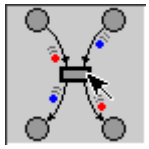
Министерство транспорта и связи Украины
Одесская национальная академия связи им. А.С. Попова

Кафедра Сетей связи



Д.А. Зайцев, Т.Р. Шмелева

Моделирование телекоммуникационных систем в CPN Tools

Учебное пособие по курсу
«Математическое моделирование информационных систем»
для подготовки магистров в отрасли связи

УТВЕРЖДЕНО
на Совете факультета
Информационных сетей
Протокол № 5
от 16.11.2006

Одесса 2008

УДК 621.39, 004.7

Рецензенты: проф. В.А. Кудряшова,
к.т.н., доц. И.А. Трегубова

Составители: д.т.н., проф. Д.А. Зайцев,
аспирант Т.Р. Шмелева

Zaitsev D.A., Shmeleva T.R. Simulating Telecommunication Systems with CPN Tools: Students' book. – Odessa: ONAT, 2006. – 60 p.

Перевод с английского:
магистранты факультета ТКС
Т.В. Кольвастр[1], Г.И. Дойжа[2]

Представлено описание возможностей моделирующей системы CPN Tools. Система разработана в Университете Орхуса (Дания) и используется для моделирования телекоммуникационных систем и сетей в курсе «Математическое моделирование информационных систем». Языком описания модели являются раскрашенные сети Петри. В настоящем пособии также изучен пример построения и исследования модели коммутируемой сети Ethernet.

Одобрено
на заседании кафедры
Сетей связи
Протокол № 4 от
10.11.2006

[1] *ICQ*: 323–989–272, *e-mail*: tatyana–ONAC@yandex.ru
[2] *ICQ*: 406–926–201, *e-mail*: hani_13@mail.ru

Содержание

Введение

CPN Tools – это специальная моделирующая система, которая использует язык сетей Петри для описания моделей. Система была разработана в Университете Орхуса в Дании и свободно распространяется для некоммерческих организаций через сайт http://www.daimi.au.dk/CPNTools/. Уровень предоставляемого сервиса позволяет классифицировать CPN Tools как промышленную моделирующую систему. Она была использована в большом количестве реальных проектов, особенно в области телекоммуникаций. В последнее время корпорация Nokia применяет CPN Tools для управляемой моделью разработки нового поколения мобильных телефонов.

1. Класс сетей Петри, реализованный в CPN Tools

CPN Tools предлагает очень мощный класс сетей Петри для описания моделей. Согласно стандартной классификации такие сети называют иерархическими временными раскрашенными сетями Петри. Было доказано, что они эквивалентны машине Тьюринга и составляет универсальную алгоритмическую систему. Таким образом, произвольный объект может быть описан с помощью этого класса сетей.

Простейшая концепция раскрашенной сети Петри использует различные типы фишек. Тип фишки определен натуральным числом и визуально представлен как цвет: 1 – красный, 2–синий, 3–зеленый, и т.д. Концепция раскрашенной сети Петри CPN Tools более сложная. Такие сети часто называют обобщенными раскрашенными сетями, так как тип фишки представлен абстрактным типом данных, как в языках программирования. Термин «раскрашенная» сохраняется исторически, но теперь очень трудно представить такие «цвета» визуально.

Временные сети Петри используют понятие модельного времени для описания продолжительности действий в реальных объектах. В отличие от классических сетей Петри, где срабатывание перехода происходит мгновенно, срабатывание перехода во временной сети связано с определенной продолжительностью или временной задержкой. Это позволяет анализировать временные характеристики реальных объектов, например, время отклика как характеристику качества обслуживания сети.

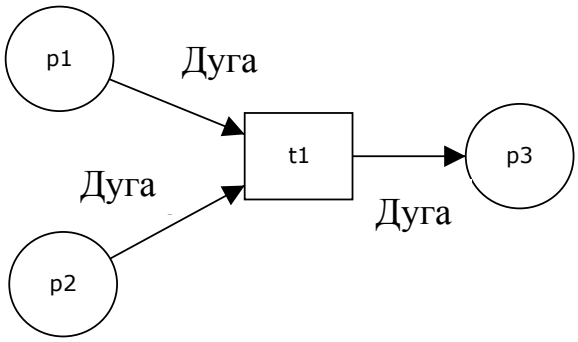
Иерархические сети обеспечивают построение сложных моделей. В таких сетях элемент может быть представлен другой сетью. В CPN Tools переход может быть замещен дополнительной сетью. Таким образом, получается вложенная конструкция: сеть внутри сети. Количество уровней иерархии не имеет принципиальных ограничений. Отметим, что, такой подход широко распространен в языках программирования, где процедуры используются для управления сложностью.

1.1. Граф сети Петри и язык CPN ML

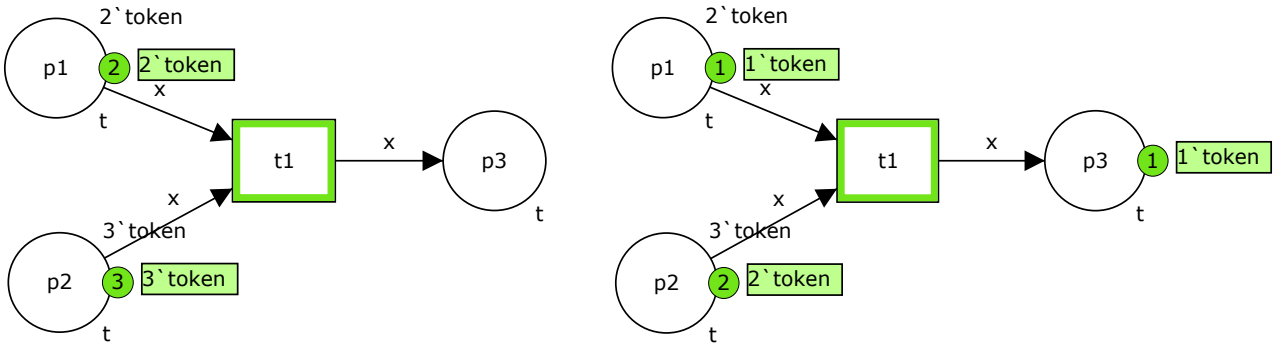
В CPN Tools язык описания моделей представляет собой сочетание графа сети Петри и языка программирования CPN ML.

Граф сети Петри – двудольный ориентированный граф. Он состоит из вершин двух типов: позиций, представленных кругами или овалами и переходов, представленных прямоугольниками:

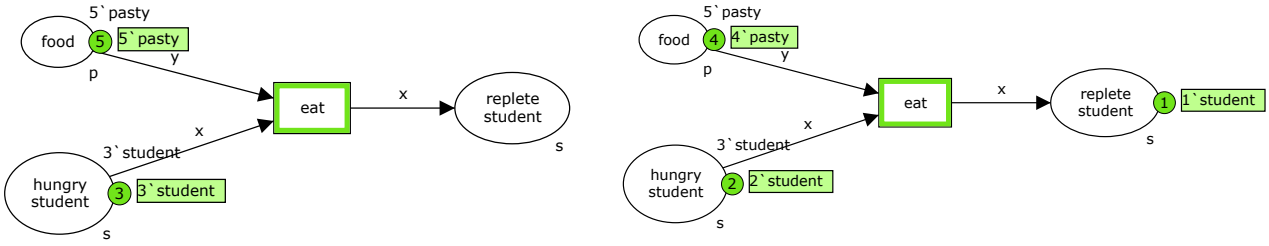
Дуги соединяют позиции и переходы:



В сетях Петри также используется понятие фишки. Фишка – динамический элемент, расположенный в позициях, который перемещается в результате срабатывания перехода:



В классических сетях Петри все фишки одинаковые и элементарные. В раскрашенных сетях Петри фишки различные. Рассмотрим пример студентов, обедающих пирогами. Имеется две фишки: `student (студент)`, `pasty (пирог)`. Голодный студент становится сытым после того, как съедает пирог:



В CPN Tools включен специальный язык программирования для описания атрибутов элементов сети. Этот язык обеспечивает описание множеств цветов, переменных, констант, функций и процедур. В вышеуказанном примере используются следующие описания:

```
colset s=unit with student;
colset p=unit with pasty;
var x:s;
var y:p;
```

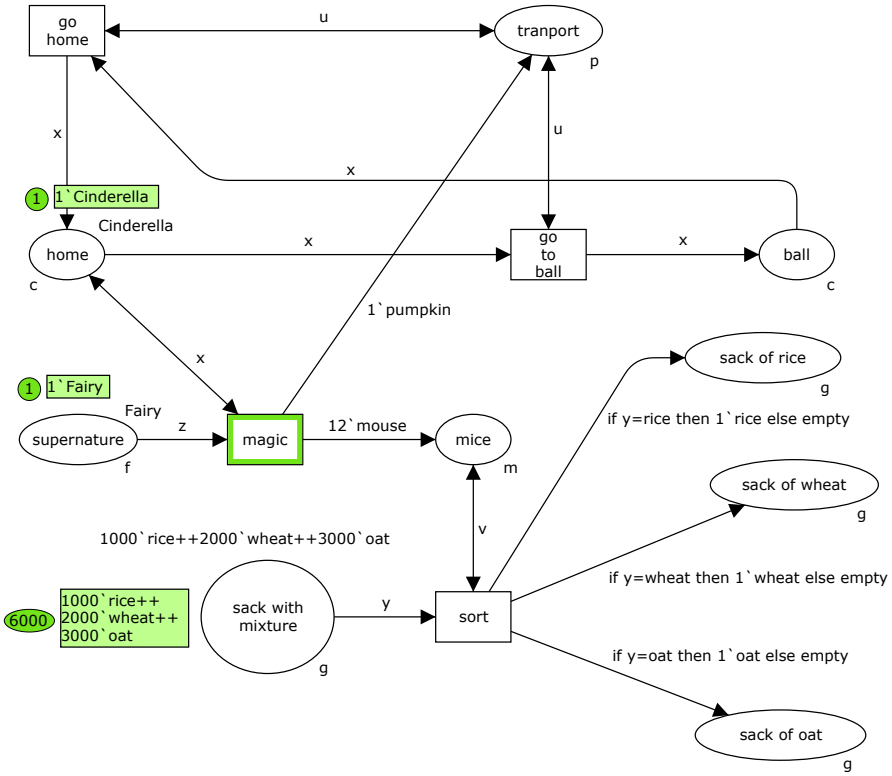
Определены два множества цветов: множество `s` с элементом `student` и множество `p` с элементом `pasty`. Позиции «`hungry student`» (`голодный студент`) и «`replete student`» (`сытый студент`) имеют множество цветов `s` с фишкой `student`. Позиция `food` (`пища`) имеет множество цветов `p` с фишкой `pasty`. Чтобы запустить переход `eat` (`кушать`), необходимо наличие двух фишек: `student` и `pasty`. Переменные `x` и `y` используются, чтобы извлечь фишки из входных позиций и поместить новую фишку в выходную позицию.

Пример иллюстрирует способ, с помощью которого могут быть обработаны различные типы фишек. В моделях телекоммуникационных систем множества цветов могут быть более сложными и представлять, например, для случая Ethernet, кадры, строки таблицы коммутации, и т.д.

В отличие от классических сетей Петри, в раскрашенных сетях у позиций, переходов и дуг есть свои атрибуты. В вышеуказанном примере позиция имеет имя – «`hungry student`», множество цветов – `s`, начальную маркировку – `3`student` и конечную маркировку – `2`student` после того, как один из студентов съел пирог. Переменная `x` позволяет выбрать произвольного студента согласно множеству цветов переменной; переменная `y` позволяет выбрать произвольный пирог. Студент, который был извлечен переменной `x` из позиции «`hungry student`», будет помещен в позицию «`replete student`», так как выходная дуга перехода `eat`, подписана именем той же самой переменной `x`.

1.2. Пример моделирования

Рассмотрим более сложный пример для предварительного изучения CPN Tools. Пример взят из хорошо известной сказки о Золушке. Мачеха приказала Золушке отсортировать зерна; Фея послала мышей сортировать зерна, в то время как Золушка едет на балл.



В этой раскрашенной сети Петри используются следующие описания множеств цветов и переменных:

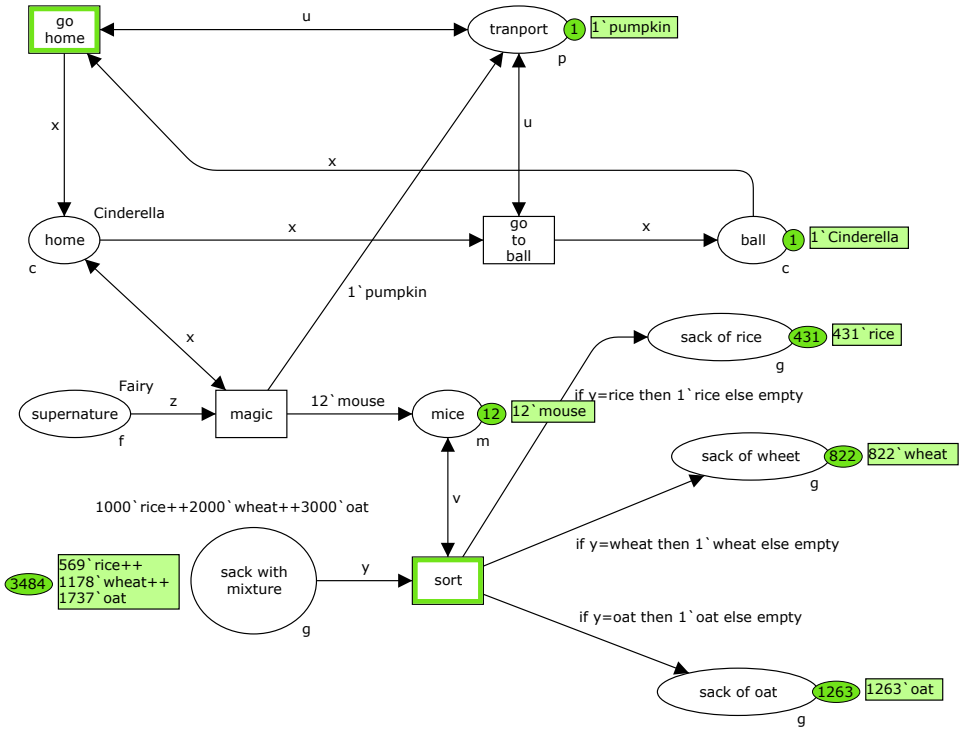
```
colset p=unit with pumpkin;
colset c=unit with Cinderella;
colset g=with rice | wheat | oat;
colset m=unit with mouse;
colset f=unit with Fairy;
var x: c;
var y: g;
var z: f;
var u: p;
var v: m;
```

В примере есть четыре множества цветов: `f` с фишками под названием `Fairy (Фея)`, `c` с фишками под названием `Cinderella (Золушка)`, `p` с фишками под названием `Pumpkin (Тыква)` и `g` с тремя видами фишек (сортами зерен) под названием `rice (рис)`, `wheat (пшеница)`, `oat (овес)`. В начальной маркировке разрешен один переход `Magic (Волшебство)`, который подсвечен зеленым цветом. Разговаривая с Золушкой, Фея создает 12 мышей и 1 тыкву переходом `Magic` и исчезает. Путешествие Золушки на балл и сортировка зерен – параллельные события, и они могут происходить одновременно в любом порядке. Тыква используется для обеспечения возможности срабатывания переходов «`go to ball`» (`ехать на балл`) и «`go home`» (`ехать домой`), чтобы перенести Золушку на балл и обратно. Мыши используются для обеспечения

возможности срабатывания перехода Sort (Сортировка), чтобы выбирать зерна различных сортов.

Обсудим направления дуг и надписи на них. В переходе Magic Cinderella не изменяется, так же как Pumpkin в переходах «go to ball» и «go home», поэтому используются двунаправленные дуги. Остальные дуги – однонаправленные. Дуга, соединяющая позицию с переходом, извлекает фишку соответствующего множества цветов. Фишка извлекается в соответствии с надписью входной дуги перехода. В этом примере все надписи представлены переменными соответствующего множества цветов. Например, у входной дуги перехода sort есть надпись y; y – переменная множества цветов g; таким образом, из позиции «sack with mixture» (мешок со смесью зерен) извлекается произвольное зерно. Более сложные атрибуты входных дуг переходов будут изучены далее.

Выходные дуги переходов создают новые фишки. Новая фишка может быть создана (заново) или может совпадать с любой фишкой, извлеченной из входной позиции. Например, в переходе «go to ball» фишка Cinderella извлекается из позиции home (дом) переменной x по атрибуту входной дуги и та же самая фишка Cinderella будет помещена в позицию ball (балл) в соответствии с атрибутом x на выходной дуге. Переход Magic более сложный: Fairy исчезает после срабатывания этого перехода, потому что она извлечена входной дугой с надписью z, и переменная z не используется в атрибутах выходных дуг; Cinderella в позиции home не изменяется переходом Magic, так как используется двунаправленная дуга с надписью x, то есть, проверяется только наличие фишки в позиции home; 12 mice (мышей) и 1 pumpkin создаются, соответствующие константы подписаны на выходных дугах. Теперь рассмотрим модель после 5000 шагов моделирования:



Золушка прибыла на балл, мыши занимаются своей работой, Фея исчезла. Мыши выбрали 431 зерно риса, 822 зерна пшеницы и 1263 зерна овса. В этой маркировке есть два разрешенных перехода: `sort` и «`go home`». Рассмотрим способ сортировки зерен в этой модели. Переход `sort` извлекает одно зерно в переменную `y`, но это зерно должно быть помещено только в одну из позиций: «`sack of rice`» (`мешок с рисом`), «`sack of wheat`» (`мешок с пшеницей`), «`sack of oat`» (`мешок с овсом`). Атрибуты выходных дуг содержат условия, выбирающие только зерна необходимого сорта, иначе выбирается специальная фишка `Empty` (`пусто`). Фишка `Empty` означает «ничего».

Подробное рассмотрение вышеуказанного примера открывает множество отличий от оригинальной сказки. Например, в данной модели вообще не рассматривается понятие времени и предупреждение Феи о полуночи. Рекомендуется исследовать эту модель и найти все ее недостатки. После изучения CPN Tools можно построить свою собственную модель, полностью соответствующую сказке.

Отметим, что вышеуказанный сказочный пример дает представление об организации моделей телекоммуникационных систем и сетей. Работа мышей похожа на функцию сетевого маршрутизатора. Реальный пример модели коммутируемой сети Ethernet приведен в Приложении. Но для того, чтобы понять, как он работает, необходимо изучить следующие главы данного учебного пособия.

2. Назначение и основные функции CPN Tools

2.1. Назначение CPN Tools

CPN Tools используется для построения и анализа моделей. Это – жизненно важная система для разработки сложных объектов в различных прикладных областях. Она широко применяется для менеджмента в производстве и бизнесе, управления производственными системами и роботами, а также транспортными средствами и ракетами, для планирования военных операций. Полный список реальных применений можно найти на домашней странице CPN Tools http://www.daimi.au.dk/CPNTools/. CPN Tools в настоящее время реализована в ОС Windows и на платформах Unix; она, по существу, является новым поколением ранее использованной системы Design-CPN.

Что касается области телекоммуникаций, CPN Tools применяется для спецификации и верификации протоколов, оценки пропускной способности сетей и качества обслуживания, проектирования телекоммуникационных устройств и сетей. В последнее время корпорация Nokia применяет CPN Tools для управляемой моделью разработки мобильных телефонов нового поколения. Это направление является перспективным и эффективным для проектирования сложных технических устройств. Раньше модели использовались только для оценки характеристик устройств или сетей в процессе их проектирования. В

управляемом моделью проектировании исходная простая модель последовательно преобразуется в заключительные спецификации системы. Процесс такого проектирования состоит в наделении модели всё большим количеством свойств и характеристик реальной системы, до тех пока она не становится технической спецификацией, необходимой для производства или инсталляции. Преимуществом этого подхода является возможность анализа системы на каждом этапе проектирования, оценки ее свойств и характеристик (соответствует ли она требованиям?). Это позволяет проектировать системы, близкие к оптимальным, так как для реальных сложных объектов формальное решение задачи оптимизации трудно реализуемо, а в большинстве случаев практически неосуществимо.

Что касается временных раскрашенных иерархических сетей Петри CPN Tools, они являются универсальной алгоритмической системой, позволяющей описывать произвольные объекты. Кроме того, язык раскрашенных сетей Петри удобен для изучения систем, особенно для систем со сложным взаимодействием между компонентами. Понятие асинхронных событий позволяет описывать системы, сохраняя естественный параллелизм их поведения. Это очень удобно для дальнейшей реализации с использованием параллельных процессоров или компьютеров с архитектурой управления потоками данных.

Наибольшие преимущества применения CPN Tools достигаются при использовании специальных процессоров сетей Петри (как аппаратных, так и программных). В этом случае заключительные спецификации системы в форме раскрашенной сети Петри могут быть загружены непосредственно в такой процессор. Существует ряд известных примеров аппаратных процессоров сетей Петри, например, сигнальные процессоры в контроллерах компании Klashka.

2.2. Основные функции CPN Tools

Основными функциями CPN Tools являются:

– создание (редактирование) моделей;

– анализ поведения моделей с помощью имитации динамики сети Петри;

– построение и анализ пространства состояний модели.

Для создания моделей предусмотрен специальный графический редактор раскрашенных сетей Петри. Редактор позволяет рисовать сети Петри на экране компьютера, вводить атрибуты элементов сети и дополнительные описания на языке CPN ML. Модель может состоять из нескольких страниц. Эти страницы связаны друг с другом для создания иерархической структуры.

Для достаточно простых моделей возможна генерация полного пространства состояний (графа достижимости). Это – лучший способ для верификации, например, телекоммуникационных протоколов. CPN Tools обеспечивает построение пространства состояний и автоматическую генерацию по нему отчёта, который содержит выводы о стандартных свойствах сетей Петри, таких как ограниченность и живость. Кроме того, предусмотрен специальный язык на основе языка CPN ML для описания запросов о

нестандартных свойствах пространства состояний, которые важны для пользователя. К сожалению, для сложных моделей пространство состояний может быть слишком большим, и его построение не представляется возможным.

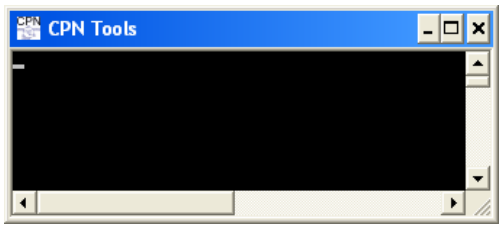
Единственный способ для анализа сложных моделей – это имитация их поведения. CPN Tools предусматривает пошаговую имитацию для поиска и устранения ошибок в разрабатываемой модели, а также автоматическое выполнение определенного количества шагов. Имитация на больших временных интервалах – это путь для статистического анализа поведения модели. Такой подход применяется для оценки характеристик телекоммуникационных сетей, например, пропускной способности и качества обслуживания.

3. Организация интерфейса CPN Tools

В CPN Tools реализован новый принцип графического взаимодействия, основанный на возможностях библиотеки MS Open GL, что позволяет быстро вводить и редактировать модели, используя инструменты на панели управления и в контекстно-зависимых меню. Предусмотрена возможность работы с двумя манипуляторами мышь (двумя руками). В этом случае левая мышь используется для взаимодействия с меню и для выбора инструментов из палитр, а правая мышь используется для рисования и редактирования сетей Петри.

3.1. Области основного окна

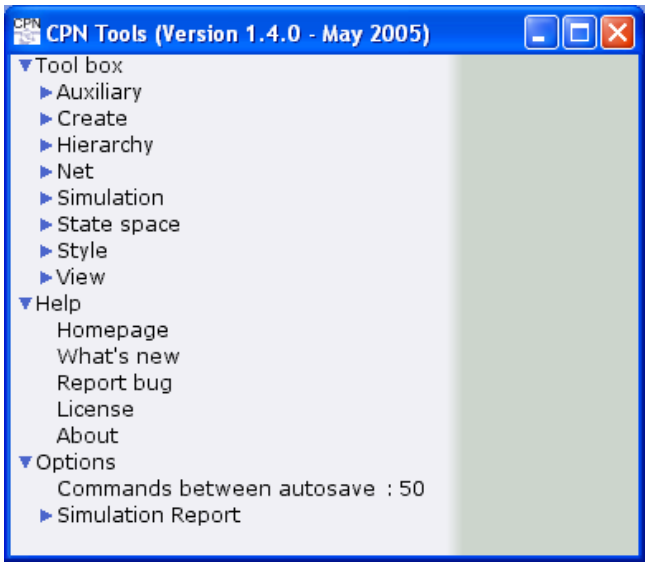
После запуска программы на экране появляются два окна CPN Tools. Одно из них – черное, является вспомогательным и служит для выходных сообщений, когда запускаются подпроцессы:



Второе окно является главным окном CPN Tools:

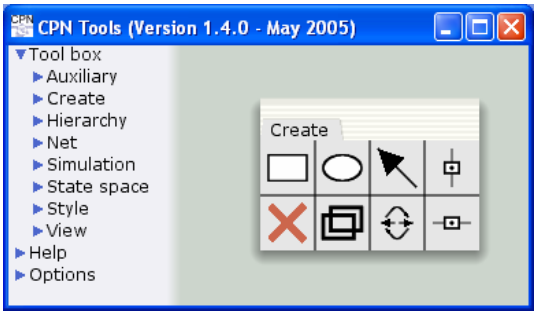
Оно состоит из двух областей: рабочей области – серого цвета и области индекса – белого цвета. Индекс состоит из панели инструментов (Tool box),

системы помощи (Help) и опций (Options), ниже выводятся описания сетей (в вышерассмотренном примере нет загруженной сети). В рабочей области графически представляются страницы сетей. Для взаимодействия с CPN Tools в главном окне есть графический курсор. Маленький треугольник в системе помечает элемент, который может быть раскрыт, нажатием на этот треугольник. Например, можно раскрыть все элементы в индексе:



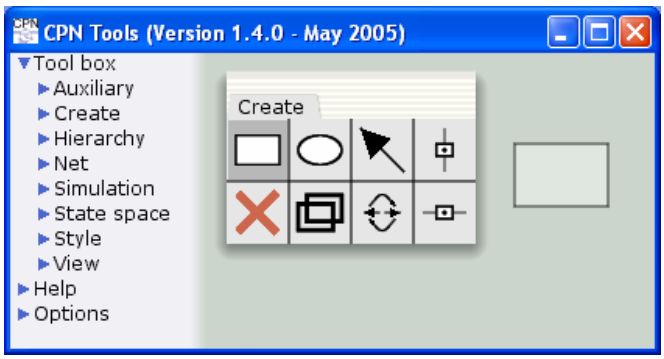
3.2. Работа с инструментами

Чтобы открыть палитру инструментов, необходимо перетащить ее мышкой из индекса в рабочую область. Откроем `Палитру для создания элементов сети (Create)`:



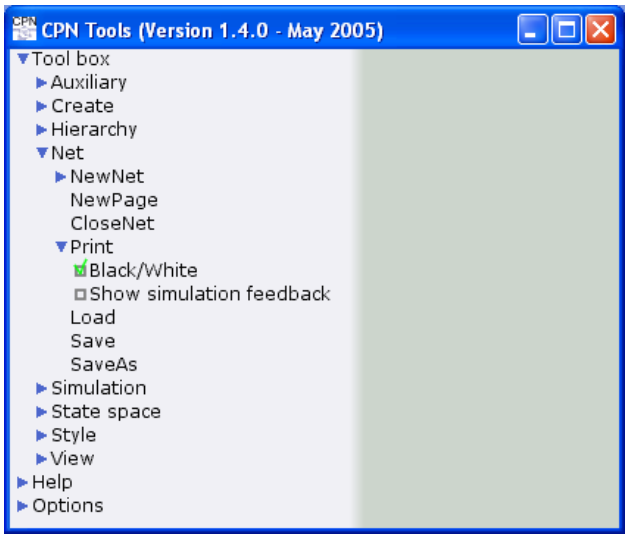
Палитра инструментов появилась в окне.

Чтобы взять инструмент из палитры, необходимо щёлкнуть на нём. Тогда курсор принимает вид соответствующего инструмента. Например, выбираем `Переход` из палитры `Палитра для создания элементов сети`:



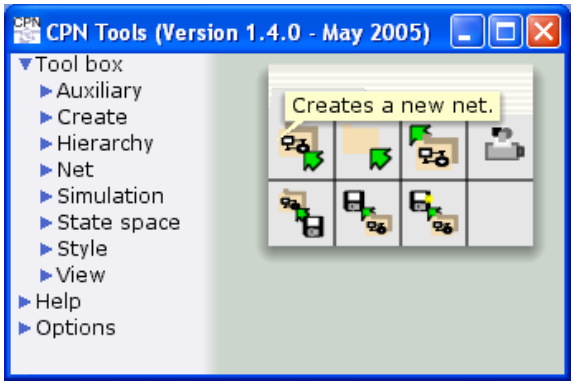
Чтобы отказаться от инструмента, необходимо перетащить его назад на палитру и нажать кнопку мыши или нажать кнопку `Esc` на клавиатуре.

У каждого инструмента есть свои опции, которые могут быть показаны и изменены при помощи нажатия на соответствующий треугольник в индексе. Например, в палитре `Сеть (Net)` можно установить свойства печати: печатать сеть цветную или черно-белую (`Black/White`):

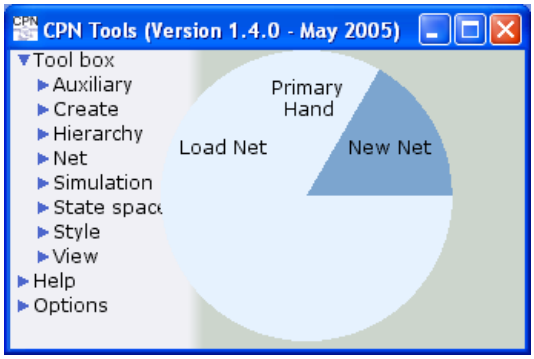


3.3. Контекстно-зависимые меню

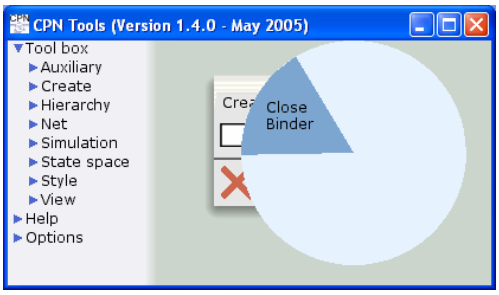
Для удобного взаимодействия CPN Tools предусматривает множество контекстно-зависимых меню, появляющихся на экране при нажатии на правую кнопку мыши. Меню имеет форму круга с названиями секторов. Чтобы сохранять меню на экране, следует удерживать кнопку мыши, передвигая сектор для выбора требуемого элемента. В большинстве случаев элементы контекстно-зависимых меню дублируют инструменты в палитрах. Например, можно создать новую сеть, используя инструмент `Новая сеть` из палитры `Сеть`:



То же самое может быть выполнено с помощью контекстно-зависимого меню:

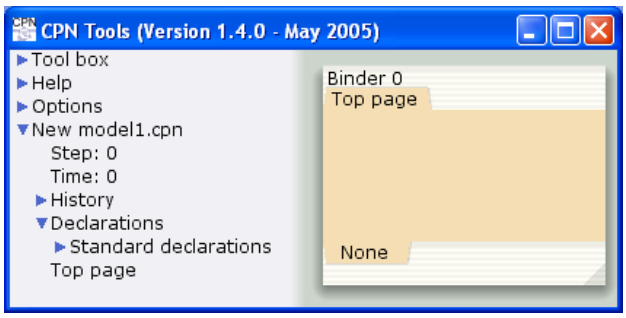


Контекстно-зависимые меню делают взаимодействие более естественным и быстрым. Необходимо только нажимать правую кнопку мыши на объекте и выбирать требуемое действие. Таким способом можно, например, закрыть палитру инструментов:



3.4. Структура моделей

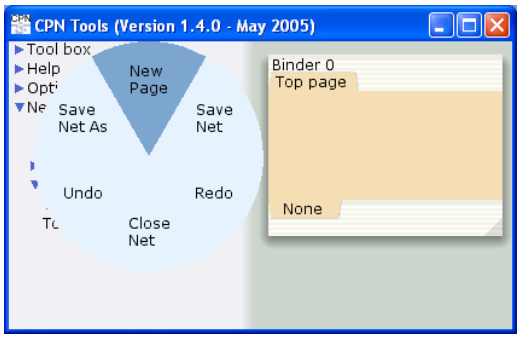
В CPN Tools модели называют сетями (net). Их описания расположены в индексе под стандартными элементами. Рассмотрим новую сеть после ее создания:



Каждая сеть в CPN Tools представлена следующими полями индекса:

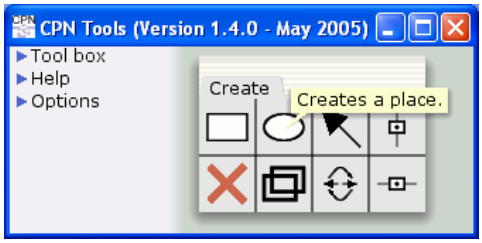
- Имя – имя соответствующего файла с расширением .cpn;
- Шаг – количество шагов, выполненных при имитации;
- Время – текущее модельное время;
- История – список команд, которые были выполнены над сетью;
- Описания – описания множеств цветов, переменных, функций, констант;
- Страницы – названия страниц сети.

В вышерассмотренном окне имя сети – model1.cpn, число шагов и время равны нулю, сеть состоит из одной страницы, названной «`Top page`». Отметим, что «`Top page`» появляется в рабочей области; используя инструменты, можно нарисовать сеть внутри этой страницы. Чтобы открыть страницу сети, необходимо перетащить её из индекса в рабочую область. Для создания новых описаний и новых страниц используются контекстно-зависимые меню:

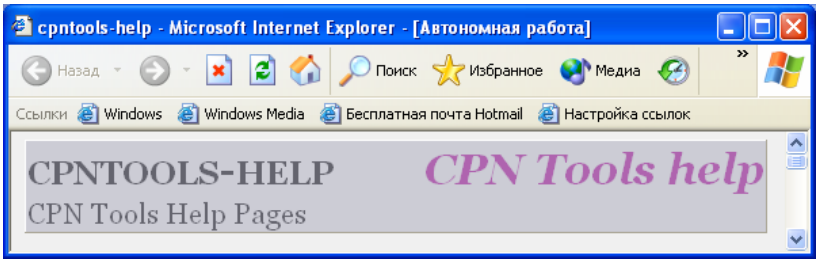


3.5. Структура системы помощи (Help)

CPN Tools предоставляет три вида помощи: всплывающие сообщения, помощь offline и помощь online. Всплывающие сообщения появляются на экране, когда курсор наводится на соответствующий элемент и сохраняется в течение нескольких секунд. Он описывает указанный объект:



При перетаскивании помощи из индекса в рабочую область запускается браузер с помощью в виде гипертекстовой страницы помощи CPN Tools:



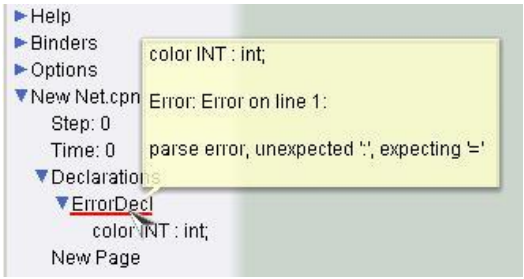
Помощь содержит много информации о CPN Tools, дополненной рассмотрением примеров построения сетей. Более специфические детали и информация, отвечающая современным требованиям, доступны на домашних страницах CPN Tools в Орхусе. В случае необходимости, система помощи запрашивает эту информацию с сайта, но запрос может быть реализован только в случае, если компьютер подключен к Интернету.

3.6. Отображение обратной связи в CPN Tools

CPN Tools обеспечивает графическую обратную связь, которая отражает текущее состояние системы. Существуют следующие виды графической обратной связи:

- Всплывающие сообщения;
- Индикатор состояний;
- Аура (подсветка);
- Изменяющаяся форма курсора.

Всплывающие сообщения – желтый прямоугольник, который предоставляет контекстно-зависимую информацию. Некоторые всплывающие сообщения появляются автоматически, в то время как другие появляются после незначительной задержки, когда курсор передвинут на соответствующий объект. Например, перемещение курсора на описание синтаксической ошибки вызовет появление сообщения об ошибке:



Всплывающие сообщения используются, чтобы показать:

- Сообщения с ошибками во время проверки синтаксиса;
- Сообщения с ошибками при моделировании сети;
- Подсказки для инструментов в палитрах;
- Подробную информацию для индикатора состояний;
- Результат применения инструмента оценки языка CPN ML;

• Полный путь к сохраненной сети. Чтобы посмотреть полный путь, переместите курсор на имя сети в индексе.

Индикатор состояния – цветной индикатор, который время от времени появляется у основания индекса. Чтобы увидеть соответствующее сообщение состояния, необходимо переместить курсор на индикатор:

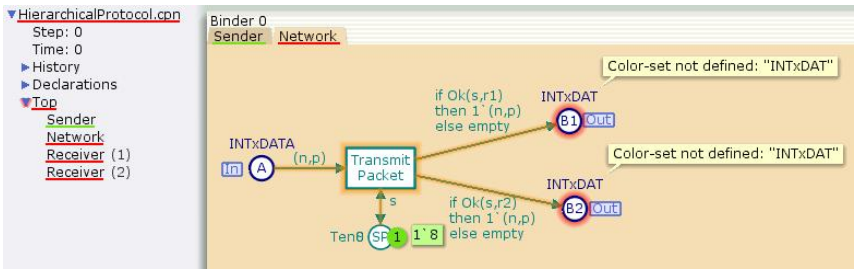
Индикатор состояния бывает следующих цветов:

• Зеленого – обозначает то, что действие было успешно завершено;

• Красного – обозначает то, что при выполнении действия произошла ошибка;

• Светло фиолетового – обозначает то, в данный момент выполняется длительная операция, такое как продолжительное моделирование.

Аура (подсветка элементов) с цветовой кодировкой, используются для выделения объектов с определенными свойствами или чтобы показать различные виды взаимосвязей между объектами. Аура связана с позициями, переходами, дугами, надписями, описаниями, ярлыками страниц, и данными индекса, такими как имена страниц и имена сетей:



Аура бывает следующих цветов:

• Ярко-красный – указывает на объекты с ошибками при проверке синтаксиса и моделировании сети;

• Темно-красный – указывает на повторяемые имена позиций и переходов при проверке синтаксиса;

• Зеленый – указывает разрешенные переходы при моделировании сети;

• Темно-синий – указывает зависимость между описаниями и другими элементами, такими как позиции, переходы и страницы;

• Цвет морской волны – указывает объект, который содержит описание;

• Оранжевый – указывает на то, что проверка синтаксиса объекта еще не началась;

• Желтый – указывает на то, что проверка синтаксиса объекта выполняется в настоящее время;

• Розовый – указывает на то, какие объединенные позиции принадлежат множеству слияния;

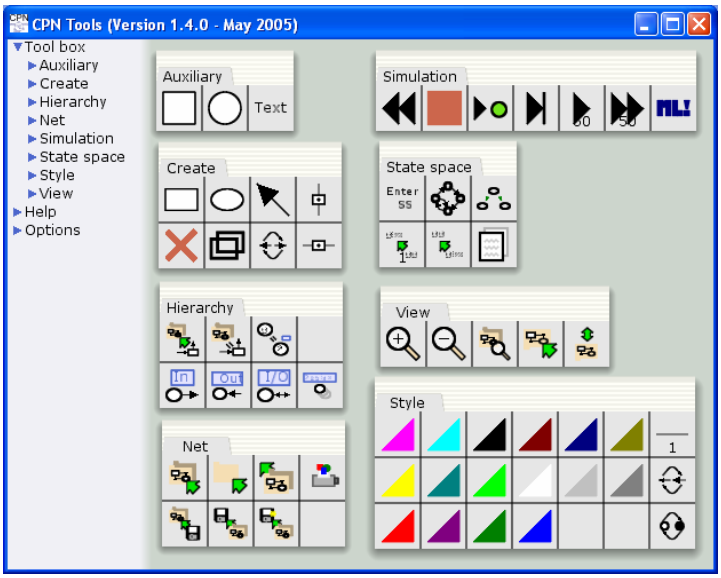
• Цвет морской волны указывает назначения порта/сокета и взаимосвязи страниц/подстраниц при работе с иерархическими сетями.

Форма курсора изменяется, чтобы показать, какие действия могут быть выполнены или выполняются. Например:

- Стандартный курсор в виде стрелки ;
- Курсор в виде руки указывает на то, что элемент может быть перемещен;
- Поперечно пересечённый вертикальная черта курсора указывает на то, что есть возможность редактировать текст;
- Курсор в виде двунаправленной стрелки показывает, что размер элемента может быть изменен. Вид стрелки показывает, в каком направлении размеры могут быть изменены: в горизонтальном, вертикальном, или в обоих одновременно.
- После того, как инструмент взят с одной из палитры, курсор изменится и примет форму выбранного инструмента;
- Для многоэтапных инструментов, то есть таких инструментов, которые применяются нажатием более, чем на один элемент, курсор покажет, какая функция инструмента далее будет применена. Примерами многоэтапных инструментов являются инструмент сокета/порта и инструмент создания подстраницы.

4. Инструменты CPN Tools

Инструменты представлены на следующих палитрах:



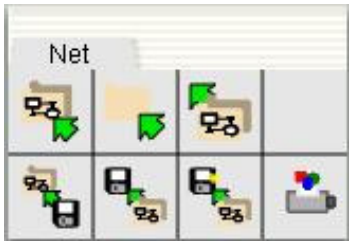
- Сетевые инструменты (Net): для операций с сетями;
- Инструменты для создания элементов сети (Create): для рисования и редактирования сетей Петри;
- Инструменты моделирования (Simulate): для имитации поведения сети;
- Инструменты пространства состояний (State Space): для построения и анализа пространства состояний;
- Инструменты иерархии (Hierarchy): для создания многоуровневых сетей;

• Инструменты стиля (Style): для указания особенностей внешнего вида сетей;

• Инструменты отображения (View): для выбора масштаба и указания групп элементов;

• Вспомогательные инструменты (Auxiliary): для повышения наглядности модели.

4.1. Сетевые инструменты



Пиктограммы на палитре обозначают следующее (слева направо и сверху вниз):

• создать новую сеть;

• создать новую страницу;

• закрыть сеть;

• загрузить сеть;

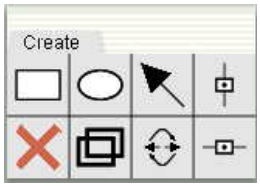
• сохранить сеть;

• сохранить сеть как;

• печатать сеть.

Чтобы создать новую сеть, следует начать с выбора инструмента «создать новую сеть» и завершить с помощью инструмента «сохранить сеть». Чтобы открыть существующую сеть, следует начать с выбора инструмента «загрузить сеть». Сети направляются на печать в файл в формате .eps (Encapsulated PostScript) и могут быть вставлены, например, как картинки в документы MS Word. Новые страницы в основном создаются для иерархических сетей.

4.2. Инструменты для создания элементов сети



Пиктограммы на палитре обозначают следующее:

• создать переход;

• создать позицию;

• создать дугу;

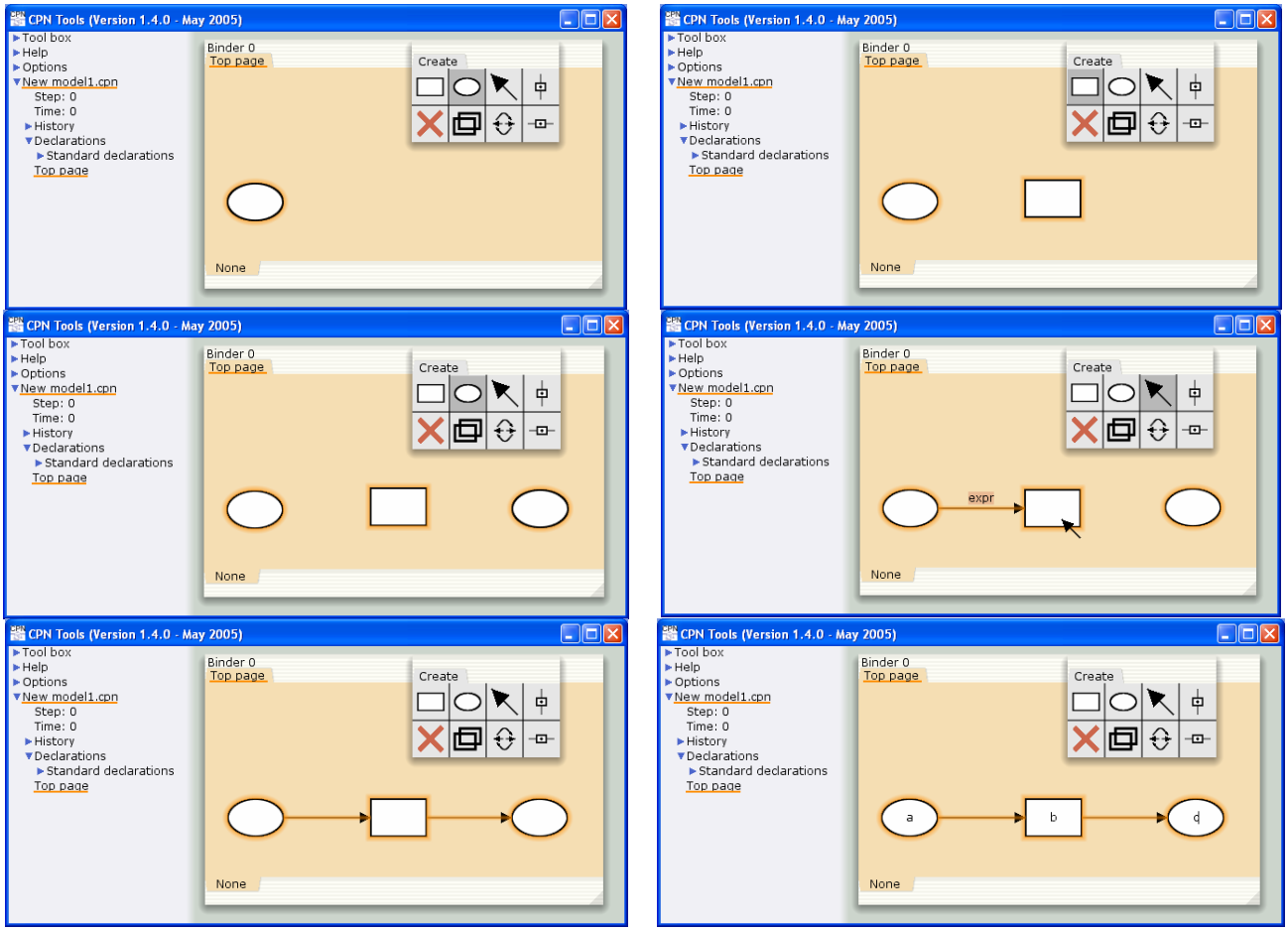
• создать вертикальную магнитную опорную линию;

• удалить элемент;

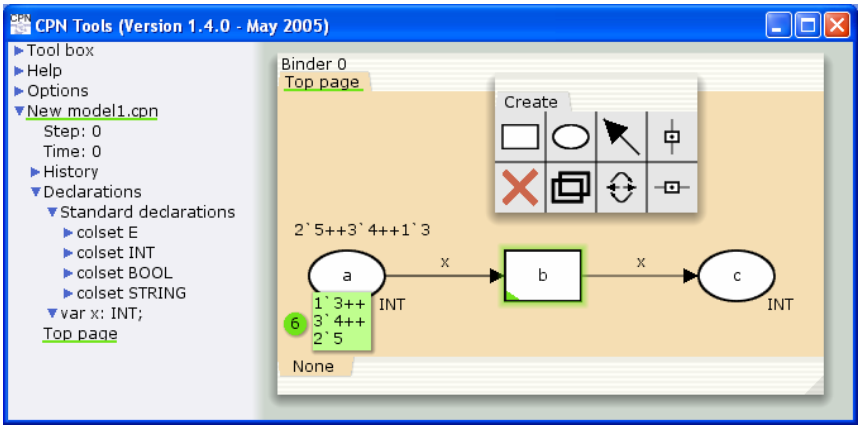
• клонировать элемент;

- переключение возможных направлений дуги;
- создать горизонтальную магнитную опорную линию.

Начнем с этой палитры для построения первой сети:

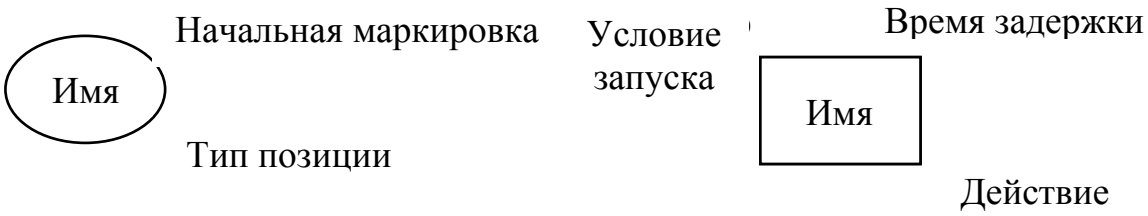


Здесь показана простая сеть Петри, но она некорректна, так как у элементов отсутствуют атрибуты. Чтобы данная сеть функционировала, будем использовать тип `INT` из стандартных описаний и добавим переменную `x` в описания:



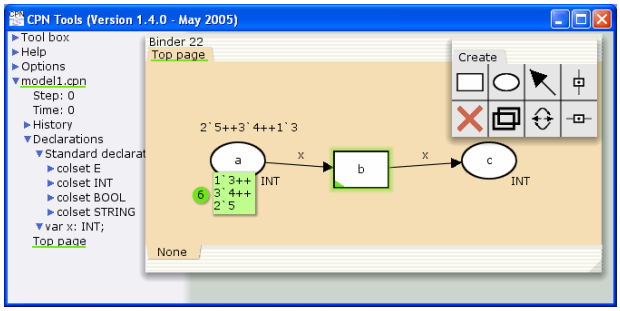
Теперь сеть корректна и будет функционировать. Отметим, что в позиции `a` содержится 6 фишек: 1 типа 3, 3 типа 4 и 2 типа 5. Покажем более детально способ ввода атрибутов элементов сети. Каждый узел имеет свое множество

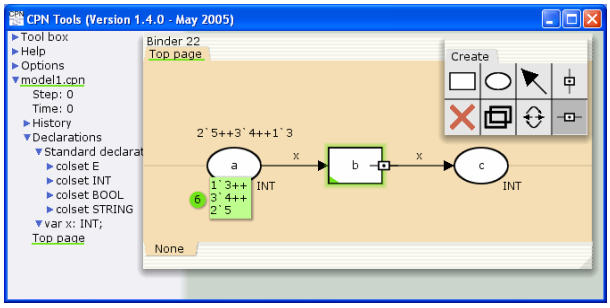
атрибутов. После указания элемента можно переключаться по его атрибутам, используя клавишу `Tab` на клавиатуре:



В вышеуказанном примере не используются атрибуты перехода. Текущая маркировка позиции автоматически надписывается CPN Tools зеленым цветом.

Магнитные опорные линии полезны для упорядоченного расположения элементов сети. Элементы автоматически перемещаются на самую близкую линию («примагничиваются»). Например:



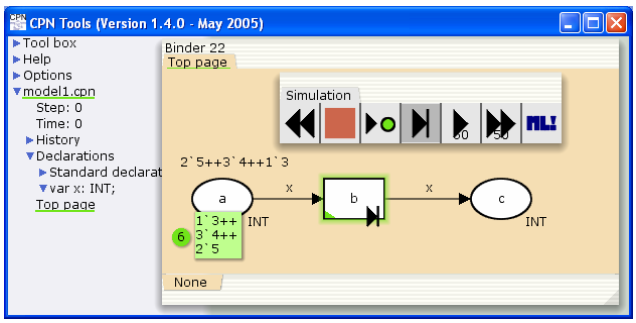


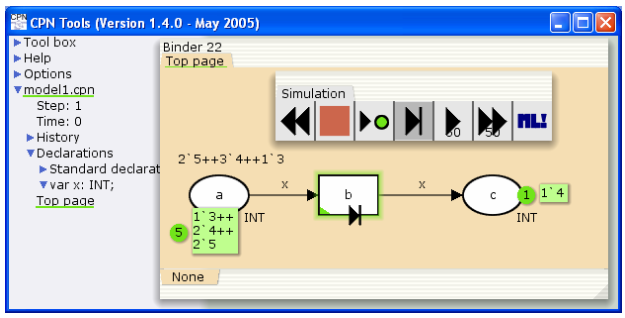
4.3. Инструменты для моделирования

Пиктограммы на палитре обозначают следующее:

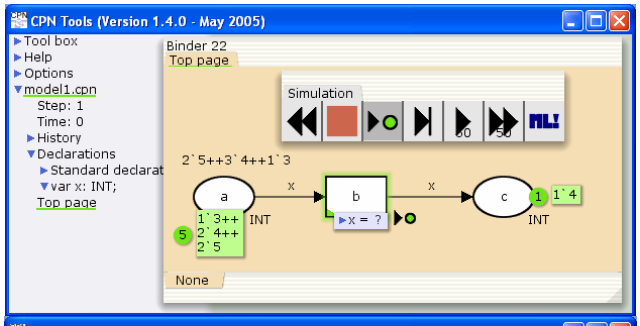
- вернуться в начальное состояние;
- остановить текущее моделирование;
- выполнить переход с выбранными параметрами фишки;
- выполнить переход;
- выполнить указанное количество переходов, показывая промежуточные маркировки;
- выполнить указанное количество переходов, не показывая промежуточные маркировки;
- оценивает текст как ML код.

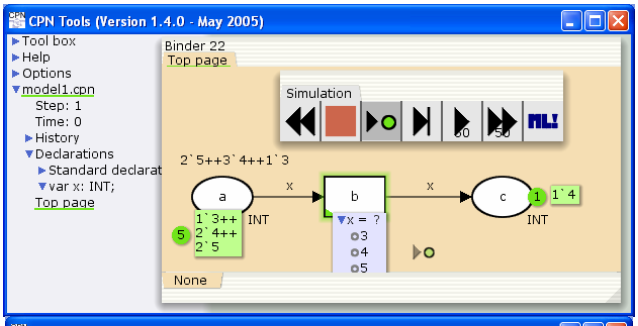
Инструменты, которые «выполняют переход», предназначены для отладки сетей путём пошагового моделирования. Чтобы запустить переход, следует кликнуть на нем либо на чистом фрагменте страницы (чтобы разрешить выбор перехода CPN Tools автоматически). Рассмотрим процесс моделирования:

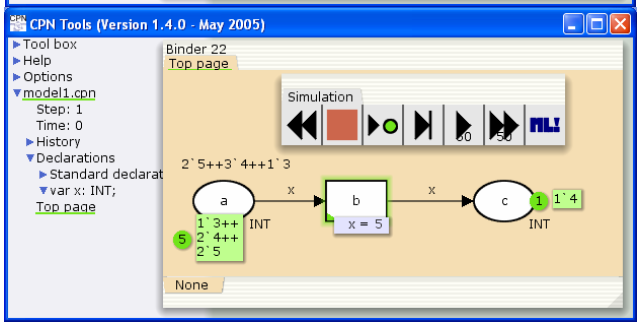


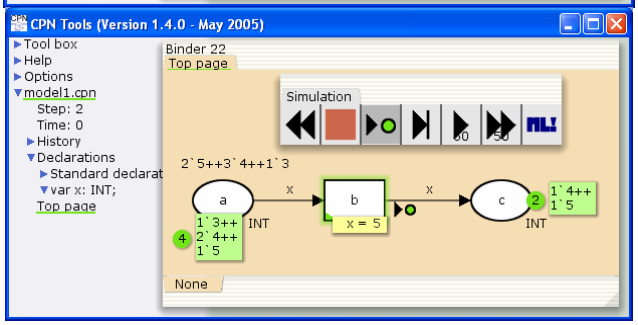


Фишка 4 была выбрана переменной `x` случайным способом из позиции `a` и перемещена переходом `b` в позицию `c`. Срабатывание перехода «с выбранными параметрами фишки» выполняется для детализированной отладки. В этом случае можно выбрать вручную фишку, как одну из тех, что удовлетворяют надписи входной дуги перехода:



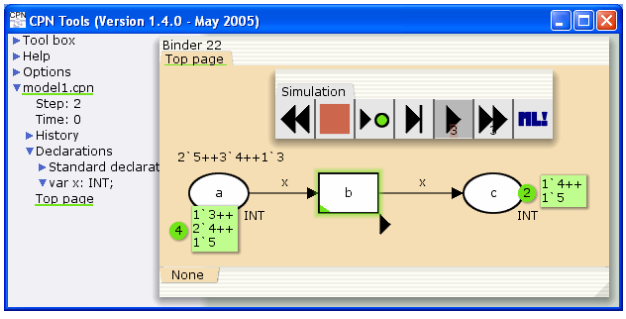


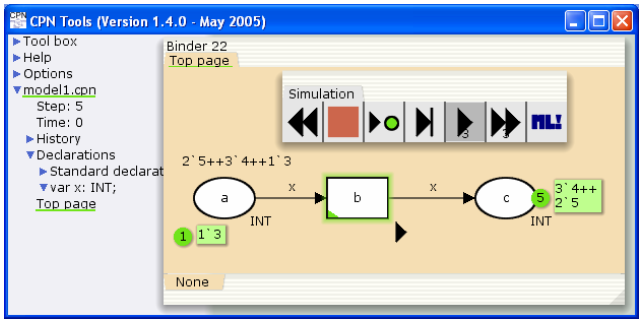




В этом примере был предложен выбор между фишками 3, 4, 5 и фишка 5 была выбрана вручную.

Что касается «срабатывания указанного количества переходов», они выбираются случайным образом CPN Tools. Можно ввести требуемое количество переходов:

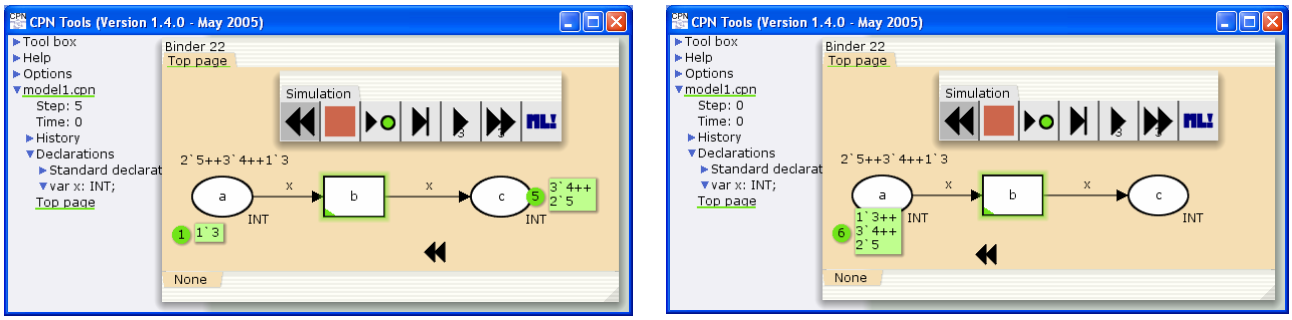




Количество переходов (три) было введено в вышерассмотренном примере. Единственное различие между двумя режимами – это то, что CPN Tools останавливается и показывает промежуточную маркировку в первом случае и

показывает только конечную маркировку во втором случае. Результат тот же самый, но режим, не показывающий промежуточные маркировки, работает гораздо быстрее. Он используется для моделирования на больших интервалах времени для сбора статистической информации.

Инструмент «остановить текущее моделирование» позволяет прервать процесс моделирования, когда что-то функционирует не так как надо, или процесс длится слишком долго. Инструмент «вернуться в начальное состояние», позволяет вернуться к исходному состоянию:



Оценка текста как ML кода требуется для принудительной проверки синтаксиса в языковых конструкциях.

4.4. Краткий обзор других инструментов

Другие инструменты CPN Tools являются либо вспомогательными, такие как `Вспомогательные (Auxiliary)`, `Стиль (Style)`, и `Отображение (View)`, либо более сложными, такие как `Иерархия (Hierarchy)`, `Пространство состояний (State Space)`. Подробное описание инструментов иерархии и пространства состояний будет приведено в последующих разделах.

`Вспомогательные` инструменты позволяют создавать прямоугольники, овалы и текстовые метки, у которых нет никакого семантического значения, но которые могут облегчить восприятие сети.

Инструменты `Стиля` используются для того, чтобы различными цветами, толщиной линий, закраской элементов, размером дуги (стрелки) подчеркнуть особенности компоновки модели для повышения наглядности и удобства восприятия сети. Ни один из этих инструментов не влияет на семантику сети.

Инструменты `Отображения` используются для изменения вида страницы и ее элементов, с помощью группирования элементов и изменения масштаба изображения.

Инструменты `Иерархии` используются для редактирования иерархической структуры сети. Палитра содержит инструменты, как для восходящего, так и для нисходящего структурирования сети.

Инструменты `Пространства состояний` используются для построения пространства состояний сети, для передачи состояний между моделирующей системой и пространством состояний, а также, для того, чтобы сформировать отчёт по пространству состояний.

5. Основы языка CPN ML

CPN Tools использует язык CPN ML для создания описаний в индексе и надписей элементов сетей (атрибутов). CPN ML обеспечивает описание множеств цветов (типов данных), переменных, функций, величин (констант). У каждой позиции раскрашенной сети Петри должен быть обязательный атрибут – определенное множество цветов; позиция может содержать фишки только указанного множества цветов. Переменные и функции используются для указания атрибутов переходов и дуг.

Описания расположены в индексе, как часть сети. Существуют стандартные предопределенные описания таких множеств цветов как: `E` – элементарный тип, `INT` – целые числа, `BOOL` – логический тип, `STRING` – строковый тип. Описания пользователя могут быть добавлены после стандартных описаний, с помощью контекстно-зависимых меню. Кроме того, для сложных сетей CPN Tools обеспечивает внешние описания, которые могут быть загружены из файла.

CPN Tools автоматически проверяет синтаксис сети при ее создании или загрузке. По цветным индикаторам (аурам) можно увидеть, насколько выполнилась проверка. Цветные индикаторы отображаются в индексе, подсвечивая имя описания или страницы, которые проверяются. Если страница открыта в графическом редакторе, аура также показана наверху в ярлыке страницы и на элементах сети. Оранжевое свечение указывает на то, что элемент в настоящее время не проверен. При загрузке сети проверка синтаксиса может занимать несколько минут. На этом этапе элементы изменяют свечение от оранжевого через желтое к отсутствию свечения (или красному, если обнаружена ошибка). Если оранжевое свечение остается, то это, вероятно, потому что что-то пропущено в описании или соответствующий элемент сети содержит несерьёзную ошибку.

Описания проверяются, начиная сверху. Если описание зависит от другого описания, представленного ниже, то выводится ошибка, указывающая, что описание не определено. Описания с ошибками повторно проверяются, когда сделано изменение в любом из них. Если в описании есть ошибка, то оно будет подсвечено красным цветом. Имя сети и все задействованные страницы также будут подсвечены красным цветом.

Красное свечение означает, что элемент проверен, но содержит ошибку. Всплывающее сообщение при указании элемента курсором поясняет причину ошибки. Другие элементы, связанные с элементом, содержащим ошибку, не проверяются, пока ошибка не будет устранена.

5.1. Простые множества цветов

CPN ML обеспечивает следующие простые множества цветов: `unit`, `boolean`, `integer`, `string`, `enumerated`, `index`.

Множество цветов **unit** содержит единственный элемент. Синтаксис описания выглядит следующим образом:

```
colset name = unit [with new_unit];
```

По умолчанию пустое имя фишки представляет собой пару круглых скобок (). В примере о Золушке использовались следующие переменные типа unit:

```
colset p=unit with pumpkin;
colset c=unit with Cinderella;
colset m=unit with mouse;
colset f=unit with Fairy;
```

Величины типа **boolean** (логический тип) бывают двух видов `true (правда)` и `false (ложь)`. Синтаксис описания выглядит следующим образом:

```
colset name = bool [with (new_false, new_true)];
```

Опции позволяют переименовать названия `true` и `false`, например, в `yes (да)` и `no (нет)`:

```
colset Answer = bool with (no, yes);
```

К логическим переменным могут быть применены следующие операции:

not b	отрицание логической переменной b, «нет»
b1 **andalso** b2	логическое умножение (конъюнкция), «и»
b1 **orelse** b2	логическое сложение (дизъюнкция), «или»

Integers (целые числа) – это числа без десятичной точки. Синтаксис описания выглядит следующим образом:

```
colset name = int [with int-exp1...int-exp2];
```

Опции позволяют ограничить множество цветов интервалом, заданным двумя выражениями: от `int-exp1` до `int-exp2`:

```
colset Dozen = int with 1..12;
```

К целым переменным могут быть применены следующие операции:
`+, -, div, mod, abs, Int.min, Int.max, ~` – унарный минус.

Strings (строковый тип) определяет последовательности символов ASCII (Американский стандартный код обмена информацией) в двойных кавычках. Синтаксис описания выглядит следующим образом:

```
colset name = string [with string-exp1..string-exp2
[and int-exp1..int-exp2]];
```

Опции определяют диапазоны допустимых символов:

```
colset LowerString = with "a".."z";
```

К строковым переменным могут быть применены следующие операции: ^ – конкатенация (операция последовательного объединения двух строк в одну), `String.size` – размер, `substring` – подстрока.

Enumerated (перечислимые) значения явно перечисляются как идентификаторы в описании. Синтаксис описания выглядит следующим образом:

```
colset name = with id0 | id1 | ... | idn;
```

В примере о Золушке использовалось следующее перечислимое множество цветов:

```
colset g=with rice | wheat | oat;
```

Indexed (индексированные переменные) – последовательности значений, состоящих из идентификатора и спецификатора индекса:

```
colset name = index id with int-exp1..int-exp2;
```

Индексированные значения имеют вид: `id i` или `id (i)`, где `i` – целое число и `int-exp1 <= i <= int-exp2`. Например, в задаче об обедающих философах можно описать философов и вилки следующим образом:

```
colset PH = index ph with 1..5;
colset FR = index fork with 1..5;
```

При этом философ `ph(1)` берет вилки: `fork(1)` и `fork(2)`.

5.2. Составные множества цветов

Составные множества цветов состоят из комбинаций простых множеств цветов. CPN ML обеспечивает следующие составные множества цветов: `product`, `record`, `union`, `list`, `subset` и `alias`. Так как `list` и `union` редко используются и являются более сложными, они будут рассмотрены в заключительном разделе.

`Product` и `record` представляют собой кортежи данных, сформированные как результат декартового произведения множеств цветов. Единственное различие между ними состоит в том, что компоненты множества цветов `product` являются безымянными, в то время как компоненты множества цветов `record` поименованы. Существует близкое сходство с типом данных record языка программирования Паскаль или со структурами языка С.

Синтаксис описания множества цветов **product** имеет вид:

colset name = **product** name1 * name2 * ... * namen;

Значения этого множества цветов имеют вид:

(v1, v2, ..., vn) где vi имеет тип namei для 1<=i<=n.

Чтобы извлечь i–й элемент из кортежа, используется следующая операция:

#i name

Синтаксис описание множества цветов **record** имеет вид:

colset name = **record** id1:name1 * id2:name2 * ... * idn:namen;

Значения этого множества цветов имеют вид:

{id1=v1, id2=v2, ..., idn=vn} где vi – это переменные типа namei для 1<=i<=n.

Чтобы извлечь i–й элемент из кортежа, используется следующая операция:

#idi name

Рассмотрим вышеуказанные множества цветов на примере описания кадра Ethernet. Кадр Ethernet состоит из адреса отправителя, адреса получателя и данных. Представим MAC-адреса множеством цветов типа integer и данные –множеством цветов типа string:

```
colset MAC = int;
colset DATA = string;
colset frame = product MAC * MAC * DATA;
colset frame1 = record src : MAC * dst : MAC * d : DATA;
```

Фреймы Ethernet могут быть представлены множеством цветов frame, либо множеством цветов frame1. Для множества цветов frame переменная x = (2, 4, «Hello»), например, описывает кадр, отправленный устройством 2 на устройство 4, содержащий приветствие «Hello». Этот же кадр для множества цветов frame1 имеет вид x1 = { dst=4, src=2, d = «Hello» }.

Чтобы извлечь адрес получателя, в множестве цветов frame, пишут:

#2 x

и в множестве цветов frame1:

#dst x1

Множество цветов **alias** имеет те же самые значения и свойства, как и ранее описанное множество цветов. Оно введено, для того чтобы использовать различные названия одного и того же множества цветов. Синтаксис описания имеет вид:

```
colset name = name0;
```

5.3. Описание переменных и констант

Переменная – это идентификатор, значение которого может быть изменено во время выполнения модели. Переменные используются в атрибутах элементов сети Петри.

Синтаксис описания **переменных** имеет вид:

```
var id1, id2, ..., idn : cs_name;
```

где `idi` – идентификаторы, `cs_name` – имя предварительно описанного множества цветов. Например:

```
var f1, f2 : frame;
var f3, f4 : frame1;
```

Описание **величины** сопоставляет некоторое значение указанному идентификатору (задаёт константу). Синтаксис описания величины имеет вид:

```
val id = exp;
```

где `id` – идентификатор и `exp` выражение CPN ML. Выражение задаёт значение, которое будет связано с идентификатором. Например:

```
val CheckFrame = (3, 5, "Ping" );
val ResponseFrame1 = {src=5, dst=3, d="OK"};
```

5.4. Функции

Функции CPN ML используют стандартные управляющие структуры языков программирования, такие как операторы «if» и «case». Но так как ML представляет собой, по существу, язык функционального программирования, то наибольшая изобразительная мощность достигается при создании рекурсивных функций.

Синтаксис описания **функции** имеет вид:

```
fun id pat1 = exp1
  | id pat2 = exp2
  | ...
  | id patn = expn;
```

где `pat1, pat2, …, patn` – шаблоны и выражения `exp1, exp2, ..., expn` имеют один и тот же тип. Описание означает, что в случае, когда фактические аргументы удовлетворяют шаблону `pati`, значение функции

будет вычислено как expi. Например, следующая функция вычисляет факториал целого числа, используя рекурсию:

```
fun fact ( 0 ) = 1
    | fact ( i ) = i * fact( i-1);
```

Управляющие структуры **if-then-else** и **case** также доступны для описания функций:

```
if bool-exp then exp1 else exp2;
```

где exp1 и exp2 имеют тот же самый тип.

```
case exp of
     pat1 => exp1
   | pat2 => exp2
   | ...
   | patn => expn;
```

где exp1, exp2, ..., expn имеют тот же самый тип. Их семантика стандартная, как в других языках программирования. Например, функция, которая вычисляет знак числа, может быть описана:

```
fun sign (x) = if x>0 then 1 else if x<0 then ~1 else 0;
```

Функция, которая возвращает название знака числа, может быть описана:

```
fun nname (x) =
          case sign(x) of
             1 => "positive"
           | ~1 => "negative"
           |  _ => "zero";
```

Подчеркивание _ в последней строке указывает на то, что строковая переменная «zero» будет выбрана для всех других значений проверяемого выражения ((sign (x)).

Структура **let** задаёт описания локальных переменных в функциях:

```
let
   val pat1 = exp1
   val pat2 = exp2
   ……
   val patn = expn
in
   exp
end;
```

Пример: вычисление размера в метрах по заданному значению в миллиметрах:

```
fun metr (x) =
  let
     val mminm=1000;
  in
     x div mminm
  end;
```

5.5. Случайные функции

Случайные функции обеспечивают возможность моделирования статистических характеристик. Например, они позволяют описывать интенсивность трафика или вероятность ошибок в телекоммуникационных системах. Существуют несколько способов описания случайного выбора в CPN Tools:

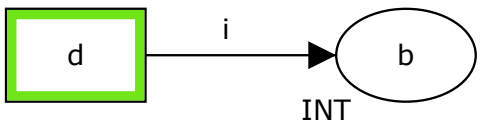
– свободные (несвязанные) переменные;

– функция `ran`;

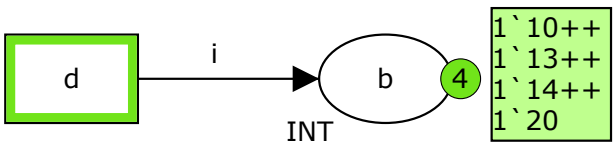
– специальные случайные функции с указанным законом распределения.

Свободные (несвязанные) переменные являются переменными выходных дуг переходов, значения которых не определены входными дугами и другими атрибутами. Им присваиваются случайные значения во время выполнения сети. Тип свободных переменных должен быть малым множеством цветов. Множества цветов могут быть классифицированы как большие или малые. Это различие определяет, какие предопределенные функции могут применяться для указанного множества цветов. Множество цветов является большим, если оно содержит много (по умолчанию более 100) пронумерованных элементов, иначе множество является малым. Множества цветов `unit, Boolean, index и enumerated` являются малыми.

В следующем примере переменная `i` – свободная переменная в диапазоне 1.. 31. На втором рисунке показано состояние модели после четырех шагов; четыре значения переменной `i` были выбраны CPN Tools случайным образом:

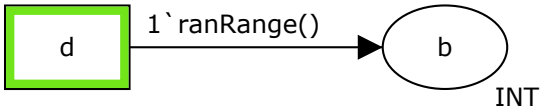


colset mday = int with 1..31;
var i : mday;

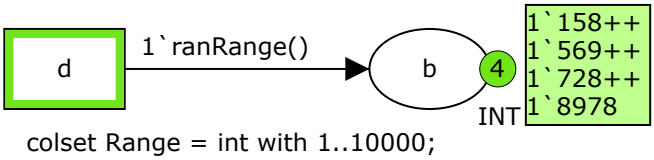


colset mday = int with 1..31;
var i : mday;

Функция `ran` генерирует случайную величину для больших множеств цветов. В следующем примере функция `ranRange()` генерирует случайную величину в диапазоне `1..10000`:



colset Range = int with 1..10000;
fun ranRange () = Range.ran();



colset Range = int with 1..10000;
fun ranRange () = Range.ran();

Система CPN Tools предоставляет также ряд специальных **случайных функций** с различными законами распределения случайных величин: Бернулли, биномиальное, Эрланга, показательное, нормальное, Пуассоновское, Стьюдента, равномерное (дискретное и непрерывное). Например, функция

```
erlang(n:int, r:real) : real
```

где `n>=1` и `r>0.0`, возвращает выборку из n–го распределения Эрланга с интенсивностью `r`.

5.6. Мультимножества

Мультимножества широко используются в CPN Tools для представления маркировки позиций и в других целях. Рассмотрим концепцию мультимножества. В отличие от обычных множеств, мультимножества содержат элементы с определенной кратностью, другими словами в определенном числе копий. Мультимножества также называются «портфелями».

Символ обратной кавычки (`) является конструктором мультимножества. Например, `3`5` – это мультимножество, состоящее из трех элементов цвета (типа) 5. Синтаксис описания мультимножества имеет вид:

```
i`c,
```

где целое `i` – неотрицательное число, в противном случае будет возвращено пустое мультимножество. Конструктор мультимножества, дополненный операторами сложения (++) и вычитания (--) мультимножеств, обеспечивает лаконичный способ задания мультимножеств. В примере модели Золушка, описанном в разделе 1, в позиции «`sack of mixture`» указана начальная маркировка:

```
1000`rice ++ 2000`wheat ++ 3000`oat
```

Это означает, что мешок содержит 1000 зерен риса, 2000 зерна пшеницы и 3000 зерен овса. Следует обратить внимание на знак конструктора мультимножества: это – обратная кавычка (`), а не апостроф ('). Для примера фрейма Ethernet с множеством цветов `frame1` содержание буфера может быть представлено как:

```
1`{src=2, dst=5, d="request"} ++ 1`{src=5, dst=2, d="answer"}
```

Указаны два кадра: первый с данными `"request"`, направленный на устройство 5 от устройства 2, и второй с данными `"answer"`, направленный на устройство 2 от устройства 5.

Для работы с мультимножествами используются следующие константы, операции и функции:

empty	константа empty создает пустое мультимножество, которое сопоставимо с любыми типами мультимножеств
ms1 **==** ms2	равенство мультимножеств
ms1 **<><>** ms2	неравенство мультимножеств
ms1 **>>** ms2	мультимножество больше, чем
ms1 **>>==** ms2	мультимножество больше или равно
ms1 **<<** ms2	мультимножество меньше, чем
ms1 **<<==** ms2	мультимножество меньше или равно
ms1 **++** ms2	сложение мультимножеств
ms1 **--** ms2	вычитание мультимножеств (ms2 должно быть меньше или равно ms1), генерирует прерывание, если ms2 не меньше или равно ms1.
i ****** ms	скалярное произведение
size ms	размер мультимножества
random ms	возвращает псевдослучайный цвет из мультимножества
cf(c,ms)	возвращает количество появлений цвета в мультимножестве
filter p ms	берет предикат p и мультимножество ms и генерирует мультимножество всех элементов ms, удовлетворяющих предикату p

Например,

```
m1 = 2`5  ++ 3`4 ++ 4`5;
m2 = 1`5 ++ 2`4 ++ 3`5;
```

тогда

```
m1 ++ m2 = 3`5 ++ 5`4 ++ 7`5
m1 – m2 = 1`5 ++ 1`4 ++ 1`3
m1 >> m2   имеет значение true
size m1 = 9
cf(4,m1) = 3
```

В CPN Tools начальная и текущая маркировки позиции представлены мультимножеством, заданным на множестве цветов позиции. При выборе фишки в переменную надписи дуги CPN Tools обеспечивает случайный выбор фишки, как с помощью функции random.

5.7. Временные мультимножества

Временные мультимножества используются в CPN Tools, чтобы представить временные задержки в модели. Описание соответствующего множества цветов должно быть дополнено модификатором timed. Операторы @, @+, и @@ + используются, чтобы добавить временные метки к цветам. Добавление временной задержки x к цвету c присоединяет временную метку с величиной, равной текущему времени модели + x, к цвету c. Следующие операции применимы к временным мультимножествам:

c **@** t	присоединение временной метки (с типом Time.time) к цвету c
ms **@+** i	добавление целой временной задержки i к каждому из цветов мультимножества ms, возвращает временное мультимножество
tms1 **+++** tms2	сложение временных мультимножеств

Например, описание

```
colset tint = int timed;
var t : tint;
t = 1`2@100 ++ 1`3@200 ++ 1`4@300;
```

означает, что фишка `2@100` может быть использована CPN Tools только когда модельное время больше или равно 100, фишка `3@200` – только когда модельное время больше или равно 200 и т.д. До наступления момента активации фишка не может быть извлечена ни одним переходом модели.

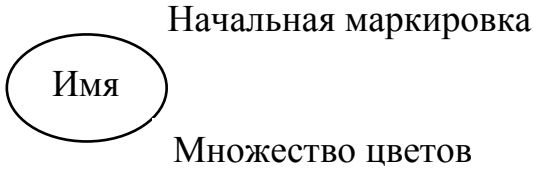
6. Язык описания моделей

В CPN Tools каждый элемент сети Петри имеет свои атрибуты, описанные на языке CPN ML. Используя `инструменты для создания элементов сети`, можно поместить элемент на страницу модели. Затем добавляются атрибуты элементов. Для этого необходимо щелкнуть мышью на соответствующем элементе и использовать кнопку `Tab` на клавиатуре для переключения атрибутов. Нажатие на клавиатуре клавиши `Esc` позволяет покинуть выбранный элемент; такой же результат может быть получен щелчком мыши в любой другой области модели. Рассмотрим атрибуты каждого элемента сети отдельно.

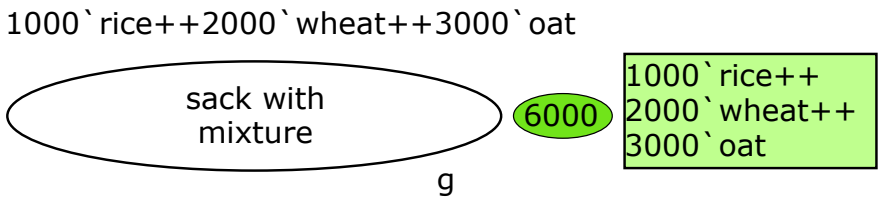
6.1. Атрибуты позиций

Существует три типа атрибутов, которые могут быть связаны с позицией. Два из них – необязательные:

- Цвет фишек – обязательный атрибут;
- Начальная маркировка – необязательный атрибут;
- Имя – необязательный атрибут.



По начальной маркировке CPN Tools автоматически создает текущую маркировку, которая отображается зеленым цветом и показывает общее число фишек и детали маркировки. Например, в примере о Золушке:

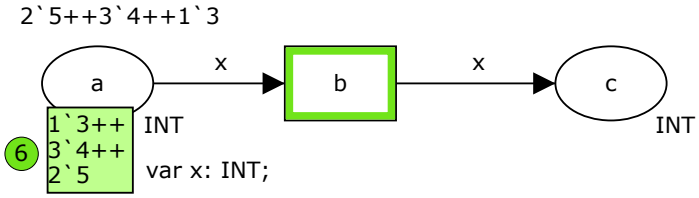


6.2. Атрибуты дуг

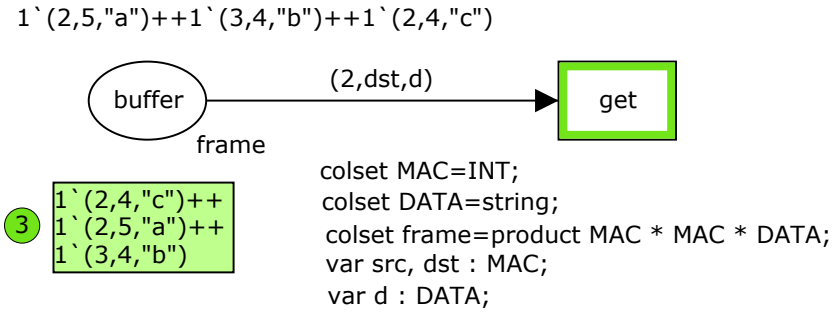
Атрибут имеет вид:

Множество цветов выражения на дуге `expr` должно соответствовать множеству цветов позиции, связанной с дугой. Если множество цветов выражения на дуге не соответствует множеству цветов позиции, связанной с дугой, то около дуги во время проверки синтаксиса появится сообщение об ошибке. Между атрибутами входных и выходных дуг перехода есть существенное различие.

Выражение входной дуги перехода является шаблоном для выбора фишки. Этот шаблон описывается как предикат, который может быть применен в функции `filter`, для соответствующего множества цветов. В самом простом случае, предикат состоит из одной переменной соответствующего множества цветов, как в вышеуказанных примерах. В следующей сети

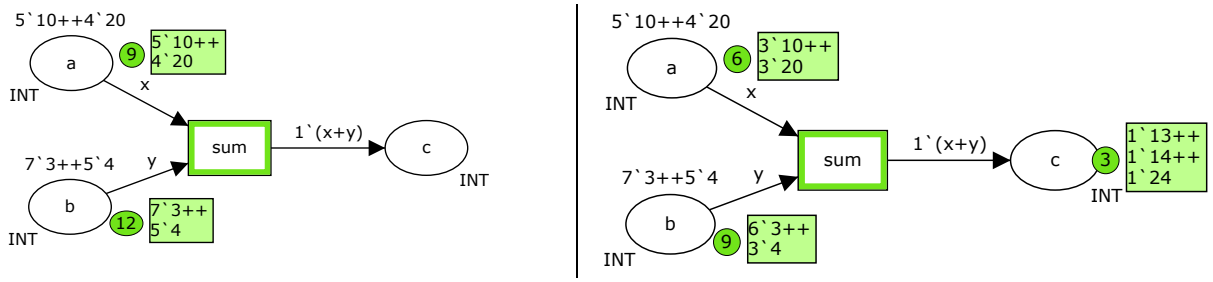


любая из 6 фишек может быть выбрана переменной `x`. Более сложный случай представляет собой выбор кадров с адресом отправителя, равным двум для кадров сети Ethernet:

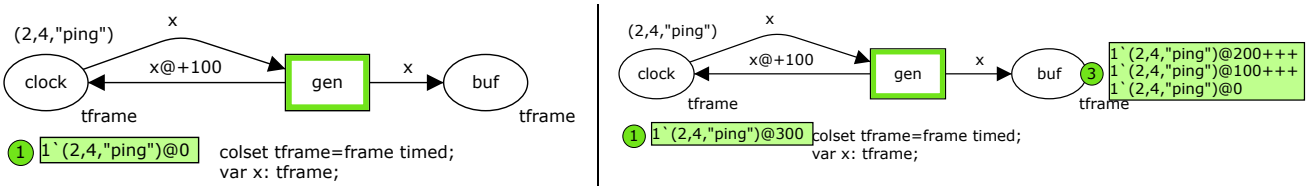


Любой из кадров `(2,5,"a")` и `(2,4,"c")` может быть выбран выражением `(2,dst,d)`.

Выражение выходной дуги перехода представляет собой конструктор для создания новых фишек. Такой конструктор часто использует переменные надписей входных дуг и в простых случаях может совпадать с одной из них. В следующем примере переход `sum` вычисляет сумму входных фишек:



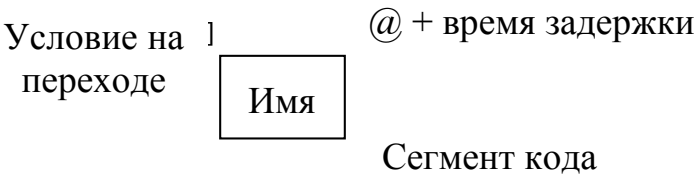
Кроме того, временные задержки могут быть применены к выходным фишкам. В следующем примере обеспечивается формирование (генерация) кадра через каждые 100 единиц модельного времени:



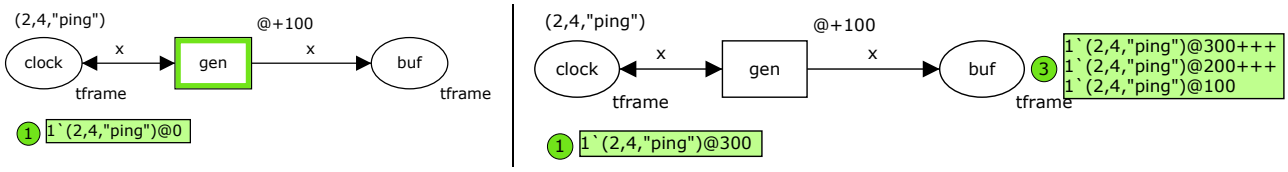
6.3. Атрибуты переходов

Существует четыре типа атрибутов, которые могут быть связаны с переходом. Все они являются необязательными:

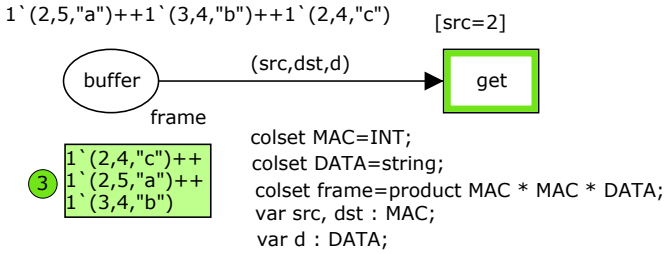
- Имя перехода;
- Условие запуска перехода;
- Время задержки;
- Сегмент кода.



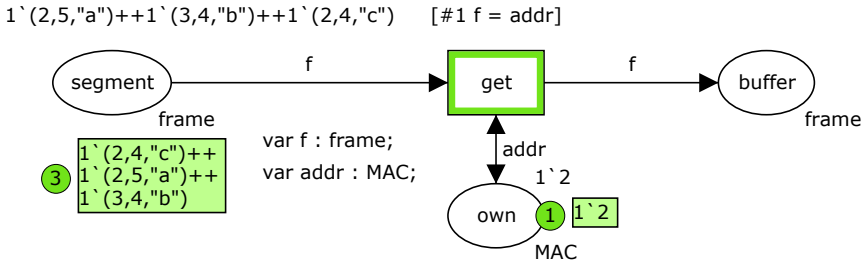
Время задержки перехода должно быть положительным целым числом. Выражение, следующее за @+, означает, что временной атрибут имеет форму `@+delay-expr`. Прежде, чем временная задержка будет добавлена, по умолчанию вводится текст надписи – @+. Задержка времени всегда добавляется относительно текущего времени. Например, если текущее время – `10` и задержка времени – @+2, тогда временная метка фишки, направленной в выходную позицию, будет равной `12`. Отсутствие атрибута времени эквивалентно нулевой задержке. В отличие от задержки дуг, задержки переходов применяются ко всем выходным фишкам перехода. В случае одной выходной дуги два способа эквивалентны. Например, сравним следующую сеть с примером в разделе 6.2:



Условие запуска перехода является логическим выражением языка CPN ML, которое оценивается как `true` (правда) или `false` (ложь). Перед тем как условие будет добавлено, выводится текст надписи по умолчанию: `[]`. Условие может быть простым логическим выражением или списком логических выражений `[b-exp1, b-exp2,..., b-expn]`. Переход срабатывает только в том случае, если условие является истинным, что также ограничивает выбор входных фишек. Например, проверка входных фреймов, представленных в разделе 6.2, может быть описана следующим образом:



Кроме того, условие допускает сравнение параметров фишек из различных позиций, используя их комбинацию в выражениях. Следующий фрагмент моделирует процесс извлечения кадров из сегмента Ethernet:



Адрес рабочей станции храниться в позиции `own`. Для его проверки используется двунаправленная дуга.

Любой переход может содержать присоединённый **сегмент кода**, который представляет собой процедуру языка ML. Сегмент кода выполняется, когда срабатывает его родительский переход. Сегмент кода может использовать переменные CPN и может связывать переменные CPN, расположенные в надписях входных дуг, которые не связаны в других атрибутах. Каждый сегмент кода может содержать:

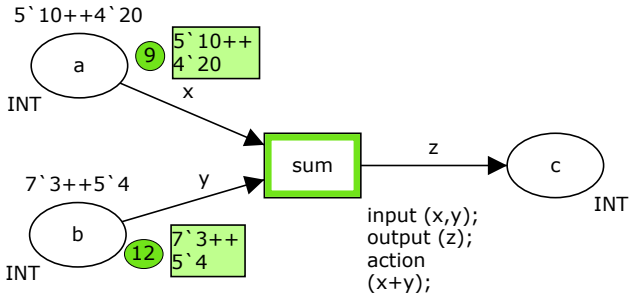
- Входной шаблон (необязательный);
- Выходной шаблон (необязательный);
- Код операций (обязательный).

Входной шаблон является кортежем переменных CPN, следующих за ключевым словом `input`. Он перечисляет переменные CPN, которые могут быть использованы в коде операций. Код операций может использовать значения этих переменных CPN, но не может изменять их. Переменные CPN, перечисленные во входном шаблоне, могут быть использованы в коде операций, если их объявили как идентификатор ML с таким же именем в области описания переменных. Если входной шаблон опущен, то подразумевается, что никакие переменные CPN не могут быть использованы в коде операций.

Выходной шаблон является кортежем переменных CPN, следующих за ключевым словом `output`. Он перечисляет переменные CPN, которые изменяются в результате выполнения кода операций. Выходной шаблон должен быть переменной CPN или кортежем переменных CPN без повторений. Если выходной шаблон опущен, то подразумевается, что никакие переменные CPN не вычисляются.

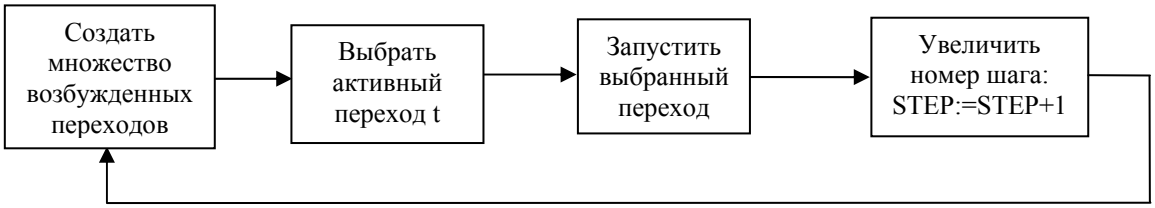
Код операций является выражением языка ML, следующим за ключевым словом `action`. Он не может содержать описание множеств цветов, переменных CPN или ссылочных переменных. Тем не менее, к кодовому выражению можно применить объявленные пользователем константы, операции, и функции. Кроме того, новые функции и константы могут быть определены для локального использования посредством `let-in-end`. Кодовое выражение выполняется как локальное описание в среде, содержащей переменные CPN, определенные во входном шаблоне. Это гарантирует, что кодовое выражение не может изменить непосредственно любые переменные CPN, а может изменить только их локальные копии. Когда кодовое выражение выполнено, полученный результат применяется к CPN переменным в выходном шаблоне. Кодовое выражение при оценке в среде, содержащей входные переменные, должно дать результаты такого же типа как выходные переменные. Если не представлено никакой выходного шаблона, то подразумевается тип результата `unit`.

Сегменты кода используются для более сложной обработки входных фишек. Например, суммирование фишек, описанное в разделе 6.2, можно представить, используя кодовый сегмент:



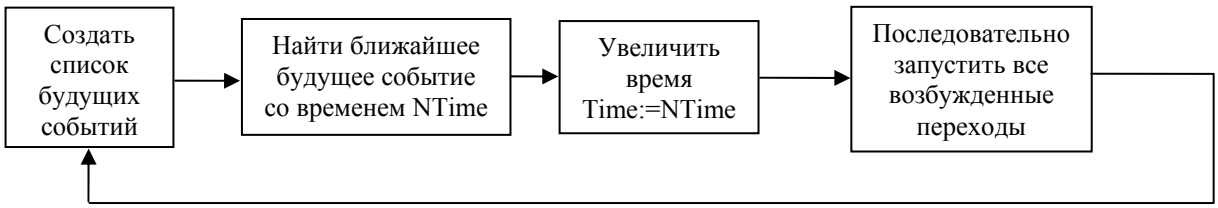
7. Особенности временных сетей CPN Tools

CPN Tools обеспечивает моделирование сетей Петри, как временных, так и невременных. Если ни одно из множеств цветов модели не имеет временной модификатор `timed`, тогда сеть считается невременной `untimed`. В этом случае CPN Tools использует только внутреннюю переменную `Step`, которая обозначает количество выполненных переходов. Алгоритм имитационного моделирования CPN Tools можно представить следующим образом:

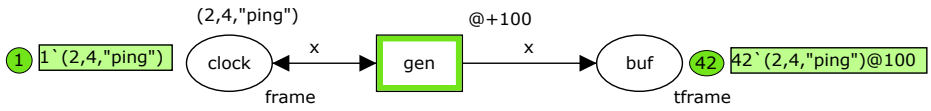


Следует обратить внимание на то, что выбор перехода может быть осуществлен вручную в пошаговом режиме моделирования, либо автоматически случайном образом при выполнении указанного количества шагов.

Для временных сетей Петри, алгоритм более сложный, так как отслеживается путь продвижения времени:

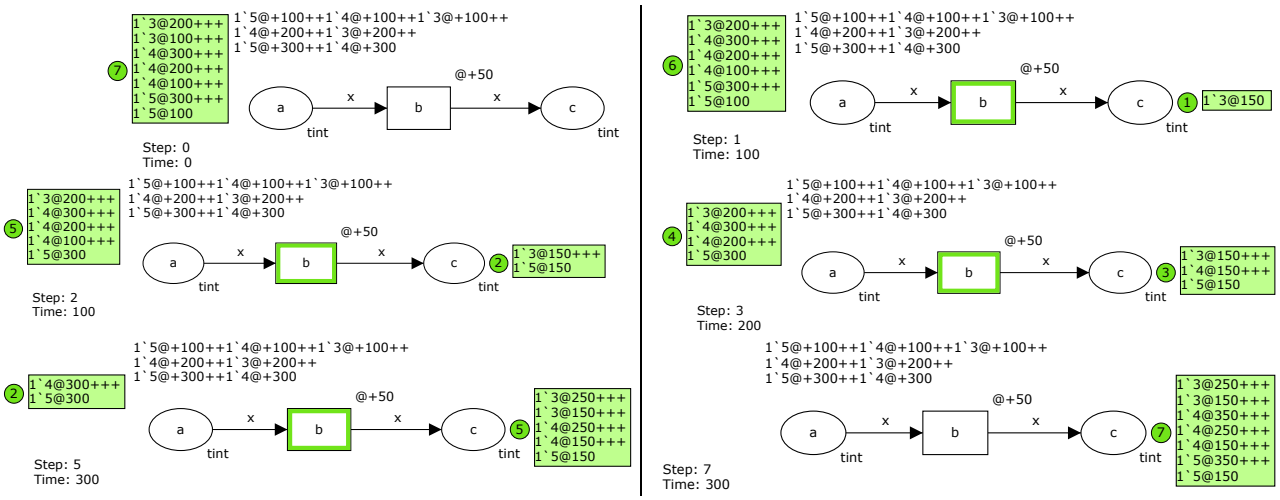


Следующий момент модельного времени не будет являться моментом времени `Time+1`. Система двигает время до ближайшего будущего события `Time:=time-of-nearest-future-event` и затем выполняет все события, которые могут произойти в этот момент времени, увеличивая переменную `Step`, описанную выше. Поэтому необходимо быть осторожным при объединении временных и невременных сетей, так как разрешенные переходы срабатывают до тех пор, пока множество разрешенных переходов не станет пустым. Это может вызвать зацикливание в поведении моделей. Например, в следующей сети время не продвигается и количество сгенерированных кадров постоянно увеличивается:



Следует отметить, что CPN Tools поддерживает простой и очень мощный класс временных сетей Петри, соответствующий использованию временных меток (штампов). Каждая фишка имеет свою временную метку `t` (k@t). Для моментов модельного времени `Time<t` фишка не обрабатывается системой моделирования; она находится в так называемом недействительном состоянии

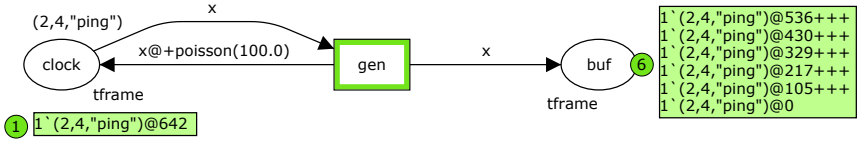
и не принимает участие в срабатывании переходов. После момента времени t фишка k «просыпается» и принимает участие в срабатывании переходов. Такой способ представления времён разрешает мгновенное срабатывание переходов. Выходные фишки, имеющие временную задержку (k@t+d), ожидают своего времени активации в соответствующих выходных позициях. Следующий пример трассировки иллюстрирует способ обработки времён в CPN Tools:



В момент времени 0 ни один переход не разрешен, так как ни у одной фишки нет временной метки 0. Следующий момент времени равен 100. CPN Tools выполняет 3 шага в этот момент времени, перемещает 3 фишки в позицию c; все эти фишки имеют начальные временные метки, равные 100. В момент времени 200 перемещаются 2 фишки с временными метками 200 и, наконец, в момент времени 300 перемещаются 2 оставшиеся фишки.

Для описания временных задержек могут использоваться задержки переходов или задержки дуг. Задержки переходов в большинстве случаев более наглядные, так как переходы моделируют действия объекта. Но использование задержек дуг дает большую гибкость при построении моделей.

Для моделирования трафика удобно использовать случайные функции в качестве временных задержек. Большой выбор случайных функций с различными законами распределения, описанных в разделе 5.6, позволяет представить особенности трафика. В следующем примере формируется (генерируется) Пуассоновский поток кадров:

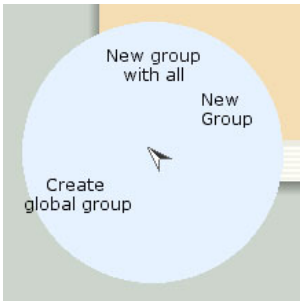


8. Работа с фрагментами сети

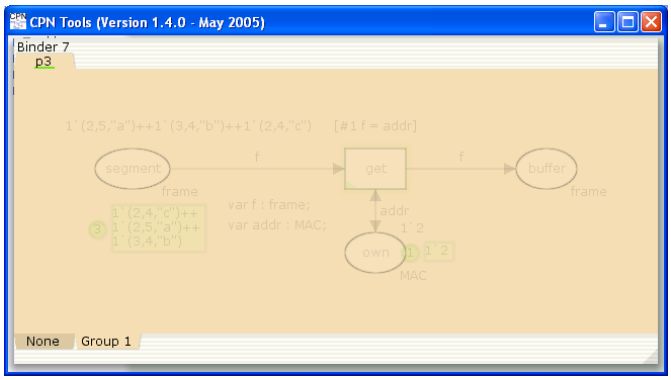
Работу с фрагментами можно рассматривать, как стандартную возможность графического редактора. Большинство графических редакторов (например, Corel Draw) обеспечивают выполнение операций с выделенными прямоугольными фрагментами изображения. В CPN Tools понятие фрагментов

сетей обобщено и сформулировано в терминах группы элементов. Группа может принимать любую форму и задаётся путём указания (включения) элементов, входящих в ее состав. Затем фрагмент можно скопировать и переместить в любое место сети, используя инструмент `Clone Tool` на палитре `инструментов для создания элементов сети (Create)`.

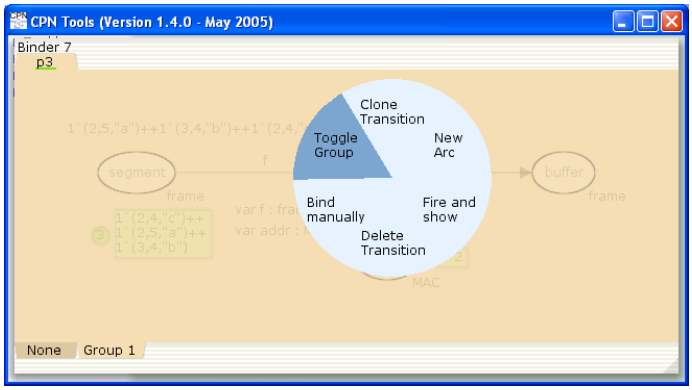
Чтобы создать группу, необходимо использовать инструмент `Новая группа (New group)`. При щелчке левой кнопкой мыши на групповом ярлычке в левой нижней области окна, появляется соответствующее меню:



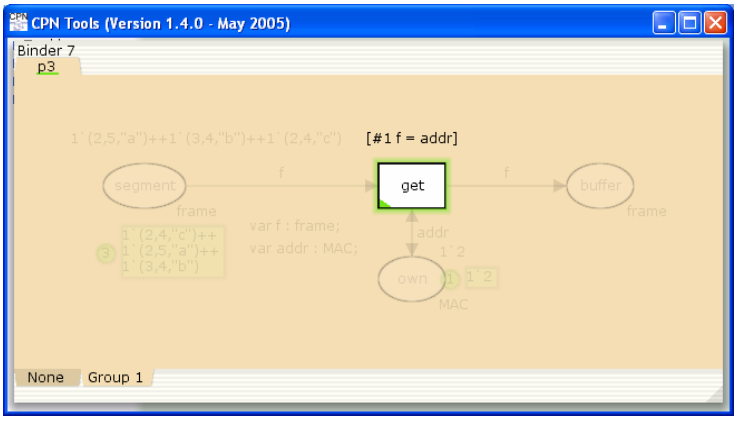
Выбираем `Новая группа (New group)`. Далее появляется новый групповой ярлычок, и новая группа устанавливается как активная группа. Все объекты на странице затемнены, что показывает отсутствие элементов в новой группе:



Для изменения активной группы необходимо кликнуть на групповом ярлычке той группы, которую необходимо активировать. Групповой ярлычок слева (первый ярлычок группы, называемый `None`) функционирует иным образом: кликнуть на этом ярлычке означает деактивировать все группы. При этом все элементы на странице принимают обычный вид, и работа возобновляется. Чтобы добавлять элемент в активную группу, необходимо использовать инструмент `Toggle group`. Например, выделим меню на элементе и кликнем `Toggle group`:



Если элемент отсутствует в активной группе, он добавляется в нее и выделяется. Атрибуты, закрепленные за элементом, автоматически добавляются в группу:



Чтобы исключить элемент из активной группы, необходимо использовать тот же инструмент `Toggle group`. Например, вызовем меню на элементе и кликнем `Toggle group`. Если элемент присутствует в активной группе, он исключается из группы (и становится затемнённым).

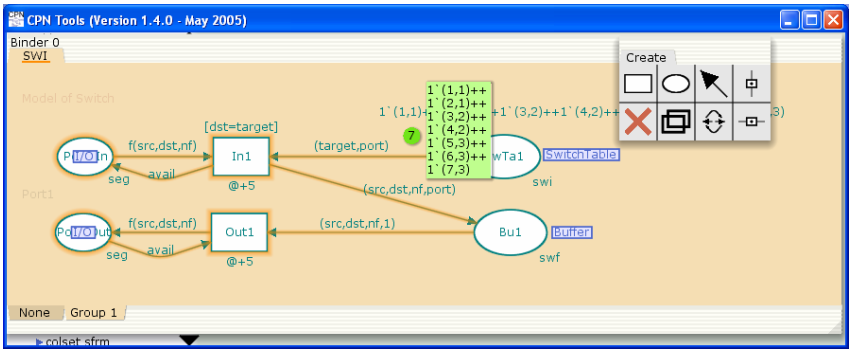
Группы используются для следующих целей:

- Изменение атрибутов;
- Перемещение групп элементов;
- Клонирование (копирование) групп элементов;
- Удаление групп элементов.

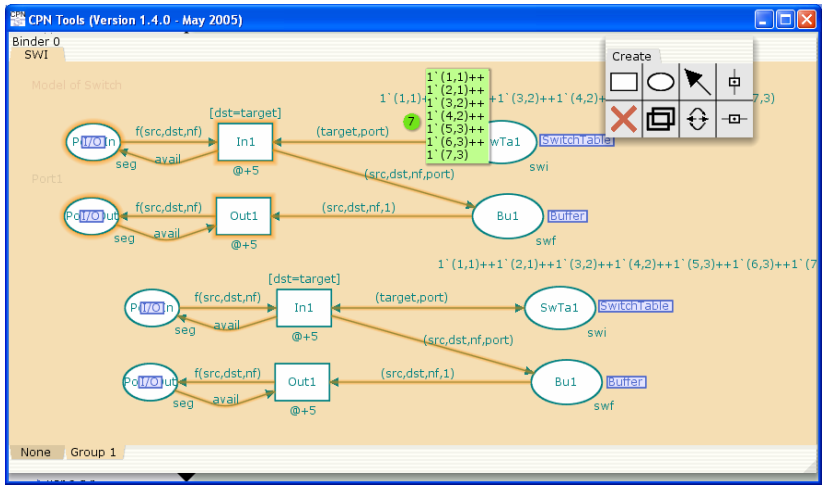
Группы могут использоваться для одновременного изменения свойств нескольких элементов. Если, например, в группе размещено несколько позиций, переходов и дуг, можно изменить их цвет одновременно. Группы могут также использоваться для одновременного перемещения нескольких различных элементов. Чтобы переместить группу элементов, необходимо перемещать один из элементов группы, и одновременно переместятся все другие элементы группы. Удаление элементов группы осуществляется аналогично вышерассмотренным действиям. Обычные группы занимают одну страницу, но с помощью глобальных групп можно манипулировать элементами нескольких страницы сети.

Клонирование групп элементов является расширенным вариантом базового клонирования, когда одновременно копируются несколько элементов.

Если выбрана группа, и элемент, к которому применяется клонирование, входит в эту группу, то будут клонированы *все элементы группы*. Как и при клонировании отдельных элементов, результатом является курсор с присоединёнными к нему клонируемыми элементами, что позволяет использовать их один или более раз при вставке в сеть. Клонирование групп является полезной операцией при создании сетей с регулярной структурой или для использования ранее созданных сетей. В Приложении П3 в примере модели коммутатора Ethernet очень удобно таким способом клонировать подмодели отдельных портов для компоновки модели коммутатора. Выделим в группу элементы первого порта:



И клонируем ее для создания моделей других портов:



Остается переместить группу в подходящее место сети и откорректировать названия элементов.

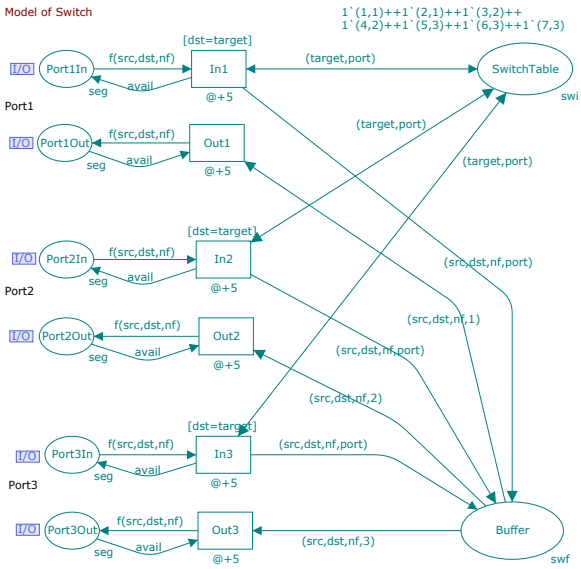
Отметим, что каждый клонируемый элемент может быть размещен в произвольной открытой сети. Таким образом, клонируемый элемент может быть перемещен между сетями. Этот способ применим и для групп элементов, то есть можно клонировать целые группы элементов в другие сети.

9. Слияние позиций

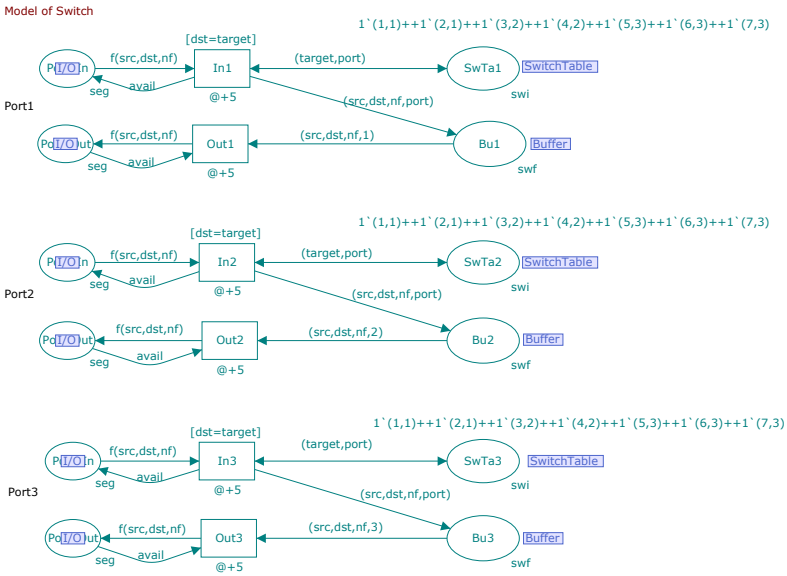
Слияние позиций обеспечивает две возможности: сделать модель более наглядной и установить связи между страницами сети. Слияние позиций можно рассматривать как первый шаг к иерархическим сетям Петри, поэтому

пиктограмма Слияние (Fusion) расположена в палитре инструментов Иерархии (Hierarchy).

Каждая позиция множества слияния дополняется тегом слияния с одним и тем же именем: все позиции множестве слияния рассматриваются CPN Tools как одна позиция. Визуально изменение маркировки одной из позиций слияния приводит к изменению маркировки всех позиций множества. Позиции множества слияния должны иметь одинаковое множество цветов. Рассмотрим пример модели коммутатора Ethernet:



Кадры извлекаются из входного канала исходного порта Port*In, помещаются в Buffer, а затем направляются в выходной канал порта назначения Port*Out. Чтобы найти номер порта назначения, используется таблица коммутации SwitchTable. Модель содержит большое количество пересечённых линий, и с увеличением количества портов становится трудно читаемой. Аналогичная модель построена в Приложении П3 с использованием двух множеств слияния: SwitchTable и Buffer.



Модель может быть легко расширена для произвольного количества портов с использованием клонирования групп элементов порта.

Слияние позиций задаётся при помощи инструмента `Assign fusion set` в палитре инструментов `Иерархии`. После применения инструмента, к позиции добавляется тег слияния. Тег размещается возле позиции и содержит имя множества слияния по умолчанию. Тег можно переместить, а имя множества слияния откорректировать, редактируя стандартный текст внутри тега слияния. Элементы множества слияния можно найти, поместив курсор поверх тега слияния. Свечение цвета морской волны указывает на то, какой позиции принадлежит тег слияния. Свечения розового цвета и подсветки указывают на другие позиции того же множества слияния и страницы, содержащие эти позиции. Множества слияний могут содержать позиции из разных страниц.

10. Построение иерархических моделей

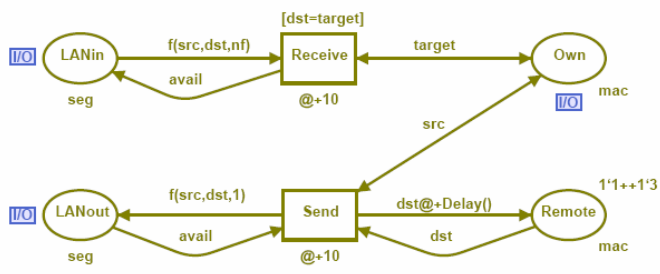
Иерархия широко применяется при проектировании. Телекоммуникационное устройство состоит из блоков, блоки – из плат, платы – из чипов, емкостей, сопротивлений и т.п. В программировании программы состоят из модулей (процедур, функций). Иерархическая модель означает вложенную конструкцию: сеть внутри сети. Любой элемент сети Петри может быть заменен вложенной сетью, но в CPN Tools используется только подстановка переходов.

10.1. Основы подстановки переходов

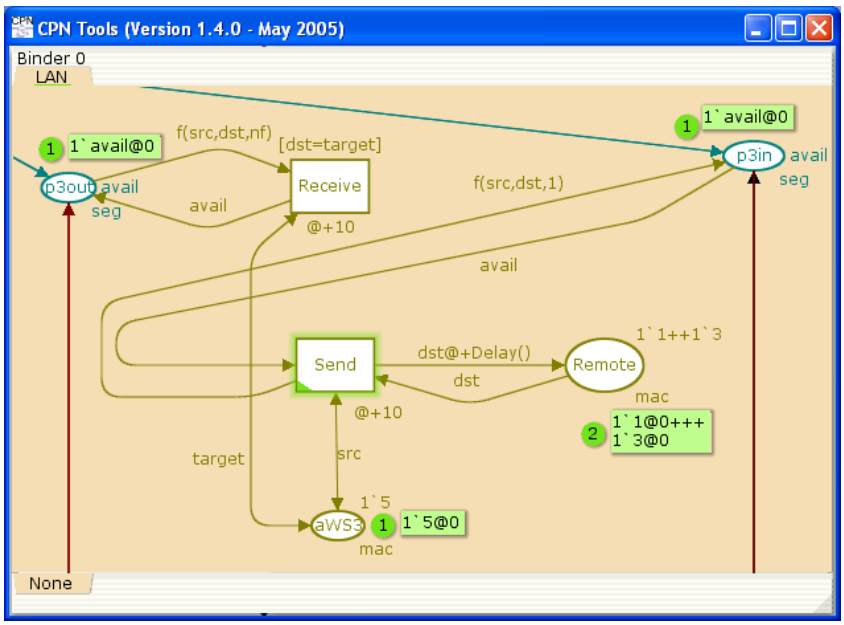
При подстановке перехода используется, по крайней мере, пара сетей. Переход сети верхнего уровня логически заменяется сетью более низкого уровня. Рассмотрим модель LAN, описанную в Приложении. На странице `LAN` переход `WS3` заменяется сетью `WS`:

Фрагмент модели LAN

Модель рабочей станции



Подстановка отображается тегом `WS` около перехода `WS3`. Логически, поведение всей сети такое же, как и при размещении подсети `WS` внутри сети `LAN`:



Позиции сети нижнего уровня, которые используются для связи с сетями верхнего уровня, называются контактными позициями или *портами* и выделены специальным тэгом (`I/O`). Соответствующие позиции сетей верхнего уровня называются *сокетами*. В рассмотренном примере портами являются `LANin`, `LANout`, `Own` на странице `WS` и сокетами – `p3out`, `p3in`, `aWS3` на странице `LAN`.

При подготовке подстановки перехода следует:

– создать обе сети, как верхнего, так нижнего уровней, расположенные на отдельных страницах модели (`LAN, WS`);

– указать порты сети нижнего уровня (`LANin`, `LANout`, `Own`);

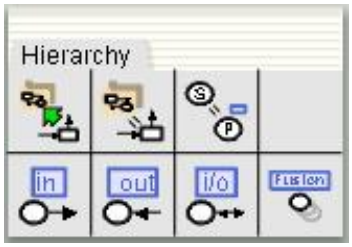
– обеспечить количество сокетов, равное количеству портов (3).

При подстановке перехода следует:

– установить соответствие сеть – переход (`WS3->WS`);

– установить соответствие порты – сокеты (`LANin->p3out, LANout->p3in, Own->aWS3`).

Порты сети низкого уровня можно отметить, используя инструменты палитры `Иерархии`:



Доступны три типа тегов: `In, Out, I/O`. Порты типа `In` используются, когда соответствующая позиция имеет только входящие дуги из сети верхнего уровня и только исходящие дуги в сеть нижнего уровня. Порты типа `Out` используются, когда соответствующая позиция имеет только исходящие дуги в сеть верхнего уровня и только входящие дуги из сети нижнего уровня. На языке

программирования их можно рассматривать как переменные ввода/вывода. Когда нет ограничений на направления дуг, используются порты типа I/O. Такая позиция может иметь произвольные дуги обоих направлений, как для сети верхнего уровня, так и для сети нижнего уровня.

Для установки соответствия страница – переход используется второй инструмент палитры Иерархии. Следует кликнуть ним на переходе (WS3) и затем на подстранице (WS). Соответствующий тег подстраницы будет прикреплен к переходу.

Установка соответствий между портами и сокетами – более сложная процедура, так как она должна быть выполнена для каждого порта подстраницы. Для этой процедуры используется третий инструмент палитры Иерархии. Необходимо открыть обе страницы на экране. Чтобы установить соответствие порт – сокет, следует кликнуть на порт, а затем – на соответствующий сокет.

Отметим, что использование иерархических сетей упрощает разработку моделей, аналогично использованию модулей в языках программирования. Во-первых, это позволяет скрыть детали и управлять сложностью модели. Во-вторых, позволяет повторно использовать подмодели. Например, в модели локальной сети LAN есть 5 рабочих станций и 2 сервера, но созданы одна подстраница для рабочей станции (WS) и одна подстраница для сервера (S).

10.2. Восходящая разработка

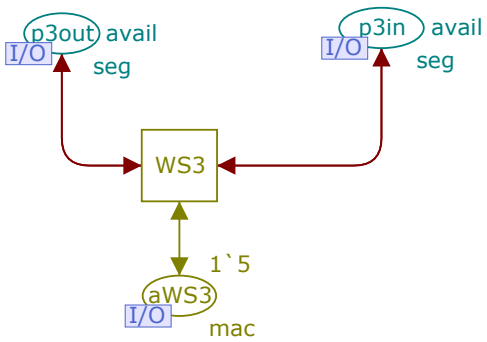
При восходящем создании иерархической сети, начинают с построения отдельных страниц. В отличие от нисходящей разработки, этот метод состоит в создание в первую очередь наиболее подробных частей модели. Далее, существующие страницы устанавливаются как подстраницы для подстановки переходов, как было описано в предыдущем подразделе. Это аналогично использованию библиотек в языках программирования.

Например, в модели локальной сети LAN следует создать страницы WS, S, MWS, SWI, а затем создать верхнюю страницу LAN, произвести подстановку переходов (WS1–WS4->WS, S1–S4->S, WS5->MWS) и отобразить соответствующие порты – сокеты.

10.3. Нисходящая разработка

При нисходящем создании иерархической сети, начинают с построения страницы высокого уровня, которая использует подстраницы (ещё не созданные) и показывает их взаимосвязи. В примере локальной сети LAN из Приложения следует сначала создать главную страницу LAN. Существует специальная функция «переместить переход на подстраницу» в палитре инструментов Иерархия (первый инструмент). В результате тег подстраницы добавляется к переходу, который является теперь замещенным переходом. Новая страница создается копированием позиций, окружающих данный переход. Страница называется по имени данного перехода. Эта страница

является начальным шаблоном для создания низкоуровневой сети. Например, применение операции к `WS3` модели локальной сети LAN создает следующий шаблон:



Чтобы создать подмодель рабочей станции следует отредактировать шаблон. Например, переименовать страницу `WS3` в `WS`, переименовать позиции `p3out` в `LANin`, `p3in` в `LANout`, `aWS3` в `Own`, удалить переход `WS3` и построить сеть `WS`, используя созданные порты. Описанный механизм автоматически выполняет установку тега подстраницы, тегов типа портов и устанавливает соответствие порт – сокет. Таким образом, это позволяет исключить множество рутинных операций.

Отметим, что построение моделей требует применения как восходящих, так и нисходящих методов разработки, так как инструмент «переместить переход на подстраницу» можно применить только один раз для каждой новой подстраницы. Например, таким способом можно создать `SWI`, `MWS`, затем `WS` для `WS1` и `S` для `S1`. Но для `WS2`–`WS4` и `S2`, используются ранее созданные `WS` и `S` восходящим способом.

11. Анализ раскрашенной сети Петри

Система CPN Tools предоставляет два основных способа анализа моделей: имитация поведения сети и формирование пространства состояний. Кроме того, у пользователя должна быть уверенность в том, что модель соответствует объекту и работает надлежащим образом. Анализ включает в себя предварительный этап, схожий для всех языков программирования и обычно называемый отладкой. На данном этапе пользователь обретает уверенность в том, что модель работает должным образом и ошибки исправлены. Предлагается также специальный способ анализа моделей, называемый, измерительными фрагментами. Для оценки характеристик моделей создаются специальные фрагменты сетей, которые вычисляют характеристики во время имитации. Такой фрагмент построен в Приложении П5 для оценки времени отклика сети Ethernet.

11.1. Отладка моделей

Отладка моделей включает в себя проверку синтаксиса и пошаговое моделирование. CPN Tools автоматически проверяет синтаксис сети при ее

создании или загрузке. По цветным индикаторам можно увидеть, насколько выполнилась проверка. Цветные индикаторы показаны в индексе, подчеркивая имя страницы, на которой находится цвет. Если страница открыта в графическом редакторе, цвет также показан наверху в ярлыке страницы, и на CPN–элементе. Оранжевое свечение указывает на то, что элемент в настоящее время не проверяется. При загрузке сети проверка синтаксиса занимает некоторое время. На этом этапе элементы изменяют свечение от оранжевого через желтое к отсутствию свечения (или красному, если обнаружена ошибка). Если оранжевое свечение остается то это, вероятно, потому, что либо что-то пропущено в описании, либо соответствующий сетевой элементе содержит несерьёзную ошибку. Свечение желтого цвета указывает на тот факт, что позиция (переход, дуга, страница или сеть) проверяются в данный момент. Описания проверяются, начиная сверху. Если описание зависит от другого описания, представленного ниже, то выдается ошибка, указывающая, что описание не определено. Описания с ошибками повторно проверяются, когда сделано изменение в любом из них. Всплывающее сообщение указывает причину ошибки. Элементы, связанные с элементом с ошибкой, например переход, связанный с позицией с ошибкой, не проверяются, пока ошибка не будет устранена. Если в описании есть ошибка, то оно будет подчеркнуто красным цветом. Имя сети и все задействованные страницы также будут подчеркнуты красным цветом. Чтобы увидеть сообщение об ошибке в описании, необходимо переместить курсор мыши на описание.

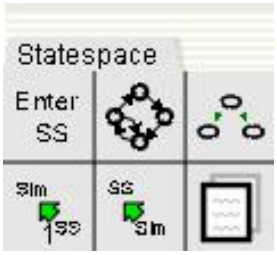
Пошаговое моделирование, описанное в разделе 4.3, используется, чтобы отследить путь передвижения фишек в модели. Например, можно отследить кадры в модели Ethernet, описанной в Приложении, на пути от рабочей станции до сервера и обратно. Можно также вручную выбрать параметры фишек при срабатывания переходов, например, выбрать входную фишку среди множества доступных. Для дальнейшей отладки используется выполнение заданного количества переходов, чтобы оценить поведение модели на больших интервалах времени.

11.2. Анализ пространства состояний

Пространство состояний раскрашенной сети Петри сложнее, чем множество достижимости или граф достижимых маркировок классической сети Петри. В классической сети Петри маркировка позиций представлена вектором натуральных чисел, а в раскрашенной сети Петри – вектором мультимножеств (временных мультимножеств).

Анализ пространства состояний возможен для достаточно небольших или простых моделей из-за известного эффекта бурного роста пространства состояний. Количество состояний для l-ограниченной сети Петри с m позиций оценивается как l^m. В телекоммуникациях анализ пространства состояний применяется, в основном, при верификации протоколов, когда необходима информация о стандартных свойствах сети, таких как ограниченность, безопасность, живость и т.д.

Палитра пространства состояний имеет вид:



Она содержит следующие инструменты:

– войти в пространство состояний (Enter SS);

– вычислить пространство состояний;

– вычислить граф сильно связных компонентов (SCC);

– из пространства состояний в имитацию (передать состояние);

– из имитации в пространство состояний (передать состояние);

– сохранить отчет.

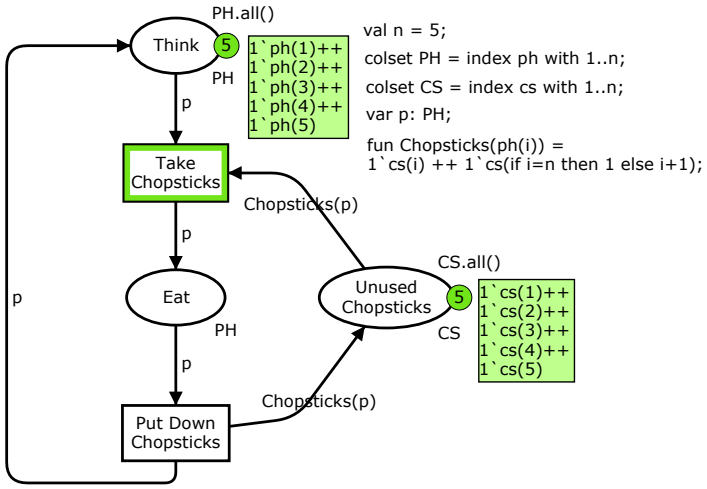
Прежде, чем пространство состояний может быть вычислено и проанализировано, необходимо сформировать код пространства состояний. Этот код создается, когда используется инструмент `Войти в пространство состояний`. Вход в пространство состояний занимает некоторое время. Затем, если ожидается, что пространство состояний будет небольшим, можно просто применить инструмент `Вычислить пространство состояний` к листу, содержащему страницу сети. Если ожидается, что пространство состояний будет большим, возможно, следует изменить настройки для инструмента `Вычислить пространство состояний`. Настройки для этого инструмента позволяют точно определить, когда остановить вычисления. Кроме того, граф сильно связных компонентов пространства состояний может быть вычислен при помощи соответствующего инструмента. Вычисленное пространство состояний сохраняется во временных файлах CPN Tools. Существует два способа проанализировать его:

– сохранить отчет в файле;

– создать запрос о пространстве состояний.

Чтобы сохранить отчет, необходимо применить инструмент `Сохранить отчет о пространстве состояний` к листу, содержащему страницу сети и ввести имя файла отчета. Содержимое отчета определяется настройками для этого инструмента. Запросы используются при исследовании свойств сети Петри с помощью написания специальных функций языка CPN ML. Они достаточно сложные и используют специальные предопределенные функции. Структура системы помощи CPN Tools содержит ссылку на описание запросов к пространству состояний.

Рассмотрим хорошо известный пример об обедающих философах:



Модель в CPN Tools имеет компактное представление благодаря использованию указанных множеств цветов для описания философов (`PH`) и палочек для еды (`CS`), а также функции `Chipsticks`, которая возвращает номера палочек, используемых философом `ph(i)`.

Сохраненный отчет имеет вид:

```
 Statistics
------------------------------------------------------------------------

  State Space
    Nodes:  11
    Arcs:   30
    Secs:   0
    Status: Full

  Scc Graph
    Nodes:  1
    Arcs:   0
    Secs:   0

 Boundedness Properties
------------------------------------------------------------------------

  Best Integers                 Bounds    Upper       Lower
  Page'Eat                      1         2           0
  Page'Think             1          5           3
  Page'Unused_Chopsticks        1         5           1

  Best Upper Multi-set Bounds
  Page'Eat 1              1`ph(1)++
                  1`ph(2)++
                  1`ph(3)++
                  1`ph(4)++
                  1`ph(5)
  Page'Think 1            1`ph(1)++
                  1`ph(2)++
                  1`ph(3)++
                  1`ph(4)++
                  1`ph(5)
  Page'Unused_Chopsticks 1
                          1`cs(1)++
                  1`cs(2)++
                  1`cs(3)++
                  1`cs(4)++
                  1`cs(5)
```

```
  Best Lower Multi-set Bounds
  Page'Eat 1                    empty
  Page'Think 1                  empty
  Page'Unused_Chopsticks 1    empty

 Home Properties
--------------------------------------------------------------------------
  Home Markings:  All

 Liveness Properties
--------------------------------------------------------------------------
  Dead Markings:  None
  Dead Transitions Instances: None

  Live Transitions Instances: All

 Fairness Properties
--------------------------------------------------------------------------
  Page'Put_Down_Chopsticks 1
                          Impartial
  Page'Take_Chopsticks 1 Impartial
--------------------------------------------------------------------------
```

Раздел статистики (Statistics) описывает размер пространства состояний и графа связных компонентов. Раздел ограниченности (Boundedness) указывает верхнюю и нижнюю границы маркировок в числовом виде и в виде мультимножества. Раздел домашних (Home) свойств указывает список домашних маркировок. Раздел свойств живости (Liveness) описывает тупики и живые переходы. Раздел свойств справедливости (Fairness) описывает тип справедливости переходов сети.

Отметим, что CPN Tools не дает возможности сохранить полное сформированное пространство состояний, которое размещается во временных файлах CPN Tools. Чтобы исследовать его вне границ стандартного отчета, следует написать специальные запросы к пространству состояний на языке CPN ML.

11.3. Имитация поведения сети

Система CPN Tools может использоваться как обычная система имитационного моделирования. Когда поведение сети достаточно сложное, можно смоделировать его на больших интервалах времени и сделать выводы о характеристиках моделируемой системы. В особенности, когда в модели широко используются случайные функции, больший интерес представляют ее статистические свойства, нежели ее пространство состояний. Например, можно имитировать поведение модели Ethernet в течение одного дня реального времени и сделать выводы о таких ее особенностях как среднее время отклика, процент коллизии и т.д.

С точки зрения обычной системы имитационного моделирования CPN Tools предоставляет недостаточно стандартных средств для такого анализа. В CPN Tools нет инструментов для управления временем, кроме быстрого исполнения заданного в палитре Моделирование количества шагов. Но есть

возможность выбрать достаточно большое количество шагов, чтобы обеспечить длительность процесса моделирования, соответствующую реальному времени. Кроме того, CPN Tools не вычисляет первичную статистическую информацию, такую как максимальное, минимальное и среднее количество фишек в позициях, частоты срабатывания переходов и т.д. Но система предоставляет язык для описания процессов накопления и вычисления характеристик; тот же самый язык раскрашенных сетей Петри может применяться для оценки статистических характеристик моделей. Такие дополнительные сети назовём измерительными фрагментами, они будут рассмотрены в следующем подразделе.

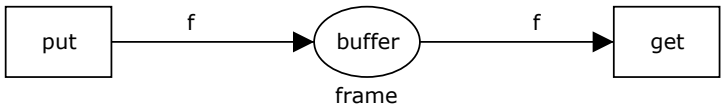
Имитационное моделирование предполагает специальную организацию экспериментов с моделью. Первое – *масштабирование времени*. Время в CPN Tools измеряется в единицах модельного времени (MTU), которые не имеет размерности, и представляется натуральным числом. Именно поэтому масштабирование времени представляет большой интерес, чтобы сделать модель реалистичной. Пример масштабирования времени описан в Приложении П7 для модели Ethernet. Определяются времена в реальных единицах (мс, нс) из описания аппаратных средств и программного обеспечения. Затем выбирается MTU, как наименьший интервал времени. Хотя для последующего развития модели рационально выбрать величину MTU ещё меньшей для моделирования более быстрой аппаратуры в будущем. Например, наименьшая задержка для модели Ethernet составляет 500 нс, но была выбрана MTU, равная 100 нс. Затем все времена модели пересчитывается в MTU. Например, 200 мс соответствует 200000 нс / 100 нс = 2000 MTU. После получения результатов моделирования время должно быть пересчитано обратно в реальные единицы времени. Например, полученное среднее время отклика равно 389 MTU или 38900 нс или 38.9 мс.

Второе – существование *стационарного режима* поведения модели. Если такой режим существует, модель сбалансирована. Увеличение длительности времени моделирования не вызывает существенное изменение ее статистических характеристик. Простейшим способом определить наличие стационарного режима является последовательное увеличение длительности времени моделирования. Если характеристики не изменяются после определенного момента времени моделирования, тогда стационарный режим имеет место. Отметим, что стационарный режим не может существовать, когда исходная сеть не сбалансирована; например, потоки со скоростью 100 Мбит/с направлены в сеть с пропускной способностью 10 Мбит/с.

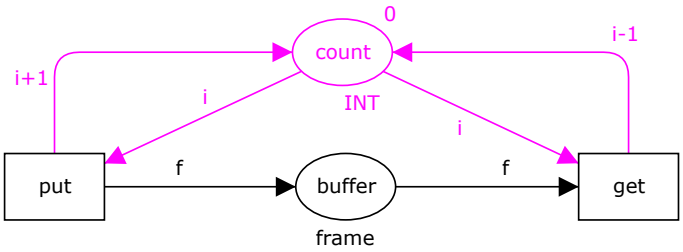
Третье – *оценка средних* (и других статистических моментов) характеристик в стационарном режиме. Предположим, что модель вычисляет характеристики, но результаты, полученные в единственном эксперименте с моделью, не представляют ценности. Должно быть проведено несколько экспериментов с моделью, и согласно математической статистике, их число должно быть около 20. Для более сложных оценок должен быть учтен доверительный интервал.

11.4. Измерительные фрагменты

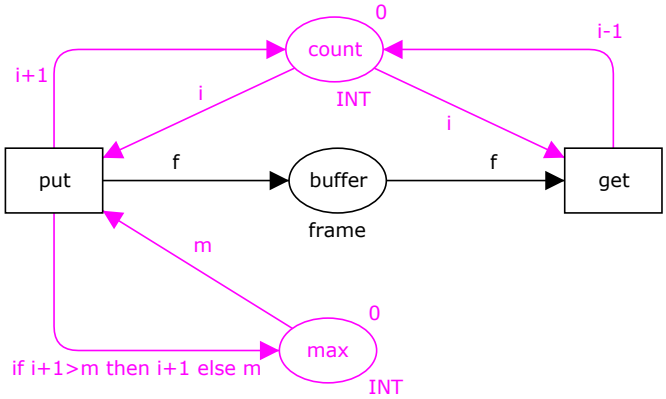
Так как язык раскрашенных сетей Петри представляет собой полную алгоритмическую систему, то он может применяться для накопления и вычисление статистических характеристик. Дополнительные фрагменты сети, добавленные к исходной модели для накопления и вычисления статистических характеристик, назовём, *измерительными фрагментами*. Рассмотрим несколько простых измерительных фрагментов. Пусть имеется буфер и переход `put`, увеличивающий его маркировку и переход `get`, уменьшающий его маркировку:



Можно легко вычислить количество фишек в буфере в данный момент, используя следующий фрагмент:



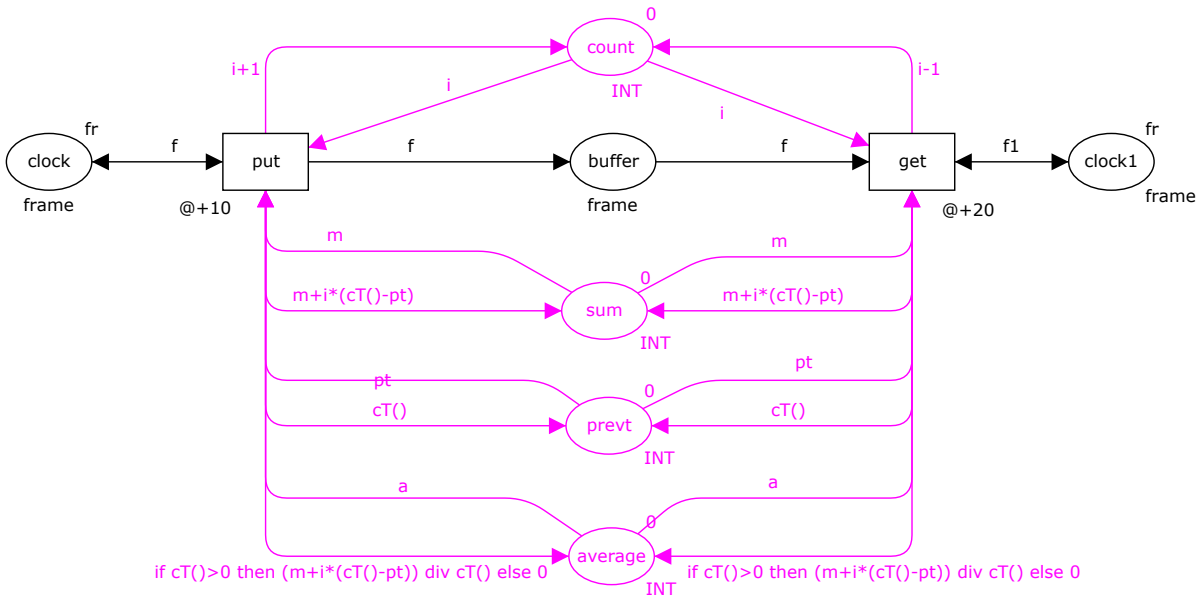
Чтобы вычислить максимальное количество фишек в позиции `buffer`, можно использовать следующий измерительный фрагмент:



Чтобы вычислить среднее число фишек, следует учитывать временные интервалы, так как среднее число распределения вычисляется по формуле:

```
ac=(c1*dt1 + c2 *dt2 + … + ck*dtk) / dt
```

где `ac` – среднее количество фишек, `ci` – величина на интервале времени `dti` и `dt` – общий интервал времени. Следующий фрагмент вычисляет среднее количество фишек в позиции `buffer`:



Перевычисления запускаются в двух случаях: при увеличении (put) и при уменьшении (get) количества фишек. Функция cT(), как описано в Приложении П2, возвращает текущее значение времени модели. Позиция `sum` сохраняет текущую сумму результатов. Позиция `prevt` сохраняет значение предыдущего момента времени, когда маркировка позиции `buffer` была изменена. Позиция `average` сохраняет среднее количество фишек в позиции `buffer`.

Более содержательный пример измерительных фрагментов для оценки времени отклика сети Ethernet описан в Приложении П5. Измерительные фрагменты могут применяться как для оценки пропускной способности телекоммуникационных сетей, так и для оценки характеристик качества обслуживания (QoS).

12. Дополнительные возможности CPN Tools

В системе CPN Tools существует множество дополнительных средств, описанных в системе онлайн помощи, а также в различных документах, например, в руководстве пользователя языка ML. Настоящий раздел содержит обзор наиболее значимых из них для моделирования телекоммуникационных систем.

12.1. Объединения

Множество цветов *объединения* является разделенным объединением ранее объявленных множеств цветов. Было указано ограничение, что позиция может содержать фишки только одного множества цветов; описание объединения множеств цветов позволяет преодолеть это ограничение. С помощью объединения (множеств цветов) можно размещать фишки различных множеств цветов в одной позиции.

Синтаксис описания имеет вид:

colset name = **union** id1[:name1] + id2[:name2] +... + idn[:namen];

Если namei пропущено, тогда idi интерпретируется как новая величина (unit), и на неё можно ссылаться как idi. Чтобы извлечь переменные определенного множества цветов, в объединении используются простые операторы:

idi v или idi(v),

где v имеет тип name.

В примере модели Ethernet, описанной в Приложении объединение множеств цветов используется для моделирования сегментов сети Ethernet. Если коллизии не учитываются, что является общим случаем в коммутируемой Ethernet, то сегмент либо свободен, либо передает кадр. Чтобы различать указанные случаи, описано специальное объединение seg:

```
colset seg = union f:frm + avail timed;
```

Величина avail означает, что сегмент свободен и доступен для передачи. Во втором случае сегмент передает фрейм f. В системе CPN Tools нет простого подхода для проверки позиции на отсутствие фишек (запрещающих дуг), для этого используется множество цветов seg. Рассмотрим подмодель коммутатора. Каждый входной канал портов коммутатора Port*In извлекает фрейм f и устанавливает вместо него метку avail. Это означает, что передача фрейма завершена и сегмент доступен и готов для передачи следующего фрейма. Каждый выходной канал портов коммутатора Port*Out ждет метку avail перед передачей, извлекает эту метку и передает кадр.

12.2. Списки

Множество цветов список состоит из последовательности элементов одного и того же множества цветов. Список является множеством цветов переменной длины. Стандартные функции дают доступ к обоим концам списка. Для обработки элементов внутри списка используются рекурсивные функции.

Синтаксис описания списка имеет вид:

```
colset name = list name0 [with int-exp1..int-exp2];
```

Модификатор with задаёт минимальную и максимальную длину списка. Величины множества цветов списка имеют вид:

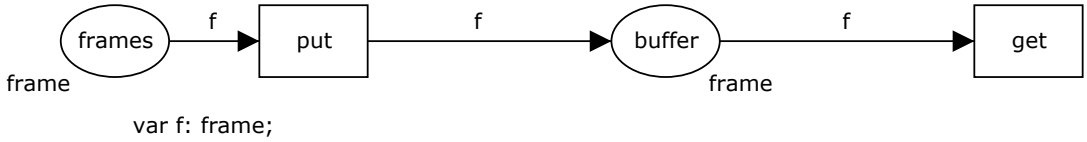
[v1, v2,..., vn], где vi типа name0 для i=1..n.

Для работы со списками используются следующие операции:

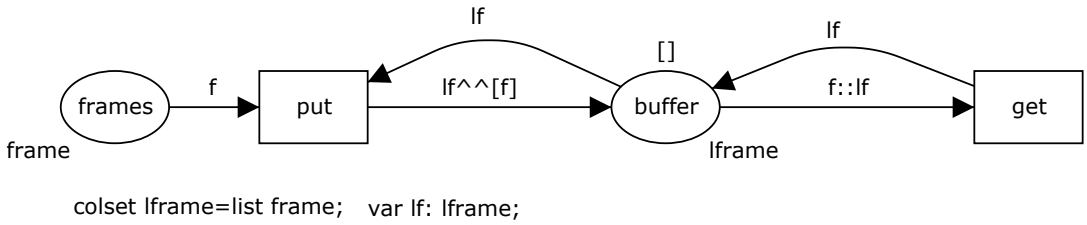
nil	пустой список (такой же как [])
e::l	добавить элемент e как голову списка l
l_1^^l_2	объединить два списка l_1 и l_2
hd l	голова, первый элемент списка l

`tl l`	хвост списка, список с исключенным первым элементом
`length l`	длина списка l
`rev l`	обратный список l
`map f l`	применяет функция f на каждом элементе списка l и возвращает список со всеми результатами
`List.nth(l,n)`	n–ый элемент в списке, где 0 <= n < длина l
`List.take(l,n)`	возвращает первые n элементов списка l
`List.drop(l,n)`	возвращает то, что осталось после исключения первых n элементов списка l
`List.exists p l`	возвращает истину, если p истинно для некоторого элемента списка l
`List.null l`	возвращает истину, если список l пустой

Обычно позиция CPN Tools обеспечивает дисциплину случайного доступа, так как случайная допустимая фишка может быть извлечена переходом:



Но телекоммуникационные устройства широко используют очереди (первым пришел – первым обслужен, FIFO), приоритетные дисциплины, стеки (последним пришел – первым обслужен, LIFO) и другие. Списочное множество цветов позволяет организовывать требуемую дисциплину обслуживания. Рассмотрим пример создания очереди (FIFO):



Чтобы понять множество цветов списка, следует поместить определенные кадры в позицию `frames` и проследить поведение этой сети. Отметим, что в данном случае позиция `buffer` содержит только одну фишку и эта фишка является списком кадров. В начальной маркировке список пустой `[]`.

В помощи CPN Tools рассматривается LIFO и приоритетная дисциплины обслуживания. Список позволяет также представлять запрещающие дуги; соответствующие примеры приведены в системе помощи CPN Tools. Более сложные примеры с рекурсивными функциями представлены в [5].

Приложение:
Оценка времени отклика сети с помощью модели коммутируемой ЛВС в форме раскрашенной сети Петри

П1. Коммутируемая ЛВС

В последнее время Ethernet стала наиболее широко распространенной технологий, применяемой в ЛВС. Новый этап популярности начался с технологии гигабитной передачи; и технология продолжает развиваться дальше (10Gbps). Концентраторы являются пассивным оборудованием, предназначенным для соединения устройств с помощью проводов. Основным элементом локальной сети (LAN), работающей по технологии Ethernet (IEEE 802.x), является коммутатор кадров. Коммутатор представляет собой устройство с множеством портов. Сегмент LAN (например, образованный с помощью концентратора) или терминальное оборудование, такое как рабочая станция или сервер, может быть подключено к любому порту. Задача коммутатора – перенаправление прибывшего кадра в порт, к которому подключено устройство назначения. Использование коммутатора позволяет уменьшить количество коллизий, так как каждый кадр передается только в порт назначения, что приводит к повышению пропускной способности. Кроме того, повышается качество защиты информации с уменьшением возможности прослушивания трафика. Пример схемы коммутируемой сети представлен на рис. 1.

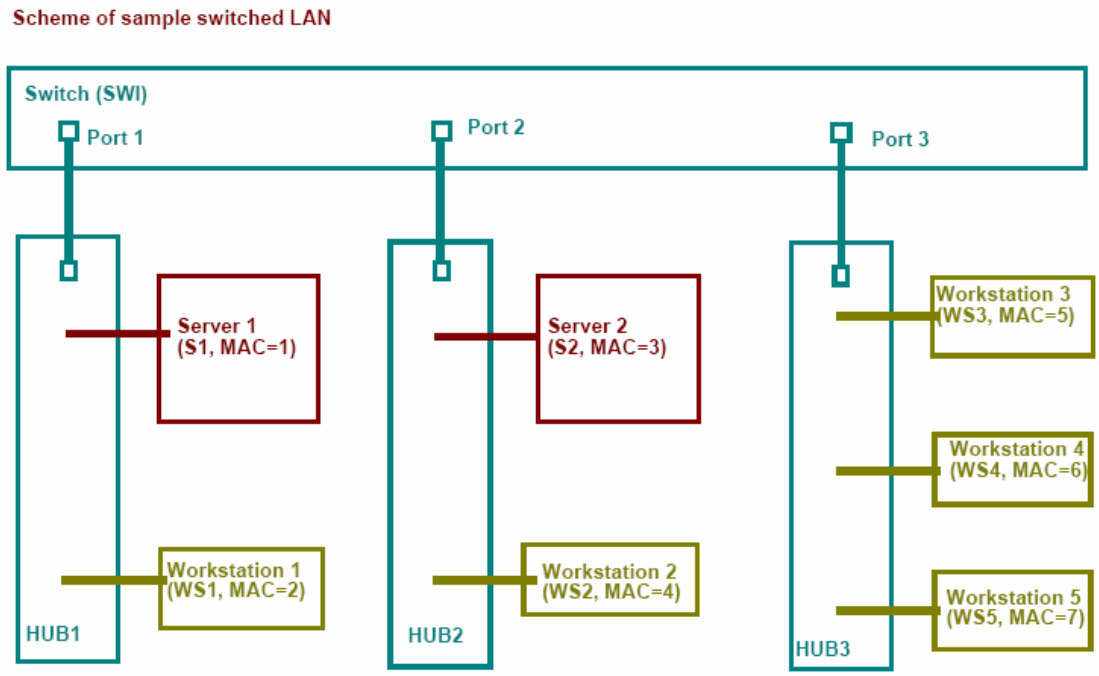


Рис. 1 – Пример схемы коммутируемой LAN

Как правило, Ethernet работает в полнодуплексном режиме, что позволяет одновременно передавать информацию в обоих направлениях. Для определения номера порта назначения входящего кадра используется статическая или

динамическая таблицы коммутации. Такая таблица содержит номер порта для каждого известного MAC адреса. Только статические таблицы коммутации будут использованы при построении модели; моделирование процессов построения динамических таблиц коммутации изучено в [5].

П2. Модель ЛВС

Модель локальной сети LAN, показанной на рис. 1, представлена на рис. 2. Опишем построенную модель. Отметим, что модель представлена в форме раскрашенной сети Петри и состоит из позиций, изображённых эллипсами, переходов, изображённых прямоугольниками, и дуг. Динамические элементы модели, представленные фишками, расположены в позициях и перемещаются по сети в результате срабатывания переходов.

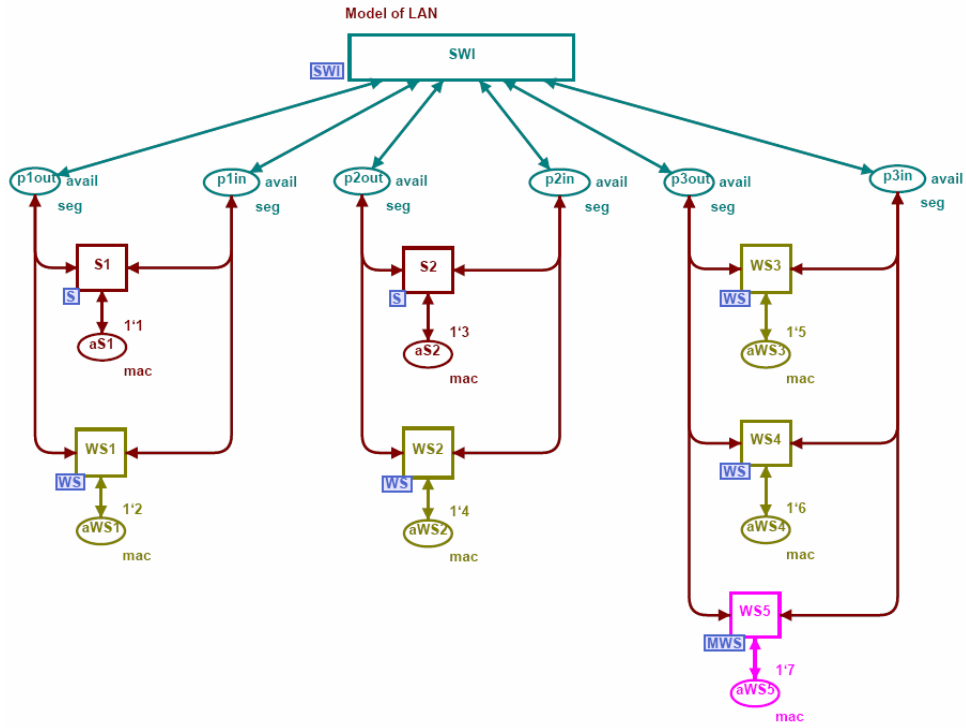


Рис. 2. – Модель примера локальной сети LAN

Элементами построенной модели являются подмодели: коммутатора (**SWI**), сервера (**S**), рабочей станции (**WS**) и измерительной рабочей станции (**MWS**). Рабочие станции **WS1–WS4** представлены одной и той же подмоделью, а именно **WS**, а рабочая станция **WS5** представлена подмоделью **MWS**. Она осуществляет измерение времени отклика сети. Серверы **S1** и **S2** представлены подмоделью **S**. Концентраторы являются пассивным оборудованием и не имеют отдельного модельного представления. Функции концентраторов смоделированы общим использованием соответствующих позиций **p*in** и **p*out** всеми подключенными устройствами. Данная модель не рассматривает коллизии.

Каждый сервер и каждая рабочая станция имеет свой MAC адрес, указанный в позициях **aS***, **aWS***. У коммутатора для каждого порта есть отдельные позиции для входящих (**p*in**) и исходящих (**p*out**) кадров. Таким

образом, моделируется полнодуплексный режим работы. Двунаправленные дуги используются для моделирования процедур обнаружения несущей. Одна дуга проверяет состояние канала, в то время как другая – выполняет передачу кадров.

Все описания множеств цветов (**colset**), переменных (**var**) и функций (**fun**) используемые в модели, представлены на рис. 3. MAC адреса Ethernet смоделированы целым числом (**colset mac**). Кадр Ethernet представлен тройкой **frm**, которая состоит из поля адреса отправителя (**src**), поля адреса получателя (**dst**), и специального поля **nfrm** для нумерации кадров, необходимого для вычисления времени отклика сети. При построении модели мы абстрагируемся от других полей кадра, предусмотренных стандартами технологии Ethernet. Множество цветов **seg** представляет однонаправленный канал, который может быть либо занят передачей кадра (**f.frm**), либо свободен (**avail**), что указывается с помощью объединения множества цветов (**union**). Отметим, что модификатор **timed** использован для фишек, которые участвуют во временных операциях, например, в задержках или метках времени.

```
colset mac = INT timed;
colset portnum = INT;
colset nfrm = INT;
colset sfrm = product nfrm * INT timed;
colset frm = product mac * mac * nfrm timed;
colset seg = union f:frm + avail timed;
colset swi = product mac * portnum;
colset swf = product mac * mac * nfrm * portnum timed;
colset remsv = product mac * nfrm timed;
var src, dst, target: mac;
var port: portnum;
var nf, rnf: nfrm;
var t1, t2, s, q, r: INT;
colset Delta = int with 1000..2000;
fun Delay() = Delta.ran();
colset dex = int with 100..200;
fun Dexec() = dex.ran();
colset dse = int with 10..20;
fun Dsend() = dse.ran();
colset nse = int with 10..20;
fun Nsend() = nse.ran();
fun cT()=IntInf.toInt(!CPN'Time.model_time)
```

Рис. 3 – Описания

В CPN Tools маркировка позиций представлена мультимножеством. Каждый элемент принадлежит мультимножеству с определённой кратностью, то есть в нескольких копиях. Например, начальная маркировка позиции **aWS2** – **1`4**. Это значит, что позиция **aWS2** содержит 1 фишку со значением 4. Объединение фишек показано знаком двойного плюса (++). Фишки временного цвета имеют вид **x @ t**; это значит, что фишка **x** может быть использована только после момента времени **t**. Запись **@+d** предназначена для представления задержки с продолжительностью **d**.

П3. Модель коммутатора

Построим модель для заданной статической таблицы коммутации. Рассмотрим отдельно входные и выходные буферы кадров для каждого порта и общий буфер для коммутируемых кадров. Модель коммутатора (**SWI**) представлена на рис. 4. Расположение хостов в соответствии с рис. 1 было использовано для построения начальной маркировки таблицы коммутации.

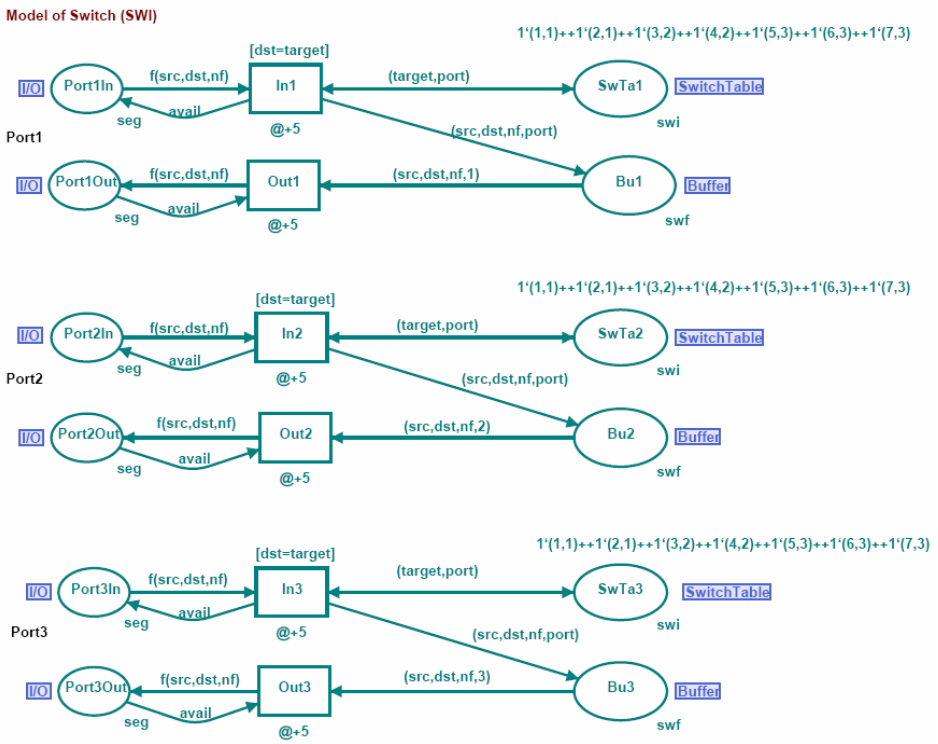


Рис. 4 – Модель коммутатора

Цвет **swi** представляет записи таблицы коммутации. Он отображает каждый известный MAC-адрес (**mac**) на соответствующий номер порта (**nport**). Цвет **swf** описывает коммутируемые кадры, ожидающие перенаправления в выходной буфер порта назначения. Поле **portnum** содержит номер соответствующего порта. Позиции **Port*In** и **Port*Out** представляют входные и выходные буферы портов соответственно. Позиция слияния **SwitchTable** моделирует таблицу коммутации; каждая фишка в этой позиции представляет строку таблицы коммутации. Например, фишка **1`(4,2)** начальной маркировки означает, что хост с MAC адресом 4 присоединён к порту 2. Позиция слияния **Buffer** соответствует буферу коммутируемых кадров. Отметим, что позиция слияния (например, **SwitchTable** или **Buffer**) представляет множество позиций. **SwitchTable** представлена позициями **SwTa1, SwTa2, SwTa3**. Позиция слияния **Buffer** представлена позициями **Bu1, Bu2, Bu3**. Это обеспечивает удобство моделирования коммутаторов с произвольным количеством портов, избегая многочисленных пересечений дуг.

Переходы **In*** моделируют обработку входящих кадров. Кадр извлекается из входного буфера порта только в случае, если таблица коммутации содержит запись с адресом, равным адресу получателя пакета (**dst=target**); при обработке

кадра номер порта назначения (**port**) сохраняется в буфере. Переходы **Out*** моделируют перенаправление коммутируемых кадров в выходные буферы портов. Атрибуты входящих дуг проверяют номер порта. Для операций коммутации устанавливается фиксированная временная задержка (@+5) и выполняется запись кадра в выходной буфер.

Необходимо более детально рассмотреть процедуры доступа CSMA в локальной сети LAN. Когда кадр извлекаются из входного буфера переходом **In***, выполняется его замещение меткой **avail**. Метка **avail** указывает на то, что канал свободен и готов к передаче. Перед тем как переход **Out*** отправляет кадр в порт, он анализирует, является ли канал доступным, проверяя наличие метки **avail**.

Отметим, что позиции **Port*In** и **Port*Out** являются контактными. Они отмечены тегом **I/O**. Контактные позиции используются для построения иерархических сетей путём подстановки перехода. Например, переход модели **SWI** на странице верхнего уровня (рис. 2) замещается целой сетью **SWI,** представленной на рис. 3. Позиции **Port*In** и **Port*Out** отображаются на позиции **p*in** и **p*out** соответственно.

П4. Модели рабочей станции и сервера

Для исследования потоков кадров, передающихся через ЛВС и для оценки времени отклика сети, необходимо построить модели терминальных устройств сети. Относительно особенностей формирования трафика следует различать рабочие станции и серверы. Для принятой степени детализации рассмотрим периодически повторяемые запросы от рабочих станций к серверам со случайными равномерно распределенными задержками. В ответ на принятый запрос сервер отправляет несколько пакетов ответа по адресу (запрашивающей) рабочей станции. Количества отправленных пакетов и временные задержки являются равномерно распределенными случайными величинами.

Модель рабочей станции (**WS**) представлена на рис. 5. Позиции **LANin** и **LANout** моделируют соответственно входящие и исходящие каналы локальной сети. Рабочая станция прослушивает сеть посредством перехода **Receive**, который получает кадры с адресом назначения, равным собственному адресу рабочей станции (**dst=target**), сохраненному в позиции **Own**. Обработка полученных кадров представлена простым поглощением их рабочей станцией. Рабочая станция отправляет периодические запросы на серверы с помощью перехода **Send**. Адреса серверов содержатся в позиции **Remote**. После отправки запроса использование адреса сервера блокируется на случайный временной интервал, заданный функцией **Delay**(). Передача кадра выполняется только в случае, если сегмент локальной сети LAN свободен, что обеспечивается путём проверки позиции **LANout** на наличие фишки **avail**. Таким образом, рабочая станция взаимодействует с несколькими серверами, сохраняя их адреса в позиции **Remote**.

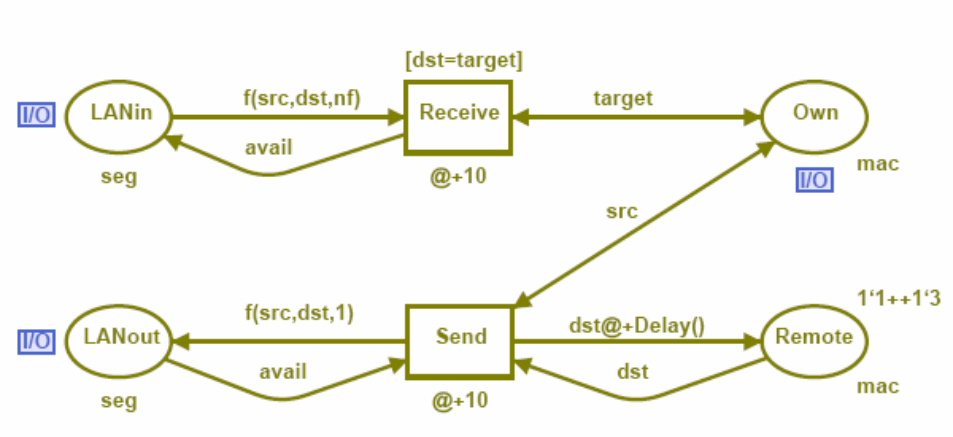


Рис. 5 – Модель рабочей станции

Отметим, что третье поле пакета, называемое **nfrm**, не используется стандартной рабочей станцией **WS**. Рабочая станция только устанавливает для него величину равную единице. Это поле используется специальной измерительной рабочей станцией **MWS**. Копии описанной модели **WS** представляют рабочие станции **WS1–WS4**. Для однозначной идентификации каждой рабочей станции используется контактная позиция **Own**. Эта позиция показана на странице верхнего уровня (рис. 2) и содержит MAC-адрес хоста.

Модель сервера (**S**) представлена на рис. 6. Прослушивание сети аналогично модели рабочей станции, но отличается тем, что адрес отправителя кадра сохраняется в позиции **Remote**. Переход **Exec** моделирует выполнение сервером запроса рабочей станции. В результате выполнения запроса сервер формирует случайное число **Nsend**() кадров ответа, которые размещаются в позиции **Reply**. Затем эти кадры передаются по одному в сеть с помощью перехода **Send**. Отметим, что номера **nf** из кадра запроса также сохраняется в позиции **Remote**. Это позволяет идентифицировать ответ тем же номером, что и запрос.

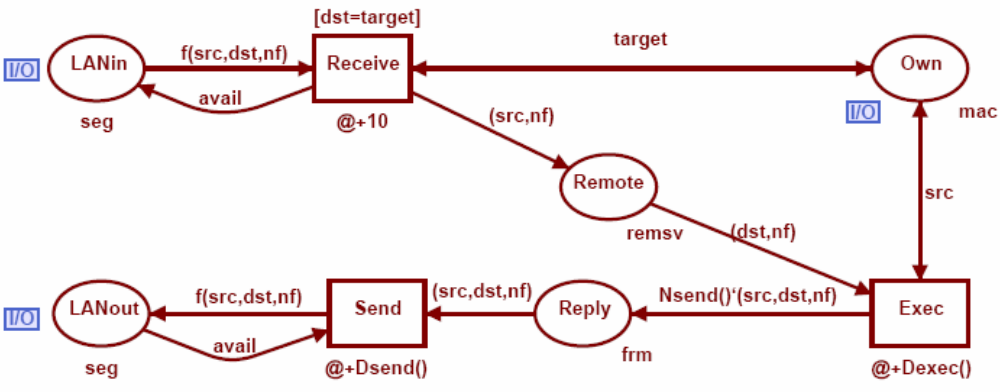


Рис. 6 – Модель сервера

П5. Модель измерительной рабочей станции

Модель измерительной рабочей станции (**MWS**) представлена на рис. 7. В сущности, она представляет собой ранее рассмотренную модель рабочей станции **WS**, дополненную измерительными элементами.

Рассмотрим измерительные элементы более подробно. Каждый пакет запроса рабочей станции обладает уникальным номером, хранящимся в позиции **num**. Время отправления запроса, сохраняется в позиции **nSnd**. Функция **cT**() определяет текущее значение модельного времени. Позиция **nSnd** хранит два значения: номер кадра запроса **nf** и время отправления запроса **cT()**.

Позиция **return** сохраняет временные метки всех вернувшихся кадров. Как время отклика сети, будем вычислять интервал времени между отправкой запроса и получением первого кадра ответа. Эта величина сохраняется в позиции **NRTs** для каждого запрос, на который получен ответ. Переход **IsFirst** распознаёт первый кадр ответа. Атрибут дуги, соединяющей переход **IsFirst** с позицией **NRTs**, вычисляет время отклика (t2–t1).

Оставшаяся часть измерительных элементов вычисляет среднее время отклика. Позиции **sum** и **quant** накапливают сумму времен и количество полученных ответов соответственно. Позиция **new** распознаёт новый ответ и запускает пересчет среднего времени отклика с помощью перехода **Culc**. Результат размещается в позиции **NRTime**.

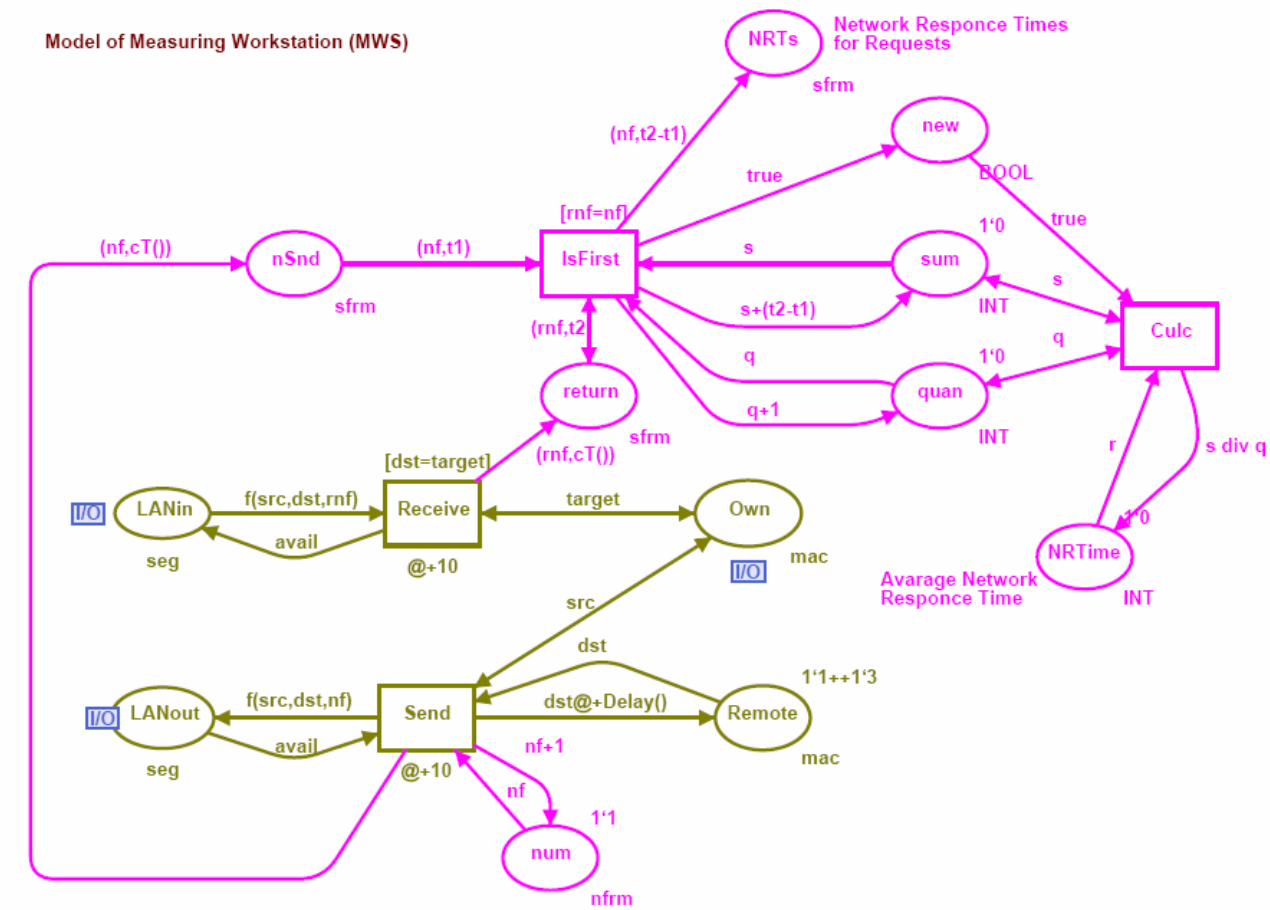


Рис. 7 – Модель измерительной рабочей станции

П6. Методика оценки

Построенная модель отлажена и протестирована в пошаговом режиме моделирования. Для этих целей трасса пакета, сформированного рабочей

станцией, отслеживалась в сети по направлениям к серверу и обратно. Также наблюдалось поведение модели в процессе автоматической имитации с отображением динамики сети – режим так называемой «игры» фишек. Это позволяет оценить модель визуально на странице верхнего уровня и подстраницах в процессе моделирования.

Для точной оценки времени отклика сети требуется рассматривать достаточно большие интервалы модельного времени. Для этого удобно использовать режим имитации без отображения промежуточных маркировок предназначенный для накопления статистики.

Образ экрана измерительной рабочей станции представлен на рис. 8. Прямоугольные метки (подсвеченные ярко зеленым цветом) описывают текущую маркировку системы моделирования; круглые метки содержат число фишек. Позиция **LANin** содержит кадр (1,5,1). Позиция **LANout** показывает свободное состояние канала **avail**. Номер последующего запроса, согласно маркировке позиции **num**, равен 7. Позиция **return** показывает, что получено 83 кадра ответов. Позиция **NRTs** содержит время отклика для каждого из 6 запросов, на которые получен ответ. Например, время отклика для запроса 5 равняется 235. Можно легко проверить, что среднее время отклика сети 389, указанное в позиции **NRTime**, равняется 2337/6 в соответствии с маркировками позиций **sum** и **quant**.

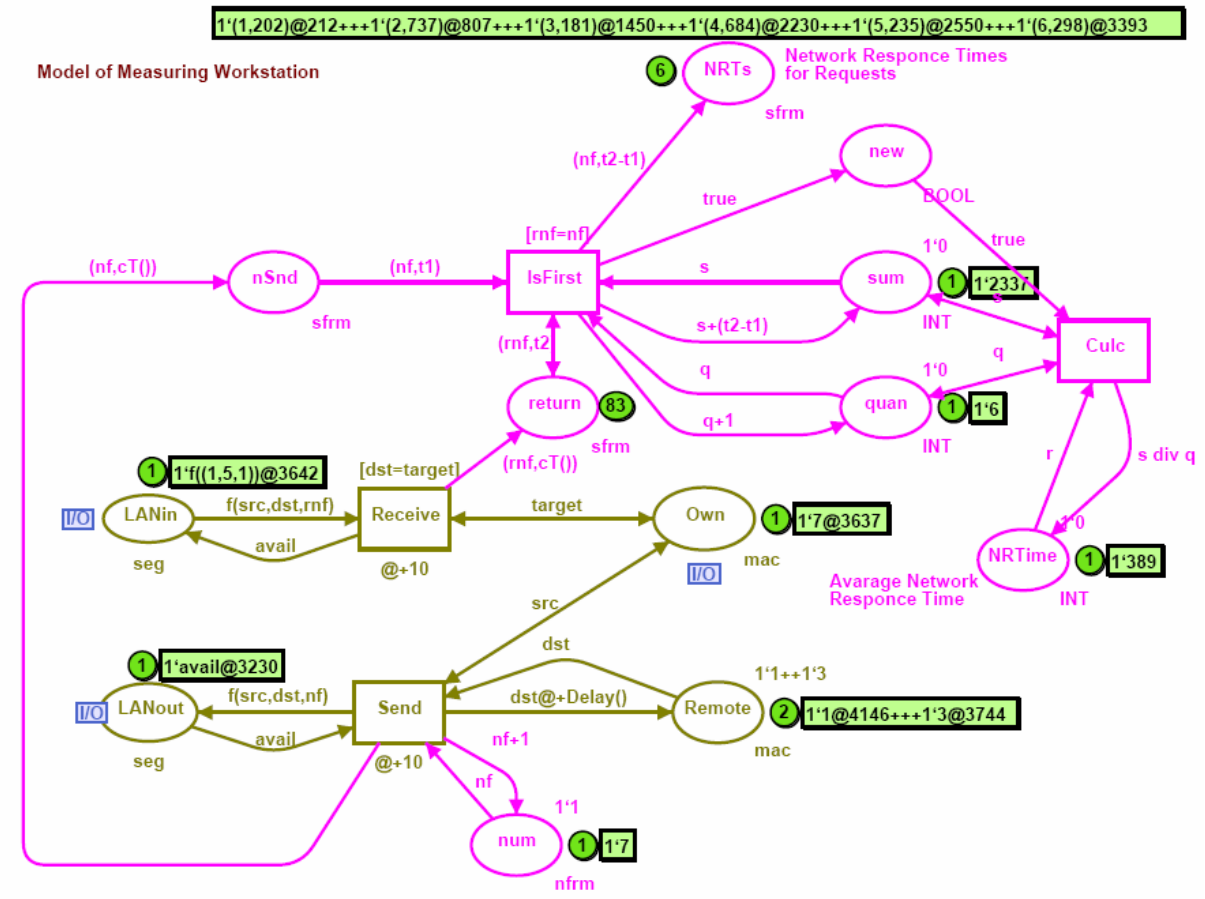


Рис. 8. Оценка времени отклика сети

П7. Параметры модели

Правильный выбор единицы измерения модельного времени является ключевым вопросом для построения адекватной модели, также как вычисление временных задержек элементов модели. Это требует точного изучения характеристик реальных сетевых аппаратных и программных средств.

Схема, показанная на рис. 1, представляет собой фрагмент ЛВС диспетчерского центра железной дороги, оснащённый специальным программным обеспечением ГИД Урал. Ядро системы составляет пару зеркальных серверов **S1** и **S2**. Рабочие станции **WS1–WS5** расположены на рабочих местах диспетчеров.

Необходимо рассмотреть производительность конкретного коммутатора ЛВС и сетевых адаптеров для вычисления временных задержек переходов **In***, **Out***, **Send**, **Receive**. Кроме того, специфика взаимодействия программного обеспечения клиент-сервер ГИД Урал должна быть использована для оценки таких параметров как задержка между запросами **Delta** и время выполнения запроса **dex**. Так как единицей информации, передаваемой по сети, является кадр, следует выразить длины сообщений в количестве кадров. Для этих целей выбрана максимальная длина кадра Ethernet равная 1.5 кбайт.

Типы используемого оборудования сети представлены в таблице 1.

Таблица 1 – Типы используемого оборудования

Устройство	Тип
Адаптер LAN	Intel EtherExpress 10/100
Коммутатор LAN	Intel SS101TX8EU
Сервер	HP Brio BA600
Рабочая станция	HP Brio BA200

В таблице 2 представлены параметры описанной модели. Операции коммутатора и сетевого адаптера смоделированы фиксированными задержками, так как они являются достаточно малыми по сравнению со временем взаимодействия клиент-сервер. Кроме того, в надёжной Ethernet передаются, в основном, кадры максимальной длины, поэтому время обработки кадра – фиксированная величина. Случайные величины представлены равномерным распределением, которое соответствует поведению программного обеспечения ГИД Урал. Минимальной временной величиной является время операции чтения/записи коммутатора ЛВС (500 нс). Но в целях будущего представления более производительного оборудования, следует выбрать меньшую единицу модельного времени (MTU), например, равную 100 нс.

Таблица 2 – Параметры модели

Параметры	Переменная/ Элемент	Реальная величина	Модельное время
Временная задержка чтения кадра коммутатором ЛВС	In*	500 нс	5
Временная задержка записи кадра коммутатором ЛВС	Out*	500 нс	5
Временная задержка чтения кадра сетевым адаптером	Receive	1 мс	10
Временная задержка чтения кадра сетевым адаптером	Send	1 мс	10
Время обработки запроса сервером	Dex	10–20 мс	100-200
Временной интервал между запросами клиента	Delta	100–200 мс	1000-2000
Длина запроса		1.2 кбит	1
Длина ответа	Nse	15–30 кбит	10-20

Таким образом, получено среднее время отклика сети равно 389 MTU или около 39 мс. Эта задержка удовлетворяет требованиям управления движением поездов.

Литература

1 Jensen K. Colored Petri Nets - Basic Concepts, Analysis Methods and Practical Use. – Springer-Verlag, 1997. – Vol. 1-3. – 673 p.

2 Albert K., Jensen K., Shapiro R. Design/CPN: A Tool Package Supporting the Use of Colored Nets // Petri Net Newsletter. – April 1989. – P. 22-35.

3 Zaitsev D.A. Switched LAN Simulation by Colored Petri Nets // Mathematics and Computers in Simulation. – 2004. – Vol. 65, № 3. – P. 245-249.

4 Zaitsev D.A. An Evaluation of Network Response Time using a Colored Petri Net Model of Switched LAN // Proc. of Fifth Workshop and Tutorial on Practical Use of Colored Petri Nets and the CPN Tools, October 8-11, 2004. – Aarhus (Denmark). – 2004. – P. 157-167.

5 Zaitsev D.A., Shmeleva T.R. Switched Ethernet Response Time Evaluation via Colored Petri Net Model // Proc. of International Middle Eastern Multiconference on Simulation and Modelling, August 28-30, 2006. – Alexandria (Egypt). – 2006. – P. 68-77.

Дополнительная литература

Зайцев Д.А., Шмелёва Т.Р. Моделирование коммутируемой локальной сети раскрашенными сетями Петри // Зв'язок, № 2(46), 2004, с. 56-60.

Зайцев Д.А. Измерительные фрагменты в моделях Петри телекоммуникационных сетей // Зв'язок №2(54), 2005, с. 65-71.

Зайцев Д.А., Шмелёва Т.Р. Основы построения параметрических моделей Петри коммутируемых сетей // Моделирование и компьютерная графика: Материалы 1-й международной научно-технической конференции, 4-7 октября 2005, Донецк, ДонНТУ, 2005, с.207-215.

Зайцев Д.А., Березнюк М.В. Исследование эффективности использования адресного пространства протокола Bluetooth // Радиоэлектроника. Информатика. Управление. - 2006, №1. - С. 57-63.

Зайцев Д.А., Сакун А.Л. Исследование эффективности технологии MPLS с помощью раскрашенных сетей Петри // Зв'язок. - 2006, №5. - С. 49-55.

Зайцев Д.А., Шмелёва Т.Р. Оценка характеристик сетей Ethernet с помощью параметрических моделей Петри // Зв'язок, № 4, 2007. - с. 62-67.