

# Tutorial 2: Traffic Lights

Ken Robinson

25th May 2000

## Contents

<b>1</b>	<b>SimpleTwoWay</b>	<b>2</b>
<b>2</b>	<b>FiveWay</b>	<b>3</b>
<b>3</b>	<b>MultiWay</b>	<b>5</b>

# 1 SimpleTwoWay

**MACHINE** *SimpleTwoWay*

## **SETS**

$DIRECTION = \{ NorthSouth, EastWest \};$   
 $LIGHT = \{ Red, Green, Amber \}$

## **VARIABLES**

*lights*

## **INVARIANT**

$lights \in DIRECTION \rightarrow LIGHT \wedge$   
 $( lights ( NorthSouth ) \in \{ Green, Amber \} \Rightarrow lights ( EastWest ) = Red ) \wedge$   
 $( lights ( EastWest ) \in \{ Green, Amber \} \Rightarrow lights ( NorthSouth ) = Red )$

## **INITIALISATION**

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

## **OPERATIONS**

**ToRed** ( *dir* )  $\hat{=}$

**PRE**  $dir \in DIRECTION \wedge lights ( dir ) = Amber$   
**THEN**  $lights ( dir ) := Red$   
**END** ;

**ToGreen** ( *dir* )  $\hat{=}$

**PRE**  $dir \in DIRECTION \wedge lights ( dir ) = Red \wedge$   
 $( dir = NorthSouth \Rightarrow lights ( EastWest ) = Red ) \wedge$   
 $( dir = EastWest \Rightarrow lights ( NorthSouth ) = Red )$   
**THEN**  $lights ( dir ) := Green$   
**END** ;

**ToAmber** ( *dir* )  $\hat{=}$

**PRE**  $dir \in DIRECTION \wedge lights ( dir ) = Green$   
**THEN**  $lights ( dir ) := Amber$   
**END**

**END**

## 2 FiveWay

This machine adds *NorthRight* and *SouthRight* lights giving five conflicting directions

**MACHINE** *FiveWay*

### SETS

$DIRECTION = \{ North , NorthRight , South , SouthRight , EastWest \} ;$   
 $LIGHTS = \{ Red , Green , Amber \}$

### VARIABLES

*lights*

### INVARIANT

$lights \in DIRECTION \rightarrow LIGHTS \wedge$   
 $( lights ( North ) \in \{ Green , Amber \} \Rightarrow$   
 $lights [ \{ EastWest , SouthRight \} ] = \{ Red \} ) \wedge$   
 $( lights ( South ) \in \{ Green , Amber \} \Rightarrow$   
 $lights [ \{ EastWest , NorthRight \} ] = \{ Red \} ) \wedge$   
 $( lights ( NorthRight ) \in \{ Green , Amber \} \Rightarrow$   
 $lights [ \{ EastWest , South \} ] = \{ Red \} ) \wedge$   
 $( lights ( SouthRight ) \in \{ Green , Amber \} \Rightarrow$   
 $lights [ \{ EastWest , North \} ] = \{ Red \} ) \wedge$   
 $( lights ( EastWest ) \in \{ Green , Amber \} \Rightarrow$   
 $lights [ \{ North , South , NorthRight , SouthRight \} ] = \{ Red \} )$

### INITIALISATION

$lights := DIRECTION \times \{ Red \}$

### OPERATIONS

**ToRed** ( *dir* )  $\hat{=}$

**PRE**  $dir \in DIRECTION \wedge lights ( dir ) = Amber$

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

**END** ;

**ToGreen** ( *dir* )  $\hat{=}$

**PRE**  $dir \in DIRECTION \wedge$

$lights ( dir ) = Red \wedge$

$( dir = North \Rightarrow lights [ \{ EastWest , SouthRight \} ] = \{ Red \} ) \wedge$

$( dir = South \Rightarrow lights [ \{ EastWest , NorthRight \} ] = \{ Red \} ) \wedge$

$( dir = NorthRight \Rightarrow lights [ \{ EastWest , South \} ] = \{ Red \} ) \wedge$

$( dir = SouthRight \Rightarrow lights [ \{ EastWest , North \} ] = \{ Red \} ) \wedge$

$( dir = EastWest \Rightarrow$

$lights [ \{ North , NorthRight , South , SouthRight \} ] = \{ Red \} )$

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

**END** ;

```
ToAmber ( dir )  $\hat{=}$   
  PRE   dir  $\in$  DIRECTION  $\wedge$  lights ( dir ) = Green  
  THEN  lights ( dir ) := Amber  
  END  
END
```

### 3 MultiWay

**MACHINE** *MultiWay*

#### **TRAFFIC LIGHT KERNEL**

This machine is due to Paul Ammann, George Mason University, with input from Jeremy Dick, B-Core (UK) and extra documentation by Ken Robinson, UNSW.

It captures the essential safety properties of traffic lights at an arbitrary junction.

#### **SETS**

*DIRECTION* models the various directions in which traffic may approach the junction.

*DIRECTION* ;

*LIGHT* models the standard tri-colour system.

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

#### **CONSTANTS**

*conflict* is a relation modelling how pairs of directions conflict with each other.

*conflict*

#### **PROPERTIES**

The tricky bits here are that

- a direction cannot conflict with itself, so that no part of the identity relation is contained in *conflict*.
- a relation is syntactically directional, having a domain and a range. We can model true pairs by insisting that the relation is its own inverse.

$conflict \in DIRECTION \leftrightarrow DIRECTION \wedge$

$conflict \cap id ( DIRECTION ) = \{ \} \wedge$

$conflict^{-1} = conflict$

#### **VARIABLES**

*lights* models the light for each direction.

*lights*

#### **INVARIANT**

Here is the safety condition: all lights that conflict with a amber or green light must be red.

Notice how the use of quantifiers is avoided here. Quantifiers are harder to prove.

$lights \in DIRECTION \rightarrow LIGHT \wedge$

$lights^{-1} ; conflict ; lights [ \{ Amber, Green \} ] \subseteq \{ Red \}$

Note that  $lights^{-1}[\{Amber, Green\}]$  gives the set of directions that are currently showing a *Green* or *Amber* light.

$(lights^{-1}; conflict)[\{Amber, Green\}]$  gives the set of directions that conflict with direction showing *Green* or *Amber*.

$(lights^{-1}; conflict; lights)[\{Amber, Green\}]$  gives the set of lights displayed in those conflicting directions.

Safety requires that set to be  $\{Red\}$  or  $\{\}$ , thus  $\subseteq \{Red\}$ .

**Note:**  $;$  in  $r_1;r_2$  is forward relational composition. Think of it as joining the arrows of  $r_1$  and  $r_2$ , where that is possible. For this composition to be valid, the range set of  $r_1$  must be the same as the domain set of  $r_2$ .

## INITIALISATION

All lights are Red to start with.

$lights := DIRECTION \times \{ Red \}$

## OPERATIONS

### ToRed

Safety: The sequencing constraint coupled with the state safety constraint implies that the lights in all conflicting directions are *Red*.

Sequencing: Only an Amber light can change to Red.

**ToRed** ( *dir* )  $\hat{=}$

**PRE**

$dir \in DIRECTION \wedge$

$lights ( dir ) = Amber$

**THEN**

$lights ( dir ) := Red$

**END ;**

### ToGreen

Safety: the lights in all conflicting directions must be *Red*.

Sequencing: Only a *Red* light can turn to *Green*.

**ToGreen** ( *dir* )  $\hat{=}$

**PRE**

$dir \in DIRECTION \wedge$

$lights ( dir ) = Red \wedge$

$conflict ; lights [ \{ dir \} ] \subseteq \{ Red \}$

**THEN**

$lights ( dir ) := Green$

**END ;**

**ToAmber**

Safety: The sequencing constraint and the state safety constraint implies that the lights in all conflicting directions will be *Red*.

Sequencing: Only a Green light can change to Amber.

**ToAmber** ( *dir* )  $\hat{=}$ **PRE** $dir \in DIRECTION \wedge$  $lights ( dir ) = Green$ **THEN** $lights ( dir ) := Amber$ **END****END**