

# System Modelling and Design

## Traffic Lights

Revision: 2, March 23, 2010

Ken Robinson

March 19, 2012

©Ken Robinson 2005-2010

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

## Contents

<b>1 Objectives of this lecture</b>	<b>1</b>
<b>2 A simple 2-way intersection</b>	<b>2</b>
<b>3 The Context Machine</b>	<b>2</b>
<b>4 The Invariant</b>	<b>3</b>
4.1 A Safe state . . . . .	4
4.2 Strengthening the Invariant . . . . .	4
4.3 Other Invariants . . . . .	4
4.4 Some Theorems . . . . .	5
<b>5 The ChangeLight Event</b>	<b>5</b>
5.1 Sequencing . . . . .	5
<b>6 A General Multi-Way Intersection</b>	<b>7</b>
6.1 The New Safety Invariant . . . . .	7
6.2 A Traffic Light Controller . . . . .	8
<b>7 A General Multi-Way Intersection Parametric Controller</b>	<b>13</b>

## 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 trafficlight 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 *LIGHTS*.

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

*Context machines must be used in Event B to define sets and constants.*

**CONTEXT** TrafficLights\_ctx

**SETS** *LIGHTS DIRECTION*

**CONSTANTS** *Red Green Amber NorthSouth EastWest OTHERDIR*

**AXIOMS**

*axm1: LIGHTS = {Red, Green, Amber}*

*axm2: Red ≠ Green*

*axm3: Red ≠ Amber*

*axm4: Green ≠ Amber*

*axm5: DIRECTION = {NorthSouth, EastWest}*

*axm6: NorthSouth ≠ EastWest*

*axm7: OTHERDIR ∈ DIRECTION → DIRECTION*

*axm8: OTHERDIR(NorthSouth) = EastWest*

*axm9: OTHERDIR(EastWest) = NorthSouth*

**THEOREMS**

*thm1*:  $\forall d. d \in \text{DIRECTION}$   
 $\Rightarrow$   
 $\text{OTHERDIR}(\text{OTHERDIR}(d)) = d$

**END**

The above can be expressed more simply as

## AXIOMS

*axiom*:  $\text{partition}(\text{LIGHTS}, \{\text{Red}\}, \{\text{Green}\}, \{\text{Amber}\})$

*axiom*:  $\text{partition}(\text{DIRECTION}, \{\text{NorthSouth}\}, \{\text{EastWest}\})$

## 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*.

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

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

$$\text{invariant} \implies \text{safe} \tag{2}$$

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 guards.

## 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**.

	<b>Red</b>	<b>Green</b>	<b>Amber</b>
<b>Red</b>	safe	safe	safe
<b>Green</b>	safe	unsafe	unsafe
<b>Amber</b>	safe	unsafe	unsafe

This leads to the invariant:

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

## 4.3 Other Invariants

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

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

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

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

$$\forall dir. dir \in DIRECTION \wedge \text{lights}(dir) \in \{Green, Amber\} \implies \text{lights}(OTHERDIR(dir)) = Red \quad (4)$$

#### 4.4 Some Theorems

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

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

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*

$$\text{lights}(dir) := light$$

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

Safety can be ensured with the guard:

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

#### 5.1 Sequencing

Changing to

**Red:** the current colour should be Amber, so  $light = Red \implies \text{lights}(dir) = Amber$

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

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

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:

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

## The ChangeLight Event

**MACHINE** ChangeLights

**SEES** TrafficLights\_ctx

**VARIABLES** *lights*

### INVARIANTS

*inv1:*  $\text{lights} \in \text{DIRECTION} \rightarrow \text{LIGHTS}$

*inv2:*  $\text{lights}(\text{NorthSouth}) \in \{\text{Green}, \text{Amber}\}$   
 $\implies \text{lights}(\text{EastWest}) = \text{Red}$

### EVENTS

#### Initialisation

**begin**

*act1:*  $\text{lights} := \{\text{NorthSouth} \mapsto \text{Red}, \text{EastWest} \mapsto \text{Red}\}$

**end**

**Event** *ToAmber*  $\hat{=}$

**any** *dir*

**when**

*grd1:*  $\text{dir} \in \text{DIRECTION}$

*grd2:*  $\text{lights}(\text{dir}) = \text{Green}$

**then**

*act1:*  $\text{lights}(\text{dir}) := \text{Amber}$

**end**

**Event** *ToGreen*  $\hat{=}$

**any** *dir*

**when**

*grd1:*  $\text{dir} \in \text{DIRECTION}$

*grd2*:  $lights(OTHERDIR(dir)) = Red$

*grd3*:  $lights(dir) = Red$

**then**

*act1*:  $lights(dir) := Green$

**end**

**Event** *ToRed*  $\hat{=}$

**any** *dir*

**when**

*grd1*:  $dir \in DIRECTION$

*grd2*:  $lights(dir) = Amber$

**then**

*act1*:  $lights(dir) := Red$

**end**

**END**

## 6 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  $dir_1$ .

### 6.1 The New Safety Invariant

The safety invariant for our generalised intersection is

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

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

## TrafficLights1\_ctx

**CONTEXT** TrafficLights1\_ctx

**SETS** LIGHTS DIRECTION

**CONSTANTS** Red Green Amber CONFLICT adir

### AXIOMS

*axm1:*  $partition(LIGHTS, \{Red\}, \{Green\}, \{Amber\})$  Three light colours

*axm2:*  $finite(DIRECTION)$  DIRECTION is a finite set of directions

*axm3:*  $CONFLICT \in DIRECTION \leftrightarrow DIRECTION$  CONFLICT relates conflicting directions

*axm4:*  $CONFLICT \cap (DIRECTION \triangleleft id) = \emptyset$  a direction cannot conflict with itself

*axm5:*  $CONFLICT^{-1} = CONFLICT$  conflicts are symmetric

*thm1:*  $\forall d \cdot d \in DIRECTION \Rightarrow d \notin CONFLICT[\{d\}]$

*thm2:*  $\forall d1, d2 \cdot d1 \notin CONFLICT[\{d2\}] \Rightarrow d2 \notin CONFLICT[\{d1\}]$

*axm6:*  $adir \in DIRECTION$  any direction (used as a parameter)

**END**

## 6.2 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 only **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.
4. towards safety, the transition from **Green** to **Red** will introduce the light colour **Amber** and delays between light transitions.

## ChangeLight1 machine

**MACHINE** ChangeLight1

**SEES** TrafficLights1\_ctx

**VARIABLES** *lights*

### INVARIANTS

*inv1:*  $lights \in DIRECTION \rightarrow \{Red, Green\}$

*inv2:*  $\forall d \cdot d \in DIRECTION \wedge lights(d) = Green$   
 $\Rightarrow lights[CONFLICT[\{d\}]] \subseteq \{Red\}$

Safety

### EVENTS

#### Initialisation

##### begin

*act1:*  $lights : |$   
 $lights' \in DIRECTION \rightarrow \{Red, Green\}$   
 $\wedge (\forall d \cdot d \in DIRECTION$   
 $\wedge lights'(d) = Green$   
 $\Rightarrow lights'[CONFLICT[\{d\}]] \subseteq \{Red\})$

##### end

**Event** *ToGreen*  $\hat{=}$

##### when

*grd1:*  $lights(adir) = Red$

##### then

*act1:*  $lights := lights$   
 $\Leftarrow (CONFLICT[\{adir\}] \times \{Red\})$   
 $\Leftarrow \{adir \mapsto Green\}$

##### end

### END

## The Refinement Model

The first stage is still abstract and would be very unsafe, if implemented literally according to the model, as it is not safe to change lights instantaneously from **Green** to **Red**. This refinement introduces **Amber** into the lights sequence and the following slave events:

**ToAmber** changes, in arbitrary sequence, all the currently **Green** lights in conflicting directions 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**.

### ChangeLight1R refinement

**MACHINE** ChangeLight1R

**REFINES** ChangeLight1

**SEES** TrafficLights1\_ctx

**VARIABLES** *xlights* Extended lights, Red, Green and Amber lights    *delay* delay between Amber and Red, and between Red and Green

### INVARIANTS

*inv1:*  $xlights \in DIRECTION \rightarrow LIGHTS$

*inv2:*  $\forall d \cdot d \in DIRECTION \wedge xlights[\{d\}] \subseteq \{Green, Amber\}$   
 $\Rightarrow$   
 $xlights[CONFLICT[\{d\}]] \subseteq \{Red\}$

*inv3:*  $xlights \triangleleft (CONFLICT[\{adir\}] \times \{Red\})$   
 $=$   
 $lights \triangleleft (CONFLICT[\{adir\}] \times \{Red\})$

*inv4:*  $xlights(adir) = Green \Rightarrow lights = xlights$

*inv5:*  $delay \subseteq DIRECTION$

*thm1:*  $finite(xlights)$

*thm2:*  $CONFLICT[\{adir\}] \triangleleft lights = CONFLICT[\{adir\}] \triangleleft xlights$

*thm3:*  $\forall d, b, a \cdot d \in DIRECTION$  changing light in direction d  
 $\wedge b \in LIGHTS \wedge a \in LIGHTS \wedge a \neq b \wedge xlights(d) = b$   
 $\Rightarrow$   
 $card((xlights \triangleleft \{d \mapsto a\}) \triangleright \{b\})$   
 $=$   
 $card(xlights \triangleright \{b\}) - 1$   
 from b (= before) to a (= after) decreases number of colour b lights by 1

*thm4:*  $\forall d, b, a \cdot d \in DIRECTION$  changing light in direction d  
 $\wedge b \in LIGHTS \wedge a \in LIGHTS \wedge a \neq b \wedge xlights(d) = b$   
 $\Rightarrow$   
 $card((xlights \triangleleft \{d \mapsto a\}) \triangleright \{a\})$   
 $=$   
 $card(xlights \triangleright \{a\}) + 1$   
 from b (= before) to a (=after) increases number of colour a lights by 1

*thm5:*  $\forall d, b, a, c. d \in \text{DIRECTION}$  changing light in direction d from b  
 $\wedge b \in \text{LIGHTS} \wedge a \in \text{LIGHTS} \wedge c \in \text{LIGHTS}$   
 $\wedge \text{xlights}(d) = b \wedge c \neq a \wedge c \neq b$   
 $\Rightarrow$   
 $\text{card}((\text{xlights} \Leftarrow \{d \mapsto a\}) \triangleright \{c\})$   
 $=$   
 $\text{card}(\text{xlights} \triangleright \{c\})$   
 (= before) to a (=after) does not change number of colour c,  $c \neq a$ ,  $c \neq b$

## EVENTS

### Initialisation

**begin**

**with**

*lights'* :  $\text{lights}' = \text{xlights}'$

*act1:*  $\text{xlights} : |$   
 $\text{xlights}' \in \text{DIRECTION} \rightarrow \{\text{Red}, \text{Green}\}$   
 $\wedge (\forall d. d \in \text{DIRECTION}$   
 $\wedge \text{xlights}'(d) = \text{Green}$   
 $\Rightarrow$   
 $\text{xlights}'[\text{CONFLICT}[\{d\}]] \subseteq \{\text{Red}\})$

*act2:*  $\text{delay} := \emptyset$

**end**

**Event** *ToGreen*  $\hat{=}$

**refines** *ToGreen*

**when**

*grd1:*  $\text{xlights}(\text{adir}) = \text{Red}$

*grd2:*  $\text{xlights}[\text{CONFLICT}[\{\text{adir}\}]] \subseteq \{\text{Red}\}$

*grd3:*  $\text{delay} = \emptyset$

**then**

*act1:*  $\text{xlights} := \text{xlights} \Leftarrow \{\text{adir} \mapsto \text{Green}\}$

**end**

**Event** *GreenToAmber*  $\hat{=}$

**Status** convergent

**any** *dir*

**when**

*grd1:*  $\text{dir} \in \text{CONFLICT}[\{\text{adir}\}]$

*grd2:*  $xlights(dir) = Green$   
*grd3:*  $xlights(adir) \neq Green$   
*grd4:*  $adir \notin delay$   
**then**  
*act1:*  $xlights(dir) := Amber$   
*act2:*  $delay := delay \cup \{dir\}$   
**end**  
**Event** *AmberToRed*  $\hat{=}$   
**Status** convergent  
**any** *dir*  
**when**  
*grd1:*  $dir \in CONFLICT[\{adir\}]$   
*grd2:*  $xlights(dir) = Amber$   
*grd3:*  $dir \notin delay$   
*grd4:*  $xlights(adir) \neq Green$   
**then**  
*act1:*  $xlights(dir) := Red$   
*act2:*  $delay := delay \cup \{dir\}$   
**end**  
**Event** *Delay*  $\hat{=}$   
**Status** convergent  
**any** *dir*  
**when**  
*grd1:*  $dir \in delay$   
**then**  
*act1:*  $delay := delay \setminus \{dir\}$   
**end**  
**VARIANT**  $4 * card(xlights \triangleright \{Green\})$   
 $+ 2 * card(xlights \triangleright \{Amber\})$   
 $+ card(delay)$   
**END**

## The Variant

The events *GreenToAmber*, *AmberToRed* and *Delay* are *slave* events; that is they are not the main event, which is **ToGreen**.

Consequently they must not be able to be enabled forever; that is they must be *convergent*.

In EventB this is verified by providing a *variant* expression: an expression that has a lower bound and must strictly decrease everytime a convergent event fires.

In this model the variant is an expression that yields a natural number —hence lower bound 0— that strictly decreases.

## The Variant

The variant is a sum of the following:

**4 times** the number of **Red** lights;

**2 times** the number of **Amber** lights;

**1 times** the number of current delays

This is derived from the *GreenToAmber* event that

- decreases the number of Red lights by 1;
- increases the number of Amber lights by 1;
- increases the number of delays by 1.

and the *AmberToRed* event that

- decreases the number of Amber lights by 1;
- increases the number of delays by 1.

## 7 A General Multi-Way Intersection Parametric Controller

### Extending the model

In this development a parameterised version of the preceding traffic light control system is presented. The difference will start to be apparent when you look at the new context in which you will notice that the constant *adir* is missing.

When you look at the new *ChangeLight* machine you will notice a few differences. First, there are some new events, or rather one renamed event **ToGreen**, which renames **ChangeLight**, and a new event **ToRed** that changes a green light to red. Notice that both of these events have parameters, so the new **ToGreen** and **ToRed** events are now genuinely parameterised and don't depend on a single constant value.

The change to parameterisation has virtually no effect on the **ChangeLight** machine but has significant effects on the refinement. The refinement now has a variable, *rdir*, which is used to the parameter for both the **ToGreen** and **ToRed** events. Also notice that there are two new variables *togreen* and *tored*. The

invariant reveals that both of these variables are of type *BOOL*. The purpose of these two variables is will be explained a little later.

Both *BOOL* variables are initialised to *FALSE*. Notice that different names have been chosen for the events in the refinement to reduce confusion now that we are refining two abstract operations.

Again, notice that the refinement of the *ToGreen* event fires only when the conflicting directions have been set to *Red*, but also notice that this event will only fire if the *BOOL* variable *togreen* is *TRUE*.

A new event **ToGreenInit** appears. The purpose of this event is to initialise the state for the sequence of events that effectively refine the *ToGreen* event. **ToGreenInit** can only fire when the *BOOL* variables *togreen* and *tored* are both *FALSE*.

New events **GreenToAmber** and **AmberToRead** produce transitions of the *Green* conflicting lights through to *Red*. Each of these events can only fire when *togreen* is *TRUE*. Notice a similar initialisation event **ToRedInit** for the refinement of the **ToRed** event.

The *BOOL* variables are required because we are refining two atomic events **ToGreen** and **ToRed** and the refinement involves a sequence of events. We need to prevent interference and preserve the atomicity of the **ToGreen** and **ToRed** events. The *BOOL* variables ensure that the complete sequence of events making up a refinement are completed before any other event can be triggered. This problem did not arise in the previous development because the **ToGreen** event had a fixed parameter and there was no possibility for interference.

### TrafficLights2\_ctx context

**CONTEXT** TrafficLights2\_ctx

**SETS** *LIGHTS* *DIRECTION*

**CONSTANTS** *Red* *Green* *Amber* *CONFLICT*

#### AXIOMS

*axm1:*  $partition(LIGHTS, \{Red\}, \{Green\}, \{Amber\})$

*axm2:*  $finite(DIRECTION)$

*DIRECTION* is a finite set of directions

*axm3:*  $CONFLICT \in DIRECTION \leftrightarrow DIRECTION$

*CONFLICT* relates conflicting directions

*axm4:*  $CONFLICT \cap (DIRECTION \triangleleft id) = \emptyset$

a direction cannot conflict with itself

*axm5:*  $CONFLICT^{-1} = CONFLICT$

conflicts are symmetric

*thm1:*  $\forall d. d \in DIRECTION \Rightarrow d \notin CONFLICT[\{d\}]$

*thm2:*  $\forall d1, d2. d1 \notin CONFLICT[\{d2\}] \Rightarrow d2 \notin CONFLICT[\{d1\}]$

**END**

## ChangeLight2 machine

**MACHINE** ChangeLight2

**SEES** TrafficLights2\_ctx

**VARIABLES** *lights*

### INVARIANTS

*inv1*:  $lights \in DIRECTION \rightarrow \{Red, Green\}$

*inv2*:  $\forall d \cdot d \in DIRECTION \wedge lights(d) = Green$   
 $\Rightarrow lights[CONFLICT[\{d\}]] \subseteq \{Red\}$

Safety

*thm1*:  $finite(lights)$

### EVENTS

#### Initialisation

**begin**

*act1*:  $lights : |lights' \in DIRECTION \rightarrow \{Red, Green\}$   
 $\wedge (\forall d \cdot d \in DIRECTION \wedge lights'(d) = Green$   
 $\Rightarrow lights'[CONFLICT[\{d\}]] \subseteq \{Red\})$

**end**

**Event** *ToGreen*  $\hat{=}$

**any** *adir*

**when**

*grd1*:  $lights(adir) = Red$

**then**

*act1*:  $lights := lights$   
 $\Leftarrow (CONFLICT[\{adir\}] \times \{Red\})$   
 $\Leftarrow \{adir \mapsto Green\}$

**end**

**Event** *ToRed*  $\hat{=}$

**any** *adir*

**when**

*grd1*:  $lights(adir) = Green$

**then**

*act1*:  $lights(adir) := Red$

**end**

**END**

## ChangeLight2R refinement

**MACHINE** ChangeLight2R

**REFINES** ChangeLight2

**SEES** TrafficLights2\_ctx

**VARIABLES** *xlights* Extended lights, Red, Green and Amber lights *delay* delay between Amber and Red, Red and Green *rdir* current argument of ToGreen or ToRed *togreen* *tored*

### INVARIANTS

*inv1:*  $xlights \in DIRECTION \rightarrow LIGHTS$

*inv2:*  $\forall d \cdot d \in DIRECTION \wedge xlights[\{d\}] \subseteq \{Green, Amber\}$   
 $\Rightarrow$   
 $xlights[CONFLICT[\{d\}]] \subseteq \{Red\}$

*inv3:*  $togreen = FALSE \wedge tored = FALSE$   
 $\Rightarrow$   
 $lights = xlights$

*inv4:*  $rdir \in DIRECTION$

*inv5:*  $delay \subseteq DIRECTION$

*inv6:*  $togreen \in BOOL$

*inv7:*  $tored \in BOOL$

*inv8:*  $togreen = TRUE \Rightarrow tored = FALSE$

*inv9:*  $togreen = TRUE \Rightarrow lights(rdir) = Red$

*inv10:*  $togreen = TRUE \Rightarrow$   
 $xlights \triangleleft (CONFLICT[\{rdir\}] \times \{Red\}) \triangleleft \{rdir \mapsto Green\}$   
 $=$   
 $lights \triangleleft (CONFLICT[\{rdir\}] \times \{Red\}) \triangleleft \{rdir \mapsto Green\}$

*thm1:*  $togreen = TRUE$   
 $\Rightarrow$   
 $CONFLICT[\{rdir\}] \triangleleft (\{rdir\} \triangleleft xlights)$   
 $=$   
 $CONFLICT[\{rdir\}] \triangleleft (\{rdir\} \triangleleft lights)$

*inv11:*  $tored = TRUE \Rightarrow lights(rdir) = Green$

*inv12:*  $tored = TRUE$   
 $\Rightarrow$   
 $xlights \triangleleft \{rdir \mapsto Red\}$   
 $=$   
 $lights \triangleleft \{rdir \mapsto Red\}$

*thm2:*  $tored = TRUE$

$\Rightarrow$

$\{rdir\} \triangleleft lights$

$=$

$\{rdir\} \triangleleft xlights$

*thm3:*  $finite(xlights)$

*thm4:*  $\forall d, b, a \cdot a \in LIGHTS \wedge a \neq b$

$\wedge xlights(d) = b$

$\Rightarrow card((xlights \triangleleft \{d \mapsto a\}) \triangleright \{b\})$

$= card(xlights \triangleright \{b\}) - 1$

(= after) decreases number of colour b lights by 1

changing light in direction d from b (= before) to a

*thm5:*  $\forall d, b, a \cdot a \in LIGHTS \wedge a \neq b$

$\wedge xlights(d) = b \Rightarrow$

$card((xlights \triangleleft \{d \mapsto a\}) \triangleright \{a\})$

$=$

$card(xlights \triangleright \{a\}) + 1$

(=after) increases number of colour a lights by 1

changing light in direction d from b (= before) to a

*thm6:*  $\forall d, b, a, c \cdot a \in LIGHTS \wedge c \in LIGHTS$

$\wedge c \neq a \wedge c \neq b$

$\wedge xlights(d) = b$

$\Rightarrow$

$card((xlights \triangleleft \{d \mapsto a\}) \triangleright \{c\})$

$=$

$card(xlights \triangleright \{c\})$

to a (=after) does not change number of colour c,  $c \neq a$ ,  $c \neq b$

changing light in direction d from b (= before)

## EVENTS

### Initialisation

**begin**

**with**

*lights'* :  $lights' = xlights'$

*act1:*  $xlights : |xlights' \in DIRECTION \rightarrow \{Red, Green\}$

$\wedge (\forall d \cdot xlights'(d) = Green$

$\Rightarrow$

$xlights'[CONFLICT[\{d\}]] \subseteq \{Red\})$

*act2:*  $delay := \emptyset$

*act3:*  $togreen, tored := FALSE, FALSE$

*act5:*  $rdir \in DIRECTION$

**end**

**Event** *ToGreen*  $\hat{=}$

**refines** *ToGreen*

**when**

*grd1*:  $togreen = TRUE$

*grd2*:  $xlights[CONFLICT[\{rdir\}]] \subseteq \{Red\}$

*grd3*:  $rdir \notin delay$

**with**

*adir*:  $adir = rdir$

**then**

*act1*:  $xlights(rdir) := Green$

*act2*:  $togreen := FALSE$

**end**

**Event** *ToGreenInit*  $\hat{=}$

**any**

*adir*

**when**

*grd1*:  $togreen = FALSE$

*grd2*:  $tored = FALSE$

*grd3*:  $xlights(adir) = Red$

**then**

*act1*:  $rdir := adir$

*act2*:  $togreen := TRUE$

**end**

**Event** *GreenToAmber*  $\hat{=}$

**Status** convergent

**any**

*dir*

**when**

*grd1*:  $togreen = TRUE$

*grd2*:  $dir \in CONFLICT[\{rdir\}]$

*grd3*:  $xlights(dir) = Green$

**grd4:**  $dir \notin delay$   
**then**  
**act1:**  $xlights(dir) := Amber$   
**act2:**  $delay := delay \cup \{dir\}$   
**end**  
**Event**  $AmberToRed \hat{=}$   
**Status** convergent  
**any**  
 $dir$   
**when**  
**grd1:**  $togreen = TRUE$   
**grd2:**  $dir \in CONFLICT[\{rdir\}]$   
**grd3:**  $xlights(dir) = Amber$   
**grd4:**  $dir \notin delay$   
**then**  
**act1:**  $xlights(dir) := Red$   
**act2:**  $delay := delay \cup \{rdir\}$   
**end**  
**Event**  $Delay \hat{=}$   
**Status** convergent  
**any**  
 $dir$   
**when**  
**grd1:**  $dir \in delay$   
**then**  
**act1:**  $delay := delay \setminus \{dir\}$   
**end**  
**Event**  $ToRed \hat{=}$   
**refines**  $ToRed$   
**when**

```

grd1: tored = TRUE
grd2: xlights(rdir) = Amber
grd3: rdir ∉ delay
with
adir: adir = rdir
then
act1: xlights(rdir) := Red
act2: tored := FALSE
end
Event ToRedInit  $\hat{=}$ 
any
adir
when
grd1: xlights(adir) = Green
grd2: tored = FALSE
grd3: togreen = FALSE
then
act1: rdir := adir
act2: tored := TRUE
end
Event ToAmber  $\hat{=}$ 
when
grd1: tored = TRUE
grd2: xlights(rdir) = Green
grd3: rdir ∉ delay
then
act1: xlights(rdir) := Amber
act2: delay := delay ∪ {rdir}
end
VARIANT  $4 * \text{card}(xlights \triangleright \{Green\})$ 
 $+ 2 * \text{card}(xlights \triangleright \{Amber\})$ 
 $+ \text{card}(delay)$ 
END

```