электронный журнал
МОЛОДЕЖНЫЙ НАУЧНО-ТЕХНИЧЕСКИЙ ВЕСТНИК
Издатель ФГБОУ ВПО "МГТУ им. Н.Э. Баумана". Эл No. ФС77-51038.

#09, сентябрь 2015

УДК 004.021

Анализ моделей параллельных процессов формализованных сетью Петри. Исключение тупиков

Омарова М.О., *студент*
Россия, 105005, г. Москва, МГТУ им. Н.Э. Баумана,
кафедра «Программное обеспечение ЭВМ и информационные технологии»

Научный руководитель: Рудаков И.В., к.т.н., доцент
Россия, 105005, г. Москва, МГТУ им. Н.Э. Баумана,
кафедра «Программное обеспечение ЭВМ и информационные технологии»
irudakov@bmstu.ru

Введение

Классическое последовательное программирование уже не приводит к созданию программ, эффективно использующих ресурсы современных вычислительных систем. В последние десятилетия большой и устойчивый интерес проявляется к математическим средствам моделирования и анализа сложных параллельных и распределенных систем, к которым относятся, например, вычислительные машины и комплексы с параллельной и распределённой архитектурой, параллельные программы и алгоритмы. При разработке подобных систем, как правило, высок риск возникновения ошибки на стадии проектирования и чрезвычайно высока цена проявления ошибки на стадии эксплуатации. Все это обуславливает интерес к формальным математическим средствам анализа, позволяющим дать ответы на вопросы о свойствах модели, например, о наличии конфликтов, тупиков. При недостаточном использовании примитивов синхронизации есть риск получить программу, имеющую в каком-то виде проблемы с синхронизацией. Средства динамического анализа параллельных программ не всегда могут решить данные проблемы. Для ряда программ требуется статический анализ параллельных алгоритмов. Наиболее сложной проблемой является диагностирование возможности тупиков.

Одним из наиболее популярных формализмов для моделирования и анализа параллельных и распределенных систем являются сети Петри. Понятие сети Петри было введено в 1962 году Карлом-Адамом Петри. С тех пор теория сетей Петри сильно разрослась и продолжает активно развиваться. Существует множество методов анализа

сети Петри, но эти методы дают только необходимые условия возникновения тупика в сети. Поэтому на данный момент является актуальной проблема обнаружения тупиков в сети и разработка методов показывающих достоверность возникновения тупиков.

Формализованное представление модели параллельных процессов сетью Петри

Сети Петри – инструмент исследования систем. В настоящее время сети Петри применяются в основном в моделировании. Во многих областях исследований явление изучается не непосредственно, а косвенно, через модель. Модель – это представление, как правило, в математических терминах того, что считается наиболее характерным в изучаемом объекте или системе. Манипулируя моделью системы, можно получить новые знания о ней, избегая опасности, дороговизну или неудобства анализа самой реальной системы. Обычно модели имеют математическую основу.[2]

Теория сетей Петри является хорошо известным и популярным формализмом, предназначенным для работы с параллельными и асинхронными системами. Основанная в начале 60-х годов немецким математиком К.А. Петри, в настоящее время она содержит большое количество моделей, методов и средств анализа, имеющих обширное количество приложений практически во всех отраслях вычислительной техники и даже вне ее.

Причиной популярности сетей Петри является математическая строгость, простота и наглядность описания параллелизма, а также удобное графическое представление модели.

Параллелизм сети Петри заключается в том, что переходы могут быть запущены один вслед за другим в любом порядке. Такие переходы называются *одновременными* и моделируют параллельное возникновение событий. Если переход в сети Петри никогда не может быть запущен, такое состояние называется тупиком. Обнаружение возможности наступления состояния тупика в сети Петри показывает наличие возможности состояния тупика и в моделируемой вычислительной среде.[1] Для разрешения состояния тупика при синхронизации процессов достаточно воспользоваться взаимоисключением доступа к разделяемому ресурсу. При выполнении параллельных процессов может возникать ситуация, когда каждый процесс, обращающийся к разделяемым данным, исключает для всех других процессов возможность одновременного с ним обращения к этим данным - это называется взаимоисключением (mutual exclusion). Два перехода в сети Петри могут находиться в *конфликте*, т. е. запуск одного из них удаляет фишку из общей входной позиции и тем самым запрещает запуск другого. Таким образом, моделируются

взаимоисключающие события системы.

Сеть Петри очень удобно представить графом состоящем из трех простых графических элементов: позиция, переход и маркер. Эти элементы и сеть в целом можно очень просто построить с помощью графического редактора Visio.

Microsoft Office Visio 2003 используется для построения схем и диаграмм различного типа, а также наглядного представления бизнес-процессов, помогает пользователям, занимающимся бизнесом и техническими специалистам наглядно представлять, оформлять и передавать информацию о процессах и системах, повышая эффективность их совместной работы. Фигуры Visio отличаются от простых рисунков тем, что они обладают набором настраиваемых свойств для отображения ключевых данных в контексте диаграммы. [5]

Для упрощения работы в редакторе Visio необходимо создать набор элементов, в котором будут находиться все необходимые для построения сети Петри элементы.

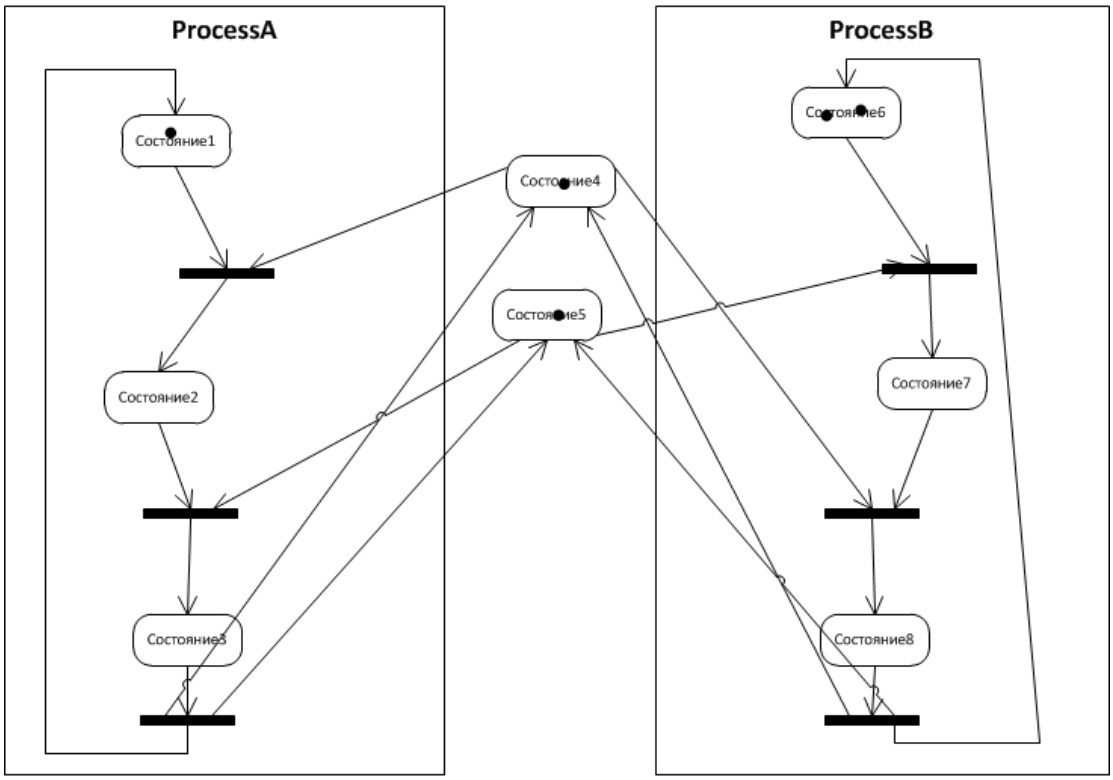

Рис. 1. Пример сети Петри построенной в редакторе MS Visio

Для обеспечения работы Microsoft Visio необходимы следующие системные требования [5]:

- операционная система Microsoft Windows XP с пакетом обновления 2 (SP2) или более поздняя версия или Microsoft Windows Server 2003 или более поздняя версия;

- компьютер и процессор ПК с процессором 500 МГц или более, 256 или более МБ ОЗУ;
- дисковод для DVD-дисков;
- ПК с процессором 1 ГГц и 512 МБ ОЗУ (или более мощный компьютер) необходим для работы с Microsoft Office Outlook 2007 с диспетчером контактов;
- жесткий диск. Для установки необходимо 2ГБ; часть этого объема будет освобождена после установки, когда исходный установочный файл будет удален;
- разрешение экрана Минимум 800x600 точек (рекомендовано 1024x768 или более);
- подключение к Интернету. Для загрузки и активации продуктов понадобится широкополосное подключение к Интернету со скоростью 128 кбит/с и выше.

А также рекомендуется использование дополнительных компонентов Microsoft Internet Explorer 6.0 с установленными пакетами обновления, для пользователей Outlook 2007 понадобится Microsoft Exchange Server 2000 или более поздней версии.

Инструмент анализа сети Петри.

Для работы с документами Microsoft office существует множество различного ПО, но среди всех широко применяемым на данный момент является библиотека ASPOSE.

Программное обеспечение Aspose представляет собой набор инструментов для работы на платформах .NET, Java, Reporting Services, JasperReports и SharePoint. Сочетая различные компоненты в составе пакета Aspose.Total Product Family, продукт гарантирует удобство в работе и предоставляет широкий спектр функций для создания и редактирования документов и отчетов. [4]

С помощью Aspose.Total Product Family Pack можно разрабатывать качественные приложения для.NET и Java платформ, экспортировать отчеты в различных форматах из Microsoft SQL Server Reporting Services и JasperReports, конвертировать документы в различные форматы без использования приложения Microsoft SharePoint.

Среди множества компонентов была выбрана библиотека Aspose.Total for Java, в связи с удобством, гибкостью и надежностью использования компоненты Aspose.Diagram в анализе диаграмм Visio.

Aspose.Diagram - программный продукт для работы с .VSD и .VDX файлами в web-приложениях ASP .NET, web-сервисах и Windows-приложениях, представляющий собой надежную альтернативу MS Visio Object Model.

Структура анализатора сети Петри. Диаграмма классов

Сеть Петри состоит из трех элементов: позиции, переходы и маркеры, опишем каждый элемент в своем классе.

Класс ShapeName задает соотношение элементов диаграммы Visio и элементов Сети Петри, только указанные здесь элементы будут распознаваться анализатором.

```
public class ShapesName {
   public static final String TRANSITION = "Переход (разветвление)";
   public static final String STATE = "Состояние";
   public static final String MARK = "Точка соединения";
}
```

Класс Shape — является базовым классом для элементов сети. Свойство id задается каждой фигуре редактором Visio, будем его использовать при работе с элементами. Поле caption является ориентиром элемента в самом документе Visio, достаточно посмотреть данные фигуры что бы понять с каким элементом мы имеем дело.

```
public class Shape {
   public final Long id;
   public String caption;
   public Shape(Long id, String caption){
      this.id = id;
      this.caption = caption;
   }
}
```

Класс State описывает позицию в Сети Петри, Число маркировок в позиции определяется счетчиком count_mark. Поля вещественного типа задаются для вычисления положения маркера относительно позиции - state. Класс содержит единственный метод isMarked() - определяющий является ли позиция маркированной.

```
public class State {
   public Long count_mark;
   public final Double PinX, PinY, width, height;
   public State(Long id, Long count_mark, String caption, Double PinX, Double PinY, Double width, Double height){
   }
   public boolean isMarked(){
      return count_mark>0;
```

```
    }
}
```

Класс Transition - описывает переход и задается множеством входящих (inputIDs) и исходящих (outputIDs) позиций. [6]

Для определения связи используется следующая функция библиотеки Aspose.Diagram [4]:

shape.connectedShapes(ConnectedShapesFlags.***CONNECTED_SHAPES_INCOMING_NODES***, **null**)- определяет все входящие элементы,

shape.connectedShapes(ConnectedShapesFlags.***CONNECTED_SHAPES_OUTGOING_NODES***, **null**)) — определяет все исходящие элементы;

Свойство active - определяет активность перехода, то есть содержание маркера во всех входных позициях перехода.

```
public class Transition {
    public long[] inputIDs;
    public long[] outputIDs;
    public boolean active
    public Transition(Long id, String caption, long[] inputIDs, long[] outputIDs){

    }
}
```

Класс Mark - является маркером определенной позиции, принадлежность к позиции определяется свойством state_id, но начальная маркировка определяется по координатам (PinX, PinY) положения относительно позиции.

Два конструктора задают начальную маркировку и каждую последующую при работе с сетью.

```
public class Mark {
    public Long state_id;
    public final Double PinX, PinY;
    public Mark(Double PinX, Double PinY){
    }
    public Mark( Long state_id){
    }
}
```

Обобщением всех описанных элементов является класс PetriNetsStructure.

Наиболее важным элементом в классе является diagram экземпляр класса Aspose.Diagram представляющий всю диаграмму документа, из которого будут получены наборы связанных элементов states, transitions, marks. Активирование перехода означает удаление маркировки из входных позиций, осуществляется функцией decMark(), и появление их в выходных позициях - функция incMark().

```
public class PetriNetsStructure {

   public static Diagram diagram;
   public static final String fileName;
   public static final String pageName = "Page-1";
   public static List<State> states = new ArrayList<State>();
   public static List<Transition> transitions = new ArrayList<Transition>();
   public static List<Mark> marks = new ArrayList<Mark>();
   public State getState(Long id){}
   public  Transition getTransition(Long id){}
   public void incMark(Long id){}
   public void decMark(Long id){}
   public Mark getMark(Long state_id){}
}
```

Классс LoadDiagram заполняет всю структуру Сети Петри и задает начальную маркировку методлом setMarks(), определяет корректность построенной сети методом validate(), корректность в том, что ни один переход не может быть входящим в переход или исходящим из перехода, верно и обратное, ни одна позиция не может входить в позицию или исходить из нее.

```
public class LoadDiagram extends PetriNetsStructure{
        private void setMarks(){}
        public  boolean validate(){}
}
```

И наконец, класс моделирующий работу Сети Петри - ActionModel.

Важной задачей этого класса является определение состояния состязаний, то есть гонок за ресурс и выявление тупиковых ситуаций. Алгоритм которых будет описан далее.

```
public class ActionModel extends PetriNetsStructure {
   List<Long> activates = new ArrayList<Long>();
   HashMap<Long, List<StringTokenizer>> outStates = new HashMap<Long,
```

```
List<StringTokenizer>>();
//Методы моделируют работу сети Петри и исключают тупиковые ситуации
public  int setActive(){}
public void activeTr(Transition tr){}
public List<Long> getDedlock(Transition curTr, List<Transition> trList){}
public List<Transition> getActivates() throws Exception {}
public void setOutTR(Long curTR, Long toTR, List<StringTokenizer> listTR){}
public Long nextTR(List<StringTokenizer> listTR, Long curTR, Long toTR, StringTokenizer
tokenizer) {}
```

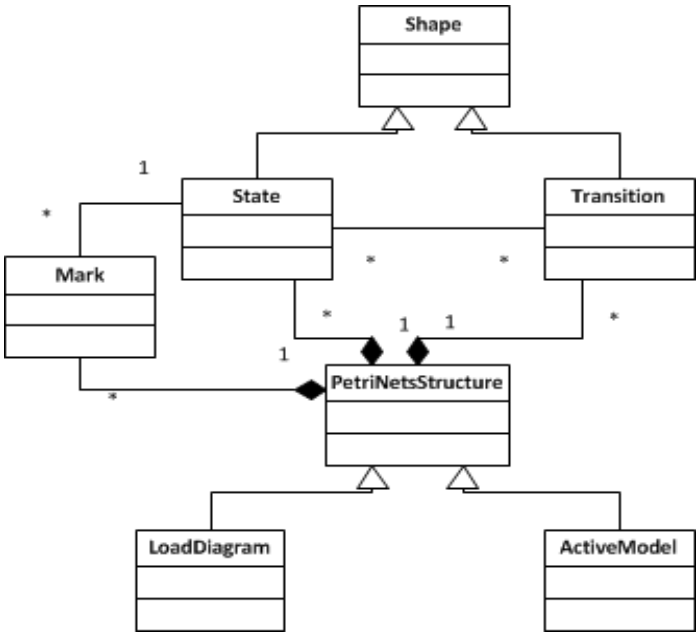

Рис. 2. Диаграмма классов

Методы анализ сетей Петри

Существует два основных метода анализа СП:

1. дерево достижимостей;
2. матричные уравнения.

Дерево достижимости представляет множество достижимости СП. Каждая *i*-я вершина дерева связывается с расширенной разметкой $\mu(i)$. В расширенной разметке число меток в позиции может быть либо неотрицательным целым, либо бесконечно большим. Бесконечное число меток обозначим символом μ. Каждая вершина классифицируется или как граничная, терминальная, дублирующая вершина, или как внутренняя. Граничными являются вершины, которые еще не обработаны алгоритмом.

После обработки граничные вершины становятся либо терминальными, либо дублирующими, либо внутренними. Терминальные (пассивные) маркировки – это маркировки в которых нет разрешенных переходов. Дублирующие маркировки – это маркировки, ранее встречающиеся в дереве. К свойствам сетей Петри, которые не возможно доказать при помощи построения дерева достижимости относятся задачи достижимости и активности, а также для определения возможностей последовательности запусков. [7]

Свойства сетей Петри, которые можно доказать построением дерева достижимости:

1. СП ограничена тогда и только тогда, когда символ ω отсутствует в дереве.
2. СП безопасна, если число фишек в каждой позиции не может превышать.
3. СП живая, если в дереве отсутствуют зацикливания и тупики.

Второй метод анализа сетей Петри при помощи матричных уравнений. Данный подход к анализу сетей Петри основан на матричном представлении сетей Петри. Альтернативным по отношению к определению сети Петри в виде *(P, T, I, O)* является определение двух матриц *D-* и *D+*, представляющих входную и выходную функции. Каждая матрица имеет *m строк (по одной на переход) и n столбцов (по одному на позицию). Матрица D- [j, i] = # (pi , I(tj)) определяет входы в переходы, а матрица D+ [j, i] = # (pi , O(tj)) определяет выходы из переходов в позиции.* С помощью матричных уравнений можно показать сохранение сети Петри. Для этого необходимо найти (ненулевой вектор) взвешивания, для которого взвешенная сумма по всем достижимым маркировкам постоянна. Сеть Петри является сохраняющей тогда и только тогда, когда существует такой положительный вектор w, что *D*w = 0. Это обеспечивает простой алгоритм проверки сохранения, а также позволяет получать вектор взвешивания w.

Матричная теория сетей Петри является инструментом для решения проблемы достижимости. Для доказательства необходимо ввести предположение, что маркировка m¢ достижима из маркировки m.[6]

Метод исключения тупиковых ситуаций

Существует несколько основных методов подхода к анализу активности сетей Петри - метод дерева достижимости, метод матричных уравнений и метод сифонов и ловушек. И метод дерева достижимости, и метод матричных уравнений могут показать необходимость, но не могут показать достаточность для доказательства наличия тупиков в модели. [3]

Для решения данной ситуации предлагается следующий метод.

Основная идея метода заключается в определении на каждом шаге перехода конфликтующего с активным переходом и построении для него всех возможных путей приводящих к срабатыванию конфликтного перехода, то есть определение последовательности срабатываний переходов приводящих к высвобождению разделяемого ресурса. Анализируя эти цепочки переходов можно определить, попадают ли конфликтные переходы в тупик, так как одна цепь срабатываний не должна исключать другую. Данный метод работает с если сеть является безопасной

СП *безопасна*, если безопасны все позиции сети. Позиция сети является *безопасной*, если число фишек в ней никогда не превышает 1.

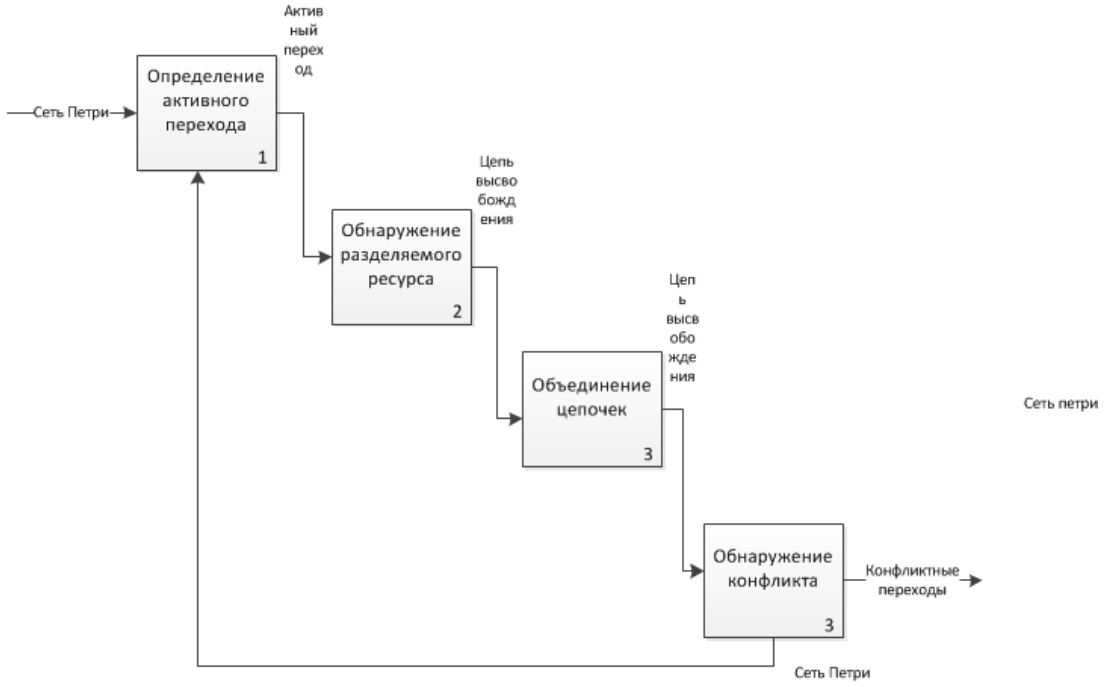

Рис. 3. Диаграмма IDEF0

Словесное описание метода:

1. Определяем все активные на данный момент переходы.
2. Если активный переход rt_i находится в состоянии состязания за ресурс с переходом tr_j, то определяем цепочку срабатывания переходов высвобождающую ресурс начиная от tr_i.
3. Если переход tr_g из цепочки срабатывания перехода tr_j уже был заблокирован, то объединяем их цепочки срабатываний.
4. Перед срабатыванием перехода tr_i , находящегося в состоянии состязания с tr_j, определяем не содержится ли переход tr_j в его цепочке срабатывания, если содержится по

исключаем данную цепочку. И так если не осталось ни одной другой цепочки для перехода tr_j, то переходим на п.7.

5. Запускаем переход tr_i.

6. Если все переходы были активированны, то переходим на п.8, иначе п.1

7. Возникла блокировка (тупиковое состояние), необходимо исключить срабатывание перехода tr_i до высвобождения ресурса разделяемого переходом tr_k.

8. Конец. Все переходы пройдены.

Пример работы анализатора сети Петри

Рассмотрим пример взаимодействия двух процессов с разделяемыми ресурсами.

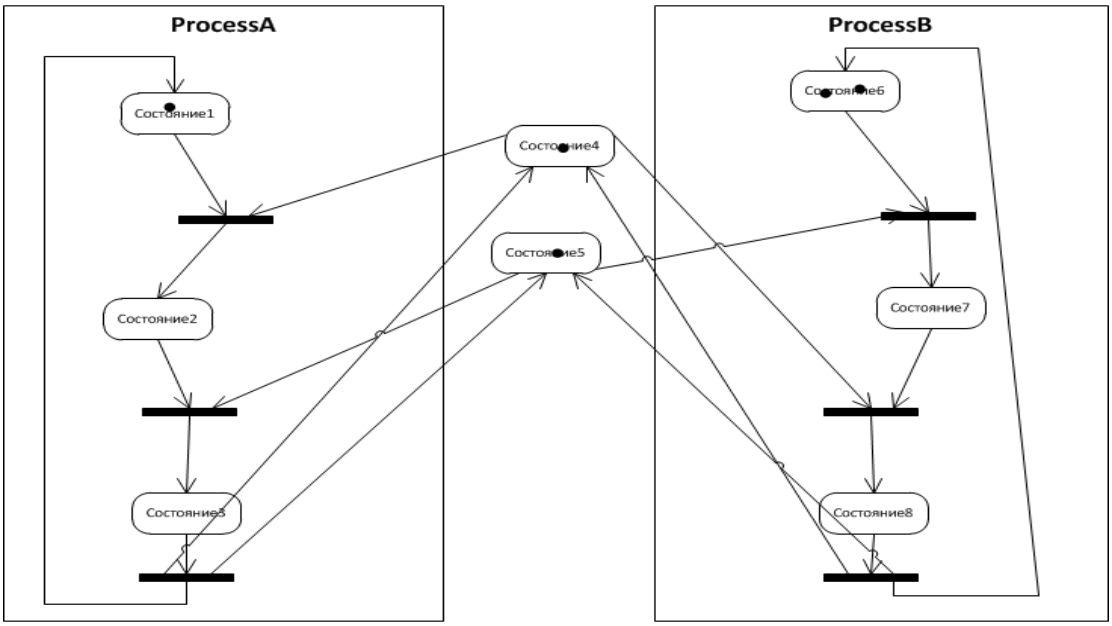

Рис. 4. Модель взаимодействия процессов

По рис. 4 видно что одновременный запуск перехода 1 процесса А и перехода 1 процесса В приведет к блокировке второго перехода обоих процессов. А теперь посмотрим, что выдаст разработанный анализатор.

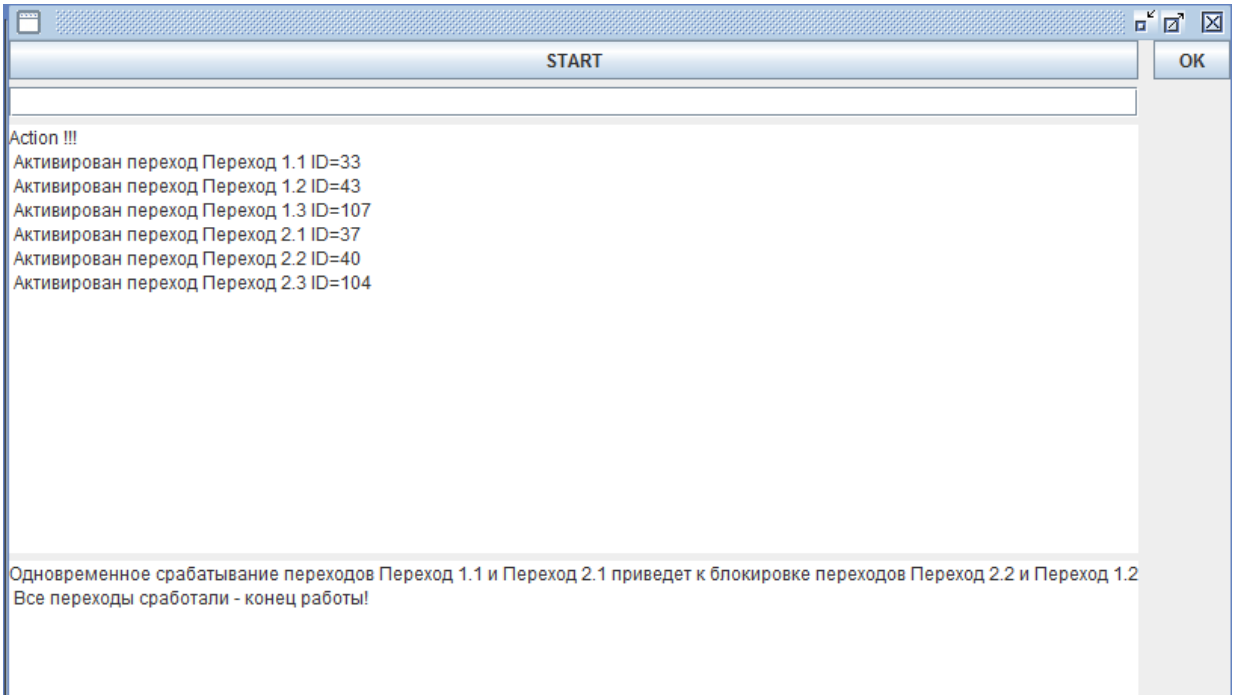

Рис. 5. Вывод анализатора

Анализ данной сети (рис. 4) по разработанному методу показал (рис. 5), что необходимо ввести взаимоисключение между переходом 1.1 и 2.1 иначе сеть зайдет в тупик в гонках за ресурс между переходами 2.2 и 1.2.

Заключение

Так как на данный момент не существует метода анализа сети Петри дающего достаточные условия для обнаружения тупика в сети, предложен новый метод анализа сети Петри для обнаружения тупиков в сети возникающих при синхронизации параллельных процессов с общей памятью. Метод показывает достаточность возникновения тупика, необходимым условием для этого является безопасность сети.

Список литературы

1. Котов В.Е. Сети Петри. М.: Наука. 1984. 160 с.
2. Советов Б.Я., Яковлев С.А. Моделирование систем. М.: Высшая школа. 2005. 344 с.
3. Документация по библиотеке Aspose. Режим доступа: http://www.aspose.com/docs/dashboard.action (дата обращения 20.05.2014).
4. Бьяфоре Б. Microsoft Visio 2007. Библия пользователя Visio 2007: учебное пособие. М.: Высшая школа, 2009. 259 с.
5. Рудаков И. В., Ребриков А. В. Неполная верификация сложных дискретных систем

// Информационные технологии. 2011. № 3. С. 31-34.

6. Рудаков И. В., Ребриков А. В. Масштабирование алгоритмов для автоматической генерации модульных тестов // Вестник МГТУ им. Н. Э. Баумана. Сер. Приборостроение. 2011. № 4. С. 119-124.
7. Рудаков И. В., Ребриков А. В. Проверка выполнения функциональных требований к алгоритму на основе структурной генерации модульных тестов // Вестник МГТУ им. Н.Э. Баумана. Сер. Приборостроение. 2012. Спец. вып. 2 «Программная инженерия». С. 67-79.