МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
ВОЛОГОДСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Кафедра автоматики и вычислительной техники

ТЕОРИЯ ВЫЧИСЛИТЕЛЬНЫХ ПРОЦЕССОВ

Методические указания к лабораторным работам

Моделирование раскрашенных сетей Петри в CPN Tools

Факультет электроэнергетический

Направления бакалавриата:

220201.62 - «Управление и информатика в технических системах»

230101.62 - «Вычислительные системы, сети и комплексы»

Вологда
2014

УДК 681.374.1

Теория вычислительных процессов: методические указания к лабораторным работам. Моделирование раскрашенных сетей Петри в CPN Tools. - Вологда: ВоГУ, 2014. – 60 с.

Рассматривается программный комплекс CPN Tools и его применение для построения моделей на базе математического аппарата раскрашенных сетей Петри, а также область применения и задачи, решаемые с помощью программы, основные функции программы, связанные с процессом построения, моделирования и анализа. Описываются средства построения моделей со сложной логикой функционирования и иерархией вложенных моделей.

Утверждено редакционно-издательским советом ВоГУ

Составители: А.А.Суконщиков, канд. техн. наук, доцент
Д.В. Кочкин, ст. преподаватель

Рецензент: А.М. Водовозов, канд. техн. наук, доцент, зав. кафедрой «Управляющие и вычислительные системы»

Лабораторная работа №1. Знакомство с CPN Tools

В данной лабораторной работе проводится первоначальное знакомство с интерфейсом и возможностями программы CPN Tools. Рассматривается процесс построения моделей, определение типов данных, объявление переменных, задание охранных выражений переходов, задание выражений на дугах. Рассматриваются вспомогательные графические элементы, предназначенные для организации элементов модели. Описывается процесс выполнения и отладки моделей. Рассмотренные возможности программы сопровождаются примерами.

Среда разработки

В качестве программной среды для изучения сетей Петри (СП) и построения моделей в данном методическом пособии используется программа, разработанная в университете Орхуса (Дания) - CPN Tools. Последняя версия программы может быть скачана с сайта http://cpntools.org/. На сайте также имеется пользовательская документация и обучающие примеры, позволяющие быстро освоить основные принципы построения моделей в CPN Tools.

Отличительной особенностью CPN Tools является наличие богатого инструментария, позволяющего анализировать различные аспекты функционирования моделей на базе СП (безопасность и ограниченность позиций, уровень активности переходов, наличие тупиковых маркировок). CPN Tools используется во множестве реальных проектов в области телекоммуникации, при моделировании сетей и сетевых устройств, а также при верификации протоколов связи.

В данной система моделирования для построения моделей используются иерархические временные раскрашенные СП (ИВР СП). ИВР СП представляет собой универсальную алгоритмическую систему и по выразительной мощности эквивалентна машине Тьюринга. Данное расширение может быть использовано для описания произвольного объекта.

Система CPN Tools поддерживает язык CPN ML, который используется для описания операций, условий и функций. Благодаря применению CPN ML программа позволяет упростить процесс построения и анализа моделей.

Имитационное моделирование в CPN Tools является дискретно-событийным, что предполагает мгновенную смену состояния СП в определенны моменты времени, называемые шагами. Наличие временных меток позволяет моделировать процессы с учетом времени выполнения.

Начало работы

После запуска CPN Tools пользователь увидит перед собой окно программы (рис. 1), разделенной на две части левая – область меню, правая – рабочая область. В области меню располагаются вкладка с инструментами для моделирования сети, вкладка с документацией и вкладка с настройками программы. После создания модели в области меню появится вкладка с элементами модели. В рабочей области располагаются окна с созданными моделями.

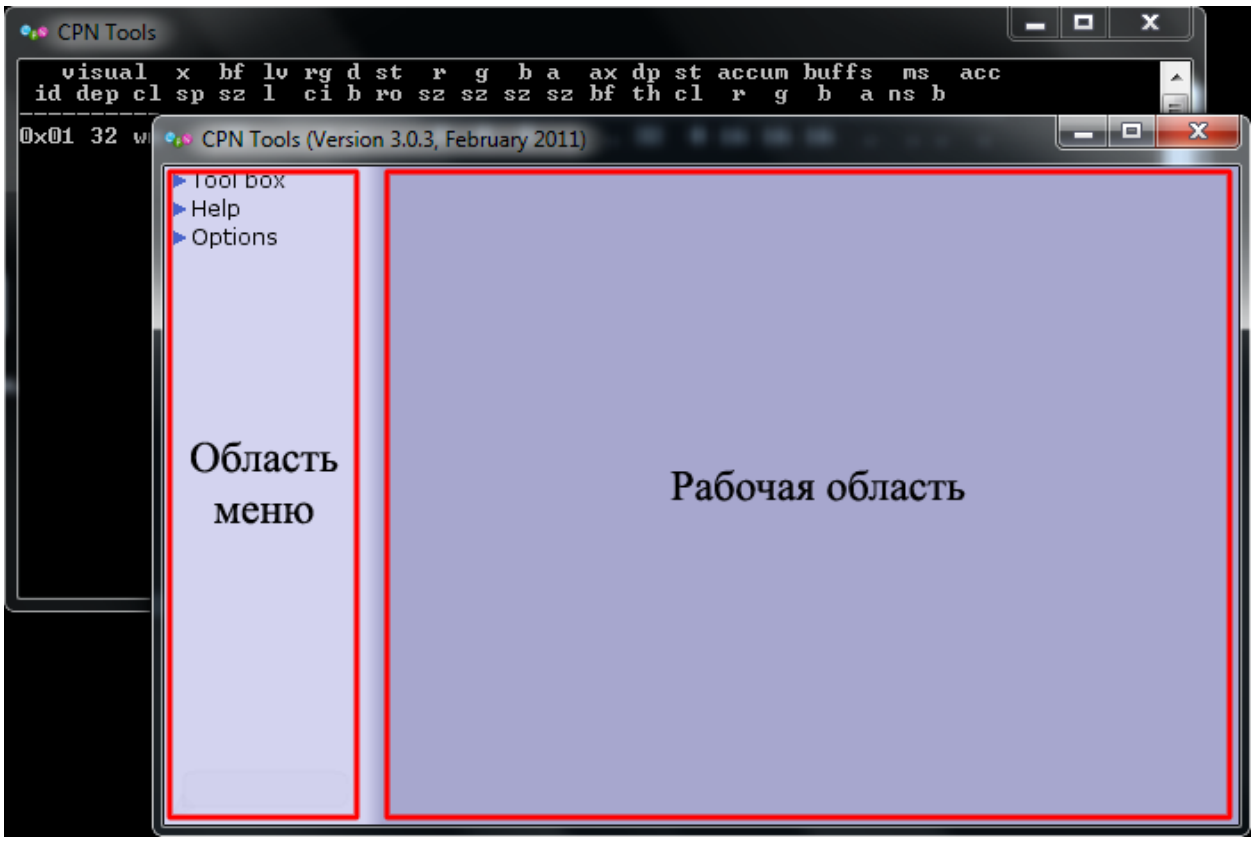


Рис. 1. Пользовательский интерфейс CPN Tools после запуска программы

Из пользовательского интерфейса доступны две функции:

- загрузка созданной сети «Load Net»
- создание новой сети «New Net»

Для создания новой сети (рис. 2) необходимо кликнуть правой кнопкой мыши в рабочей области и, удерживая кнопку в появившемся меню, выбрать пункт «New net», передвинув на него указатель мыши. Аналогичным образом организована работа с другими меню в программе CPN Tools.

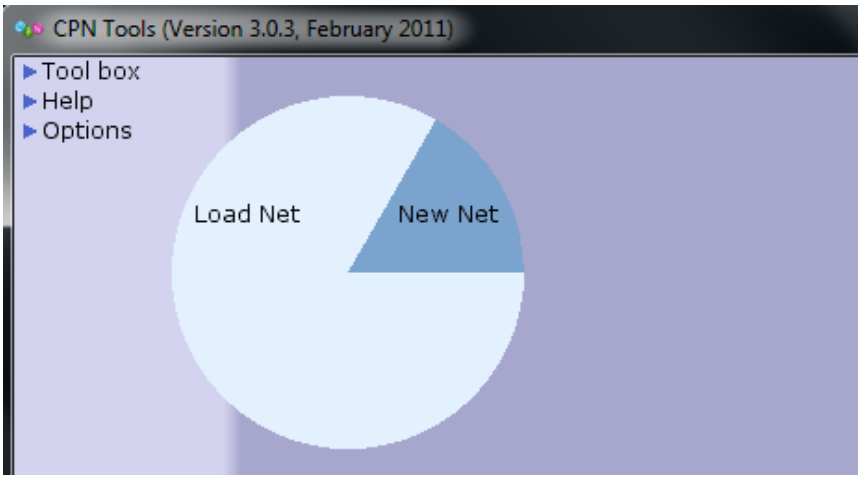


Рис. 2. Создание новой сети в CPN Tools

После создания новой сети окно программы примет вид, изображенный на рисунке 3. При этом в области меню появится новый пункт «New net.cpn», раскрыв который мы сможем изменить параметры модели, объявить типы данных и ввести новые переменные. Кроме этого правый клик мыши на пункте «New net.cpn» откроет контекстное меню, в котором можно будет сохранить сеть, отменить/повторить действие, создать новую страницу и закрыть сеть.

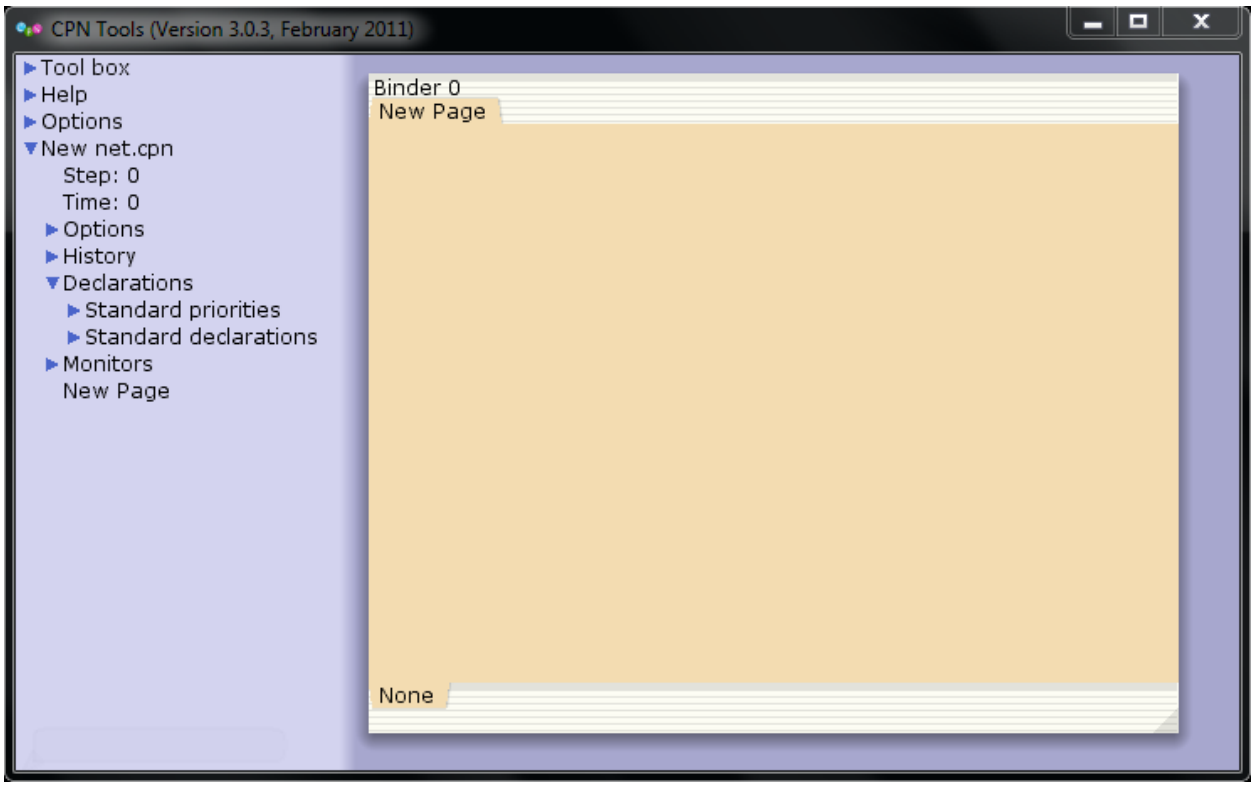


Рис. 3. Внешний вид программы CPN Tools после создания новой сети

Перед началом работы необходимо выполнить еще несколько действий. Кликните на пункте «Tool box» рабочей области, в раскрывшемся списке выберите пункт «Create», кликните на нем мышкой и, удерживая кнопку мыши перетащите на рабочую область. Аналогичным образом поступите с пунктом «Simulation». После выполнения данных действий будут добавлены панели для создания сети Петри (Create) и для управления процессом моделирования (Sim), а пользовательский интерфейс программы примет вид, изображенный на рисунке 4.

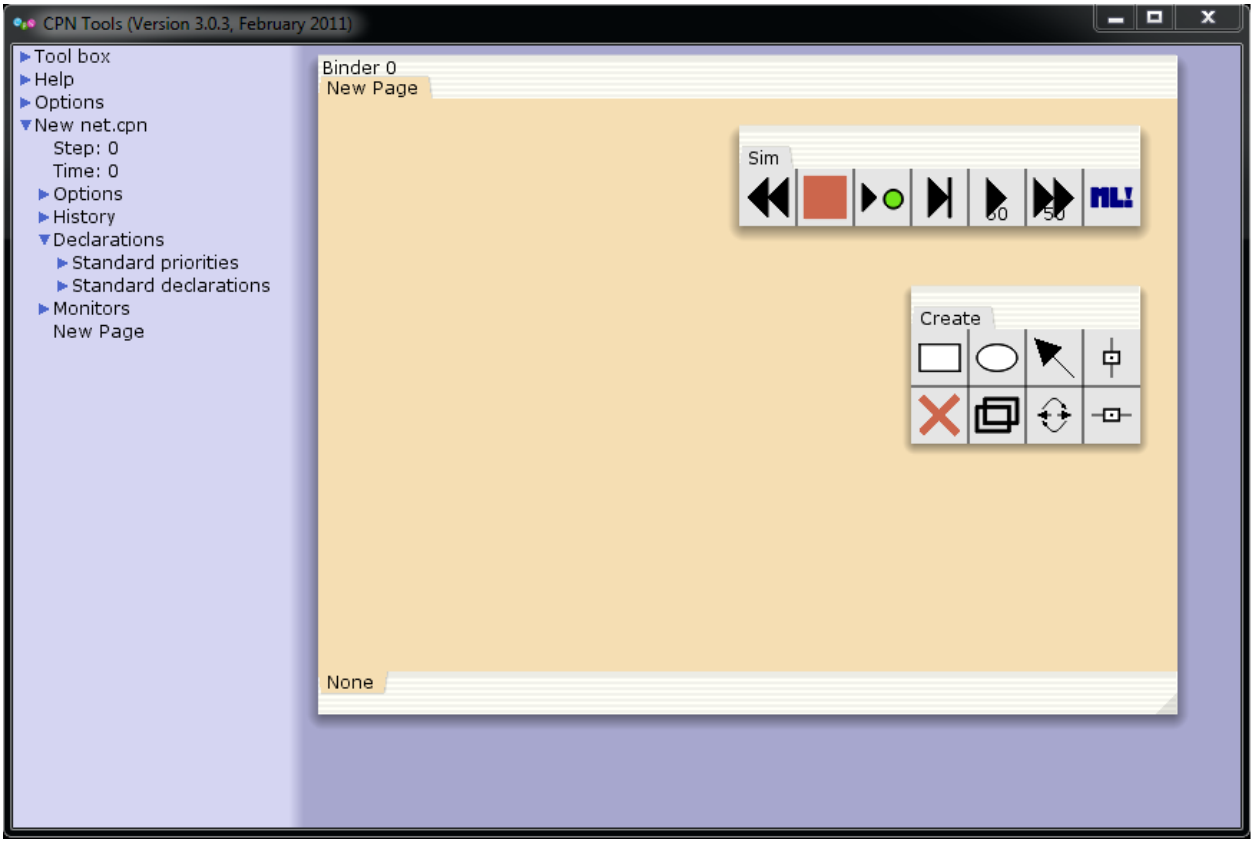


Рис. 4. Внешний вид программы CPN Tools после добавления новых панелей управления

Инструменты панели Create с их подробным описанием представлены в таблице 1.

Инструменты панели Simulation с их подробным описанием представлены в таблице 2.

Таблица 1

Инструменты панели Create

Изображение инструмента	Описание инструмента
	Создание перехода. Для выполнения необходимо кликнуть на инструменте, а затем на модели в тех местах, где необходимо создать переходы.
	Создание позиции. Для выполнения необходимо кликнуть на инструменте, а затем на модели в тех местах, где необходимо создать позиции.
	Создание дуги. Для выполнения необходимо кликнуть на инструменте, а затем на элементах модели, которые нужно соединить. Соединить можно только позицию с переходом. При этом дуга будет направлена от первого выбранного элемента к второму.
	Создание вертикальной линии. Для выполнения необходимо кликнуть на инструменте, а затем на модели. Вертикальная и горизонтальная линии предназначены для организации модели. Элементы, находящиеся вблизи линии, притягиваются к ней. С помощью данного инструмента можно улучшить восприятие модели.
	Удаление элемента. Для удаления необходимо кликнуть на инструменте, а затем на элементах модели, которые нужно удалить.
	Клонирование элемента. Для клонирования необходимо выбранным инструментом кликнуть на элементе, который должен быть клонирован. После этого курсор примет вид выбранного элемента. Затем, кликая на модели, мы каждый раз будем создавать копию элемента. Клонированные элементы не переименовываются. Это необходимо сделать самостоятельно.
	Изменение направления дуги. Для выполнения необходимо выбранным инструментом кликнуть на соответствующей дуге. Дуга может быть как односторонняя, так и двусторонняя. Двусторонняя дуга применяется для экономии места модели и представляет собой аналог двух дуг, направленных в противоположном направлении и имеющих одинаковые выражения.
	Создание горизонтальной линии. Аналогично созданию вертикальной линии.

Таблица 2

Инструменты панели Simulation

Изображение инструмента	Описание инструмента
	Перевод сети в начальное состояние. При этом всем переходам устанавливается их начальная маркировка. Для выполнения необходимо выбранным инструментом кликнуть на модели.
	Остановка выполнения модели. Для выполнения необходимо выбранным инструментом кликнуть на модели.
	Выполнение активного перехода с заданием конкретной маркировки. Подробнее данный инструмент рассмотрен далее.
	Выполнение одного шага моделирования, одного перехода. Для выполнения необходимо выбранным инструментом кликнуть на модели. При этом произойдет срабатывание одного из активных переходов. При клике на активный переход произойдет его срабатывание.
	Выполнение заданного числа шагов с определенным временным интервалом между ними и визуальным изменением маркировки сети. Для выполнения необходимо выбранным инструментом кликнуть на модели. Процесс моделирования может быть остановлен.
	Выполнение заданного числа шагов, без демонстрации промежуточной маркировки. Для выполнения необходимо выбранным инструментом кликнуть на модели.
	Проверка выделенного текста на правильность с точки зрения синтаксиса встроенного языка программирования CPN ML. Для выполнения необходимо выбранным инструментом кликнуть на тексте, содержащем код CPN ML.

Для задания количества шагов моделирования необходимо кликнуть правой кнопкой мыши на соответствующем инструменте и в появившемся контекстном меню выбрать пункт «Set options». Для инструмента моделирования с визуализацией промежуточных результатов, аналогичным способом можно задать время между шагами моделирования.

Построение сети

Рассмотрим процесс создания моделей в среде CPN Tools, построим простую модель (рис. 5). Для добавления в сеть позиций и переходов необходимо кликнуть левой клавишей мыши на соответствующем элементе панели Create, а затем кликнуть на модели для добавления элементов сети. Для изменения имени позиции или переходе необходимо кликнуть на элементе левой клавишей мыши и ввести имя.

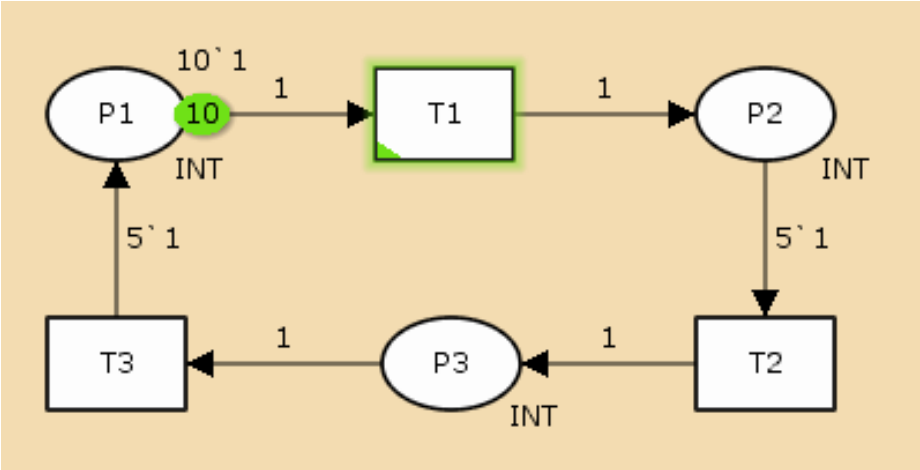


Рис. 5. Пример простой сети Петри в CPN Tools

Для каждой позиции необходимо задать тип и начальную маркировку. Задание имени не обязательно, однако при задании имени необходимо следить за его уникальностью, не допускается наличие одинаковых имен позиций и переходов.

Для задания типа позиции необходимо кликнуть левой клавишей мыши на соответствующей позиции, нажать клавишу Tab и ввести имя типа. Изначально при создании модели доступны четыре типа позиций: *INT, UNIT, BOOL, STRING*. В качестве примера введем тип *INT*.

Для задания начальной маркировки необходимо кликнуть левой клавишей мыши на соответствующей позиции, нажать клавишу Tab два раза и ввести значение начальной маркировки. Синтаксис выражения, определяющего начальную маркировку, определяется следующим образом: A_1++A_2++...++A_n, где элемент A_i задается, как *num`value*, где *num* – количество меток, *value* – значения меток, в соответствии с типом позиции.

Обратите внимание, что количество меток и их значение разделяются знаком апостроф (клавиша тильда в английской раскладке, слева от единицы, в левом верхнем углу клавиатуры).

Выражение *5`1++10`2* задает начальную маркировку позиции, состоящую из 15 меток, 5 меток со значением 1 и 10 меток со значением 2.

Следующим шагом является задание выражений на дугах. Выражения на дугах предназначены для определения того, какие метки будут извлекаться из входных позиций, а какие будет добавляться к выходным позициям. Выражения на входной и выходной дугах могут быть различными. Выражения на дугах имеют такой же синтаксис, как и выражения для задания начальной маркировки. Например, запись *1`1++1`2* обозначает, что из входной позиции, в случае срабатывания перехода будет извлечена одна метка со значением 1 и 1 метка со значением 2. Аналогично для выходной позиции, при срабатывании перехода в ней будет добавлена одна метка со значением 1 и 1 метка со значением 2.

Для начальной маркировки допускается сокращенная форма записи, например запись *1* аналогично записи *1`1*, а запись *2* аналогична записи *1`2*. В общем виде *value* аналогична *1`value*.

Выполнение сети

После того как мы построили сеть (рис. 5), задали типы позиций, начальную маркировку и выражения на дугах, мы можем начать моделирование сети.

Для начала моделирования необходимо кликнуть левой клавишей мышки на одном из действий вкладки «Simulation», а затем, когда курсор примет вид выбранного действия кликнуть на модели. После этого будет осуществлено выбранное действие.

Обратите внимание, что активные переходы подсвечиваются зеленым цветом. В случае, если не была введена некоторая информация (тип позиции, выражение на дуге), соответствующий элемент или несколько элементов будут подсвечены коричневым цветом. Подсветка элементов желтым цветом означает, что в данный момент выполняется проверка правильности введенных данных для данного элемента. Такая ситуация возникает в момент загрузки модели, либо после изменения типа данных.

Воспользуемся инструментом пошагового выполнения модели и проследим, как меняется маркировка сети после срабатывания активных переходов.

Типы данных, переменные

Модифицируем рассмотренную нами сеть (рис. 5), для этого объявим две переменные *value* и *filter* типа *INT* (рис. 6). Объявление переменных в данном примере будет выглядеть следующим образом: *var value,filter:INT;*

Обратите внимание, что объявление переменных должно следовать за объявлением типа данных переменных, в случае переменных *value* и *filter* за типом *INT*. В противном случае программа выдаст сообщение об ошибке и переменную нельзя будет использовать в модели. Данное правило действует и при объявлении новых типов меток.

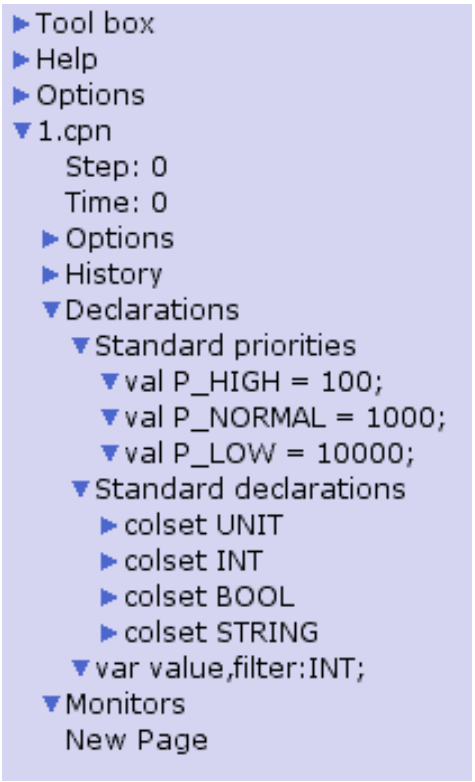


Рис. 6. Объявление переменных

После объявления переменных, добавим новую позицию (*P4*), изменим начальную маркировку позиции *P1*, изменим выражения на дугах, введем переменные и зададим охранное выражение перехода *T2* (рис. 7).

Обратите внимание, что в данном примере в качестве выражений на дугах используются переменные (*value* - имя переменной), а не конкретные числа. В случае срабатывания перехода *T1* из позиции *P1* будет извлечена одна метка со значением 1, 2 или 3. Это значение и будет значением переменной. Выражение *value* используется и на выходной дуге перехода *T1*, это значит, что в позицию *P2* будет помещена метка со значением *value*, которая была извлечена из позиции *P1*.

Если имя переменной используется в выражениях на выходных дугах перехода, то такая переменная должна присутствовать и в выражении на одной из входных дуг. В случае, если это не будет сделано, программа сообщит о том, что в выражении на выходной дуге используется неизвестная переменная.

В случае, если переменная присутствует в выражении на более чем одной входной дуге перехода, программа не сообщит об ошибке, однако переход не будет активирован, даже при наличии меток во входных позициях.

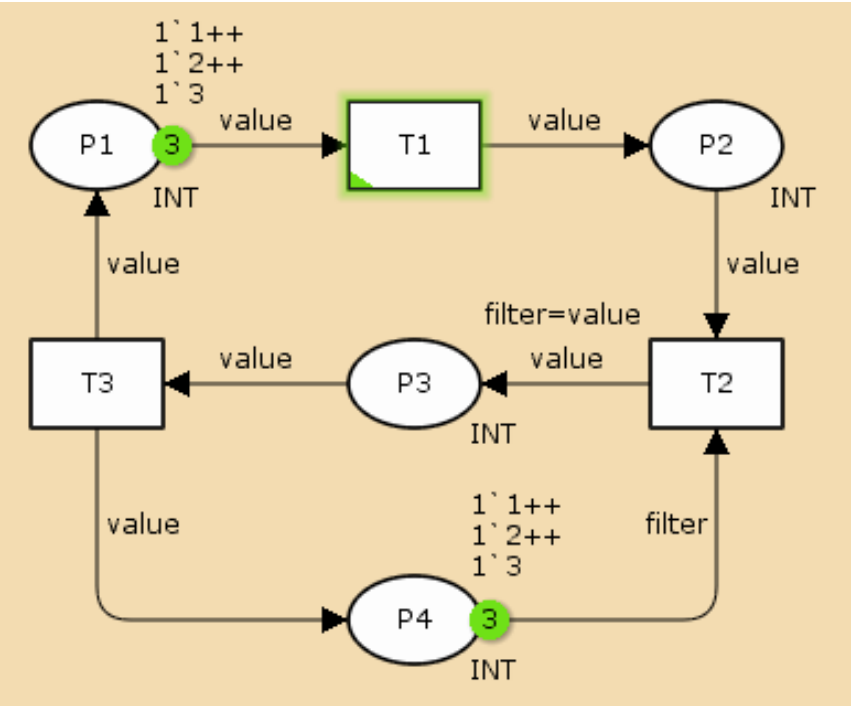


Рис. 7. Пример сети в CPN Tools с использованием переменных и охранных выражений

Для объявления переменных необходимо выполнить следующую последовательность действий:

В пункте меню вашей модели (New net.cpn, если она еще не была сохранена) кликнуть левой клавишей мышки на пункте Declarations. В открывшемся списке вы увидите два элемента (блока):

- Standard priorities, содержит определения стандартных приоритетов, *val P_HIGH = 100; val P_NORMAL = 1000; val P_LOW = 10000*;
- Standard declarations, содержит определение стандартных типов (цветов) *colset UNIT = unit; colset INT = int; colset BOOL = bool; colset STRING = string*;

Свои переменные и типы данных лучше создавать в отдельных блоках. Это позволит улучшить восприятие модели. Для создания нового блока кликните правой кнопкой мышки на элементе Declarations и в появившемся меню выберите пункт «New Block». Введите имя блока, а затем, кликнув на нем мышкой, перетащите вниз списка. Это необходимо, т.к. объявление переменных должно идти после объявления типов.

Синтаксис объявления переменных выглядит следующим образом:

var id1, id2, …, idn : cs_name ,

где *id1 … idn* – идентификаторы переменной, *cs_name* – имя типа (цвета), объявленного ранее.

Идентификатор переменной может состоять из букв, цифр, знака подчеркивания и апострофа ('). Начинаться идентификатор должен с буквы.

Охранные выражения

Охранное выражение – это выражение, которое определяет условие срабатывания перехода. Другими словами, переход может сработать только в том случае, если в его входных позициях найдутся метки с такими значениями, что охранное выражение примет значение истины. В охранном выражении может использоваться переменный из выражений на входных дугах перехода.

Для задания охранного выражения перехода необходимо кликнуть на нем мышкой, нажать клавишу Tab (при этом в левом верхнем углу перехода появится текстовое поли доступное для изменения) и ввести охранное выражение.

Синтаксис охранного выражения имеет следующий вид:

Bool-exp1 Operation Bool-exp2 Operation … Operation Bool-expn,

где *Bool-exp* – логическое выражение вида *переменная оператор переменная*, оператор: равно (=), больше (>), меньше (<), не равно (<>);

Operation – логическая связка, и – *andalso*, или – *orelse*.

В данном примере введем охранное выражение *filter=value* переходу *T2*, оно означает, что переход T2 может сработать в том случае, если в позициях *P2* и *P4* найдутся метки с одинаковыми значениями. При срабатывании перехода эти метки будет извлечены из соответствующих входных позиций.

Отладка моделей

При построении больших моделей важным этапом являются отладки и проверка правильности функционирования. CPN Tools имеет несколько инструментов для отладки моделей. Во первых может быть использован инструмент пошагового выполнения на вкладки «Simulation». При пошаговом моделировании происходит срабатывание одного из активных переходов и изменение маркировки.

Вторым инструментом отладки моделей является выполнение активных переходов с заданной маркировкой. Для этого необходимо выбрать соответствующее действие на вкладе «Simulation», кликнуть на один из активных переходов и выбрать доступные значения для переменных на входных дугах перехода, как показано на рисунке 8. В данном примере активным переходом является *T2* с двумя входными переменными *i,j*. Для переменной *i* доступные значения 1 и 2, а для *j-й* 3 и 4.

После выбора значений входных переменных они будут показаны в прямоугольнике под переходом. При повторном нажатии на переход он будет активирован.

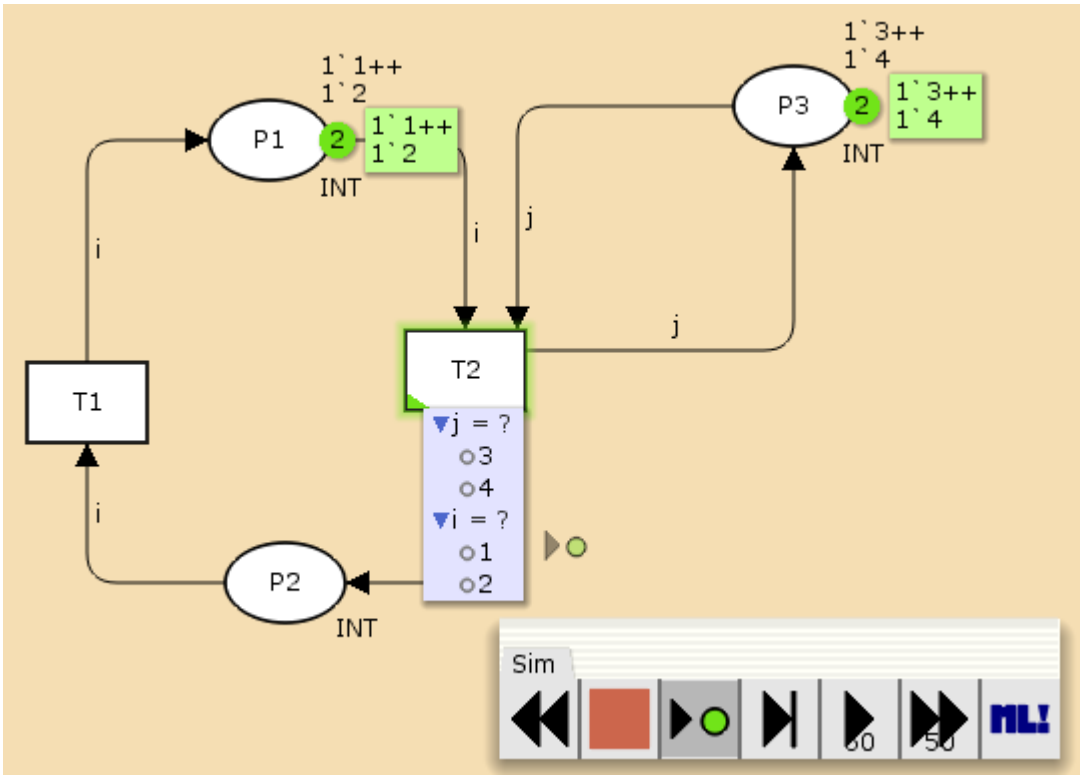


Рис. 8. Выполнение перехода с заданной маркировкой

Для отладки моделей также может быть использовано выполнение нескольких шагов моделирования с отображением промежуточных маркировок. Такой режим выполнения может быть удобен при тестировании большого количества шагов.

Задание для самостоятельной работы

1. Постройте модель (рис. 7) установите маркировку позиций *P1, P4* в соответствии с вариантом задания (таблица 3). Варианту *(n1, n2, n3) (v1, v2, v3)* будет соответствовать маркировка *n1`v1++n2`v2++n3`v3*.
2. Построить сеть, приведенную на рисунке 9. Сеть должна осуществить генерацию *N* меток (всего) типа *INT* со значениями *v1, v2, v3* соответственно. После генерации метки должны быть отсортированы в соответствии со значениями в позициях *P4, P5, P6* соответственно. Варианты задания приведены в таблице 4, в виде *N, v1, v2, v3*.

Таблица 3

Маркировка позиций *P1, P4* в соответствии с номером варианта

Вариант	Маркировка	Вариант	Маркировка
1	(2,4,5) (2,7,6)	8	(3,6,5) (6,2,4)
2	(5,2,4) (4,5,2)	9	(5,9,2) (3,5,1)
3	(7,3,1) (5,2,6)	10	(7,2,5) (6,3,5)
4	(3,8,1) (2,1,4)	11	(6,9,2) (5,6,3)
5	(3,2,7) (9,1,5)	12	(8,8,4) (5,7,2)
6	(7,1,4) (5,4,5)	13	(4,1,8) (6,8,3)
7	(2,5,8) (9,5,8)	14	(9,3,5) (9,4,2)

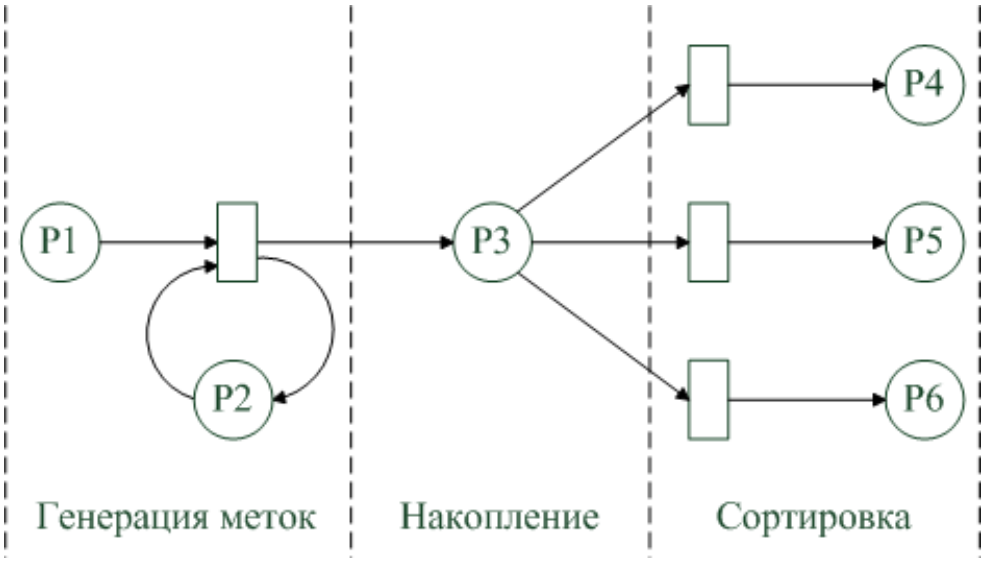


Рис. 9. Схема сети для задания номер 2

Таблица 4

Количество и значения меток в соответствии с номером варианта

Вариант	Задание	Вариант	Задание
1	1500, 3, 4, 1	8	1350, 2, 5, 6
2	1250, 5, 7, 2	9	1650, 6, 4, 7
3	1300, 6, 2, 8	10	1600, 7, 5, 8
4	1550, 8, 6, 9	11	1400, 8, 4, 3
5	1700, 2, 4, 5	12	1550, 1, 2, 3
6	1450, 5, 6, 7	13	1700, 5, 8, 2
7	1400, 6, 9, 8	14	1650, 9, 8, 7

3. Модифицировать сеть из задания 2 таким образом, чтобы можно было генерировать метки с заранее известным процентным распределением по значениям.

Содержание отчета

Цель работы, задание в соответствии с вариантом.

Содержание отчета по заданию 1.

1. Скриншот окна программы CPN Tools с построенной моделью, на скриншоте кроме самой модели должно быть объявление переменных.

2. Маркировка всех позиций после каждого шага моделирования в виде таблицы с тремя графами 1 – номер шага и имя сработавшего перехода, 2 – название позиции, 3 – маркировка позиции. В таблицу внести данные по пяти шагам моделирования.

Содержание отчета по заданию 2.

1. Скриншот окна программы CPN Tools с построенной моделью, на скриншоте кроме самой модели должно быть объявление переменных.

2. Модель необходимо выполнить не менее пяти раз. По результатам каждого процесса моделирования необходимо привести маркировку позиций *P4, P5, P6*, убедиться, что метки отсортированы правильно. Для *P4, P5, P6* привести количество меток в каждой позиции и их суммарное количество. Убедиться в том, что суммарное количество меток равно количеству *N* задания. Рассчитать процентное распределение меток по позициям *P4, P5, P6*.

3. Посчитать общее количество меток в позициях *P4, P5, P6* за все эксперименты, рассчитать процентное соотношение количества меток.

Содержание отчета по заданию 3 аналогично содержанию отчета по заданию 2.

В заключении работы должны быть сформулированы выводы, отражающие содержание проделанной работы и полученные результаты.

Контрольные вопросы

1. Что такое охранное выражение перехода? Каким образом осуществляется его задание?

2. Что такое выражение на дуге?

3. Каким образом осуществляется объявление переменных?

4. Использование переменных в выражениях на входных и выходных дугах перехода.

5. Какие типы данных (цвета) существуют в CPN Tools по умолчанию?

6. Что такое приоритет срабатывания перехода, каким образом он меняет процедуру срабатывания?

Лабораторная работа №2
Изучение возможностей CPN Tools, встроенные функции, массивы

В данной лабораторной работе рассматриваются возможности CPN Tools по работы со списками, пример объявления типа списка, пример модели с добавлением и удалением элементов из очереди, приводится перечень встроенных функций CPN ML для работы со списками с их кратким описанием. Рассматривается пример работы со сложными структурами данных, доступ к полям структуры и выражения на выходных дугах для генерации меток типа *record*.

Работа со списками

При разработке модели реального объект, устройства или информационной системы может потребоваться возможность сохранения запросов пользователи или пакетов данных в очереди, пока они будут ожидать освобождения обрабатывающего устройства. Для организации очередей в CPN Tools предусмотрен специальный тип данных:

colset LIST_NAME = list EL_CS_NAME [with INT-EXP1..INT-EXP2],

где *LIST_NAME* - имя нового типа данных (списка элементов заданного типа), *EL_CS_NAME* - тип элементов из которых будет состоять список, *INT-EXP1* - целочисленное выражение, задающее минимальное количество элементов в списке, *INT-EXP2* - целочисленное выражение, задающее максимальное количество элементов в списке. Часть, заключенная в квадратные скобки, не является обязательной. Тип списка, состоящего из логических переменных, может быть объявлен следующим образом:

colset BoolList = list BOOL.

Чтобы задать содержимое списка, необходимо перечислить элементы списка в квадратных скобках *[element_1, element_2, ... , element_n]*, для задания пустого списка можно записать квадратные скобки без элементов *[]*.

CPN Tools поддерживает ряд функций для работы со списками. В таблице 5 приведены некоторые из этих функций.

Таблица 5

Функции CPN Tools для работы со списками

Функция со списком параметров	Описание функции
nil	Задание пустого списка, аналогично записи *[]*. Данная запись может быть использована для задания начальных маркировок позициям соответствующего типа
e::l	Удаление первого (головного) элемента списка *l* и запись его в переменную *e*
hd l	Получение первого элемента списка *l*
tl l	Получение списка *l* после удаления головного элемента
length l	Получение длины списка *l*
rev l	Запись списка *l* с обратном порядке
map f l	Применение функции *f* к элементам списка *l*. Результатом будет список, заполненный возвращенными значениями функций *f*
List.nth(l,n)	Получение элемента списка *l* с индексом *n*, при этом *0 <= n < length l*
List.take(l,n)	Получение списка из *n* первых элементов списка *l*
List.drop(l,n)	Получение списка, получившегося после отбрасывания *n* первых элементов списка *l*
List.null l	Проверки списка на наличие элементов. Возвращает *true,* если список пуст, в противном случае возвращает *false*
Дополнительные функции для работы со списками	
l1^^l2	Сцепление списка *l1* и *l2*. Результирующий список содержит последовательно элементы *l1* и *l2*
mem l x	Проверка наличия элемента *x* в списке *l*. Возвращает *true,* если элемент *x* присутствует в списке *l*, в противном случае возвращает *false*
remdupl l	Удаление повторяющихся элементов из списка *l*
rm x l	Удаление первого элемента *x* из списка *l*
rmall x l	Удаление всех элементов *x* из списка *l*
intersect l1 l2	Получение списка, содержащего элементы, входящие в пересечение списка *l1* и *l2*
ins l x	Вставка элемента *x* в конец списка *l*
ins_new l x	Вставка элемента *x* в конец списка *l* при условии, что список *l* не содержит *x*
sort lt_fun l	Сортировка списка *l* при помощи функции *lt_fun*, которая используется для определения наименьшего из двух элементов. Для всех типов может быть использована стандартная функция *cs.lt*, где *cs* это имя типа, например *INT.lt*

Пример организации очереди приведен на рисунке 10. В данном примере введен новый тип для представления списка целых чисел INT_LIST и объявлены две переменные.

colset INT_LIST = list INT;

var i:INT;

var il:INT_LIST;

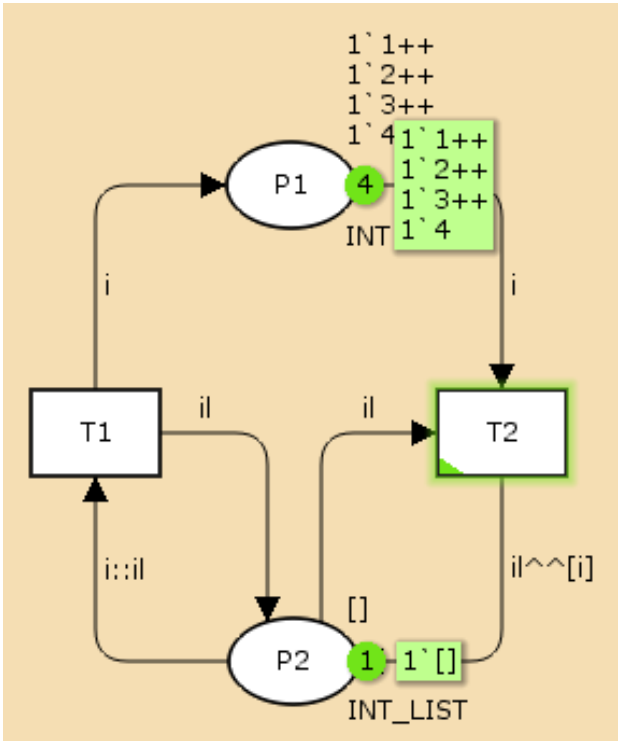


Рис. 10. Реализация очереди в CPN Tools

P1 – позиция, содержащая начальный набор меток *1`1++1`2++1`3++1`4*; *P2* – позиция, представляющая очередь, начальной маркировкой данной позиции является пустой список *[]*; *T2* – переход, осуществляющий добавление элемента *i* в конец очереди *il*, *il^^[i]*; *T1* – переход, осуществляющий извлечение из очереди головного элемента *i::il*.

На рисунке 11 представлен пример использования функций для работы со списками. Функции для работы со списками заданы на выходных дугах переходов *IsNull*, *Reverse* и *Sort*. На входных дугах данных переходов задана переменная *il* типа *INT_LIST*. *IsNull* - проверяет отсутствие элементов в списке *il*, *Reverse* - инвертирует элементы в списке *il*, *Sort* - сортирует элементы списка *il* по возрастанию.

При работе со списками необходимо обращать внимание на возвращаемое значение функции. Оно должно соответствовать типу позиции, для которой дуга с выражением является входной.

Тип данных record

При построении модели обработки запросов может возникнуть необходимость представления запроса не одной меткой типа *INT*, а некоторой более сложной структурой, содержащей множество полей различных типов. Для представления таких структур в CPN Tools предусмотрен специальный тип данных *record*. Объявление такого типа выглядит следующим образом:

*colset CS_NAME = record ID1:CS1 * ID2:CS2 * ... * IDN:CSN,*

где *CS_NAME* - имя нового типа, *ID1, ID2,..., IDN* - имена полей структуры, *CS1, CS2,..., CSN* - типы соответствующих полей структуры. Таким образом объявление структуры, содержащей три поля типа *INT,* будет выглядеть следующим образом:

*colset INT_REC=record i1:INT*i2:INT*i3:INT.*

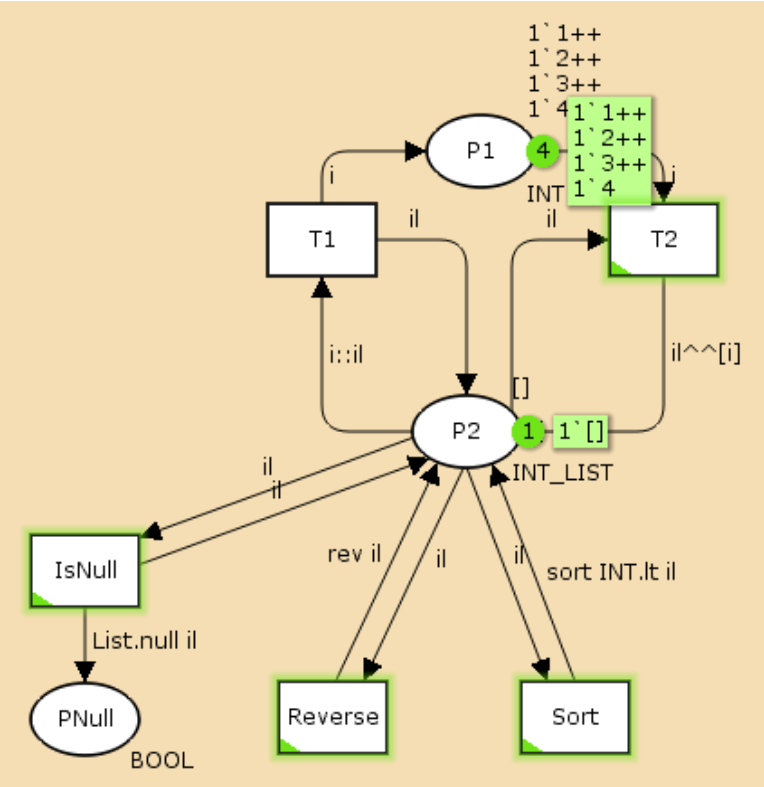


Рис. 11. Применение функций для работы со списками

Для задания меток типа *record* может быть использована следующая форма записи:

{ID1=VALUE1, ID2=VALUE2, ..., IDN=VALUEN},

где *ID1, ID2, ... , IDN* - имена полей структуры, *VALUE1, VALUE2, ... , VALUEN* - значения соответствующих полей структуры.

Для получения доступа к отдельным полям структуры необходимо использовать следующую запись: *#idi rec*, где *idi* - имя поля структуры, *rec* - имя переменной типа *record*.

Использование типа данных record при моделировании

На рисунке 12 показан пример работы со структурами в CPN Tools. В данном примере осуществляется генерация меток типа record, добавление меток в очередь, извлечение меток из очереди и доступ к полям структуры. Для

этого примера были объявлены два типа данных *INT_REC* - структура из трех целых чисел и *INT_REC_LIST* - список из элементов типа *INT_REC*.

*colset INT_REC = record i1:INT*i2:INT*i3:INT;*

colset INT_REC_LIST = list INT_REC;

var irl:INT_REC_LIST;

var v1,v2,v3:INT;

var ir1:INT_REC;

Позиции *P1, P2, P3* содержат метки целого типа, которые будут использованы для генерации меток типа *INT_REC*. Переход *Gen* осуществляет генерацию меток типа *INT_REC* на основе входных переменных *v1, v2, v3*. Для этого используется выражение *1`{i1=v1,i2=v2,i3=v3}* на выходной дуге перехода *Gen*. *P4* – позиция, осуществляющая накопление меток типа *INT_REC*. *Put* – переход, осуществляющий добавление меток в конец очереди. *Queue* – позиция, моделирующая очередь из элементов *INT_REC*. *Get* - переход, осуществляющий: изъятие головного элемента *ir1* из списка *irl*, добавление одной метки в позицию *Num*, добавление по одной метке в позиции *O1, O2, O3* со значениями полей *i1, i2, i3* структуры *ir1* соответственно. *O1, O2, O3* – позиции, осуществляющие накопление значений полей *i1, i2, i3* соответственно, по результатам моделирования можно будет оценить, какое из значений каждое поле принимало чаще всего. *Num* - позиция, ограничивающая максимальное количество одновременно существующих, сгенерированных меток в модели.

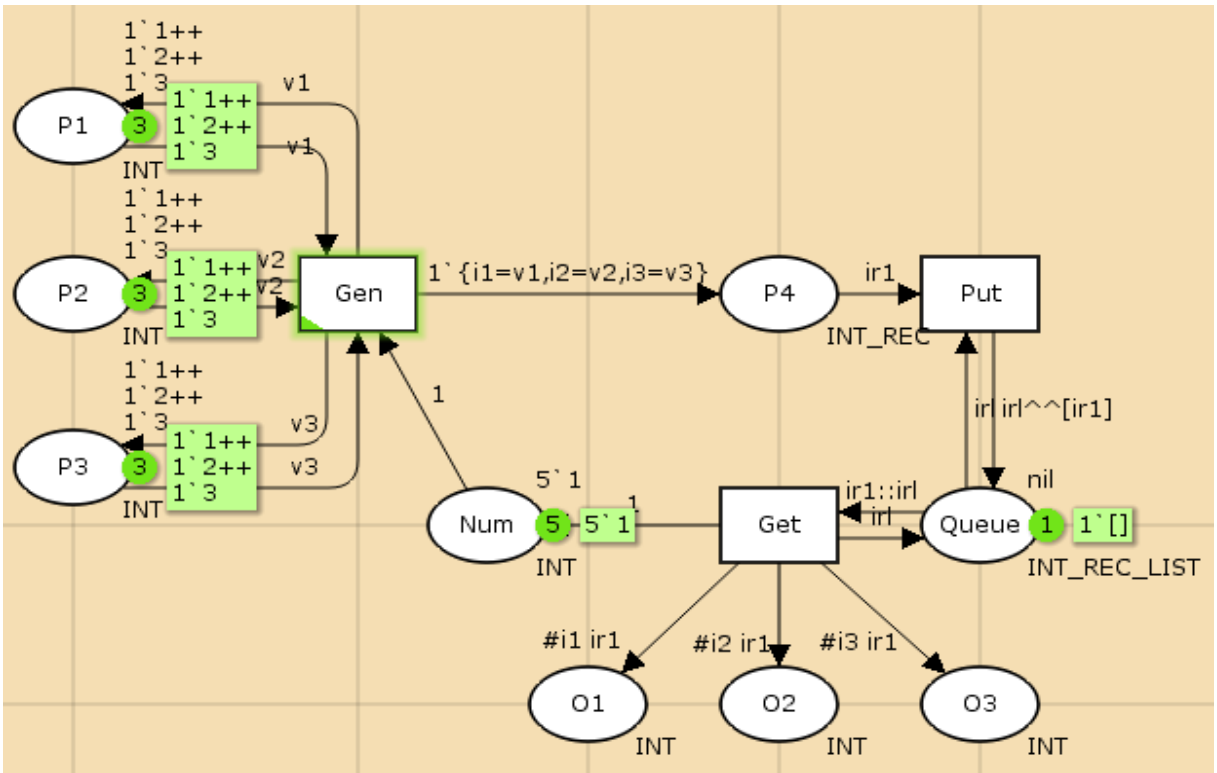


Рис. 12. Работа со структурами

Тип *record* может использоваться для представления информации о запросах пользователя (например имя, номер запроса, приоритет, код запроса и др.), содержания полей пакета данных (например адрес источника и адрес назначения, размер пакета, поле данных, приоритет и др.) или для представления других сложных типов данных в моделях.

Задание для самостоятельной работы

1. Взять за основу пример с очередью (рисунок 10), модифицировать его таким образом, чтобы добавление и извлечение меток соответствовало структуре данных стек. В отчете представить скриншот модели и комментарии относительно внесенных изменений.

2. Разработать модель работы отдела обслуживания клиентов в банке. Отдел обслуживает физических и юридических лиц, в банк могут обращаться клиенты с различными запросами: открытие счета, закрытие счета, взятие кредита, внесение денег на счет, снятие денег со счета. В трех последних случаях клиент совершает операцию с некоторой суммой денег.

Таким образом может быть сформирована структура данных для представления клиента, состоящая из трех полей целого типа: 1 – тип клиента, 2 – тип операции, 3 – сумма денег для операции (если операция это предусматривает, 0 в противном случае).

В отделе предусмотрено две очереди для физических и юридических лиц. Клиенты из каждой очереди обслуживаются отдельным оператором. Каждый оператор подсчитывает количество обслуженных клиентов, количество запросов разного типа и сумму денег по всем обслуженным клиентам.

Один из возможных запросов в данном отделе не обслуживается, соответствующие клиенты направляются в другой отдел. Это может быть смоделировано простым отбрасыванием меток.

Предусмотреть ограничение количества клиентов, одновременно ожидающих запроса QMax.

Предполагаемая структура модели представлена на рисунке 13.

Смоделировать работу отдела по обслуживанию 1000 клиентов. В отчет должна попасть следующая статистика: количество шагов моделирования, количество обслуженных клиентов по типу, количество запросов каждого типа по каждому типу клиентов, количество клиентов которым было отказано в обслуживании, сумма денег по каждому типу клиентов.

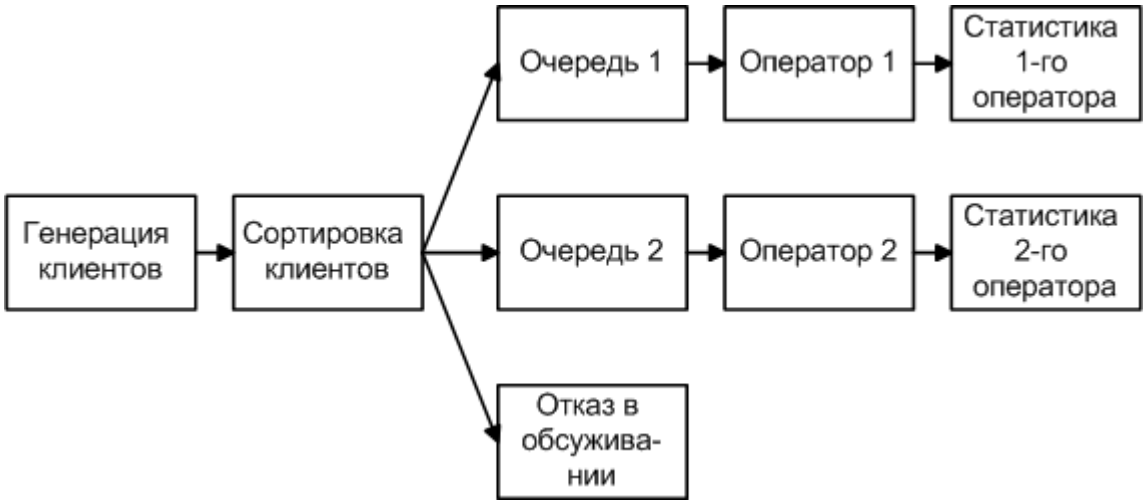


Рис. 13. Структура модели

В отчете представить скриншот модели объявленных типов данных и переменных, использованных в модели.

Содержание отчета

Отчет должен содержать номер варианта и задание, отчет по каждому заданию, заключение, отражающее содержание и результаты выполненной работы, ответы на контрольные вопросы.

Контрольные вопросы

1. Как выглядит определение списка элементов некоторого типа?
2. Какие функции предусмотрены для добавления и извлечения элементов из списка?
3. Как выглядит определение типа структуры, содержащей несколько полей некоторого типа?
4. Как осуществляется доступ к полям структуры?
5. Каким образом осуществляется генерация меток типа *record*?

Лабораторная работа №3. Время в CPN Tools

В данной лабораторной работе рассматриваются возможности программы CPN Tools по моделированию процессов с учетом времени. Рассматриваются способы задания временных значений при моделировании, а также функции для получения текущего модельного времени. Рассматривается связь модельного и реального времени, приводятся примеры моделей с использованием временных меток. Изучаются функции генерации случайных величин с различными законами распределения и применение этих функций при моделировании задержек выполнения. Описывается способ и рассматривается пример хранения статистики о результатах моделирования в позициях сети.

Время в CPN Tools

При построении моделей реальных устройств требуется рассмотрение динамики процесса и определение временных характеристик работы модели. Реальные процессы развиваются во времени, поэтому и математический аппарат, предназначенный для их моделирования, должен иметь возможности представления событий во времени, так как модели внутренней структуры и логики часто бывает недостаточно.

Для моделирования процессов во времени, сети Петри расширяются введением временных меток в маркеры. Временная метка показывает, с какого момента времени маркер будет доступен в позиции. При определении активности переходов во временной сети Петри учитываются только те маркеры, у которых значение времени меньше либо равно текущему времени модели.

Время в моделях CPN Tools является безразмерной величиной и представляется в виде неотрицательного целого числа. Текущее значение модельного времени располагается во вкладке модели (рис. 14), рядом со значением шага моделирования. Если моделирование ведется без использования временных меток, то значение текущего модельного времени не будет меняться при выполнении модели.

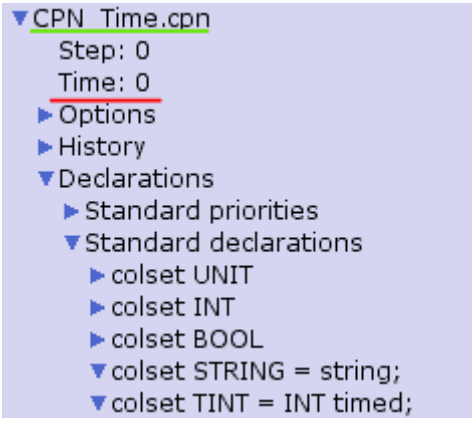


Рис. 14.- Текущее модельное время в интерфейсе CPN Tools

В случае, если на текущем шаге моделирования нет активных переходов с временем, меньшим либо равным текущему, мо-

дельное время увеличивается до тех пор, пока в сети не появится активный переход.

Связь модельного и реального времени

Так как в CPN Tools время представляется в виде целого числа, а в реальности оно непрерывно, необходимо установить взаимосвязь между значениями модельного времени и их реальными эквивалентами. Для этого может быть использовано два подхода:

1. Некоторому интервалу реального времени (например 1 секунда) выбирается количество тактов модельного времени (например 10). В соответствии с этим выбором устанавливаются задержки для всех действий в сети. Если некоторое действие длится 25 секунд, то в модели оно будет длиться *25*10 = 250* тактов модельного времени. Аналогично, зная количество шагов модельного времени, которое занял некоторый процесс, можно вычислить его длительность в секундах, 600 тактов равно *600 / 10 = 60* секунд.
2. Выбирается наименее длительный процесс, который необходимо моделировать, например минимальное время обслуживания простейшего запроса пользователя к информационной системе, равное 1 мс. Этому времени будут соответствовать 10 шагов модельного времени. В соответствии с данным выбором устанавливаются остальные задержки в модели.

При выборе модельного времени следует выбирать соответствие с запасом. Если мы построим модель с расчетом 1 такт равен 1 секунде, то мы не сможем смоделировать интервал времени меньше 1 секунды или чуть больше, например 1.1 секунда. Так как в жизни длительность каждого события может несколько отличаться в разных экспериментах, то и в модели следует это предусмотреть. Для этого можно построить ту же модель с расчетом 10 тактов на одну секунду реального времени.

Использование временных меток в моделях

Перед тем как вводить временные задержки в модель сети Петри в CPN Tools, необходимо определить новые типы данных для временных меток. Формат временных меток выглядит следующим образом:

colset NEW_TYPE = TYPE timed,

где *colset* – ключевое слово для определения типа, *timed* – ключевое слово, показывающее, что маркеры данного типа имеют временные метки, *NEW_TYPE*

– название нового типа, *TYPE* – название типа, который был определен ранее, без добавления временной метки.

В качестве примера рассмотрим тип, определяющий множество целых чисел с временными метками. Данный тип может быть задан следующим образом:

colset TINT = INT timed.

Значение времени в метках может быть изменено с помощью задания выражений на дугах. Для модификации временной метки применяется следующий формат выражения на дуге:

expr@+k,

где *expr* – выражение на дуге без учета времени, *@+k* – запись означающая установку времени в метке, как *текущее + k*. Пример задания временных меток приведен на рисунке 15.

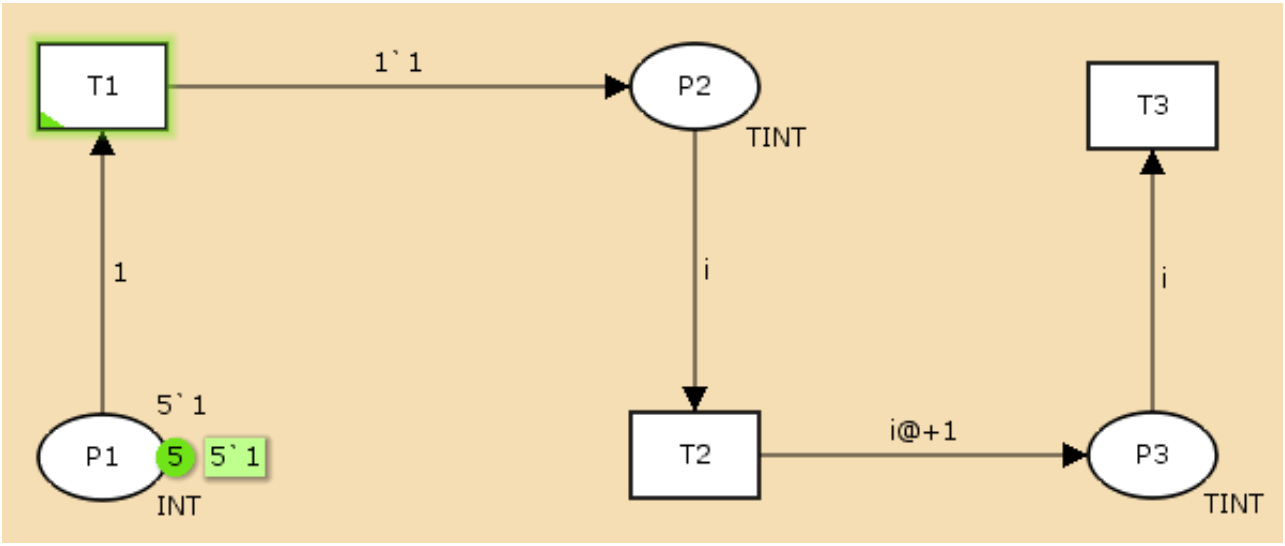


Рис. 15. Пример сети с временными позициями и модификацией временных меток в выражениях на дугах

Временные метки могут быть заданы при срабатывании перехода. Для этого необходимо модифицировать переход следующим образом: выделить переход, нажать клавишу tab дважды и ввести значение вида *@+k*. Пример сети Петри с заданием временных меток при срабатывании перехода приведен на рисунке 16.

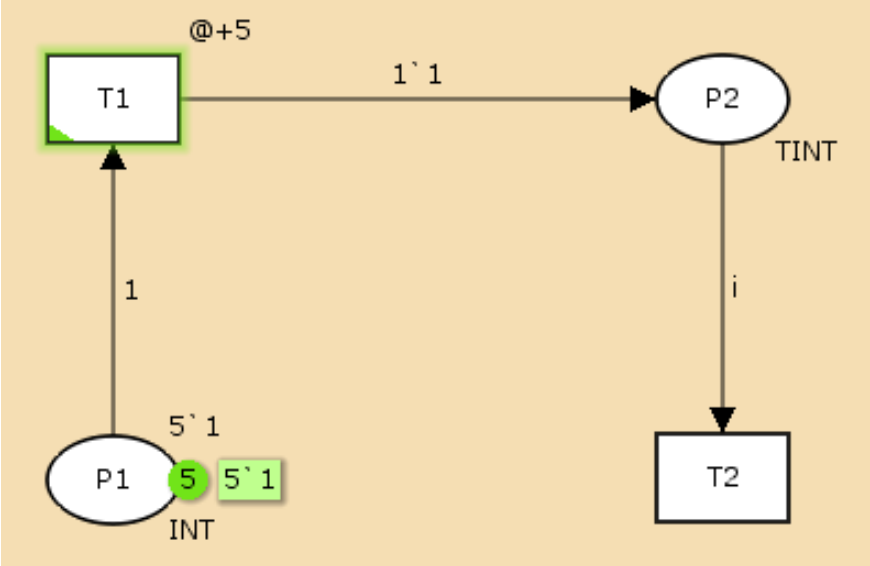


Рис. 16. Задание временных меток при срабатывании переходов

Временная метка будет добавлена ко всем маркерам, которые будут направлены в позиция *P2*, при срабатывании перехода *T1*, однако на работу самого перехода T1 такая модификация не повлияет.

Для задания задержек срабатывания перехода модифицируем сеть Петри на рисунке 16. Для этого добавим одну позицию и зададим выражения на дугах (рисунок 17).

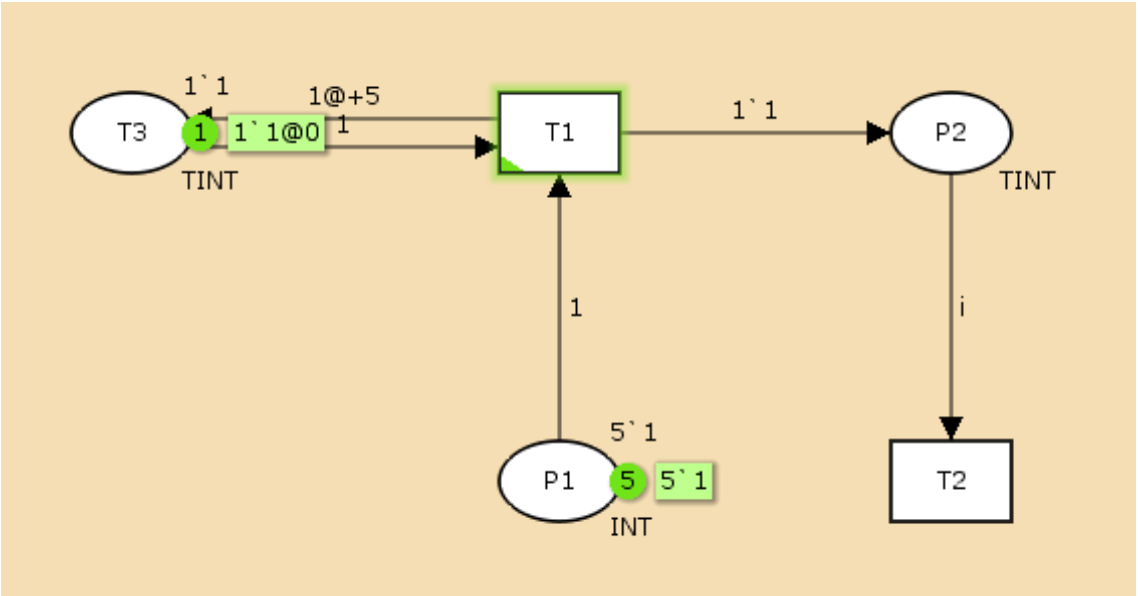


Рис. 17. Задание задержки срабатывания перехода

После модификации сети (рисунок 17) переход *T1* будет срабатывать каждые 5 тактов модельного времени. Так как одна из его входных позиций содержит маркеры с временными метками, которые становятся доступными через 5 тактов модельного времени после срабатывания перехода *T1*. Используя такой подход возможно задавать время работы некоторых участков модели, например время обработки пакетов данных или время обслуживания клиентов.

Получение текущего модельного времени

При построении моделей часто возникает задача получения текущего значения модельного времени. Для этого необходимо воспользоваться следующей функцией:

fun curTime()=IntInf.toInt(!CPN'Time.model_time).

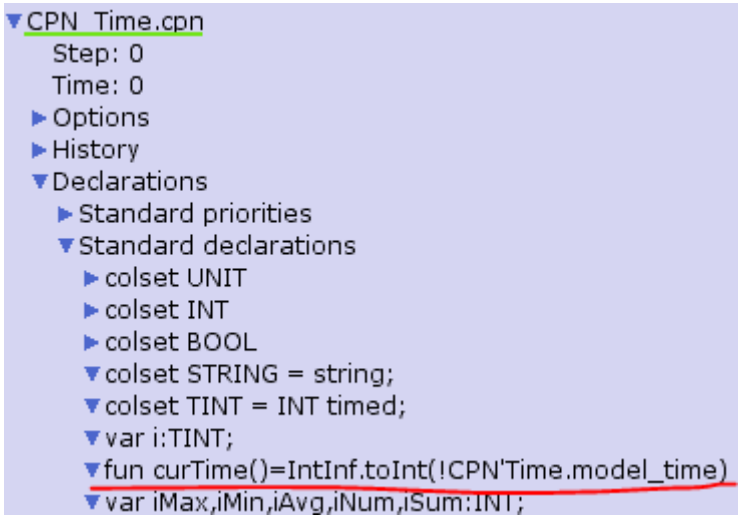


Рис. 18 - Объявление и реализация функции curTime() в CPN Tools

Текст данной функции вводится в блок вместе с объявлениями типов и переменных (рисунок 18).

Пример использования функции *curTime()* для генерации маркеров с текущим значением времени представлен на рисунке 19. Обратите внимание на выражение на выходной дуге перехода *T1*, в данном примере значение времени становится значением маркера, а не временной метки. При обработке таких меток это значение может быть использовано для расчета времени движения метки по сети.

Для работы со временем в CPN Tools есть еще несколько функций. В частности альтернативным способом получения текущего времени может быть вызов функции *time()*, однако результатом данной функции является число в специальном формате *intinf*, предназначенном для хранения больших чисел. Для преобразования этого значения в тип *INT* может быть использована функция *IntInf.toInt(time())* и функция *ModelTime.toString(time())* – для преобразования в тип *STRING*.

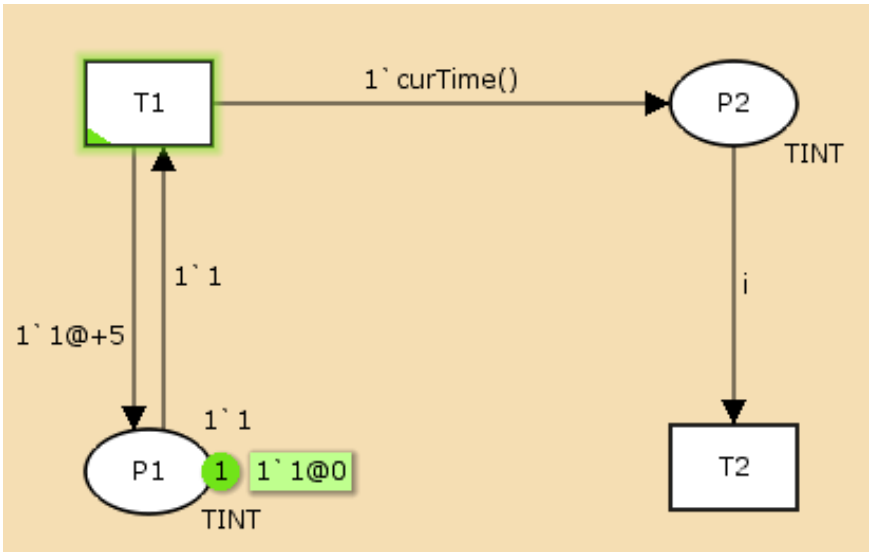


Рис. 19. Использование функции curTime()

Для получения значения текущего шага моделирования предусмотрена функция *step()*. Она также возвращает значение типа *intinf*. Пример использования функций *time()* и *step()* представлен на рисунке 20.

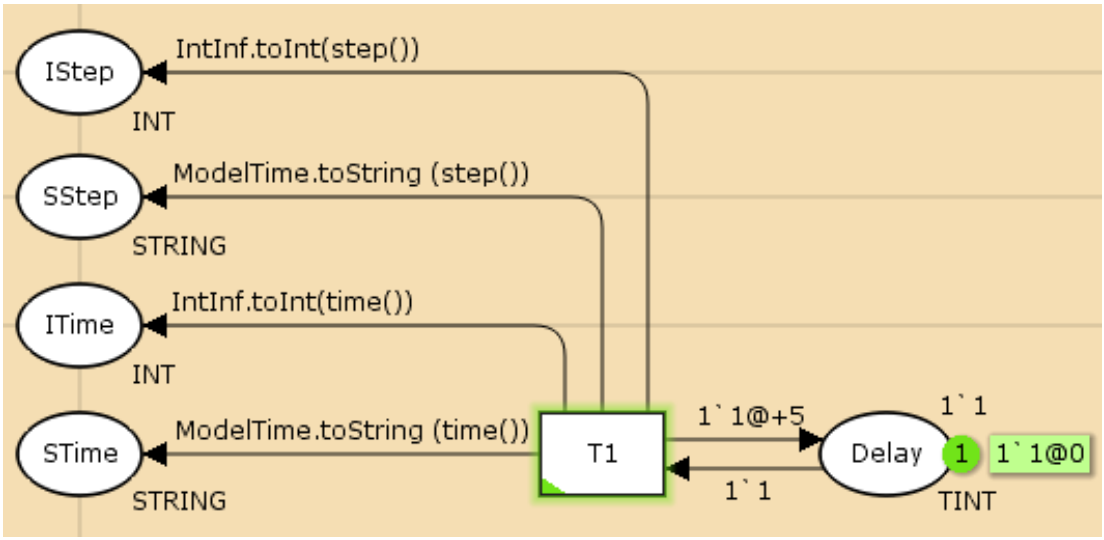


Рис. 20. Пример использования функций time() и step()

T1 – переход, осуществляющий генерацию меток со значением текущего времени и шага моделирования в форматах *INT* и *STRING*. *Delay* – позиция, предназначенная для хранения маркеров с временными метками. *IStep*, *SStep*, *ITime*, *STime* – позиции, предназначенные для хранения меток со значениями шага и времени моделирования в форматах *INT* и *STRING* соответственно.

Генерация случайных величин

При построении модели реальной системы временные интервалы не всегда могут быть представлены в виде определенного числа, вместо этого в моделях время почти всегда задается в виде случайной величины, распределенной по некоторому закону. Для задания случайных величин могут быть использованы следующие функции:

uniform(real min, real max),

где *max* и *min* – максимальное и минимальное значения случайной величины.

normal(real M, real D),

где *M* и *D* – математическое ожидание и дисперсия случайной величины. Математическое ожидание характеризует среднее значение случайной величины, а дисперсия возможное отклонение конкретного значения от математического ожидания.

В помощи по программе CPN Tools вы найдете еще несколько функций для задания случайных величин.

При задании временных задержек могут быть использованы только целые числа, а функции *uniform* и *normal* возвращают действительные. Для преобразования результатов работы функции к приемлемому виду воспользуемся функциями *floor(real v)* и *ceil(real v)*. Функция *floor* округляет число до ближайшего целого в меньшую сторону, а функция *ceil* – в большую.

Пример использования функций генерации случайных чисел и функции *ceil* представлен на рисунке 21.

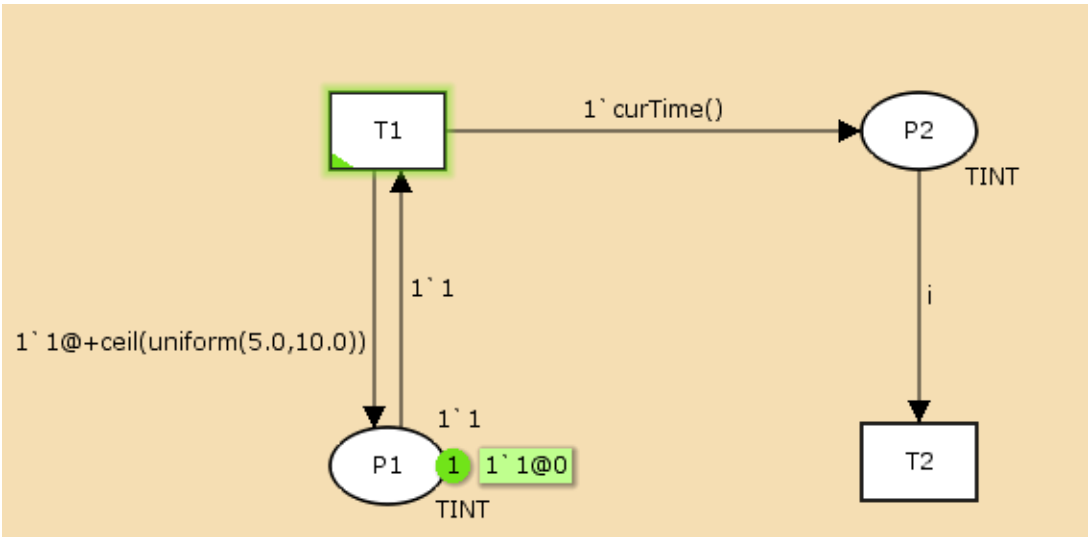


Рис. 21. Пример использования функции генерации случайных чисел

Помимо рассмотренных функций *uniform* и *normal* CPN ML содержит еще несколько функций для моделирования случайных величин с другими законами

распределения. Перечень функций с их подробным описанием и диапазонами допустимых значений для входных параметров представлен в таблице 6.

Таблица 6

Функции CPN ML для генерации случайных величин

Функция со списком параметров	Описание функции
bernoulli(p:real) : int	Распределение Бернулли. Диапазон допустимых значений для параметра *p 0.0<=p<=1.0*. Тип возвращаемого значения целое число, указан в объявлении функции после двоеточия. В этой и остальных функциях, в случае выхода значений параметра за пределы допустимых границ будет сгенерировано исключение и моделирование прекратится
binomial(n:int, p:real) : int	Биноминальное распределение. Диапазон значений параметров: *n>=1* и *0.0<=p<=1.0*
chisq(n:int) : real	Распределение кси-квадрат. Диапазон значений параметра *n>=1*
discrete (a:int, b:int) : int	Дискретное равномерное распределение. Параметр *a* меньше либо равен параметру *b*, *a<=b*
erlang (n:int, r:real) : real	Гамма распределение с целым параметром *n* (распределение Эрланга). Диапазон значений параметров *n>=1* и *r>0.0*
exponential(r:real) : real	Экспоненциальное распределение. Диапазон значений параметра *r>0.0*
normal(n:real, v:real) : real	Нормальное распределение. Диапазон значений параметра *v>=0.0*
poisson(m:real) : int	Распределение Пуассона. Диапазон значений параметра *m>0.0*
student (n:int) : real	Распределение Стьюдента. Диапазон значений параметра *n>=1*
uniform(a:real, b:real) : real	Равномерное распределение. Параметра *a* меньше либо равен параметру *b*, *a<=b*
rayleigh(s:real) : real	Распределение Рэлея. Диапазон значений параметра *s>=0.0*
gamma(l:real, k:real) : real	Гамма распределение. Диапазон значений параметров *l,k>=0.0*
beta(a:real, b:real) : real	Бета распределение. Диапазон значений параметров *a,b>=0.0*

Большинство функций принимают в качестве параметров выражения типа *real*, для работы этих функций необходимо либо явно указать значение параметра с указанием части числа после десятичного разделителя, например *1.0* или *11.34*, либо преобразовать число к типу *real* следующим способом: *real int_value*, где: *int_value* - это выражение типа *INT*.

Для задания значений маркерам на основе возвращаемого значения типа *real* и для передачи целочисленных параметров в функции необходимо округлить результат с помощью функций *floor(real v)* или *ceil(real v)*. Функция *floor(real v)* округляет действительное число до ближайшего целого, меньшего либо равного *v*. Функция *ceil(real v)* округляет действительное число до ближайшего целого, большего либо равного *v*.

Использование в качестве параметра функции выражения с типом, отличным от ожидаемого, приведет к ошибке в модели и к невозможности её выполнения.

Статистика

Для накопления статистики о результатах работы моделей могут быть использованы разные методы. В данной работе будет рассмотрен метод накопления информации в позициях. На рисунке 22 приведен пример сети, в которой осуществляется генерация меток со случайными значениями и подсчет некоторых характеристик множества сгенерированных случайных чисел. *Max* – позиция для хранения максимального значения случайного числа, *Min* – позиция для хранения минимального значения случайного числа, *Avg* – позиция для хранения среднего значения случайного числа, *Num* – вспомогательная позиция для хранения значения количества сгенерированных случайных чисел. *Summ* – вспомогательная позиция для хранения суммы сгенерированных случайных чисел. В данной модели используются следующие переменные *var i,iMax,iMin,iAvg,iNum,iSum:INT;*

Накопление статистики является важным элементом процесса моделирования.

Некоторые характеристики модели могут быть вычислены в реальном времени, в процессе моделирования, для других могут потребоваться дополнительные расчеты в специализированных математических приложениях. Во втором случае результаты моделирования удобно сохранить в файл для последующего анализа. Инструменты, позволяющие сделать это, будут рассмотрены в лабораторных работах 4 и 5.

В данном разделе представлен лишь небольшой перечень характеристик, которые могут быть получены на этапе моделирования. Воспользовавшись встроенными функциями CPN Tools, возможно получить более сложные характеристики, однако это повысит сложность расчетов и увеличит время моделирования.

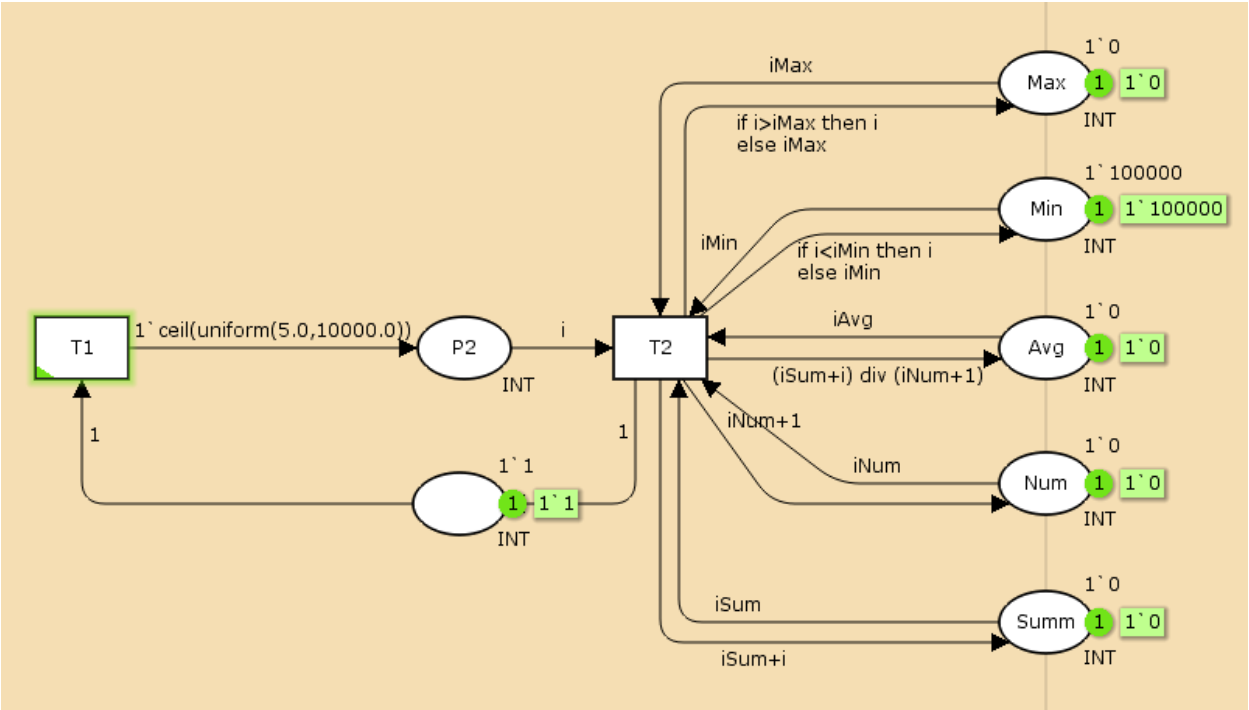


Рис. 22. Накопление статистики

Задание для самостоятельной работы

Построить модель, позволяющую анализировать работу центра технической поддержки. Существует многоканальный телефонный номер, на который поступают звонки пользователей с вопросами. Так как звонков много, некоторым пользователям приходится ждать, пока один из операторов освободится. Пользователи разделены на группы по приоритетам (*N* групп), для каждой группы реализована отдельная очередь.

В центре технической поддержки работают несколько операторов (*M* человек). Свободный оператор должен выбрать пользователя из наиболее приоритетной очереди.

Каждый пользователь обращается в службу технической поддержки с одной из *K* проблем. Каждой проблеме соответствует время, за которое оператор может найти решение *KMin* и *KMax* для равномерного распределения времени ответа и *KN* и *KV* для нормально распределенного времени ответа.

В случае, если количество пользователей, одновременно ожидающих звонка, достигло величины *UMax*, то всем последующим пользователям будет отказано в технической поддержке

Смоделировать работу центра технической поддержки по обслуживанию 1000 клиентов. Задание процентного соотношения количества клиентов в ка-

ждой группе приоритетов осуществляется студентом самостоятельно, при этом число клиентов в каждой группе не должно быть меньше 5% от общего числа. Предполагаемая структура модели представлена на рисунке 23.

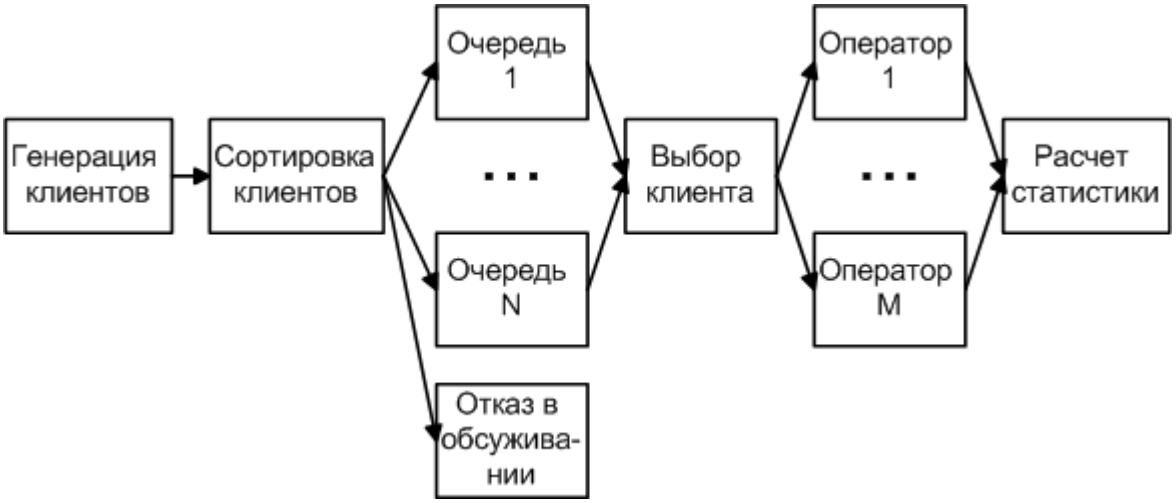


Рис. 23. Предполагаемая структура модели центра технической поддержки

Таблица 7

Значения параметров для построения модели

№	*N*	*M*	*K*	*KMin*	*KMax*	*KN*	*KV*	*UMax*
1	2	6	10	-	-	60	20	50
2	3	5	12	45	80	-	-	35
3	4	4	15	-	-	70	25	45
4	2	6	13	50	90	-	-	55
5	3	5	16	-	-	55	25	40
6	4	4	11	40	85	-	-	50
7	2	6	14	-	-	80	30	35
8	3	5	12	30	90	-	-	50
9	4	4	15	-	-	50	20	45
10	2	6	17	55	75	-	-	50
11	3	5	13	-	-	65	15	50
12	4	4	11	45	85	-	-	55
13	2	6	12	-	-	90	35	45
14	3	5	14	45	70	-	-	35
15	4	4	15	-	-	65	25	40

Смоделировать работу центра технической поддержки в течение 24 часов при тех же параметрах.

Значения параметров *N, M, K, KMin, KMax, KN, KV, UMax* и закон распределения времени ответа оператора в соответствии с номером варианта представлены в таблице 7. Время ответа задано в секундах, студенту необхо-

димо самостоятельно определить соответствующее количество тактов модельного времени.

Содержание отчета

1. Задание на лабораторную работу с выбранным вариантом. В задании указать выбранные значения параметров из таблицы 5.
2. Скриншот модели, определение типов данных и переменных, использованных в модели.
3. В отчете должны быть представлены следующие характеристики, связанные с работой центра технической поддержки (по результатам обслуживания 1000 клиентов):
    а. Среднее время ожидания клиентов в очереди для каждой группы приоритетов;
    б. Общее количество обслуженных клиентов;
    в. Количество обслуженных клиентов по группам приоритета;
    г. Количество клиентов которым было отказано в обслуживании.
4. Результаты аналогичные пункту 3 для моделирования 24 часов работы центра.
5. Ответы на контрольные вопросы.
6. Заключение, отражающее содержание и результаты выполненной работы.

Контрольные вопросы

1. Каким образом осуществляется задание типа данных с учетом временных меток?
2. Каким образом осуществляется установка значений временных меток маркерам в CPN Tools?
3. Каким образом может быть реализована задержка срабатывания перехода?
4. Каким образом осуществляется связь реального и модельного времени?
5. Как изменяется порядок срабатывания переходов с введением временных меток в маркеры?
6. Как осуществляет генерация случайных величин? Приведите некоторые функции, использующиеся для генерации случайных величин с различными законами распределения.

Лабораторная работа №4
Иерархия моделей, работа с файлами, объявление функций

В данной лабораторной работе рассматриваются возможности CPN Tools по построению моделей с вложенными компонентами. Описываются восходящее и нисходящее моделирования, рассматривается пример использования подмоделей. Вводится понятие подстановочных переходов. Рассматриваются альтернативные способы связи частей модели с помощью общих позиций. Приводится пример написания пользовательских функций, пример использования операторов ветвления *if* и *case,* а также пример задания локальных переменных функций. В завершение лабораторной работы рассматривается тип данных *product* (кортеж). Приводится пример работы с кортежами, создания меток типа *product* и доступа к элементам кортежа.

Иерархия моделей

Модели реальных объектов часто достигают больших размеров, что затрудняет их визуальное восприятие и модификацию. Для сохранения наглядности модель может быть структурирована, это можно сделать, выделив главные и подчиненные части модели либо представить модель в виде набора взаимодействующих компонентов. При этом в каждом компоненте (подмодели) будут присутствовать элементы, отвечающие за обмен данными с другими моделями, а также элементы, реализующие основные функции.

Построение моделей с иерархической структурой происходит в три этапа.

1. Построение модели, задающей структуру. В дальнейшем в этой модели некоторые переходы будут заменены на подмодели (рисунок 24). Если этот этап выполняется первым, то такой подход называется нисходящим проектированием. В данном подходе вначале реализуется верхний уровень иерархии моделей. А затем происходит моделирование отдельных подсистем и их функций.
2. Построение подмоделей, моделирующих отдельные функциональные элементы (рисунок 25). Если этот этап выполняется первым, то такой подход называется восходящим проектированием. Реализация модели начинается с построения отдельных функциональных элементов, которые в дальнейшем объединяются в более сложную структуру.

3. Замена переходов на подмодели с помощью инструментов вкладки Hierarchy (рисунок 26). Когда готова модель, представляющая структуру системы, некоторые переходы заменяются на модели составляющих частей.

Построение родительской и вложенной моделей осуществляется на различных страницах. Для добавления страниц к проекту нажмите правой кнопкой мыши на имя вашей модели в области меню. В открывшемся контекстном меню выберите пункт New Page.

На рисунке 24 переход *SN* является подстановочным и содержит вложенную модель.

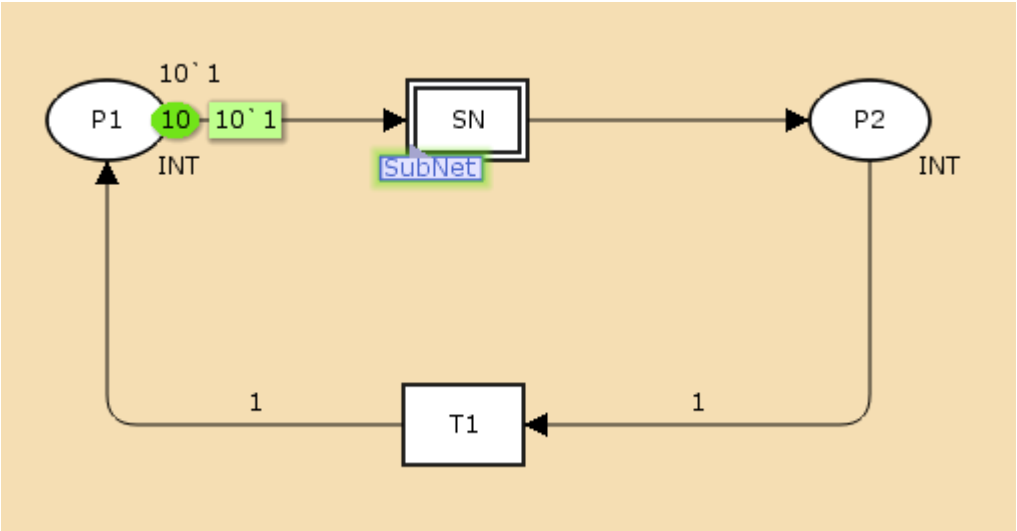


Рис. 24. Модель с вложенной моделью

На рисунке 25 представлена модель подстановочного перехода *SN* (рис. 24). Позиция *In* является входной, ей соответствует позиция *P1* модели верхнего уровня. Позиция *Out* является выходной, ей соответствует позиция *P2*.

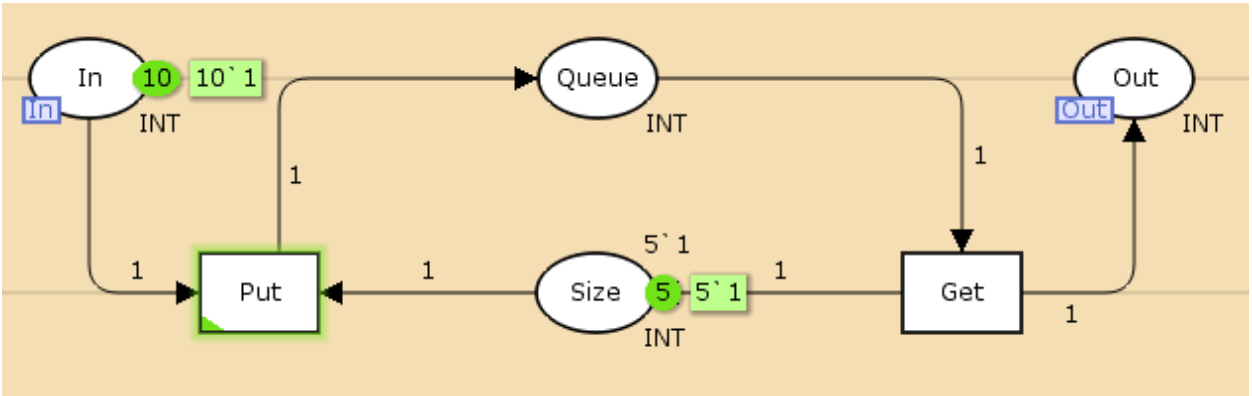


Рис. 25. Вложенная модель

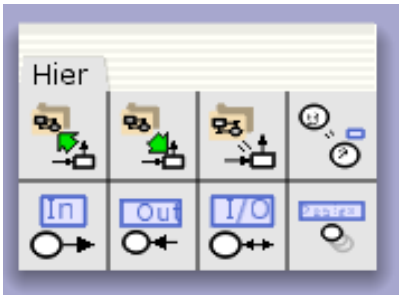


Рис. 26. Инструменты для построения иерархии моделей

Каждая вложенная модель создается на отдельной странице. Для создания новой страницы необходимо кликнуть правой кнопкой мыши на имени модели в области меню и выбрать пункт New Page. После этого к модели будет добавлена новая страница. Новую страницу можно перетащить в рабочую область, при этом будет создано окно с её содержимым, либо перетащить в существующее окно, тогда в окне появится новая вкладка с элементами модели данной страницы.

Подробное описание инструментов вкладки Hierarchy представлено в таблице 8.

Таблица 8

Описание инструментов вкладки Hierarchy

Изображение инструмента	Описание инструмента
	Поместить переход в отдельную подмодель. При этом будет сохранена функциональность модели верхнего уровня и будут автоматически созданы входные и выходные позиции, необходимые для связи подмодели и модели верхнего уровня. Для этого необходимо, выбрав инструмент, кликнуть на соответствующем переходе.
	Заменить подстановочный переход содержимым соответствующей подмодели. При этом все элементы вложенной модели будут вписаны в модель верхнего уровня. Для этого необходимо, выбрав инструмент, кликнуть на соответствующем подстановочном переходе.
	Связать подмодель и подстановочный переход. Для этого необходимо, выбрав инструмент, кликнуть на переходе, который должен стать подстановочным, затем выбрать страницу с подмоделью, которая будет помещена в данный переход, кликнув на имени страницы во вкладке модели в области меню.
	Установить связь между позицией в подмодели и соответствующей ей позиции в модели верхнего уровня. Позиции должны быть одного типа. Также в модели верхнего уровня должна присутствовать дуга, связывающая соединяемую позицию и подстановочный переход, в котором находится соответствующая позиция. Для этого необходимо выбрав инструмент, кликнуть на позиции внутри вложенной модели, после того как произойдет автоматический переход в модель верхнего уровня, кликнуть на соответствующей позиции

Окончание табл. 8

	Сделать позицию входной в данной модели. Для этого необходимо, выделив инструмент, кликнуть на соответствующей позиции.
	Сделать позицию выходной в данной модели. Для этого необходимо, выделив инструмент, кликнуть на соответствующей позиции.
	Сделать позицию входной и выходной одновременно в данной модели. Для этого необходимо, выделив инструмент, кликнуть на соответствующей позиции.
	Сделать позицию общей и поместить её в именованное множество позиций. При изменении маркировки любой из позиций данного множества аналогичные изменения произойдут и в других позициях множества. Для этого необходимо выбрать инструмент, кликнуть на позиции, а затем в появившемся синем прямоугольнике ввести имя множества общих позиций.

Применяя данные инструменты, можно построить модель с требуемой иерархией и связями компонентов.

Построение модели с вложенными подмоделями

Для построения модели с вложенными компонентами можно воспользоваться методологией нисходящего проектирования и выполнить следующие действия:

1. Выполнить разбиение системы на функциональные элементы, выявить иерархию элементов и их связи
2. Построить модель в терминах сети Петри, в которой все компоненты будут представлены переходами, выполнить связь компонентов
3. Разработать модели компонентов. Сделать переходы модели верхнего уровня подстановочными, связав с моделями соответствующих компонентов.

В случае, если компоненты модели также имеют иерархическую структуру или состоят из набора элементов, для них могут быть выполнены описанные шаги.

Связь моделей через множество общих позиций

Для связи отдельных компонентов модели может быть использован альтернативный подход, основанный на задании множества общих позиций. Общие позиции в этом случае выступают в качестве шины данных, соединяю-

щей различные участки модели. Однако строить всю иерархию компонентов только на основе общих позиций не следует, так как такие связи носят неявный характер, их сложно отслеживать, изменять и анализировать.

Наилучшим решением будет комбинированный подход, в котором иерархия компонентов будет задаваться с помощью подстановочных переходов, а функции управления и анализа будут реализованы с помощью общих позиций. В таком случае общие позиции могут быть вынесены на отдельную страницу и использованы для установки начальных значений параметров модели либо для сбора статистики о ходе моделирования.

Функции

CPN Tools представляет возможности по реализации пользователем собственных функций, необходимых для построения модели. Объявление функции выглядит следующим образом:

fun fun_name (parameters) = expression,

где *fun_name* - имя функции, *parameters* - список параметров функции (может быть пустым), *expression* - выражение, соответствующее возвращаемому значению функции. Вызов данной функции будет выглядеть следующим образом: *fun_name().*

Рассмотрим примеры некоторых функций. Функция *Summ* предназначена для суммирования значений параметров, функция возвращает маркер со значением, равным сумме параметров *a* и *b*.

fun Summ(a,b)=1`(a+b).

Результат функции зависит от значений параметров. Для организации ветвления могут быть применены конструкции *case var_name of* и *if bool_expr then expr1 else expr2.*

```
fun ToString(a)=
case a of
   1=> "One"
 | 2 => "Two"
 | 3 => "Three"
 | _ => "Else"
```

```
fun IfFun(a)=
if a=1  then 1`"One"
else 1`"Not one"
```

Функции *Max2* и *Max3* предназначены для нахождения максимального значения среди параметров.

```
fun Max2(v1, v2)=
if v1>v2 then v1
else v2
```

```
fun Max3(v1, v2, v3)=
let
  val pat1 = Max2(v1,v2)
in
  Max2(pat1,v3)
end;
```

В функции *Max3* применяется конструкция, предназначенная для задания локальных переменных в теле функции:

```
let
  val pat1 = exp1
  val pat2 = exp2
  ...
  val patn = expn
in
  exp
end,
```

где *pat1, pat2, ... , patn* - имена локальных переменных, *exp1, exp2, ... , expn* - выражения, определяющие значения соответствующих локальных переменных, *exp* - выражение, определяющее значение, возвращаемое функцией.

Для улучшения наглядности модели пользовательские функции следует привести отдельно от определений типов и объявлений переменных в специальном блоке (рис. 27).

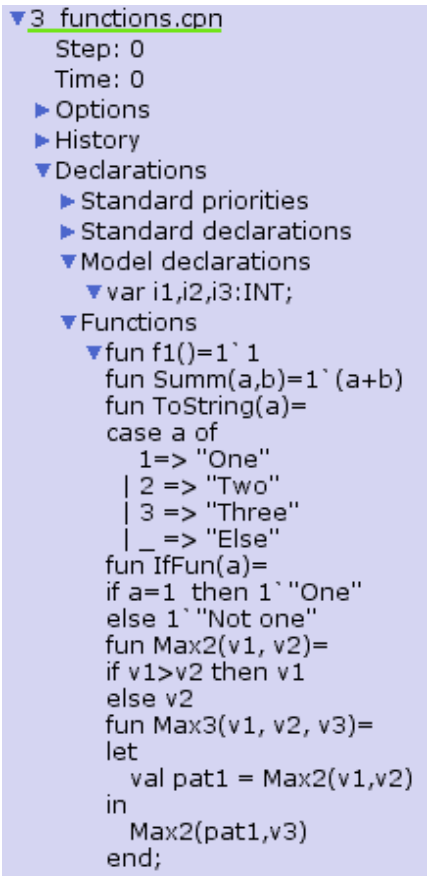


Рис. 27. Объявление пользовательских функций в интерфейсе CPN Tools

Для рассмотрения примера работы с функциями были объявлены следующие переменные:

var i1,i2,i3:INT.

Пример использования пользовательских функций приведен на рисунке 28. Позиции *P1, P2, P3* содержат маркеры, которые используются в качестве значений переменных *i1, i2, i3* на входных дугах перехода *T1*. *Summ* – позиция, содержащая сумму двух переменных *i1* и *i2*, *StringVal* – позиция, содержащая

маркер с результатом функции *ToString*, с параметром *i3*, *Max2*, *Max3* – позиции, содержащие маркеры с наибольшим значением входных параметров функций *Max2(v1, v2)* и *Max3(v1, v2, v3)* соответственно.

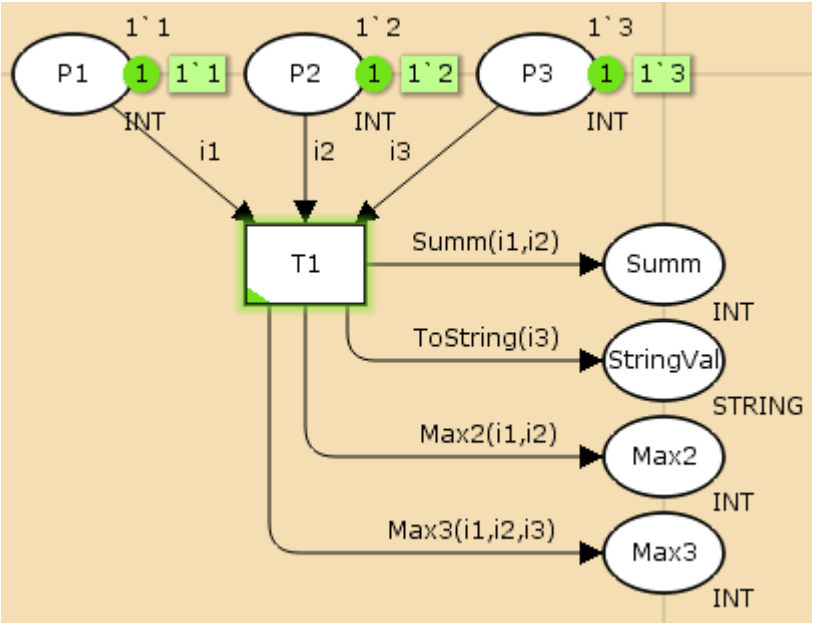


Рис. 28. Пример использования пользовательских функций

Применение функций позволяет сократить количество элементов в модели, повысить наглядность и приблизить модель к предметной области, для которой она строится.

Использование кортежей

При построении сложных моделей часто возникает потребность в задании маркеру массива значений. В качестве альтернативы типу *record* удобно использовать тип *product*, представляющий собой кортеж (tuple) значений различных типов. Синтаксис объявления типа данных кортежей выглядит следующим образом:

*colset CS_NAME = product TYPE_1 * TYPE_2 * ... * TYPE_n;*

n>=2,

где *CS_NAME* - имя нового типа данных, *TYPE_1*, *TYPE_2*, ... , *TYPE_n* - имена типов данных, элементы которых входят в кортеж, *n* - количество элементов кортежа.

Как можно видеть из синтаксиса кортежа, основное его отличие от типа *record* заключается в отсутствии именованных элементов. В кортеже обращение к элементам осуществляется по индексу:

#index product_variable,

где *index* - целое число, индекс адресуемого элемента (индексация начинается с единицы), *product_variable* - имя переменной типа *product.*

Генерация меток типа product в выражениях на дугах осуществляется следующим образом:

1`(v1, v2, …, vn),

где *v1, v2, ... , vn* - значения элементов кортежа, при этом их количество и типы должны соответствовать типам и количеству элементов, присутствующих в определении.

Пример использования типа *product* приведен на рисунке 29. В данном примере использован тип *PTEST* и переменная *pt.*

*colset PTEST = product INT * INT;*

var pt:PTEST;

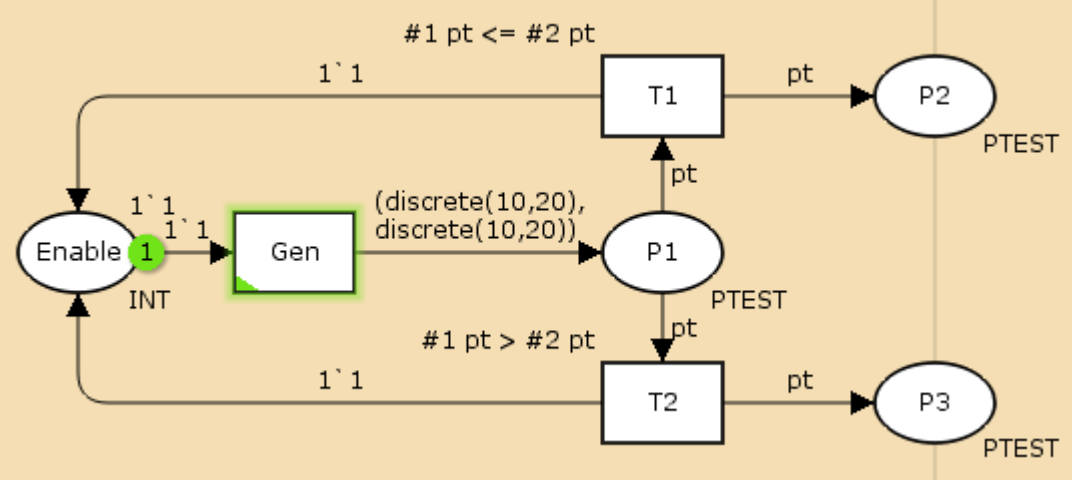


Рис. 29. Пример использования типа product в модели

Gen – переход, осуществляющий генерацию маркеров типа *PTEST*, элементам кортежа присваиваются случайные значения. *P1* – позиция, в которую поступают сгенерированные маркеры. *T1, T2* – переходы, которые осуществляют перемещение маркеров в позиции *P2* и *P3* соответственно. Какой из переходов при этом сработает, определяется на основе охранного выражения. *Enable* – позиция, отвечающая за то, чтобы в каждый шаг модельного времени в позиции *P1* находилось не более одного маркера.

Задание для самостоятельной работы

Построить модель, позволяющую оценивать задержки, возникающие при работе пользователя с web-приложением. Модель должна включать в свой состав следующие компоненты:

- Модель пользователя. Задачей данного компонента является генерация запросов к серверу с web-приложением.
- Модель канала данных. Канал данных может быть смоделирован с помощью внесения постоянной задержки к временной метке проходящих через него маркеров.
- Модель сетевого устройства, через которые проходят пакеты данных с запросами пользователя. Модель сетевого устройства представляет собой очередь, в которой пакеты данных ожидают отправки в сеть.
- Модель сервера, на котором развернуто web-приложение. Сервер web-приложения добавляет случайную задержку к временной метке маркера, моделирующего запрос (пакет данных) и отправляет маркер назад пользователю.

Предполагаемая структура модели представлена на рисунке 30.

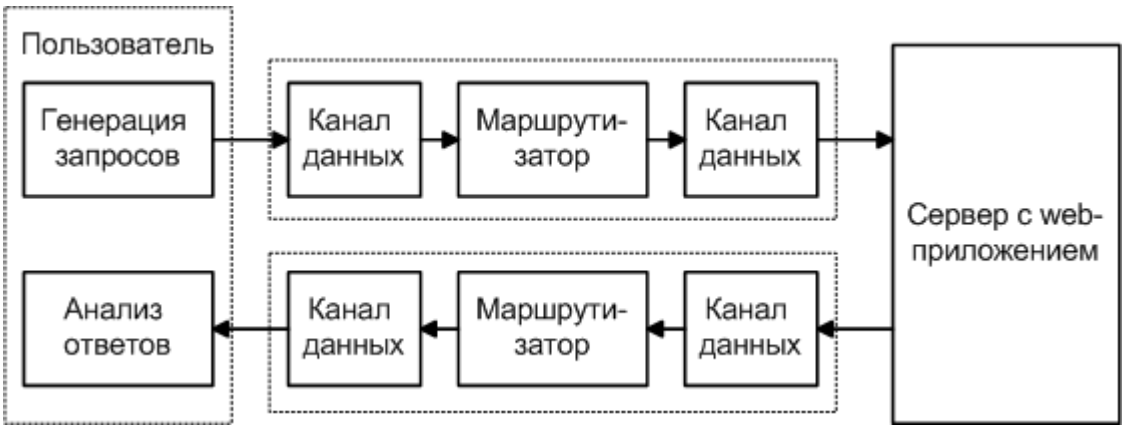


Рис. 30. Рекомендуемая структура модели

Пользовательский запрос и ответ сервера помещаются в один пакет данных. Все пакеты данных имеют одинаковую длину (1500 байт), а соответственно и одинаковое время передачи через канал данных. В маршрутизаторах организована одна очередь ограниченной длины *QMax* для всех входных пакетов. В случае, если во время поступления очередного пакета на входной интерфейс маршрутизатора очередь заполнена полностью, поступающий пакет отбрасывается без уведомления пользователя. Повторной отправки потерянных запросов не происходит.

На пути от пользователя до сервера и обратно насчитывается по одному маршрутизатору, соединенных каналами данных с пропускной способностью 10 мегабит/секунду.

Сервер может одновременно обрабатывать 1 запрос пользователя, при этом *K* запросов будут ожидать в очереди. В случае, если достигнуто предельное число ожидающих запросов, все вновь поступившие запросы отбрасываются.

Интервал между созданием запросов является случайной величиной, равномерно распределенной от *GMin* до *GMax*. Время обработки запросов является случайной величиной, равномерно распределенной от *RMin* до *RMax*.

Провести стрессовое тестирование web-приложения. Необходимо подобрать такие значения задержки генерации пакетов, чтобы запросы отправлялись с максимальной загрузкой канала данных.

Значения параметров *K, GMin, GMax, RMin, RMax, QMax* и закон распределения времени обработки запросов в соответствии с номером варианта представлены в таблице 9. Временные интервалы заданы в миллисекундах, студенту необходимо самостоятельно определить соответствующее количество тактов модельного времени.

Таблица 9

Значения параметров для построения модели

№	*K*	*GMin*	*GMax*	*RMin*	*RMax*	*QMax*
1	50	0.8	2.0	1.0	1.5	100
2	55	1.0	1.5	0.5	1.9	120
3	70	1.1	2.3	0.7	1.8	110
4	45	0.9	2.0	1.0	1.5	150
5	65	0.8	2.3	0.8	2.3	140
6	60	0.7	2.8	0.9	2.6	110
7	50	0.9	2.5	1.1	2.0	130
8	45	0.9	2.2	0.9	2.2	120
9	65	1.0	1.8	1.1	1.7	150
10	75	0.8	2.1	0.9	1.9	140
11	45	1.1	2.7	0.6	3.2	100
12	70	1.0	2.0	0.9	2.1	110
13	60	1.1	2.3	1.1	2.3	130
14	55	0.8	2.5	0.9	2.6	140
15	45	1.0	2.1	0.8	2.7	150

Содержание отчета

1. Задание на лабораторную работу с выбранным вариантом. В задании указать выбранные значения параметров из таблицы 9.
2. Скриншот модели, определение типов данных и переменных, использованных в модели.
3. Для каждого сценария выполнения модели (с параметрами таблицы 7, стрессовое тестирование) привести следующие результаты:
   а. Количество запросов обработанных web-приложением;
   б. Среднее, максимальное, минимальное время обработки запроса, с учетом задержки передачи (передача 1 + обработка + передача 2);
   в. Количество отброшенных пакетов данных маршрутизаторами;
   г. Количество пользовательских запросов отклоненных сервером.
4. Ответы на контрольные вопросы.
5. Заключение отражающее содержание и результаты выполненной работы.

Контрольные вопросы

1. Приведите примеры в которых имеет смысл разбивать модель на компоненты (подмодели).
2. Каким образом в CPN Tools осуществляется построение подмоделей и их связь между собой?
3. Что такое общие позиции и в каких случаях они могут применяться?
4. Что такое подстановочные переходы?
5. Каким образом осуществляется написание пользовательских функций в CPN Tools? Приведите пример функции.
6. Приведите пример функции с использованием операторов ветвления *if then else* и *case*.
7. Каким образом осуществляется задание локальных переменных в функциях?
8. Что такое тип данных *product*? Каким его объявить?
9. Как осуществляется доступ к элементам кортежей? Как создаются метки типа *product*?

Лабораторная работа №5
Вывод результатов моделирования, пространство состояний, работа с файлами

В данной лабораторной работе рассматриваются стандартные средства CPN Tools, предназначенные для вывода информации о ходе моделирования (мониторы). Рассмотрен пример использования мониторов для получения данных о динамике раскрашенной сети Петри, а также пример вывода данных в файл. Рассматривается анализ раскрашенных сетей Петри на основе пространства состояний и средств CPN Tools, предназначенные для его построения.

Стандартные средства вывода информации о процессе моделирования

При моделировании различных процессов возникает необходимость накопления и сохранения статистической информации о процессе моделирования (информация о маркировке, о срабатывании переходов, о созданных метках и другая). Для решения этой задачи в программе CPN Tools предусмотрены специальные инструменты мониторинга выполнения сети (мониторы).

Для использования данных инструментов необходимо открыть панель в области меню «Tool box → Monitoring» (рисунок 31), кликнув на ней левой кнопкой мыши и переместив в рабочую область.

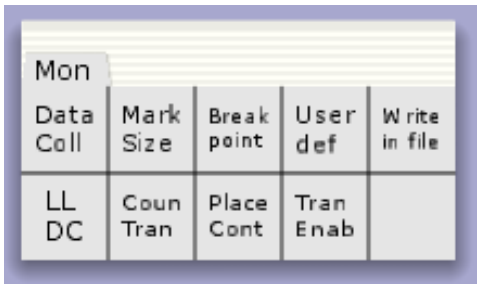


Рис. 31. Внешний вид панели инструментов мониторинга процесса выполнения сети Петри в программе CPN Tools

С помощью панели Monitoring переходам и позициям сети могут быть назначены различные мониторы, с помощью которых может быть зафиксировано изменение различных характеристик во время выполнения. Для назначения мониторов позициям и переходам необходимо присвоить им уникальные имена, при этом не допускается наличие пустых имен.

Для добавления монитора к элементу сети необходимо кликнуть левой кнопкой мыши на названии монитора, а затем на позиции или переходе, которому следует назначить данный монитор. После добавления монитора к элементу сети будет добавлена соответствующая запись в области меню «Назва-

ние_модели → Monitors → Название_монитора». Для удобства мониторы могут быть переименованы.

Рассмотрим некоторые мониторы и их назначение:

Mark Size – Монитор, осуществляющий фиксацию и запись в файл информацию о количестве меток в позиции

Count Tran – Монитор, осуществляющий фиксацию и запись в файл информацию о срабатывании перехода

Break point – Монитор, осуществляющий остановку моделирования при срабатывании перехода

Tran Enab – Монитор, осуществляющий остановку моделирования при достижении активности некоторым переходом

Place Cont – Монитор, осуществляющий остановку моделирования при появлении метки в позиции либо при удалении всех меток в зависимости от флага «If is empty». Флаг устанавливается в области меню в «Название_модели → Monitors → Название_монитора → Type: Place content break point → If is empty». Если данный флаг установлен, то моделирование прекращается при удалении меток из позиции. В противном случае моделирование прекращается при добавлении помещения метки в позицию.

При добавлении мониторов элементам сети необходимо учитывать то, что мониторы, предназначенные для переходов, не могут быть применены к позициям и наоборот.

Добавление мониторов позициям и переходам сети, файлы результатов

После того как мониторы добавлены к позициям и переходам сети, может быть осуществлено моделирование. Результаты работы мониторов при этом будут записаны в файлы с расширением *.log*. Для файлов будет создан отдельный каталог «*output\logfiles*» в каталоге с моделью. В каталоге «*output*» будет располагаться файл «*PerfReport.html*», в котором будут представлены некоторые обобщенные данные по результатам моделирования.

Результаты моделирования записываются только при последовательном моделировании множества шагов без отображения текущей маркировки. При пошаговом моделировании результаты также не записываются. Пример модели, элементам которой были добавлены мониторы, представлен на рисунке 32.

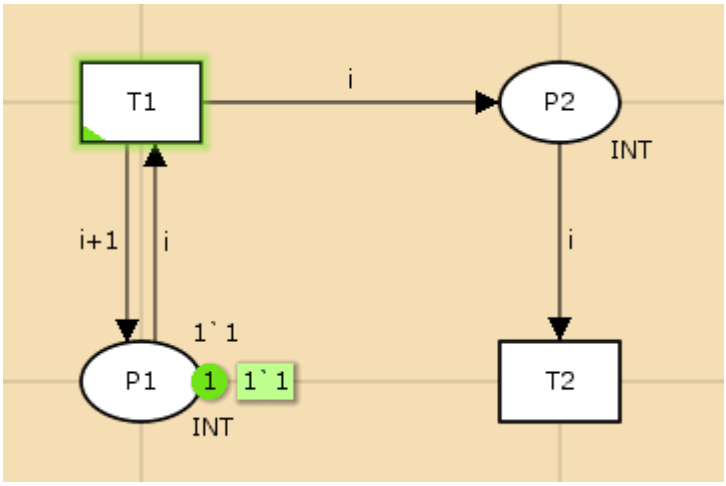


Рис. 32. Пример модели сети Петри с назначенными мониторами

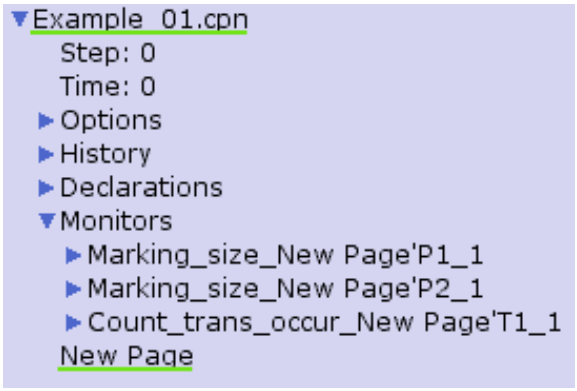


Рис. 33. Вкладка Monitors после добавления мониторов

Добавим мониторы для записи текущей маркировки позициям *P1* и *P2* и монитор для фиксации количества срабатываний переходу *T1*. После этого вкладка Monitors нашей модели примет следующий вид (рисунок 33).

Проведем моделирование 50 шагов работы сети и рассмотрим полученные результаты. По результатам моделирования будут сгенерированы следующие файлы, расположенные в каталоге с моделью.

.\output\PerfReport.html
.\output\logfiles\Count_trans_occur_New_Page'T1_1.log
.\output\logfiles\Marking_size_New_Page'P1_1.log
.\output\logfiles\Marking_size_New_Page'P2_1.log

По имени файла можно определить тип монитора, название элемента, к которому относится данный монитор, и название страницы, на которой располагается соответствующий элемент. Открыв файл «*PerfReport.html*» в любом web-браузере, вы увидите таблицу с результатами моделирования по созданным мониторам (рисунок 34). В данной таблице для каждого перехода записано количество срабатываний, сумма значений измеряемой характеристики, среднее, минимальное и максимальное значения. Данный файл содержит общую информацию от использованных мониторах. Подробная статистика для каждого монитора находится в соответствующих файлах.

CPN Tools Simulation Performance Report
Net: /cygdrive/C/Users/d/Desktop/cpn/Example_01.cpn

Note that these statistics have been calculated for data that is not necessarily independent or identically distributed.

Untimed statistics					
Name	**Count**	**Sum**	**Avrg**	**Min**	**Max**
Count_trans_occur_New_Page'T1_1	30	30	1.000000	1	1
Marking_size_New_Page'P1_1	51	51	1.000000	1	1
Marking_size_New_Page'P2_1	51	173	3.392157	0	10

Simulation steps executed: 50
Model time: 0

Generated: Mon Mar 25 12:26:48 2013

Рис. 34. Результаты моделирования в файле «PerfReport.html»

Рассмотрим содержание текстового файла с результатами моделирования «.*output\logfiles\Marking_size_New_Page'P2_1.log*». Данные в текстовом файле представлены в виде таблицы с четырьмя столбцами: *Data* – количество меток в позиции на данном шаге моделирования; *Counter* – номер записи; *Step* – шаг моделирования; *Time* – значение модельного времени.

В таблице 10 представлены данные по 20-ти первым шагам моделирования для монитора *Marking_size_New_Page'P1_1*.

Таблица 10

Результаты моделирования, маркировка позиции P2

data	counter	step	time		data	counter	step	time
0	1	0	0		3	12	11	0
1	2	1	0		4	13	12	0
0	3	2	0		3	14	13	0
1	4	3	0		4	15	14	0
0	5	4	0		3	16	15	0
1	6	5	0		4	17	16	0
0	7	6	0		3	18	17	0
1	8	7	0		4	19	18	0
0	9	8	0		5	20	19	0
1	10	9	0		4	21	20	0
2	11	10	0					

Аналогичные таблицы с результатами получаются по всем мониторам «*Mark Size*» и «*Count Tran*».

Файлы

Помимо стандартных мониторов в CPN Tools есть возможность вывода данных непосредственно в файл. Данная возможность может быть использована в случае, если возможностей мониторов не достаточно, либо необходимо выводить данные, соблюдая определенное форматирование.

Перед началом работы с файлом необходимо объявить глобальную переменную, которая будет ссылкой на открытый файл:

globref outfile = TextIO.stdOut.

Глобальные переменные объявляются с помощью ключевого слова *globref*, запись *TextIO.stdOut* означает начальное значение данной переменной. Переменная *outfile* имеет тип *TextIO.outstream* (выходной поток). Для получения значения глобальной переменной используется запись *!global_variable*, где *global_variable* - имя глобальной переменной, для присвоения значения запись *global_variable := new_value;* где *new_value* - выражение, значение которого будет присвоено глобальной переменной.

Вывод в файл осуществляется при срабатывании перехода *Action*, в тексте годового сегмента (рис. 35). Для доступа к кодовому сегменту перехода необходимо выделить переход и три раза нажать клавишу tab. Кодовый сегмент выглядит следующим образом:

```
input (i);
action
  outfile := (TextIO.openAppend("OutFile.txt"));
  TextIO.output(!outfile, ("Value: "^Int.toString(i)^"\n"));
  TextIO.closeOut(!outfile)
```

Данный кодовый сегмент состоит из двух секций *input* и *action*. В секции *input* вводятся имена переменных, используемых в кодовом сегменте. В секции *action* происходит непосредственное выполнение команд. Пример вывода данных в файл представлен на рисунке 35.

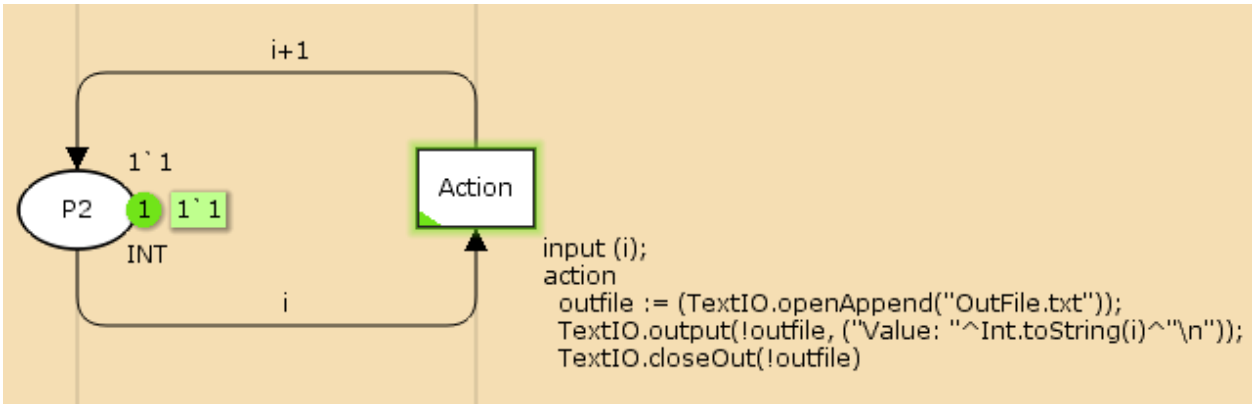


Рис. 35. Пример вывода в файл

Кодовый сегмент состоит из трех команд:

•*outfile := (TextIO.openAppend("OutFile.txt"));* - открытие текстового файла в каталоге с моделью. Файл открывается на добавление. Ссылка на данный файл присваивается глобальной переменной *outfile*.

•*TextIO.output(!outfile, ("Value: "^Int.toString(i)^"\n"));* - вывод строки в файл *outfile*, первым параметром является ссылка на открытый файл. Выводимая строка является результатом склейки трех строк *"Value: "*, значения переменной i, преобразованной в строку *Int.toString(i)*, и строки *"\n"*, предназначенной для записи в файл символа конца строки.

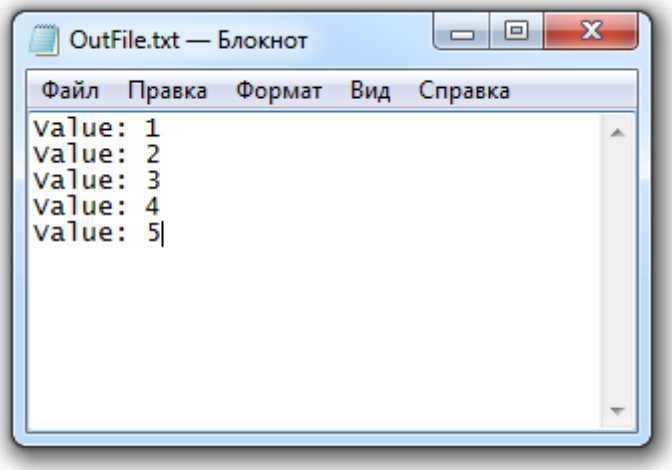


Рис. 36. Текстовый файл с результатами моделирования

•*TextIO.closeOut(!outfile)* - закрытие файла *outfile*.

В момент срабатывания перехода *Action* осуществляется запись строки в файл *"OutFile.txt"*. После пяти шагов моделирования содержимое файла выглядит, как показано на рисунке 36.

Используя данный метод, можно собрать требуемую статистическую информацию о процессе моделирования и представить её в формате, удобном для обработки.

Пространство состояний

По аналогии с деревом достижимых маркировок в ординарных сетях Петри, для раскрашенных сетей вводится понятие пространства состояний. Пространство состояний представляется в виде графа, узлами которого являются достижимые маркировки сети Петри. Отличает пространство состояний от дерева достижимости то, что в дереве достижимости маркировка представляется количеством меток в позиции, а в пространстве состояний при задании маркировки учитывается тип позиции, значения и количество меток.

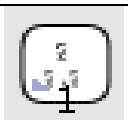
На рисунке 37 представлен внешний вид панели инструментов, предназначенных для построения пространства состояний раскрашенной сети Петри.

Рис. 37. Панель инструментов построения пространства состояний

В таблице 11 приведены основные инструменты, необходимые для построения пространства состояний раскрашенной сети Петри в программе CPN Tools.

Таблица 11

Инструменты построения пространства состояний

Изображение инструмента	Описание инструмента
Enter SS	Инструмент для входа в пространство состояний. Перед началом работы необходимо использовать данный инструмент, кликнув на нем левой кнопкой мыши, а затем кликнув на странице модели, для которой будет построено пространство состояний. Данный инструмент осуществляет подготовку CPN Tools к построению пространства состояний.
	Инструмент построения пространства состояний. После того как был выполнен вход в пространство состояний, CPN Tools осуществляет его расчет.
	Инструмент для отображения узла пространства состояния с заданным номером. По умолчанию отображается корневая вершина. Для начала построения пространства состояний кликните левой кнопкой мыши на данном инструменте, а затем на странице модели.
	Инструмент для отображения потомков (достижимых маркировок) заданного узла пространства состояний. Для выполнения необходимо выбрать инструмент и кликнуть на требуемом узле пространства состояний.
	Инструмент для отображения родителя данного узла пространства состояний. Для выполнения необходимо выбрать инструмент и кликнуть на требуемом узле пространства состояний.

Перед использованием инструмента «вход в пространство состояний» необходимо проверить модель и убедиться в том, что она не содержит синтаксических ошибок (элементы с синтаксическими ошибками обведены красным цветом). Все позиции и переходы должны иметь уникальные имена. В модели не должно быть позиций и переходов с пустым именем. В случае, если вышеперечисленные условия не выполнены, начать работу с пространством состояний будет невозможно.

В качестве примера рассмотрим сеть Петри, моделирующую взаимную блокировку процессов при работе с разделяемыми ресурсами (рисунок 38).

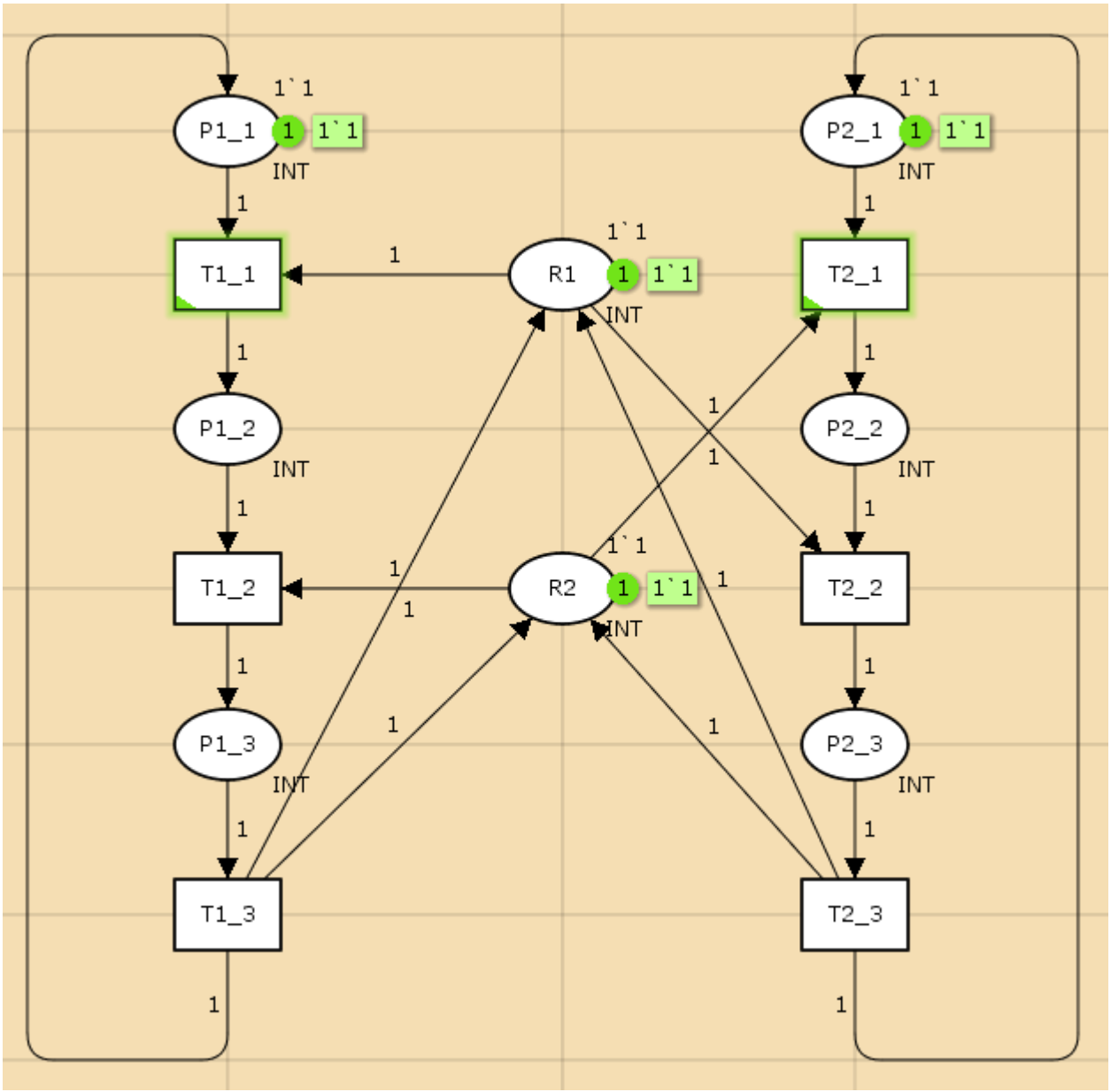


Рис. 38. Сеть Петри, моделирующая взаимную блокировку процессов при работе с разделяемыми ресурсами

Пространство состояний для данной модели представлено на рисунке 39. В пространстве состояний легко заметить вершину (№ 5), не имеющую потомков и являющуюся тупиковой. Таким образом можно сделать вывод о том, что два моделируемых процесса могут попасть в ситуацию взаимной блокировки, кроме того, изучив последовательность срабатывания переходов, приприведших модель к тупиковой маркировке, можно выделить недопустимую последовательность действий.

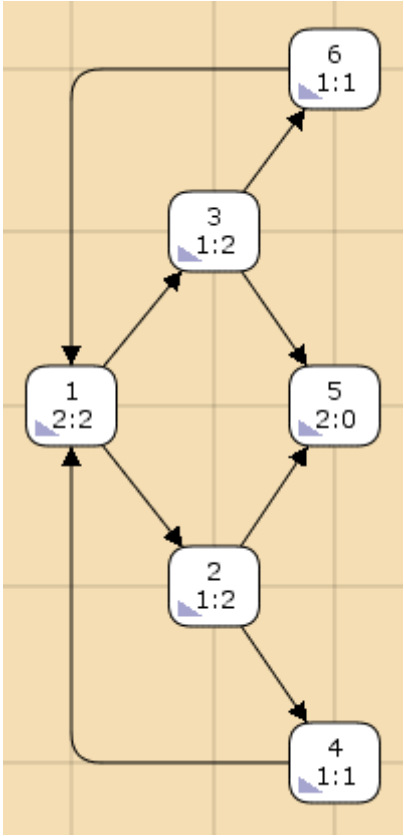


Рис. 39. Пространство состояний модели взаимной блокировки процессов

Данная сеть Петри моделирует работу двух процессов, осуществляющих параллельный доступ к двум разделяемым ресурсам №1 и №2 (позиции *R1* и *R2* соответственно). Работа процесса характеризуется тремя изменениями состояния: получение доступа к ресурсу №1 (переходы *T1_1* и *T2_2* для первого и второго процесса соответственно), получение доступа к ресурсу №2 (переходы *T1_2* и *T2_1* для первого и второго процесса соответственно), освобождения ресурсов №1 и №2 (переходы *T1_3* и *T2_3* для первого и второго процесса соответственно).

Рассмотрим пространство состояний подробнее и выделим возможные последовательности срабатывания переходов. Последовательности (*T1_1* → *T1_2* → *T1_3*) и (*T2_1* → *T2_2* → *T2_3*) не приводят к блокировке процессов, в то время как последовательность (*T1_1* → *T2_1*) или последовательность (*T2_1* → *T1_1*) приводят к блокировке, так как каждый процесс получает доступ к одному из ресурсов, оставляя при этом второй процесс в состоянии ожидания ресурса.

Для получения более полной информации о пространстве состояний предусмотрена возможность отображения имени сработавшего перехода на дуге между двумя узлами и маркировки, соответствующей конкретному узлу пространства состояний. Для отображения имени перехода необходимо кликнуть левой кнопкой мыши на дуге. Для отображения маркировки узла пространства состояний необходимо кликнуть левой кнопкой мыши на треугольнике, расположенном в левом нижнем углу изображения, соответствующего требуемо-

му узлу пространства состояний. Пространство состояний с отображением маркировок в узлах и сработавших переходов представлено на рисунке 40.

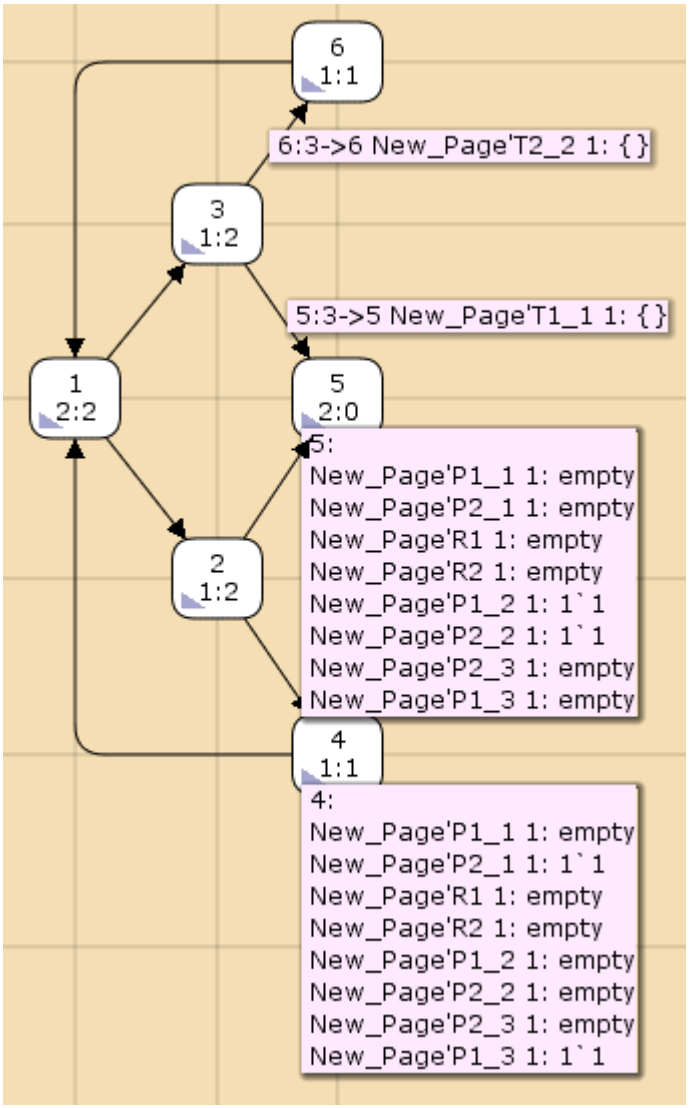


Рис. 40. Пространство состояний, отображение переходов на дугах и маркировки узлов

Пространство состояний является мощным средством анализа раскрашенных сетей Петри. Оно может быть использовано для выявления многих свойств модели (наличие тупиков, безопасность, ограниченность, достижимость и другие). Однако для сложных моделей размеры пространства состояний могут быть чрезвычайно большими. В таких случаях стоит воспользоваться статистическими методами для анализа данных, собранных в ходе моделирования.

Задание для самостоятельной работы

1. Модифицировать модель из лабораторной работы №4. Воспользовавшись стандартными средствами CPN Tools для вывода информации о ходе моделирования, сохранить информацию по отправленным и отброшенным пакетам в маршрутизаторах. Сохранить статистику по работе сервера web-приложения, используя возможности вывода в файл. Файл со статистикой должен содержать строки с информацией об обработанных, либо отброшенных запросах. Для всех запросов необходимо сохранить их номер, для выполненных необходимо сохранить время, затраченное на обработку.
2. Построить сеть, моделирующую конкуренцию произвольного количества процессов за произвольное количество ресурсов. В качестве основы может быть использована модель, рисунок 38. Модель должна быть построена таким образом, чтобы для задания количества ресурсов и процессов достаточно было изменить начальную маркировку соответствующих позиций. Построить пространство состояний для полученной модели. Для того чтобы процесс выполнил работу, ему требуются все ресурсы. Количество ресурсов и процессов в зависимости от варианта задания представлено в таблице 12.

Таблица 12

Варианты заданий

№ варианта	Количество процессов	Количество ресурсов
1	2	4
2	4	3
3	4	2
4	3	3
5	3	4
6	3	2
7	2	3
8	4	4
9	5	2
10	2	5

Содержание отчета

1. Задание на лабораторную работу с выбранным вариантом.
2. По заданию №1:
    a. Привести полученные результаты, 20-30 записей из сгенерированного монитором файла, пояснить результаты моделирования;

б. Представить данные (20-30 записей) по работе сервера web-приложений, пояснить результаты. Также необходимо представить текст кодового сегмента, который был использован для вывода данных в файл.

3. По заданию №2 представить изображение разработанной модели.
4. Представить изображение получившегося пространства состояний.
5. Ответы на контрольные вопросы.
6. Заключение, отражающее содержание и результаты выполненной работы.

Контрольные вопросы

1. Какие существуют инструменты в CPN Tools для вывода информации о ходе выполнения моделей. Какие характеристики могут быть получены?
2. Какие средства предусмотрены в CPN Tools для остановки процесса моделирования в зависимости от состояния сети?
3. Что такое пространство состояний раскрашенной сети Петри?
4. В чем отличие пространства состояний раскрашенной сети Петри от дерева достижимости ординарной сети Петри?
5. Каким образом осуществляется построение пространства состояний в CPN Tools?
6. Каким образом осуществляется работа с файлами в CPN Tools?

Содержание

Подписано в печать 25.12.2013. Усл. печ. л. . Тираж .
Печать офсетная. Бумага офисная. Заказ № _____

Отпечатано: РИО ВоГУ, г. Вологда, ул. Ленина, 15