Controller for cars on a bridge

Based on original slides by J. R. Abrial
(http://wiki.event-b.org/index.php/Event-B_Language)
which present the second chapter of the Event-B book
(available at http://www.event-b.org/abook.html).

Please refer to the chapter & slides
to fully understand this abridged slide set

Purpose of this Lecture (1)

- To present an example of system development

- Our approach: a series of more and more accurate models

- This approach is called refinement

- The models formalize the view of an external observer

- With each refinement observer “zooms in” to see more details

Purpose of this Lecture (2)

- Each model will be analyzed and proved to be correct

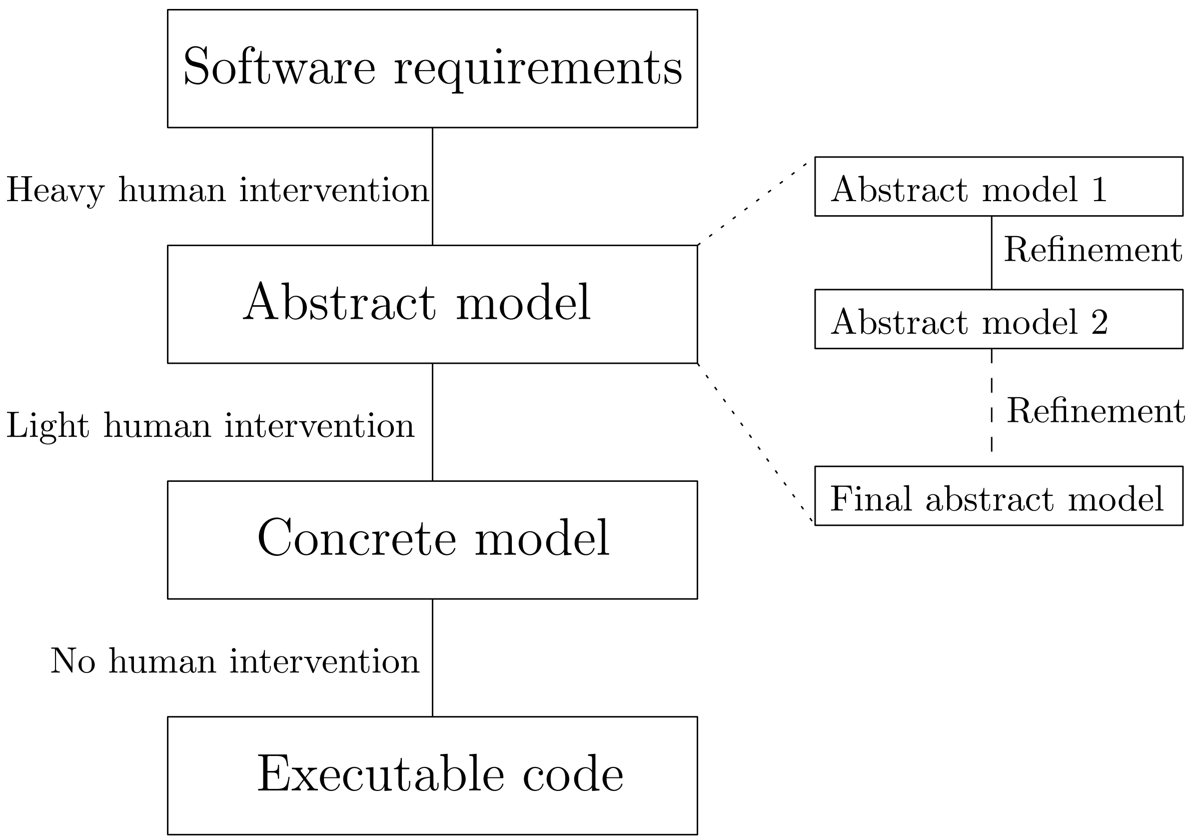
- The aim is to obtain a system that will be correct by construction

- The correctness criteria are formulated as proof obligations

- Proofs will be performed by using the sequent calculus

- Inference rules used in the sequent calculus will be reviewed

Reminder



A Requirements Document (1)

- The system we are going to build is a piece of software connected to some equipment.

- There are two kinds of requirements:
  - those concerned with the equipment, labeled EQP,
  - those concerned with the function of the system, labeled FUN.

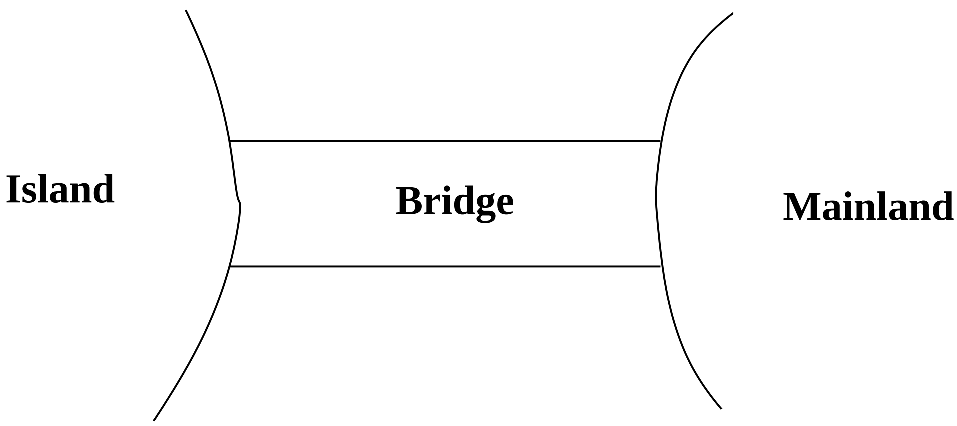
- The function of this system is to control cars on a narrow bridge.

- This bridge is supposed to link the mainland to a small island.

A Requirements Document (2)

The system is controlling cars on a bridge between the mainland and an island	FUN-1

- This can be illustrated as follows



A Requirements Document (3)

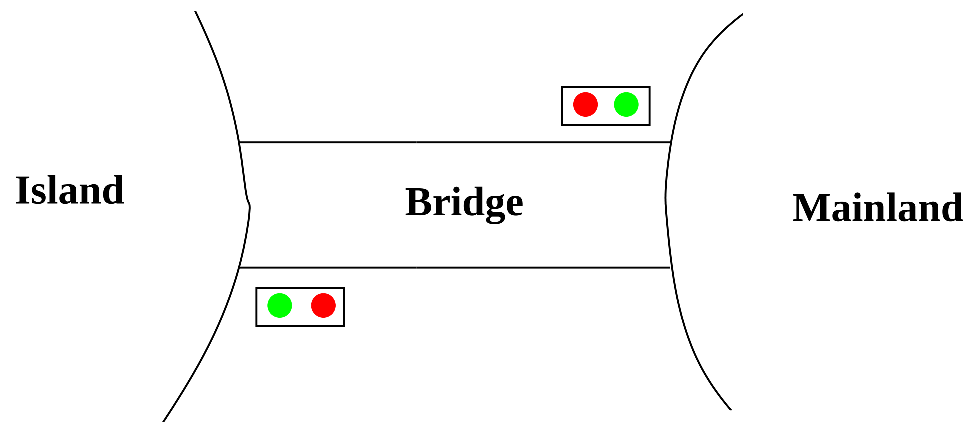
- The controller is equipped with two traffic lights with two colors.

The system has two traffic lights with two colors: green and red	EQP-1

A Requirements Document (4)

- One of the traffic lights is situated on the mainland and the other one on the island. Both are close to the bridge.

- This can be illustrated as follows



A Requirements Document (5)

The traffic lights control the entrance to the bridge at both ends of it	EQP-2

- Drivers are supposed to obey the traffic light by not passing when a traffic light is red.

Cars are not supposed to pass on a red traffic light, only on a green one	EQP-3

A Requirements Document (6)

- There are also some car sensors situated at both ends of the bridge.

- These sensors are supposed to detect the presence of cars intending to enter or leave the bridge.

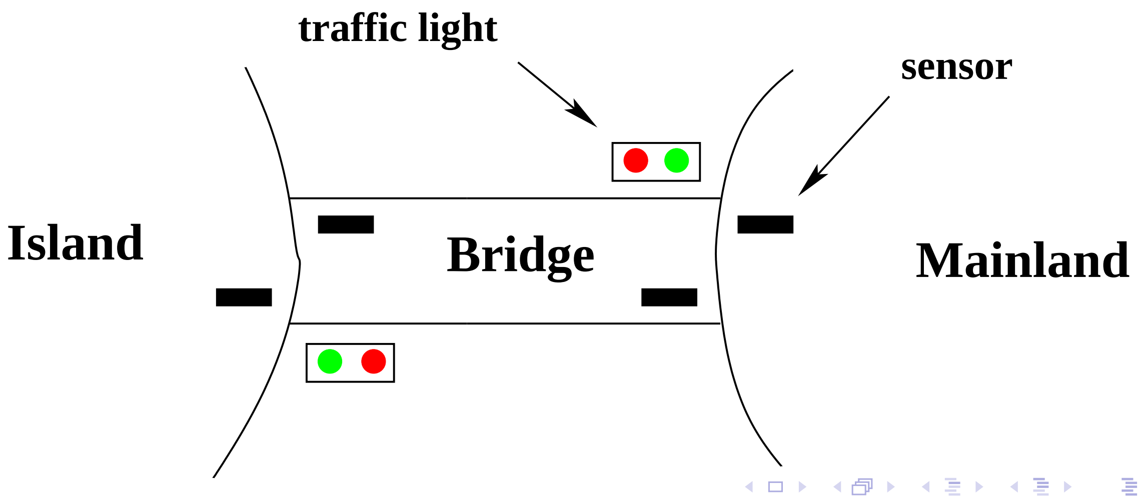
- There are four such sensors. Two of them are situated on the bridge and the other two are situated on the mainland and on the island.

The system is equipped with four car sensors each with two states: on or off	EQP-4

A Requirements Document (7)

The sensors are used to detect the presence of cars entering or leaving the bridge	EQP-5

- The pieces of equipment can be illustrated as follows:



A Requirements Document (8)

- This system has two main constraints: the number of cars on the bridge and the island is limited and the bridge is one way.

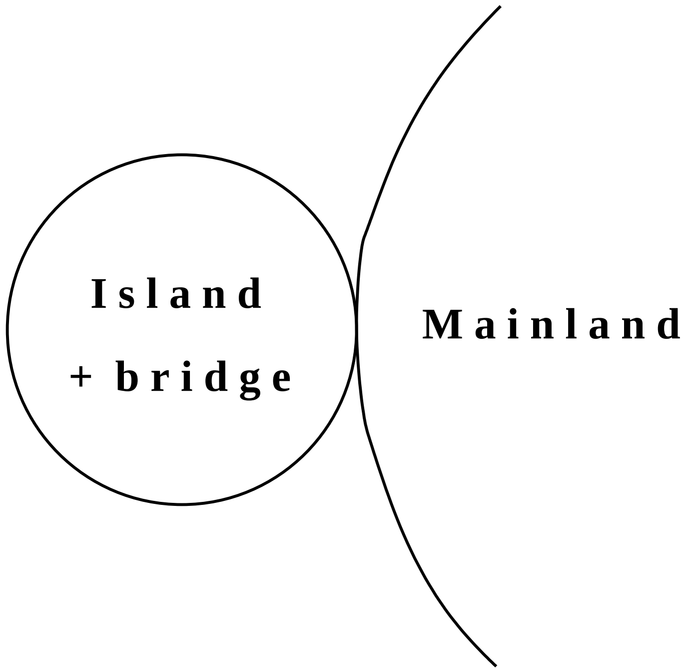
The number of cars on the bridge and the island is limited	FUN-2

The bridge is one way or the other, not both at the same time	FUN-3

Our Refinement Strategy

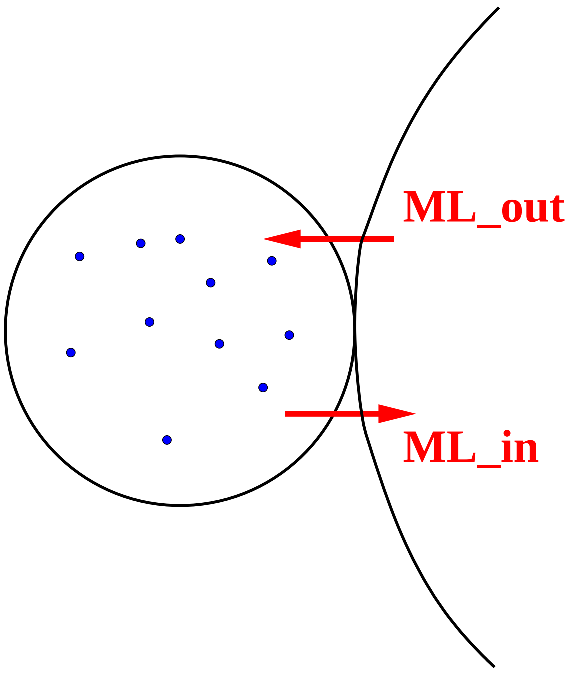
- Initial model: Limiting the number of cars (FUN-2)
- First refinement: Introducing the one-way bridge (FUN-3)
- Second refinement: Introducing the traffic lights (EQP-1,2,3)
- Third refinement: Introducing the sensors (EQP-4,5)

A Situation as Seen from the Sky



What would be a suitable events and state in this system?

Two Events that may be Observed



Formalizing the State: Constants and Axioms

- STATIC PART of the state: constant d with axiom **axm0_1**

constant: d	**axm0_1:** $d \in \mathbb{N}$

- d is the maximum number of cars allowed on the Island-Bridge

- **axm0_1** states that d is a natural number

- Constant d is a member of the set $\mathbb{N} = \{0, 1, 2, , \ldots\}$

Formalizing the State: variable

- DYNAMIC PART: variable v with invariants **inv0_1** and **inv0_2**

variable: n	**inv0_1:** $n \in \mathbb{N}$ **inv0_2:** $n \leq d$

- n is the effective number of cars on the Island-Bridge

- n is a natural number (**inv0_1**)

- n is always smaller than or equal to d (**inv0_2**): this is FUN_2

Formalizing the two Events: an Approximation

- Event ML_out increments the number of cars

ML_out
$n := n + 1$

- Event ML_in decrements the number of cars

ML_in
$n := n - 1$

- An event is denoted by its name and its action (an assignment)

Why an Approximation?

These events are approximations for two reasons:

1. They might be refined (made more precise) later

2. They might be insufficient at this stage because not consistent with the invariant

We have to perform a proof in order to verify this consistency.

Summary of Proof Obligations

ML_out / **inv0_1** / INV

$$\begin{array}{l} d \in \mathbb{N} \\ n \in \mathbb{N} \\ n \leq d \\ \vdash \\ n + 1 \in \mathbb{N} \end{array}$$

ML_out / **inv0_2** / INV

$$\begin{array}{l} d \in \mathbb{N} \\ n \in \mathbb{N} \\ n \leq d \\ \vdash \\ n + 1 \leq d \end{array}$$

ML_in / **inv0_1** / INV

$$\begin{array}{l} d \in \mathbb{N} \\ n \in \mathbb{N} \\ n \leq d \\ \vdash \\ n - 1 \in \mathbb{N} \end{array}$$

ML_in / **inv0_2** / INV

$$\begin{array}{l} d \in \mathbb{N} \\ n \in \mathbb{N} \\ n \leq d \\ \vdash \\ n - 1 \leq d \end{array}$$

A Failed Proof Attempt: ML_out / inv0_2 / INV

$$\begin{array}{l} d \in \mathbb{N} \\ n \in \mathbb{N} \\ n \leq d \\ \vdash \\ n + 1 \leq d \end{array} \quad \text{MON} \quad \begin{array}{l} n \leq d \\ \vdash \\ n + 1 \leq d \end{array} \quad ?$$

- We put a **?** to indicate that we have no rule to apply

- The proof fails: we cannot conclude with rule **INC** ($n < d$ needed)

$$\frac{}{\mathbf{n} < \mathbf{m} \vdash \mathbf{n} + 1 \leq \mathbf{m}} \quad \textbf{INC}$$

A Failed Proof Attempt: ML_in / inv0_1 / INV

$d \in \mathbb{N}$ $n \in \mathbb{N}$ $n \leq d$ $\vdash$ $n - 1 \in \mathbb{N}$	MON	$n \in \mathbb{N}$ $\vdash$ $n - 1 \in \mathbb{N}$	?

- The proof fails: we cannot conclude with rule **P2′** ($0 < n$ needed)

$$\frac{}{0 < \mathbf{n} \ \vdash \ \mathbf{n} - 1 \in \mathbb{N}} \quad \mathbf{P2'}$$

Reasons for Proof Failure

- We needed hypothesis $n < d$ to prove ML_out / **inv0_2** / INV

- We needed hypothesis $0 < n$ to prove ML_in / **inv0_1** / INV

ML_out $n := n + 1$	**ML_in** $n := n - 1$

- We are going to add $n < d$ as a guard to event ML_out

- We are going to add $0 < n$ as a guard to event ML_in

Improving the Events: Introducing Guards

ML_out **when** $n < d$ **then** $n := n + 1$ **end**	ML_in **when** $0 < n$ **then** $n := n - 1$ **end**

- We are adding guards to the events

- The guard is the necessary condition for an event to “occur”

A Formal Proof of: ML_out / inv0_2 / INV

$$\begin{array}{l} d \in \mathbb{N} \\ n \in \mathbb{N} \\ n \leq d \\ n < d \\ \vdash \\ n + 1 \leq d \end{array} \quad \text{MON} \quad \begin{array}{l} n < d \\ \vdash \\ n + 1 \leq d \end{array} \quad \textbf{INC}$$

- Now we can conclude the proof using rule **INC**

$$\frac{}{\mathbf{n} < \mathbf{m} \ \vdash \ \mathbf{n} + 1 \leq \mathbf{m}} \ \textbf{INC}$$

A Formal Proof of: ML_in / inv0_1 / INV

$$\begin{array}{l} d \in \mathbb{N} \\ n \in \mathbb{N} \\ n \leq d \\ 0 < n \\ \vdash \\ n - 1 \in \mathbb{N} \end{array} \quad \text{MON} \quad \begin{array}{l} 0 < n \\ \vdash \\ n - 1 \in \mathbb{N} \end{array} \quad \text{P2'}$$

- Now we can conclude the proof using rule **P2′**

$$\frac{}{0 < \mathbf{n} \vdash \mathbf{n} - 1 \in \mathbb{N}} \quad \mathbf{P2'}$$

A Missing Requirement

- It is possible for the system to be blocked if both guards are false

- We do not want this to happen

- We figure out that one important requirement was missing

Once started, the system should work for ever	FUN-4

Proof Obligation: Deadlock Freedom

- Given c with axioms $A(c)$ and v with invariants $I(c, v)$

- Given the guards $G_1(c, v), \ldots, G_m(c, v)$ of the events

- We have to prove the following:

$A(c)$ $I(c, v)$ $\vdash$ $G_1(c, v) \ \lor \ \ldots \ \lor \ G_m(c, v)$	DLF

Applying the Deadlock Freedom PO

axm0_1 **inv0_1** **inv0_2** $\vdash$ Disjunction of guards	$d \in \mathbb{N}$ $n \in \mathbb{N}$ $n \leq d$ $\vdash$ $n < d \ \lor \ 0 < n$

- This cannot be proved with the inference rules we have so far

- $n \leq d$ can be replaced by $n = d \ \lor \ n < d$

- We continue our proof by a case analysis:
  - case 1: $n = d$
  - case 2: $n < d$

Proof of Deadlock Freedom

$$
\dfrac{
  \dfrac{
    \dfrac{}{n < d \vdash n < d}\;\text{HYP}
  }{n < d \vdash n < d \lor 0 < n}\;\text{OR Right}
  \qquad
  \dfrac{
    \dfrac{
      \dfrac{?}{\vdash 0 < d}\;?
    }{\vdash d < d \lor 0 < d}
  }{n = d \vdash n < d \lor 0 < n}\;\text{EQ\_LR}
}{
  \dfrac{
    \dfrac{n < d \lor n = d \vdash n < d \lor 0 < n}{n \le d \vdash n < d \lor 0 < n}\;\text{Peano arith}
  }{d \in \mathbb{N}, n \in \mathbb{N}, n \le d \vdash n < d \lor 0 < n}\;\text{MON}
}\;\text{OR Left}
$$

Problem: d must be positive.

Adding the Forgotten Axiom

- If d is equal to 0, then no car can ever enter the Island-Bridge

axm0_2: $0 < d$

Initial Model: Conclusion

- Thanks to the proofs, we discovered 3 errors

- They were corrected by:
  - adding guards to both events
  - adding an axiom

- The interaction of modeling and proving is an essential element of Formal Methods with Proofs

Summary of Initial Model

constant: d

variable: n

axm0_1: $d \in \mathbb{N}$

axm0_2: $d > 0$

inv0_1: $n \in \mathbb{N}$

inv0_2: $n \leq d$

init
  $n := 0$

ML_out
  when
    $n < d$
  then
    $n := n + 1$
  end

ML_in
  when
    $0 < n$
  then
    $n := n - 1$
  end

Our Refinement Strategy

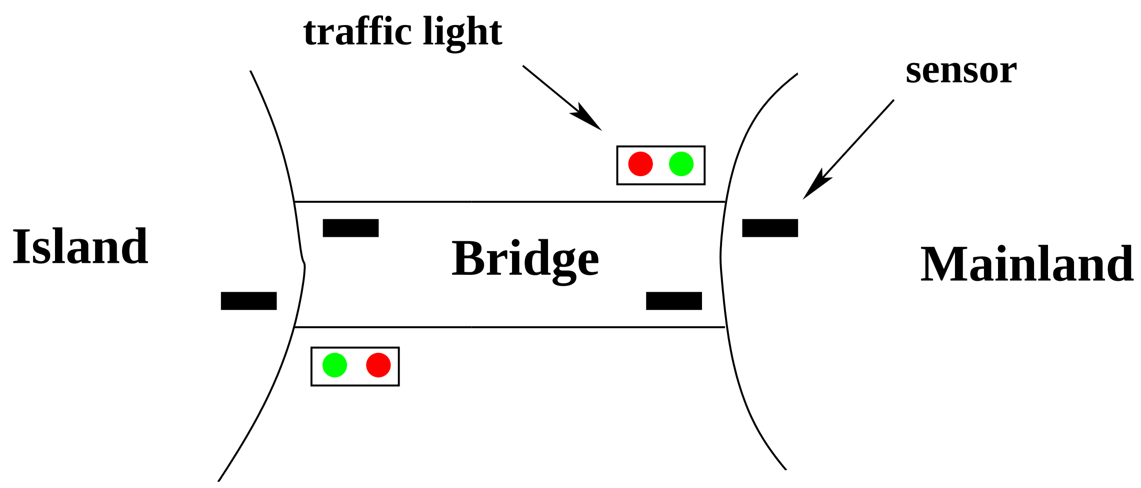
- Initial model: Limiting the number of cars (FUN-2)

- First refinement: Introducing the one way bridge (FUN-3)

- Second refinement: Introducing the traffic lights (EQP-1,2,3)

- Third refinement: Introducing the sensors (EQP-4,5)

Reminder of the physical system



- We go down with our parachute

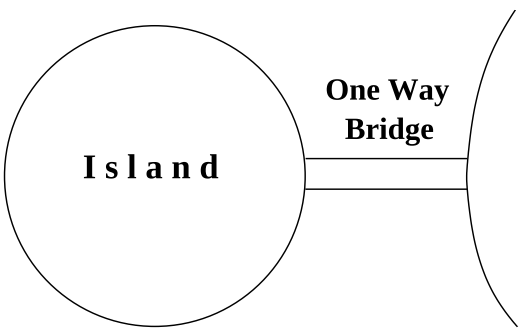
- Our view of the system gets more accurate

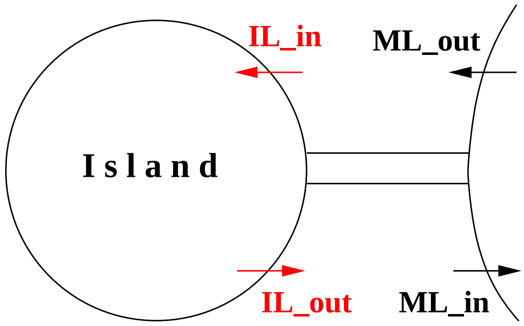
- We introduce the bridge and separate it from the island

- We refine the state and the events

- We also add two new events: IL_in and IL_out

- We are focusing on FUN-3: one-way bridge

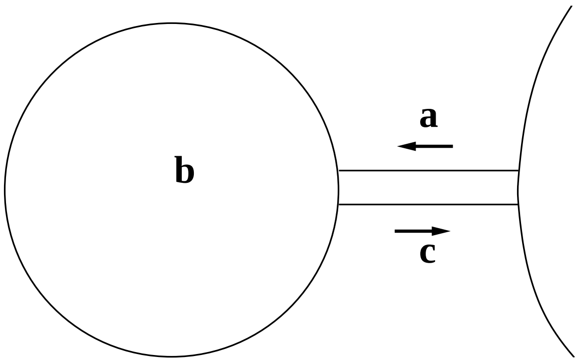




Suggestions?

Suggestions for the new state and invariant?

Introducing Three New Variables: a, b, and c



- a denotes the number of cars on bridge going to island
- b denotes the number of cars on island
- c denotes the number of cars on bridge going to mainland
- a, b, and c are the concrete variables
- They replace the abstract variable n

Refining the State: Formalizing Variables a, b, and c

- Variables a, b, and c denote natural numbers

$$\boxed{\begin{array}{lcl} a & \in & \mathbb{N} \\ b & \in & \mathbb{N} \\ c & \in & \mathbb{N} \end{array}}$$

- Relating the concrete state (a, b, c) to the abstract state (n)

$$\boxed{a + b + c = n}$$

- Formalizing the new invariant: one way bridge (this is FUN-3)

$$\boxed{a = 0 \lor c = 0}$$

- This is the reason of the new variables: express more properties

Refining the State: Summary

constants: d

variables: a, b, c

inv1_1: $a \in \mathbb{N}$

inv1_2: $b \in \mathbb{N}$

inv1_3: $c \in \mathbb{N}$

inv1_4: $a + b + c = n$

inv1_5: $a = 0 \quad \lor \quad c = 0$

- Invariants **inv1_1** to **inv1_5** are called the concrete invariants

- **inv1_4** glues the abstract state, n, to the concrete state, a, b, c

B-A Predicates: Preserved Variables

ML_out
when
$a + b < d$
$c = 0$
then
$a := a + 1$
end

ML_in
when
$0 < c$
then
$c := c - 1$
end

Before-after predicates showing the unmodified variables:

$$a' = a + 1 \quad \wedge \quad b' = b \quad \wedge \quad c' = c$$

$$a' = a \quad \wedge \quad b' = b \quad \wedge \quad c' = c - 1$$

Intuition about Refinement

The concrete model behaves as specified by the abstract model (i.e., concrete model does not exhibit any new behaviors)

To show this we have to prove that

1. every concrete event is simulated by its abstract counterpart (event refinement: following slides)
2. to every concrete initial state corresponds an abstract one (initial state refinement: later)

We will make these two conditions more precise and formalize them as proof obligations.

Intuition about refinement (1)

(abstract_)ML_out
when
$n < d$
then
$n := n + 1$
end

(concrete_)ML_out
when
$a + b < d$
$c = 0$
then
$a := a + 1$
end

- The concrete version is not contradictory with the abstract one

- When the concrete version is enabled then so is the abstract one

- Executions seem to be compatible

Proof Obligations for Refinement

- The concrete guard is stronger than the abstract one

- Each concrete action is compatible with its abstract counterpart

Proving Correct Refinement: the Situation

Constants c with axioms $A(c)$

Abstract variables v with abstract invariant $I(c,v)$

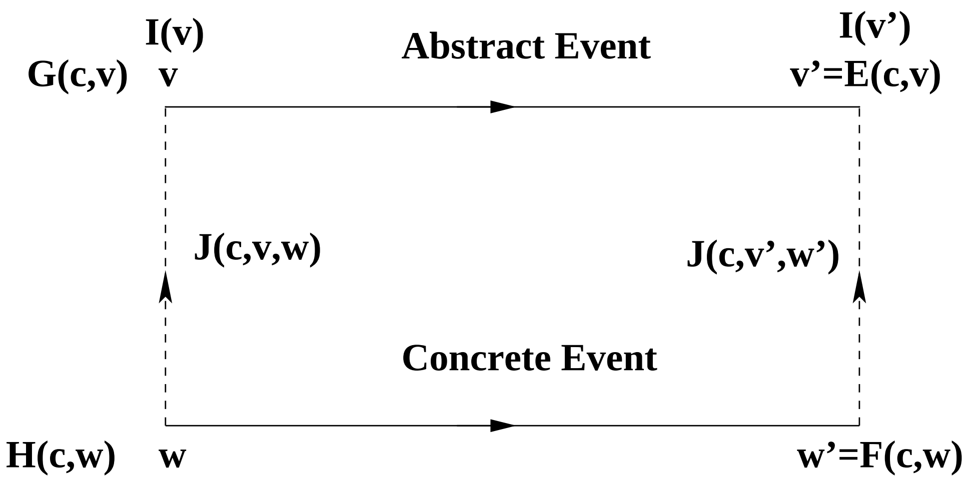
Concrete variables w with concrete invariant $J(c,v,w)$

Abstract event with guards $G(c,v)$: $G_1(c,v), G_2(c,v), \ldots$

Abstract event with before-after predicate $v' = E(c,v)$

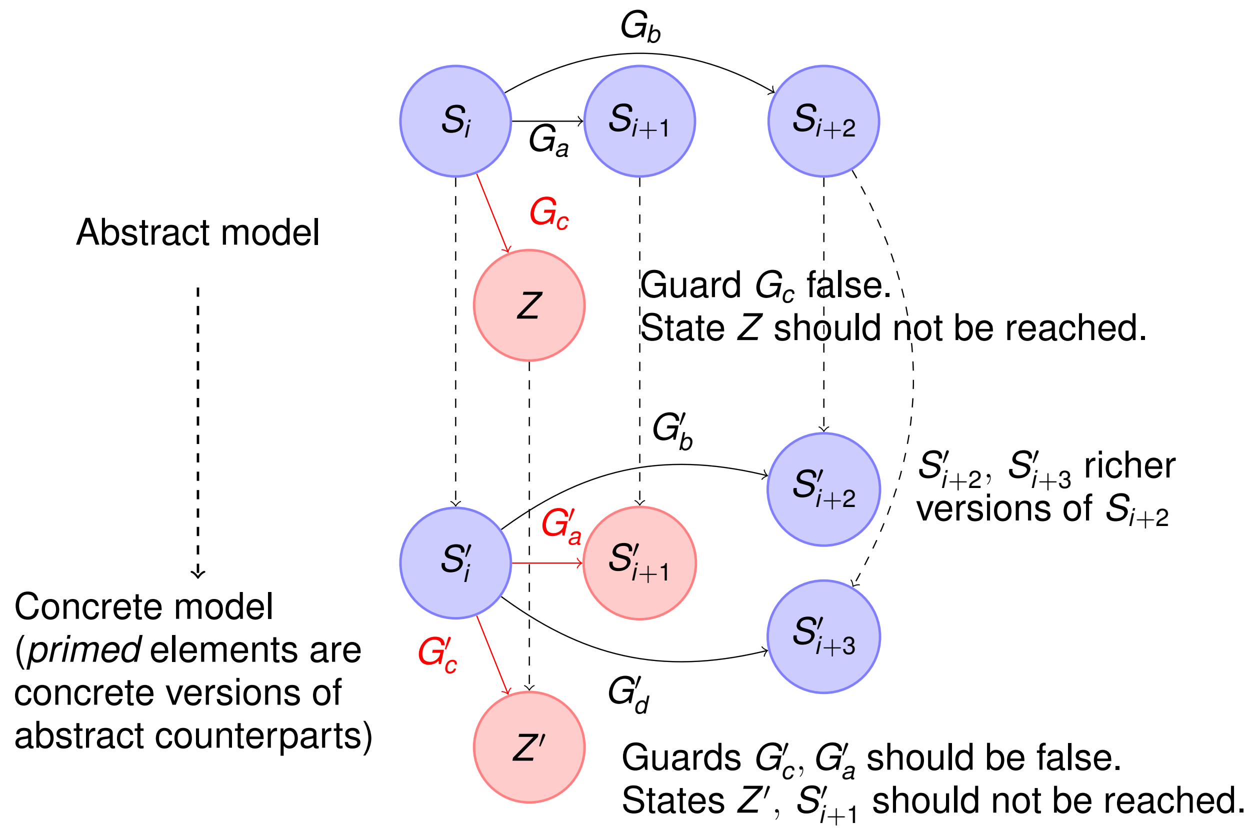
Concrete event with guards $H(c,w)$ and b-a predicate $w' = F(c,w)$

Correctness of Event Refinement



1. The concrete guard is stronger than the abstract one (Guard Strengthening, following slides)
2. Each concrete action is simulated by its abstract counterpart (Concrete Invariant Preservation, later)

The Essence of GRD



Key property: *Whenever a concrete guard is enabled, the corresponding abstract guard must be enabled too, i.e.,* $G' \Rightarrow G$

Proof Obligation: Guard Strengthening

Axioms Abstract Invariant Concrete Invariant Concrete Guard $\vdash$ Abstract Guard	$A(c)$ $I(c, v)$ $J(c, v, w)$ $H(c, w)$ $\vdash$ $G_i(c, v)$

GRD

Correctness of Invariant Refinement

$$
\begin{tikzcd}
G(c,v)\ \overset{I(c,v)}{v} \arrow[r, "\text{Abstract Event}"] & v' = \overset{I(c,v')}{E}(c,v) \\
H(c,w)\ w \arrow[u, "{J(c,v,w)}"'] \arrow[r, "\text{Concrete Event}"] & w' = F(c,w) \arrow[u, "{J(c,v',w')}"']
\end{tikzcd}
$$

Axioms $A(c)$ Abstract Invariants $I(c,v)$ Concrete Invariants $J(c,v,w)$ Concrete Guards $H(c,w)$ $\vdash$ Modified Concrete Invariant

$A(c)$ $I(c,v)$ $J(c,v,w)$ $H(c,w)$ $\vdash$ $J_j(c, E(c,v), F(c,w))$	INV

Adding New Events

- new events add transitions that have no abstract counterpart

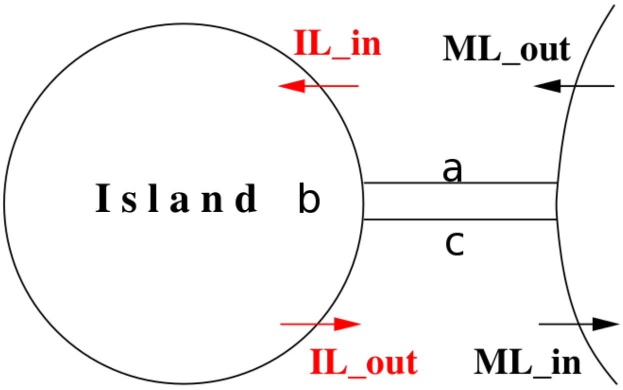
- can be seen as a kind of internal steps (w.r.t. abstract model)

- can only be seen by an observer who is “zooming in”

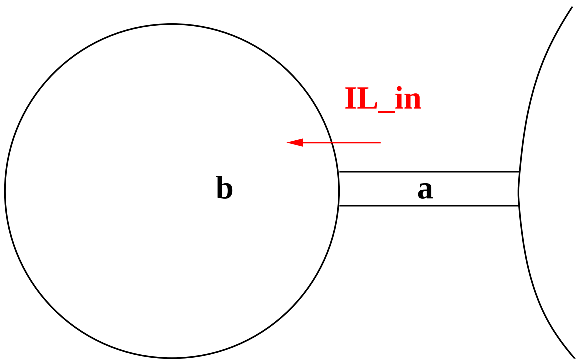
- temporal refinement: refined model has a finer time granularity

Suggestions?

Suggestions for IL_IN and IL_OUT?

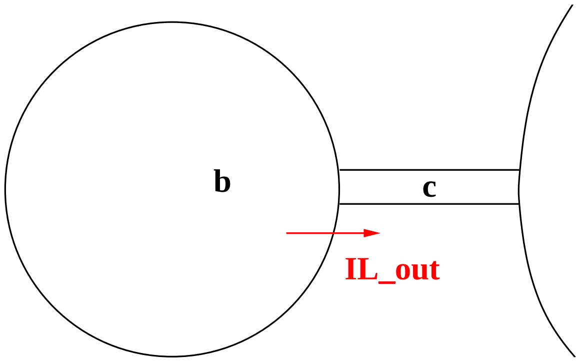


New Event IL_in



IL_in
  when
    $0 < a$
  then
    $a := a - 1$
    $b := b + 1$
  end

New Event IL_out



IL_out
  when
    $0 < b$
    $a = 0$
  then
    $b := b - 1$
    $c := c + 1$
  end

Refining New Events

- Modification of model ($a + b + c = n$) lead to refined events
  - ML_in, ML_out
- Refinement subject to proof obligations
  - Guard strengthening (for every refined event) — GRD
  - Invariant preservation (for every invariant and action) — INV
- Need to discharge the same proofs for new events
  - GRD: postulate existence of invisible event **skip**
    - **true** guard (so GRD is trivial)
    - action does not change state: $S' = S$
    - **skip** was happening, but we did not perceive its effects
  - INV: Same as before.
  - VAR: new events must not diverge.
    - Convergence was proven for abstract events.
    - It has to be proven for the new events.

Event IL_in Refines skip (1)

axm0_1 **axm0_2** **inv0_1** **inv0_2** **inv1_1** **inv1_2** **inv1_3** **inv1_4** **inv1_5** Concrete guards of IL_in $\vdash$ Modified Invariant **inv1_4**	$d \in \mathbb{N}$ $0 < d$ $n \in \mathbb{N}$ $n \leq d$ $a \in \mathbb{N}$ $b \in \mathbb{N}$ $c \in \mathbb{N}$ $a + b + c = n$ $a = 0 \;\vee\; c = 0$ $0 < a$ $\vdash$ $a - 1 + b + 1 + c = n$	IL_in / **inv1_4** / INV

IL_in
  when
    $0 < a$
  then
    $a := a - 1$
    $b := b + 1$
  end

Proof Obligation: Convergence of New Events (2)

Axioms $A(c)$, invariants $I(c, v)$, concrete invariant $J(c, v, w)$

New event with guard $H(c, w)$ and b-a predicate $w' = F(c, w)$

Variant $V(c, w)$

Axioms Abstract invariants Concrete invariants Concrete guard $\vdash$ Modified Var. $<$ Var.	$A(c)$ $I(c, v)$ $J(c, v, w)$ $H(c, w)$ $\vdash$ $V(c, F(c, w)) < V(c, w)$	VAR

Suggestions for a VARIANT?

(i.e., an expression which decreases
every time IL_out and IL_in are executed)

```
IL_in
  when
    0 < a
  then
    a := a − 1
    b := b + 1
  end
```

```
IL_out
  when
    0 < b
    a = 0
  then
    b := b − 1
    c := c + 1
  end
```

Proposed Variant

variant_1: $2 * a + b$

- Weighted sum of a and b

Relative Deadlock Freedom

There a no new deadlocks in the concrete model, that is, all deadlocks of the concrete model are already present in the abstract model.

Proof obligation requires that whenever some abstract event is enabled then so is some concrete event.

This proof obligaiton is optional (depending on system under study).

Proof Obligation: Relative Deadlock Freedom

The $G_i(c, v)$ are the abstract guards

The $H_i(c, v)$ are the concrete guards

If some abstract guard is true then so is some concrete guard:

$A(c)$ $I(c, v)$ $J(c, v, w)$ $G_1(c, v) \lor \ldots \lor G_m(c, v)$ $\vdash$ $H_1(c, w) \lor \ldots \lor H_n(c, w)$	DLF

axm0_1
axm0_2
inv0_1
inv0_2
inv1_1
inv1_2
inv1_3
inv1_4
inv1_5
Disjunction of abstract guards
$\vdash$
Disjunction of concrete guards

$$
\begin{array}{l}
d \in \mathbb{N} \\
0 < d \\
n \in \mathbb{N} \\
n \leq d \\
a \in \mathbb{N} \\
b \in \mathbb{N} \\
c \in \mathbb{N} \\
a + b + c = n \\
a = 0 \quad \lor \quad c = 0 \\
0 < n \quad \lor \quad n < d \\
\vdash \\
(a + b < d \ \land \ c = 0) \ \lor \\
c > 0 \quad \lor \quad a > 0 \\
(b > 0 \ \land \ a = 0)
\end{array}
$$

DLF

ML_out
when
$a + b < d$
$c = 0$
then
$a := a + 1$
end

ML_in
when
$c > 0$
then
$c := c - 1$
end

IL_in
when
$a > 0$
then
$a := a - 1$
$b := b + 1$
end

IL_out
when
$b > 0$
$a = 0$
then
$b := b - 1$
$c := c + 1$
end

constants: d

variables: a, b, c

inv1_1: $a \in \mathbb{N}$

inv1_2: $b \in \mathbb{N}$

inv1_3: $c \in \mathbb{N}$

inv1_4: $a + b + c = n$

inv1_5: $a = 0 \quad \lor \quad c = 0$

variant1: $2 * a + b$

Events of the First Refinement

init
$a := 0$
$b := 0$
$c := 0$

ML_in
when
$0 < c$
then
$c := c - 1$
end

ML_out
when
$a + b < d$
$c = 0$
then
$a := a + 1$
end

IL_in
when
$0 < a$
then
$a := a - 1$
$b := b + 1$
end

IL_out
when
$0 < b$
$a = 0$
then
$b := b - 1$
$c := c + 1$
end

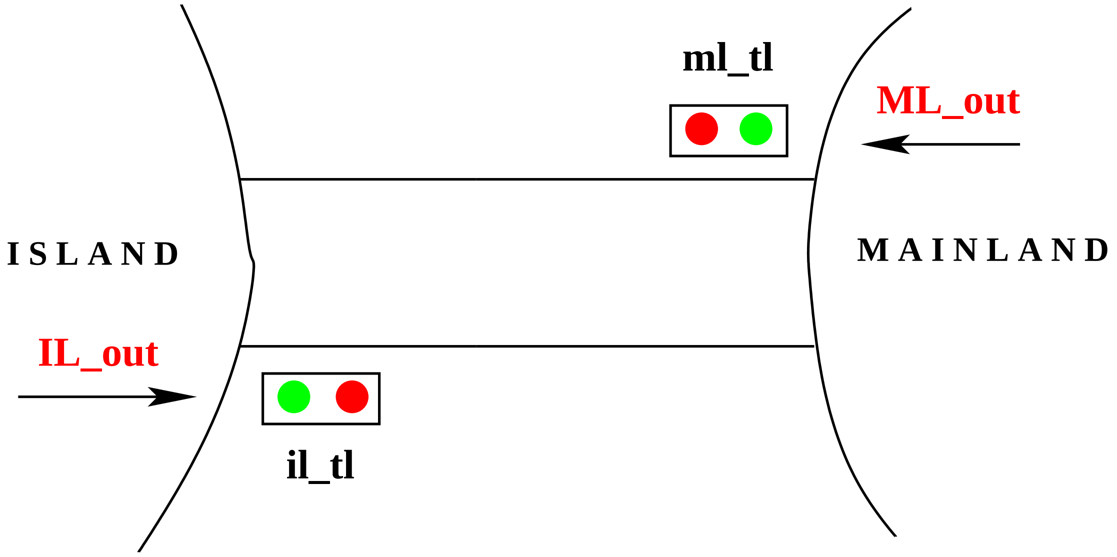
Our Refinement Strategy

- Initial model: Limiting the number of cars (FUN-2)

- First refinement: Introducing the one way bridge (FUN-3)

- Second refinement: Introducing the traffic lights (EQP-1,2,3)

- Third refinement: Introducing the sensors (EQP-4,5)



Extending Constants and Variables

set:	*COLOR*
constants:	*red, green*

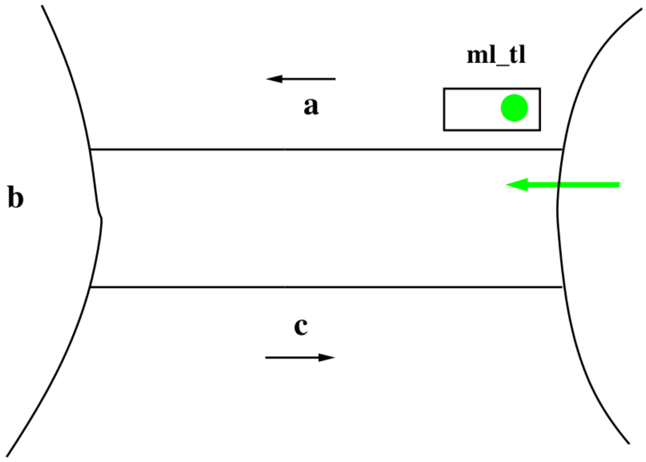
axm2_1:	$COLOR = \{red, green\}$
axm2_2:	$red \neq green$

$il\_tl \in COLOR$

$ml\_tl \in COLOR$

Note: IL_in and ML_in not modified now (cars can leave bridge without problems)

Extending the Invariant



Invariant for ml_tl?

Discussion on Implication Direction

Why

$$ml\_tl = \textbf{green} \Rightarrow c = 0 \land a + b < d$$

and not

$$c = 0 \land a + b < d \Rightarrow ml\_tl = \textbf{green}$$

or

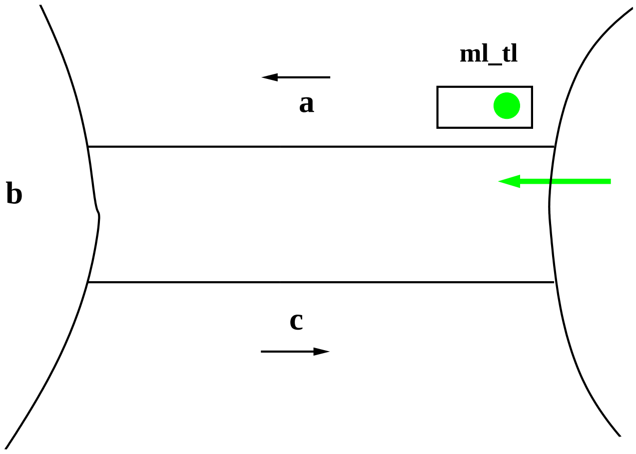
$$c = 0 \lor a + b < d \Rightarrow ml\_tl = \textbf{green}$$

?

Hint:

- What would be an undesirable situation?
- Which invariant prevents it?

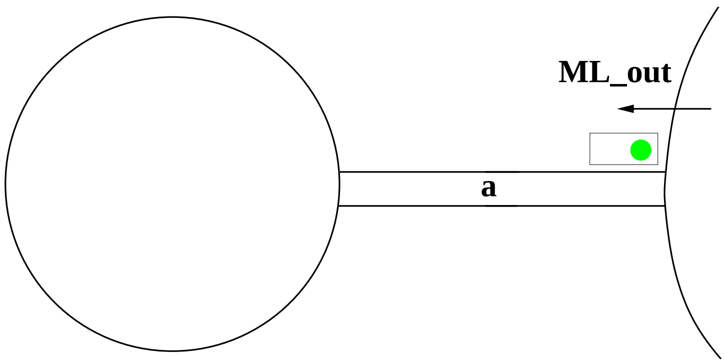
Extending the Invariant (1)



- A green mainland traffic light implies safe access to the bridge

$$ml\_tl = \mathbf{green} \;\Rightarrow\; c = 0 \;\wedge\; a + b < d$$

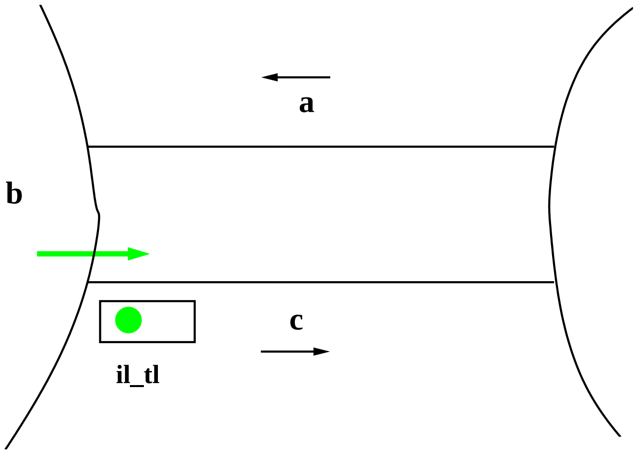
Refining Event ML_out



(abstract_)ML_out
when
$c = 0$
$a + b < d$
then
$a := a + 1$
end

(concrete_)ML_out
when
$ml\_tl = \mathbf{green}$
then
$a := a + 1$
end

Extending the Invariant (2)



- A green island traffic light implies safe access to the bridge

$$\boxed{il\_tl = \text{green} \;\Rightarrow\; a = 0 \;\wedge\; 0 < b}$$

Summary of State Refinement (so far)

variables: a, b, c, ml_tl, il_tl

inv2_1: $ml\_tl \in COLOR$

inv2_2: $il\_tl \in COLOR$

inv2_3: $ml\_tl = \text{green} \;\Rightarrow\; a + b < d \;\wedge\; c = 0$

inv2_4: $il\_tl = \text{green} \;\Rightarrow\; 0 < b \;\wedge\; a = 0$

Proving the Refinement of the Four Old Events

- We have to apply 3 Proof Obligations:
  - GRD,
  - SIM,
  - INV

- On 4 events: ML_out, IL_out, ML_in, IL_in

- And 2 main invariants:

inv2_3:	$ml\_tl = \textbf{green} \;\Rightarrow\; a + b < d \;\wedge\; c = 0$
inv2_4:	$il\_tl = \textbf{green} \;\Rightarrow\; 0 < b \;\wedge\; a = 0$

Proving Preservation of inv2_4 by Event ML_out

axm0_1	$d \in \mathbb{N}$	
axm0_2	$0 < d$	
axm2_1	$COLOR = \{\textbf{green}, \textbf{red}\}$	
axm2_2	$\textbf{green} \neq \textbf{red}$	
inv0_1	$n \in \mathbb{N}$	
inv0_2	$n \leq d$	
inv1_1	$a \in \mathbb{N}$	
inv1_2	$b \in \mathbb{N}$	
inv1_3	$c \in \mathbb{N}$	
inv1_4	$a + b + c = n$	ML_out / **inv2_4** / INV
inv1_5	$a = 0 \;\vee\; c = 0$	
inv2_1	$ml\_tl \in COLOR$	
inv2_2	$il\_tl \in COLOR$	
inv2_3	$ml\_tl = \textbf{green} \Rightarrow a + b < d \wedge c = 0$	
inv2_4	$il\_tl = \textbf{green} \Rightarrow 0 < b \wedge a = 0$	
Guard of event ML_out	$ml\_tl = \textbf{green}$	
$\vdash$	$\vdash$	
Modified invariant **inv2_4**	$il\_tl = \textbf{green} \Rightarrow 0 < b \wedge a + 1 = 0$	

```
ML_out
  when
    ml_tl = green
  then
    a := a + 1
  end
```

The Solution

- In both cases, we were stopped by attempting to prove the following

$$\begin{array}{l} \textbf{green} \neq \textbf{red} \\ il\_tl = \textbf{green} \\ ml\_tl = \textbf{green} \\ \vdash \\ 1 = 0 \end{array}$$

Both traffic lights are assumed to be green!

- This indicates that an "obvious" invariant was missing
- In fact, at least one of the two traffic lights must be red

inv2_5: $ml\_tl = \textbf{red} \quad \lor \quad il\_tl = \textbf{red}$

Going back to the Requirements Document

inv2_5: $ml\_tl = \textbf{red} \quad \lor \quad il\_tl = \textbf{red}$

This could have been deduced from these requirements

The bridge is one way or the other, not both at the same time	FUN-3

Cars are not supposed to pass on a red traffic light, only on a green one	EQP-3

What we Have to Prove

- ML_out / **inv2_4** / INV done

- IL_out / **inv2_3** / INV done

- ML_out / **inv2_3** / INV

- IL_out / **inv2_4** / INV

- ML_tl_green / **inv2_5** / INV

- IL_tl_green / **inv2_5** / INV

Proving Preservation of inv2_3 by Event ML_out

axm0_1	$d \in \mathbb{N}$	
axm0_2	$0 < d$	
axm2_1	$COLOR = \{\textsf{green}, \textsf{red}\}$	
axm2_2	$\textsf{green} \neq \textsf{red}$	
inv0_1	$n \in \mathbb{N}$	
inv0_2	$n \leq d$	
inv1_1	$a \in \mathbb{N}$	
inv1_2	$b \in \mathbb{N}$	
inv1_3	$c \in \mathbb{N}$	
inv1_4	$a + b + c = n$	
inv1_5	$a = 0 \;\lor\; c = 0$	
inv2_1	$ml\_tl \in COLOR$	
inv2_2	$il\_tl \in COLOR$	**ML_out** / inv2_3 / **INV**
inv2_3	$ml\_tl = \textsf{green} \Rightarrow a + b < d \land c = 0$	
inv2_4	$il\_tl = \textsf{green} \Rightarrow 0 < b \land a = 0$	
Guard of ML_out	$ml\_tl = \textsf{green}$	
$\vdash$	$\vdash$	
Modified **inv2_3**	$ml\_tl = \textsf{green} \Rightarrow a + 1 + b < d \land c = 0$	

```
ML_out
  when
    ml_tl = green
  then
    a := a + 1
  end
```

Tentative Proof (cont'd)

$\ldots$

$a+b<d$
$c=0$
$ml\_tl = \text{green}$
$\vdash$
$a+1+b<d \;\wedge\; c=0$

AND_R

$a+b<d$
$c=0$
$ml\_tl = \text{green}$
$\vdash$
$a+1+b<d$

MON

$a+b<d$
$ml\_tl = \text{green}$
$\vdash$
$a+1+b<d$

?

$a+b<d$
$c=0$
$ml\_tl = \text{green}$
$\vdash$
$c=0$

MON

$c=0 \;\vdash\; c=0$

HYP

- This requires splitting the ML_out in two separate events ML_out_1 and ML_out_2

```
ML_out_1
  when
    ml_tl = green
    a + 1 + b < d
  then
    a := a + 1
  end
```

```
ML_out_2
  when
    ml_tl = green
    a + 1 + b = d
  then
    a := a + 1
    ml_tl := red
  end
```

Intuitive Explanation


```
ML_out_1
  when
    ml_tl = green
    a + 1 + b < d
  then
    a := a + 1
  end
```

```
ML_out_2
  when
    ml_tl = green
    a + 1 + b = d
  then
    a := a + 1
    ml_tl := red
  end
```

- When $a+1+b=d$ then only one more car can enter the island

- Consequently, the traffic light ml_tl must be turned red (while the car enters the bridge)

Proofs Can be Done Now

- inv2_3 preservation by ML_out_1 can be proven now.
- Same with inv2_3 preservation by ML_out_2.
- Something similar happens with proving preservation of inv2_4 by IL_out:

IL_out_1
when
$il\_tl = \textbf{green}$
$b \neq 1$
then
$b, c := b - 1, c + 1$
end

IL_out_2
when
$il\_tl = \textbf{green}$
$b = 1$
then
$b, c := b - 1, c + 1$
$il\_tl := \textbf{red}$
end

- When b = 1, then only one car remains in the island.
- Consequently, the traffic light il_tl can be turned red (after this car has left).

Correcting the New Events

But the new invariant **inv2_5** is not preserved by the new events

inv2_5: $ml\_tl = \textbf{red} \quad \vee \quad il\_tl = \textbf{red}$

Unless we correct them as follows:

ML_tl_green
when
$ml\_tl = \textbf{red}$
$a + b < d$
$c = 0$
then
$ml\_tl := \textbf{green}$
$il\_tl := \textbf{red}$
end

IL_tl_green
when
$il\_tl = \textbf{red}$
$0 < b$
$a = 0$
then
$il\_tl := \textbf{green}$
$ml\_tl := \textbf{red}$
end

Summary of the Proof Situation

- Correct event refinement: OK

- Absence of divergence of new events: FAILURE

- Absence of deadlock: ?

Divergence of the New Events


```
ML_tl_green
  when
    ml_tl = red
    a + b < d
    c = 0
  then
    ml_tl := green
    il_tl := red
  end
```

```
IL_tl_green
  when
    il_tl = red
    0 < b
    a = 0
  then
    il_tl := green
    ml_tl := red
  end
```

When a and c are both equal to 0 and b is positive, then both events are always alternatively enabled

The lights can change colors very rapidly

Solving Divergence

- Regulate when lights can turn red / green.
- Turn green only when at least one car has passed in the other direction.
- Two additional variables:
  $\boxed{\textbf{inv2\_6:}\ \text{ml\_pass} \in \{0, 1\}}$
  $\boxed{\textbf{inv2\_7:}\ \text{il\_pass} \in \{0, 1\}}$
- Their values are changed / consulted by {IL,ML}_out_{1,2} and {ML,TL}_tl_green (not detailed here - please refer to the book chapter).
- $\boxed{\text{Variant: } \textbf{variant\_2:}\ \text{ml\_pass} + \text{il\_pass}}$
- Convergence can be proven with it

Our Refinement Strategy

- Initial model: Limiting the number of cars (FUN_2)

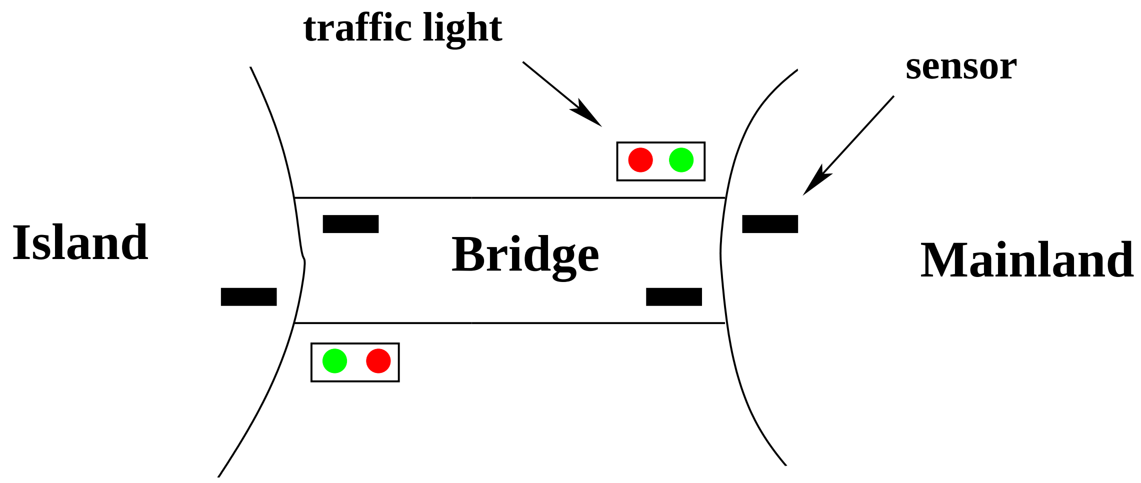
- First refinement: Introducing the one way bridge (FUN_3)

- Second refinement: Introducing the traffic lights (EQP_1,2,3)

- Third refinement: Introducing the sensors (EQP_4,5)

Third Refinement: Adding Car Sensors

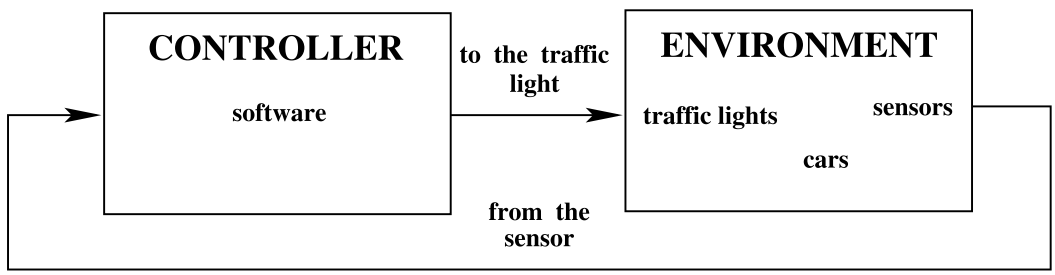
Reminder of the physical system



Closed Model

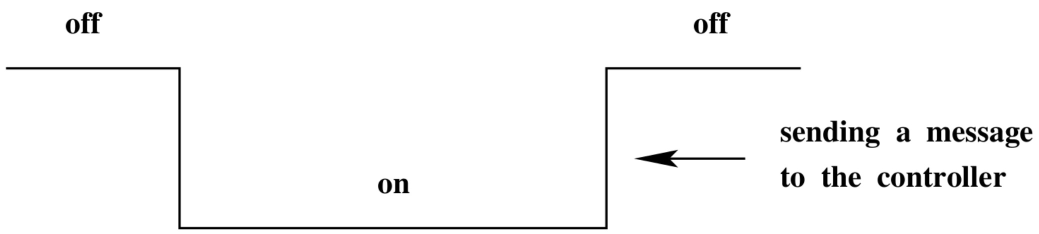
-We want to clearly identify in our model:

- The controller
- The environment
- The communication channels between the two

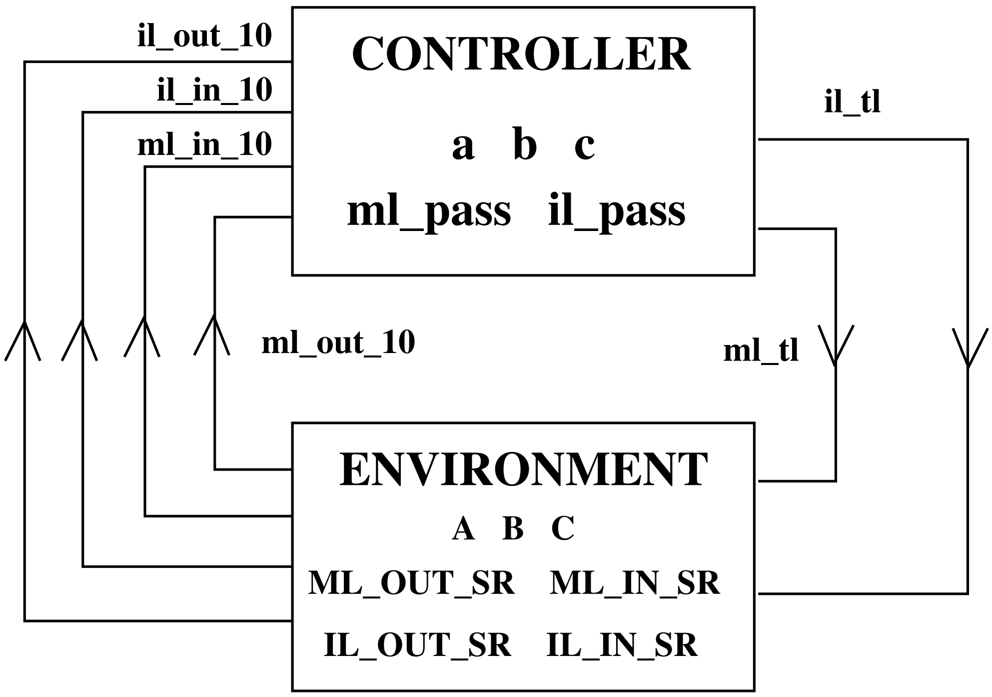


Controller and Environment, Input and Output

- Controller variables: **a, b, c, ml_pass, il_pass**
- Environment variables: **A, B, C, ML_OUT_SR, ML_ IN_SR, IL_OUT_SR, IL_IN_SR**
  - These new variables denote physical objects
- Output channels: **ml_tl, il_tl**.
- Input channels: **ml_out_10, ml_in_10, il_in_10, il_out_10**.
- A message is sent when a sensor moves from “on” to “off”:



Summary



Constants

carrier sets: $\ldots, SENSOR$

constants: $\ldots, on, off$

axm3_1: $SENSOR \ = \ \{on, off\}$

axm3_2: $on \neq off$

Variables (1)

$\mathrm{inv3\_1:} \quad ML\_OUT\_SR \ \in \ SENSOR$

$\mathrm{inv3\_2:} \quad ML\_IN\_SR \ \in \ SENSOR$

$\mathrm{inv3\_3:} \quad IL\_OUT\_SR \ \in \ SENSOR$

$\mathrm{inv3\_4:} \quad IL\_IN\_SR \ \in \ SENSOR$

Variables (2)

$$\begin{aligned}
&\text{inv3_5}: && A \in \mathbb{N}\\
&\text{inv3_6}: && B \in \mathbb{N}\\
&\text{inv3_7}: && C \in \mathbb{N}\\
&\text{inv3_8}: && ml_out_10 \in \text{BOOL}\\
&\text{inv3_9}: && ml_in_10 \in \text{BOOL}\\
&\text{inv3_10}: && il_out_10 \in \text{BOOL}\\
&\text{inv3_11}: && il_in_10 \in \text{BOOL}
\end{aligned}$$

Invariants (1)

When sensors are on, there are cars on them

$$\begin{aligned}
&\text{inv3_12}: && IL_IN_SR = on \Rightarrow A > 0\\
&\text{inv3_13}: && IL_OUT_SR = on \Rightarrow B > 0\\
&\text{inv3_14}: && ML_IN_SR = on \Rightarrow C > 0
\end{aligned}$$

The sensors are used to detect the presence of cars entering or leaving the bridge	EQP-5

Invariants (2)

Drivers obey the traffic lights

$$\begin{array}{ll} \mathrm{inv3\_15}: & ml\_out\_10 = \mathrm{TRUE} \ \Rightarrow \ ml\_tl = green \\ \mathrm{inv3\_16}: & il\_out\_10 = \mathrm{TRUE} \ \Rightarrow \ il\_tl = green \end{array}$$

Cars are not supposed to pass on a red traffic light, only on a green one	EQP-3

Invariants (3)

When a sensor is "on", the previous information is treated

$$\begin{array}{ll} \mathrm{inv3\_17}: & IL\_IN\_SR = on \ \Rightarrow \ il\_in\_10 = \mathrm{FALSE} \\ \mathrm{inv3\_18}: & IL\_OUT\_SR = on \ \Rightarrow \ il\_out\_10 = \mathrm{FALSE} \\ \mathrm{inv3\_19}: & ML\_IN\_SR = on \ \Rightarrow \ ml\_in\_10 = \mathrm{FALSE} \\ \mathrm{inv3\_20}: & ML\_OUT\_SR = on \ \Rightarrow \ ml\_out\_10 = \mathrm{FALSE} \end{array}$$

The controller must be fast enough so as to be able to treat all the information coming from the environment	FUN-5

Invariants (4)

Linking the physical and logical cars (1)

$$
\begin{array}{ll}
\text{inv3\_21}: & il\_in\_10 = \text{TRUE} \ \wedge \ ml\_out\_10 = \text{TRUE} \ \Rightarrow \ A = a \\
\text{inv3\_22}: & il\_in\_10 = \text{FALSE} \ \wedge \ ml\_out\_10 = \text{TRUE} \ \Rightarrow \ A = a + 1 \\
\text{inv3\_23}: & il\_in\_10 = \text{TRUE} \ \wedge \ ml\_out\_10 = \text{FALSE} \ \Rightarrow \ A = a - 1 \\
\text{inv3\_24}: & il\_in\_10 = \text{FALSE} \ \wedge \ ml\_out\_10 = \text{FALSE} \ \Rightarrow \ A = a
\end{array}
$$

Invariants (5)

Linking the physical and logical cars (2)

$$
\begin{array}{ll}
\text{inv3\_25}: & il\_in\_10 = \text{TRUE} \ \wedge \ il\_out\_10 = \text{TRUE} \ \Rightarrow \ B = b \\
\text{inv3\_26}: & il\_in\_10 = \text{TRUE} \ \wedge \ il\_out\_10 = \text{FALSE} \ \Rightarrow \ B = b + 1 \\
\text{inv3\_27}: & il\_in\_10 = \text{FALSE} \ \wedge \ il\_out\_10 = \text{TRUE} \ \Rightarrow \ B = b - 1 \\
\text{inv3\_28}: & il\_in\_10 = \text{FALSE} \ \wedge \ il\_out\_10 = \text{FALSE} \ \Rightarrow \ B = b
\end{array}
$$

$$
\begin{array}{ll}
\text{inv3\_29}: & il\_out\_10 = \text{TRUE} \ \wedge \ ml\_out\_10 = \text{TRUE} \ \Rightarrow \ C = c \\
\text{inv3\_30}: & il\_out\_10 = \text{TRUE} \ \wedge \ ml\_out\_10 = \text{FALSE} \ \Rightarrow \ C = c + 1 \\
\text{inv3\_31}: & il\_out\_10 = \text{FALSE} \ \wedge \ ml\_out\_10 = \text{TRUE} \ \Rightarrow \ C = c - 1 \\
\text{inv3\_32}: & il\_out\_10 = \text{FALSE} \ \wedge \ ml\_out\_10 = \text{FALSE} \ \Rightarrow \ C = c
\end{array}
$$

Invariants (6)

The basic properties hold for the physical cars

$$\text{inv3\_33}: \quad A = 0 \;\vee\; C = 0$$

$$\text{inv3\_34}: \quad A + B + C \leq d$$

The number of cars on the bridge and the island is limited	FUN-2

The bridge is one way or the other, not both at the same time	FUN-3

Adding New PHYSICAL Events (1)

ML_out_arr
when
$ML\_OUT\_SR = off$
$ml\_out\_10 = \text{FALSE}$
then
$ML\_OUT\_SR := on$
end

ML_in_arr
when
$ML\_IN\_SR = off$
$ml\_in\_10 = \text{FALSE}$
$C > 0$
then
$ML\_IN\_SR := on$
end

IL_in_arr
when
$IL\_IN\_SR = off$
$il\_in\_10 = \text{FALSE}$
$A > 0$
then
$IL\_IN\_SR := on$
end

IL_out_arr
when
$IL\_OUT\_SR = off$
$il\_out\_10 = \text{FALSE}$
$B > 0$
then
$IL\_OUT\_SR := on$
end

Adding New PHYSICAL Events (2)

ML_out_dep
  when
    $ML\_OUT\_SR = on$
    $ml\_tl = green$
  then
    $ML\_OUT\_SR := off$
    $ml\_out\_10 := \mathrm{TRUE}$
  end

ML_in_dep
  when
    $ML\_IN\_SR = on$
  then
    $ML\_IN\_SR := off$
    $ml\_in\_10 := \mathrm{TRUE}$
    $C = C - 1$
  end

IL_in_dep
  when
    $IL\_IN\_SR = on$
  then
    $IL\_IN\_SR := off$
    $il\_in\_10 := \mathrm{TRUE}$
    $A = A - 1$
    $B = B + 1$
  end

IL_out_dep
  when
    $IL\_OUT\_SR = on$
    $il\_tl = green$
  then
    $IL\_OUT\_SR := off$
    $il\_out\_10 := \mathrm{TRUE}$
    $B = B - 1$
    $C = C + 1$
  end

Refining Abstract Events (1)

ML_out_1
  when
    $\underline{ml\_out\_10 = \mathrm{TRUE}}$
    $a + b + 1 \neq d$
  then
    $a := a + 1$
    $ml\_pass := 1$
    $\underline{ml\_out\_10 := \mathrm{FALSE}}$
  end

ML_out_2
  when
    $\underline{ml\_out\_10 = \mathrm{TRUE}}$
    $a + b + 1 = d$
  then
    $a := a + 1$
    $ml\_tl := red$
    $ml\_pass := 1$
    $\underline{ml\_out\_10 := \mathrm{FALSE}}$
  end

(abstract-)ML_out_1
  when
    $ml\_tl = \textbf{green}$
    $a + b + 1 \neq d$
  then
    $a := a + 1$
    $ml\_pass := 1$
  end

(abstract-)ML_out_2
  when
    $ml\_tl = \textbf{green}$
    $a + b + 1 = d$
  then
    $a := a + 1$
    $ml\_pass := 1$
    $ml\_tl := \textbf{red}$
  end

Refining Abstract Events (2)

IL_out_1
when
$\underline{il\_out\_10 = \mathrm{TRUE}}$
$b \neq 1$
then
$b := b - 1$
$c := c + 1$
$il\_pass := 1$
$\underline{il\_out\_10 := \mathrm{FALSE}}$
end

IL_out_2
when
$\underline{il\_out\_10 = \mathrm{TRUE}}$
$b = 1$
then
$b := b - 1$
$c := c + 1$
$il\_tl := red$
$il\_pass := 1$
$\underline{il\_out\_10 := \mathrm{FALSE}}$
end

(abstract-)IL_out_1
when
$il\_tl =$ green
$b \neq 1$
then
$b := b - 1$
$c := c + 1$
$il\_pass := 1$
end

(abstract-)IL_out_2
when
$il\_tl =$ green
$b = 1$
then
$b := b - 1$
$c := c + 1$
$il\_pass := 1$
$il\_tl :=$ red
end

Refining Abstract Events (3)

ML_in
when
$\underline{ml\_in\_10 = \mathrm{TRUE}}$
$0 < c$
then
$c := c - 1$
$\underline{ml\_in\_10 := \mathrm{FALSE}}$
end

IL_in
when
$\underline{il\_in\_10 = \mathrm{TRUE}}$
$0 < a$
then
$a := a - 1$
$b := b + 1$
$\underline{il\_in\_10 := \mathrm{FALSE}}$
end

(abstract-)ML_in
when
$0 < c$
then
$c := c - 1$
end

(abstract-)IL_in
when
$0 < a$
then
$a := a - 1$
$b := b + 1$
end

Refining Abstract Events (4)

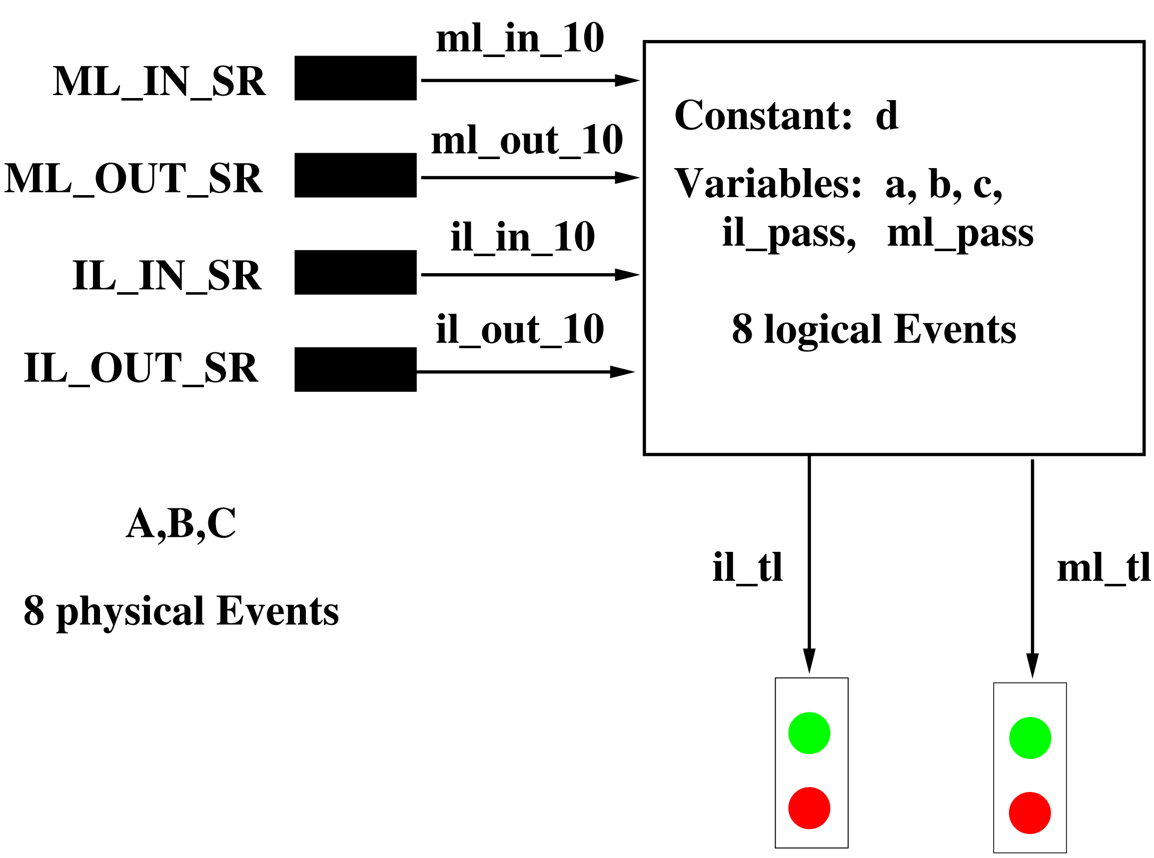
ML_tl_green
when
$ml\_tl = red$
$a + b < d$
$c = 0$
$il\_pass = 1$
$\underline{il\_out\_10 = \mathrm{FALSE}}$
then
$ml\_tl := green$
$il\_tl := red$
$ml\_pass := \mathrm{FALSE}$
end

IL_tl_green
when
$il\_tl = red$
$a = 0$
$ml\_pass = 1$
$\underline{ml\_out\_10 = \mathrm{FALSE}}$
then
$il\_tl := green$
$ml\_tl := red$
$il\_pass := \mathrm{FALSE}$
end

(abstract-)ML_tl_green
when
$ml\_tl =$ red
$a + b < d$
$c = 0$
$il\_pass = 1$
then
$ml\_tl :=$ green
$il\_tl :=$ red
$ml\_pass := 0$
end

(abstract-)IL_tl_green
when
$il\_tl =$ red
$0 < b$
$a = 0$
$ml\_pass = 1$
then
$il\_tl :=$ green
$ml\_tl :=$ red
$il\_pass := 0$
end

Final Structure of the Controller



Questions on Proving

- What is to be systematically proved?
    - Invariant preservation
    - Correct refinements of transitions
    - No divergence of new transitions
    - No deadlock introduced in refinements

- When are these proofs done?

Questions on Proving (cont'd)

- Who states what is to be proved?
    - An automatic tool: the Proof Obligation Generator

- Who is going to perform these proofs?
    - An automatic tool: the Prover
    - Sometimes helped by the Engineer (interactive proving)

About Tools

- Three basic tools:
  - Proof Obligation Generator
  - Prover
  - Model translators into Hardware or Software languages

- These tools are embedded into a Development Data Base

- Such tools already exist in the Rodin Platform

Summary of Proofs on Example

- This development required 253 proofs
  - Initial model: 7 (0)
  - 1st refinement: 27 (0)
  - 2nd refinement: 81 (1)
  - 3rd refinement: 138 (3)

- All proved automatically (except 4) by the Rodin Platform