся доступными одна за другой в том порядке, в каком они помещались в это место. При появлении в месте fir доступной фишки переход *reset*1 становится готовым (рис. 6, а). Срабатывание этого перехода удаляет фишку из места fir и уменьшает значение фишки в месте *counter* на единицу.

Переход *reset*2 на рис. 6, б возможен, когда место *enbl* содержит 1, а значение фишки в месте *counter* строго больше единицы. Последнее означает, что в месте fir стала доступной фишка, помещенная туда при какой-то из предыдущих установок таймера E-перехода. Последовательные срабатывания перехода *reset*2 удаляют из места fir все фишки кроме последней и уменьшают значение фишки в месте *counter* до единицы. При таких условия становится возможным переход *start*, срабатывание которого произойдет, когда последняя фишка в месте fir станет доступной. Очевидно, что переходы *reset*1 и *reset*2 можно заменить одним со спусковой функцией [(v=0) $\vee$ ((v=1) & (n>1))].

На рис. 7 подсеть, моделирующая приставку *delay(d1,d2)*, приведена

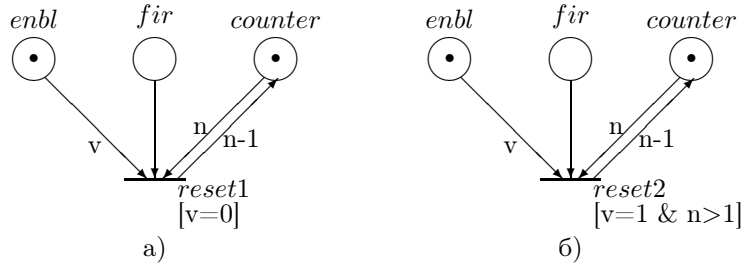

Рис. 6. Удаление фишек из места fir

полностью. Пометка +@ на дуге из места fir в переход *timer* символически обозначает временную пометку @+ran‘$FI()$, осуществляющую выбор произвольного значения из целочисленного интервала [$d1$, $d2$].

Моделирование приставок *delay(d1)* и *delay(d1,*)* отличается от моделирования приставки *delay(d1,d2)* выбором временной пометки на дуге от перехода *timer* к месту fir. Приставке *delay(d1)* соответствует пометка @ + $d1$, т. е. к текущему модельному времени добавляется всегда одно и то же значение — $d1$. Сложнее обстоит дело с приставкой *delay(d1,*)*, так как каждая разметка в раскрашенной сети, снабженной временным механизмом Йенсена, существует в рамках замкнутого временного интервала [18]. Нам представляется разумным использовать при моделировании приставки *delay(d1,*)* выбор значения задержки из интервала [$d1$, $d2$], где значение $d2$ много больше, чем $d1$. Такое допущение не является сильным нарушением временной семантики Estelle, так как несмотря на существование обозначения для бесконечности, E-переход не может вечно оставаться возможным и не выполниться [4]. Если E-переход с задержкой был возможным хотя бы в течение одного такта вычислений, то он будет предлагаться к выполнению на каждом следующем такте и рано или поздно выполнится, если только не перестанет быть возможным.

До сих пор за рамками рассмотрения осталась организация фазы управления, в ходе которой после завершения очередного шага вычислений осуществляется проверка условий возможности всех E-переходов. В сети, моделирующей спецификацию, шаг вычислений завершается возвращением фишки в место *State*, где отображается управляющее состояние модуля. При этом становятся готовыми как первые переходы подсетей для приставок *provided*, так и переходы *start* в подсетях тех E-переходов, отсчет задержек которых закончился. Необходимо га-

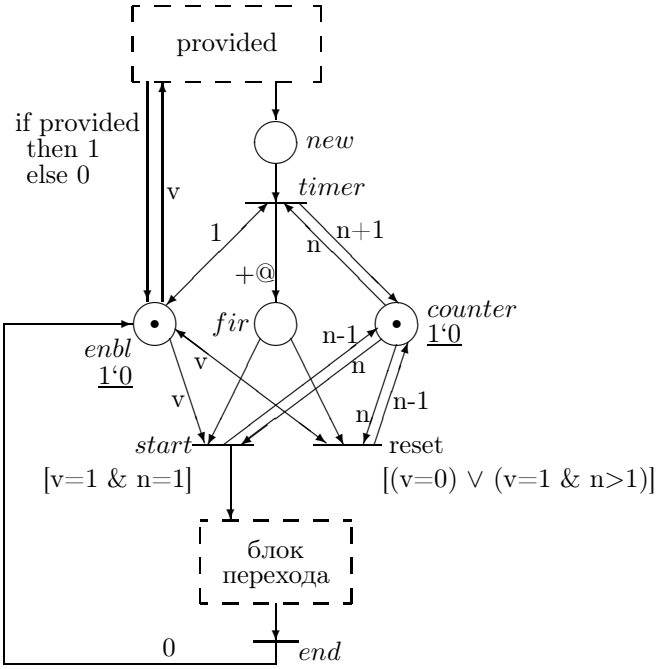

Рис. 7. Моделирование приставки *delay(d1,d2)* без пометок на входных дугах

рантировать, что срабатывание какого-нибудь перехода *start* и, следовательно, начало следующего шага вычислений не произойдет раньше, чем завершится проверка условий возможности E-переходов. Для этого можно использовать тот же механизм, что и в п. 4.1. Однако в этом случае необходимо расширить раскрашенные сети приоритетами, что не предусмотрено в системе Design/CPN.

На рис. 8 показано, каким образом моделирование фазы управления может быть организовано без использования в сети переходов с приоритетами. Подсеть, сопоставляемая E-переходу с приставкой *delay*, содержит два дополнительных места: s_ch и p_ch. Как и ранее, место s_ch — входное для первого перехода подсети для приставки *provided* и выходное для служебных переходов *end* всех подсетей, которые моделируют E-переходы рассматриваемого модуля. Место p_ch становится выходным для последнего перехода подсети для приставки *provided*. Кроме того, место p_ch является входным для перехода *start* и для

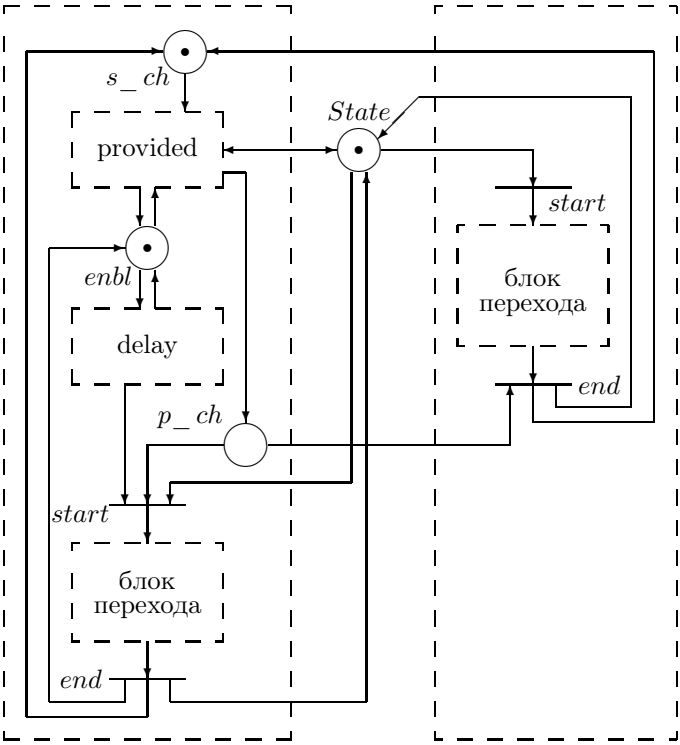

Рис. 8. Моделирование фазы управления

всех переходов *end*, кроме относящегося к подсети того же E-перехода. Места $s\_\,ch$ и $p\_\,ch$ могут содержать бесцветные фишки. При начальной разметке место $s\_\,ch$ содержит фишку, а место $p\_\,ch$ — пустое.

Таким образом, при проверке истинности приставки *provided* бесцветная фишка изымается из места $s\_\,ch$ и добавляется в место $p\_\,ch$. Если условие возможности E-перехода было выполнено, и время задержки истекло, переход *start* сработает. Если E-переход был невозможен, то фишка в месте $p\_\,ch$ будет оставаться до тех пор, пока не произойдет выполнение другого E-перехода. Осталось изучить случай, когда условие возможности рассматриваемого E-перехода было выполнено, и началось выполнение другого E-перехода, пока отсчитывалась задержка. По построению сети, пока не завершилось функционирование подсети, моделирующей какой-нибудь E-переход, место *State* остается

пустым. Так как *State* — входное для всех переходов *start*, выполнение E-перехода с задержкой не может начаться, даже если фишка в месте fir уже стала доступна. Срабатывание перехода *end*, относящегося к подсети выполняемого E-перехода, изымает фишку из места p_ch и помещает в s_ch. В результате переход *start* снова становится невозможен, и начинается проверка приставки *provided*.

В заключение остается заметить, что описанную конструкцию можно использовать и в модели раскрашенных сетей, расширенных временным механизмом Мерлина. Использование предложенной конструкции приводит к увеличению размеров сети: добавляются места — по одному на каждый переход с приставкой *delay*, — и дуги, связывающие эти места с переходами сети. Однако отпадает необходимость присваивать максимальный приоритет первому переходу подсети для приставки *provided*, что упрощает построение моделирующей сети, если спецификация содержала E-переходы как с приставками *delay*, так и с приставками *priority*.

ЗАКЛЮЧЕНИЕ

В работе [12] был описан алгоритм трансляции Estelle-спецификаций в раскрашенные сети Петри, обогащенные временными конструкциями и приоритетами. Для расширения раскрашенных сетей использовался временной механизм, предложенный Мерлином [20]. Данный механизм определил способ моделирования Estelle-таймеров, который в дальнейшем был реализован в рамках экспериментальной системы EPV.

В настоящий момент в ИСИ СО РАН ведется работа по переносу процедуры трансляции Estelle-спецификаций в среду системы Design/CPN, где реализован ряд методов анализа раскрашенных сетей. Однако система Design/CPN использует модель времени, отличную от использованной в работе [12].

В данной работе предложен способ моделирования Estelle-таймеров в терминах временной модели, принятой в системе Design/CPN. Кроме того, описан метод организации фазы управления, использующий только дополнительные структурные построения. Благодаря этому, при моделировании Estelle-переходов с задержками не возникает необходимости дополнительно усложнять раскрашенные сети, наряду со временем используя приоритеты.

Автор выражает благодарность В. А. Непомнящему за постоянное внимание и полезные замечания, сделанные в ходе написания этой рабо-

ты, и Т. Г. Чуриной за множество плодотворных обсуждений тонкостей временного поведения Estelle-спецификаций и сетей Петри.

СПИСОК ЛИТЕРАТУРЫ

1. **Зайцев С. С.** Описание и реализация протоколов сетей ЭВМ. — М.: Наука, 1989.
2. **Budkowski S., Dembinski P.** An introduction to Estelle: a specification language for distributed systems // Computer Networks. — 1988. — Vol. 14. — P. 3—23.
3. **Ferenc B., Hogrefe D., Sarma A.** SDL with applikations from protocol specification. — Englewood Cliffs, NJ: Prentice Hall, 1991.
4. **Information** Processing Systems — Open Systems Interaction — ESTELLE: A Formal Description Technique Based on an Extended State Transition Model: Inernational standard. — ISO 9074, 1989.
5. **Котов В. Е.** Сети Петри. — М.: Наука, 1984.
6. **Richier J. L., Rodriguez C., Sifakis J., Voiron J.** Verification in XESAR of the Sliding Window protocol // Proc. IFIP Intern. Sympos. on Protocol Specification, Testing and Verification VII. — Amsterdam: North-Holland, 1987. — P. 235—248.
7. **Dimitrov V., Petkov A.** Verification oriented Estelle specification of communication protocols // Research into Networks and Distributed Applications. — Amsterdam: North-Holland, 1988. — P. 953—960.
8. **Lai R., Jirachiefpattana A.** Verifying Estelle protocol specifications using numerical Petri nets // Comput. Syst. Sci & Eng. — 1996. — Vol. 11, N 1. — P. 15—33.
9. **Fisher J., Dimitrov E.** Verification of SDL'92 specifications using extended Petri nets // Proc. IFIP 15th Intern. Conf. on Protocol Specification, Testing and Verification. — Warsaw, Poland, 1995. — P. 455—458.
10. **Kettunen E., Montonen E., Tuuliniemi T.** An interactive PrT-Net tool for verification of SDL-specifications: Techn. Rep. No. 3. — Helsinki Univ. of Technology, Digital System Laboratory, 1988.
11. **Bause F., Kabutz H., Kemper P., Kritzinger P.** SDL and Petri net perfomance analysis of communicating systems // Proc. IFIP 15th Intern. Symp. on Protocol Specification, Testing and Verification. — Warsaw, Poland, 1995. — P. 259—272.
12. **Верификация** Estelle-спецификаций распределенных систем посредством раскрашенных сетей Петри // Непомнящий В. А., Быстров А. В. и др. — Новосибирск, 1998. — 140 с.
13. **Churina T. G., Okunishnikova E. V.** Coloured Petri nets approach to the validation of Estelle specifications // Proc. of Workshop on Concurrency, Specification and Programming. — Warsaw, Poland, 1997. — P. 25—36.
14. **Churina T. G., Okunishnikova E. V.** Modelling Estelle specifications using coloured Petri nets //Joint NCC&IIS Bull., Comp. Science, 8, 1998. — P. 19—38.
15. **Чурина Т. Г.** Способ построения раскрашенных сетей Петри, моделирующих SDL-системы. — Препринт 56, Институт систем информатики им. А.П.Ершова СО РАН, Новосибирск, 1998.
16. **Jensen K.** Coloured Petri nets: A high level language for system design and analysis // Lect. Notes Comput. Sci. — 1991. — Vol. 483 — P. 343—416.
17. **Jensen K.** Coloured Petri nets: Basic concepts, analysis methods and practical use. — Berlin a. o.: Springer-Verlag, 1996. — Vol. 1. Basic concepts.
18. **Jensen K.** Coloured Petri nets: Basic concepts, analysis methods and practical use. — Berlin a. o.: Springer-Verlag, 1996. — Vol. 2. Analysis methods.

19. **Jensen K.** Coloured Petri nets: Basic concepts, analysis methods and practical use. — Berlin a. o.: Springer-Verlag, 1997. — Vol. 3. Practical use.
20. **Berthomieu B., Diaz M.** Modelling and verification of time dependent systems using time Petri nets // IEEE Transact. on Software Eng. — 1991. — Vol. 17, N 3. — P. 259—273.
21. **Kristensen L. M., Christensen S., Jensen K.** The practitioner's guide to coloured Petri nets // International Journal on Software Tools for Technology Transfer — 1998. — Vol. 2, N 2. — P. 98—132.

Е. В. Окунишникова

ПРЕДСТАВЛЕНИЕ ВРЕМЕННЫХ КОНСТРУКЦИЙ ESTELLE В РАЗЛИЧНЫХ МОДЕЛЯХ ВРЕМЕННЫХ СЕТЕЙ ПЕТРИ

Препринт
70

Рукопись поступила в редакцию 06.12.1999
Рецензент И. В. Тарасюк
Редактор З. В. Скок

Подписано в печать 24.12.1999
Формат бумаги 60×84 1/16 Объем 1,8 уч.-изд.л., 2,0 п.л.
Тираж 100 экз.

ЗАО РИЦ "Прайс-курьер", 630090, пр. Акад. Лаврентьева, 6