



**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ**

Брянский государственный технический университет

Утверждаю

Ректор университета

_____ О.Н. Федонин

«_____» _____ 2019г.

КОМПЬЮТЕРНОЕ МОДЕЛИРОВАНИЕ

**ПОСТРОЕНИЕ МОДЕЛЕЙ ДИСКРЕТНО-СОБЫТИЙНЫХ
СИСТЕМ В ПРОГРАММНОМ КОМПЛЕКСЕ ANYLOGIC**

Методические указания
к выполнению лабораторной работы №9
для студентов очной формы обучения
по направлению подготовки
09.03.01 «Информатика и вычислительная техника»

Брянск 2019

УДК 004.65

Компьютерное моделирование. Построение моделей дискретно-событийных систем в программном комплексе Anylogic [Электронный ресурс]: методические указания к выполнению лабораторной работы № 9 для студентов очной формы обучения по направлению подготовки 09.03.01 «Информатика и вычислительная техника». – Брянск: БГТУ, 2019. – 21 с.

Разработала
А.А.Трубакова,
асс.

Рекомендовано кафедрой «Информатика и программное обеспечение»
БГТУ (протокол № 4 от 24.12.2018г.)

Методические указания публикуются в авторской редакции

1. ЦЕЛЬ РАБОТЫ

Целью работы является ознакомление со средой имитационной моделирования *AnyLogic* и основными принципами построения моделей дискретно-событийных систем на примере модели светофора, регулирующего движение на пешеходном переходе.

Продолжительность работы – 2 часа.

2. ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ

Изучение основных возможностей и принципов работы в среде имитационного моделирования *AnyLogic*.

Изучение возможных состояний светофора и событий, происходящих в системе.

Построение модели работы светофора в среде *AnyLogic*.

Самостоятельное построение имитационной модели светофора, регулирующего движение автотранспорта на перекрестке.

3. ТЕОРЕТИЧЕСКАЯ ЧАСТЬ

3.1. Реагирующие системы и стейтчарты

Реагирующую систему (*reactive system*) А. Пнуэли определил, как систему, постоянно ожидающую внешних или внутренних событий и реагирующую на них. Реагирующие системы являются типичными системами дискретно-событийного типа: события происходят в дискретные моменты времени, реакция системы на события состоит в изменении переменных состояния этих систем, и формально такая реакция является мгновенной.

При задании реагирующих систем оказалось удобным использовать стейтчарты (или карты состояний). Стейтчарты представляют собой графический язык диаграмм (фактически, расширение обычных графов

переходов). Язык стейтчартов, впервые введенный Д. Харелом, в настоящее время широко применяется для спецификации, моделирования, верификации и прототипирования протоколов коммуникации, встроенных систем управления в авиации, на транспорте, в бытовой и научной электронике. Они также являются средством определения поведения объектов в *UML*.

Стейтчарты строятся из состояний и переходов между ними. Система может находиться в каждый момент только в одном состоянии. Переходы из состояния в состояние случаются, если происходит событие, связанное с этим переходом, и условие, связанное с переходом (если оно есть), выполнено. На диаграмме (рис. 1) система из состояния *A* переходит в состояние *B*, если наступит событие *A* и при этом условие *P* будет выполнено.

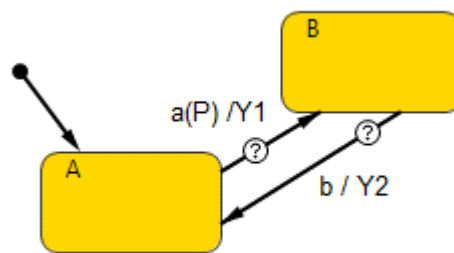


Рис. 1.

Рис. 1. Простой граф переходов

Событием может быть, например, истечение таймаута, переключение в истину предиката (условия), определенного на переменных модели, и т. п. Графически состояния представляются прямоугольниками или овалами, а переходы – дугами. Короткая стрелка-указатель, входящая в состояние *A*, говорит о том, что это *состояние начальное*: в начальный момент времени система будет находиться именно в этом состоянии. Очевидно, что у системы может быть ровно одно начальное состояние. С каждым переходом может быть связано некоторое действие:

- изменение переменных;
- получение сообщения;
- срабатывание таймаута;

- прибытие агента и т. п.

С каждым состоянием также могут быть связаны действия. Одно действие выполняется в момент входа в это состояние, другое действие выполняется при выходе из состояния. На рис. 1 $Y1$ и $Y2$ условно обозначены действия, выполняемые при срабатывании соответствующих переходов.

Рис. 1 представляет простейший стейтchart с двумя элементарными состояниями. В общем случае в стейтчартах можно дополнительно использовать расширения этой простейшей модели переходов:

- 1) иерархические состояния (гиперсостояния);
- 2) исторические состояния;
- 3) условные переходы и некоторые другие возможности.

Рассмотрим их по порядку.

Иерархические состояния или *гиперсостояния* вводятся для того, чтобы объединить несколько состояний, имеющих одну и ту же реакцию на событие. На рис. 2 справа гиперсостояние D позволяет упростить граф переходов, представленный слева: переход в состояние A при наступлении события b происходит вне зависимости от того, в каком из состояний (B или C) находилась система. Иными словами, два стейтчарта на рис. 2 эквивалентны.

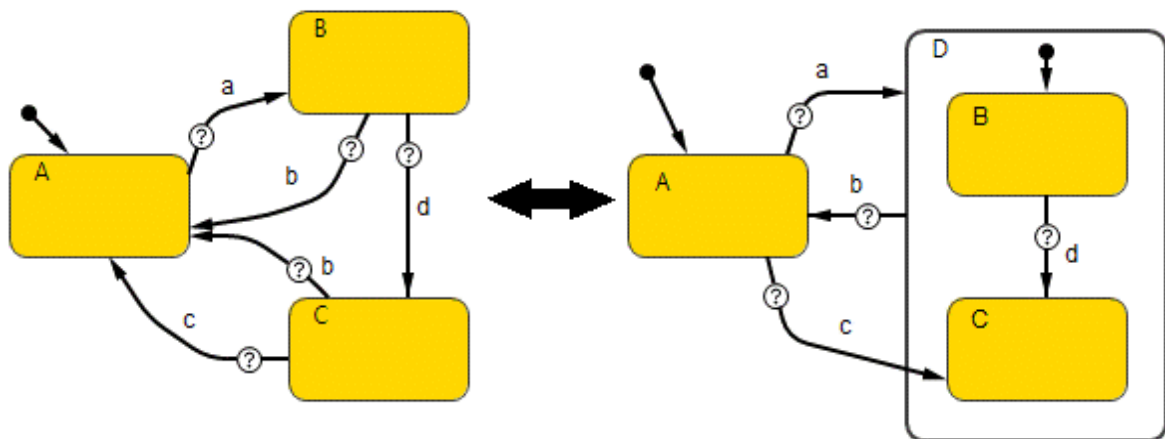


Рис. 2. Упрощение графа переходов с использованием гиперсостояния

Каждое гиперсостояние требует, чтобы точно одно из включенных в него состояний было помечено, как начальное. Это позволяет трактовать переход из состояния A при наступлении события a в гиперсостояние D , как переход

из A в элементарное состояние B . На рис. 2 начальным состоянием системы является состояние A , а состояние B является начальным только для множества состояний $\{B, C\}$, входящих в гиперсостояние D .

Историческое состояние хранит то состояние внутри данного гиперсостояния, в котором система находилась последний раз.

На рис. 3 при наступлении события a система вернется в то состояние из множества состояний $\{A, B, C\}$, в котором она была последний раз (независимо от того, какими переходами связаны эти состояния). Исторические состояния удобны, например, для описания продолжения функционирования системы после прерываний.

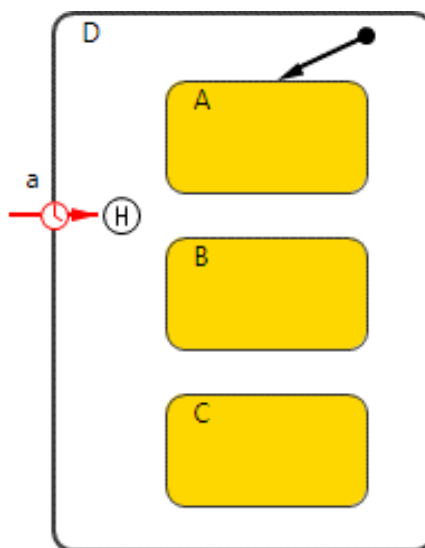


Рис. 3. Историческое состояние

Условные состояния позволяют отложить проверку логического условия. Такая отложенная проверка удобна, например, в том случае, если определить дальнейшие действия системы можно только после реакции на событие. Например, пусть событием является приход сообщения, а реакция на него зависит от содержания этого сообщения. На рис. 4 представлен фрагмент стейтчарта, описывающего эту ситуацию. По появлении сообщения (событие a), если система находилась в состоянии A , то она перейдет в состояние C только в том случае, если условие P выполнено. Если условие P не выполнено, система перейдет в состояние D . В этом примере удобно использовать именно условное состояние, в котором система не

задерживается, а после приема сообщения мгновенно определяет, в какое состояние перейти.

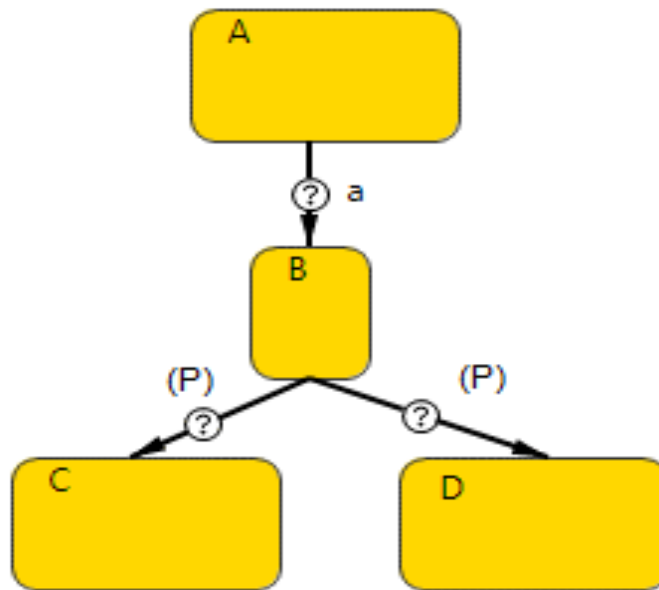


Рис. 4. Условное состояние

4. ПОСТРОЕНИЕ МОДЕЛИ СВЕТОФОРА

4.1. Постановка проблемы

Светофор, регулирующий движение транспорта на пешеходном переходе, может находиться в следующих состояниях: разрешение движения транспорта (зеленый), приготовиться к запрещающему сигналу (мигающий зеленый), приготовиться к остановке (желтый), запрет движения (красный) и приготовиться к движению (красный и желтый) (рис. 5).

Светофор работает в автоматическом режиме, циклически. В каждом состоянии светофор находится определенный постоянный период времени.

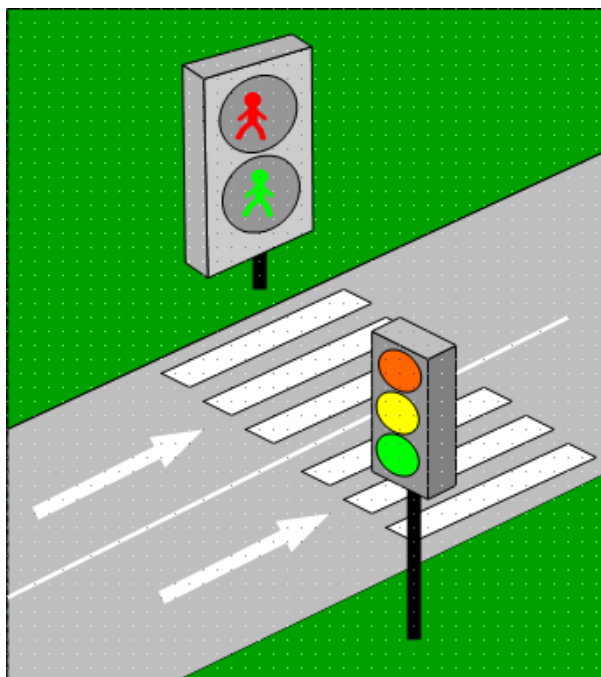


Рис. 5. Простейший регулируемый пешеходный переход

4.2. Построение модели

Создайте новый проект под названием *PedestrianCross* и назовите класс корневого активного объекта *Model*.

Наша модель будет иметь только один активный объект, представляющий светофор, поэтому корневой объект *Model* будет единственным активным объектом нашей модели. На диаграмму класса активного объекта *Model* поместите **Начало диаграммы состояний** из панели *Диаграмма состояний*, заметьте, что *AnyLogic* может работать с элементами, набранными кириллицей. Справа появится окно свойств этого нового объекта, в котором в поле имени будет стоять предопределенное имя *p0*. Замените это имя на *traffic_light*.

После двойного щелчка мыши на иконке стейтчарта *traffic_light* откроется окно редактора этого стейтчарта с уже введенным одним состоянием (с именем *p0*) с входящей в него стрелкой, показывающей, что это *начальное состояние*. Имя состояния, как и все другие его параметры, можно редактировать в окне его свойств либо при выделенном состоянии можно нажать клавишу *<F2>*, что позволяет выделить имя в самом состоянии для его редактирования.

Для того чтобы построить стейтchart, следует использовать кнопки (рис. 6) из раздела *Диаграмма состояний* в палитре.

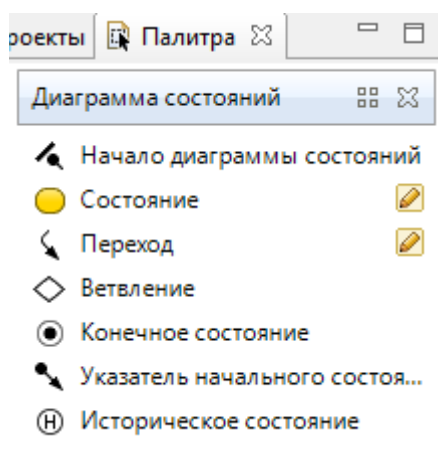






Рис. 6. Палитра раздела диаграммы состояний

С помощью различных кнопок рисуется стейтchart. Назначение каждой из них описано ниже:

- кнопка  **Состояние** рисует как простое состояния, так и гиперсостояние;
- кнопка  **Переход** используется для рисования переходов между состояниями;
- кнопка  **Указатель начального состоя...** определяет начальное состояние как всего стейтchartа, так и в каждом гиперсостоянии;
- кнопка  **Конечное состояние** используется для рисования состояния, являющегося «финальным» в поведении активного объекта. Графически такое состояние удобно изображать специальным символом, хотя конкретные действия, определяющие, что нужно делать в этом финальном состоянии, разработчик должен добавить сам.
- Комментарии, как обычно, помещаются на поле редактора с помощью кнопки **Да Текст** (раздел *Презентация* из палитры инструментов).

Заметьте, что для любого выделенного объекта справа появляется окно его свойств, в котором можно изменить параметры и, в частности, имя объекта, если это необходимо. Структурные ошибки при рисовании стейтчарта – повисшие переходы, дублированные указатели начального состояния и т. п. – выделяются в поле редактора красным цветом. Чтобы имя объекта – состояния или перехода – появилось в поле редактора, следует выбрать опцию *Отображать имя* в нижней части окна свойств объекта.

В соответствии с алгоритмом работы светофора кроме начального состояния в модель нужно ввести дополнительные состояния (рис. 7). Начальное состояние назовите *go* (движение транспорту разрешено – горит зеленый), затем светофор переходит в состояния *attention* (внимание – мигающий зеленый), *slow* (приготовиться к остановке – горит желтый), остановка транспорта *stop* (запрет движения – горит красный) и *ready* (приготовиться к движению – горят красный и желтый). Состояние *attention* удобно представить гиперсостоянием с парой переключающихся элементарных состояний: в одном из них зеленый горит (состояние *A*), в другом – нет (состояние *B*). Постройте все эти состояния и соедините их соответствующими переходами.



Рис. 7. Стейтчарт светофора

Рассмотрим, как задать условия срабатывания переходов. Переходы в нашем автоматическом светофоре выполняются по таймауту, т. е. по истечении интервала времени, который прошёл с момента прихода системы в данное состояние. Пусть в состоянии *go* светофор должен находиться 25 с, затем 7 с зелёный сигнал мигает, 4 с горит жёлтый в состоянии *slow*, в течение 20 с движение запрещено и 4 с светофор находится в состоянии *ready*. Масштаб времени примем такой: единица модельного времени соответствует 1 с реального физического времени.

Для того чтобы задать такие условия срабатывания переходов, сделайте активным переход *t1*, в поле свойств *Происходит* выберите вариант *По таймауту*, а в поле Таймаут введите 25. Аналогично задайте условия

срабатывания других переходов. Между состояниями А и В пусть переходы срабатывают через 1 единицу времени (1 с горит зеленый свет, затем 1 с не горит). Заметьте, что переход срабатывает (в модельном времени) мгновенно.

Запустите модель на выполнение. Чтобы можно было наблюдать переходы между состояниями стейтчарта при работе модели, откройте окно стейтчарта двойным щелчком левой кнопки мыши на его изображении в окне корневого объекта модели с именем *root*. Активное в данный момент состояние подсвечивается красным. Проведите эксперименты с моделью при различных масштабах времени.

В каждом состоянии светофора должен гореть вполне определенный сигнал: в состоянии *go* должен гореть зеленый, в состоянии *ready* должны гореть красный и желтый одновременно и т. п. Откройте окно редактора структуры объекта *Model*. Определите три переменные логического (*boolean*) типа *red*, *yellow* и *green*, которые будут принимать истинное значение тогда, когда у светофора должен гореть соответствующий «глаз»: красный, желтый и зеленый (рис. 8). Начальные значения этих булевых переменных можно не задавать: по умолчанию они будут равны *false*.

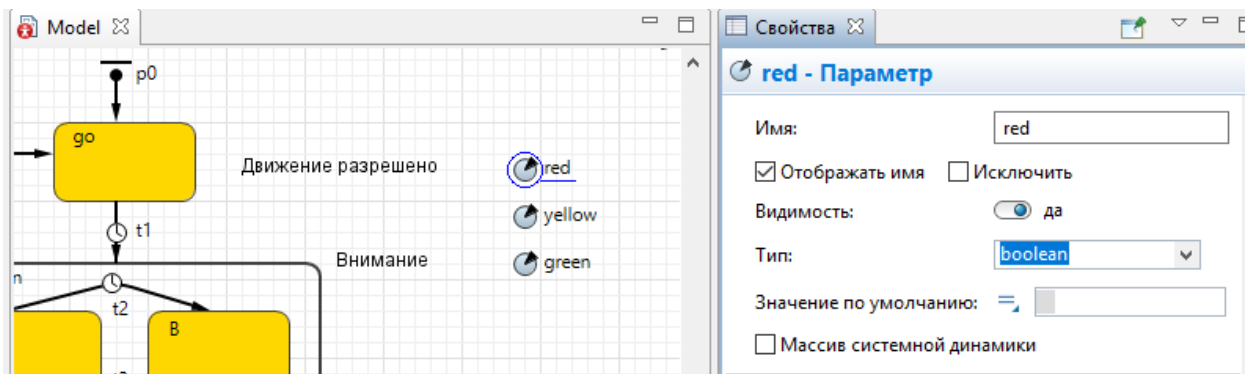


Рис. 8. Введение булевых переменных

Стейтчарт построен именно для управления значениями этих переменных, каждое состояние отвечает за зажигание своего света или комбинации светов. Например, в состоянии *go* должен гореть зеленый, при входе в состояние *stop* должен загореться красный свет (а остальные гореть не должны), а в состоянии *ready* должны гореть красный и желтый и т. п. Именно это мы и должны определить.

Откройте окно свойств состояния *go* и в поле *Действие при входе* запишите *green=true;*, а в поле *Действие при выходе* запишите *green=false;*. То же самое нужно определить для состояния *B*, а у состояния *A* эти поля нужно оставить без изменения – когда светофор находится в этом состоянии, он вообще не горит (все переменные имеют значения *false*). Аналогично, в состоянии *slow* нужно включить желтый сигнал, т. е. *при входе* в это состояние установить переменную *yellow* в *true*, а *при выходе* из этого состояния установить ее в *false*. Для состояния *stop* то же нужно сделать с переменной *red*, а для состояния *ready* следует обе переменные – *red* и *yellow* – установить в *true* *при входе* и установить в *false* *при выходе* из него.

Запустите модель на выполнение при различных масштабах времени. В окне *root* в дереве переменных и параметров модели переменные *green*, *yellow* и *red* будут переключаться между значениями истина и ложь (представляемыми здесь 1 и 0) в соответствии с алгоритмом переключения светофора.

4.3. Создание анимации

Анимация для этой модели весьма просто строится средствами *AnyLogic*. Все графические объекты в анимации имеют статические характеристики, кроме цвета сигналов светофора. Светофор строится из трех эллипсов и дополнительных элементов (скругленный прямоугольник, линии). Динамическое значение цвета верхнего сигнала светофора необходимо установить так: если переменная *red* истинна, то цвет должен быть *Color.red*, в противном случае его цвет нужно установить *Color.black* (серый). Это записывается следующим выражением – также фрагментом кода языка *Java* в *Свойствах* овалов (динамические значения, определенные в параметрах внешнего вида цвета и заливки):

red? Color.red : Color.black

Цвет среднего и нижнего овалов, представляющих сигналы светофора, следует установить в поле их динамических значений соответственно так:

yellow? Color.yellow : Color.black

green? Color.green : Color.black

здесь *Color* – класс *Java*, а *black* – предопределенная в этом классе константа, обозначающая чёрный цвет.

4.4. Срабатывание перехода по сигналу

Добавим к модели пешеходного перехода второй светофор, для пешеходов. Он имеет только два света, зеленый и красный, и три состояния: разрешающее переход (зеленый), внимание (мигающий зеленый) и запрещающее переход (красный). В модель добавим еще две булевские переменные *greenP* и *redP*, их значения будут устанавливаться в состояниях еще одного стейтчарта, который будет управлять светофором пешеходов. Добавим новый стейтчарт в окне редактора объекта *Model*, назвав его *pedestrian_light*.

Поскольку управление светофором пешеходов похоже на управление светофором трафика, новый стейтчарт можно построить изменением уже построенного стейтчарта *traffic_light*. Измените имя стейтчарта на *pedestrian_light* и отредактируйте сам стейтчарт так, чтобы в нем были только состояния *go1*, *attention1* и *stop1*.

Изменения, которые должны быть проведены с этим стейтчартом, такие: начальное состояние для светофора пешеходов должно быть *stop1* – противоположным начальному состоянию светофора трафика; нужно удалить состояния *slow* и *ready*.

В результате новый стейтчарт должен иметь вид (рис. 9).

Установка переменных *при входе* и *выходе* из состояний этого стейтчарта должна, конечно, относиться к переменным *redP* и *greenP*, управляющим зажиганием света именно пешеходного светофора.

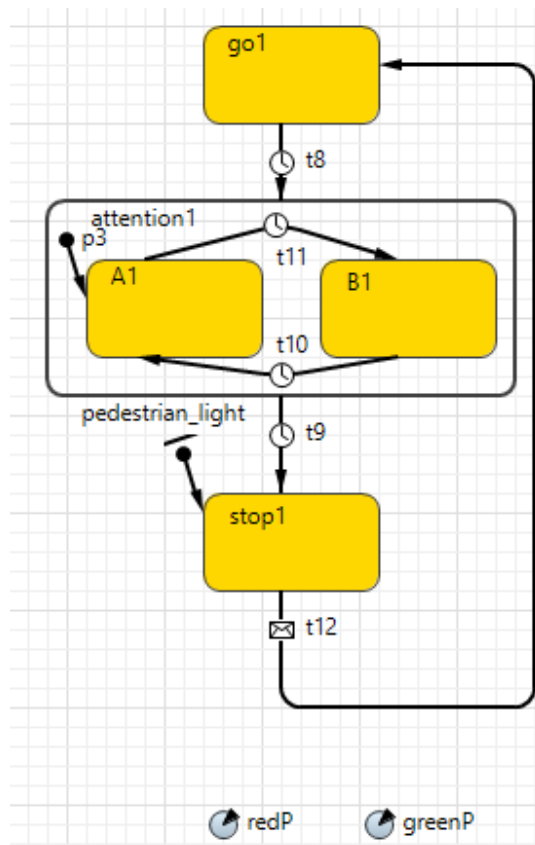


Рис. 9. Стейтчарт пешеходного светофора

Рассмотрим теперь условия срабатывания переходов стейтчартов между состояниями. Очевидно, что необходимо корректно синхронизировать срабатывания переходов двух стейтчартов так, чтобы всегда, когда светофор пешеходов находится в состояниях *go1* или *attention1*, светофор трафика обязательно находился бы в состоянии *stop*. Этого можно добиться подбором таймаутов срабатывания переходов. Однако более разумно сделать это, посылая специальные разрешающие сигналы из одного стейтчарта в другой (рис. 6).

Введем два сигнала: *ТРАФФИК* и *ПЕШЕХОДЫ*. Пусть в стейтчарте управления светофором пешеходов переход *t12* может сработать, только если получен сигнал *ПЕШЕХОДЫ*, который будет генерироваться в стейтчарте управления трафиком при переходе *t5* в состояние, в котором запрещено движение транспорта. Пусть в стейтчарте *traffic_light* переход *t6* может сработать, только если получен сигнал *ТРАФФИК*, который генерируется в

стейтчарте управления движением пешеходов при переходе t_9 в состояние, запрещающее движение пешеходов.

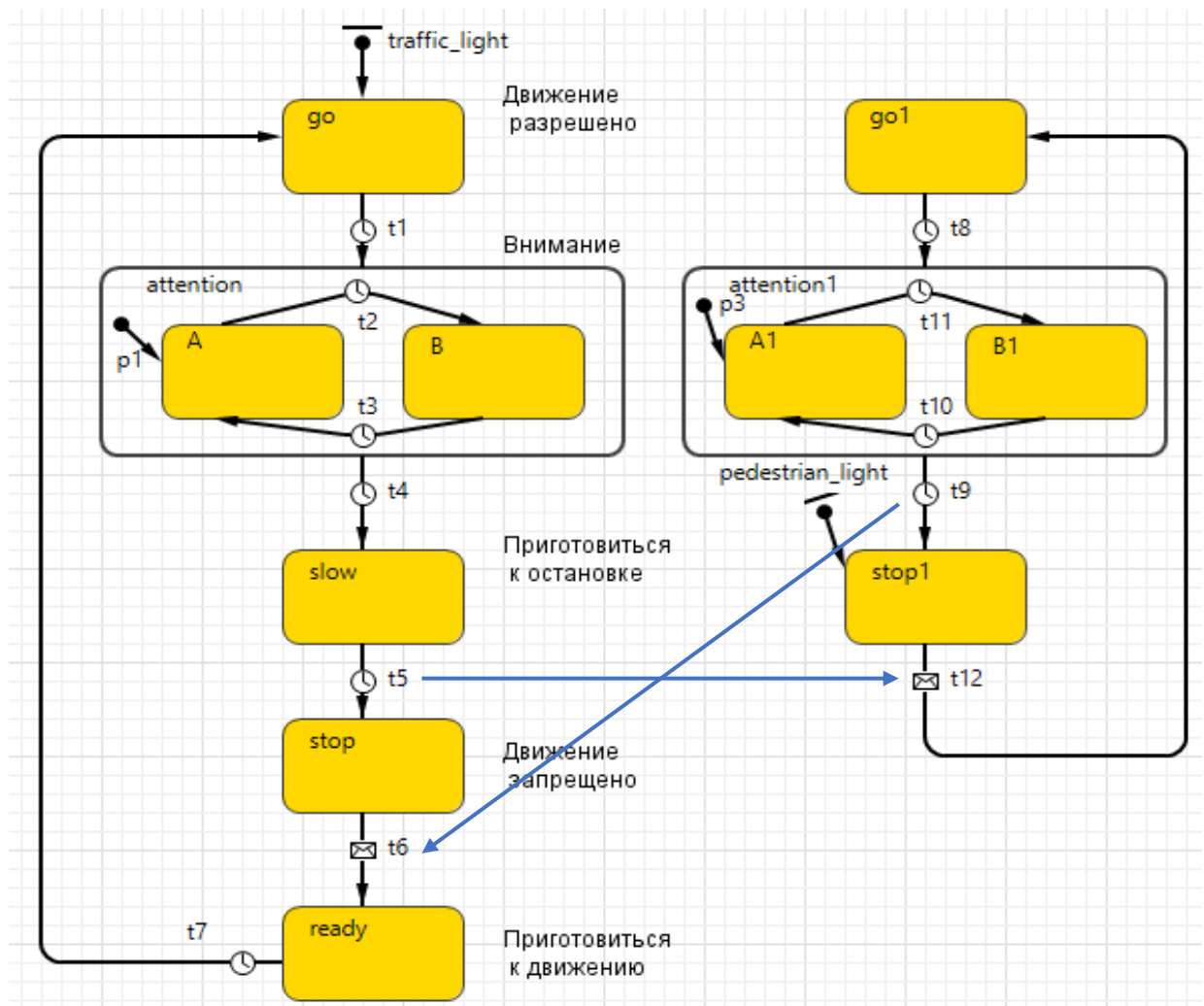


Рис. 10. Синхронизация поведений при помощи сигналов

В *AnyLogic* для генерации сигналов существует функция `fireEvent(<сигнал>)`, которая должна вызываться в том стейтчарте, которому предназначен сигнал. Если из некоторого активного объекта мы хотим послать (произвольный) сигнал *AAA* стейтчарту с именем *stchart*, то необходимо в этом активном объекте вызвать функцию `fireEvent (<сигнал>)` этого стейтчарта, т. е. выполнить действие `stchart.fireEvent ("AAA")`. Если стейтчарт *stchart* находится в состоянии, в котором он ожидает этот сигнал, то соответствующий переход сработает. Если стейтчарт находится в состоянии, в котором он не ожидает этого сигнала, никакой реакции не будет, а полученный сигнал потеряется. В состоянии *s* стейтчарт ожидает сигнала *AAA*,

если в поле *Происходит* окна свойств одного из переходов из состояния *s* выбран вариант *При получении сообщения*, а в поле *Сообщение* этого окна указан этот сигнал AAA.

В нашей модели в поле *Действие* перехода *t5* стейтчарта *traffic_light* вставьте команду *pedestrian_light.fireEvent("ПЕШЕХОДЫ")*; а в это же поле перехода *t9* стейтчарта *pedestrian_light* вставьте команду *traffic_light.fireEvent("ТРАФФИК")*;

Таким образом, каждый из светофоров будет информировать другого о своем переходе в состояние запрещения движения. Для перехода по разрешающему сигналу в стейтчарте *traffic_light* в поле *Происходит* окна свойств перехода *t6* выберите вариант *При получении сообщения*, свойство *Осуществлять переход* задайте *При получении заданного сообщения*, а в поле *Сообщение* этого же окна наберите сигнал "ТРАФФИК". Для перехода по разрешающему сигналу в стейтчарте *pedestrian_light* в поле *Происходит* окна свойств перехода *t12* выберите вариант *При получении сообщения*, а в поле *Сообщение* этого же окна наберите сигнал "ПЕШЕХОДЫ". Остальные переходы этих стейтчартов пусть срабатывают по таймаутам. Рис. 10 показывает, как задать параметры переходов стейтчарта *traffic_light*. Проверьте по рис. 11 установленные параметры переходов стейтчартов. Запустите модель на выполнение.

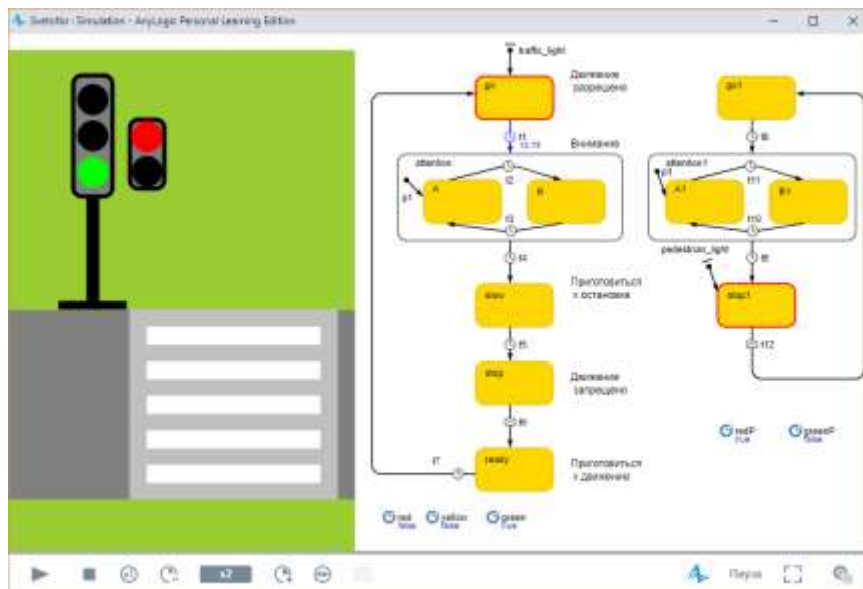


Рис. 11. Результат моделирования движения на пешеходном переходе

5. ЗАДАНИЕ ДЛЯ САМОСТОЯТЕЛЬНОГО ВЫПОЛНЕНИЯ

1. Дополните созданную модель до светофора, регулирующего движение на перекрестке с пешеходным переходом (по сути в модели должно получиться три светофора: 2 для регулирования движения автотранспорта и один для пешеходов).
2. Измените модель светофора с кнопкой таким образом, чтобы моделировалось автоматическое нажатие кнопки *ПЕРЕХОД* в случайные моменты времени в интервале от 5 секунд до 1 минуты.
3. Измените модель автоматического светофора таким образом, чтобы по кнопке *НОЧЬ* включался режим мигания желтого света, а по кнопке *ДЕНЬ* светофор переходил в нормальный режим работы.

Формой отчета по данной лабораторной работе является построенная в системе *AnyLogic* модель и ее тестирование.

6. КОНТРОЛЬНЫЕ ВОПРОСЫ

СПИСОК РЕКОМЕНДУЕМОЙ ЛИТЕРАТУРЫ

1. Боев, В.Д. Компьютерное моделирование [Электронный ресурс] / В.Д. Боев, Р.П. Сыпченко. – 2-е изд. – М.: Интернет-Университет Информационных Технологий (ИНТУИТ), 2016. – 525 с. – Режим доступа: <http://www.iprbookshop.ru/73655.html>.
2. Ашихмин, В.Н. Введение в математическое моделирование [Электронный ресурс]: учебное пособие / В.Н. Ашихмин [и др.]. – М.: Логос, 2016. – 440 с. – Режим доступа: <http://www.iprbookshop.ru/66414.html>.
3. Тупик, Н.В. Компьютерное моделирование [Электронный ресурс]: учебное пособие / Н.В. Тупик. – 2-е изд. – Саратов: Вузовское образование, 2019. – 230 с. – Режим доступа: <http://www.iprbookshop.ru/79639.html>.
4. Осоргин, А.Е. AnyLogic 6. Лабораторный практикум. / А.Е. Осоргин. – Самара: ПГК, 2011. – 100 с.
- 5.

Компьютерное моделирование. Построение моделей дискретно-событийных систем в программном комплексе Anylogic: методические указания к выполнению лабораторной работы № 9 для студентов очной формы обучения по направлению подготовки 09.03.01 «Информатика и вычислительная техника»

ТРУБАКОВА АННА АЛЕКСЕЕВНА

Научный редактор Д. А. Коростелев
Компьютерный набор А.А. Трубакова
Иллюстрации А.А. Трубакова

Подписано в печать __.__.__. Усл.печ.л. 0,58 Уч.-изд.л. 0,58

Брянский государственный технический университет
241035, Брянск, бульвар 50 лет Октября, 7 БГТУ
Кафедра «Информатика и программное обеспечение», тел. 56-09-84

Сопроводительный лист на издание в авторской редакции

Название работы Компьютерное моделирование. Построение моделей дискретно-событийных систем в программном комплексе Anylogic: методические указания к выполнению лабораторной работы № 9 для студентов очной формы обучения по направлению подготовки 09.03.01 «Информатика и вычислительная техника»

Актуальность и соответствующий научно-методический уровень подтверждаю _____

(подпись научного редактора)

Рукопись сверена и проверена автором _____

(подпись автора)

Рекомендуется к изданию _____

(подпись заведующего кафедрой)