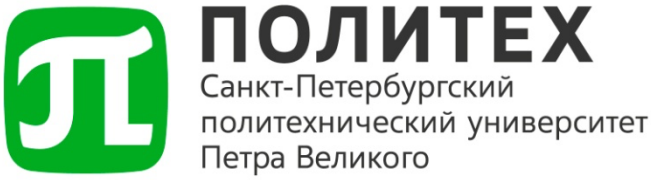
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ОБРАЗОВАНИЯ
«САНКТ-ПЕТЕРБУРГСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ ПЕТРА ВЕЛИКОГО»
ВШ программной инженерии



UPPAAL

по дисциплине «Верификация и тестирование программного обеспечения»

Выполнили
Студенты 3530202/80202 группы — А.М. Потапова, Р. Ж. Айдаров, Н.С. Парфенов

Руководитель — Е.А. Павлов

Санкт-Петербург
2023 г.

Содержание

Описание инструмента

UPPAAL — это среда для моделирования и верификации систем реального времени, разработанная совместно отделом фундаментальных исследований в области компьютерных наук Ольборгского университета в Дании и департаментом информационных технологий Уппсальского университета в Швеции. Она подходит для систем, которые могут быть смоделированы как набор недетерминированных процессов с конечной структурой управления и вещественными тактовыми сигналами, взаимодействующими по каналам или совместно используемым переменным. Типичные области применения включают контроллеры реального времени и протоколы связи, в частности, те, где аспекты синхронизации имеют решающее значение.

a. Архитектура

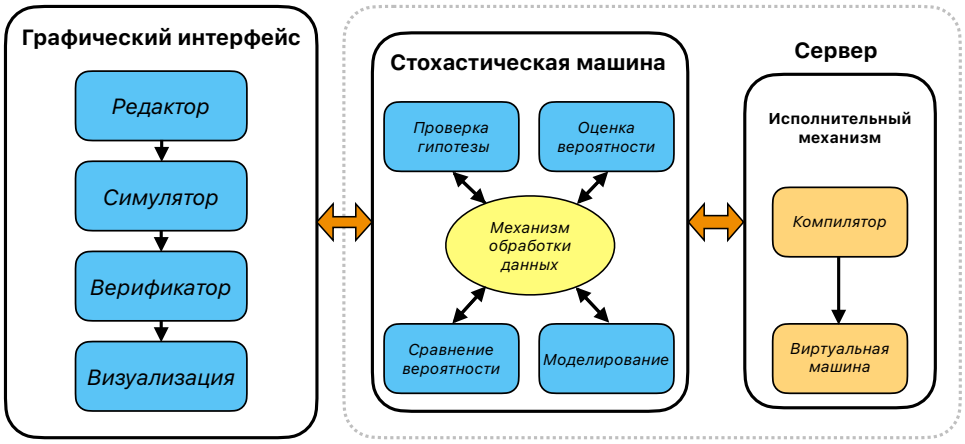


Рисунок 1. Архитектура UPPAAL.

Обзор архитектуры UPPAAL [1], представленной на рисунке 1:

1. Графический пользовательский интерфейс инструмента позволяет пользователю рисовать шаблоны автоматов в редакторе, создавать экземпляры и компоновать их в систему, моделировать систему для легкой проверки, проверять запросы и визуализировать количественные ответы в форме сюжетов в визуализаторе.

2. Исполнительный механизм Uppaal реализует стохастическую семантику взаимодействующих гибридных автоматов и включает виртуальную машину для выполнения кода модели. Здесь пользователь может контролировать точность анализа по ряду статистических параметров. Uppaal также предоставляет распределенные реализации проверки гипотез и оценки вероятности, демонстрирующие линейное ускорение.

3. Результаты, полученные в ходе анализа, могут быть обработаны для визуализации различными способами: диаграммы Ганта для отдельных прогонов, отслеживающие желаемые переменные, графики функций плотности и накопленные функции распределения относительно заданных свойств.

b. Принцип работы

Uppaal использует архитектуру клиент-сервер, разделяя инструмент на графический пользовательский интерфейс и механизм проверки моделей. Пользовательский интерфейс, или клиент, реализован на Java, а механизм, или сервер, скомпилирован для разных платформ. Как следует из названий, эти два компонента могут работать на разных машинах, поскольку они взаимодействуют друг с другом через TCP/IP. Существует также автономная версия движка, которую можно использовать в командной строке. [2]

- **Клиент**

Идея этого инструмента состоит в том, чтобы смоделировать систему с синхронизированными автоматами с помощью графического редактора, симулировать ее, чтобы убедиться, что она ведет себя так, как предполагалось, и, наконец, проверить ее правильность в отношении набора свойств. Графический интерфейс (GUI) Java-клиента отражает эту идею и разделен на три основные части: редактор, симулятор и верификатор, доступный через три «вкладки».

- **Редактор**

Система определяется как сеть синхронизированных автоматов, называемых в инструменте процессами, соединенными параллельно. Процесс создается из параметризованного шаблона. Редактор разделен на две части: панель дерева для доступа к различным шаблонам и объявлениям и редактор холста/текста для рисования. Местоположения помечены именами и инвариантами, а ребра помечены условиями защиты (например, e == id), синхронизациями (например, go?) и назначениями (например, x := 0). Дерево слева дает доступ к различным частям описания системы:

– Глобальное объявление содержит глобальные целочисленные переменные, часы, каналы синхронизации и константы.

- Шаблоны содержат различные параметризованные временные автоматы. Шаблон может иметь локальные объявления переменных, каналов и констант.
- Назначения процессов – шаблоны реализуются в процессах. Раздел назначения процесса содержит объявления для этих экземпляров.
- Определение системы – список процессов в системе.

- **Симулятор**

Симулятор можно использовать тремя способами: пользователь может запустить систему вручную и выбрать, какие переходы выполнять, можно включить случайный режим, чтобы система работала сама по себе, или пользователь может просмотреть трассировку (сохраненную или импортированы из верификатора), чтобы увидеть, как достижимы определенные состояния. Он разделен на четыре части:

- Часть управления используется для выбора и запуска разрешенных переходов, прохождения трассировки и переключения случайной симуляции.
- Представление переменных показывает значения целочисленных переменных и ограничения часов. Uppaal не показывает конкретные состояния с фактическими значениями часов. Поскольку таких состояний бесконечно много, Uppaal вместо этого показывает наборы конкретных состояний, известных как символические состояния. Все конкретные состояния в символическом состоянии имеют один и тот же вектор местоположения и одни и те же значения для дискретных переменных. Возможные значения часов описываются набором ограничений. Проверка часов в символическом состоянии — это именно те, которые удовлетворяют всем ограничениям.
- Системное представление показывает все созданные автоматы и активные местоположения текущего состояния.

– Диаграмма последовательности сообщений показывает синхронизацию между различными процессами, а также активные местоположения на каждом этапе.

- **Верификатор**

Пользователь может проверить модель одного или нескольких свойств, вставить или удалить свойства и переключить представление для просмотра свойств или комментариев в списке. Когда свойство выбрано, можно отредактировать его определение или комментарии, чтобы неформально задокументировать значение свойства. Панель состояния внизу показывает связь с сервером.
Когда генерация трасс включена и средство проверки модели находит трассу, пользователю предлагается импортировать ее в симулятор.
Удовлетворяющие свойства отмечены зеленым цветом, а нарушенные — красным. Если в меню опций выбрано чрезмерное или недостаточное приближение, может случиться так, что проверка с использованием используемого приближения не даст результатов. В этом случае свойства отмечены желтым цветом.

- **Автономный верификатор**

При выполнении больших задач проверки часто бывает неудобно выполнять их из графического интерфейса. Для таких ситуаций больше подходит автономный верификатор командной строки под названием verifyta. Это также упрощает выполнение проверки на удаленной машине UNIX с запасом памяти. Он принимает командную строку аргументы для всех параметров, доступных в графическом интерфейсе

c. Описание языка

Модель системы в UPPAAL состоит из сети процессов, описываемых как расширенные временные автоматы. Описание модели состоит из трех частей: ее глобальных и локальных объявлений, шаблонов автоматов и определения системы. *[6]*

- **Объявления**

Являются либо глобальными, либо локальными (для шаблона) и могут содержать объявления ограниченных целых чисел, каналов (хотя локальные каналы бесполезны), массивов, записей и типов. Синтаксис описывается грамматикой для объявлений:

```
Declarations  ::= (VariableDecl | TypeDecl | [Function] | [ChanPriority])*
VariableDecl  ::= [Type] VariableID (',' VariableID)* ';'
VariableID    ::= ID [ArrayDecl]* [ '=' Initialiser ]
Initialiser   ::= [Expression]
                | '{' Initialiser (',' Initialiser)* '}'
TypeDecls     ::= 'typedef' Type ID ArrayDecl* (',' ID ArrayDecl*)* ';'
```

Рисунок 2. Синтаксис объявлений.

Глобальные объявления также могут содержать не более одного объявления приоритета канала.

– *Типы*

Существует 6 предопределенных типов: int, bool, clock, chan, double и string. Типы массивов и записей могут быть определены для всех типов, кроме string. Квалификаторы типов: константы, мета-переменные, массивы, переменные-записи, скалярные переменные.

– *Функции*

Могут быть объявлены наряду с другими объявлениями. Синтаксис для функций определяется следующим образом:

```
Function        ::= [Type] [ID] '(' [Parameters] ')' Block
Block           ::= '{' LocalDeclaration* Statement* '}'
LocalDeclation  ::= TypeDeclaration | VariableDeclaration
Statement       ::= Block
                  | ';'
                  | [Expression] ';'
                  | ForLoop
                  | Iteration
                  | WhileLoop
                  | DoWhileLoop
                  | IfStatement
                  | ReturnStatement

ForLoop         ::= 'for' '(' [Expression] ';' [Expression] ';' [Expression] ')' Statement
Iteration       ::= 'for' '(' [ID] ':' [Type] ')' Statement
WhileLoop       ::= 'while' '(' [Expression] ')' Statement
DoWhile         ::= 'do' Statement 'while' '(' [Expression] ')' ';'
IfStatment      ::= 'if' '(' [Expression] ')' Statement [ 'else' Statement ]
ReturnStatement ::= 'return' [ [Expression] ] ';'
```

Рисунок 3. Синтаксис объявления функций.

Объявления внутри функций включают только объявления переменных и типов. Объявления вложенных функций и рекурсия не поддерживаются.

– *Внешние функции*

Могут быть созданы вместе с другими объявлениями. Являются локальными по отношению к текущей области, определяемой грамматикой:

```
ExternDecl   = 'import'  String '{' [FwdDeclList] '}'
FwdDeclList  = FwdDecl ';' |
               FwdDeclList FwdDecl ';'
FwdDecl      = [ID '='] Type ID '(' [Parameters] ')'
```

Рисунок 4. Синтаксис объявления внешних функций.

Рекомендуется всегда использовать полный путь к файлу библиотеки. Если используются целые числа во внешней функции, то рекомендуется определить полный диапазон целых чисел, чтобы избежать проблем с целыми числами, ограниченными UPPAAL.

```
const int INT32_MAX = 2147483647;
typedef int[INT32_MIN,INT32_MAX] int32_t;

import "/home/user/lib/externallib.so" {
  int32_t get_a_number();
  void set_a_number(int32_t n);

  importWithNewName = bool is_the_wold_safe();
}
```

Рисунок 5. Пример внешней функции с полным путем к файлу.

- **Шаблоны**

UPPAAL предоставляет богатый язык для определения шаблонов в виде расширенных временных автоматов. В отличие от классических временных автоматов, временные автоматы в UPPAAL могут использовать богатый язык выражений для тестирования и обновления переменных, типов записей, вызова пользовательских функций и т.д.

Автомат шаблона состоит из локаций и ребер. Шаблон также может иметь локальные объявления и параметры. Экземпляр шаблона создается назначением процесса (в определении системы).

- **Параметры**

Шаблоны и функции параметризованы. Синтаксис параметров определяется грамматикой для параметров:

```
Parameters ::= [ Parameter (',' Parameter)* ]
Parameter  ::= [Type] [ '&' ] [ID] [ArrayDecl]*
```

Рисунок 6. Синтаксис параметров.

В отличие от глобальных и локальных объявлений, список параметров не должен заканчиваться точкой с запятой.

– *Вызов по ссылке и вызов по значению*

Параметры могут быть объявлены как имеющие семантику вызова по значению или вызова по ссылке. Синтаксис взят из C++, где идентификатор параметра, вызываемого по ссылке, имеет префикс амперсанда в объявлении параметра. Параметры вызова по значению не имеют префикса амперсанда. Каналы всегда должны быть эталонными параметрами. Примечание: Параметры массива должны иметь префикс амперсанда для передачи по ссылке, это не соответствует семантике C.

- **Определение системы**

В определении системы определяется модель системы. Такая модель состоит из одного или нескольких параллельных процессов, локальных и глобальных переменных и каналов.

Глобальные переменные, каналы и функции могут быть определены в определении системы с использованием грамматики для объявлений. Они наиболее полезны при предоставлении фактических аргументов формальным параметрам шаблонов. Объявления в определении системы и в разделе объявления верхнего уровня являются частью модели системы. Процессы модели системы определяются в форме строки объявления системы с использованием грамматики для System:

Рисунок 7. Синтаксис объявления системы.

Системная строка содержит список шаблонов, которые будут созданы в процессах. Процессы могут быть расставлены по приоритетам.

Шаблоны без параметров создаются точно в одном процессе с тем же именем, что и шаблон. Параметризованные шаблоны приводят к запуску одного процесса для каждой комбинации аргументов, т.е. UPPAAL автоматически привязывает любые свободные параметры шаблона. Любой такой параметр должен быть либо целым числом с ограничением по значению, либо скаляром по значению. На отдельные процессы можно ссылаться в выражениях, используя грамматику для процесса, приведенную ниже.

Рисунок 8. Синтаксис процесса.

- **Приоритеты**

Учитывая некоторый порядок приоритета переходов, интуиция подсказывает, что в данный момент времени переход разрешен только в том случае, если не включен переход с более высоким приоритетом. Таким образом происходит блокировка перехода с более низким приоритетом. Приоритетные порядки преобразуются в приоритетный порядок для тау-переходов и синхронизирующих переходов. Переходы с задержкой по-прежнему недетерминированы (если только не используются срочные каналы). Приоритеты могут быть назначены каналам и процессам системы.

– *Приоритеты по каналам*

Объявление приоритета канала может быть вставлено в любое место в разделе глобальных объявлений системы (только одно для каждой системы). Объявление приоритета состоит из списка каналов, где разделитель ‘<’ определяет более высокий уровень приоритета для каналов, перечисленных в его правой части. Уровень приоритета по умолчанию используется для всех каналов, которые не упомянуты, включая переходы tau.

```
ChanPriority ::= 'chan' 'priority' (ChanExpr | 'default') ((',' | '<') (ChanExpr | 'default'))* ';'
ChanExpr ::= ID
           | ChanExpr '[' Expression ']'
```

Рисунок 9. Синтаксис приоритета.

– *Приоритеты процессов*

Задаются в системной строке с использованием разделителя ‘<’ для определения более высокого приоритета для процессов справа от нее. Если в списке указан экземпляр набора шаблонов, все процессы в наборе будут иметь одинаковый приоритет.

– *Синхронизация*

При синхронизации приоритеты процессов неоднозначны, поскольку в таком переходе задействовано более одного процесса. При синхронизации

нескольких процессов приоритет перехода определяется наивысшим приоритетом процессов. Это относится как к стандартной синхронизации, так и к трансляции.

- **Области видимости и применения**

Правила области видимости определяют, на какой элемент ссылается имя в данном контексте. Контекст является либо локальным (для шаблона процесса), либо глобальным (в описании системы).

В локальном контексте имена всегда ссылаются на локальные объявления или формальные параметры (если имя определено локально), в противном случае на глобально объявленное имя.

В глобальном контексте название всегда ссылается на глобальную декларацию.

Примечание: В каждом контексте существует только одно пространство имен. Это означает, что в каждом контексте все объявленные часы, целочисленные переменные, константы, местоположения и формальные параметры должны иметь уникальные имена. Однако локальные имена могут затенять глобально объявленные имена.

d. Особенности

Два основных критерия дизайна Uppaal – эффективность и простота использования. Применение технологии поиска «на лету» имеет решающее значение для эффективности проверки модели в данной инструментальной среде. Другим важным ключом к эффективности является применение символической техники, которая сводит проблемы верификации к проблемам эффективной манипуляции и решения ограничений. Чтобы упростить моделирование и откладку, средство проверки моделей Uppaal может автоматически генерировать диагностическую трассировку, которая объясняет, почему свойство удовлетворяется (или не удовлетворяется) описанию системы. Диагностические трассы, сгенерированные средством проверки моделей, могут автоматически загружаться в симулятор, который может использоваться для визуализации и исследования трассы. [3]

Преимущества:

- Простое в использовании.
- Бесплатное для научных исследований.
- Имеет в основе простую, но при этом довольно мощную математическую модель.

Недостатки:

- Малопригодно для разработки больших систем.
- Не поддерживает описание иерархии вложенности компонентов систем.

e. Практическое применение инструмента

UPPAAL был применен в ряде отраслевых тематических исследований, таких как:

- **Сети для предотвращения SCADA-атак в режиме реального времени:**

Предложен метод обнаружения атак, нацеленных на SCADA-системы. Используя метод проверки модели, мы моделируем журналы временных рядов, собранные из SCADA-систем, в сеть синхронизированных автоматов и характеризуем поведение SCADA-системы при атаке с помощью временной логики. Экспериментальная оценка, проведенная на газораспределительной системе, подтверждает эффективность предложенного метода.

- **Моделирование и проверка бизнес-активности веб-служб:**

Спецификация WS-Business Activity определяет два протокола координации для обеспечения согласованного результата длительно работающих распределенных приложений. Мы используем средство проверки моделей UPPAAL для анализа бизнес-соглашения с типом протокола завершения согласования.

- **Протокол атомарных транзакций веб-служб:**

В данном примере UPPAAL используется для проверки свойства согласованности протокола WS-AT. Этот протокол является частью WS-Coordination framework и описывает алгоритм достижения соглашения о результатах распределенной транзакции.

- **Протокол аудио/видео Bang & Olufsen:**

Это протокол управления звуком в реальном времени. Протокол разработан Bang & Olufsen для передачи сообщений между аудио/видео

компонентами по одной шине. Хотя было известно, что он неисправен, ошибка не была обнаружена с помощью обычных методов тестирования. При использовании UPPAAL автоматически создается трассировка ошибок, которая выявляет ошибку. Кроме того, предлагается исправление, которое автоматически подтверждается с помощью UPPAAL.

- **Протокол предотвращения столкновений:**

Протокол реализован поверх среды, подобной Ethernet, такой как протокол CSMA/CD. Это делается для того, чтобы обеспечить верхнюю границу задержки связи между узлами сети. Он был разработан и доказал свою корректность с использованием UPPAAL. Два установленных основных свойства показывают, что протокол не допускает коллизий, и он действительно обеспечивает верхнюю границу задержки связи между пользователями (при условии идеальной среды).

- **Коммерческий протокол полевой шины:**

Это коммерческий протокол связи по полевой шине, используемый в различных промышленных средах по всему миру. Протокол был разработан и внедрен компанией ABB для приложений, критически важных для безопасности, например, для управления технологическими процессами. Это тематическое исследование является одним из крупнейших, где применялся UPPAAL. В ходе тематического исследования был обнаружен ряд недостатков в логике протокола и реализации TIS, и источники ошибок были отлажены на основе абстрактных моделей протокола; там, где были предложены соответствующие улучшения.

- **Контроллер коробки передач:**

В этом примере UPPAAL применяется для проектирования и анализа прототипа контроллера коробки передач для транспортных средств от Mecel AB. Контроллер коробки передач — это компонент распределенной системы

реального времени, которая управляет современным автомобилем. Запросы на передачу от водителя передаются через человеко-машинный интерфейс по сети связи на контроллер коробки передач. Контроллер осуществляет фактическое переключение передач, приводя в действие компоненты системы нижнего уровня (такие как сцепление, двигатель и коробка передач). При проектировании контроллера символический симулятор UPPAAL применяется для проверки поведения системы. Правильность конструкции регулятора коробки передач устанавливается автоматическими проверками 46 свойств, полученных на основе требований, указанных Mecel AB.

- **Системы LEGO ® MINDSTORMS ™ - Синтез управляющих программ:**

В этой работе рассматривается проблема синтеза производственных графиков и управляющих программ для модели завода по серийному производству, построенной в LEGO ® MINDSTORMS ™ RCX ™. Описана временная автоматная модель установки, которая точно отражает уровень абстракции, необходимый для синтеза управляющих программ. Это делает модель очень детализированной и сложной для автоматического анализа. Чтобы решить эту проблему, представлен общий способ добавления руководства к модели путем дополнения ее дополнительными переменными руководства и ограничителями перехода. Применение этого метода делает задачу синтеза контроля выполнимой для завода, производящего до 60 партий. Для сравнения, только две партии могли быть запланированы без направляющих. Синтезированные управляющие программы были выполнены на заводе. Выполнение этого выявило некоторые ошибки модели.

- **Алгоритм для липсинка:**

Алгоритм используется для синхронизации нескольких информационных потоков, передаваемых по сети связи, в данном случае аудио- и видеопотоков мультимедийного приложения. Спецификация

алгоритма смоделирована и проверена в UPPAAL. Интересно, что проверка выявляет некоторые ошибки в алгоритме синхронизации, например, что взаимоблокировочные ситуации могут возникать до того, как после ошибки будут достигнуты предварительно описанные состояния ошибки.

- **Потоковое транслирование мультимедиа:**

Протокол представляет спецификацию и проверку мультимедийного потока в UPPAAL. Мультимедийные потоки являются строительными блоками распределенных мультимедийных приложений. Они выражают непрерывную передачу произвольных типов носителей, например аудио или видео. Поток описывается в модели UPPAAL, а затем определенные свойства в реальном времени (называемые свойствами качества обслуживания в мультимедийной области) проверяются в model-checker. Особое внимание уделяется проверке пропускной способности и сквозной задержки

- **Philips Audio Protocol:**

Протокол разработан и внедрен компанией Philips для обмена управляющей информацией между компонентами аудиооборудования с использованием манчестерского кодирования. Правильность кодирования зависит от временных задержек между сигналами.

Инструкция по установке инструмента

Для загрузки и установки необходимой версии Uppaal нужно:

1. Выбрать необходимую версию программного обеспечения на странице официального сайта [4]:

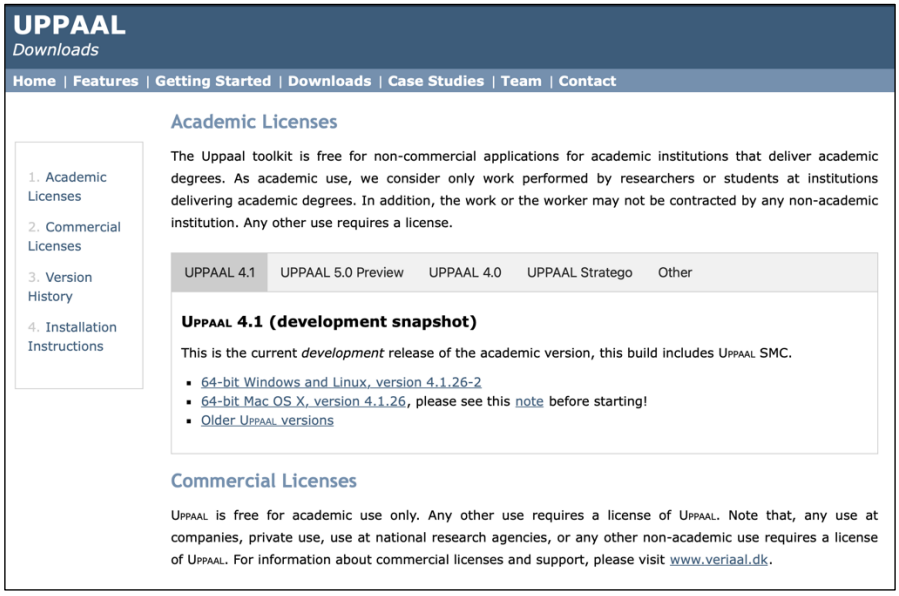


Рисунок 10. Главная страница сайта UPPAAL.

2. Заполнить лицензионное соглашение и подтвердить его:

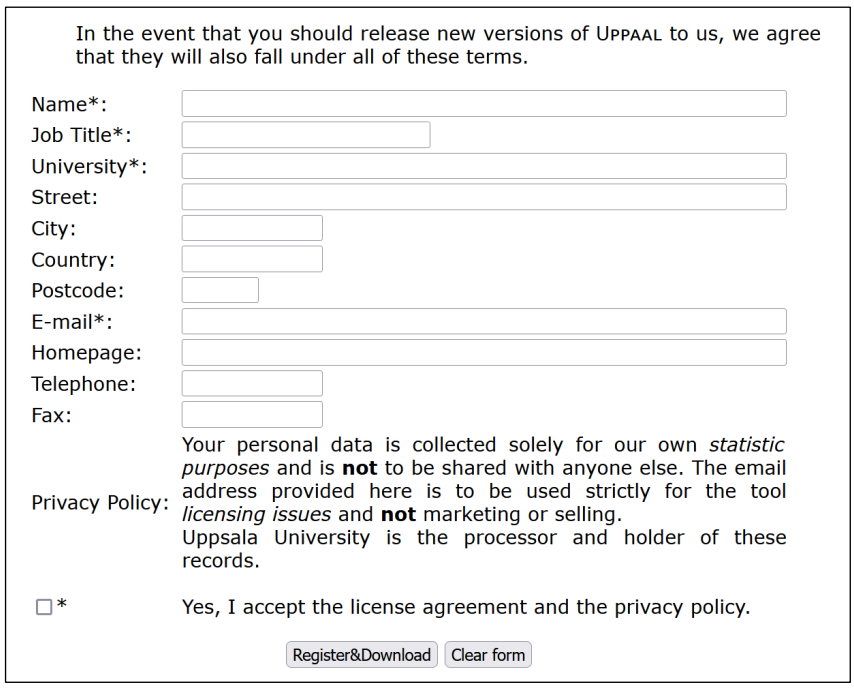


Рисунок 11. Лицензионное соглашение на скачивание UPPAAL.

3. Загрузить zip-файл, который содержит необходимые файлы для установки программного обеспечения.

4. Распаковать загруженный архив. В нем должны присутствовать следующие файлы: uppaal.jar, uppaal и каталоги bin-Linux, bin-Win32 и demo. Все bin-каталоги должны содержать два файла: server(.exe) и verifyta(.exe), а также некоторые дополнительные файлы, в зависимости от платформы. Демонстрационный каталог должен содержать несколько демонстрационных файлов с расширениями ‘.xml’ и ‘.q’.

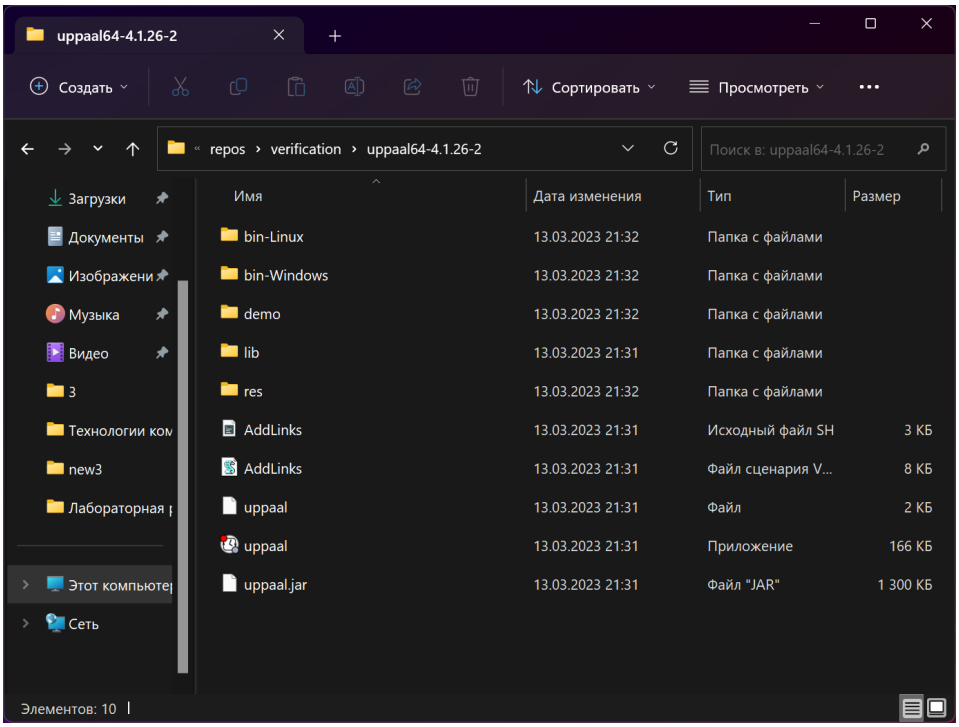


Рисунок 12. Распакованный архив для установки UPPAAL.

5. Убедиться, что в вашей системе настроена как минимум Java 11. Графический интерфейс Uppaal не будет работать без установленной Java.

6. Чтобы запустить Uppaal в системах Linux, нужно запустить сценарий запуска с именем uppaal. Для запуска в системах Windows необходимо запустить файл uppaal.jar.

Пример работы

В качестве примера смоделируем семафор, который управляет доступом к одноколейному мосту с нескольких железных дорог, как показано на *рисунке 13*. [5]

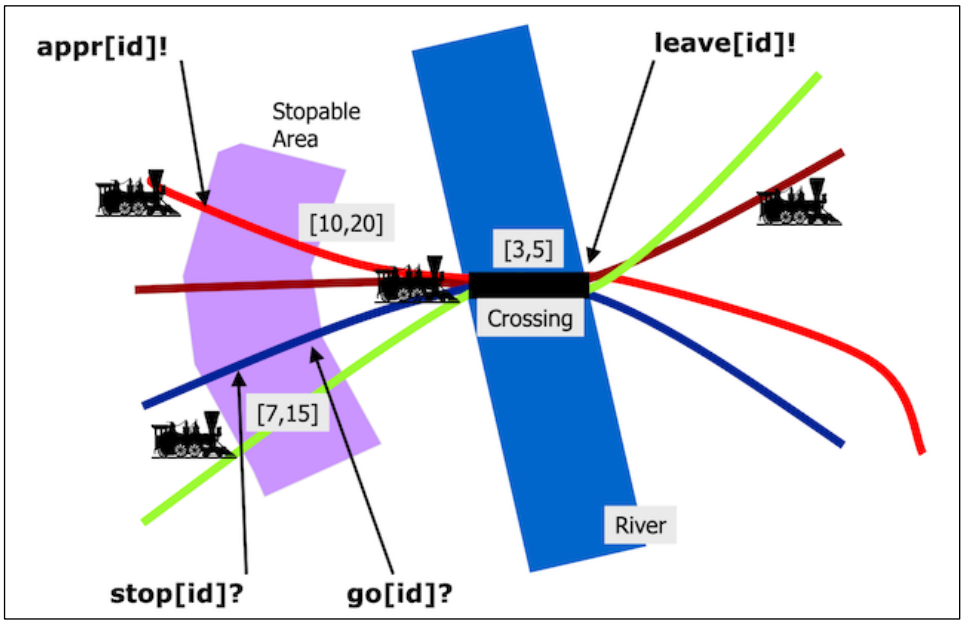


Рисунок 13. Принцип работы семафора.

Семафор должен гарантировать отсутствие коллизий. Треки и семафоры общаются через синхронные сообщения:

- трек подает appr сигнал, когда поезд приближается к мосту. Поезд достигает моста в течение 10–20 секунд после подачи appr сигнала (если поезд не был остановлен до этого).
- когда семафор получает appr сигнал, у него есть до 10 секунд, чтобы отправить stop сигнал поезду.
- остановленный поезд перезапускается семафором с сообщением go. Затем поезду требуется от 7 до 15 секунд, чтобы добраться до моста.
- поезду нужно от 3 до 5 секунд, чтобы пересечь мост. Затем он подает leave сигнал, когда он пересек мост.

Описание шаблонов:

- коммуникации моделируются как синхронные каналы appr, stop, go, leave в глобальных переменных
- шаблон Train, показанный на *рисунке 14*, параметризуется идентификатором id, который отличает экземпляры шаблона. Синхронизированный автомат реализует описанный выше протокол Train.
- шаблон Semaphore, показанный на *рисунке 15*, реализует взаимоисключающий доступ к мосту. Он останавливает любой поезд, приближающийся к мосту, если другой поезд уже находится на мосту. Когда поезд уходит с моста, Semaphore перезапускает один из ожидающих поездов. С этой целью он хранит набор ожидающих поездов в виде вектора логических значений.

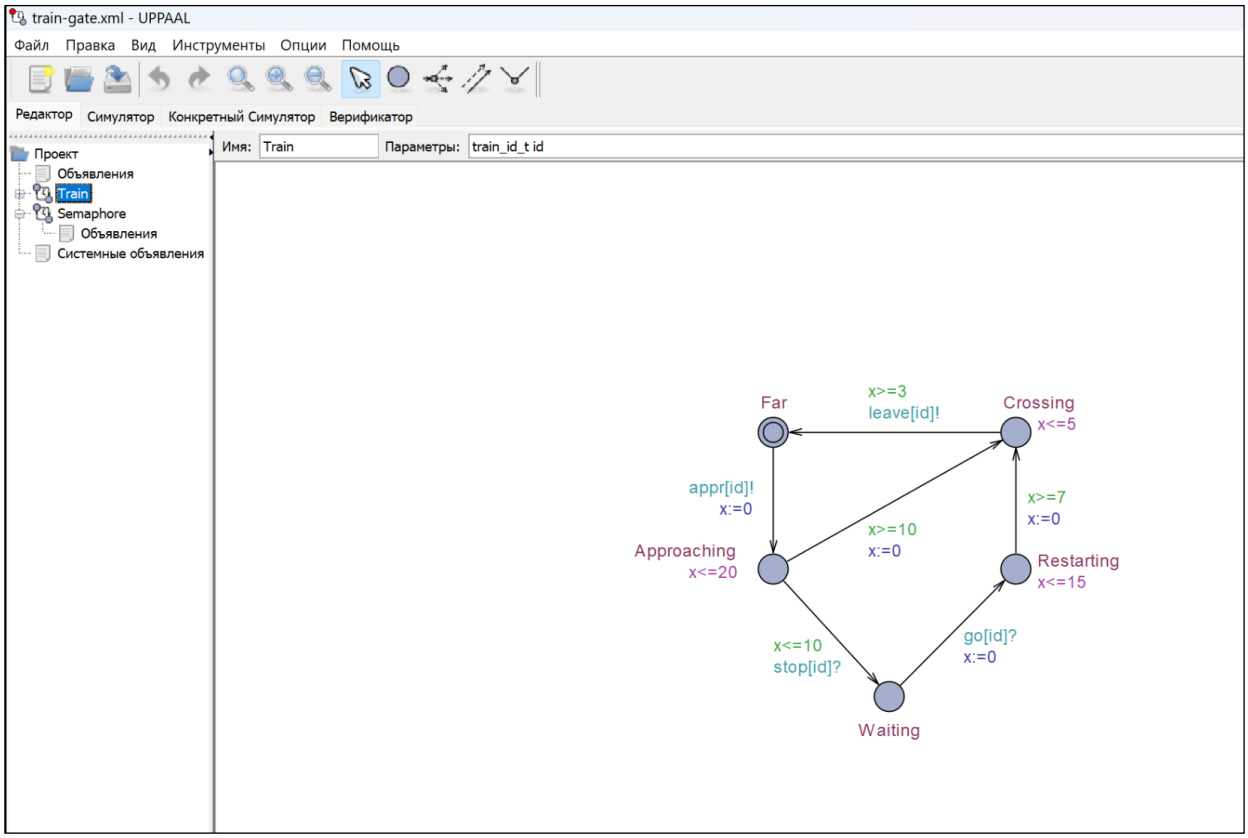


Рисунок 14. Шаблон Train.

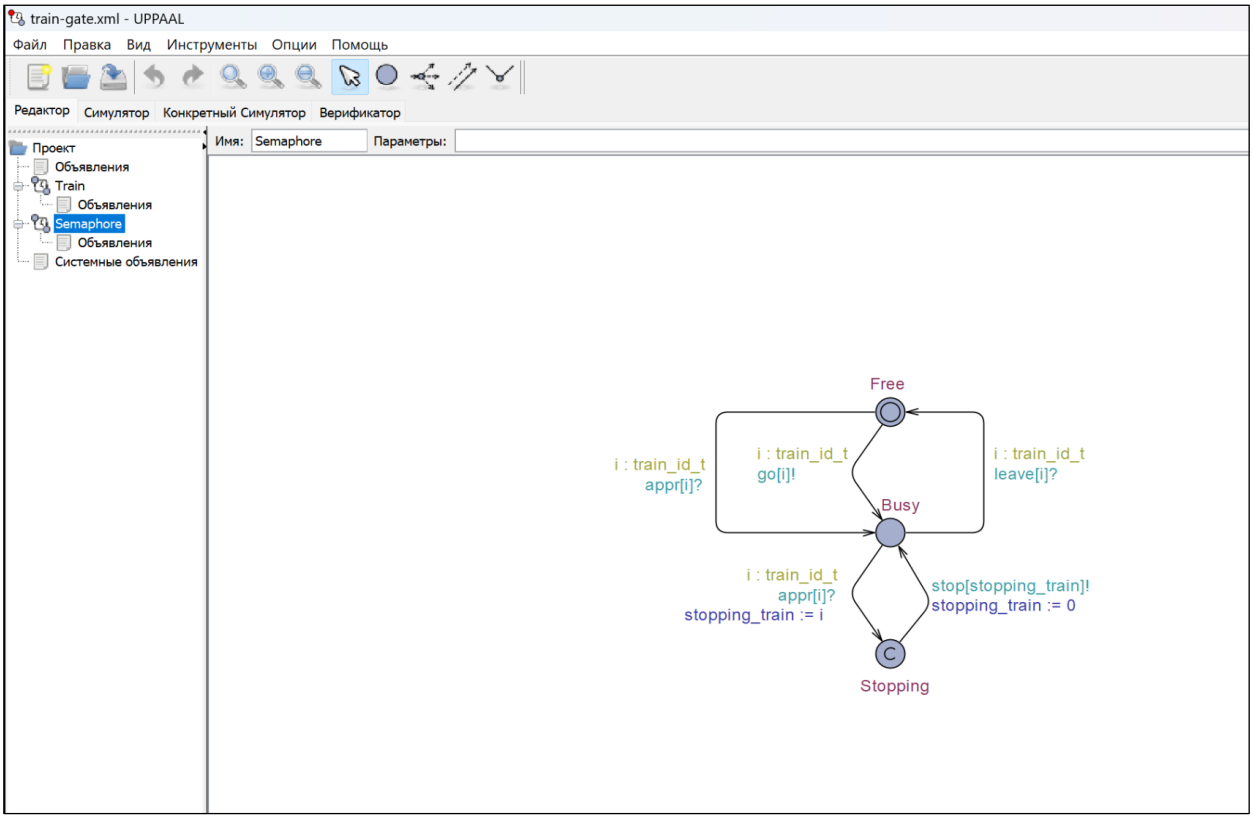


Рисунок 15. Шаблон Semaphore.

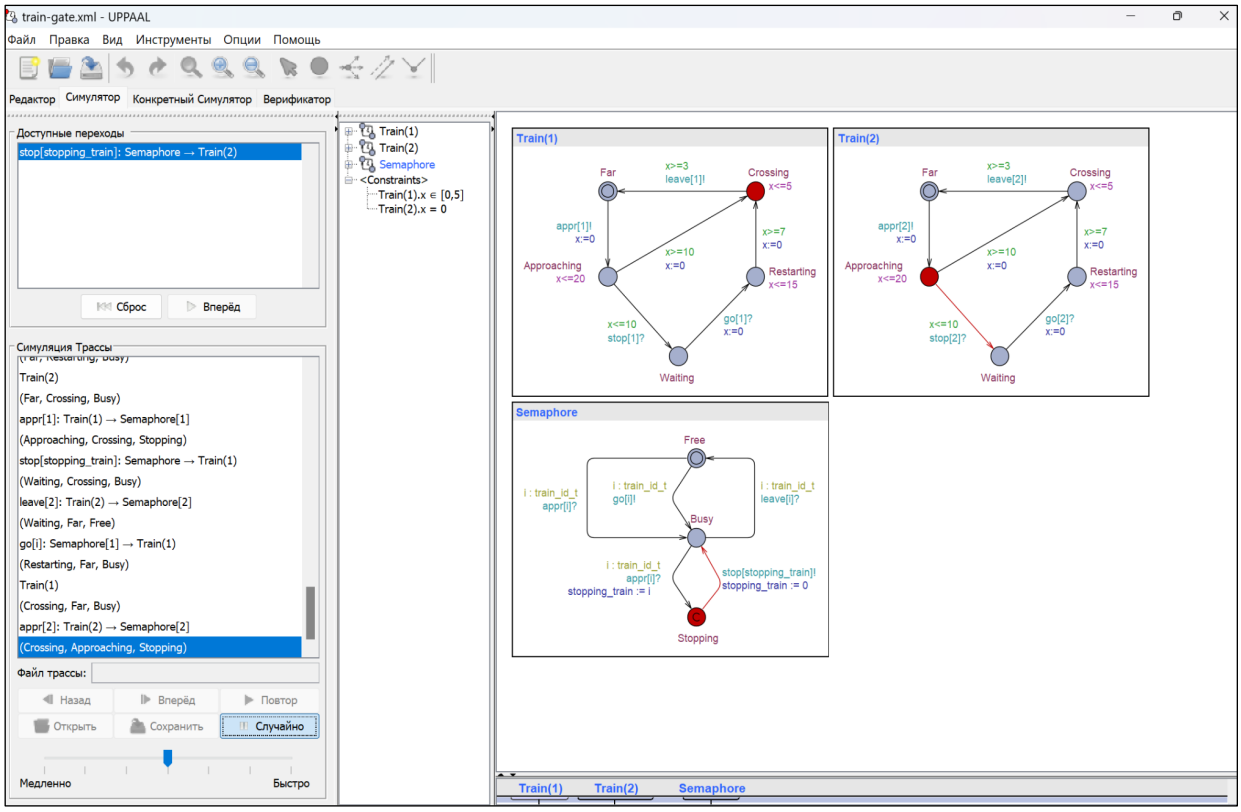


Рисунок 16. Окно симуляции.

Мы хотим проверить следующие свойства:

- Отсутствие взаимоблокировки: система не может быть заблокирована: A[] not deadlock.
- Нет столкновения поездов: A[] forall (i : train_id_t) forall (j : train_id_t) (Train(i).Crossing and Train(j).Crossing imply i == j).
- Всякий раз, когда приближается поезд 1, он в конце концов пересекает мост: Train(1).Approaching --> Train(1).Crossing.

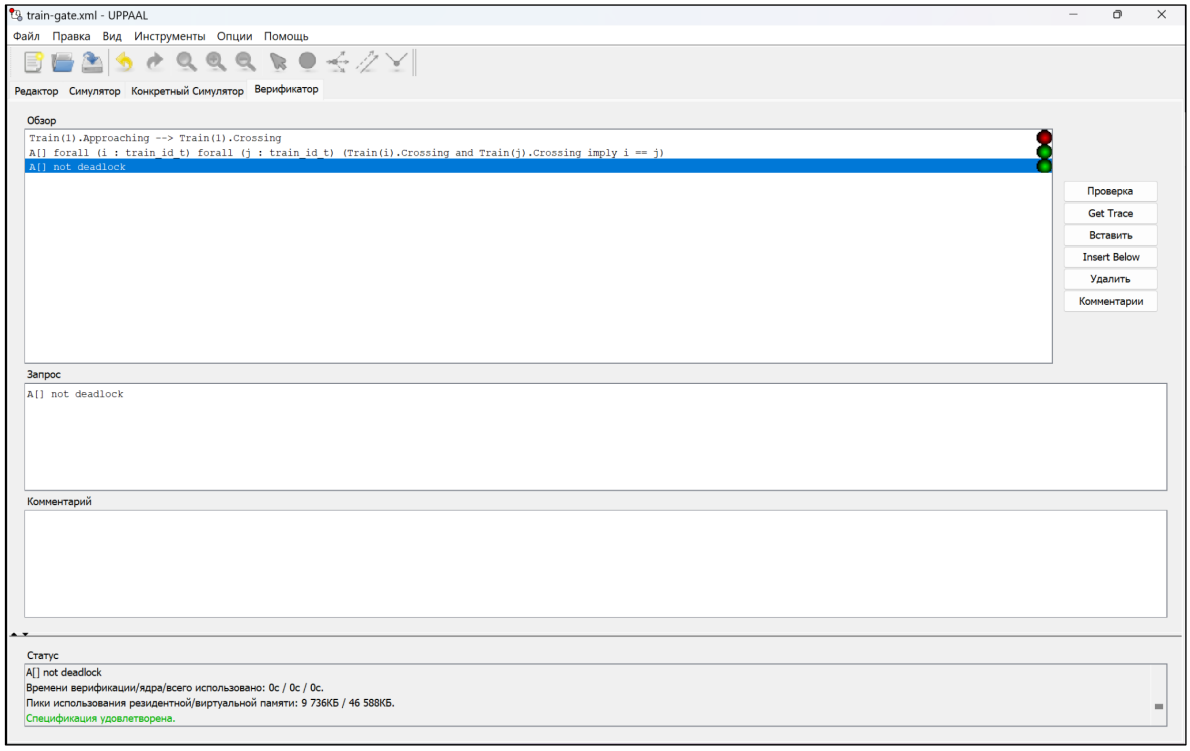


Рисунок 17. Окно верификации.

Из результатов верификации на *рисунке 17* следует, что свойство Train(1).Approaching --> Train(1).Crossing **не выполняется**.

Литература

1. Alexandre David, Kim G. Larsen, Axel Legay, Marius Mikuˇcionis, Danny Bøgsted Poulsen. 2012. Chapter 4. UPPAAL-SMC. //Runtime Verification of Biological Systems.
2. Gerd Behrmann, Alexandre David, and Kim G. Larsen. 2004. Chapter 4. Overview of the Uppaal Toolkit. // A Tutorial on Uppaal.
3. Логики CTL и TCTL – Особенности UPPAAL [Электронный ресурс]. URL: https://lvk.cs.msu.ru/~dimawolf/SoftwareReliability/Lection05.pdf (дата обращения 03.03.2023).
4. UPPAAL | Downloads [Электронный ресурс]. URL: https://uppaal.org/downloads/ (дата обращения 07.03.2023).
5. Introduction to formal verification with UPPAAL [Электронный ресурс]. URL: https://www.labri.fr/perso/herbrete/etr-2021/etr-2021-tp.html (дата обращения 07.03.2023).
6. UPPAAL | System definition [Электронный ресурс]. URL: https://docs.uppaal.org/language-reference/system-description/system-definition/ (дата обращения 10.03.2023).