

System Modelling and Design

Traffic Lights

Revision: 1.1, April 10, 2008

Ken Robinson

May 21, 2008

©Ken Robinson 2005

[mailto::k.robinson@unsw.edu.au](mailto:k.robinson@unsw.edu.au)

Contents

1 Objectives of this lecture	1
2 A simple 2-way intersection	2
3 The Context Machine	2
4 The Invariant	3
4.1 A Safe state	3
4.2 Strengthening the Invariant	4
4.3 Other Invariants	4
4.4 Some Theorems	5
5 The ChangeLight Event	5
5.1 Sequencing	5
6 Refining ChangeLight	6
7 A General Multi-Way Intersection	8
7.1 The New Safety Invariant	8
7.2 A Traffic Light Controller	9

1 Objectives of this lecture

- To explore the use of a state invariant to ensure safety.
- To explore the specification of some simple traffic light controllers for a simple traffic light controlled intersection.

- To expand safety to a general intersection.
- To model a traffilight controller for a general intersection

2 A simple 2-way intersection

Consider traffic lights at the intersection of two roads, one running *North-South* and the other *East-West*. There are four sets of lights, each capable of showing **Red**, **Green** and **Amber**, placed at *North*, *East*, *South* and *West* positions.

The *North* and *South* lights are always identical, as are the *East* and *West* lights.

There are no *right-turn* lights.

Lights should change in the sequence:

Red → **Green** → **Amber** → **Red** → ...

We wish to specify a traffic light controller that ensures safety and the correct sequencing.

3 The Context Machine

We will introduce a context machine containing the enumerated sets *DIRECTION* and *LIGHT*.

We will also specify a constant (function) *OTHERDIR* that maps each direction to the other direction.

Context machines must be used in Event B (eB) to define sets and constants.

CONTEXT TwoWay_ctx

SETS

LIGHTS

DIRECTION

CONSTANTS

Red

Green

Amber

NorthSouth

EastWest

OTHERDIR

AXIOMS

$LIGHTS = \{Red, Green, Amber\}$
 $Red \neq Green$
 $Red \neq Amber$
 $Green \neq Amber$
 $DIRECTION = \{NorthSouth, EastWest\}$
 $NorthSouth \neq EastWest$
 $OTHERDIR \in DIRECTION \rightarrow DIRECTION$
 $OTHERDIR(NorthSouth) = EastWest$
 $OTHERDIR(EastWest) = NorthSouth$

END

The Simple TwoWay Machine

The **TwoWay** machine

1. sees the context machine and has a state of one variable, *lights*, which is a total function from *DIRECTION* to *LIGHT*;
2. has a single event **ChangeLight** to change the lights;
3. ensures that lights change in the sequence **Red, Green, Amber, ...**;
4. maintains a safe intersection;

4 The Invariant

1. We wish to formulate an invariant that will ensure safety.
2. Whenever the state is unsafe, the invariant must be *false*.

$$\neg(\text{safe}) \implies \neg(\text{invariant}) \quad (1)$$

3. Conversely, whenever the invariant is *true*, the state should be safe.

$$\text{invariant} \implies \text{safe} \quad (2)$$

4. Of course, 1 and 2 are equivalent; one is the *contrapositive* of the other: $P \implies Q \equiv \neg Q \implies \neg P$
5. If we have the *weakest* invariant, then whenever the state is safe, the invariant will be *true*.
6. We need to find adequately strong preconditions.

4.1 A Safe state

The state invariant should be:

1. *false* for all unsafe states *true* for all safe states

2. An initial attempt at an invariant might be

$$\neg(\text{lights}(\text{NorthSouth}) = \text{Green} \wedge \text{lights}(\text{EastWest}) = \text{Green})$$

which, using the predicate identities

$$\neg(P \wedge Q) \equiv \neg P \vee \neg Q \equiv P \implies \neg Q,$$

may be written

$$\text{lights}(\text{NorthSouth}) = \text{Green} \implies \neg(\text{lights}(\text{EastWest}) = \text{Green})$$

If used as a term in a larger predicate, the above implication may need to be parenthesised.

4.2 Strengthening the Invariant

Clearly, the light in both directions cannot be either **Green** or **Amber**.

	Red	Green	Amber
Red	safe	safe	safe
Green	safe	unsafe	unsafe
Amber	safe	unsafe	unsafe

This leads to the invariant:

$$\begin{aligned} & \neg(\text{lights}(\text{NorthSouth}) \in \{\text{Green}, \text{Amber}\} \wedge \\ & \text{lights}(\text{EastWest}) \in \{\text{Green}, \text{Amber}\}) \\ \equiv & \\ & (\text{lights}(\text{NorthSouth}) \in \{\text{Green}, \text{Amber}\} \implies \\ & \neg(\text{lights}(\text{EastWest}) \in \{\text{Green}, \text{Amber}\})) \\ \equiv & \\ & (\text{lights}(\text{NorthSouth}) \in \{\text{Green}, \text{Amber}\} \implies \\ & \text{lights}(\text{EastWest}) = \text{Red}) \end{aligned} \tag{3}$$

4.3 Other Invariants

There are other invariants that adequately express safety for a two-way intersection:

$$\text{lights}(\text{NorthSouth}) = \text{Red} \vee \text{lights}(\text{EastWest}) = \text{Red}$$

$$\text{Red} \in \text{ran}(\text{lights})$$

But these conditions do not generalise to intersections with more than two ways. Indeed the expression of the invariant that best generalises is

$$\begin{aligned} & \forall \text{dir}. \text{dir} \in \text{DIRECTION} \wedge \text{lights}(\text{dir}) \in \{\text{Green}, \text{Amber}\} \implies \\ & \text{lights}(\text{OTHERDIR}(\text{dir})) = \text{Red} \end{aligned} \tag{4}$$

4.4 Some Theorems

In this model of **TwoWay** we add the following alternative formulations of the invariant as theorems:

- $\forall dir. (dir \in DIRECTION \wedge lights(dir) \in \{Green, Amber\}) \implies lights(OTHERDIR(dir)) = Red$
- $lights(EastWest) \in \{Green, Amber\} \implies lights(NorthSouth) = Red$

Theorems contain properties that are implied by the invariant, and hence discharging the proof obligations for the assertions proves that each conjunct in the assertions is implied by the invariant.

5 The ChangeLight Event

The ChangeLight event will have two parameters dir and $light$.

The event will change the lights in direction dir to colour $light$

$$lights(dir) := light$$

The guards need to be strong enough to ensure both safety and correct sequencing.

Safety can be ensured with the guard:

$$(light \in \{Green, Amber\}) \implies lights(OTHERDIR(dir)) = Red$$

5.1 Sequencing

Changing to

Red: the current colour should be **Green**, so $light = Red \implies lights(dir) = Green$

Green: the current colour should be **Red**, so $light = Green \implies lights(dir) = Red$

Amber: the current colour should be **Green**, so $light = Amber \implies lights(dir) = Green$

Notice that the sequencing guards for both **Red** and **Amber** together with the invariant ensure safety, we only need to be concerned with safety for changing to **Green**.

Final Guards for ChangeLight

The final guards that ensure both safety and correct sequencing are:

$$\begin{aligned} light = Red &\implies lights(dir) = Amber \\ light = Green &\implies lights(dir) = Red \\ light = Green &\implies lights(OTHERDIR(dir)) = Red \\ light = Amber &\implies lights(dir) = Green \end{aligned}$$

The ChangeLight Event

ChangeLight $\hat{=}$

A single operation to change light

any

dir

light

where

light = *Red* \Rightarrow *lights*(*dir*) = *Amber*

light = *Green* \Rightarrow *lights*(*dir*) = *Red*

light = *Green* \Rightarrow *lights*(*OTHERDIR*(*dir*)) = *Red*

light = *Amber* \Rightarrow *lights*(*dir*) = *Green*

Sequencing

Sequencing

Safety

Sequencing

then

lights(*dir*) := *light*

end

6 Refining ChangeLight

We will now refine the **TwoWay** machine and give a simple refinement of **ChangeLight**, in which we refine that event into 3 partial refinements **ToRed**, **ToGreen**, and **ToAmber** that respectively change a light to **Red**, **Green** or **Amber**.

This demonstrates an equivalence between the single event and the 3 simpler events.

TwoWayR

MACHINE TwoWayR

REFINES TwoWay

SEES TwoWay_ctx

VARIABLES

lights

EVENTS

Initialisation

begin

lights := {*NorthSouth* \mapsto *Red*, *EastWest* \mapsto *Red*}

end

ToRed $\hat{=}$

Refines ChangeLight

any

dir

light

where

dir ∈ *DIRECTION*

lights(dir) = *Amber*

light = *Red*

then

lights(dir) := *Red*

end

ToGreen ≐

Refines ChangeLight

any

dir

light

where

dir ∈ *DIRECTION*

lights(dir) = *Red*

lights(OTHERDIR(dir)) = *Red*

light = *Green*

then

lights(dir) := *Green*

end

ToAmber ≐

Refines ChangeLight

any

dir

light

where

dir ∈ *DIRECTION*

lights(dir) = *Green*

light = *Amber*

then

lights(dir) := *Amber*

end

END

7 A General Multi-Way Intersection

We will now expand to a completely general intersection with many *directions*, denoted by the finite set *DIRECTION*. To define directions that *conflict* with one another we will use a relation *CONFLICT* that maps each direction to all other directions, with which it conflicts. Clearly, *CONFLICT* must have the following properties:

- No direction conflicts with itself.
- Conflicts are symmetric: if dir_1 conflicts with dir_2 then dir_2 conflicts with dir_1 .

7.1 The New Safety Invariant

The safety invariant for our generalised intersection is

$$\begin{aligned} \forall d. d \in DIRECTION \wedge lights(d) \in \{Green, Amber\} \implies \\ lights[CONFLICT[\{d\}]] = \{Red\} \end{aligned} \quad (5)$$

This can be seen as a generalisation of the earlier safety invariant (4).

ChangeLight_ctx

CONTEXT TrafficLights_ctx

SETS

LIGHTS

DIRECTION

CONSTANTS

Red

Green

Amber

CONFLICT

adir

AXIOMS

$LIGHTS = \{Red, Green, Amber\}$

$Red \neq Green$

$Red \neq Amber$

$Green \neq Amber$

$finite(DIRECTION)$

DIRECTION is a finite set of directions

$CONFLICT \in DIRECTION \leftrightarrow DIRECTION$	CONFLICT relates conflicting
directions	
$CONFLICT \cap id(DIRECTION) = \emptyset$	a direction cannot conflict with itself
$CONFLICT^{-1} = CONFLICT$	conflicts are symmetric
$adir \in DIRECTION$	

END

7.2 A Traffic Light Controller

We will now model a traffic light controller that responds to the following requirements:

1. A controller is required that can be used to control the whole of a general intersection.
2. The controller must be able to set the light in any direction to **Green** and must do so in a safe transition sequence.
3. The intersection must always be in a safe state.

Our Response

Our response to the preceding request is

1. to produce a machine with a single event that changes the light in any arbitrary direction, modelled by a constant *adir*, to **Green**, setting all lights in conflicting directions to **Red**;
2. to model the top-level state as having on **Red** and **Green**, these being the “steady state” light colours;
3. to refine that machine to model the process of changing the state while maintaining a safe state.

ChangeLight machine

MACHINE ChangeLight

SEES TrafficLights_ctx

VARIABLES

lights

INVARIANTS

$lights \in DIRECTION \rightarrow \{Red, Green\}$	
$\forall d \cdot d \in DIRECTION \wedge lights(d) = Green$	
$\Rightarrow lights[CONFLICT[\{d\}]] \subseteq \{Red\}$	Safety

EVENTS

Initialisation

begin

$lights : lights' \in DIRECTION \rightarrow \{Red, Green\}$
$\wedge (\forall d \cdot d \in DIRECTION \wedge lights'(d) = Green$
$\Rightarrow lights'[CONFLICT[\{d\}]] \subseteq \{Red\})$

end

ToGreen $\hat{=}$

begin

$lights := lights \triangleleft \{adir \mapsto Green\}$
 $\triangleleft (CONFLICT[\{adir\}] \times \{Red\})$

end

END

The Refinement Model

The refinement model introduces **Amber** into the lights sequence and the following slave events:

ToAmber changes, in arbitrary sequence, all the currently **Green** lights to **Amber**;

ToRed: changes the **Amber** lights to **Red**;

Delay: models a delay between **Amber** and **Red** and between **Red** and the final setting of the light in the *adir* direction to **Green**.

The refinement of ToGreen waits until all the conflicting directions have been changed to **Red** and then sets the light in the *adir* direction to **Green**.

ChangeLightR1 refinement

MACHINE ChangeLightR1

REFINES ChangeLight

SEES TrafficLights_ctx

VARIABLES

xlights Extended lights, Red, Green and Amber lights

delay delay between Amber and Red or Red and Green

INVARIANTS

$xlights \in DIRECTION \rightarrow LIGHTS$

$\forall d \cdot d \in DIRECTION \wedge xlights[\{d\}] \subseteq \{Green, Amber\}$

$\Rightarrow xlights[CONFLICT[\{d\}]] \subseteq \{Red\}$

$CONFLICT[\{adir\}] \triangleleft lights = CONFLICT[\{adir\}] \triangleleft xlights$ Only change lights

for *adir* and $CONFLICT[\{adir\}]$

$xlights(adir) = Green \Rightarrow lights = xlights$

$delay \subseteq DIRECTION$

EVENTS

Initialisation

begin

with

$lights' = xlights'$
 $xlights : xlights' \in DIRECTION \rightarrow \{Red, Green\}$
 $\wedge (\forall d \cdot d \in DIRECTION \wedge xlights'(d) = Green$
 $\Rightarrow xlights'[CONFLICT[\{d\}]] \subseteq \{Red\})$
 $delay := \emptyset$

end

ToGreen $\hat{=}$

Refines ToGreen

when

$xlights[CONFLICT[\{adir\}]] \subseteq \{Red\}$
 $adir \notin delay$

then

$xlights := xlights \Leftarrow \{adir \mapsto Green\}$

end

ToAmber $\hat{=}$

Which is convergent

any

dir

where

$dir \in DIRECTION$
 $dir \in CONFLICT[\{adir\}]$
 $xlights(dir) = Green$
 $xlights(adir) \neq Green$
 $dir \notin delay$

then

$xlights(dir) := Amber$
 $delay := delay \cup \{dir\}$

end

ToRed $\hat{=}$

Which is convergent

any

dir

where

$dir \in DIRECTION$
 $dir \in CONFLICT[\{adir\}]$
 $xlights(dir) = Amber$
 $dir \notin delay$
 $xlights(adir) \neq Green$

then

$xlights(dir) := Red$
 $delay := delay \cup \{adir\}$

end

Delay $\hat{=}$

Which is convergent

any

dir

where

$dir \in delay$

then

$delay := delay \setminus \{dir\}$

end

VARIANT $4 * card(xlights^{-1}[\{Green\}]) + 2 * card(xlights^{-1}[\{Amber\}]) + card(delay)$

END