System Modelling and Design

Traffic Lights

Revision: 1.4, March 21, 2007

Ken Robinson

School of Computer Science & Engineering
The University of New South Wales, Sydney Australia

March 22, 2007

© Ken Robinson 2005

mailto::k.robinson@unsw.edu.au
Outline II

- The precondition of ToGreen
- Equivalent predicates
- The precondition of ToAmber

The TwoWay Machine
Objectives of this lecture

- To explore the specification of some simple traffic light controllers.
- To explore the use of a state invariant to ensure safety.
- To explore the use of preconditions that ensure an operation will not violate the state invariant, when the precondition is satisfied.
- To use the animator to illustrate these explorations.
Objectives of this lecture

- To explore the specification of some simple traffic light controllers.
- To explore the use of a state invariant to ensure safety.
- To explore the use of preconditions that ensure an operation will not violate the state invariant, when the precondition is satisfied.
- To use the animator to illustrate these explorations.
Objectives of this lecture

- To explore the specification of some simple traffic light controllers.
- To explore the use of a state invariant to ensure safety.
- To explore the use of preconditions that ensure an operation will not violate the state invariant, when the precondition is satisfied.
- To use the animator to illustrate these explorations.
Objectives of this lecture

- To explore the specification of some simple traffic light controllers.
- To explore the use of a state invariant to ensure safety.
- To explore the use of preconditions that ensure an operation will not violate the state invariant, when the precondition is satisfied.
- To use the animator to illustrate these explorations.
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.
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.
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 $\rightarrow$ Green $\rightarrow$ Amber $\rightarrow$ Red $\rightarrow$ …

We wish to specify a traffic light controller that ensures safety and the correct sequencing.
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.
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.
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 $\rightarrow$ Green $\rightarrow$ Amber $\rightarrow$ Red $\rightarrow$ ...

We wish to specify a traffic light controller that ensures safety and the correct sequencing.
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 $\Rightarrow$ Green $\Rightarrow$ Amber $\Rightarrow$ Red $\Rightarrow$ …

We wish to specify a traffic light controller that ensures safety and the correct sequencing.
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.
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 $\rightarrow$ Green $\rightarrow$ Amber $\rightarrow$ Red $\rightarrow$ ...

We wish to specify a traffic light controller that ensures safety and the correct sequencing.
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.
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.
We will introduce a context machine containing the enumerated sets $DIRECTION$ and $LIGHT$.

We will also specify a constant (function) $OTHER_DIR$ that maps each direction into the other direction.

Context machines are used commonly and frequently in the B Method ($B$) to define sets and constants.
We will introduce a context machine containing the enumerated sets \textit{DIRECTION} and \textit{LIGHT}.

We will also specify a constant (function) \textit{OTHER\_DIR} that maps each direction into the other direction.

Context machines are used commonly and frequently in B to define sets and constants.
We will introduce a context machine containing the enumerated sets \textit{DIRECTION} and \textit{LIGHT}.

We will also specify a constant (function) \textit{OTHER\_DIR} that maps each direction into the other direction.

\textit{Context machines are used commonly and frequently in B to define sets and constants.}
MACHINE TwoWay_Ctx

SETS

\[ \text{DIRECTION} = \{ \text{NorthSouth, EastWest} \} \; ; \]
\[ \text{LIGHT} = \{ \text{Red, Green, Amber} \} \]

CONSTANTS OTHER_DIR

PROPERTIES

\[ \text{OTHER_DIR} \in \text{DIRECTION} \rightarrow \text{DIRECTION} \land \]
\[ \text{OTHER_DIR} = \{ \text{NorthSouth} \leftrightarrow \text{EastWest}, \text{EastWest} \leftrightarrow \text{NorthSouth} \} \]

END
We will write a basic machine with no non-trivial state invariant and no non-trivial preconditions.

The machine will see the context machine and have a state of one variable, \textit{lights}, which is a total function from \textit{DIRECTION} to \textit{LIGHT}.

There is one operation: \textit{ChangeLight(dir, colour)} that changes the light to \textit{colour} in the direction \textit{dir}.
The Simple Two Way Machine

1. We will write a basic machine with no non-trivial state invariant and no non-trivial preconditions.
2. The machine will see the context machine and have a state of one variable, \( \text{lights} \), which is a total function from \( \text{DIRECTION} \) to \( \text{LIGHT} \).
3. There is one operation: \( \text{ChangeLight}(\text{dir}, \text{colour}) \) that changes the light to \( \text{colour} \) in the direction \( \text{dir} \)
We will write a basic machine with no non-trivial state invariant and no non-trivial preconditions.

The machine will see the context machine and have a state of one variable, \textit{lights}, which is a total function from \textit{DIRECTION} to \textit{LIGHT}.

There is one operation: \textit{ChangeLight(dir, colour)} that changes the light to \textit{colour} in the direction \textit{dir}. 

The SimpleTwoWay Machine
1. We will write a basic machine with no non-trivial state invariant and no non-trivial preconditions.

2. The machine will see the context machine and have a state of one variable, $lights$, which is a total function from $DIRECTION$ to $LIGHT$.

3. There is one operation: $\text{ChangeLight}(dir, colour)$ that changes the light to $colour$ in the direction $dir$. 
MACHINE SimpleTwoWay0
SEES TwoWayCtx
VARIABLES lights
INARIANT lights ∈ DIRECTION → LIGHT
INITIALISATION lights := { NorthSouth ↦ Red, EastWest ↦ Red }

OPERATIONS
  ChangeLight( dir, colour ) ≜
  pre dir ∈ DIRECTION ∧ colour ∈ LIGHT
  then lights( dir ) := colour
end

END
Use the animator to explore the behaviour. When animating, choose to display the invariant, normally turned off.

It is, of course, trivial to establish that the controller is unsafe.

Try to formulate an invariant that will ensure safety.

Whenever the state is unsafe, the invariant must be false.

\[ \neg (\text{safe}) \Rightarrow \neg (\text{invariant}) \]  \hspace{1cm} (1)

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

\[ \text{invariant} \Rightarrow \text{safe} \]  \hspace{1cm} (2)

Of course, 1 and 2 are equivalent; one is the contrapositive of the other: \( P \Rightarrow Q \equiv \neg Q \Rightarrow \neg P \).

If we have the weakest invariant, then whenever the state is safe, the invariant will be true.

Find adequately strong preconditions.
The Invariant

1. Use the animator to explore the behaviour. When animating, choose to display the invariant, normally turned off.

2. It is, of course, trivial to establish that the controller is unsafe.

3. Try to formulate an invariant that will ensure safety.

4. Whenever the state is unsafe, the invariant must be false.

\[ \neg (\text{safe}) \Rightarrow \neg (\text{invariant}) \]  \hspace{1cm} (1)

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

\[ \text{invariant} \Rightarrow \text{safe} \]  \hspace{1cm} (2)

6. Of course, 1 and 2 are equivalent; one is the contrapositive of the other: \( P \Rightarrow Q \equiv \neg Q \Rightarrow \neg P \)

7. If we have the weakest invariant, then whenever the state is safe, the invariant will be true.

8. Find adequately strong preconditions.
Use the animator to explore the behaviour. When animating, choose to display the invariant, normally turned off.

It is, of course, trivial to establish that the controller is unsafe.

Try to formulate an invariant that will ensure safety.

Whenever the state is unsafe, the invariant must be false.

\[ \neg(safe) \Rightarrow \neg(invariant) \] (1)

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

\[ invariant \Rightarrow safe \] (2)

Of course, 1 and 2 are equivalent; one is the contrapositive of the other: \[ P \Rightarrow Q \equiv \neg Q \Rightarrow \neg P \]

If we have the weakest invariant, then whenever the state is safe, the invariant will be true.

Find adequately strong preconditions.
1. Use the animator to explore the behaviour. When animating, choose to display the invariant, normally turned off.

2. It is, of course, trivial to establish that the controller is unsafe.

3. **Try to formulate an invariant that will ensure safety.**

4. Whenever the state is unsafe, the invariant must be false.

   \[ \neg (\text{safe}) \Rightarrow \neg (\text{invariant}) \]  

   \( (1) \)

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

   \[ \text{invariant} \Rightarrow \text{safe} \]  

   \( (2) \)

6. Of course, 1 and 2 are equivalent; one is the **contrapositive** of the other: \( P \Rightarrow Q \equiv \neg Q \Rightarrow \neg P \)

7. If we have the **weakest** invariant, then whenever the state is safe, the invariant will be true.

8. Find adequately strong preconditions.
The Invariant

1. Use the animator to explore the behaviour. When animating, choose to display the invariant, normally turned off.

2. It is, of course, trivial to establish that the controller is unsafe.

3. Try to formulate an invariant that will ensure safety.

4. Whenever the state is unsafe, the invariant must be \textit{false}.

\[ \neg (\text{safe}) \Rightarrow \neg (\text{invariant}) \quad (1) \]

5. Conversely, whenever the invariant is \textit{true}, the state should be safe.

\[ \text{invariant} \Rightarrow \text{safe} \quad (2) \]

6. Of course, 1 and 2 are equivalent; one is the \textit{contrapositive} of the other: \[ P \Rightarrow Q \equiv \neg Q \Rightarrow \neg P \]

7. If we have the \textit{weakest} invariant, then whenever the state is safe, the invariant will be \textit{true}.

8. Find adequately strong preconditions.
The Invariant

1. Use the animator to explore the behaviour. When animating, choose to display the invariant, normally turned off.
2. It is, of course, trivial to establish that the controller is unsafe.
3. Try to formulate an invariant that will ensure safety.
4. Whenever the state is unsafe, the invariant must be false.

\[ \neg (\text{safe}) \Rightarrow \neg (\text{invariant}) \]  
(1)

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

\[ \text{invariant} \Rightarrow \text{safe} \]  
(2)

6. Of course, 1 and 2 are equivalent; one is the contrapositive of the other: \( P \Rightarrow Q \equiv \neg Q \Rightarrow \neg P \)

7. If we have the weakest invariant, then whenever the state is safe, the invariant will be true.

8. Find adequately strong preconditions.
The Invariant

1. Use the animator to explore the behaviour. When animating, choose to display the invariant, normally turned off.

2. It is, of course, trivial to establish that the controller is unsafe.

3. Try to formulate an invariant that will ensure safety.

4. Whenever the state is unsafe, the invariant must be *false*.

   \[ \neg (\text{safe}) \Rightarrow \neg (\text{invariant}) \]  

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

   \[ \text{invariant} \Rightarrow \text{safe} \]  

6. Of course, 1 and 2 are equivalent; one is the *contrapositive* of the other: \( P \Rightarrow Q \equiv \neg Q \Rightarrow \neg P \)

7. If we have the *weakest* invariant, then whenever the state is safe, the invariant will be *true*.

8. Find adequately strong preconditions.
The Invariant

1. Use the animator to explore the behaviour. When animating, choose to display the invariant, normally turned off.

2. It is, of course, trivial to establish that the controller is unsafe.

3. Try to formulate an invariant that will ensure safety.

4. Whenever the state is unsafe, the invariant must be false.

\[
\neg (safe) \Rightarrow \neg (invariant) \quad (1)
\]

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

\[
invariant \Rightarrow safe \quad (2)
\]

6. Of course, 1 and 2 are equivalent; one is the contrapositive of the other: \( P \Rightarrow Q \equiv \neg Q \Rightarrow \neg P \)

7. If we have the weakest invariant, then whenever the state is safe, the invariant will be true.

8. Find adequately strong preconditions.
The Invariant

1. Use the animator to explore the behaviour. When animating, choose to display the invariant, normally turned off.

2. It is, of course, trivial to establish that the controller is unsafe.

3. Try to formulate an invariant that will ensure safety.

4. Whenever the state is unsafe, the invariant must be false.

\[ \neg(safe) \Rightarrow \neg(invariant) \]  \hspace{1cm} (1)

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

\[ invariant \Rightarrow safe \]  \hspace{1cm} (2)

6. Of course, 1 and 2 are equivalent; one is the contrapositive of the other: \( P \Rightarrow Q \equiv \neg Q \Rightarrow \neg P \)

7. If we have the weakest invariant, then whenever the state is safe, the invariant will be true.

8. Find adequately strong preconditions.
Clearly, the light in both directions cannot be either Green or Amber.

<table>
<thead>
<tr>
<th></th>
<th>Red</th>
<th>Green</th>
<th>Amber</th>
</tr>
</thead>
<tbody>
<tr>
<td>Red</td>
<td>safe</td>
<td>safe</td>
<td>safe</td>
</tr>
<tr>
<td>Green</td>
<td>safe</td>
<td>unsafe</td>
<td>unsafe</td>
</tr>
<tr>
<td>Amber</td>
<td>safe</td>
<td>unsafe</td>
<td>unsafe</td>
</tr>
</tbody>
</table>

This leads to the invariant:

\[
\neg (\text{lights}(\text{NorthSouth}) \in \{\text{Green, Amber}\} \land \text{lights}(\text{EastWest}) \in \{\text{Green, Amber}\})
\]

\[
\equiv
\neg (\text{lights}(\text{NorthSouth}) \in \{\text{Green, Amber}\} \Rightarrow \neg (\text{lights}(\text{EastWest}) \in \{\text{Green, Amber}\}))
\]

\[
\equiv
(\text{lights}(\text{NorthSouth}) \in \{\text{Green, Amber}\} \Rightarrow \text{lights}(\text{EastWest}) = \text{Red})
\]
Clearly, the light in both directions cannot be either Green or Amber.

This leads to the invariant:

\[ \neg (\text{lights}(\text{NorthSouth}) \in \{\text{Green}, \text{Amber}\} \land \text{lights}(\text{EastWest}) \in \{\text{Green}, \text{Amber}\}) \]

\[ \equiv (\text{lights}(\text{NorthSouth}) \in \{\text{Green}, \text{Amber}\} \rightarrow \neg (\text{lights}(\text{EastWest}) \in \{\text{Green}, \text{Amber}\})) \]

\[ \equiv (\text{lights}(\text{NorthSouth}) \in \{\text{Green}, \text{Amber}\} \rightarrow \text{lights}(\text{EastWest}) = \text{Red}) \]
Strengthening the invariant

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

<table>
<thead>
<tr>
<th></th>
<th>Red</th>
<th>Green</th>
<th>Amber</th>
</tr>
</thead>
<tbody>
<tr>
<td>Red</td>
<td>safe</td>
<td>safe</td>
<td>safe</td>
</tr>
<tr>
<td>Green</td>
<td>safe</td>
<td>unsafe</td>
<td>unsafe</td>
</tr>
<tr>
<td>Amber</td>
<td>safe</td>
<td>unsafe</td>
<td>unsafe</td>
</tr>
</tbody>
</table>

This leads to the invariant:

$$
\neg \left( \text{lights}(\text{NorthSouth}) \in \{\text{Green}, \text{Amber}\} \land \text{lights}(\text{EastWest}) \in \{\text{Green}, \text{Amber}\} \right)
\equiv
\left( \text{lights}(\text{NorthSouth}) \in \{\text{Green}, \text{Amber}\} \Rightarrow
\neg \left( \text{lights}(\text{EastWest}) \in \{\text{Green}, \text{Amber}\} \right) \right)
\equiv
\left( \text{lights}(\text{NorthSouth}) \in \{\text{Green}, \text{Amber}\} \Rightarrow \text{lights}(\text{EastWest}) = \text{Red} \right)$$
## Strengthening the invariant

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

<table>
<thead>
<tr>
<th></th>
<th>Red</th>
<th>Green</th>
<th>Amber</th>
</tr>
</thead>
<tbody>
<tr>
<td>Safe</td>
<td>safe</td>
<td>safe</td>
<td>safe</td>
</tr>
<tr>
<td></td>
<td>unsafe</td>
<td>unsafe</td>
<td>unsafe</td>
</tr>
</tbody>
</table>

This leads to the invariant:

\[
\neg (\text{lights}(\text{NorthSouth}) \in \{\text{Green, Amber}\} \land \\
\text{lights}(\text{EastWest}) \in \{\text{Green, Amber}\}) \\
\equiv \\
(\text{lights}(\text{NorthSouth}) \in \{\text{Green, Amber}\} \Rightarrow \\
\neg (\text{lights}(\text{EastWest}) \in \{\text{Green, Amber}\})) \\
\equiv \\
(\text{lights}(\text{NorthSouth}) \in \{\text{Green, Amber}\} \Rightarrow \\
\text{lights}(\text{EastWest}) = \text{Red})
\]
**Strengthening the invariant**

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

<table>
<thead>
<tr>
<th>Red</th>
<th>Green</th>
<th>Amber</th>
</tr>
</thead>
<tbody>
<tr>
<td>safe</td>
<td>safe</td>
<td>safe</td>
</tr>
<tr>
<td>unsafe</td>
<td>unsafe</td>
<td>unsafe</td>
</tr>
<tr>
<td>safe</td>
<td>unsafe</td>
<td>unsafe</td>
</tr>
</tbody>
</table>

This leads to the invariant:

\[
\neg (\text{lights}(\text{NorthSouth}) \in \{\text{Green}, \text{Amber}\} \land \text{lights}(\text{EastWest}) \in \{\text{Green}, \text{Amber}\})
\equiv
\neg (\text{lights}(\text{NorthSouth}) \in \{\text{Green}, \text{Amber}\} \Rightarrow
\neg (\text{lights}(\text{EastWest}) \in \{\text{Green}, \text{Amber}\}))
\equiv
\neg (\text{lights}(\text{NorthSouth}) \in \{\text{Green}, \text{Amber}\} \Rightarrow \text{lights}(\text{EastWest}) = \text{Red})
Clearly, the light in both directions cannot be either Green or Amber.

<table>
<thead>
<tr>
<th></th>
<th>Red</th>
<th>Green</th>
<th>Amber</th>
</tr>
</thead>
<tbody>
<tr>
<td>Red</td>
<td>safe</td>
<td>safe</td>
<td>safe</td>
</tr>
<tr>
<td>Green</td>
<td>safe</td>
<td>unsafe</td>
<td>unsafe</td>
</tr>
<tr>
<td>Amber</td>
<td>safe</td>
<td>unsafe</td>
<td>unsafe</td>
</tr>
</tbody>
</table>

This leads to the invariant:

\[
\neg (\text{lights}(\text{NorthSouth}) \in \{\text{Green, Amber}\} \land \text{lights}(\text{EastWest}) \in \{\text{Green, Amber}\})
\]

\[
\equiv
\]

\[
(\text{lights}(\text{NorthSouth}) \in \{\text{Green, Amber}\} \Rightarrow
\neg (\text{lights}(\text{EastWest}) \in \{\text{Green, Amber}\}))
\]

\[
\equiv
\]

\[
(\text{lights}(\text{NorthSouth}) \in \{\text{Green, Amber}\} \Rightarrow
\text{lights}(\text{EastWest}) = \text{Red})
\]
Clearly, the light in both directions cannot be either **Green** or **Amber**.

<table>
<thead>
<tr>
<th></th>
<th>Red</th>
<th>Green</th>
<th>Amber</th>
</tr>
</thead>
<tbody>
<tr>
<td>Red</td>
<td>safe</td>
<td>safe</td>
<td>safe</td>
</tr>
<tr>
<td>Green</td>
<td>safe</td>
<td>unsafe</td>
<td>unsafe</td>
</tr>
<tr>
<td>Amber</td>
<td>safe</td>
<td>unsafe</td>
<td>unsafe</td>
</tr>
</tbody>
</table>

This leads to the invariant:

\[
\neg (\text{lights}(\text{NorthSouth}) \in \{\text{Green}, \text{Amber}\} \land \\
\text{lights}(\text{EastWest}) \in \{\text{Green}, \text{Amber}\})
\equiv \\
(\text{lights}(\text{NorthSouth}) \in \{\text{Green}, \text{Amber}\} \Rightarrow \\
\neg (\text{lights}(\text{EastWest}) \in \{\text{Green}, \text{Amber}\}))
\equiv \\
(\text{lights}(\text{NorthSouth}) \in \{\text{Green}, \text{Amber}\} \Rightarrow \\
\text{lights}(\text{EastWest}) = \text{Red})
\]
### Strengthening the invariant

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

<table>
<thead>
<tr>
<th></th>
<th>Red</th>
<th>Green</th>
<th>Amber</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Red</strong></td>
<td>safe</td>
<td>safe</td>
<td>safe</td>
</tr>
<tr>
<td><strong>Green</strong></td>
<td>safe</td>
<td>unsafe</td>
<td>unsafe</td>
</tr>
<tr>
<td><strong>Amber</strong></td>
<td>safe</td>
<td>unsafe</td>
<td>unsafe</td>
</tr>
</tbody>
</table>

This leads to the invariant:

\[
\neg (\text{lights}(\text{NorthSouth}) \in \{\text{Green, Amber}\} \land \text{lights}(\text{EastWest}) \in \{\text{Green, Amber}\})
\]

\[\equiv (\text{lights}(\text{NorthSouth}) \in \{\text{Green, Amber}\} \Rightarrow \\
\neg (\text{lights}(\text{EastWest}) \in \{\text{Green, Amber}\})))
\]

\[\equiv (\text{lights}(\text{NorthSouth}) \in \{\text{Green, Amber}\} \Rightarrow \\
\text{lights}(\text{EastWest}) = \text{Red})
\]
Clearly, the light in both directions cannot be either Green or Amber.

<table>
<thead>
<tr>
<th></th>
<th>Red</th>
<th>Green</th>
<th>Amber</th>
</tr>
</thead>
<tbody>
<tr>
<td>Red</td>
<td>safe</td>
<td>safe</td>
<td>safe</td>
</tr>
<tr>
<td>Green</td>
<td>safe</td>
<td>unsafe</td>
<td>unsafe</td>
</tr>
<tr>
<td>Amber</td>
<td>safe</td>
<td>unsafe</td>
<td>unsafe</td>
</tr>
</tbody>
</table>

This leads to the invariant:

\[ \neg (\text{lights}(\text{NorthSouth}) \in \{\text{Green, Amber}\} \land \text{lights}(\text{EastWest}) \in \{\text{Green, Amber}\}) \]

\[ \equiv \]

\[ (\text{lights}(\text{NorthSouth}) \in \{\text{Green, Amber}\} \Rightarrow \neg (\text{lights}(\text{EastWest}) \in \{\text{Green, Amber}\})) \]

\[ \equiv \]

\[ (\text{lights}(\text{NorthSouth}) \in \{\text{Green, Amber}\} \Rightarrow \text{lights}(\text{EastWest}) = \text{Red}) \]
Clearly, the light in both directions cannot be either Green or Amber.

<table>
<thead>
<tr>
<th></th>
<th>Red</th>
<th>Green</th>
<th>Amber</th>
</tr>
</thead>
<tbody>
<tr>
<td>Red</td>
<td>safe</td>
<td>safe</td>
<td>safe</td>
</tr>
<tr>
<td>Green</td>
<td>safe</td>
<td>unsafe</td>
<td>unsafe</td>
</tr>
<tr>
<td>Amber</td>
<td>safe</td>
<td>unsafe</td>
<td>unsafe</td>
</tr>
</tbody>
</table>

This leads to the invariant:

\[
\neg (\text{lights}(\text{NorthSouth}) \in \{\text{Green, Amber}\} \land \\
\text{lights}(\text{EastWest}) \in \{\text{Green, Amber}\})
\]

\[
\equiv \\
(\text{lights}(\text{NorthSouth}) \in \{\text{Green, Amber}\} \Rightarrow \\
\neg (\text{lights}(\text{EastWest}) \in \{\text{Green, Amber}\}))
\]

\[
\equiv \\
(\text{lights}(\text{NorthSouth}) \in \{\text{Green, Amber}\} \Rightarrow \\
\text{lights}(\text{EastWest}) = \text{Red})
\]
There are other invariants that adequately express safety for a two-way intersection:

\[ \text{lights}(\text{NorthSouth}) = \text{Red} \lor \text{lights}(\text{EastWest}) = \text{Red} \]

\[ \text{Red} \in \text{ran}(\text{lights}) \]

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

\[ \forall \text{dir}. (\text{dir} \in \text{DIRECTION} \land \text{lights}(\text{dir}) \in \{\text{Green}, \text{Amber}\} \Rightarrow \text{lights}(\text{OTHER_DIR}(\text{dir})) = \text{Red}) \]
Other Invariants

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

\[
\text{lights}(\text{NorthSouth}) = \text{Red} \lor \text{lights}(\text{EastWest}) = \text{Red}
\]

\[
\text{Red} \in \text{ran(lights)}
\]

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

\[
\forall \ dir. (dir \in \text{DIRECTION} \land \text{lights}(dir) \in \{\text{Green, Amber}\} \Rightarrow \text{lights}(\text{OTHER_DIR(dir)}) = \text{Red})
\]
Other Invariants

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

\[
lights(NorthSouth) = Red \lor lights(EastWest) = Red
\]

\[
Red \in \text{ran}(lights)
\]

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

\[
\forall dir.\ (dir \in DIRECTION \land lights(dir) \in \{Green, Amber\} \Rightarrow lights(\text{OTHER_DIR}(dir)) = Red)
\]
There are other invariants that adequately express safety for a two-way intersection:

\[ \text{lights}(\text{NorthSouth}) = \text{Red} \lor \text{lights}(\text{EastWest}) = \text{Red} \]

\[ \text{Red} \in \text{ran}(\text{lights}) \]

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

\[ \forall \text{dir}. \left( \text{dir} \in \text{DIRECTION} \land \text{lights}(\text{dir}) \in \{\text{Green}, \text{Amber}\} \Rightarrow \text{lights} (\text{OTHER\_DIR}(\text{dir})) = \text{Red} \right) \]
Other Invariants

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

\[ \text{\textit{lights}}(\text{NorthSouth}) = \text{Red} \lor \text{\textit{lights}}(\text{EastWest}) = \text{Red} \]

\[ \text{Red} \in \text{ran}(\text{\textit{lights}}) \]

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

\[ \forall \text{dir}. (\text{dir} \in \text{\textit{DIRECTION}} \land \text{\textit{lights}}(\text{dir}) \in \{\text{Green, Amber}\} \Rightarrow \text{\textit{lights}}(\text{OTHER\_DIR}(\text{dir})) = \text{Red}) \]
Other Invariants

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

\[ \text{lights(NorthSouth)} = \text{Red} \lor \text{lights(EastWest)} = \text{Red} \]

\[ \text{Red} \in \text{ran(lights)} \]

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

\[ \forall \text{dir}. (\text{dir} \in \text{DIRECTION} \land \text{lights(dir)} \in \{\text{Green}, \text{Amber}\} \Rightarrow \text{lights(OTHER_DIR(dir)) = Red}) \]
Further Animation

Try animating with the invariant on the preceding slide, using the animation script on the following slide.
ANIMATE
   SimpleTwoWay0.test1.anm
PARAMETER_VALUES
   ?
SETS_VALUES
   ?
CONSTANTS_VALUES
   OTHER_DIR = {NorthSouth |-> EastWest, EastWest |-> NorthSouth}
ENUM_SETS_VALUES
   DIRECTION = {NorthSouth, EastWest};
   LIGHT = {Red, Green, Amber}
OPERATIONS
   INI_SimpleTwoWay0;
   ChangeLight(NorthSouth,Green);
   ChangeLight(EastWest,Green);
   undo;
   ChangeLight(EastWest,Amber);
   undo;
   ChangeLight(NorthSouth,Amber);
   ChangeLight(EastWest,Amber)
END
We now have states and arguments of the operation \textit{ChangeLight} that lead to a state that violates the state invariant.

This is not satisfactory!

The preconditions need to be strengthened so that the post-state violates the invariant \textit{only if} the precondition is \textit{false}.

This illustrates how preconditions and invariants collaborate in achieving safety.

Notice that preconditions ensure safety by imposing a proof obligation to be discharged in any context in which the operation is used.

A possible precondition is

\[(\text{colour} \in \{\text{Green, Amber}\} \Rightarrow \text{lights(OTHER\_DIR(dir))} = \text{Red})\]
We now have states and arguments of the operation \textit{ChangeLight} that lead to a state that violates the state invariant.

This is not satisfactory!

The preconditions need to be strengthened so that the post-state violates the invariant \textit{only if} the precondition is \textit{false}.

This illustrates how preconditions and invariants collaborate in achieving safety.

Notice that preconditions ensure safety by imposing a proof obligation to be discharged in any context in which the operation is used.

A possible precondition is

\[
\text{colour} \in \{\text{Green, Amber}\} \Rightarrow \\
\text{lights(OTHER\_DIR(dir))} = \text{Red}
\]
Strengthening the Precondition

We now have states and arguments of the operation *ChangeLight* that lead to a state that violates the state invariant.

This is not satisfactory!

The preconditions need to be strengthened so that the post-state violates the invariant *only if* the precondition is *false*.

This illustrates how preconditions and invariants collaborate in achieving safety.

Notice that preconditions ensure safety by imposing a proof obligation to be discharged in any context in which the operation is used.

A possible precondition is

\[(\text{colour} \in \{\text{Green, Amber}\} \Rightarrow \text{lights(OTHER\_DIR(dir))} = \text{Red})\]
Strengthening the Precondition

We now have states and arguments of the operation *ChangeLight* that lead to a state that violates the state invariant.

This is not satisfactory!

The preconditions need to be strengthened so that the post-state violates the invariant *only if* the precondition is *false*.

This illustrates how preconditions and invariants collaborate in achieving safety.

Notice that preconditions ensure safety by imposing a proof obligation to be discharged in any context in which the operation is used.

A possible precondition is

\[(colour \in \{\text{Green, Amber}\} \Rightarrow \text{lights(OTHER_DIR(dir))} = \text{Red})\]
We now have states and arguments of the operation \textit{ChangeLight} that lead to a state that violates the state invariant.

This is not satisfactory!

The preconditions need to be strengthened so that the post-state violates the invariant \textit{only if} the precondition is \textit{false}.

This illustrates how preconditions and invariants collaborate in achieving safety.

Notice that preconditions ensure safety by imposing a proof obligation to be discharged in any context in which the operation is used.

A possible precondition is

\[
\text{(colour } \in \{\text{Green, Amber}\} \implies \text{lights(OTHER\_DIR(dir)) } = \text{Red})
\]
Strengthening the Precondition

We now have states and arguments of the operation *ChangeLight* that lead to a state that violates the state invariant.

This is not satisfactory!

The preconditions need to be strengthened so that the post-state violates the invariant *only if* the precondition is *false*.

This illustrates how preconditions and invariants collaborate in achieving safety.

Notice that preconditions ensure safety by imposing a proof obligation to be discharged in any context in which the operation is used.

A possible precondition is

\[(\text{colour} \in \{\text{Green, Amber}\} \Rightarrow \text{lights}(\text{OTHER}_\text{DIR}(\text{dir})) = \text{Red})\]
Strengthening the Precondition

We now have states and arguments of the operation \textit{ChangeLight} that lead to a state that violates the state invariant.

This is not satisfactory!

The preconditions need to be strengthened so that the post-state violates the invariant \textit{only if} the precondition is \textit{false}.

This illustrates how preconditions and invariants collaborate in achieving safety.

Notice that preconditions ensure safety by imposing a proof obligation to be discharged in any context in which the operation is used.

A possible precondition is

\[
(colour \in \{\text{Green, Amber}\} \Rightarrow \text{lights(OTHER\_DIR(dir))} = \text{Red})
\]
We now have states and arguments of the operation *ChangeLight* that lead to a state that violates the state invariant.

This is not satisfactory!

The preconditions need to be strengthened so that the post-state violates the invariant *only if* the precondition is *false*.

This illustrates how preconditions and invariants collaborate in achieving safety.

Notice that preconditions ensure safety by imposing a proof obligation to be discharged in any context in which the operation is used.

A possible precondition is

\[
\begin{align*}
\text{colour} & \in \{\text{Green, Amber}\} \Rightarrow \\
\text{lights}(\text{OTHER}\_\text{DIR}(\text{dir})) & = \text{Red}
\end{align*}
\]
In this model of SimpleTwoWay we put the following alternative formulations of the invariant into the assertions:

- $\forall dir. (dir \in DIRECTION \land lights(dir) \in \{\text{Green}, \text{Amber}\} \Rightarrow lights(\text{OTHER\_DIR}(dir)) = \text{Red})$

- $lights(\text{EastWest}) \in \{\text{Green}, \text{Amber}\} \Rightarrow lights(\text{NorthSouth}) = \text{Red}$

Assertions 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.
The SimpleTwoWay machine II

MACHINE  SimpleTwoWay
SEES  TwoWay_Ctx
VARIABLES  lights
INVARIANT
\[ \text{lights} \in \text{DIRECTION} \rightarrow \text{LIGHT} \land \]
\[ ( \text{lights} ( \text{NorthSouth} ) \in \{ \text{Green} , \text{Amber} \} \Rightarrow \text{lights} ( \text{EastWest} ) = \text{Red} ) \]
ASSERTIONS
\[ \forall \ \text{dir} . ( \text{dir} \in \text{DIRECTION} \land \text{lights} ( \text{dir} ) \in \{ \text{Green} , \text{Amber} \} \Rightarrow \]
\[ \text{lights} ( \text{OTHER(DIR(\text{dir}))} = \text{Red} ) \land \]
\[ ( \text{lights} ( \text{EastWest} ) \in \{ \text{Green} , \text{Amber} \} \Rightarrow \text{lights} ( \text{NorthSouth} ) = \text{Red} ) \]
THE SIMPLETWOWAY MACHINE III

INITIALISATION

\[ \text{lights} := \{ \text{NorthSouth} \mapsto \text{Red}, \text{EastWest} \mapsto \text{Red} \} \]

OPERATIONS

\[ \text{ChangeLight} ( \text{dir}, \text{colour} ) \triangleq \]

\[ \begin{array}{l}
\text{pre} \quad \text{dir} \in \text{DIRECTION} \land \text{colour} \in \text{LIGHT} \land \\
\quad ( \text{colour} \in \{ \text{Green}, \text{Amber} \} \Rightarrow \text{lights} ( \text{OTHER_DIR} ( \text{dir} ) ) = \text{Red} ) \\
\text{then} \quad \text{lights} ( \text{dir} ) := \text{colour} \\
\text{end} \\
\end{array} \]

END
Sequencing

The SimpleTwoWay machine ensures safety, but does not enforce any sequencing.

To achieve the desired sequencing we will introduce a new machine: TwoWay, that INCLUDES SimpleTwoWay and has three operations ToRed(dir), ToGreen(dir), and ToAmber(dir) for changing the lights.

The notion is that the body of each operation will use the operation ChangeLight(dir, colour) to change the colour.

The precondition of each operation will constrain the sequencing, and also must ensure that the precondition of ChangeLight is satisfied.
Sequencing

The **SimpleTwoWay** machine ensures safety, but does not enforce any sequencing.

To achieve the desired sequencing we will introduce a new machine: **TwoWay**, that *INCLUDES* SimpleTwoWay and has three operations: ToRed(dir), ToGreen(dir), and ToAmber(dir) for changing the lights. The notion is that the body of each operation will use the operation ChangeLight(dir, colour) to change the colour.

The precondition of each operation will constrain the sequencing, and also must ensure that the precondition of ChangeLight is satisfied.
The **SimpleTwoWay** machine ensures safety, but does not enforce any sequencing.

To achieve the desired sequencing we will introduce a new machine: **TwoWay**, that *INCLUDES SimpleTwoWay* and has three operations `ToFRed(dir)`, `ToFGreen(dir)`, and `ToFAmber(dir)` for changing the lights.

The notion is that the body of each operation will use the operation `ChangeLight(dir, colour)` to change the colour.

The precondition of each operation will constrain the sequencing, and also must ensure that the precondition of `ChangeLight` is satisfied.
Sequencing

The SimpleTwoWay machine ensures safety, but does not enforce any sequencing.

To achieve the desired sequencing we will introduce a new machine: TwoWay, that INCLUDES SimpleTwoWay and has three operations ToRed(dir), ToGreen(dir), and ToAmber(dir) for changing the lights.

The notion is that the body of each operation will use the operation ChangeLight(dir, colour) to change the colour.

The precondition of each operation will constrain the sequencing, and also must ensure that the precondition of ChangeLight is satisfied.
The SimpleTwoWay machine ensures safety, but does not enforce any sequencing.

To achieve the desired sequencing we will introduce a new machine: TwoWay, that INCLUDES SimpleTwoWay and has three operations ToRed(dir), ToGreen(dir), and ToAmber(dir) for changing the lights. The notion is that the body of each operation will use the operation ChangeLight(dir, colour) to change the colour.

The precondition of each operation will constrain the sequencing, and also must ensure that the precondition of ChangeLight is satisfied.
Since the precondition of $\text{ChangeLight}$ is always satisfied for the colour $\text{Red}$, we need to only be concerned with sequencing. Hence the precondition is

$$\text{lights}(\text{dir}) = \text{Amber}$$
Since the precondition of \textit{ChangeLight} is always satisfied for the colour \textit{Red}, we need to only be concerned with sequencing. Hence the precondition is

\[ \text{lights}(\text{dir}) = \text{Amber} \]
Since the precondition of \textit{ChangeLight} is always satisfied for the colour \textcolor{red}{Red}, we need to only be concerned with sequencing. Hence the precondition is

\[ \text{lights}(\text{dir}) = \text{Amber} \]
Since the precondition of \textit{ChangeLight} is always satisfied for the colour \texttt{Red}, we need to only be concerned with sequencing. Hence the precondition is

\[
lights(dir) = \texttt{Amber}
\]
The precondition of ToGreen

When setting a light to $\text{Green}$ the precondition of ChangeLight requires

$$\text{lights}(\text{OTHER\_DIR}(\text{dir})) = \text{Red}$$

Sequencing requires

$$\text{lights}(\text{dir}) = \text{Red}$$

Thus the precondition is the conjunction

$$\text{lights}(\text{dir}) = \text{Red} \land \text{lights}(\text{OTHER\_DIR}(\text{dir})) = \text{Red}$$
The precondition of ToGreen

When setting a light to \textit{Green} the precondition of ChangeLight requires

\[ \text{lights}(\text{OTHER\_DIR}(\text{dir})) = \text{Red} \]

Sequencing requires

\[ \text{lights}(\text{dir}) = \text{Red} \]

Thus the precondition is the conjunction

\[ \text{lights}(\text{dir}) = \text{Red} \wedge \text{lights}(\text{OTHER\_DIR}(\text{dir})) = \text{Red} \]
The precondition of ToGreen

When setting a light to \textit{Green} the precondition of ChangeLight requires

\[ \text{lights}(\text{OTHER}\_\text{DIR}(\text{dir})) = \text{Red} \]

Sequencing requires

\[ \text{lights}(\text{dir}) = \text{Red} \]

Thus the precondition is the conjunction

\[ \text{lights}(\text{dir}) = \text{Red} \land \text{lights}(\text{OTHER}\_\text{DIR}(\text{dir})) = \text{Red} \]
The precondition of ToGreen

When setting a light to *Green* the precondition of ChangeLight requires

\[ \text{lights(OTHER_DIR(dir))} = \text{Red} \]

Sequencing requires

\[ \text{lights(dir)} = \text{Red} \]

Thus the precondition is the conjunction

\[ \text{lights(dir)} = \text{Red} \land \text{lights(OTHER_DIR(dir))} = \text{Red} \]
When setting a light to \textit{Green} the precondition of \texttt{ChangeLight} requires

\[
lights(OTHER\_DIR(dir)) = \text{Red}
\]

Sequencing requires

\[
lights(dir) = \text{Red}
\]

Thus the precondition is the conjunction

\[
lights(dir) = \text{Red} \land lights(OTHER\_DIR(dir)) = \text{Red}
\]
The precondition of ToGreen

When setting a light to \textit{Green} the precondition of ChangeLight requires

\[ \text{lights(OTHER\_DIR(dir))} = \text{Red} \]

Sequencing requires

\[ \text{lights(dir)} = \text{Red} \]

Thus the precondition is the conjunction

\[ \text{lights(dir)} = \text{Red} \land \text{lights(OTHER\_DIR(dir))} = \text{Red} \]
The precondition of ToGreen

When setting a light to \textit{Green} the precondition of ChangeLight requires

\[ \text{lights}(\text{OTHER}_{-}\text{DIR}(dir)) = \text{Red} \]

Sequencing requires

\[ \text{lights}(dir) = \text{Red} \]

Thus the precondition is the conjunction

\[ \text{lights}(dir) = \text{Red} \land \text{lights}(\text{OTHER}_{-}\text{DIR}(dir)) = \text{Red} \]
Equivalent predicates

This may be expressed by a number of equivalent predicates, when combined with the sequencing predicate \( \text{lights}(\text{dir}) = \text{Red} \):

1. \( \text{dir} = \text{NorthSouth} \Rightarrow \text{lights}(\text{EastWest}) = \text{Red} \)
   \( \text{dir} = \text{EastWest} \Rightarrow \text{lights}(\text{NorthSouth}) = \text{Red} \)
2. \( \forall \text{dir} \in \text{DIRECTION} \Rightarrow \text{lights}(\text{dir}) = \text{Red} \)
3. \( \text{lights}[\text{DIRECTION}] = \{ \text{Red} \} \)
4. \( \text{ran}(\text{lights}) = \{ \text{Red} \} \)

It is worth noting that alternatives 2, 3 and 4 if generalized to more directions, are too strong.

\[ \text{This uses relational image } r[s] = \text{ran}(s < r) \]
This may be expressed by a number of equivalent predicates, when combined with the sequencing predicate \( \text{lights}(\text{dir}) = \text{Red} \):

1. \( \text{dir} = \text{NorthSouth} \Rightarrow \text{lights}(\text{EastWest}) = \text{Red} \) \land \( \text{dir} = \text{EastWest} \Rightarrow \text{lights}(\text{NorthSouth}) = \text{Red} \)

2. \( \forall \text{dir}. (\text{dir} \in \text{DIRECTION} \Rightarrow \text{lights}(\text{dir}) = \text{Red}) \)

3. \( \text{lights}([\text{DIRECTION}]) = \{\text{Red}\} \)

4. \( \text{ran} (\text{lights}) = \{\text{Red}\} \)

It is worth noting that alternatives 2, 3 and 4 if generalized to more directions, are too strong.

\[ \text{This uses relational image } r|s| = \text{ran}(s < r) \]
Equivalent predicates

This may be expressed by a number of equivalent predicates, when combined with the sequencing predicate $\text{lights}(\text{dir}) = \text{Red}$:

1. $\text{dir} = \text{NorthSouth} \Rightarrow \text{lights}(\text{EastWest}) = \text{Red} \land \text{dir} = \text{EastWest} \Rightarrow \text{lights}(\text{NorthSouth}) = \text{Red}$
2. $\forall \text{dir}. (\text{dir} \in \text{DIRECTION} \Rightarrow \text{lights}(\text{dir}) = \text{Red})$
3. $\text{lights}[\text{DIRECTION}] = \{\text{Red}\}$
4. $\text{ran}(\text{lights}) = \{\text{Red}\}$

It is worth noting that alternatives 2, 3 and 4 if generalized to more directions, are too strong.

\[^{1}\text{This uses relational image } r[s] = \text{ran}(s \triangleleft r)\]
Equivalent predicates

This may be expressed by a number of equivalent predicates, when combined with the sequencing predicate \( \text{lights}(\text{dir}) = \text{Red} \):

1. \( \text{dir} = \text{NorthSouth} \Rightarrow \text{lights}(\text{EastWest}) = \text{Red} \) \( \land \)
   \( \text{dir} = \text{EastWest} \Rightarrow \text{lights}(\text{NorthSouth}) = \text{Red} \)

2. \( \forall \text{dir} . (\text{dir} \in \text{DIRECTION} \Rightarrow \text{lights} (\text{dir}) = \text{Red}) \)

3. \( \text{lights}[\text{DIRECTION}] = \{ \text{Red} \} \)

4. \( \text{ran}(\text{lights}) = \{ \text{Red} \} \)

It is worth noting that alternatives 2, 3 and 4 if generalized to more directions, are too strong.

\(^1\text{This uses relational image } r[s] = \text{ran}(s \circ r)\)
Equivalent predicates

This may be expressed by a number of equivalent predicates, when combined with the sequencing predicate $\text{lights}(\text{dir}) = \text{Red}$:

1. \( \text{dir} = \text{NorthSouth} \Rightarrow \text{lights}(\text{EastWest}) = \text{Red} \quad \wedge \quad \text{dir} = \text{EastWest} \Rightarrow \text{lights}(\text{NorthSouth}) = \text{Red} \)
2. \( \forall \text{dir} \cdot (\text{dir} \in \text{DIRECTION} \Rightarrow \text{lights}(\text{dir}) = \text{Red}) \)
3. \( \text{lights}[\text{DIRECTION}] = \{\text{Red}\} \)
4. \( \text{ran}(\text{lights}) = \{\text{Red}\} \)

It is worth noting that alternatives 2, 3 and 4 if generalized to more directions, are too strong.

\^This uses relational image \( r[s] = \text{ran}(s \triangleleft r) \)
Equivalent predicates

This may be expressed by a number of equivalent predicates, when combined with the sequencing predicate $\text{lights}(\text{dir}) = \text{Red}$:

1. $\text{dir} = \text{NorthSouth} \Rightarrow \text{lights}(\text{EastWest}) = \text{Red} \land \text{dir} = \text{EastWest} \Rightarrow \text{lights}(\text{NorthSouth}) = \text{Red}$

2. $\forall \text{dir}. (\text{dir} \in \text{DIRECTION} \Rightarrow \text{lights}(\text{dir}) = \text{Red})$

3. $\text{lights}[^\text{DIRECTION}] = \{\text{Red}\}$

4. $\text{ran}(%math\text{lights}) = \{\text{Red}\}$

It is worth noting that alternatives 2, 3 and 4 if generalized to more directions, are too strong.

---

$^1$This uses relational image $r[s] = \text{ran}(s < r)$
Equivalent predicates

This may be expressed by a number of equivalent predicates, when combined with the sequencing predicate \( \text{lights}(dir) = \text{Red} \):

1. \( dir = \text{NorthSouth} \Rightarrow \text{lights}(\text{EastWest}) = \text{Red} \) \( \land \) \( dir = \text{EastWest} \Rightarrow \text{lights}(\text{NorthSouth}) = \text{Red} \)
2. \( \forall dir. (dir \in \text{DIRECTION} \Rightarrow \text{lights}(dir) = \text{Red}) \)
3. \( \text{lights}[\text{DIRECTION}] = \{\text{Red}\} \)
4. \( \text{ran}(\text{lights}) = \{\text{Red}\} \)

It is worth noting that alternatives 2, 3 and 4 if generalized to more directions, are too strong.

---

\(^1\)This uses relational image \( r[s] = \text{ran}(s \triangleleft r) \)
Equivalent predicates

This may be expressed by a number of equivalent predicates, when combined with the sequencing predicate $\text{lights}(dir) = \text{Red}$:

1. $\text{dir} = \text{NorthSouth} \Rightarrow \text{lights}(\text{EastWest}) = \text{Red}$ and $\text{dir} = \text{EastWest} \Rightarrow \text{lights}(\text{NorthSouth}) = \text{Red}$

2. $\forall \text{dir}.(\text{dir} \in \text{DIRECTION} \Rightarrow \text{lights}(\text{dir}) = \text{Red})$

3. $\text{lights}[\text{DIRECTION}] = \{\text{Red}\}$ \(^1\)

4. $\text{ran} (\text{lights}) = \{\text{Red}\}$

It is worth noting that alternatives 2, 3 and 4 if generalized to more directions, are too strong.

---

\(^1\) This uses \textit{relational image} $r[s] = \text{ran}(s \triangleleft r)$
In order to satisfy the precondition of $ChangeLight$, the light in the other direction must be showing $Red$.

But, since the sequencing condition is $\text{lights(dir)} = Green$, the state invariant implies that

$$\text{lights(OTHER\_DIR(dir))} = Red.$$ 

So, only the sequencing condition is required

$$\text{lights(dir)} = Green.$$
The precondition of ToAmber

In order to satisfy the precondition of *ChangeLight*, the light in the other direction must be showing *Red*.

But, since the sequencing condition is

\[ \text{lights}(\text{dir}) = \text{Green}, \]

the state invariant implies that

\[ \text{lights}(\text{OTHER}_\text{DIR}(\text{dir})) = \text{Red}. \]

So, only the sequencing condition is required

\[ \text{lights}(\text{dir}) = \text{Green}. \]
In order to satisfy the precondition of \textit{ChangeLight}, the light in the other direction must be showing \textbf{Red}.

But, since the sequencing condition is

\[
\text{\textit{lights}}(\text{dir}) = \text{\textit{Green}},
\]

the state invariant implies that

\[
\text{\textit{lights}}(\text{OTHER\_DIR(dir)}) = \text{\textbf{Red}}.
\]

So, only the sequencing condition is required

\[
\text{\textit{lights}}(\text{dir}) = \text{\textit{Green}}.
\]
In order to satisfy the precondition of *ChangeLight*, the light in the other direction must be showing **Red**.

But, since the sequencing condition is

\[
lights(dir) = \text{Green},
\]

the state invariant implies that

\[
lights(\text{OTHER_DIR}(dir)) = \text{Red}.
\]

So, only the sequencing condition is required

\[
lights(dir) = \text{Green}
\]
The precondition of ToAmber

In order to satisfy the precondition of \textit{ChangeLight}, the light in the other direction must be showing \textbf{Red}.

But, since the sequencing condition is

\[ \text{lights(dir)} = \text{Green}, \]

the state invariant implies that

\[ \text{lights(OTHER\_DIR(dir)}) = \text{Red}. \]

So, only the sequencing condition is required

\[ \text{lights(dir)} = \text{Green} \]
The precondition of ToAmber

In order to satisfy the precondition of \textit{ChangeLight}, the light in the other direction must be showing \textcolor{red}{Red}. But, since the sequencing condition is

\[
lights(dir) = \textcolor{green}{Green},
\]

the state invariant implies that

\[
lights(OTHER\_DIR(dir)) = \textcolor{red}{Red}.
\]

So, only the sequencing condition is required

\[
lights(dir) = \textcolor{green}{Green}
\]
The precondition of ToAmber

In order to satisfy the precondition of \textit{ChangeLight}, the light in the other direction must be showing \textcolor{red}{Red}.

But, since the sequencing condition is
\[ \text{\textit{lights}}(\text{dir}) = \text{\textit{Green}}, \]
the state invariant implies that
\[ \text{\textit{lights}}(\text{\textit{OTHER}\_\textit{DIR}}(\text{dir})) = \text{\textit{Red}}. \]

So, only the sequencing condition is required
\[ \text{\textit{lights}}(\text{dir}) = \text{\textit{Green}} \]
The TwoWay Machine I

MACHINE TwoWay
SEES TwoWay_Ctx
INCLUDES SimpleTwoWay

OPERATIONS
The TwoWay Machine II

ToRed (dir) ≜
  pre  dir ∈ DIRECTION ∧ lights (dir) = Amber
  then  ChangeLight (dir, Red)
  end ;

ToGreen (dir) ≜
  pre  dir ∈ DIRECTION ∧ lights (dir) = Red ∧
       lights (OTHER_DIR (dir)) = Red
  then  ChangeLight (dir, Green)
  end ;

ToAmber (dir) ≜
  pre  dir ∈ DIRECTION ∧ lights (dir) = Green
  then  ChangeLight (dir, Amber)
  end

END