Минобрнауки России

Санкт-Петербургский государственный политехнический университет

Институт информационных технологий и управления

**Кафедра «Информационные и управляющие системы»**

**КУРСОВАЯ РАБОТА**

**Разработка контроллера светофоров и его верификация**

по дисциплине «Распределенные алгоритмы и протоколы»

Выполнил

студент гр. 6084/12 А.Лукашин

Руководитель

ст. преподаватель Шошмина И.В.

«\_\_» \_\_\_\_\_\_\_\_\_\_ 2013 г.

Санкт-Петербург

2013

Оглавление

[Введение 3](#_Toc372573814)

[Постановка задачи 3](#_Toc372573815)

[Реализация 4](#_Toc372573816)

[Основная идея 4](#_Toc372573817)

[Модель на языке Promela 4](#_Toc372573818)

[LTL правила 10](#_Toc372573819)

[Безопасность 10](#_Toc372573820)

[Живость 11](#_Toc372573821)

[Справедливость 11](#_Toc372573822)

[Результат моделирования 11](#_Toc372573823)

[Выводы 11](#_Toc372573824)

# Введение

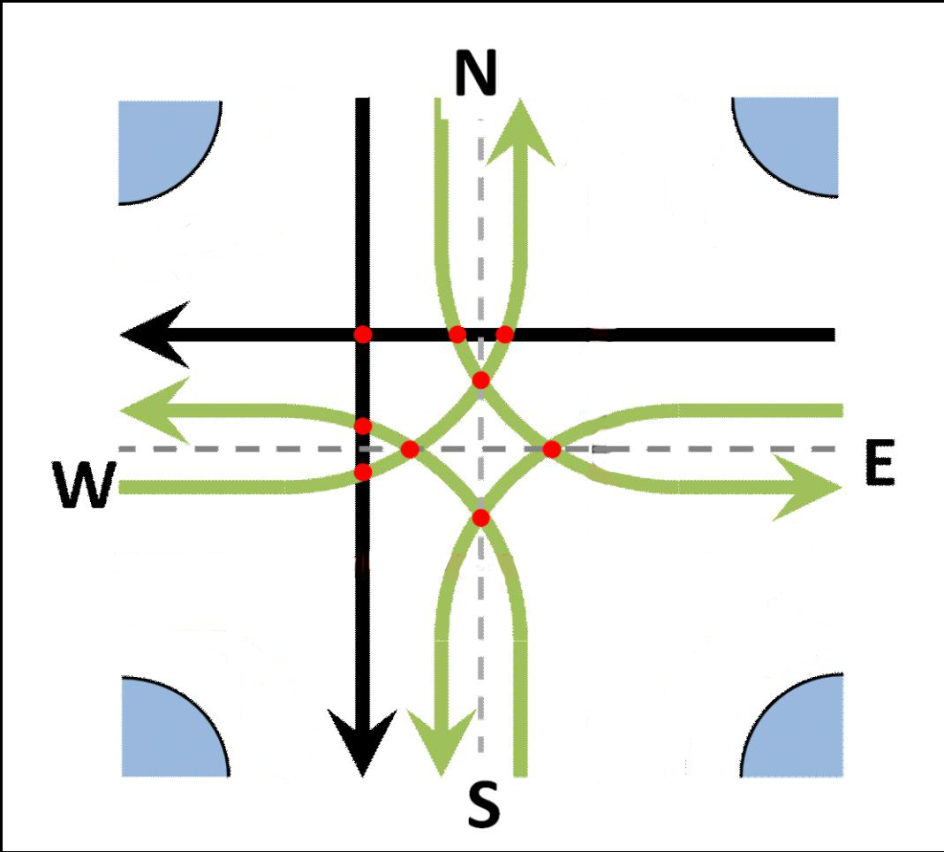
Верификация – это проверка модели (алгоритма, системы) на корректность работы, с учетом заданных правил. В настоящее время вопрос корректности работы программных продуктов является крайне актуальным. На программное обеспечивание возлагаются все новые и новые функции, при это в некоторых областях возникновение ошибки может приводить к катастрофическим последствиям. Такие тенденции позволяют говорить о важности применения и развития верификационных средств.

В данной курсовой работе рассматривается модель управления движением на перекретске, при условии,что каждое направление контролируется своим светофором. При этом для model checking используется система Spin. Данная система позволяет описывать модель на языке Promela и задавать требуемые свойства системы при помощи выражений LTL. Такой подход позволяет проверить корректность модели перед ее реализацией.

# Постановка задачи

Вариант (1, 12, 15).

Пересечения: {(NS, WN), (NE, EW),(SW,ES)}



Для заданного варианта необходимо создать набор свойств (LTL), которые определяют корректность работы модели (например отсутствие варианта, когда движение разрешено всем), а так же описать саму модель на языке Promela.

# Реализация

## Основная идея

Основной идеей данной реализации является отождествление пересечений направлений движения с ресурсами, которые необходимо получить контролеру для разрешения безопасного проезда. Данных подход обладает целым рядом преимуществ, из которых можно выделить:

1. Контролеры, управляющие пересекающимися направлениями, не могут дать зеленый свет одновременно
2. «Дружественные» контроллеры имеют возможность пропускать поток независимо
3. Данный подход довольно просто в реализации

## Модель на языке Promela

Данная модель была построена на основе следующих постулатов:

1. Каждый контроллер светофора является отдельным процессом.
2. Сигнал светофора может быть двух видов: красный – зеленый
3. Состояние светофора описывается глобальными переменными
4. Датчики движения также являются глобальными переменными
5. Движение (траффик машин) генерируется внешним, по отношению к контроллерам, процессом случайным образом

Реализация:

*/\* Course work made by Anton Lukashin group 6084-12 \*/*

*/\* Exercise 1,12,15 (WN,NS) (NE, EW) (SW, ES)\*/*

*/\* Types of signals \*/*

*mtype = {Red, Green};*

*/\* Lights signals \*/*

*mtype NS\_L = Red;*

*mtype WN\_L = Red;*

*mtype NE\_L = Red;*

*mtype EW\_L = Red;*

*mtype ES\_L = Red;*

*mtype SW\_L = Red;*

*/\* Car traffic sensors \*/*

*bool NS\_S = false;*

*bool WN\_S = false;*

*bool NE\_S = false;*

*bool EW\_S = false;*

*bool ES\_S = false;*

*bool SW\_S = false;*

*/\* Car crossing checkers \*/*

*bool NS\_C = true;*

*bool WN\_C = true;*

*bool NE\_C = true;*

*bool EW\_C = true;*

*bool ES\_C = true;*

*bool SW\_C = true;*

*/\*Safety\*/*

*ltl p0\_1 {[] !(NS\_L==Green && WN\_L==Green && SW\_L==Green && EW\_L==Green)}*

*ltl p0\_2 {[] !(WN\_L==Green && NS\_L==Green && NE\_L==Green && SW\_L==Green && EW\_L==Green)}*

*ltl p0\_3 {[] !(EW\_L==Green && NE\_L==Green && WN\_L==Green && NS\_L==Green)}*

*ltl p0\_4 {[] !(NE\_L==Green && WN\_L==Green && ES\_L==Green && EW\_L==Green)}*

*ltl p0\_5 {[] !(SW\_L==Green && ES\_L==Green && WN\_L==Green && NS\_L==Green)}*

*ltl p0\_6 {[] !(ES\_L==Green && SW\_L==Green && NE\_L==Green)}*

*/\*Liveness\*/*

*ltl p1\_1 {[] ((NS\_S && (NS\_L==Red)) -> <> (NS\_L==Green))}*

*ltl p1\_2 {[] ((WN\_S && (WN\_L==Red)) -> <> (WN\_L==Green))}*

*ltl p1\_3 {[] ((NE\_S && (NE\_L==Red)) -> <> (NE\_L==Green))}*

*ltl p1\_4 {[] ((EW\_S && (EW\_L==Red)) -> <> (EW\_L==Green))}*

*ltl p1\_5 {[] ((ES\_S && (ES\_L==Red)) -> <> (ES\_L==Green))}*

*ltl p1\_6 {[] ((SW\_S && (SW\_L==Red)) -> <> (SW\_L==Green))}*

*/\*Fairness\*/*

*ltl p2\_1 {[] <> !((NS\_L==Green) && NS\_S)}*

*ltl p2\_2 {[] <> !((WN\_L==Green) && WN\_S)}*

*ltl p2\_3 {[] <> !((NE\_L==Green) && NE\_S)}*

*ltl p2\_4 {[] <> !((EW\_L==Green) && EW\_S)}*

*ltl p2\_5 {[] <> !((ES\_L==Green) && ES\_S)}*

*ltl p2\_6 {[] <> !((SW\_L==Green) && SW\_S)*

*/\* Synchronization channels \*/*

*chan NS\_WN\_EW = [0] of {bool};*

*chan NS\_WN\_SW = [0] of {bool};*

*chan NE\_WN\_EW = [0] of {bool};*

*chan NE\_ES = [0] of {bool};*

*chan ES\_SW = [0] of {bool};*

*init*

*{*

*atomic{*

*NS\_WN\_EW ! true;*

*NS\_WN\_SW ! true;*

*NE\_WN\_EW ! true;*

*NE\_ES ! true;*

*ES\_SW ! true;*

*};*

*}*

*/\* Traffic generation process \*/*

*active proctype gen\_t ()*

*{*

*do*

*:: true ->*

*if*

*:: NS\_C -> NS\_S = !NS\_S; NS\_C = false;*

*:: else -> skip;*

*fi*

*:: true ->*

*if*

*:: WN\_C -> WN\_S = !WN\_S; WN\_C = false;*

*:: else -> skip;*

*fi*

*:: true ->*

*if*

*:: NE\_C -> NE\_S = !NE\_S; NE\_C = false;*

*:: else -> skip;*

*fi*

*:: true ->*

*if*

*:: EW\_C -> EW\_S = !EW\_S; EW\_C = false;*

*:: else -> skip;*

*fi*

*:: true ->*

*if*

*:: ES\_C -> ES\_S = !ES\_S; ES\_C = false;*

*:: else -> skip;*

*fi*

*:: true ->*

*if*

*:: SW\_C -> SW\_S = !SW\_S; SW\_C=false;*

*:: else -> skip;*

*fi*

*od;*

*}*

*/\* NS controller \*/*

*active proctype NS ()*

*{*

*do*

*/\* Wait for resources \*/*

*:: if*

*:: NS\_S ->*

*NS\_WN\_EW ? true; NS\_WN\_SW ? true;*

*NS\_L = Green;*

*NS\_C = true;*

*if*

*/\* Wait for end of car queue \*/*

*:: !NS\_S -> skip;*

*fi;*

*NS\_L = Red;*

*NS\_WN\_EW ! true; NS\_WN\_SW ! true;*

*:: else -> skip;*

*fi;*

*od;*

*}*

*/\* WN controller \*/*

*active proctype WN ()*

*{*

*do*

*:: if*

*:: WN\_S ->*

*/\* Wait for resources \*/*

*NS\_WN\_EW ? true; NS\_WN\_SW ? true; NE\_WN\_EW ? true;*

*WN\_L = Green;*

*WN\_C = true;*

*if*

*/\* Wait for end of car queue \*/*

*:: !WN\_S -> skip;*

*fi;*

*WN\_L = Red;*

*NS\_WN\_EW ! true; NS\_WN\_SW ! true; NE\_WN\_EW ! true;*

*:: else -> skip;*

*fi;*

*od;*

*}*

*/\* NE controller \*/*

*active proctype NE ()*

*{*

*do*

*/\* Wait for resources \*/*

*:: if*

*:: NE\_S ->*

*NE\_WN\_EW ? true; NE\_ES ? true;*

*NE\_L = Green;*

*NE\_C = true;*

*if*

*/\* Wait for end of car queue \*/*

*:: !NE\_S -> skip;*

*fi;*

*NE\_L = Red;*

*NE\_WN\_EW ! true; NE\_ES ! true;*

*:: else -> skip;*

*fi;*

*od;*

*}*

*/\* NE controller \*/*

*active proctype EW ()*

*{*

*do*

*/\* Wait for resources \*/*

*:: if*

*:: EW\_S ->*

*NS\_WN\_EW ? true; NE\_WN\_EW ? true;*

*EW\_L = Green;*

*EW\_C = true;*

*if*

*/\* Wait for end of car queue \*/*

*:: !EW\_S -> skip;*

*fi;*

*EW\_L = Red;*

*NS\_WN\_EW ! true; NE\_WN\_EW ! true;*

*:: else -> skip;*

*fi;*

*od;*

*}*

*/\* ES controller \*/*

*active proctype ES ()*

*{*

*do*

*/\* Wait for resources \*/*

*:: if*

*:: ES\_S ->*

*ES\_SW ? true; NE\_ES ? true;*

*ES\_L = Green;*

*ES\_C = true;*

*if*

*/\* Wait for end of car queue \*/*

*:: !ES\_S -> skip;*

*fi;*

*ES\_L = Red;*

*ES\_SW ! true; NE\_ES ! true;*

*:: else -> skip;*

*fi;*

*od;*

*}*

*/\* SW controller \*/*

*active proctype SW ()*

*{*

*do*

*/\* Wait for resources \*/*

*:: if*

*:: SW\_S ->*

*NS\_WN\_SW ? true; ES\_SW ? true;*

*SW\_L = Green;*

*SW\_C = true;*

*if*

*/\* Wait for end of car queue \*/*

*:: !SW\_S -> skip;*

*fi;*

*SW\_L = Red;*

*NS\_WN\_SW ! true; ES\_SW ! true;*

*:: else -> skip;*

*fi;*

*od;*

*}*

## LTL правила

Данные правила на языке темпоральной логики оперируют двумя базовыми понятиями:

1. G ([] – в системе Spin) – описывает свойство, которое должно выполняться всегда
2. F (<> - в системе Spin) – описывает свойство, которое должно выполнится когда-то в будущем

Правила данной системы:

### Безопасность

Для данной модели правила безопасности звучат на естественном языке следующим образом: «Никогда не будет такой ситуации, что на данном направлении будет гореть зеленый свет, и на всех пересекающих это направление дорогах тоже будет зеленый»

1. NS - {[] !(NS\_L==Green && WN\_L==Green && SW\_L==Green && EW\_L==Green)}
2. WN – {[] !(WN\_L==Green && NS\_L==Green && NE\_L==Green && SW\_L==Green && EW\_L==Green)}
3. NE - {[] !(NE\_L==Green && WN\_L==Green && ES\_L==Green && EW\_L==Green)}
4. EW - {[] !(EW\_L==Green && NE\_L==Green && WN\_L==Green && NS\_L==Green)}
5. SW - {[] !(SW\_L==Green && ES\_L==Green && WN\_L==Green && NS\_L==Green)}
6. ES - {[] !(ES\_L==Green && SW\_L==Green && NE\_L==Green)}

### Живость

Для данной модели правила живости звучат на естественном языке следующим образом: «Всегда выполняется, если светофор горит красным и датчик показывает наличие машин, следовательно в будущем данный светофор будет гореть зеленым»

1. NS - {[] ((NS\_S && (NS\_L==Red)) -> <> (NS\_L==Green))}
2. WN - {[] ((WN\_S && (WN\_L==Red)) -> <> (WN\_L==Green))}
3. NE - {[] ((NE\_S && (NE\_L==Red)) -> <> (NE\_L==Green))}
4. EW - {[] ((EW\_S && (EW\_L==Red)) -> <> (EW\_L==Green))}
5. SW - {[] ((SW\_S && (SW\_L==Red)) -> <> (SW\_L==Green))}
6. ES - {[] ((ES\_S && (ES\_L==Red)) -> <> (ES\_L==Green))}

### Справедливость

Для данной модели правила справедливости звучат на естественном языке следующим образом: «Невозможна такая ситуация, что в конкретном направлении движется непрерывный поток машин(т.е. светофор должен неопределенно часто менять свой сигнал на запрещающий)»

1. NS - {[] !((NS\_L==Green) && NS\_S)}
2. WN - {[] !((WN\_L==Green) && WN\_S)}
3. EW - {[] !((NE\_L==Green) && NE\_S)}
4. NE - {[] !((EW\_L==Green) && EW\_S)}
5. SW - {[] !((ES\_L==Green) && ES\_S)}
6. ES - {[] !((SW\_L==Green) && SW\_S)}

# Результаты моделирования

# Выводы