Курсовая работа. Разработка контроллера светофоров на перекрестке и его верификация

В курсовой работе ставится задача разработки контроллера светофоров для управления проездом перекрестка. Каждое направление движения транспорта на перекрестке регулируется своим светофором. Датчики, установленные на соответствующих полосах, фиксируют наличие автомобилей в каждом направлении. В случае отсутствия автомобилей в некотором направлении, соответствующий светофор горит красным. При появлении автомобилей на полосе, процесс управления светофором соответствующего направления начинает борьбу за ресурсы. Ресурсами в этой задачи являются области перекрестка, в которых возможны пересечения данного направления движения с другими направлениями, его пересекающими. Если процесс получает нужные ресурсы, то он переключает свой светофор на зеленый, разрешая движение транспорта в этом направлении. Ядром работы является построение корректных алгоритмов захвата и освобождения ресурсов в процессах управления светофорами.

Выполнение курсовой работы состоит в разработке контроллера, выполняющего описанную выше задачу управления движением транспорта по нескольким пересекающимся направлениям на сложном перекрестке. Контроллер состоит из нескольких параллельных процессов (по одному для каждого направления) и глобальных разделяемых переменных, которые могут модифицироваться (устанавливаться) и читаться этими процессами. Каждый процесс управляет своим светофором, разрешающим или запрещающим движением по перекрестку в конкретном направлении. В число локальных переменных процесса входит переменная, значение которой – разрешающий (зеленый) или запрещающий (красный) свет связанного с процессом светофора. Глобальные переменные используются для реализации алгоритма синхронизации процессов, гарантирующего корректное использование процессами общих разделяемых ресурсов.

Разработку алгоритмов управления светофорами предлагается выполнить аналогично тому, как это сделано ниже в п. 2 пособия для простейшего перекрестка. В п.3 пособия представлена схема более сложного перекрестка с несколькими пересекающимися направлениями движения. Каждому студенту дается индивидуальное задание, содержащее только несколько направлений движения. Другие направления движения по этому перекрестку в его задании можно игнорировать.

Рассматривается задача разработки контроллера светофоров для управления проездом автомобилей перекрестка. Датчики, установленные в каждом направлении, фиксируют наличие автомобилей. В случае отсутствия автомобилей все светофоры горят красным светом. При появлении автомобилей светофоры начинают борьбу за ресурс – перекресток. По выигравшему направлению светофор переключается на зеленый сигнал – машины начинают соответствующее движение. Необходимо составить алгоритм работы контроллера светофоров всего перекрестка (состоящего из нескольких параллельных взаимодействующих процессов) и проверить его корректность относительно заданных свойств безопасности и живости.

Для выполнения курсовой работы необходимо, отталкиваясь от простейшего алгоритма (п. 1), построить модель работы контроллера светофоров на языке Promela для индивидуального плана проезда (п. 2) и верифицировать полученную модель средствами SPIN, в случае наличия ошибок изменить алгоритм работы контроллера так, чтобы он работал корректно.

1 Алгоритмы управления светофорами простого перекрестка

При выполнении курсовой работы студент может руководствоваться примером разработки контроллера простого перекрестка, который представлен в этом разделе.

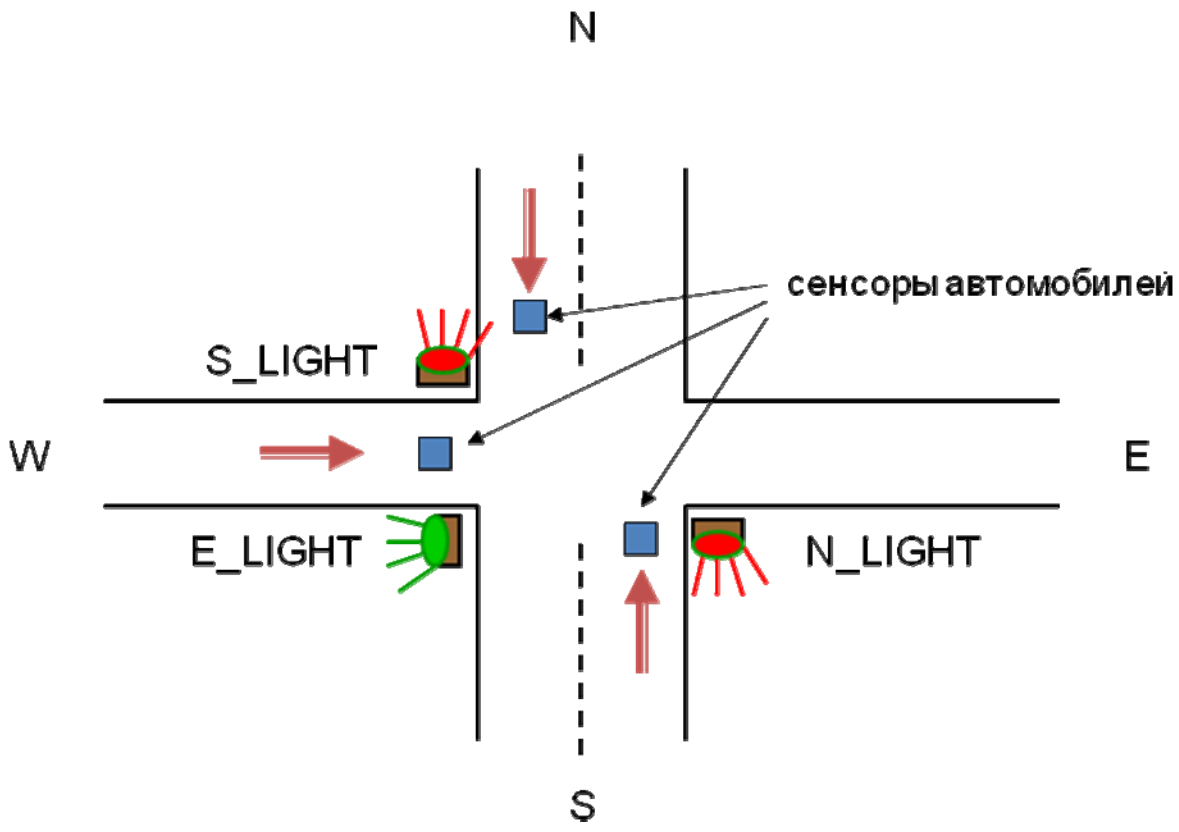

Рис. 1. Схема проезда простого перекрестка.

Построим контроллер, управляющий светофорами перекрестка рис. 1 в направлениях N (на север), S (на юг) и E (на восток). Внешние события N_SENSE, S_SENSE и E_SENSE указывают, что сенсор данного направления обнаружил автомобиль, желающий получить право проезда в соответствующем направлении (на север, юг или восток). Движение разрешается при зеленом сигнале светофора в соответствующем направлении. Для простоты в этом примере будем считать, что:

a) автомобили на перекрестке не поворачивают, а только пересекают его;
b) цикл переключения светофоров: *красный-зеленый-красный-зеленый ...* .

Контроллер будем строить из трех параллельно функционирующих процессов, N, S и E. Каждый процесс управляет своим светофором, разрешающим движение в своем направлении. Выход контроллера – установка конкретного значения цвета светофоров: локальные переменные процессов N_LIGHT, S_LIGHT и E_LIGHT принимают значения RED и GREEN.Процесс N читает и может сбросить свою локальную булеву переменную N_REQ, которая указывает, что поступил запрос (request) на проезд в северном направлении. Эта переменная автоматически устанавливается событием N_SENSE:

```
while true do if (!N_REQ && N_SENSE) then N_REQ := true fi od
```

Этот цикл выполняется в процессе, моделирующем внешнюю среду, который работает параллельно с процессом N. То же справедливо для процессов S и E.

Для синхронизации процессов следует ввести переменные, с которыми могут работать все процессы. Введем глобальные булевы переменные (флаги направлений движения), которые назовем NS_LOCK и WE_ LOCK. Процессы N, S и E будут взаимодействовать через эти общие булевы переменные.

Все процессы, представленные ниже, записаны на Паскалеподобном языке. Алгоритм процесса N, регулирующий движение на север, может быть представлен так:

```
process N (){
  while true do            /* в бесконечном цикле */
    if ( N_REQ )           /* если есть запрос на проезд на север */
      wait ( !WE_LOCK );   /* то ждем освобождения поперечного направления, */
      NS_LOCK := true;     /* устанавливаем замок на направление С-Ю */
      N_LIGHT := GREEN;    /* и разрешаем проезд в северном направлении; */
      wait ( !N_SENSE );   /* ждем, когда на север проедут все машины */
      if (S_LIGHT=RED) NS_LOCK := false fi;
      /* если светофор на встречном направлении красный, снимаем замок */
      N_LIGHT := RED;      /* устанавливаем свой светофор в красный */
      N_REQ := false;      /* сбрасываем запрос на проезд на север */
    fi
  od
}
```

Процесс S строится аналогично.

Алгоритм процесса E, регулирующего движение на восток:

```
process E (){
  while true do            /* в бесконечном цикле */
    if ( E_REQ )           /* если есть запрос на проезд на восток */
     WE_LOCK := true;      /* то устанавливаем свой замок */
     E_LIGHT := GREEN;     /* и разрешаем проезд на восток */
     wait ( !NS_LOCK );    /* ждем, когда процессы N и S снимут замок */
     wait ( !E_SENSE );    /* ждем, когда на восток все проедут */
     WE_LOCK := false;     /* снимаем свой замок */
     E_LIGHT := RED;       /* устанавливаем свой светофор в красный */
     E_REQ := false;       /* сбрасываем запрос на проезд на восток */
    fi
  od
}
```

Алгоритмы этих процессов выглядят вполне разумно, однако корректность их не доказана. Как и в любых параллельных системах, в этой системе параллельных процессов ошибки очень вероятны. Если при проверке корректности этой или подобной системы процессов выявятся ошибки, их нужно будет скорректировать.

2 Задание для курсовой работы. Общее описание проезда по перекрестку

Для выполнения курсовой работы выбран более сложный перекресток с четырьмя двусторонними направлениями движения. В каждом направлении имеются 3 полосы движения для трех траекторий движения: одна – для проезда прямо по перекрестку, одна – для поворота направо, одна – для поворота налево. Движение по каждой траектории регулируется своим светофором: один светофор – для проезда прямо через перекресток, один (стрелка) – для поворота направо, один (стрелка) – для поворота налево.

Поворот направо разрешен всегда. Однако светофор, регулирующий движение по правой полосе горит красным, если машин на этой полосе нет. При появлении машин он загорается

зеленым и горит зеленым до тех пор, пока поток машин не прекращается. Когда поток машин прекратился, светофор загорается красным.

Светофор, регулирующий движение по любой траектории, загорается зеленым, если, во-первых, есть запрос от машин в соответствующей полосе и, во-вторых, по всем тем направлениям, с которыми существует пересечение, движение запрещено – на них горит красный сигнал.

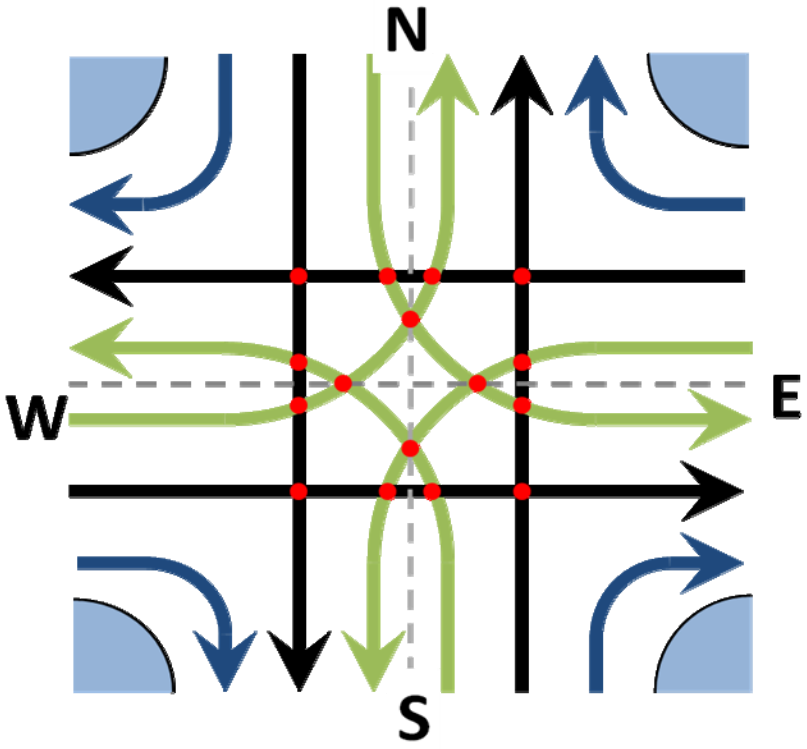

Рис.2 Схема перекрестка.

План проезда по перекрестку у каждого студента индивидуальный. Он строится из рис. 2 выбором нескольких (2-3-х) пересечений, индивидуально заданных для каждого студента.

Варианты пересечений

1	WN,NS	9	SN,WE
2	WN,NE	10	SN,EW
3	WN,SW	11	SN,ES
4	WN,EW	12	NE,EW
5	NS,SW	13	NE,ES
6	NS,WE	14	SW,WE
7	NS,EW	15	SW,ES
8	SN,NE	16	WE,ES

Свой вариант пересечений получите у преподавателя!

Допустим, что у нас заданы 2 пересечения, например, 1, 10. Отсюда следует, что по направлениям WN, NS, SN, EW – проезд есть. По всем другим направлениям в данной конкретной конфигурации перекрестка (для данного варианта задания студенту) полос для проезда нет. Нарисуйте Ваш перекресток.

Процессы контроллера для Вашего индивидуального перекрестка нужно построить на языке Promela. Для их построения можно взять за основу процессы управления простым перекрестком, представленные в п.1 (*они написаны не на языке Promela!*). Учтите, что в этих процессах могут быть ошибки синхронизации. Все ошибки нужно выявить с помощью системы верификации Spin на основе выдаваемых системой контрпримеров, демонстрирующих те сценарии движения на перекрестке, в которых процессы управляют своими светофорами неверно.

Дополнительные требования к модели контроллера.

1) При разработке алгоритма контроллера следует сохранить наличие процессов, управляющих внешней средой, регистрирующей автомобили, и всех процессов, регулирующих движение по перекрестку, для каждого направления.
2) При написании кода контроллера на Promela не следует злоупотреблять использованием конструкций `atomic`. Использование `atomic` возможно только для решения технических задач (например, для принудительного вывода значения переменной сразу после ее изменения), а не для решения проблем синхронизации процессов. Иными словами, использование `atomic` принимается, только если описанные далее свойства удовлетворяются независимо от наличия/отсутствия этих конструкций.

3 Свойства контроллера. Верификация алгоритма

Корректность построенного Вами контроллера следует проверить относительно следующих свойств:

Безопасность:

```
G !(( N_LIGHT=GREEN || S_LIGHT=GREEN) and (E_LIGHT = GREEN))
```

– никогда не будет разрешен проезд в пересекающихся направлениях.

Живость:

```
G ( N_SENSE and ( N_LIGHT=RED ) ⇒ F ( N_LIGHT=GREEN ) );
G ( S_SENSE and ( S_LIGHT=RED ) ⇒ F ( S_LIGHT=GREEN ) );
G ( E_SENSE and ( E_LIGHT=RED ) ⇒ F ( E_LIGHT=GREEN ) )
```

– при появлении машины, ей всегда предоставится возможность проезда в нужном направлении (возможно, не сразу).

Эти свойства должны быть выполнены при следующих ограничениях:

Справедливость:

```
GF !( (N_LIGHT=GREEN) & N_SENSE );
GF !( (S_LIGHT=GREEN) & S_SENSE );
GF !( (E_LIGHT=GREEN) & E_SENSE );
```

– предполагаем, что каждый сенсор устанавливается в *false* неопределенно часто, т.е. в каждом направлении не движется непрерывный поток машин.