2. Controlling Cars on a Bridge

Jean-Raymond Abrial

2009

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.

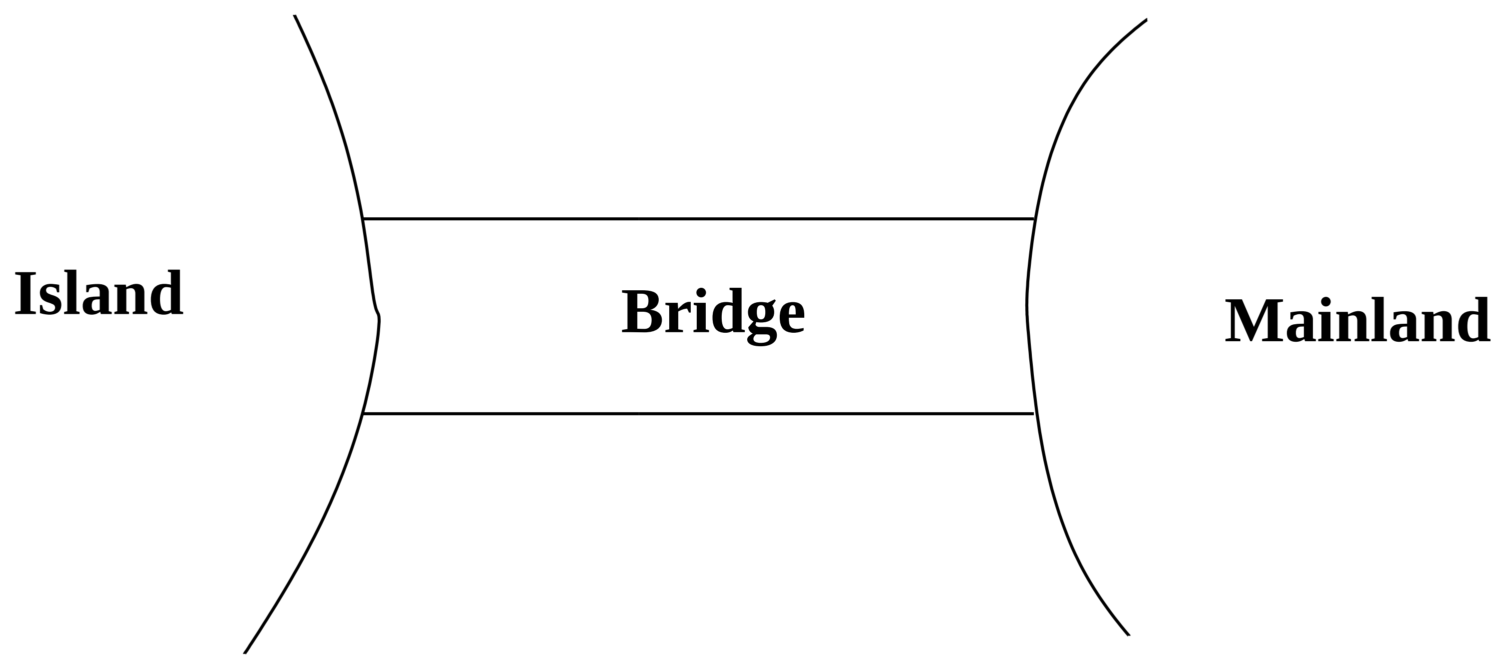
- The function of this system is to control cars on a narrow bridge.

- 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)

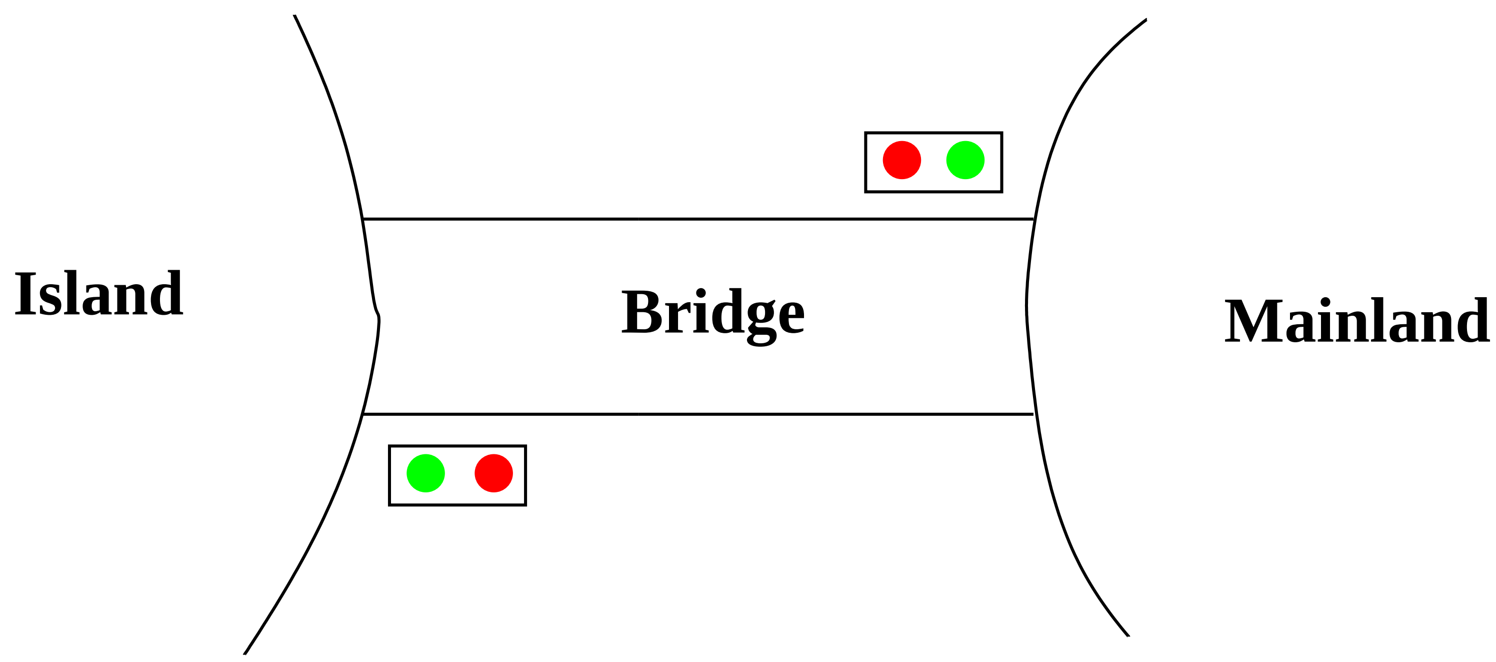
- The controller is equipped with two traffic lights with two colors.

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.

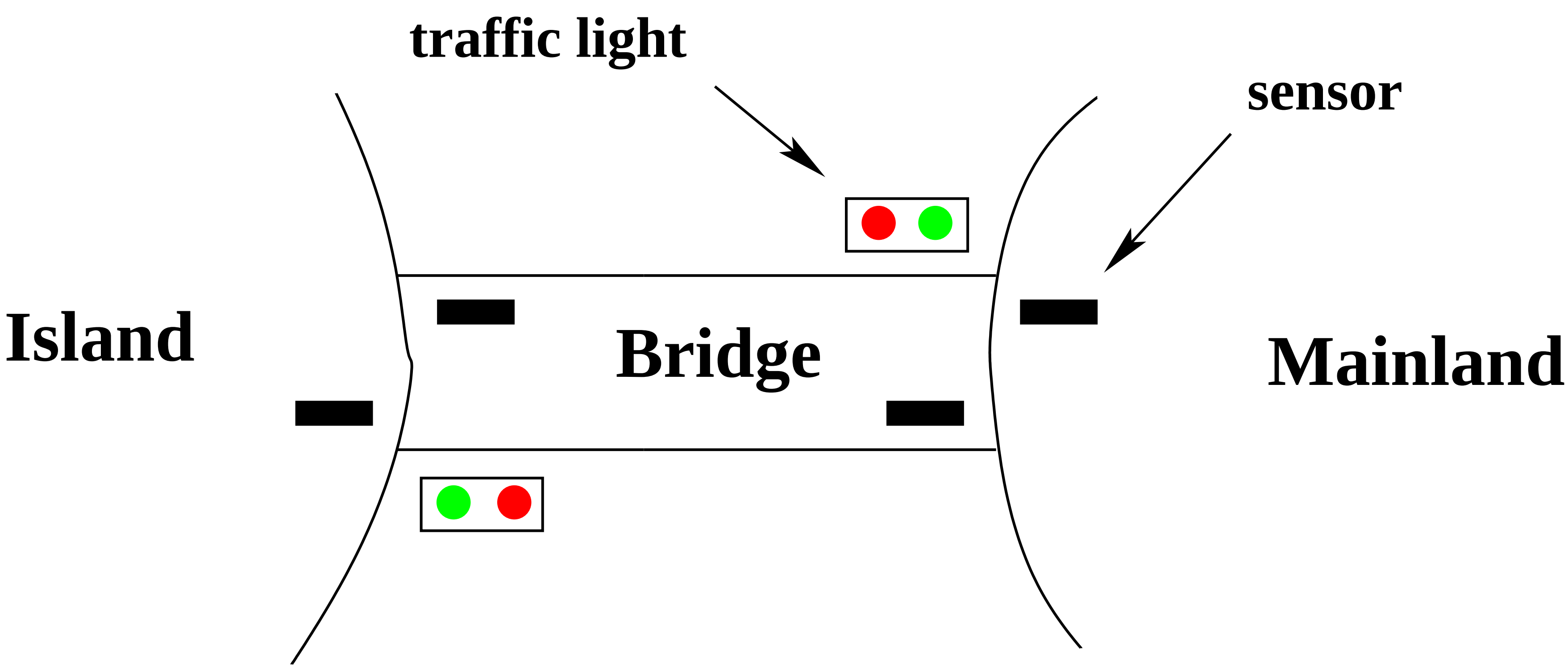
- There are four such sensors. Two of them are situated on the bridge and the other two are situated on the mainland and on the island.

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

The Reference Document (1)

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

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

The Reference Document (2)

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

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

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

The Reference Document (3)

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

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

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)

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)

Initial Model

- It is very simple

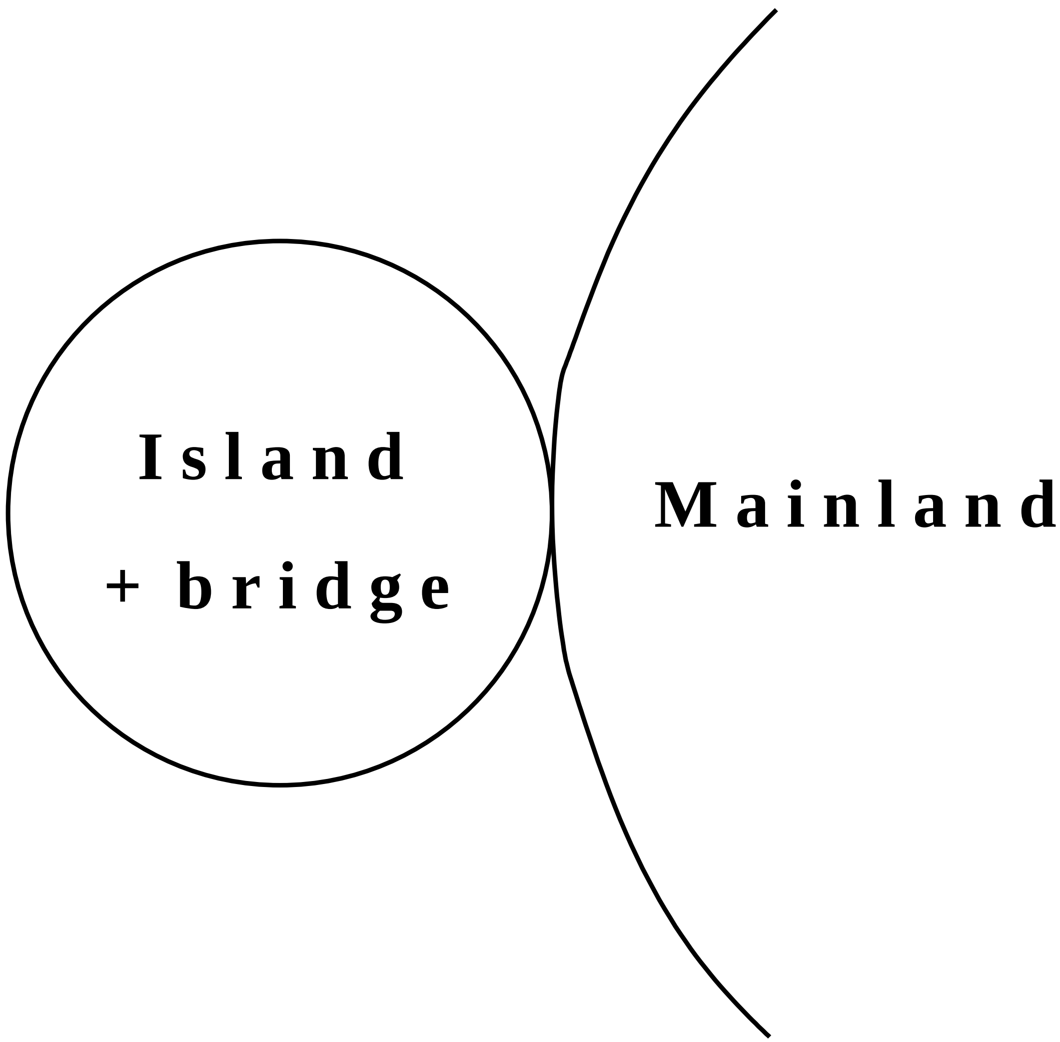
- We completely ignore the equipment: traffic lights and sensors

- We do not even consider the bridge

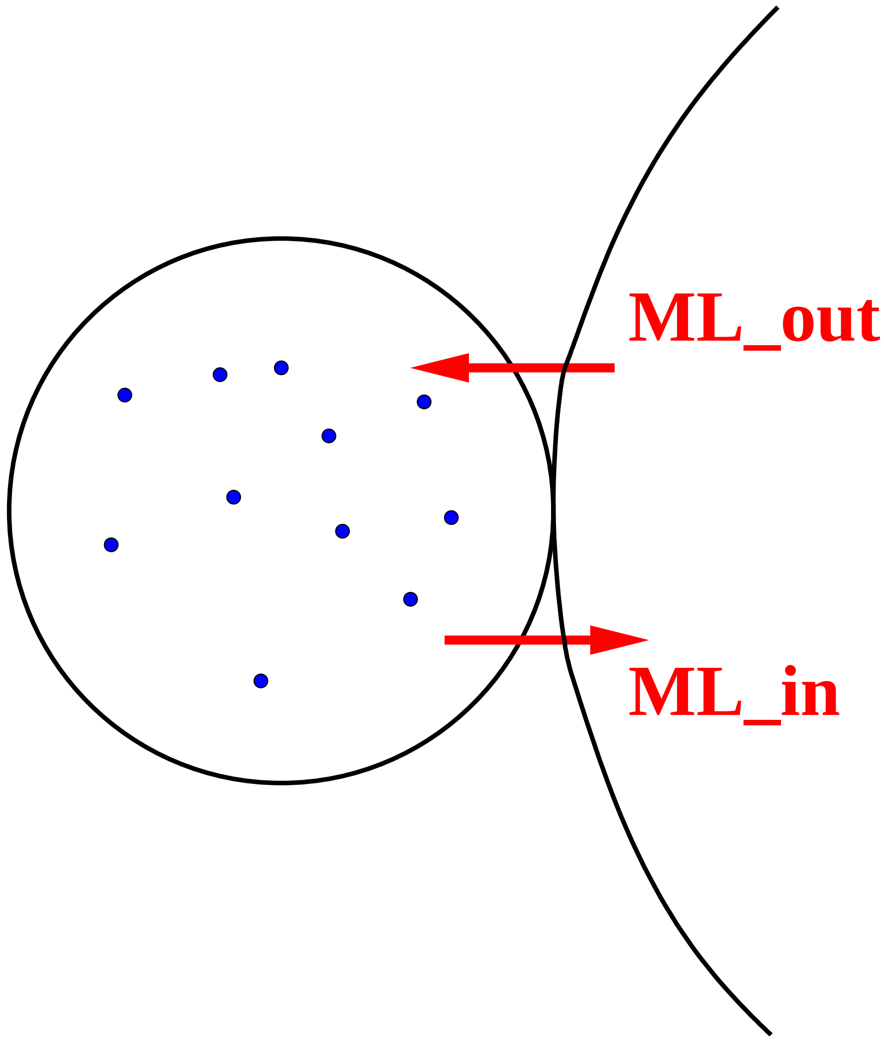
- We are just interested in the pair “island-bridge”

- We are focusing FUN-2: limited number of cars on island-bridge

A Situation as Seen from the Sky



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

Naming Conventions

- Labels **axm0_1**, **inv0_1**, ... are chosen systematically

- The label **axm** or **inv** recalls the purpose:

  axiom of constants or invariant of variables

- The **0** as in **inv0_1** stands for the initial model.

- Later we will have **inv1_1** for an invariant of refinement **1**, etc.

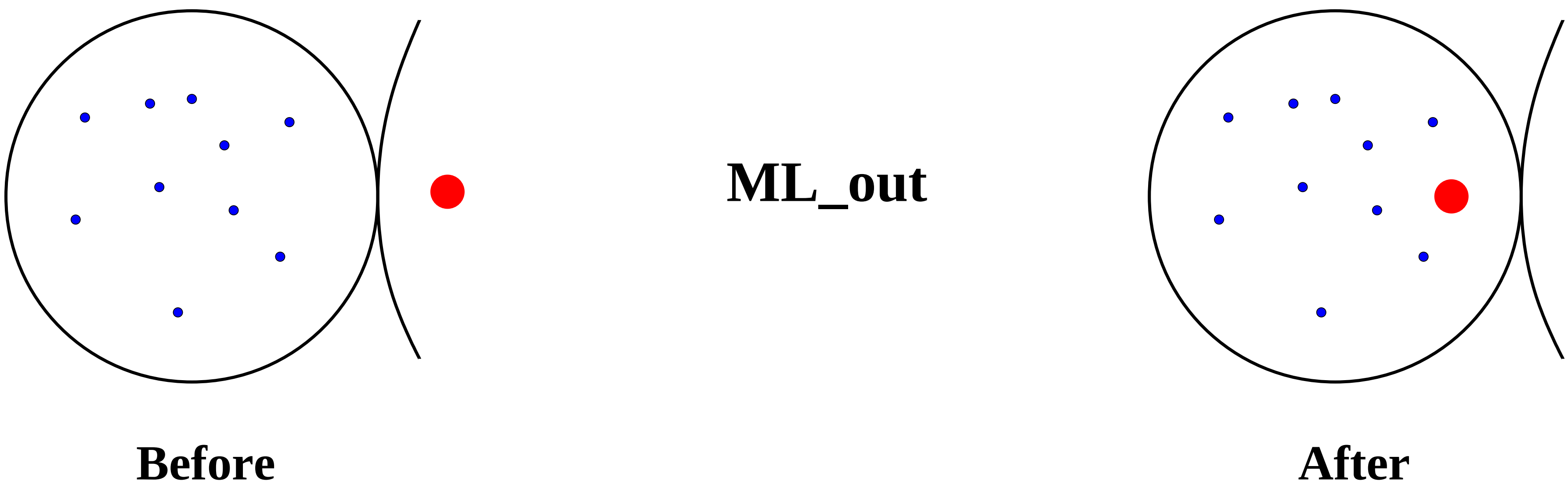
- The **1** like in **inv0_1** is a serial number

- Any convention is valid as long as it is systematic

Event ML_out

- This is the first transition (or event) that can be observed

- A car is leaving the mainland and entering the Island-Bridge

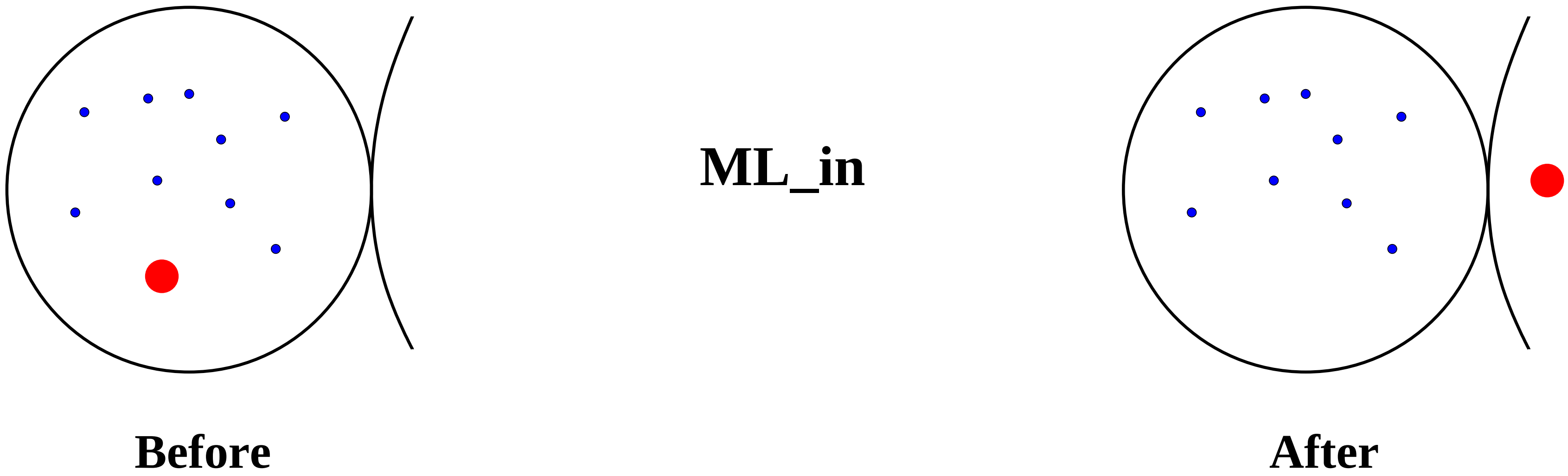


- The number of cars in the Island-Bridge is incremented

Event ML_in

- We can also observe a second transition (or event)

- A car leaving the Island-Bridge and re-entering the mainland



- The number of cars in the Island-Bridge is decremented

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.

Invariants

- An invariant is a constraint on the allowed values of the variables

- An invariant must hold on all reachable states of a model

- To verify that this holds we must show that

  1. the invariant holds for initial states (later), and
  2. the invariant is preserved by all events (following slides)

- We will formalize these two statements as proof obligations (POs)

- We need a rigorous proof showing that these POs indeed hold

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_1 / INV

$$\begin{array}{|l|}\hline \\ \quad d \;\in\; \mathbb{N} \\ \quad n \;\in\; \mathbb{N} \\ \quad n \le d \\ \quad n < d \\ \vdash \\ \quad n+1 \in \mathbb{N} \\ \\ \hline \end{array} \quad \text{MON} \quad \begin{array}{|l|}\hline \\ \quad n \;\in\; \mathbb{N} \\ \vdash \\ \quad n+1 \in \mathbb{N} \\ \\ \hline \end{array} \quad \text{P2}$$

- Adding new assumptions to a sequent does not affect its provability

A Formal Proof of: ML_out / inv0_2 / INV

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

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

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

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:

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

Adding the Forgotten Axiom

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

$$\boxed{\textbf{axm0\_2:} \quad 0 < d}$$

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

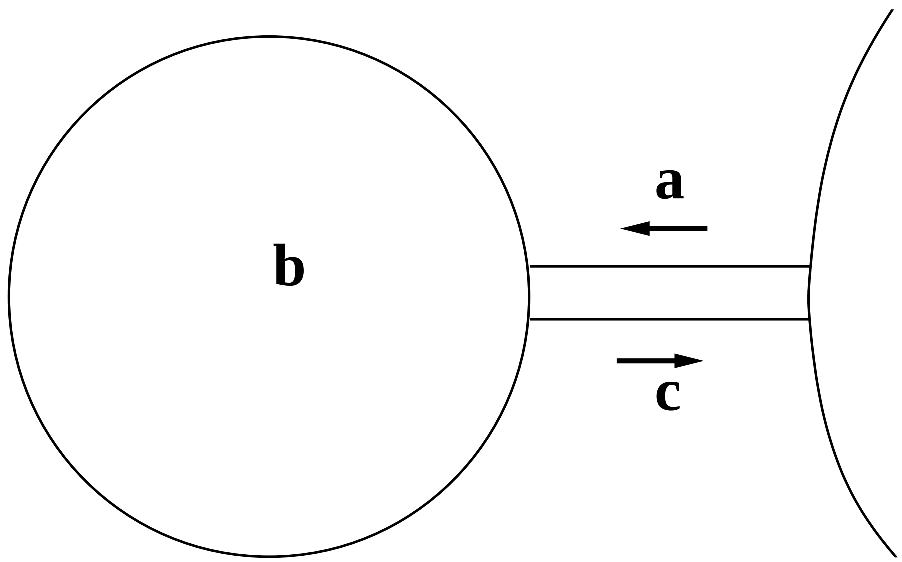
- Initial model: Limiting the number of cars (FUN-2)

- 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)

Introducing Three New Variables: a, b, and c



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

Refining the State: Summary

constants: d

variables: a, b, c

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

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

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

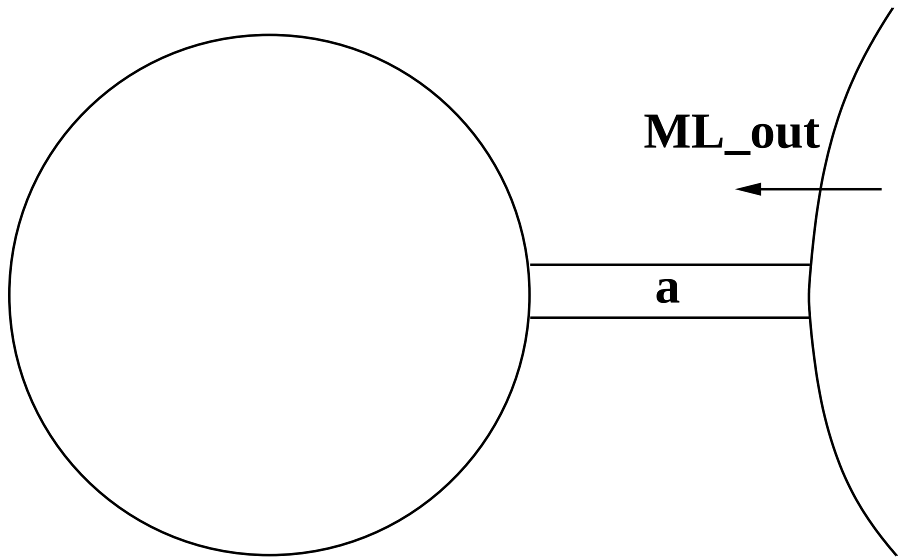
inv1_4: $a + b + c = n$

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

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

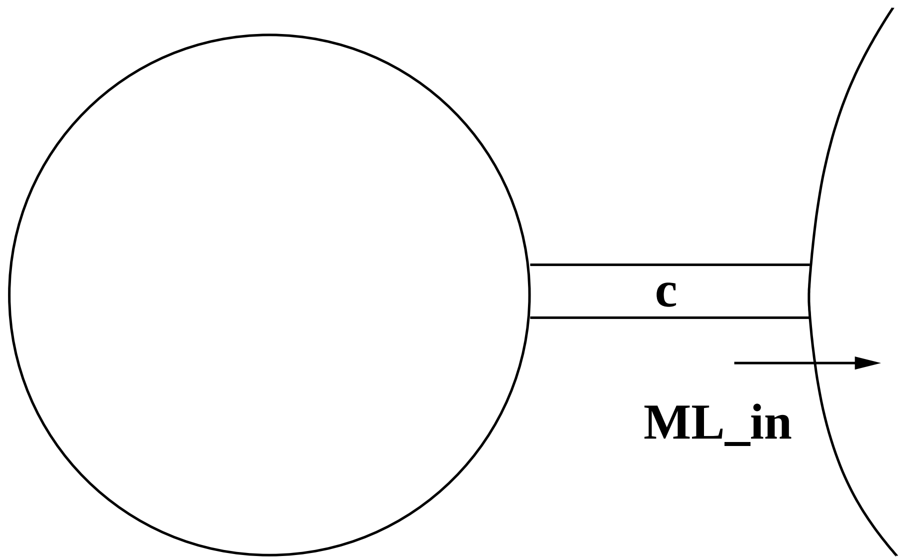
- **inv1_4** glues the abstract state, n, to the concrete state, a, b, c

Proposal for Refining Event ML_out



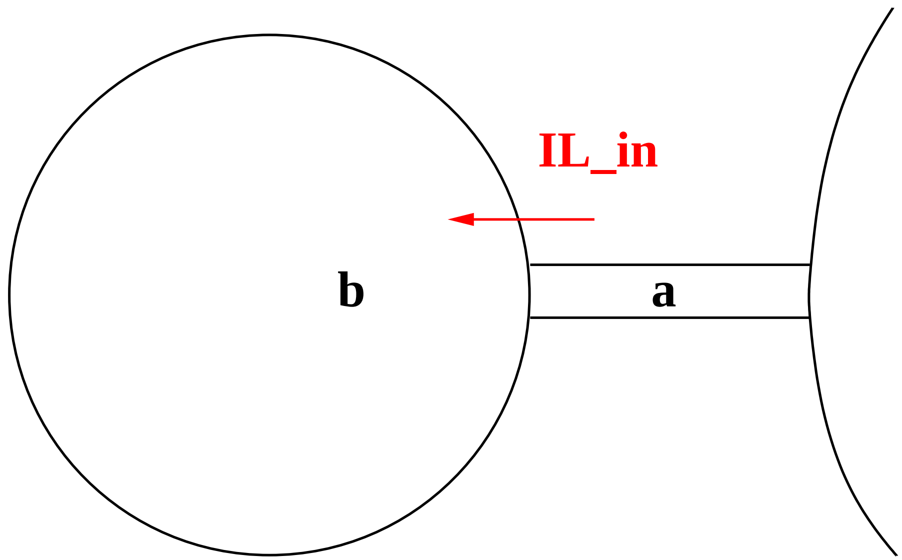
ML_out
when
$a + b < d$
$c = 0$
then
$a := a + 1$
end

Proposal for Refining Event ML_in



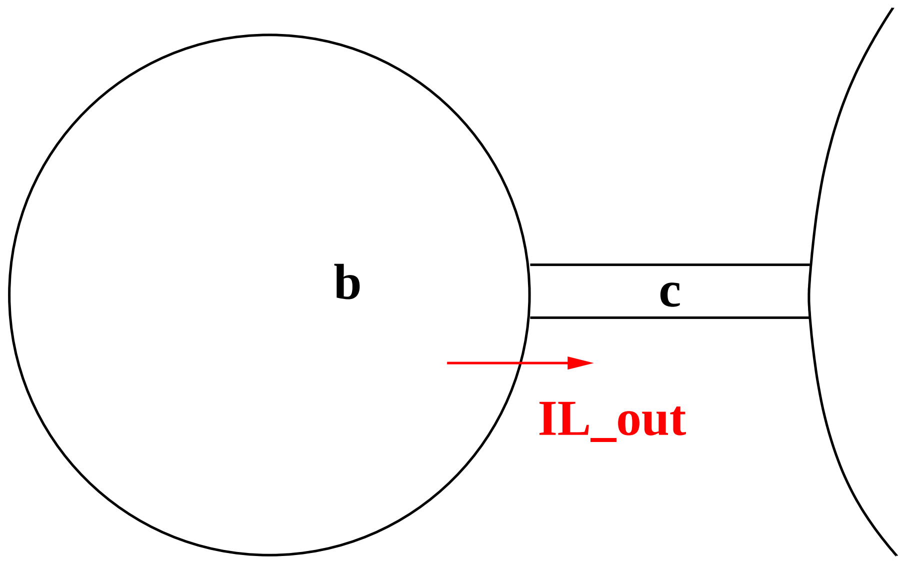
ML_in
  when
    $0 < c$
  then
    $c := c - 1$
  end

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

State of the First Refinement

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

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

Applying Guard Strengthening to Event ML_out

axm0_1
axm0_2
inv0_1
inv0_2
inv1_1
inv1_2
inv1_3
inv1_4
inv1_5
Concrete guards of ML_out

$\vdash$
Abstract guard of ML_out

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

ML_out / GRD

(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

Proof of ML_out / GRD

$$\begin{array}{|l|}\hline d \in \mathbb{N} \\ 0 < d \\ n \in \mathbb{N} \\ n \leq d \\ a \in \mathbb{N} \\ b \in \mathbb{N} \\ c \in \mathbb{N} \\ a+b+c=n \\ a=0 \;\vee\; c=0 \\ a+b<d \\ c=0 \\ \vdash \\ n<d \\ \hline \end{array} \quad \text{MON} \quad \begin{array}{|l|}\hline a+b+c=n \\ a+b<d \\ c=0 \\ \vdash \\ n<d \\ \hline \end{array} \quad \text{EQ\_LR} \quad \begin{array}{|l|}\hline a+b+0=n \\ a+b<d \\ \vdash \\ n<d \\ \hline \end{array} \quad \text{ARITH} \;\ldots$$

$$\ldots \quad \begin{array}{|l|}\hline a+b=n \\ a+b<d \\ \vdash \\ n<d \\ \hline \end{array} \quad \text{EQ\_LR} \quad \begin{array}{|l|}\hline n<d \\ \vdash \\ n<d \\ \hline \end{array} \quad \text{HYP}$$

The "rule" name ARITH stands for simple arithmetic simplifications.