

Моделирование сетей Петри в программе Platform Independent Petri Net Editor

Вихляев Дмитрий Романович

Приамурский государственный университет имени Шолом-Алейхема

Студент

Аннотация

В данной работе приводятся пример моделирования параллельного алгоритма с помощью сети Петри. Моделирование производится в оконном приложении Platform Independent Petri Net Editor. В исследовании описываются правила построения сетей Петри. Приводится пример распознавания взаимной блокировки и анимационный режим программы.

Ключевые слова: сеть Петри, параллельный алгоритм, моделирование параллельных алгоритмов.

Modeling of Petri nets in the Platform Independent Petri Net Editor program

Vikhlyaev Dmitry Romanovich

Sholom-Aleichem Priamursky State University

Student

Abstract

This paper provides an example of modeling a parallel algorithm using a Petri net. The simulation is performed in the windowed application Platform Independent Petri Net Editor. The study describes the rules for building Petri nets. An example of interlocking recognition and the animation mode of the program are given.

Keywords: Petri net, parallel algorithm, simulation of parallel algorithms.

1 Введение

1.1 Актуальность

Современные вычисления основаны на многоядерных и распределенных системах, где распараллеливание стало необходимостью для повышения производительности. Параллельные вычислительные системы применяются в некоторых физических дисциплинах. В астрофизике производят моделирование астрофизических объектов, таких как звездные недра и сверхновые. В физике высоких энергий определяют можно ли при помощи стандартных моделей описать явления на субъядерном уровне. Точное моделирование поведения различных ускорителей и реалистическое моделирование характеристик кварк-глюонной плазмы проводят в физике ускорителей и ядерной физике с помощью параллельных алгоритмов. Нанотехнологии ведут расчеты однородных и неоднородных каталитических моделей. В науке о жизни развитие расчетных моделей атомного уровня и

сложных биомолекул. С другой стороны, разработка параллельных алгоритмов сталкивается с фундаментальными проблемами, включая, помимо прочего, недетерминированное поведение выполнения, состояния гонки и состояния взаимоблокировки. Такие ошибки плохо поддаются поиску через методы отладки. Поэтому современные решения нуждаются в инструментах формального моделирования, позволяющих анализировать поведение системы до ее реализации. Сети Петри предлагают математический аппарат для такого анализа, они проводят верификацию корректности параллельных процессов и предотвращают возможные конфликты ещё на этапе проектирования.

1.2 Обзор исследований

В работе М.В.Мальков, С.Н.Малыгина приводятся множество примеров моделирования Сетей Петри [1]. Исследование В.И.Волчихина с соавторами демонстрирует логику и моделирование агентно-базированных метакомпьютерных систем с повышенным уровнем параллельности [2]. В. А.Игнатенко, В.З.Магергут описали динамические процессы при помощи информационной сети Петри [3]. В.А.Игнатенко, В.З.Магергут исследовали информационную сеть Петри как инструмент для параллельной обработки алгоритмов управления [4]. И.Г.Фёдоров провёл метод отображения исполняемой модели бизнес-процесса в сети Петри [5].

1.3 Цель исследования

Цель исследования – реализовать моделирование параллельных алгоритмов и процессов с помощью формального аппарата сетей Петри в программе Platform Independent Petri Net Editor

2 Материалы и методы

В работе описывается интерфейс и набор функций программы Platform Independent Petri Net Editor (PIPE) версии 4.3.0.

3 Результаты и обсуждения

Сеть Петри строится на основе кружков, которые называются позициями, и прямоугольников, которые называются переходами. Кружки и прямоугольники соединяются стрелками, причём стрелка всегда идёт либо от кружка к прямоугольнику, либо от прямоугольника к кружку. Внутри кружков находятся фишки, обозначающие текущее состояние системы. Переход может сработать, если в каждом кружке, из которого в него ведёт стрелка, есть хотя бы одна фишка. В момент срабатывания переход забирает по одной фишке из этих кружков и добавляет по одной фишке в те кружки, в которые от него идут стрелки. Перемещение фишек описывает работу систем с параллельными или последовательными процессами.

Структурные элементы:

- позиции (P): Множество позиций, которые могут содержать метки (фишки). На схеме обозначаются кругами или S;

- переходы (T): Множество переходов, которые срабатывают при выполнении условий. На схеме обозначаются прямоугольниками.
- дуги (I и O): Соединяют позиции и переходы. Дуги могут быть входными (от позиции к переходу) или выходными (от перехода к позиции), но не соединяют два элемента одного типа;
- мультиграф: Сеть допускает кратные дуги от одной вершины к другой (от одной позиции к одному переходу или наоборот), что создает мультиграф. Дуги имеют направление;
- маркировка (m): Состояние сети, определяемое числом меток (фишек) в каждой позиции;
- веса дуг: Каждая дуга может иметь вес, определяющий количество фишек.

Переход (t) разрешен (или "активен"), если каждая входная позиция (p) для этого перехода содержит не меньше фишек, чем вес соответствующей входной дуги (наименьший вес, если веса не равны 1).

При срабатывании перехода (t), из каждой входной позиции (p) удаляется такое же количество фишек, сколько указано на входной дуге (I(t)). В каждую выходную позицию (p) помещается такое количество фишек, сколько указано на выходной дуге (O(t)).

Platform Independent Petri Net Editor (PIPE) – это платформенно-независимый инструмент с открытым исходным кодом для создания, моделирования и анализа сетей Петри, включая обобщённые стохастические сети Петри. Программа является платформенно-независимой, так как написана на языке программирования Java. Программа PIPE включает функции, позволяющие охватывать классические сети позиций/переходов (P/T Nets) для качественного анализа. Также содержит обобщенные стохастические сети Петри (GSPN), которые предназначены для количественной оценки производительности и временных характеристик сложных систем.

Для пользователя предусмотрен графическим редактор, который позволяет конструировать модель, задавать начальные условия, веса дуг и временные параметры. Для отладки логики работы сети предусмотрен режим анимации, позволяющий визуально наблюдать за "игрой токенов" в реальном времени. Программа использует модульную архитектуру, которая позволяет подключать плагины для генерации графов достижимости, расчета структурных инвариантов решения Марковских цепей, связанных с GSPN. На рисунке 1 изображено главное окно программы PIPE, открывающееся при запуске.

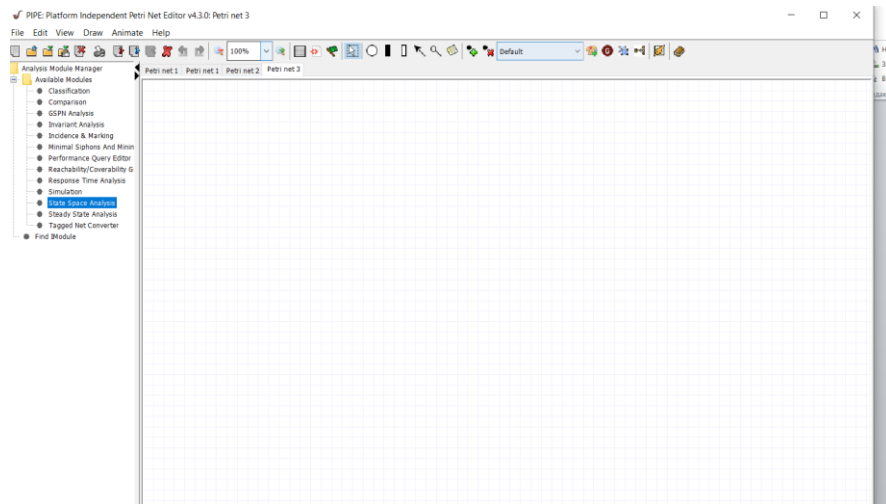


Рис.1. Главное окно программы PIPE

Сверху имеется панель инструментов, где среди основных компонентов оконной программы, доступны объекты для построения сети Петри (рис.2).



Рис.2. Инструменты панели инструментов для построения сети Петри

Внутреннее окно программы содержит полотно для построения графов. Каждое полотно представляется как отдельный файл формата XML. Пример маркированной сети Петри, построенной в программе PIPE, представлено на рисунке 3.

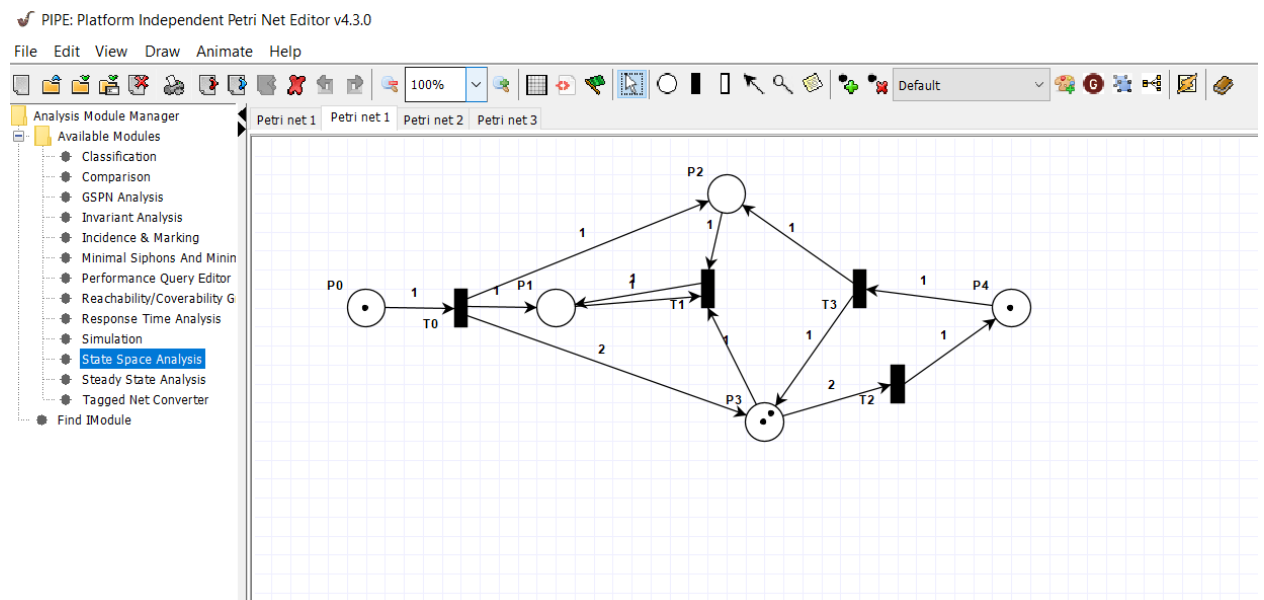


Рис.3. Инструменты панели инструментов для построения сети Петри

Для построенной сети, программа предоставляет метод включения в анимационном режиме, где отображает доступные переходы, и реализуют анимированный процесс перехода при клике мыши (рис.4).

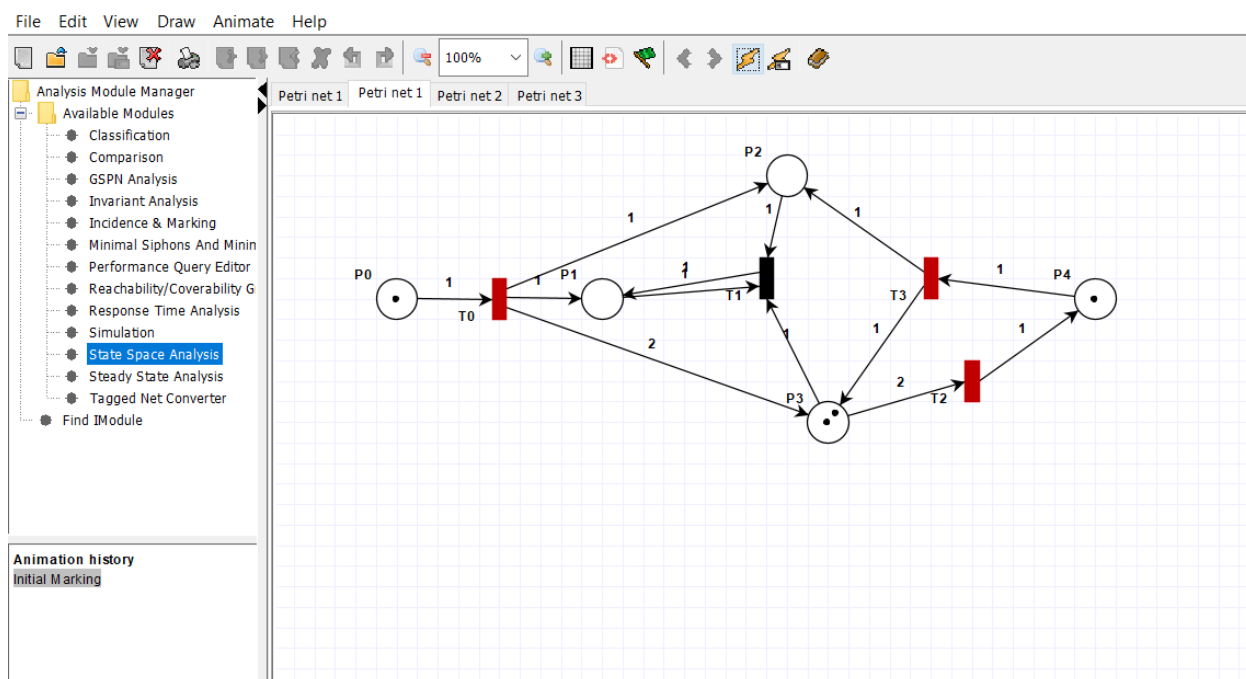


Рис.4. Анимационный режим

В представленной сети Петри изначально доступны 3 только переходы T0, T2, T3. Переход T1 не разрешен, так как ни позиция P1, ни позиция P2, являющиеся входами перехода T1 не содержат ни одной фишки. Так как переходы T0, T2, T3 разрешены, любой из них может быть запущен. Если запущен переход T3, то происходит удаление фишки из каждого входа и помещение фишки в каждый выход. При этом одна фишка удаляется из P4, одна фишка помещается в P2, а количество фишек в P3 увеличивается с двух до трех. Новая маркировка, полученная в результате запуска перехода T3, показана на рисунке 5.

PIPE: Platform Independent Petri Net Editor v4.3.0

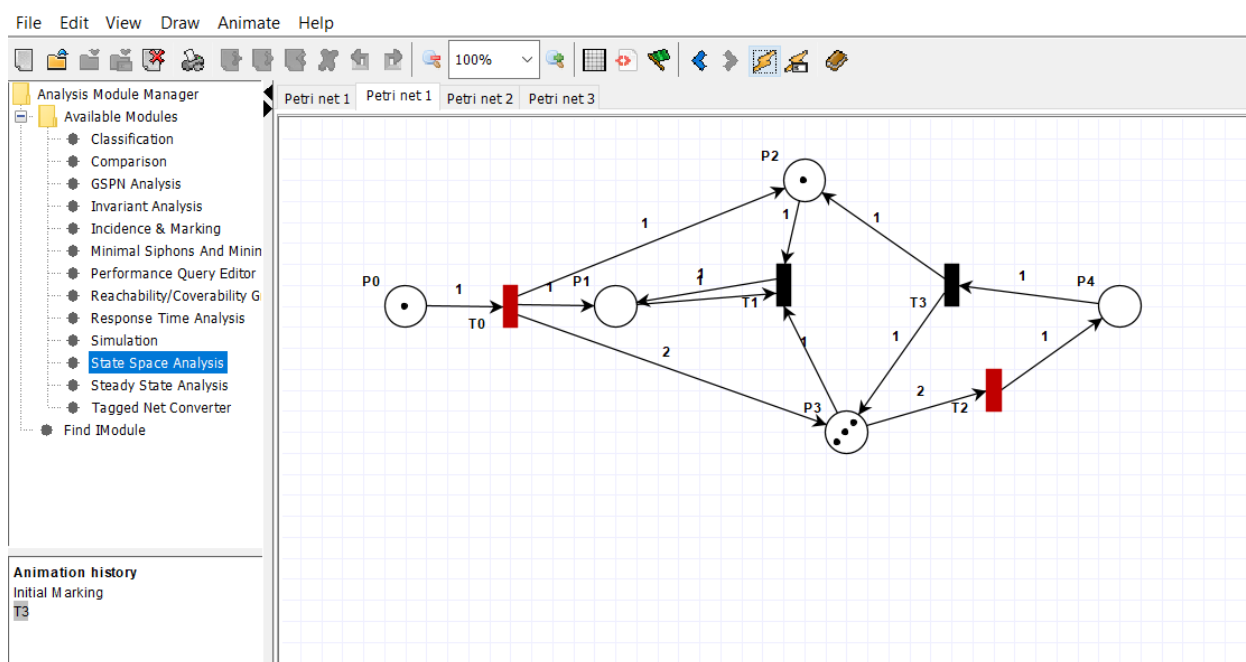


Рис.5. Маркировка, полученная в результате запуска перехода T3

История реализованных переходов отображается в левом нижнем окне программы. В анимированном режиме на верхней панели инструментов имеются стрелки для хождения по истории переходов.

Среди доступных модульных инструментов программы, представленных списком в левом верхнем окне, есть модуль (State Space Analysis) позволяющее определить наличие взаимной блокировки сети и указать кратчайший маршрут (рис.6).

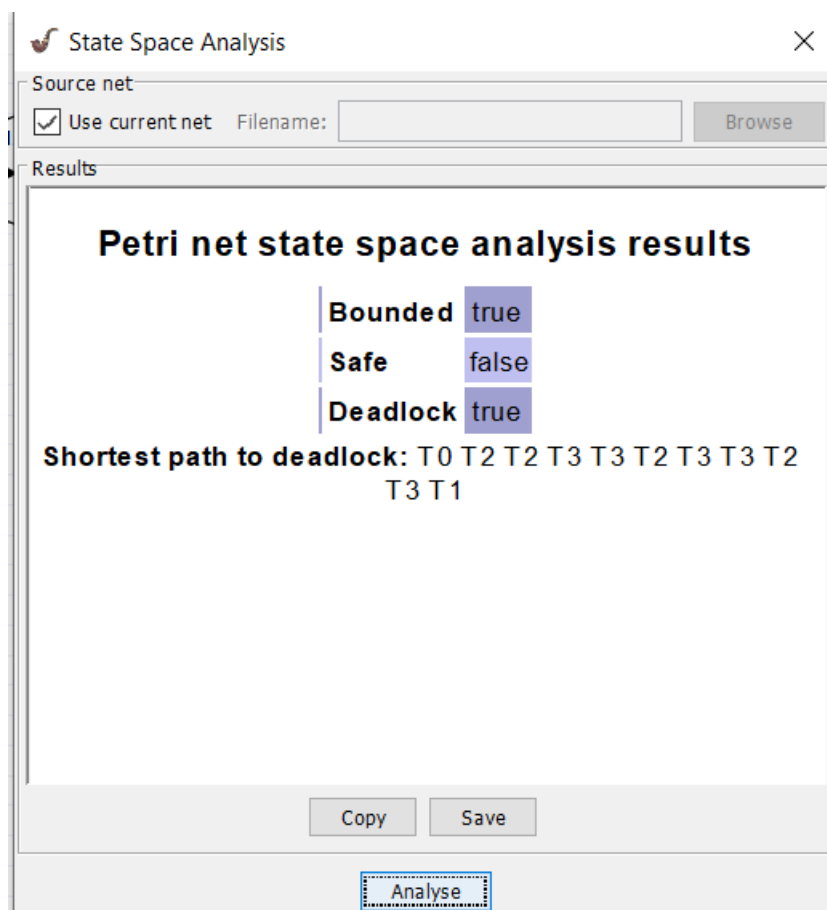


Рис.6. Результат работы модуля State Space Analysis

Что бы проверить корректность работы модуля, были реализована последовательность шагов, которая должна привести сеть в состояние взаимной блокировки (рис. 7).

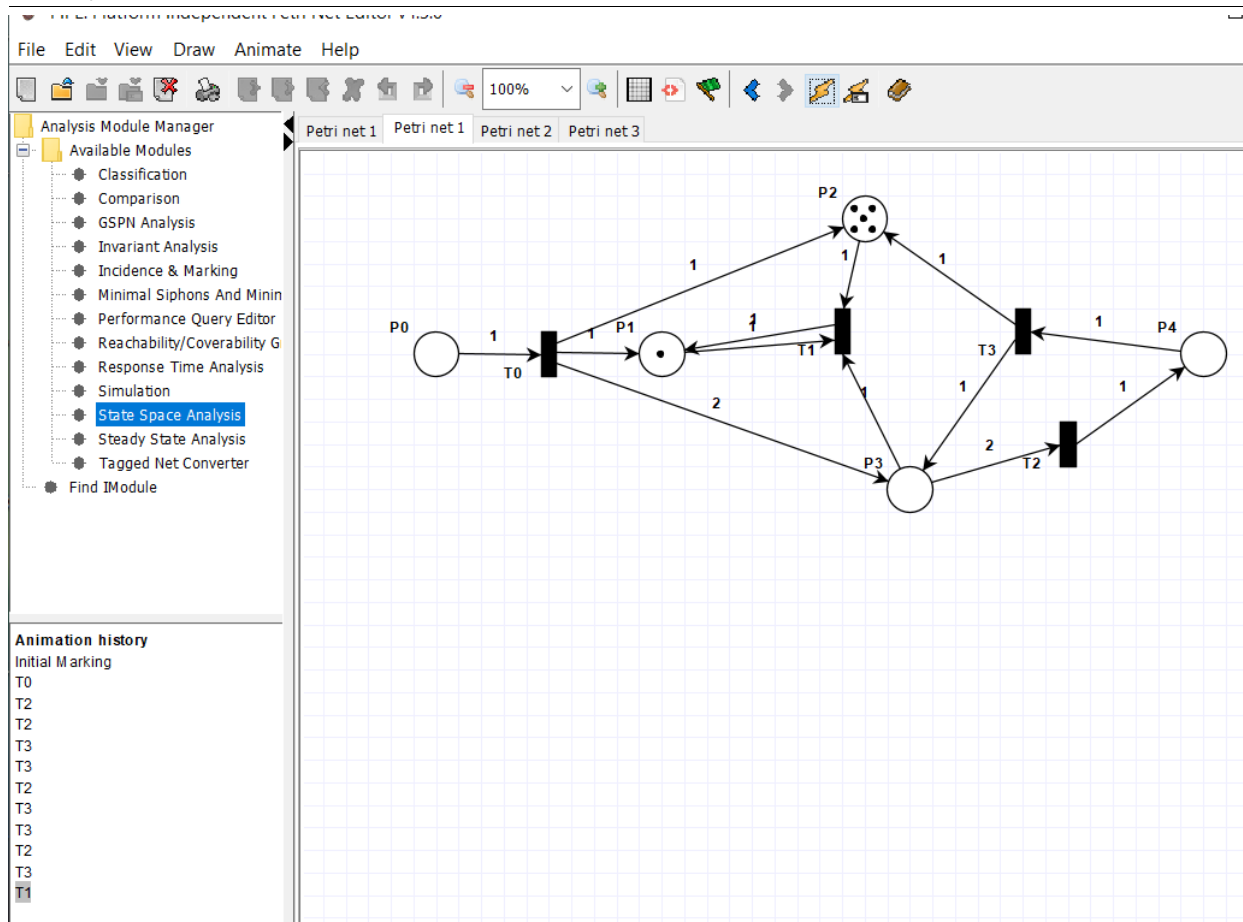


Рис.7. Состояние взаимной блокировки

Взаимная блокировка часто является следствием фундаментальных требований к системе, например, совместное использование ограниченных ресурсов, асинхронный доступ. Существуют три основных подхода к "правильным" системам в контексте взаимных блокировок:

Предотвращение (Prevention): Система проектируется таким образом, что одно из четырех необходимых условий для возникновения блокировки никогда не выполняется. Это идеальный, но не всегда возможный на практике подход.

Избегание (Avoidance): Система динамически принимает решения о выделении ресурсов, используя алгоритмы (например, алгоритм банкира), чтобы гарантировать, что система всегда останется в "безопасном состоянии".

Обнаружение и восстановление (Detection and Recovery): Система позволяет блокировкам возникать, но имеет механизмы для их обнаружения (например, построение графа ожидания ресурсов) и последующего принудительного разрешения (например, принудительное завершение одного из процессов-участников с последующим откатом транзакции).

Таким образом, правильная система гарантирует живучесть (liveness) и безопасность (safety), используя один из этих механизмов для управления ситуациями взаимной блокировки. На рисунках 8,9 представлены сеть Петри программы, исключающая взаимную блокировку в своей работе, и результат анализа данной сети.

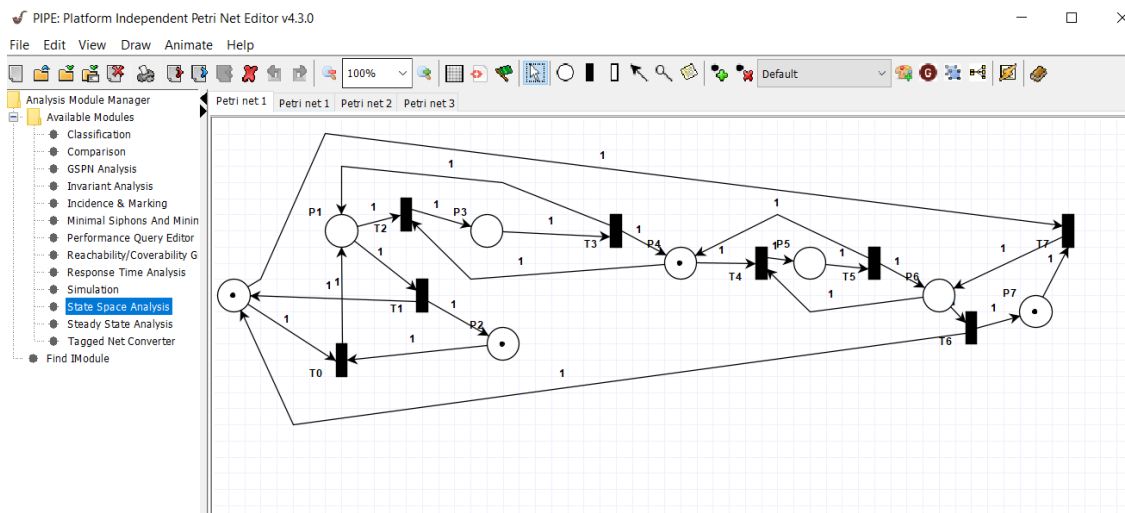


Рис.8. Пример сети Петри исключающей взаимную блокировку

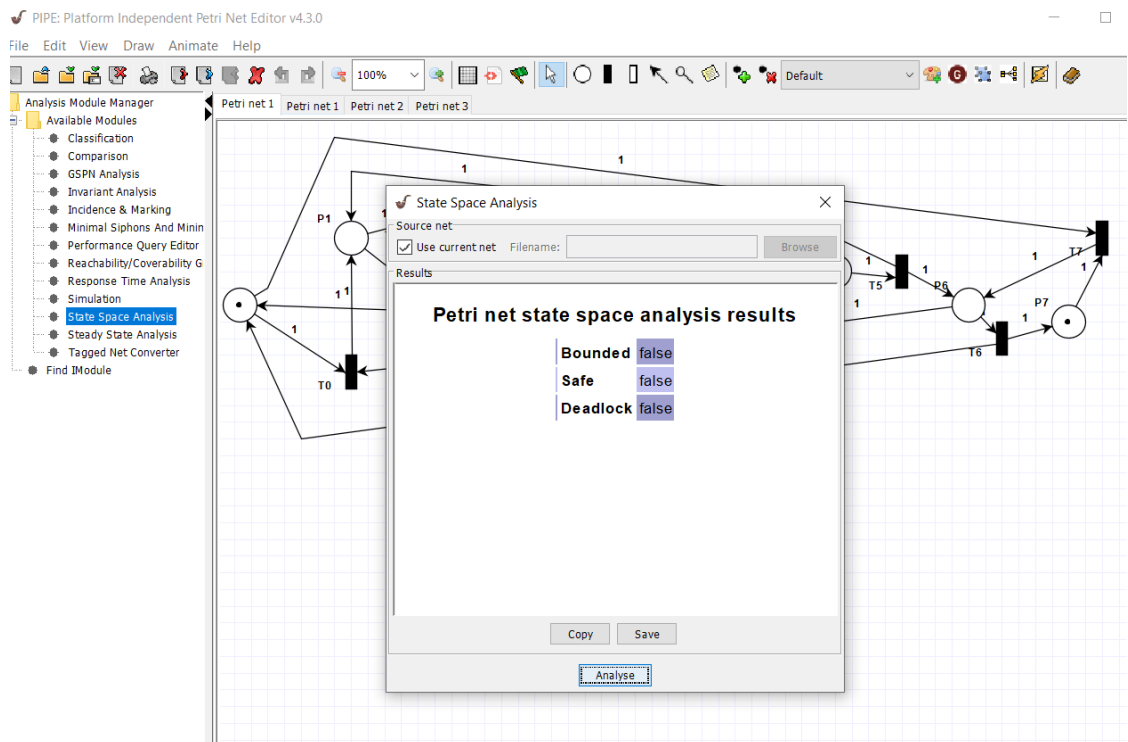


Рис.9. Результат анализа сети Петри исключающей взаимную блокировку

Практическое моделирование в среде PIPE показало, что сети Петри позволяют обнаруживать потенциальные проблемы (взаимные блокировки, состояния гонки) на этапе проектирования, до реализации в коде.

4 Выводы

Из результатов исследования можно сделать вывод, что формальное моделирование сетей Петри позволяет проектировать параллельные алгоритмы, сокращая время по сравнению с традиционной отладкой программ. Приведён пример задачи, для которой была разработана и проанализирована формальная модель, демонстрирующая ключевые аспекты

параллельных вычислений: синхронизацию процессов, управление общими ресурсами и предотвращение конфликтов. Инструменты визуального моделирования делают данный метод доступным для разносторонних разработчиков, работающих с многопоточными и распределенными системами.

Библиографический список

1. Мальков М.В., Малыгина С.Н. Сети Петри и моделирование //Труды Кольского научного центра РАН. 2010. №. 3. С. 35-40.
2. Волчихин, В.И., Карамышева, Н.С., Зинкин, С.А., Гурин Е.И. Алгоритмика, логика и моделирование агентно-базированных метакомпьютерных систем с повышенным уровнем параллельности //Известия высших учебных заведений. Поволжский регион. Технические науки. 2022. №. 2 (62). С. 5-25.
3. Игнатенко В.А., Магергут В.З. Описание динамических процессов при помощи информационной сети Петри //Экономика. Информатика. 2011. Т. 19. №. 13-1 (108). С. 161-179.
4. Игнатенко В.А., Магергут В.З. Информационная сеть Петри как инструмент для параллельной обработки алгоритмов управления //Экономика. Информатика. 2011. Т. 20. №. 19-1 (114). С. 126-133.
5. Фёдоров И.Г. Метод отображения исполняемой модели бизнес-процесса в сети Петри //Статистика и экономика. 2013. №. 4. С. 178-183.