Controlling Cars on a Bridge

Jean-Raymond Abrial
(edited by Thai Son Hoang)

Department of Computer Science
Swiss Federal Institute of Technology Zürich (ETH Zürich)

Bucharest DEPLOY 2-day Course, 14th-16th July, 2010

Outline

Outline

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

What you will Learn

- The concepts of state and events for defining models
- Some principles of system development:
  invariants and refinement
- A refresher of classical logic and simple arithmetic foundations
- A refresher of formal proofs

Outline

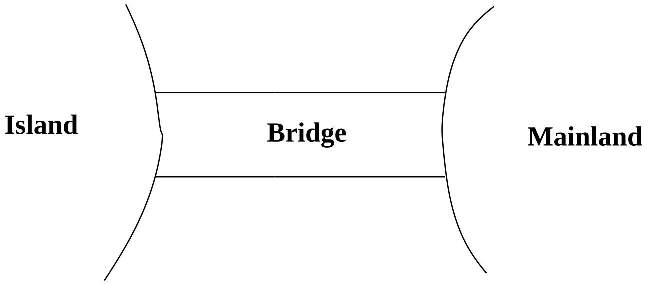
A Requirements Document (1)

- 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)

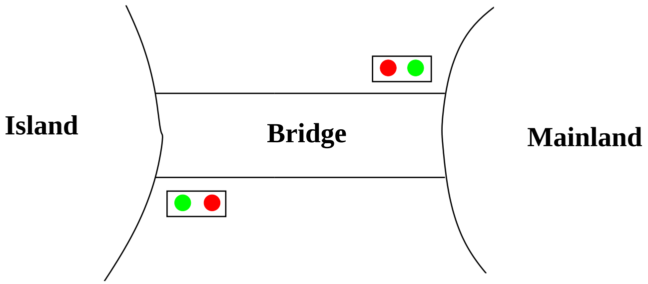
- The controller is equipped with two traffic lights with two colors.

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.

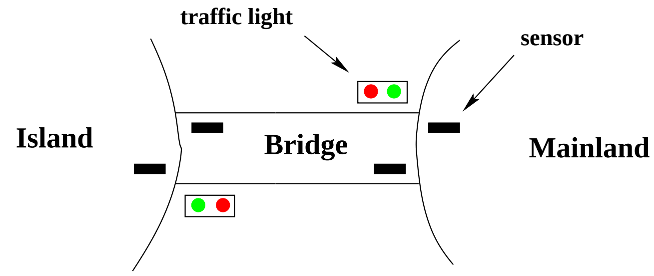
- There are four such sensors. Two of them are situated on the bridge and the other two are situated on the mainland and on the island.

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

Outline

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)

Outline

Initial Model

- It is very simple

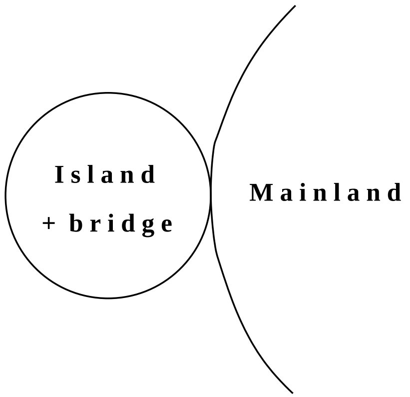
- We completely ignore the equipment: traffic lights and sensors

- We do not even consider the bridge

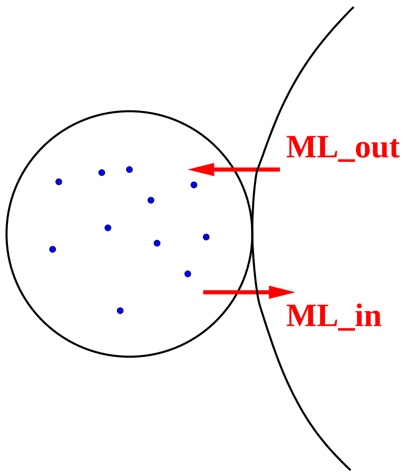
- We are just interested in the pair “island-bridge”

- 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: $n \leq d$

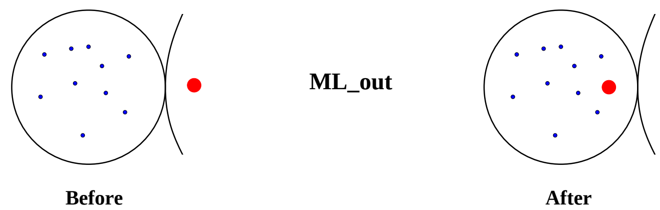
- n is the effective number of cars on the Island-Bridge
- n is a natural number (**inv0_1**)
- n is always smaller than or equal to d (**inv0_2**): this is FUN_2

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

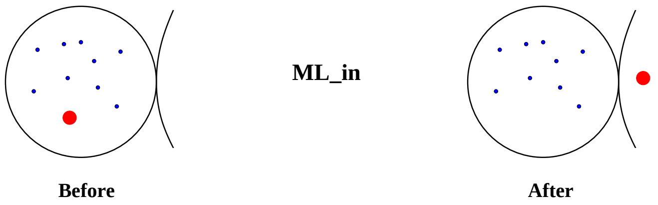


- The number of cars in the Island-Bridge is incremented

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

$$\boxed{\begin{array}{l}\text{ML\_out}\\ \quad n := n + 1\end{array}}$$

- Event ML_in decrements the number of cars

$$\boxed{\begin{array}{l}\text{ML\_in}\\ \quad n := n - 1\end{array}}$$

- 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

Towards the Proof: Before-after Predicates

- To each event can be associated a before-after predicate

- It describes the relation between the values of the variable(s) *just before* and *just after* the event occurrence

- The before-value is denoted by the variable name, say n

- The after-value is denoted by the *primed* variable name, say n'

Before-after Predicate Examples

The Events

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

The corresponding before-after predicates

$n' = n+1$	$n' = n-1$

These representations are equivalent.

About the Shape of the Before-after Predicates

- The before-after predicates we have shown are very simple

$$n' = n+1 \qquad\qquad n' = n-1$$

- The after-value n' is defined as a function of the before-value n

- This is because the corresponding events are deterministic

- In later lectures, we shall consider some non-deterministic events:

$$n' \;\in\; \{n+1, n+2\}$$

Intuition about Invariant Preservation

- Let us consider invariant **inv0_1**

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

- And let us consider event ML_out with before-after predicate

$$n' = n+1$$

- Preservation of **inv0_1** means that we have (just after ML_out):

$$n' \in \mathbb{N} \quad \text{that is} \quad n+1 \in \mathbb{N}$$

Being more Precise

- Under hypothesis $n \in \mathbb{N}$ the conclusion $n+1 \in \mathbb{N}$ holds

- This can be written as follows

$$n \in \mathbb{N} \;\vdash\; n+1 \in \mathbb{N}$$

- This type of statement is called a sequent (next slide)

- Sequent above: invariant preservation proof obligation for **inv0_1**

- More General form of this PO will be introduced shortly

Sequents

- A sequent is a formal statement of the following shape

$$\boxed{\mathrm{H} \;\vdash\; \mathbf{G}}$$

- H denotes a set of predicates: the hypotheses (or assumptions)

- **G** denotes a predicate: the goal (or conclusion)

- The symbol "$\vdash$", called the turnstyle, stands for provability.
It is read: "Assumptions H yield conclusion **G**"

Proof Obligation: Invariant Preservation (1)

- We collectively denote our set of constants by c

- We denote our set of axiomss by $A(c)$: $A_1(c), A_2(c), \ldots$

- We collectively denote our set of variables by v

- We denote our set of invariants by $I(c,v)$: $I_1(c,v), I_2(c,v), \ldots$

Proof Obligation: Invariant Preservation (2)

- We are given an event with before-after predicate $v' = E(c,v)$

- The following sequent expresses preservation of invariant $I_i(c,v)$:

$A(c),\; I(c,v) \;\vdash\; I_i(c, E(c,v))$	INV

- It says: $I_i(c, E(c,v))$ provable under hypotheses $A(c)$ and $I(c,v)$

- We have given the name INV to this proof obligation

Explanation of the Proof Obligation

$A(c),\ I(c,v)\ \vdash\ I_i(c,E(c,v))$	INV

- We assume that $A(c)$ as well as $I(c,v)$ hold just before the occurrence of the event represented by $v' = E(c,v)$

- Just after the occurrence, invariant $I_i(c,v)$ becomes $I_i(c,v')$, that is, $I_i(c,E(c,v))$

- The predicate $I_i(c,E(c,v))$ must then hold for $I_i(c,v)$ to be an invariant

Vertical Layout of Proof Obligations

- The proof obligation

$A(c),\ I(c,v)\ \vdash\ I_i(c,E(c,v))$	INV

can be re-written vertically as follows:

Axioms
Invariants
$\vdash$
Modified Invariant

$A(c)$ $I(c,v)$ $\vdash$ $I_i(c,E(c,v))$	INV

Back to our Example

- We have two events

ML_out
$n := n + 1$

ML_in
$n := n - 1$

- And two invariants

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

inv0_2: $n \leq d$

- Thus, we need to prove four proof obligations

Proof obligation for ML_out and **inv0_1**

ML_out
$n := n + 1$

$(n' = n + 1)$

Axiom **axm0_1**
Invariant **inv0_1**
Invariant **inv0_2**
$\vdash$
Modified Invariant **inv0_1**

$d \in \mathbb{N}$
$n \in \mathbb{N}$
$n \leq d$
$\vdash$
$n + 1 \in \mathbb{N}$

- This proof obligation is named: ML_out / **inv0_1** / INV

Proof obligation for ML_out and **inv0_2**

ML_out
$n := n + 1$

$(n' = n + 1)$

Axiom **axm0_1**
Invariant **inv0_1**
Invariant **inv0_2**
$\vdash$
Modified Invariant **inv0_2**

$d \in \mathbb{N}$
$n \in \mathbb{N}$
$n \leq d$
$\vdash$
$n + 1 \leq d$

- This proof obligation is named: ML_out / **inv0_2** / INV

Proof obligation for ML_in and **inv0_1**

ML_in
$n := n - 1$

$(n' = n - 1)$

Axiom **axm0_1**
Invariant **inv0_1**
Invariant **inv0_2**
$\vdash$
Modified Invariant **inv0_1**

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

- This proof obligation is named: ML_in / **inv0_1** / INV

Proof obligation for ML_in and **inv0_2**

ML_in
$n := n - 1$

$(n' = n - 1)$

Axiom **axm0_1**
Invariant **inv0_1**
Invariant **inv0_2**
$\vdash$
Modified Invariant **inv0_2**

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

- This proof obligation is named: ML_in / **inv0_2** / INV

Summary of Proof Obligations

ML_out / **inv0_1** / INV

$d \in \mathbb{N}$
$n \in \mathbb{N}$
$n \leq d$
$\vdash$
$n + 1 \in \mathbb{N}$

ML_out / **inv0_2** / INV

$d \in \mathbb{N}$
$n \in \mathbb{N}$
$n \leq d$
$\vdash$
$n + 1 \leq d$

ML_in / **inv0_1** / INV

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

ML_in / **inv0_2** / INV

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

Informal Proof of ML_out / **inv0_1** / INV

$$\begin{array}{l} d \in \mathbb{N} \\ n \in \mathbb{N} \\ n \leq d \\ \vdash \\ n+1 \in \mathbb{N} \end{array} \quad \xrightarrow[\text{hypotheses}]{\text{remove}} \quad \begin{array}{l} n \in \mathbb{N} \\ \vdash \\ n+1 \in \mathbb{N} \end{array} \quad \text{obvious}$$

- In the first step, we remove some irrelevant hypotheses
- In the second and final step, we accept the sequent as it is
- We have implicitly applied inference rules
- For rigorous reasoning we will make these rules explicit

Inference Rules

$$\frac{\mathbf{H_1} \vdash \mathbf{G_1} \quad \cdots \quad \mathbf{H_n} \vdash \mathbf{G_n}}{\mathbf{H} \vdash \mathbf{G}} \quad \text{RULE\_NAME}$$

- Above horizontal line: n sequents called antecedents ($n \geq 0$)
- Below horizontal line: exactly one sequent called consequent
- To prove the consequent, it is sufficient to prove the antecedents
- A rule with no antecedent ($n = 0$) is called an axiom

Inference Rule: Monotonicity of Hypotheses

- The rule that removes hypotheses can be stated as follows:

$$\frac{\mathbf{H} \vdash \mathbf{G}}{\mathbf{H}, \mathbf{H'} \vdash \mathbf{G}} \quad \text{MON}$$

- It expresses the monotonicity of the hypotheses

Some Arithmetic Rules of Inference

- The Second Peano Axiom

$$\frac{}{\mathbf{n} \in \mathbb{N} \vdash \mathbf{n}+1 \in \mathbb{N}} \quad \text{P2}$$

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

More Arithmetic Rules of Inference

- Axioms about ordering relations on the integers

$$\frac{}{\mathbf{n} < \mathbf{m} \;\vdash\; \mathbf{n}+1 \le \mathbf{m}}\ \text{INC}$$

$$\frac{}{\mathbf{n} \le \mathbf{m} \;\vdash\; \mathbf{n}-1 \le \mathbf{m}}\ \text{DEC}$$

Application of Inference Rules

- Consider again the 2nd Peano axiom:

$$\frac{}{\mathbf{n} \in \mathbb{N} \;\vdash\; \mathbf{n}+1 \in \mathbb{N}}\ \text{P2}$$

- It is a rule schema where **n** is called a meta-variable

- It can be applied to following sequent by matching $a+b$ with **n**:

$$a+b \in \mathbb{N} \;\vdash\; a+b+1 \in \mathbb{N}$$

Proofs

- A proof is a tree of sequents with axioms at the leaves.

- The rules applied to the leaves are axioms.

- Each sequent is labeled with (name of) proof rule applied to it.

- The sequent at the root of the tree is called the root sequent.

- The purpose of a proof is to establish the truth of its root sequent.

A Formal Proof of: ML_out / **inv0_1** / INV

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

- Proof requires only application of two rules: **MON** and **P2**

A Failed Proof Attempt: ML_out / **inv0_2** / INV

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

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

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

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

A Failed Proof Attempt: ML_in / **inv0_1** / INV

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

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

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

A Formal Proof of: ML_in / **inv0_2** / INV

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

$$\frac{}{\mathbf{n} \leq \mathbf{m} \;\vdash\; \mathbf{n}-1 \leq \mathbf{m}} \quad \text{DEC}$$

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

- 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	ML_in
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”

Proof Obligation: General Invariant Preservation

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

- Given an event with guard $G(c, v)$ and b-a predicate $v' = E(c, v)$

- We modify the Invariant Preservation PO as follows:

Axioms Invariants Guard of the event $\vdash$ Modified Invariant	$A(c)$ $I(c, v)$ $G(c, v)$ $\vdash$ $I_i(c, E(c, v))$	INV

A Formal Proof of: ML_out / **inv0_1** / INV

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

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

A Formal Proof of: ML_out / **inv0_2** / INV

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

- Now we can conclude the proof using rule INC

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

A Formal Proof of: ML_in / **inv0_1** / INV

$d \in \mathbb{N}$ $n \in \mathbb{N}$ $n \leq d$ $0 < n$ $\vdash$ $n - 1 \in \mathbb{N}$	MON	$0 < n$ $\vdash$ $n - 1 \in \mathbb{N}$	**P2'**

- Now we can conclude the proof using rule P2′

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

A Formal Proof of: ML_in / **inv0_2** / INV

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

- Again, the proof still works after the addition of a new assumption

Re-proving the Events: No Proofs Fail

$d \in \mathbb{N}$ $n \in \mathbb{N}$ $n \leq d$ $n < d$ $\vdash$ $n + 1 \in \mathbb{N}$	$d \in \mathbb{N}$ $n \in \mathbb{N}$ $n \leq d$ $n < d$ $\vdash$ $n + 1 \leq d$
$d \in \mathbb{N}$ $n \in \mathbb{N}$ $n \leq d$ $0 < n$ $\vdash$ $n - 1 \in \mathbb{N}$	$d \in \mathbb{N}$ $n \in \mathbb{N}$ $n \leq d$ $0 < n$ $\vdash$ $n - 1 \leq d$

Initialization

- Our system must be initialized (with no car in the island-bridge)

- The initialization event is never guarded

- It does not mention any variable on the right hand side of $:=$

-Its before-after predicate is just an after predicate

init $n := 0$	After predicate	$n' = 0$

Proof Obligation: Invariant Establishment

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

- Given an init event with after predicate $v' = K(c)$

- The Invariant Establishment PO is the following:

Axioms $\vdash$ Modified Invariant	$A(c)$ $\vdash$ $I_i(c, K(c))$	INV

Applying the Invariant Establishment PO

axm0_1 $\vdash$ Modified **inv0_1**	$d \in \mathbb{N}$ $\vdash$ $0 \in \mathbb{N}$	**inv0_1** / INV
axm0_1 $\vdash$ Modified **inv0_2**	$d \in \mathbb{N}$ $\vdash$ $0 \leq d$	**inv0_2** / INV

More Arithmetic Inference Rules

- First Peano Axiom

$$\frac{}{\vdash \; 0 \in \mathbb{N}} \quad \text{P1}$$

- Third Peano Axiom (slightly modified)

$$\frac{}{\mathbf{n} \in \mathbb{N} \;\vdash\; 0 \leq \mathbf{n}} \quad \text{P3}$$

Proofs of Invariant Establishment

$d \in \mathbb{N}$ $\vdash$ $0 \in \mathbb{N}$	MON	$\vdash$ $0 \in \mathbb{N}$	P1

$d \in \mathbb{N}$ $\vdash$ $0 \leq d$	P3

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

Applying the Deadlock Freedom PO

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

- This cannot be proved with the inference rules we have so far
- $n \leq d$ can be replaced by $n = d \ \vee \ n < d$
- We continue our proof by a case analysis:
  - case 1: $n = d$
  - case 2: $n < d$

Inference Rules for Disjunction

- Proof by case analysis

$$\frac{\mathbf{H}, \mathbf{P} \ \vdash \ \mathbf{R} \qquad \mathbf{H}, \mathbf{Q} \ \vdash \ \mathbf{R}}{\mathbf{H}, \ \mathbf{P} \vee \mathbf{Q} \ \vdash \ \mathbf{R}} \quad \text{OR\_L}$$

- Choice for proving a disjunctive goal

$$\frac{\mathbf{H} \ \vdash \ \mathbf{P}}{\mathbf{H} \ \vdash \ \mathbf{P} \vee \mathbf{Q}} \quad \text{OR\_R1} \qquad \frac{\mathbf{H} \ \vdash \ \mathbf{Q}}{\mathbf{H} \ \vdash \ \mathbf{P} \vee \mathbf{Q}} \quad \text{OR\_R2}$$

Proof of Deadlock Freedom

$$\begin{array}{l} d \in \mathbb{N} \\ n \in \mathbb{N} \\ n \leq d \\ \vdash \\ n < d \;\lor\; 0 < n \end{array} \quad \text{MON} \quad \begin{array}{l} n \leq d \\ \vdash \\ n < d \;\lor\; 0 < n \end{array} \quad \ldots$$

Proof of Deadlock Freedom (cont'd)

$$\begin{array}{l} n \leq d \\ \vdash \\ n < d \;\lor\; 0 < n \end{array} \quad \text{OR\_L} \left\{ \begin{array}{l} \begin{array}{l} n < d \\ \vdash \\ n < d \;\lor\; 0 < n \end{array} \quad \ldots \\ \\ \begin{array}{l} n = d \\ \vdash \\ n < d \;\lor\; 0 < n \end{array} \quad \ldots \end{array} \right.$$

Proof of Deadlock Freedom (cont'd)

$$\left\{ \begin{array}{l} \begin{array}{l} n < d \\ \vdash \\ n < d \;\lor\; 0 < n \end{array} \quad \text{OR\_R1} \quad n < d \;\vdash\; n < d \quad ? \\ \\ \begin{array}{l} n = d \\ \vdash \\ n < d \;\lor\; 0 < n \end{array} \quad ? \end{array} \right.$$

- The first **?** seems to be obvious
- The second **?** can be (partially) solved by applying the equality

More Inference Rules: Identity and Equality

- The identity axiom (conclusion holds by hypothesis)

$$\frac{}{\mathbf{P} \vdash \mathbf{P}} \; \mathbf{HYP}$$

- Rewriting an equality (**EQ_LR**) and reflexivity of equality (**EQL**)

$$\frac{\mathbf{H(F)},\ \mathbf{E} = \mathbf{F} \;\vdash\; \mathbf{P(F)}}{\mathbf{H(E)},\ \mathbf{E} = \mathbf{F} \;\vdash\; \mathbf{P(E)}} \; \mathbf{EQ\_LR} \qquad \frac{}{\vdash\; \mathbf{E} = \mathbf{E}} \; \mathbf{EQL}$$

Proof of Deadlock Freedom (end)

$$\left\{\begin{array}{l} \boxed{\begin{array}{l} n < d \\ \vdash \\ n < d \ \lor \ 0 < n \end{array}} \ \text{OR\_R1} \ \boxed{n < d \ \vdash \ n < d} \ \text{HYP} \\ \\ \boxed{\begin{array}{l} n = d \\ \vdash \\ n < d \ \lor \ 0 < n \end{array}} \ \text{EQ\_LR} \ \boxed{\vdash \ d < d \ \lor \ 0 < d} \ \text{OR\_R2} \\ \\ \qquad\qquad\qquad\qquad \text{OR\_R2} \ \boxed{\vdash \ 0 < d} \ \ ? \end{array}\right.$$

- We still have a problem: d must be positive!

Overview
The Requirement Document
Formal Models
Initial Model
First Refinement
Second Refinement
Third Refinement

Adding the Forgotten Axiom

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

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

Overview
The Requirement Document
Formal Models
Initial Model
First Refinement
Second Refinement
Third Refinement

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

Overview
The Requirement Document
Formal Models
Initial Model
First Refinement
Second Refinement
Third Refinement

Proof Obligations for Initial Model

- We have seen three kinds of proof obligations:

  - The Invariant Establishment PO: INV

  - The Invariant Preservation PO: INV

  - The Deadlock Freedom PO (optional): DLF

Proof Obligations for Initial Model (cont'd)

Axioms $\vdash$ Modified Invariant	INV

Axioms Invariants Guard of the event $\vdash$ Modified Invariant	INV

Axiom Invariants $\vdash$ Disjunction of the guards	DLF

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: $n \leq d$

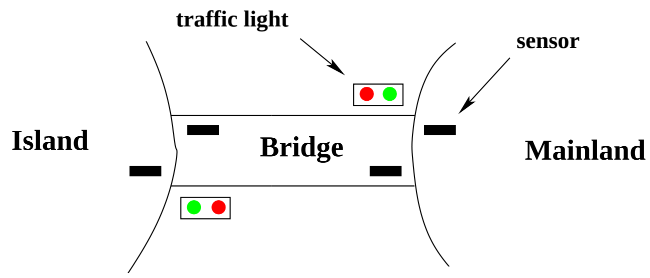
init
  $n := 0$

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

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

Outline

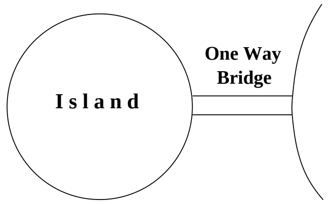
Reminder of the physical system

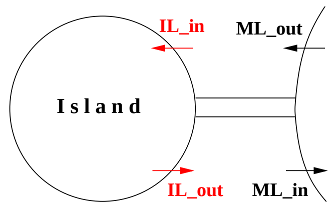


First Refinement: Introducing a One-Way Bridge

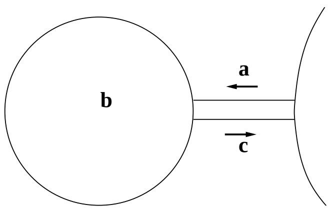
- We go down with our parachute
- Our view of the system gets more accurate
- We introduce the bridge and separate it from the island
- We refine the state and the events
- We also add two new events: IL_in and IL_out
- We are focusing on FUN-3: one-way bridge

First Refinement: Introducing a one Way Bridge





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

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

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

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

Refining the State: Introducing New Invariants

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

$$a + b + c = n$$

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

$$a = 0 \ \lor \ c = 0$$

Refining the State: Summary

constants: d

variables: a, b, c

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

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

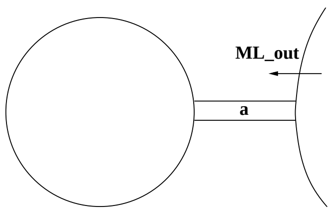
inv1_3: $c \in \mathbb{N}$

inv1_4: $a + b + c = n$

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

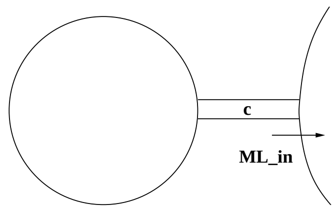
- Invariants **inv1_1** to **inv1_5** are called the concrete invariants
- **inv1_4** glues the abstract state, n, to the concrete state, a, b, c

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

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:

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

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

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

Intuition about refinement (2)

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

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

- Same remarks as in the previous slide

- But this has to be confirmed by well-defined proof obligations

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 $A(c)$

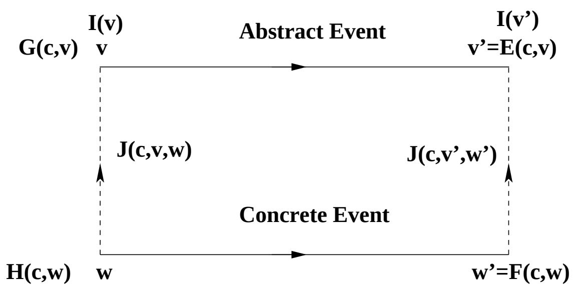
Abstract variables v with abstract invariant $I(c, v)$

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

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

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

Proof Obligations for Guard Strengthening

- ML_out / GRD

- ML_in / GRD

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

$d \in \mathbb{N}$
$0 < d$
$n \in \mathbb{N}$
$n \leq d$
$a \in \mathbb{N}$
$b \in \mathbb{N}$
$c \in \mathbb{N}$
$a + b + c = n$
$a = 0 \;\vee\; c = 0$
$a + b < d$
$c = 0$
$\vdash$
$n < d$

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

$d \in \mathbb{N}$
$0 < d$
$n \in \mathbb{N}$
$n \leq d$
$a \in \mathbb{N}$
$b \in \mathbb{N}$
$c \in \mathbb{N}$
$a + b + c = n$
$a = 0 \;\vee\; c = 0$
$a + b < d$
$c = 0$
$\vdash$
$n < d$

MON

$a + b + c = n$
$a + b < d$
$c = 0$
$\vdash$
$n < d$

EQ_LR

$a + b + 0 = n$
$a + b < d$
$\vdash$
$n < d$

ARITH ...

...

$a + b = n$
$a + b < d$
$\vdash$
$n < d$

EQ_LR

$n < d$
$\vdash$
$n < d$

HYP

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

Applying Guard Strengthening to Event ML_in

axm0_1
axm0_2
inv0_1
inv0_2
inv1_1
inv1_2
inv1_3
inv1_4
inv1_5
Concrete guard of ML_in
$\vdash$
Abstract guard of ML_in

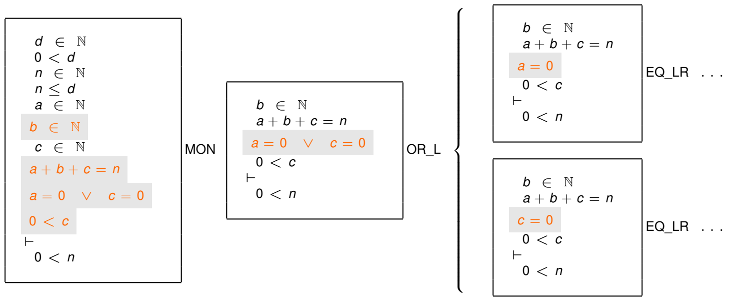
$d \in \mathbb{N}$
$0 < d$
$n \in \mathbb{N}$
$n \leq d$
$a \in \mathbb{N}$
$b \in \mathbb{N}$
$c \in \mathbb{N}$
$a + b + c = n$
$a = 0 \;\vee\; c = 0$
$0 < c$
$\vdash$
$0 < n$

ML_in / GRD

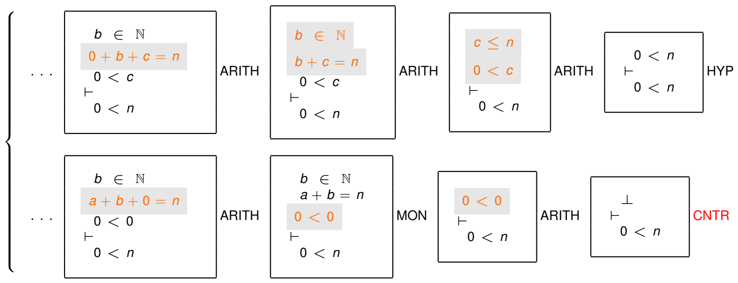
(abstract-)ML_in
when
$0 < n$
then
$n := n - 1$
end

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

Proof of ML_in / GRD



Proof of ML_in / GRD



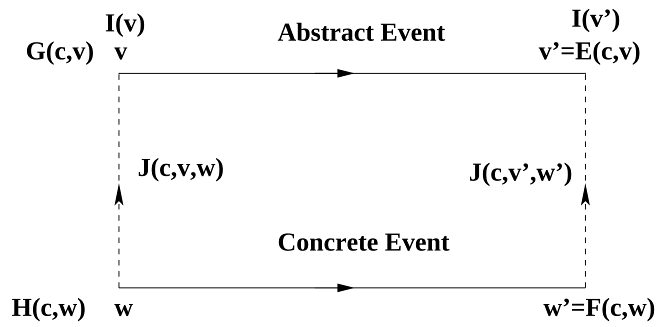
An Additional Rule: the Contradiction Rule

- In the previous proof, we have used and additional inference rule

- It says that a false hypothesis entails any goal

$$\frac{}{\bot \;\vdash\; \mathbf{P}} \quad \text{CNTR}$$

Correctness of Invariant Refinement



Proof Obligation: Invariant Refinement

Axioms Abstract Invariants Concrete Invariants Concrete Guards $\vdash$ 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

Overview of Proof Obligations

- ML_out / GRD done
- ML_in / GRD done
- ML_out / **inv1_4** / INV
- ML_out / **inv1_5** / INV
- ML_in / **inv1_4** / INV
- ML_in / **inv1_5** / INV

Applying Invariant Refinement 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 $\vdash$ Modified Invariant **inv1_4**	$d \in \mathbb{N}$ $0 < d$ $n \in \mathbb{N}$ $n \leq d$ $a \in \mathbb{N}$ $b \in \mathbb{N}$ $c \in \mathbb{N}$ $a+b+c=n$ $a=0 \;\lor\; c=0$ $a+b<d$ $c=0$ $\vdash$ $a+1+b+c=n+1$	ML_out / **inv1_4** / INV

(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 / inv1_4 / INV

$d \in \mathbb{N}$
$0 < d$
$n \in \mathbb{N}$
$n \leq d$
$a \in \mathbb{N}$
$b \in \mathbb{N}$
$c \in \mathbb{N}$
$a+b+c=n$
$a=0 \;\lor\; c=0$
$a+b<d$
$c=0$
$\vdash$
$a+1+b+c=n+1$

MON

$a+b+c=n$
$\vdash$
$a+1+b+c=n+1$

ARITH ...

...

$a+b+c=n$
$\vdash$
$a+b+c+1=n+1$

EQ_LR

$\vdash \; n+1=n+1$

EQL

Applying Invariant Refinement 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
$\vdash$
Modified Invariant **inv1_5**

$$\begin{array}{l} d \in \mathbb{N} \\ 0 < d \\ n \in \mathbb{N} \\ n \leq d \\ a \in \mathbb{N} \\ b \in \mathbb{N} \\ c \in \mathbb{N} \\ a+b+c=n \\ a=0 \;\vee\; c=0 \\ a+b<d \\ c=0 \\ \vdash \\ a+1=0 \;\vee\; c=0 \end{array}$$

ML_out / **inv1_5** / INV

(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 / inv1_5 / INV

$$\begin{array}{l} d \in \mathbb{N} \\ 0 < d \\ n \in \mathbb{N} \\ n \leq d \\ a \in \mathbb{N} \\ b \in \mathbb{N} \\ c \in \mathbb{N} \\ a+b+c=n \\ a=0 \;\vee\; c=0 \\ a+b<d \\ c=0 \\ \vdash \\ a+1=0 \;\vee\; c=0 \end{array}$$

MON

$$\begin{array}{l} c=0 \\ \vdash \\ a+1=0 \;\vee\; c=0 \end{array}$$

OR_R2

$$\begin{array}{l} c=0 \\ \vdash \\ c=0 \end{array}$$

HYP

Applying Invariant Refinement to Event ML_in

axm0_1
axm0_2
inv0_1
inv0_2
inv1_1
inv1_2
inv1_3
inv1_4
inv1_5
Concrete guards of ML_in
$\vdash$
Modified Invariant **inv1_4**

$$\begin{array}{l} d \in \mathbb{N} \\ 0 < d \\ n \in \mathbb{N} \\ n \leq d \\ a \in \mathbb{N} \\ b \in \mathbb{N} \\ c \in \mathbb{N} \\ a+b+c=n \\ a=0 \;\vee\; c=0 \\ 0<c \\ \vdash \\ a+b+c-1=n-1 \end{array}$$

ML_in / **inv1_4** / INV

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

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

Proof of ML_in / inv1_4 / INV

$$\begin{array}{l} d \in \mathbb{N} \\ 0 < d \\ n \in \mathbb{N} \\ n \leq d \\ a \in \mathbb{N} \\ b \in \mathbb{N} \\ c \in \mathbb{N} \\ a+b+c=n \\ a=0 \;\vee\; c=0 \\ 0<c \\ \vdash \\ a+b+c-1=n-1 \end{array}$$

MON

$$\begin{array}{l} a+b+c=n \\ \vdash \\ a+b+c-1=n-1 \end{array}$$

EQ_LR

$$\vdash \; n-1=n-1$$

EQL

Applying Invariant Refinement to Event ML_in

axm0_1
axm0_2
inv0_1
inv0_2
inv1_1
inv1_2
inv1_3
inv1_4
inv1_5
Concrete guards of ML_in
$\vdash$
Modified Invariant **inv1_5**

$$\begin{array}{l} d \in \mathbb{N} \\ 0 < d \\ n \in \mathbb{N} \\ n \leq d \\ a \in \mathbb{N} \\ b \in \mathbb{N} \\ c \in \mathbb{N} \\ a+b+c=n \\ a=0 \;\lor\; c=0 \\ 0 < c \\ \vdash \\ a=0 \;\lor\; c-1=0 \end{array}$$

ML_in / **inv1_5** / INV

```
(abstract-)ML_in
  when
    0 < n
  then
    n := n − 1
  end
```

```
(concrete-)ML_in
  when
    0 < c
  then
    c := c − 1
  end
```


Proof of ML_in / inv1_5 / INV

$$\begin{array}{l} d \in \mathbb{N} \\ 0 < d \\ n \in \mathbb{N} \\ n \leq d \\ a \in \mathbb{N} \\ b \in \mathbb{N} \\ c \in \mathbb{N} \\ a+b+c=n \\ a=0 \;\lor\; c=0 \\ 0 < c \\ \vdash \\ a=0 \;\lor\; c-1=0 \end{array}
\quad \text{MON} \quad
\begin{array}{l} a=0 \;\lor\; c=0 \\ 0 < c \\ \vdash \\ a=0 \;\lor\; c-1=0 \end{array}
\quad \text{OR_L} \quad
\left\{ \begin{array}{ll} \begin{array}{l} a=0 \\ 0<c \\ \vdash \\ a=0 \;\lor\; c-1=0 \end{array} & \text{MON} \;\ldots \\ \begin{array}{l} c=0 \\ 0<c \\ \vdash \\ a=0 \;\lor\; c-1=0 \end{array} & \text{EQ_LR} \;\ldots \end{array} \right.$$

Proof of ML_in / inv1_5 / INV

$$\begin{array}{l} d \in \mathbb{N} \\ 0 < d \\ n \in \mathbb{N} \\ n \leq d \\ a \in \mathbb{N} \\ b \in \mathbb{N} \\ c \in \mathbb{N} \\ a+b+c=n \\ a=0 \;\lor\; c=0 \\ 0 < c \\ \vdash \\ a=0 \;\lor\; c-1=0 \end{array}
\quad \text{MON} \quad
\begin{array}{l} a=0 \;\lor\; c=0 \\ 0 < c \\ \vdash \\ a=0 \;\lor\; c-1=0 \end{array}
\quad \text{OR_L} \quad
\left\{ \begin{array}{ll} \begin{array}{l} a=0 \\ 0<c \\ \vdash \\ a=0 \;\lor\; c-1=0 \end{array} & \text{MON} \;\ldots \\ \begin{array}{l} c=0 \\ 0<c \\ \vdash \\ a=0 \;\lor\; c-1=0 \end{array} & \text{EQ_LR} \;\ldots \end{array} \right.$$

$$\left\{ \begin{array}{l} \ldots \quad \begin{array}{l} a=0 \\ \vdash \\ a=0 \;\lor\; c-1=0 \end{array} \quad \text{OR\_R1} \quad a=0 \;\vdash\; a=0 \quad \text{HYP} \\ \ldots \quad 0<0 \;\vdash\; a=0 \;\lor\; -1=0 \quad \text{ARITH} \quad \bot \;\vdash\; a=0 \;\lor\; -1=0 \quad \text{CNTR} \end{array} \right.$$

Refining the Initialization Event init

- Concrete initialization

```
init
  a := 0
  b := 0
  c := 0
```

- Corresponding after predicate

$$a' = 0 \;\land\; b' = 0 \;\land\; c' = 0$$

Proof Obligation: Initialization Refinement

Constants c with axioms $A(c)$

Concrete invariant $J(c, v, w)$

Abstract initialization with after predicate $v' = K(c)$

Concrete initialization with after predicate $w' = L(c)$

Axioms $\vdash$ Modified concrete invariants	$A(c)$ $\vdash$ $J_j(c, K(c), L(c))$	INV

Overview of Proof Obligations

- ML_out / GRD done
- ML_in / GRD done
- ML_out / **inv1_4** / INV done
- ML_out / **inv1_5** / INV done
- ML_in / **inv1_4** / INV done
- ML_in / **inv1_5** / INV done
- **inv1_4** / INV
- **inv1_5** / INV

Applying the Initialization Refinement PO

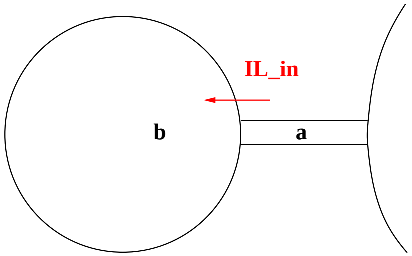
axm0_1 **axm0_2** $\vdash$ Modified concrete invariant **inv1_4** $(a + b + c = n)$	$d \in \mathbb{N}$ $d > 0$ $\vdash$ $0 + 0 + 0 = 0$

axm0_1 **axm0_2** $\vdash$ Modified concrete invariant **inv1_5** $(a = 0 \ \lor \ c = 0)$	$d \in \mathbb{N}$ $d > 0$ $\vdash$ $0 = 0 \ \lor \ 0 = 0$

Adding New Events

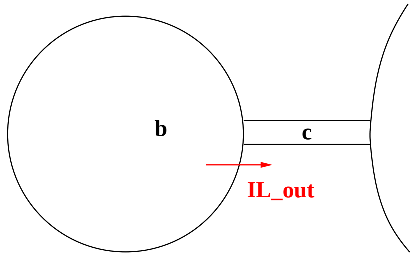
- new events add transitions that have no abstract counterpart
- can be seen as a kind of internal steps (w.r.t. abstract model)
- can only be seen by an observer who is “zooming in”
- temporal refinement: refined model has a finer time granularity

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

Several Actions Done Together

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

Before-after predicates

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

$$a' = a \;\wedge\; b' = b - 1 \;\wedge\; c' = c + 1$$

The empty assignment: skip

The before-after predicate of skip in the initial model

$$n' = n$$

The before-after predicate of skip in the first refinement

$$a' = a \;\wedge\; b' = b \;\wedge\; c' = c$$

The guard of the skip event is true.

Refinement Proof Obligations for New Events

(1) A new event must refine an implicit event, made of a skip action

- Guard strengthening is trivial
- Need to prove invariant refinement

(2) The new events must not diverge

- To prove this we have to exhibit a variant
- The variant yields a natural number (could be more complex)
- Each new event must decrease this variant

Overview of Proof Obligations

- ML_out / GRD done
- ML_in / GRD done
- ML_out / **inv1_4** / INV done
- ML_out / **inv1_5** / INV done
- ML_in / **inv1_4** / INV done
- ML_in / **inv1_5** / INV done
- **inv1_4** / INV done
- **inv1_5** / INV done
- IL_in / **inv1_4** / INV
- IL_in / **inv1_5** / INV
- IL_out / **inv1_4** / INV
- IL_out / **inv1_5** / INV

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
⊢
Modified Invariant **inv1_4**

$$
\begin{array}{l}
d \in \mathbb{N} \\
0 < d \\
n \in \mathbb{N} \\
n \leq d \\
a \in \mathbb{N} \\
b \in \mathbb{N} \\
c \in \mathbb{N} \\
a + b + c = n \\
a = 0 \;\vee\; c = 0 \\
0 < a \\
\vdash \\
a - 1 + b + 1 + c = n
\end{array}
$$

IL_in / **inv1_4** / INV

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


Proof of IL_in / inv1_4 / INV

$$
\begin{array}{l}
d \in \mathbb{N} \\
0 < d \\
n \in \mathbb{N} \\
n \leq d \\
a \in \mathbb{N} \\
b \in \mathbb{N} \\
c \in \mathbb{N} \\
a + b + c = n \\
a = 0 \;\vee\; c = 0 \\
0 < a \\
\vdash \\
a - 1 + b + 1 + c = n
\end{array}
$$

MON

$$
\begin{array}{l}
a + b + c = n \\
\vdash \\
a - 1 + b + 1 + c = n
\end{array}
$$

ARITH

$$
\begin{array}{l}
a + b + c = n \\
\vdash \\
a + b + c = n
\end{array}
$$

HYP

Event IL_in Refines skip (2)

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_5**

$$\begin{array}{l} d \in \mathbb{N} \\ 0 < d \\ n \in \mathbb{N} \\ n \leq d \\ a \in \mathbb{N} \\ b \in \mathbb{N} \\ c \in \mathbb{N} \\ a+b+c=n \\ a=0 \quad \vee \quad c=0 \\ 0 < a \\ \vdash \\ a-1=0 \quad \vee \quad c=0 \end{array}$$

IL_in / **inv1_5** / INV

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

Proof of IL_in / inv1_5 / INV

$$\begin{array}{l} d \in \mathbb{N} \\ 0 < d \\ n \in \mathbb{N} \\ n \leq d \\ a \in \mathbb{N} \\ b \in \mathbb{N} \\ c \in \mathbb{N} \\ a+b+c=n \\ a=0 \quad \vee \quad c=0 \\ 0 < a \\ \vdash \\ a-1=0 \quad \vee \quad c=0 \end{array}$$

MON

$$\begin{array}{l} a=0 \quad \vee \quad c=0 \\ 0 < a \\ \vdash \\ a-1=0 \quad \vee \quad c=0 \end{array}$$

OR_L $\cdots$

Proof of IL_in / inv1_5 / INV

$$\begin{array}{l} d \in \mathbb{N} \\ 0 < d \\ n \in \mathbb{N} \\ n \leq d \\ a \in \mathbb{N} \\ b \in \mathbb{N} \\ c \in \mathbb{N} \\ a+b+c=n \\ a=0 \quad \vee \quad c=0 \\ 0 < a \\ \vdash \\ a-1=0 \quad \vee \quad c=0 \end{array}$$

MON

$$\begin{array}{l} a=0 \quad \vee \quad c=0 \\ 0 < a \\ \vdash \\ a-1=0 \quad \vee \quad c=0 \end{array}$$

OR_L $\cdots$

$\cdots$

$$\begin{array}{l} a=0 \\ 0 < a \\ \vdash \\ a-1=0 \quad \vee \quad c=0 \end{array}$$

EQ_LR

$$\begin{array}{l} 0 < 0 \\ \vdash \\ -1=0 \quad \vee \quad c=0 \end{array}$$

ARITH

$$\begin{array}{l} \perp \\ \vdash \\ -1=0 \quad \vee \quad c=0 \end{array}$$

CNTR

$$\begin{array}{l} c=0 \\ 0 < a \\ \vdash \\ a-1=0 \quad \vee \quad c=0 \end{array}$$

MON

$$\begin{array}{l} c=0 \\ \vdash \\ a-1=0 \quad \vee \quad c=0 \end{array}$$

OR_R2

$$c=0 \quad \vdash \quad c=0$$

HYP

Proof Obligation: Convergence of New Events (1)

Axioms $A(c)$, invariants $I(c,v)$, concrete invariant $J(c,v,w)$
New event with guard $H(c,w)$
Variant $V(c,w)$

Axioms Abstract invariants Concrete invariants Concrete guard of a new event $\vdash$ Variant $\in \mathbb{N}$	$A(c)$ $I(c,v)$ $J(c,v,w)$ $H(c,w)$ $\vdash$ $V(c,w) \in \mathbb{N}$	NAT

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

Proposed Variant

variant_1: $2 * a + b$

- Weighted sum of a and b

Overview of Proof Obligations

– ML_out / GRD done

– ML_in / GRD done

– ML_out / **inv1_4** / INV done

– ML_out / **inv1_5** / INV done

– ML_in / **inv1_4** / INV done

– ML_in / **inv1_5** / INV done

– **inv1_4** / INV done

– **inv1_5** / INV done

– IL_in / **inv1_4** / INV done

– IL_in / **inv1_5** / INV done

– IL_out / **inv1_4** / INV done

– IL_out / **inv1_5** / INV done

– IL_in / NAT

– IL_out / NAT

– IL_in / VAR

– IL_out / VAR

Decreasing of the Variant by Event IL_in

axm0_1 **axm0_2** **inv0_1** **inv0_2** **inv1_1** **inv1_2** **inv1_3** **inv1_4** **inv1_5** Concrete guard of IL_in $\vdash$ Modified variant $<$ Variant	$d \in \mathbb{N}$ $0 < d$ $n \in \mathbb{N}$ $n \leq d$ $a \in \mathbb{N}$ $b \in \mathbb{N}$ $c \in \mathbb{N}$ $a + b + c = n$ $a = 0 \;\lor\; c = 0$ $0 < a$ $\vdash$ $2 * (a - 1) + b + 1 < 2 * a + b$	IL_in / VAR

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

Decreasing of the Variant by Event IL_out

axm0_1
axm0_2
inv0_1
inv0_2
inv1_1
inv1_2
inv1_3
inv1_4
inv1_5
Concrete guards of IL_out
$\vdash$
Modified variant $<$ Variant

$$
\begin{array}{l}
d \in \mathbb{N} \\
0 < d \\
n \in \mathbb{N} \\
n \leq d \\
a \in \mathbb{N} \\
b \in \mathbb{N} \\
c \in \mathbb{N} \\
a + b + c = n \\
a = 0 \quad \lor \quad c = 0 \\
0 < b \\
a = 0 \\
\vdash \\
2 * a + b - 1 < 2 * a + b
\end{array}
$$

IL_out / VAR

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


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:

$$
\begin{array}{l|c}
\begin{array}{l}
A(c) \\
I(c, v) \\
J(c, v, w) \\
G_1(c, v) \lor \ldots \lor G_m(c, v) \\
\vdash \\
H_1(c, w) \lor \ldots \lor H_n(c, w)
\end{array} & \text{DLF}
\end{array}
$$

Applying the Relative Deadlock Freedom PO

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
```

More Inference Rules: Negation and Conjunction

$$\frac{\mathbf{H}, \neg\mathbf{P} \;\vdash\; \mathbf{Q}}{\mathbf{H} \;\vdash\; \mathbf{P} \lor \mathbf{Q}} \quad \text{NEG}$$

$$\frac{\mathbf{H}, \mathbf{P}, \mathbf{Q} \;\vdash\; \mathbf{R}}{\mathbf{H},\; \mathbf{P} \land \mathbf{Q} \;\vdash\; \mathbf{R}} \quad \text{AND\_L}$$

$$\frac{\mathbf{H} \;\vdash\; \mathbf{P} \qquad \mathbf{H} \;\vdash\; \mathbf{Q}}{\mathbf{H} \;\vdash\; \mathbf{P} \land \mathbf{Q}} \quad \text{AND\_R}$$

Proof of DLF

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

...

Proof of DLF

$$\ldots \quad \begin{array}{l} a \in \mathbb{N} \\ a+b+c=n \\ n>0 \;\lor\; n<d \\ c=0 \\ \vdash \\ (a+b<d \;\land\; c=0) \;\lor \\ a>0 \;\lor \\ (b>0 \;\land\; a=0) \end{array} \quad \text{EQ\_LR} \quad \begin{array}{l} a \in \mathbb{N} \\ a+b+0=n \\ n>0 \;\lor\; n<d \\ \vdash \\ (a+b<d \;\land\; 0=0) \;\lor \\ a>0 \;\lor \\ (b>0 \;\land\; a=0) \end{array} \quad \text{NEG}$$

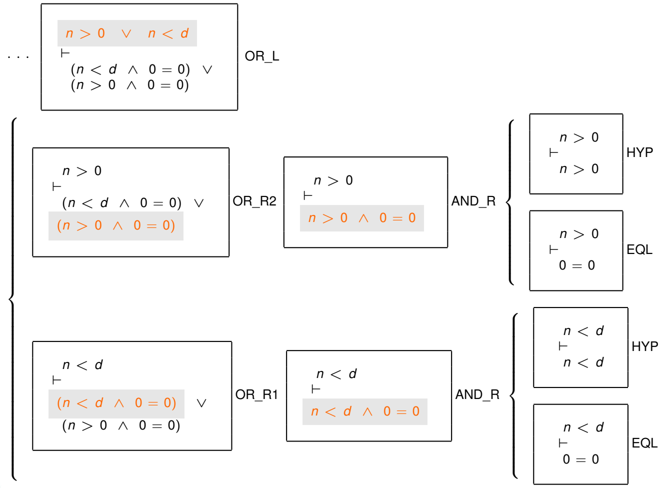
$$\begin{array}{l} a \in \mathbb{N} \\ a+b+0=n \\ n>0 \;\lor\; n<d \\ \neg\,(a>0) \\ \vdash \\ (a+b<d \;\land\; 0=0) \;\lor \\ (b>0 \;\land\; a=0) \end{array} \quad \text{ARITH} \quad \ldots$$

Proof of DLF (cont'd)

$$\ldots \quad \begin{array}{l} a+b+0=n \\ n>0 \;\lor\; n<d \\ a=0 \\ \vdash \\ (a+b<d \;\land\; 0=0) \;\lor \\ (b>0 \;\land\; a=0) \end{array} \quad \text{EQ\_LR} \quad \begin{array}{l} 0+b+0=n \\ n>0 \;\lor\; n<d \\ \vdash \\ (0+b<d \;\land\; 0=0) \;\lor \\ (b>0 \;\land\; 0=0) \end{array} \quad \text{ARITH}$$

$$\begin{array}{l} b=n \\ n>0 \;\lor\; n<d \\ \vdash \\ (b<d \;\land\; 0=0) \;\lor \\ (b>0 \;\land\; 0=0) \end{array} \quad \text{EQ\_LR} \quad \ldots$$

Proof of DLF (cont'd)



Overview of Proof Obligations

– ML_out / GRD done
– ML_in / GRD done
– ML_out / **inv1_4** / INV done
– ML_out / **inv1_5** / INV done
– ML_in / **inv1_4** / INV done
– ML_in / **inv1_5** / INV done
– **inv1_4** / INV done
– **inv1_5** / INV done
– IL_in / **inv1_4** / INV done
– IL_in / **inv1_5** / INV done
– IL_out / **inv1_4** / INV done
– IL_out / **inv1_5** / INV done

– IL_in / NAT done
– IL_out / NAT done
– IL_in / VAR done
– IL_out / VAR done
– DLF done

Summary of Refinement POs

- For old events:
  - Strengthening of guards: GRD
  - Concrete invariant preservation: INV
- For new events:
  - Refining the implicit skip event: INV
  - Absence of divergence: NAT and VAR
- For all events:
  - Relative deadlock freedom: DLF

Proof Obligations for Refinement (1/2)

Axioms Abstract invariants Concrete invariants Concrete guards $\vdash$ Abstract guard	GRD

Axioms Abstract invariants Concrete invariants Concrete guard $\vdash$ Modified concrete invariant	INV

Axioms $\vdash$ Modified concrete invariant	INV

Proof Obligations for Refinement (2/2)

Axioms Abstract invariants Concrete invariants Concrete guards of a new event $\vdash$ Variant $\in \mathbb{N}$	NAT

Axioms Abstract invariants Concrete invariants Concrete guards of a new event $\vdash$ Modified variant $<$ Variant	VAR

Axioms Abstract invariants Concrete invariants Disjunction of abstract events guards $\vdash$ Disjunction of concrete events guards	DLF

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

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
```

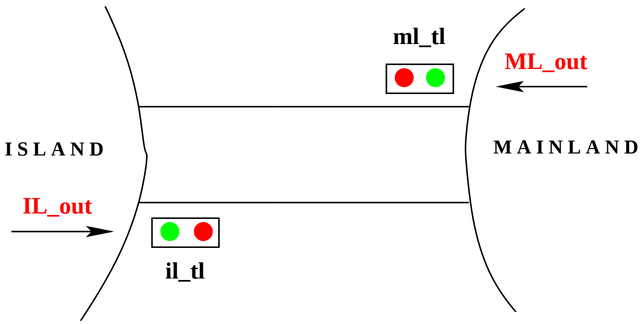
```
ML_out
  when
    a + b < d
    c = 0
  then
    a := a + 1
  end
```

```
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
```


Outline

Second Refinement: Introducing Traffic Lights



Extending the Constants

set: $COLOR$

constants: $red, green$

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

axm2_2: $green \neq red$

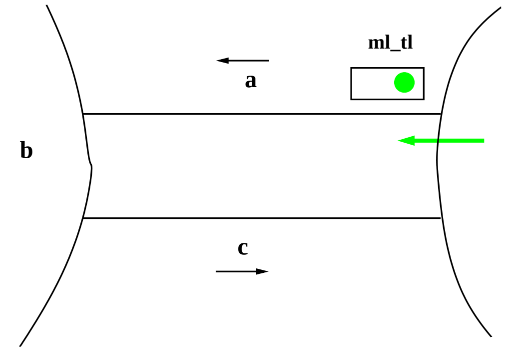
Extending the Variables

$il\_tl \in COLOR$

$ml\_tl \in COLOR$

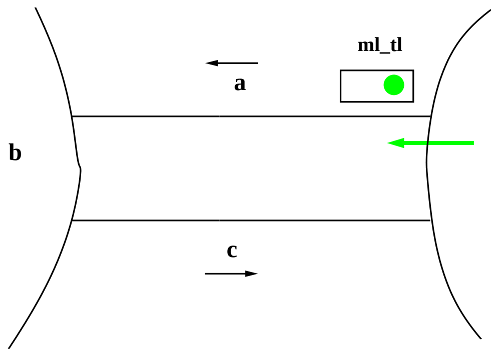
Remark: Events IL_in and ML_in are not modified in this refinement

Extending the Invariant (1)



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

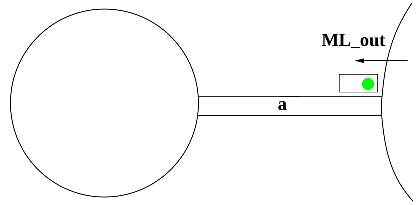
Extending the Invariant (1)



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

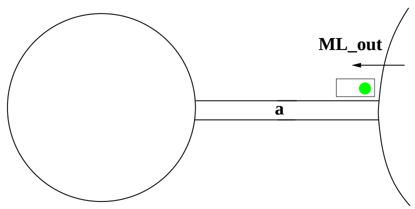
$$ml\_tl = \text{green} \;\Rightarrow\; c = 0 \;\wedge\; a + b < d$$

Refining Event ML_out



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

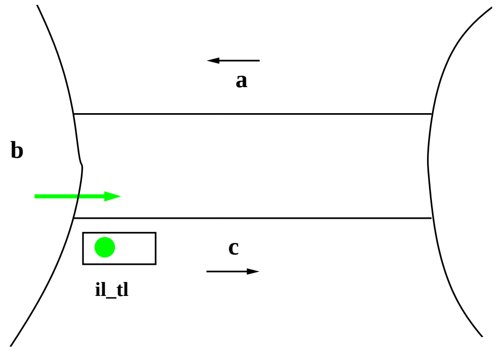
Refining Event ML_out



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

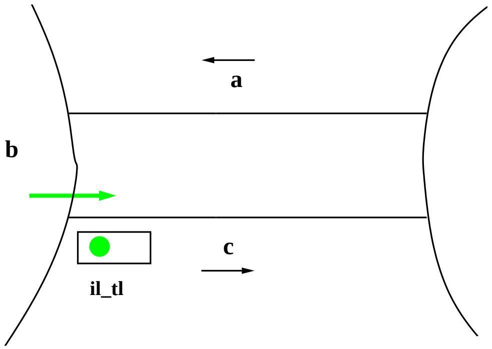
(concrete_)ML_out
when
$ml\_tl = \text{green}$
then
$a := a + 1$
end

Extending the Invariant (2)



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

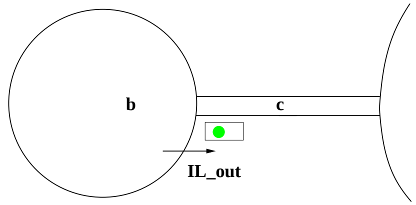
Extending the Invariant (2)



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

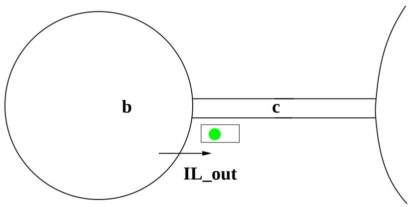
$$il\_tl = \text{green} \;\Rightarrow\; a = 0 \;\wedge\; 0 < b$$

Refining Event IL_out



(abstract_)IL_out
when
$a = 0$
$0 < b$
then
$b, c := b - 1, c + 1$
end

Refining Event IL_out



(abstract_)IL_out
when
$a = 0$
$0 < b$
then
$b, c := b - 1, c + 1$
end

(concrete_)IL_out
when
$il\_tl = \text{green}$
then
$b, c := b - 1, c + 1$
end

New Events ML_tl_green and IL_tl_green

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

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

- Turning lights to green when proper conditions hold

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

Summary of Old Events (so far)

ML_out
  when
    $ml\_tl = \text{green}$
  then
    $a := a + 1$
  end

IL_out
  when
    $il\_tl = \text{green}$
  then
    $b := b - 1$
    $c := c + 1$
  end

Events ML_in and IL_ in are unchanged

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

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

Superposition

variables: a, b, c, ml_tl, il_tl

- Variables a, b, and c were present in the previous refinement

- Variables ml_tl and il_tl are superposed to a, b, and c

- We have thus to extend rule INV

Superposition: Introduction of a new Rule

Abstract_Event
  when
    $G(c, u, v)$
  then
    $u := E(c, u, v)$
    $v := M(c, u, v)$
  end

Concrete_Event
  when
    $H(c, v, w)$
  then
    $v := N(c, v, w)$
    $w := F(c, v, w)$
  end

Axioms Abstract invariants Concrete invariants Concrete guards $\Rightarrow$ Same actions on common variables	$A(c)$ $I(c, u, v)$ $J(c, u, v, w)$ $H(c, v, w)$ $\Rightarrow$ $M(c, u, v) = N(c, v, w)$	SIM

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

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

Proving the Refinement of the Four Old Events

ML_out **when** $c = 0$ $a + b < d$ **then** $a := a + 1$ **end**	IL_out **when** $a = 0$ $0 < b$ **then** $b := b - 1$ $c := c + 1$ **end**	ML_in **when** $0 < c$ **then** $c := c - 1$ **end**	IL_in **when** $0 < a$ **then** $a := a - 1$ $b := b + 1$ **end**

ML_out **when** $ml\_tl = \text{green}$ **then** $a := a + 1$ **end**	IL_out **when** $il\_tl = \text{green}$ **then** $b := b - 1$ $c := c + 1$ **end**	ML_in **when** $0 < c$ **then** $c := c - 1$ **end**	IL_in **when** $0 < a$ **then** $a := a - 1$ $b := b + 1$ **end**

- SIM is completely trivial since the actions are the same

Proving the Refinement of the Four Old Events

ML_out **when** $c = 0$ $a + b < d$ **then** $a := a + 1$ **end**	IL_out **when** $a = 0$ $0 < b$ **then** $b := b - 1$ $c := c + 1$ **end**	ML_in **when** $0 < c$ **then** $c := c - 1$ **end**	IL_in **when** $0 < a$ **then** $a := a - 1$ $b := b + 1$ **end**

ML_out **when** $ml\_tl = \text{green}$ **then** $a := a + 1$ **end**	IL_out **when** $il\_tl = \text{green}$ **then** $b := b - 1$ $c := c + 1$ **end**	ML_in **when** $0 < c$ **then** $c := c - 1$ **end**	IL_in **when** $0 < a$ **then** $a := a - 1$ $b := b + 1$ **end**

- GRD is also trivial

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

ML_out **when** $c = 0$ $a + b < d$ **then** $a := a + 1$ **end**	IL_out **when** $a = 0$ $0 < b$ **then** $b := b - 1$ $c := c + 1$ **end**	ML_in **when** $0 < c$ **then** $c := c - 1$ **end**	IL_in **when** $0 < a$ **then** $a := a - 1$ $b := b + 1$ **end**

ML_out **when** $ml\_tl = \text{green}$ **then** $a := a + 1$ **end**	IL_out **when** $il\_tl = \text{green}$ **then** $b := b - 1$ $c := c + 1$ **end**	ML_in **when** $0 < c$ **then** $c := c - 1$ **end**	IL_in **when** $0 < a$ **then** $a := a - 1$ $b := b + 1$ **end**

- INV applied to ML_in and IL_in holds trivially

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

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

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

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

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

ML_out
  when
    $ml\_tl = \text{green}$
  then
    $a := a + 1$
  end

IL_out
  when
    $il\_tl = \text{green}$
  then
    $b := b - 1$
    $c := c + 1$
  end

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

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

- INV applied to ML_out and IL_out raise some difficulties

What we Have to Prove

- ML_out / **inv2_4** / INV
- IL_out / **inv2_3** / INV
- ML_out / **inv2_3** / INV
- IL_out / **inv2_4** / INV

More Logical Rules of Inferences

- Rules about implication

$$\frac{\mathbf{H, P, Q} \;\vdash\; \mathbf{R}}{\mathbf{H},\; \mathbf{P},\; \mathbf{P} \Rightarrow \mathbf{Q} \;\vdash\; \mathbf{R}} \quad \text{IMP\_L}$$

$$\frac{\mathbf{H, P} \;\vdash\; \mathbf{Q}}{\mathbf{H} \;\vdash\; \mathbf{P} \Rightarrow \mathbf{Q}} \quad \text{IMP\_R}$$

- Rules about negation

$$\frac{\mathbf{H} \;\vdash\; \mathbf{P}}{\mathbf{H},\; \neg\mathbf{P} \;\vdash\; \mathbf{Q}} \quad \text{NOT\_L}$$

$$\frac{\mathbf{H, P} \;\vdash\; \mathbf{Q} \qquad \mathbf{H, P} \;\vdash\; \neg\mathbf{Q}}{\mathbf{H} \;\vdash\; \neg\mathbf{P}} \quad \text{NOT\_R}$$

Proving Preservation of **inv2_4** by Event ML_out

axm0_1	$d \in \mathbb{N}$
axm0_2	$0 < d$
axm2_1	$COLOR = \{\text{green}, \text{red}\}$
axm2_2	$\text{green} \neq \text{red}$
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	$a = 0 \;\vee\; c = 0$
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	$il\_tl = \text{green} \Rightarrow 0 < b \;\wedge\; a = 0$
Guard of event ML_out	$ml\_tl = \text{green}$
$\vdash$	$\vdash$
Modified invariant **inv2_4**	$il\_tl = \text{green} \Rightarrow 0 < b \;\wedge\; a + 1 = 0$

ML_out / **inv2_4** / INV

ML_out
  when
    $ml\_tl = \text{green}$
  then
    $a := a + 1$
  end

Tentative Proof

$$\begin{array}{l} d \in \mathbb{N} \\ 0 < d \\ COLOR = \{green, red\} \\ green \neq red \\ n \in \mathbb{N} \\ n \leq d \\ a \in \mathbb{N} \\ b \in \mathbb{N} \\ c \in \mathbb{N} \\ a+b+c=n \\ a=0 \;\vee\; c=0 \\ ml\_tl \in COLOR \\ il\_tl \in COLOR \\ ml\_tl = green \;\Rightarrow\; a+b<d \;\wedge\; c=0 \\ il\_tl = green \;\Rightarrow\; 0<b \;\wedge\; a=0 \\ ml\_tl = green \\ \vdash \\ il\_tl = green \;\Rightarrow\; 0<b \;\wedge\; a+1=0 \end{array}$$

MON

$$\begin{array}{l} green \neq red \\ il\_tl = green \;\Rightarrow\; 0<b \;\wedge\; a=0 \\ ml\_tl = green \\ \vdash \\ il\_tl = green \;\Rightarrow\; 0<b \;\wedge\; a+1=0 \end{array}$$

IMP_R . . .

Tentative Proof

$$\begin{array}{l} d \in \mathbb{N} \\ 0 < d \\ COLOR = \{green, red\} \\ green \neq red \\ n \in \mathbb{N} \\ n \leq d \\ a \in \mathbb{N} \\ b \in \mathbb{N} \\ c \in \mathbb{N} \\ a+b+c=n \\ a=0 \;\vee\; c=0 \\ ml\_tl \in COLOR \\ il\_tl \in COLOR \\ ml\_tl = green \;\Rightarrow\; a+b<d \;\wedge\; c=0 \\ il\_tl = green \;\Rightarrow\; 0<b \;\wedge\; a=0 \\ ml\_tl = green \\ \vdash \\ il\_tl = green \;\Rightarrow\; 0<b \;\wedge\; a+1=0 \end{array}$$

MON

$$\begin{array}{l} green \neq red \\ il\_tl = green \;\Rightarrow\; 0<b \;\wedge\; a=0 \\ ml\_tl = green \\ \vdash \\ il\_tl = green \;\Rightarrow\; 0<b \;\wedge\; a+1=0 \end{array}$$

IMP_R . . .

. . .

$$\begin{array}{l} green \neq red \\ il\_tl = green \;\Rightarrow\; 0<b \;\wedge\; a=0 \\ ml\_tl = green \\ il\_tl = green \\ \vdash \\ 0<b \;\wedge\; a+1=0 \end{array}$$

IMP_L

$$\begin{array}{l} green \neq red \\ 0<b \;\wedge\; a=0 \\ ml\_tl = green \\ il\_tl = green \\ \vdash \\ 0<b \;\wedge\; a+1=0 \end{array}$$

AND_L . . .

Tentative Proof (cont'd)

. . .

$$\begin{array}{l} green \neq red \\ 0<b \\ a=0 \\ ml\_tl = green \\ il\_tl = green \\ \vdash \\ 0<b \;\wedge\; a+1=0 \end{array}$$

AND_R

$$\begin{array}{l} green \neq red \\ 0<b \\ a=0 \\ ml\_tl = green \\ il\_tl = green \\ \vdash \\ 0<b \end{array}$$

MON

$$0<b \;\vdash\; 0<b$$

HYP

$$\begin{array}{l} green \neq red \\ 0<b \\ a=0 \\ ml\_tl = green \\ il\_tl = green \\ \vdash \\ a+1=0 \end{array}$$

EQ_LR

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

ARITH

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

?

Proving Preservation of **inv2_3** by Event IL_out

axm0_1	$d \in \mathbb{N}$
axm0_2	$0 < d$
axm2_1	$COLOR = \{green, red\}$
axm2_2	$green \neq red$
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	$a=0 \;\vee\; c=0$
inv2_1	$ml\_tl \in COLOR$
inv2_2	$il\_tl \in COLOR$
inv2_3	$ml\_tl = green \;\Rightarrow\; a+b<d \;\wedge\; c=0$
inv2_4	$il\_tl = green \;\Rightarrow\; 0<b \;\wedge\; a=0$
Guard of IL_out	$il\_tl = green$
$\vdash$	$\vdash$
Modified **inv2_3**	$ml\_tl = green \;\Rightarrow\; a+b-1<d \;\wedge\; c+1=0$

IL_out / **inv2_3** / INV

```
IL_out
  when
    il_tl = green
  then
    b := b − 1
    c := c + 1
  end
```

Tentative Proof

$$
\begin{array}{l}
d \in \mathbb{N} \\
0 < d \\
COLOR = \{\text{green}, \text{red}\} \\
\text{green} \neq \text{red} \\
n \in \mathbb{N} \\
n \leq d \\
a \in \mathbb{N} \\
b \in \mathbb{N} \\
c \in \mathbb{N} \\
a + b + c = n \\
a = 0 \;\vee\; c = 0 \\
ml\_tl \in COLOR \\
il\_tl \in COLOR \\
ml\_tl = \text{green} \Rightarrow a + b < d \;\wedge\; c = 0 \\
il\_tl = \text{green} \Rightarrow 0 < b \;\wedge\; a = 0 \\
il\_tl = \text{green} \\
\vdash \\
ml\_tl = \text{green} \Rightarrow a + b - 1 < d \;\wedge\; c + 1 = 0
\end{array}
$$

MON

$$
\begin{array}{l}
\text{green} \neq \text{red} \\
ml\_tl = \text{green} \Rightarrow a + b < d \;\wedge\; c = 0 \\
il\_tl = \text{green} \\
\vdash \\
ml\_tl = \text{green} \Rightarrow a + b - 1 < d \;\wedge\; c + 1 = 0
\end{array}
$$

IMP_R . . .

Overview
The Requirement Document
Formal Models
Initial Model
First Refinement
Second Refinement
Third Refinement

Tentative Proof

$$
\begin{array}{l}
d \in \mathbb{N} \\
0 < d \\
COLOR = \{\text{green}, \text{red}\} \\
\text{green} \neq \text{red} \\
n \in \mathbb{N} \\
n \leq d \\
a \in \mathbb{N} \\
b \in \mathbb{N} \\
c \in \mathbb{N} \\
a + b + c = n \\
a = 0 \;\vee\; c = 0 \\
ml\_tl \in COLOR \\
il\_tl \in COLOR \\
ml\_tl = \text{green} \Rightarrow a + b < d \;\wedge\; c = 0 \\
il\_tl = \text{green} \Rightarrow 0 < b \;\wedge\; a = 0 \\
il\_tl = \text{green} \\
\vdash \\
ml\_tl = \text{green} \Rightarrow a + b - 1 < d \;\wedge\; c + 1 = 0
\end{array}
$$

MON

$$
\begin{array}{l}
\text{green} \neq \text{red} \\
ml\_tl = \text{green} \Rightarrow a + b < d \;\wedge\; c = 0 \\
il\_tl = \text{green} \\
\vdash \\
ml\_tl = \text{green} \Rightarrow a + b - 1 < d \;\wedge\; c + 1 = 0
\end{array}
$$

IMP_R . . .

. . .

$$
\begin{array}{l}
\text{green} \neq \text{red} \\
ml\_tl = \text{green} \Rightarrow a + b < d \;\wedge\; c = 0 \\
il\_tl = \text{green} \\
ml\_tl = \text{green} \\
\vdash \\
a + b - 1 < d \;\wedge\; c + 1 = 0
\end{array}
$$

IMP_L

$$
\begin{array}{l}
\text{green} \neq \text{red} \\
a + b < d \;\wedge\; c = 0 \\
il\_tl = \text{green} \\
ml\_tl = \text{green} \\
\vdash \\
a + b - 1 < d \;\wedge \\
c + 1 = 0
\end{array}
$$

AND_L . . .

Overview
The Requirement Document
Formal Models
Initial Model
First Refinement
Second Refinement
Third Refinement

Tentative Proof (cont'd)

. . .

$$
\begin{array}{l}
\text{green} \neq \text{red} \\
a + b < d \\
c = 0 \\
il\_tl = \text{green} \\
ml\_tl = \text{green} \\
\vdash \\
a + b - 1 < d \;\wedge \\
c + 1 = 0
\end{array}
$$

AND_R

$$
\begin{array}{l}
\text{green} \neq \text{red} \\
a + b < d \\
c = 0 \\
il\_tl = \text{green} \\
ml\_tl = \text{green} \\
\vdash \\
a + b - 1 < d
\end{array}
$$

MON

$$
a + b < d \;\vdash\; a + b - 1 < d
$$

DEC

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

EQ_LR

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

ARITH

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

?

Overview
The Requirement Document
Formal Models
Initial Model
First Refinement
Second Refinement
Third Refinement

The Solution

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

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

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: $ml\_tl = \text{red} \;\vee\; il\_tl = \text{red}$

Completing the Proof

$$\begin{array}{l} \text{green} \neq \text{red} \\ ml\_tl = \text{red} \;\lor\; il\_tl = \text{red} \\ il\_tl = \text{green} \\ ml\_tl = \text{green} \\ \vdash \\ 1 = 0 \end{array} \quad \text{OR\_L}$$

$$\begin{array}{l} \text{green} \neq \text{red} \\ ml\_tl = \text{red} \\ il\_tl = \text{green} \\ ml\_tl = \text{green} \\ \vdash \\ 1 = 0 \end{array} \xrightarrow{\text{EQ\_LR}} \begin{array}{l} \text{green} \neq \text{red} \\ \text{green} = \text{red} \\ il\_tl = \text{green} \\ \vdash \\ 1 = 0 \end{array} \quad \text{NOT\_L} \cdots$$

$$\begin{array}{l} \text{green} \neq \text{red} \\ il\_tl = \text{red} \\ il\_tl = \text{green} \\ ml\_tl = \text{green} \\ \vdash \\ 1 = 0 \end{array} \xrightarrow{\text{EQ\_LR}} \begin{array}{l} \text{green} \neq \text{red} \\ \text{green} = \text{red} \\ ml\_tl = \text{green} \\ \vdash \\ 1 = 0 \end{array} \quad \text{NOT\_L} \cdots$$

$$\cdots \begin{array}{l} \text{green} = \text{red} \\ il\_tl = \text{green} \\ \vdash \\ \text{green} = \text{red} \end{array} \xrightarrow{\text{MON}} \text{green} = \text{red} \;\vdash\; \text{green} = \text{red} \quad \text{HYP}$$

$$\cdots \begin{array}{l} \text{green} = \text{red} \\ ml\_tl = \text{green} \\ \vdash \\ \text{green} = \text{red} \end{array} \xrightarrow{\text{MON}} \text{green} = \text{red} \;\vdash\; \text{green} = \text{red} \quad \text{HYP}$$

Going back to the Requirements Document

inv2_5: $ml\_tl = \text{red} \;\lor\; il\_tl = \text{red}$

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	$COLOR = \{\text{green}, \text{red}\}$
axm2_2	$\text{green} \neq \text{red}$
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	$il\_tl \in COLOR$
inv2_3	$ml\_tl = \text{green} \Rightarrow a + b < d \;\land\; c = 0$
inv2_4	$il\_tl = \text{green} \Rightarrow 0 < b \;\land\; a = 0$
Guard of ML_out	$ml\_tl = \text{green}$
$\vdash$	$\vdash$
Modified **inv2_3**	$ml\_tl = \text{green} \Rightarrow a + 1 + b < d \;\land\; c = 0$

ML_out / **inv2_3** / INV

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

Tentative Proof

$$
\begin{array}{l}
d \in \mathbb{N} \\
0 < d \\
COLOR = \{\text{green}, \text{red}\} \\
\text{green} \neq \text{red} \\
n \in \mathbb{N} \\
n \leq d \\
a \in \mathbb{N} \\
b \in \mathbb{N} \\
c \in \mathbb{N} \\
a + b + c = n \\
a = 0 \;\lor\; c = 0 \\
\mathit{ml\_tl} \in COLOR \\
\mathit{il\_tl} \in COLOR \\
\mathit{ml\_tl} = \text{green} \Rightarrow a + b < d \;\land\; c = 0 \\
\mathit{il\_tl} = \text{green} \Rightarrow 0 < b \;\land\; a = 0 \\
\mathit{ml\_tl} = \text{green} \\
\vdash \\
\quad \mathit{ml\_tl} = \text{green} \Rightarrow a + 1 + b < d \;\land\; c = 0
\end{array}
\quad \text{MON} \quad
\boxed{\begin{array}{l}
\mathit{ml\_tl} = \text{green} \Rightarrow a + b < d \;\land\; c = 0 \\
\vdash \\
\mathit{ml\_tl} = \text{green} \Rightarrow a + 1 + b < d \;\land\; c = 0
\end{array}}
\quad \text{IMP\_R} \cdots
$$

Tentative Proof

$$
\begin{array}{l}
d \in \mathbb{N} \\
0 < d \\
COLOR = \{\text{green}, \text{red}\} \\
\text{green} \neq \text{red} \\
n \in \mathbb{N} \\
n \leq d \\
a \in \mathbb{N} \\
b \in \mathbb{N} \\
c \in \mathbb{N} \\
a + b + c = n \\
a = 0 \;\lor\; c = 0 \\
\mathit{ml\_tl} \in COLOR \\
\mathit{il\_tl} \in COLOR \\
\mathit{ml\_tl} = \text{green} \Rightarrow a + b < d \;\land\; c = 0 \\
\mathit{il\_tl} = \text{green} \Rightarrow 0 < b \;\land\; a = 0 \\
\mathit{ml\_tl} = \text{green} \\
\vdash \\
\quad \mathit{ml\_tl} = \text{green} \Rightarrow a + 1 + b < d \;\land\; c = 0
\end{array}
\quad \text{MON} \quad
\boxed{\begin{array}{l}
\mathit{ml\_tl} = \text{green} \Rightarrow a + b < d \;\land\; c = 0 \\
\vdash \\
\mathit{ml\_tl} = \text{green} \Rightarrow a + 1 + b < d \;\land\; c = 0
\end{array}}
\quad \text{IMP\_R} \cdots
$$

$$
\cdots
\boxed{\begin{array}{l}
\mathit{ml\_tl} = \text{green} \Rightarrow a + b < d \;\land\; c = 0 \\
\mathit{ml\_tl} = \text{green} \\
\vdash \\
a + 1 + b < d \;\land\; c = 0
\end{array}}
\quad \text{IMP\_L} \quad
\boxed{\begin{array}{l}
a + b < d \;\land\; c = 0 \\
\mathit{ml\_tl} = \text{green} \\
\vdash \\
a + 1 + b < d \;\land\; c = 0
\end{array}}
\quad \text{AND\_L} \cdots
$$

Tentative Proof (cont'd)

$$
\cdots
\boxed{\begin{array}{l}
a + b < d \\
c = 0 \\
\mathit{ml\_tl} = \text{green} \\
\vdash \\
a + 1 + b < d \;\land\; c = 0
\end{array}}
\quad \text{AND\_R} \quad
\left\{
\begin{array}{l}
\boxed{\begin{array}{l}
a + b < d \\
c = 0 \\
\mathit{ml\_tl} = \text{green} \\
\vdash \\
a + 1 + b < d
\end{array}}
\quad \text{MON} \quad
\boxed{\begin{array}{l}
a + b < d \\
\mathit{ml\_tl} = \text{green} \\
\vdash \\
a + 1 + b < d
\end{array}}
\quad ? \\[2ex]
\boxed{\begin{array}{l}
a + b < d \\
c = 0 \\
\mathit{ml\_tl} = \text{green} \\
\vdash \\
c = 0
\end{array}}
\quad \text{MON} \quad
\boxed{c = 0 \;\vdash\; c = 0}
\quad \text{HYP}
\end{array}
\right.
$$

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

ML_out_1
when
  $\mathit{ml\_tl} = \text{green}$
  $a + 1 + b < d$
then
  $a := a + 1$
end

ML_out_2
when
  $\mathit{ml\_tl} = \text{green}$
  $a + 1 + b = d$
then
  $a := a + 1$
  $\mathit{ml\_tl} := \text{red}$
end

Intuitive Explanation

ML_out_1
when
  $\mathit{ml\_tl} = \text{green}$
  $a + 1 + b < d$
then
  $a := a + 1$
end

ML_out_2
when
  $\mathit{ml\_tl} = \text{green}$
  $a + 1 + b = d$
then
  $a := a + 1$
  $\mathit{ml\_tl} := \text{red}$
end

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

- Consequently, the traffic light $\mathit{ml\_tl}$ must be turned red (while the car enters the bridge)

Proving Preservation of inv2_3 by Event ML_out_1

axm0_1	$d \in \mathbb{N}$
axm0_2	$0 < d$
axm2_1	$COLOR = \{green, red\}$
axm2_2	$green \neq red$
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	$a = 0 \;\vee\; c = 0$
inv2_1	$ml\_tl \in COLOR$
inv2_2	$il\_tl \in COLOR$
inv2_3	$ml\_tl = green \;\Rightarrow\; a + b < d \;\wedge\; c = 0$
inv2_4	$il\_tl = green \;\Rightarrow\; 0 < b \;\wedge\; a = 0$
Guard of ML_out_1	$ml\_tl = green$ $a + 1 + b < d$
$\vdash$	$\vdash$
Modified **inv2_3**	$ml\_tl = green \;\Rightarrow\; a + 1 + b < d \;\wedge\; c = 0$

ML_out_1 / **inv2_3** / INV

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


Proof

$$\begin{array}{l}
d \in \mathbb{N}\\
0 < d\\
COLOR = \{green, red\}\\
green \neq red\\
n \in \mathbb{N}\\
n \leq d\\
a \in \mathbb{N}\\
b \in \mathbb{N}\\
c \in \mathbb{N}\\
a + b + c = n\\
a = 0 \;\vee\; c = 0\\
ml_tl \in COLOR\\
il_tl \in COLOR\\
ml_tl = green \;\Rightarrow\; a + b < d \;\wedge\; c = 0\\
il_tl = green \;\Rightarrow\; 0 < b \;\wedge\; a = 0\\
ml_tl = green\\
a + 1 + b < d\\
\vdash\\
ml_tl = green \;\Rightarrow\; a + 1 + b < d \;\wedge\; c = 0
\end{array}$$

MON

$$\begin{array}{l}
ml_tl = green \;\Rightarrow\; a + b < d \;\wedge\; c = 0\\
a + 1 + b < d\\
\vdash\\
ml_tl = green \;\Rightarrow\; a + 1 + b < d \;\wedge\; c = 0
\end{array}$$

IMP_R · · ·

Proof

$$\begin{array}{l}
d \in \mathbb{N}\\
0 < d\\
COLOR = \{green, red\}\\
green \neq red\\
n \in \mathbb{N}\\
n \leq d\\
a \in \mathbb{N}\\
b \in \mathbb{N}\\
c \in \mathbb{N}\\
a + b + c = n\\
a = 0 \;\vee\; c = 0\\
ml_tl \in COLOR\\
il_tl \in COLOR\\
ml_tl = green \;\Rightarrow\; a + b < d \;\wedge\; c = 0\\
il_tl = green \;\Rightarrow\; 0 < b \;\wedge\; a = 0\\
ml_tl = green\\
a + 1 + b < d\\
\vdash\\
ml_tl = green \;\Rightarrow\; a + 1 + b < d \;\wedge\; c = 0
\end{array}$$

MON

$$\begin{array}{l}
ml_tl = green \;\Rightarrow\; a + b < d \;\wedge\; c = 0\\
a + 1 + b < d\\
\vdash\\
ml_tl = green \;\Rightarrow\; a + 1 + b < d \;\wedge\; c = 0
\end{array}$$

IMP_R · · ·

· · ·

$$\begin{array}{l}
ml_tl = green \;\Rightarrow\; a + b < d \;\wedge\; c = 0\\
ml_tl = green\\
a + 1 + b < d\\
\vdash\\
a + 1 + b < d \;\wedge\; c = 0
\end{array}$$

IMP_L

$$\begin{array}{l}
a + b < d \;\wedge\; c = 0\\
ml_tl = green\\
a + 1 + b < d\\
\vdash\\
a + 1 + b < d \;\wedge\; c = 0
\end{array}$$

AND_L · · ·

Proof (cont'd)

· · ·

$$\begin{array}{l}
a + b < d\\
c = 0\\
ml_tl = green\\
a + 1 + b < d\\
\vdash\\
a + 1 + b < d \;\wedge\; c = 0
\end{array}$$

AND_R

$$\begin{array}{l}
a + b < d\\
c = 0\\
ml_tl = green\\
a + 1 + b < d\\
\vdash\\
a + 1 + b < d
\end{array}$$

MON

$$\begin{array}{l}
a + 1 + b < d\\
\vdash\\
a + 1 + b < d
\end{array}$$

HYP

$$\begin{array}{l}
a + b < d\\
c = 0\\
ml_tl = green\\
a + 1 + b < d\\
\vdash\\
c = 0
\end{array}$$

MON

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

HYP

Proving Preservation of inv2_3 by Event ML_out_2

axm0_1	$d \in \mathbb{N}$
axm0_2	$0 < d$
axm2_1	$COLOR = \{\text{green}, \text{red}\}$
axm2_2	$\text{green} \neq \text{red}$
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	$a = 0 \quad \vee \quad c = 0$
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	$il\_tl = \text{green} \Rightarrow 0 < b \wedge a = 0$
Guard of ML_out_2	$ml\_tl = \text{green}$
	$a + 1 + b = d$
$\vdash$	$\vdash$
Modified **inv2_3**	$\text{red} = \text{green} \Rightarrow a + 1 + b < d \wedge c = 0$

ML_out_2 / **inv2_3** / INV

ML_out_2
when
$ml\_tl = \text{green}$
$a + 1 + b = d$
then
$a := a + 1$
$ml\_tl := red$
end

Proof

$d \in \mathbb{N}$
$0 < d$
$COLOR = \{\text{green}, \text{red}\}$
$\text{green} \neq \text{red}$
$n \in \mathbb{N}$
$n \leq d$
$a \in \mathbb{N}$
$b \in \mathbb{N}$
$c \in \mathbb{N}$
$a + b + c = n$
$a = 0 \quad \vee \quad c = 0$
$ml\_tl \in COLOR$
$il\_tl \in COLOR$
$ml\_tl = \text{green} \Rightarrow a + b < d \wedge c = 0$
$il\_tl = \text{green} \Rightarrow 0 < b \wedge a = 0$
$ml\_tl = \text{green}$
$a + 1 + b = d$
$\vdash$
$\text{red} = \text{green} \Rightarrow a + 1 + b < d \wedge c = 0$

MON

$\text{green} \neq \text{red}$
$\vdash$
$\text{red} = \text{green} \Rightarrow a + 1 + b < d \wedge c = 0$

IMP_R

Proof

$d \in \mathbb{N}$
$0 < d$
$COLOR = \{\text{green}, \text{red}\}$
$\text{green} \neq \text{red}$
$n \in \mathbb{N}$
$n \leq d$
$a \in \mathbb{N}$
$b \in \mathbb{N}$
$c \in \mathbb{N}$
$a + b + c = n$
$a = 0 \quad \vee \quad c = 0$
$ml\_tl \in COLOR$
$il\_tl \in COLOR$
$ml\_tl = \text{green} \Rightarrow a + b < d \wedge c = 0$
$il\_tl = \text{green} \Rightarrow 0 < b \wedge a = 0$
$ml\_tl = \text{green}$
$a + 1 + b = d$
$\vdash$
$\text{red} = \text{green} \Rightarrow a + 1 + b < d \wedge c = 0$

MON

$\text{green} \neq \text{red}$
$\vdash$
$\text{red} = \text{green} \Rightarrow a + 1 + b < d \wedge c = 0$

IMP_R

. . .

$\text{green} \neq \text{red}$
$\text{red} = \text{green}$
$\vdash$
$a + 1 + b < d \wedge c = 0$

EQ_LR

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

NOT_L

$\vdash$
$\text{green} = \text{green}$

EQL

What we Have to Prove

- ML_out / **inv2_4** / INV done
- IL_out / **inv2_3** / INV done
- ML_out / **inv2_3** / INV done
- IL_out / **inv2_4** / INV
- ML_tl_green / **inv2_5** / INV
- IL_tl_green / **inv2_5** / INV

Proving Preservation of **inv2_4** by Event IL_out

axm0_1	$d \in \mathbb{N}$
axm0_2	$0 < d$
axm2_1	$COLOR = \{\text{green}, \text{red}\}$
axm2_2	$\text{green} \neq \text{red}$
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	$il\_tl \in COLOR$
inv2_3	$ml\_tl = \text{green} \ \Rightarrow \ a + b < d \ \land \ c = 0$
inv2_4	$il\_tl = \text{green} \ \Rightarrow \ 0 < b \ \land \ a = 0$
Guard of event IL_out	$il\_tl = \text{green}$
$\vdash$	$\vdash$
Modified invariant **inv2_4**	$il\_tl = \text{green} \ \Rightarrow \ 0 < b - 1 \ \land \ a = 0$

IL_out / **inv2_4** / INV

IL_out
when
$il\_tl = \text{green}$
then
$b := b - 1$
$c := c + 1$
end

Tentative Proof

$$\begin{array}{l} d \in \mathbb{N} \\ 0 < d \\ COLOR = \{\text{green}, \text{red}\} \\ \text{green} \neq \text{red} \\ n \in \mathbb{N} \\ n \leq d \\ a \in \mathbb{N} \\ b \in \mathbb{N} \\ c \in \mathbb{N} \\ a + b + c = n \\ a = 0 \ \lor \ c = 0 \\ ml\_tl \in COLOR \\ il\_tl \in COLOR \\ ml\_tl = \text{green} \Rightarrow a + b < d \ \land \ c = 0 \\ il\_tl = \text{green} \Rightarrow 0 < b \ \land \ a = 0 \\ il\_tl = \text{green} \\ \vdash \\ il\_tl = \text{green} \Rightarrow 0 < b - 1 \ \land \ a = 0 \end{array} \quad \text{MON} \quad \begin{array}{l} il\_tl = \text{green} \Rightarrow 0 < b \ \land \ a = 0 \\ il\_tl = \text{green} \\ \vdash \\ il\_tl = \text{green} \Rightarrow 0 < b - 1 \ \land \\ \quad a = 0 \end{array} \quad \text{IMP\_R}$$

$$\ldots \quad \begin{array}{l} il\_tl = \text{green} \Rightarrow 0 < b \ \land \ a = 0 \\ il\_tl = \text{green} \\ \vdash \\ 0 < b - 1 \ \land \ a = 0 \end{array} \quad \text{IMP\_L} \quad \begin{array}{l} 0 < b \ \land \ a = 0 \\ \vdash \\ 0 < b - 1 \ \land \ a = 0 \end{array} \quad \text{AND\_L}$$

Tentative Proof (cont'd)

$$\ldots \quad \begin{array}{l} 0 < b \\ a = 0 \\ \vdash \\ 0 < b - 1 \ \land \ a = 0 \end{array} \quad \text{AND\_R} \left\{ \begin{array}{l} \begin{array}{l} 0 < b \\ a = 0 \\ \vdash \\ 0 < b - 1 \end{array} \quad \text{MON} \quad \begin{array}{l} 0 < b \\ \vdash \\ 0 < b - 1 \end{array} \quad ? \\ \\ \begin{array}{l} 0 < b \\ a = 0 \\ \vdash \\ a = 0 \end{array} \quad \text{MON} \quad \begin{array}{l} a = 0 \\ \vdash \\ a = 0 \end{array} \quad \text{HYP} \end{array} \right.$$

- This requires splitting the concrete IL_out in two separate events IL_out_1 and IL_out_2

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

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

Intuitive Explanation

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

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

Proving Preservation of **inv2_4** by Event IL_out_1

axm0_1	$d \in \mathbb{N}$
axm0_2	$0 < d$
axm2_1	$COLOR = \{\text{green}, \text{red}\}$
axm2_2	$\text{green} \neq \text{red}$
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	$il\_tl \in COLOR$
inv2_3	$ml\_tl = \text{green} \Rightarrow a + b < d \land c = 0$
inv2_4	$il\_tl = \text{green} \Rightarrow 0 < b \land a = 0$
Guard of event IL_out_1	$il\_tl = \text{green}$ $b \neq 1$
$\vdash$	$\vdash$
Modified invariant **inv2_4**	$il\_tl = \text{green} \Rightarrow 0 < b - 1 \land a = 0$

IL_out_1 / **inv2_4** / INV

```
IL_out_1
  when
    il_tl = green
    b ≠ 1
  then
    b, c := b − 1, c + 1
  end
```


Proof

$d \in \mathbb{N}$
$0 < d$
$COLOR = \{\text{green}, \text{red}\}$
$\text{green} \neq \text{red}$
$n \in \mathbb{N}$
$n \leq d$
$a \in \mathbb{N}$
$b \in \mathbb{N}$
$c \in \mathbb{N}$
$a + b + c = n$
$a = 0 \lor c = 0$
$ml\_tl \in COLOR$
$il\_tl \in COLOR$
$ml\_tl = \text{green} \Rightarrow a + b < d \land c = 0$
$il\_tl = \text{green} \Rightarrow 0 < b \land a = 0$
$il\_tl = \text{green}$
$b \neq 1$
$\vdash$
$il\_tl = \text{green} \Rightarrow 0 < b - 1 \land a = 0$

MON

$il\_tl = \text{green} \Rightarrow 0 < b \land a = 0$
$il\_tl = \text{green}$
$b \neq 1$
$\vdash$
$il\_tl = \text{green} \Rightarrow 0 < b - 1 \land a = 0$

IMP_R

…

$il\_tl = \text{green} \Rightarrow 0 < b \land a = 0$
$il\_tl = \text{green}$
$b \neq 1$
$\vdash$
$0 < b - 1 \land a = 0$

IMP_L

$0 < b \land a = 0$
$b \neq 1$
$\vdash$
$0 < b - 1 \land a = 0$

AND_L

Proof (cont'd)

…

$0 < b$
$a = 0$
$b \neq 1$
$\vdash$
$0 < b - 1 \land a = 0$

AND_R

$0 < b$
$a = 0$
$b \neq 1$
$\vdash$
$0 < b - 1$

MON

$0 < b$
$b \neq 1$
$\vdash$
$0 < b - 1$

ARITH

$0 < b - 1$
$\vdash$
$0 < b - 1$

HYP

$0 < b$
$a = 0$
$b \neq 1 \ \vdash$
$a = 0$

MON

$a = 0$
$\vdash$
$a = 0$

HYP

Proving Preservation of **inv2_4** by Event IL_out_2

axm0_1	$d \in \mathbb{N}$
axm0_2	$0 < d$
axm2_1	$COLOR = \{\text{green}, \text{red}\}$
axm2_2	$\text{green} \neq \text{red}$
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	$il\_tl \in COLOR$
inv2_3	$ml\_tl = \text{green} \Rightarrow a + b < d \land c = 0$
inv2_4	$il\_tl = \text{green} \Rightarrow 0 < b \land a = 0$
Guard of event IL_out_2	$il\_tl = \text{green}$ $b = 1$
$\vdash$	$\vdash$
Modified invariant **inv2_4**	$\text{red} = \text{green} \Rightarrow 0 < b - 1 \land a = 0$

IL_out_1 / **inv2_4** / INV

```
IL_out_2
  when
    il_tl = green
    b = 1
  then
    b, c, il_tl := b − 1, c + 1, red
  end
```

Proof

$$
\begin{array}{l}
d \in \mathbb{N} \\
0 < d \\
COLOR = \{\text{green}, \text{red}\} \\
\text{green} \neq \text{red} \\
n \in \mathbb{N} \\
n \leq d \\
a \in \mathbb{N} \\
b \in \mathbb{N} \\
c \in \mathbb{N} \\
a + b + c = n \\
a = 0 \quad \lor \quad c = 0 \\
ml\_tl \in COLOR \\
il\_tl \in COLOR \\
ml\_tl = \text{green} \Rightarrow a + b < d \land c = 0 \\
il\_tl = \text{green} \Rightarrow 0 < b \land a = 0 \\
il\_tl = \text{green} \\
b = 1 \\
\vdash \\
\text{red} = \text{green} \Rightarrow 0 < b - 1 \land a = 0
\end{array}
$$

MON

$$
\begin{array}{l}
\text{green} \neq \text{red} \\
\vdash \\
\text{red} = \text{green} \Rightarrow 0 < b - 1 \land a = 0
\end{array}
$$

IMP_R

...

$$
\begin{array}{l}
\text{green} \neq \text{red} \\
\text{red} = \text{green} \\
\vdash \\
0 < b - 1 \land a = 0
\end{array}
$$

EQ_LR

$$
\begin{array}{l}
\text{green} \neq \text{green} \\
\vdash \\
0 < b - 1 \land a = 0
\end{array}
$$

NOT_L

$$
\begin{array}{l}
\vdash \\
\text{green} = \text{green}
\end{array}
$$

EQL

Overview
The Requirement Document
Formal Models
Initial Model
First Refinement
Second Refinement
Third Refinement

What we Have to Prove

- ML_out / **inv2_4** / INV done
- IL_out / **inv2_3** / INV done
- ML_out / **inv2_3** / INV done
- IL_out / **inv2_4** / INV done
- ML_tl_green / **inv2_5** / INV
- IL_tl_green / **inv2_5** / INV

Overview
The Requirement Document
Formal Models
Initial Model
First Refinement
Second Refinement
Third Refinement

Correcting the New Events

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

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

Unless we correct them as follows:

```
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
```


Overview
The Requirement Document
Formal Models
Initial Model
First Refinement
Second Refinement
Third Refinement

Summary of the Proof Situation

- Correct event refinement: OK
- Absence of divergence of new events: FAILURE
- Absence of deadlock: ?

Divergence of the New Events

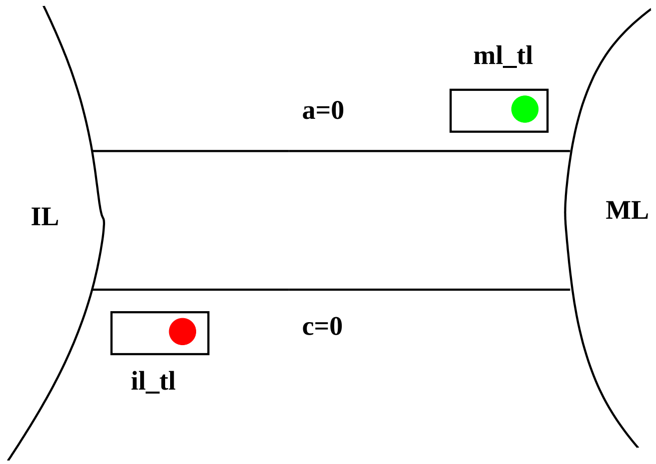
ML_tl_green
  when
    $ml\_tl = \text{red}$
    $a + b < d$
    $c = 0$
  then
    $ml\_tl := \text{green}$
    $il\_tl := \text{red}$
  end

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

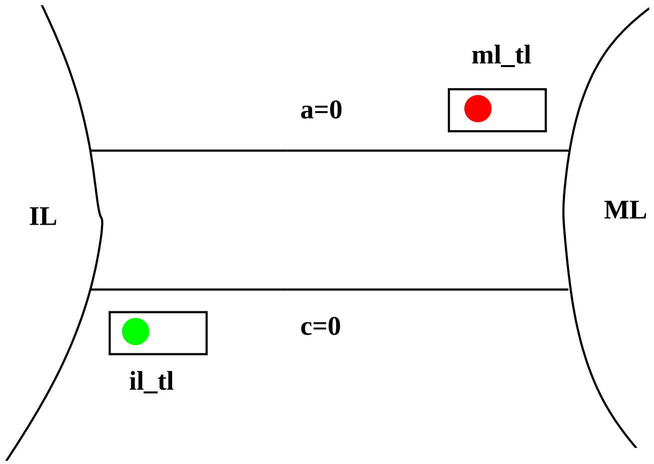
When a and c are both equal to 0 and b is positive, then both events are always alternatively enabled

The lights can change colors very rapidly

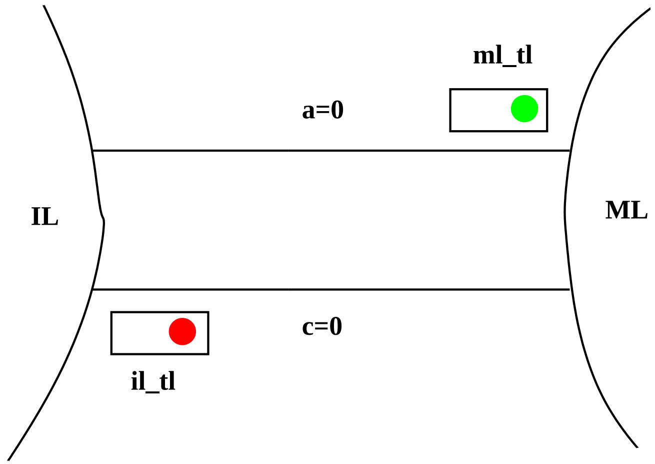
ML_tl_green and IL_tl_green can run for ever



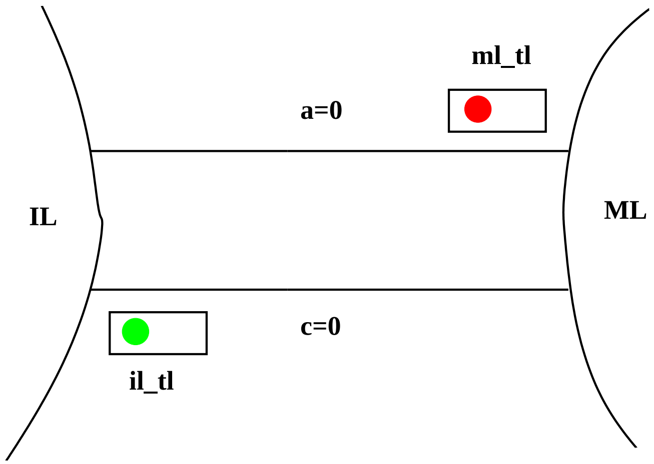
ML_tl_green and IL_tl_green can run for ever



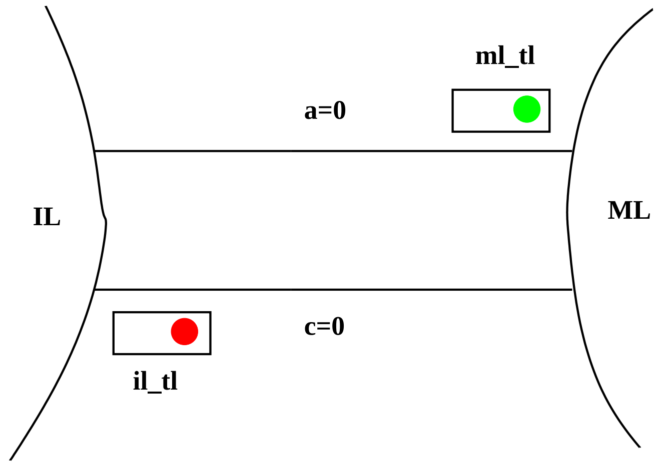
ML_tl_green and IL_tl_green can run for ever



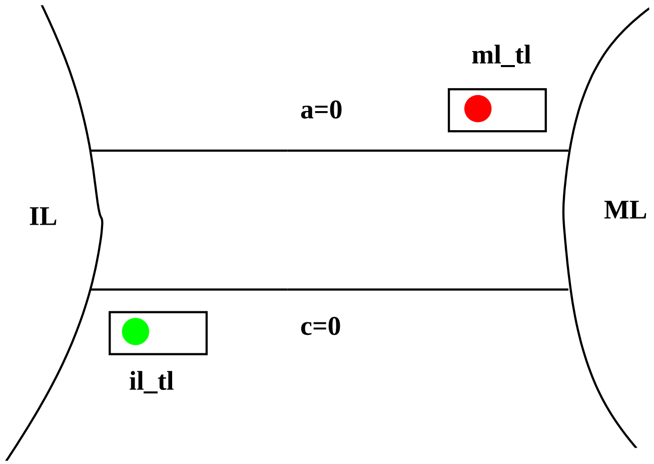
ML_tl_green and IL_tl_green can run for ever



ML_tl_green and IL_tl_green can run for ever



ML_tl_green and IL_tl_green can run for ever



Solution

- Allowing each light to turn green only when at least one car has passed in the other direction

- For this, we introduce two additional variables:

inv2_6:	$ml\_pass \in \{0,1\}$
inv2_7:	$il\_pass \in \{0,1\}$

Modifying Events ML_out_1 and ML_out_2

ML_out_1
when
$ml\_tl = \text{green}$
$a + 1 + b < d$
then
$a := a + 1$
$ml\_pass := 1$
end

ML_out_2
when
$ml\_tl = \text{green}$
$a + 1 + b = d$
then
$a := a + 1$
$ml\_tl := \text{red}$
$ml\_pass := 1$
end

Modifying Events ML_out_1 and ML_out_2

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

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

Modifying Events ML_tl_gree and IL_tl_green

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

IL_tl_green
when
$il\_tl = \text{red}$
$0 < b$
$a = 0$
$ml\_pass = 1$
then
$il\_tl := \text{green}$
$ml\_tl := \text{red}$
$il\_pass := 0$
end

Proving Absence of Divergence

We exhibit the following variant

variant_2: $ml\_pass + il\_pass$

To be Proved

$$\begin{array}{l} ml\_tl = \text{red} \\ a + b < d \\ c = 0 \\ il\_pass = 1 \\ \Rightarrow \\ il\_pass + 0 < \\ ml\_pass + il\_pass \end{array}$$

$$\begin{array}{l} il\_tl = \text{red} \\ b > 0 \\ a = 0 \\ ml\_pass = 1 \\ \Rightarrow \\ ml\_pass + 0 < \\ ml\_pass + il\_pass \end{array}$$

This cannot be proved. This suggests the following invariants:

inv2_8: $ml\_tl = \text{red} \Rightarrow ml\_pass = 1$

inv2_9: $il\_tl = \text{red} \Rightarrow il\_pass = 1$

No Deadlock (1)

$$\begin{array}{l} 0 < d \\ ml\_tl \in \{\text{red}, \text{green}\} \\ il\_tl \in \{\text{red}, \text{green}\} \\ ml\_pass \in \{0, 1\} \\ il\_pass \in \{0, 1\} \\ a \in \mathbb{N} \\ b \in \mathbb{N} \\ c \in \mathbb{N} \\ ml\_tl = \text{red} \Rightarrow ml\_pass = 1 \\ il\_tl = \text{red} \Rightarrow il\_pass = 1 \\ \Rightarrow \\ (ml\_tl = \text{red} \wedge a + b < d \wedge c = 0 \wedge il\_pass = 1) \vee \\ (il\_tl = \text{red} \wedge a = 0 \wedge b > 0 \wedge ml\_pass = 1) \vee \\ ml\_tl = \text{green} \vee il\_tl = \text{green} \vee a > 0 \vee c > 0 \end{array}$$

No Deadlock (2)

The previous statement reduces to the following, which is true

$$\begin{array}{l} 0 < d \\ a \in \mathbb{N} \\ b \in \mathbb{N} \\ c \in \mathbb{N} \\ \Rightarrow \\ (a + b < d \wedge c = 0) \vee \\ (a = 0 \wedge b > 0) \vee \\ a > 0 \vee \\ c > 0 \end{array} \quad \rightsquigarrow \quad \begin{array}{l} 0 < d \\ b \in \mathbb{N} \\ \Rightarrow \\ b < d \vee b > 0 \end{array}$$

Second Refinement: Conclusion

- Thanks to the proofs:
  - We discovered 4 errors
  - We introduced several additional invariants
  - We corrected 4 events
  - We introduced 2 more variables

Conclusion: we Introduced the Superposition Rule

Axioms Abstract invariants Concrete invariants Concrete guards $\vdash$ Same actions on common variables	SIM

Summary of Second Refinement: the State (1)

variables: $a, b, c,$
$ml\_tl, il\_tl, ml\_pass, il\_pass$

inv2_1: $ml\_tl \in \{\text{red}, \text{green}\}$

inv2_2: $il\_tl \in \{\text{red}, \text{green}\}$

inv2_3: $ml\_tl = 1 \Rightarrow a + b < d \wedge c = 0$

inv2_4: $il\_tl = 1 \Rightarrow 0 < b \wedge a = 0$

Summary of Second Refinement: the State (2)

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

inv2_6: $ml\_pass \in \{0, 1\}$

inv2_7: $il\_pass \in \{0, 1\}$

inv2_8: $ml\_tl = \text{red} \Rightarrow ml\_pass = 1$

inv2_9: $il\_tl = \text{red} \Rightarrow il\_pass = 1$

variant2: $ml\_pass + il\_pass$

Summary of Second Refinement: the Event (1)

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

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

Summary of Second Refinement: the Event (2)

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

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

Summary of Second Refinement: the Event (3)

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

IL_tl_green
when
$il\_tl = \text{red}$
$0 < b$
$a = 0$
$ml\_pass = 1$
then
$il\_tl := \text{green}$
$ml\_tl := \text{red}$
$il\_pass := 0$
end

Summary of Second Refinement: the Event (4)

- These events are identical to their abstract versions

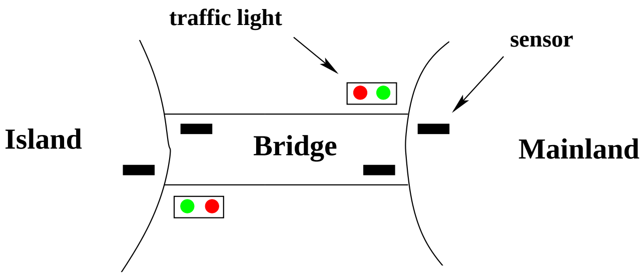
ML_in
when
$0 < c$
then
$c := c - 1$
end

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

Outline

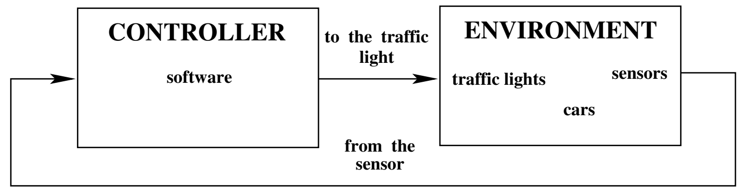
Third Refinement: Adding Car Sensors

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 Variables

Contoller variables: a,

b,

c,

ml_pass,

il_pass

Environment Variables

These new variables denote physical objects
Environment variables: A,

B,

C,

ML_OUT_SR,

ML_IN_SR,

IL_OUT_SR,

IL_IN_SR

Output Channel Variables

Output channels: *ml_tl*,

il_tl

Output Channel Variables

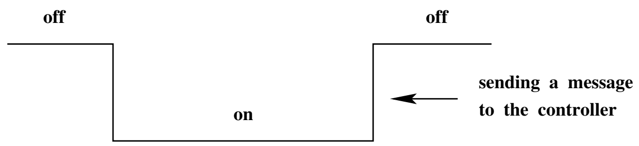
Input channels: *ml_out_10*,

ml_in_10,

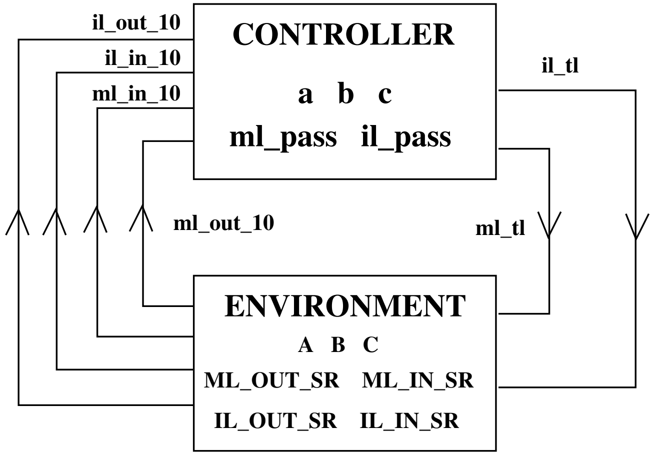
il_in_10,

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)

inv3_1 : $ML\_OUT\_SR \in SENSOR$

inv3_2 : $ML\_IN\_SR \in SENSOR$

inv3_3 : $IL\_OUT\_SR \in SENSOR$

inv3_4 : $IL\_IN\_SR \in SENSOR$

Variables (2)

inv3_5 : $A \in \mathbb{N}$

inv3_6 : $B \in \mathbb{N}$

inv3_7 : $C \in \mathbb{N}$

inv3_8 : $ml\_out\_10 \in \mathrm{BOOL}$

inv3_9 : $ml\_in\_10 \in \mathrm{BOOL}$

inv3_10 : $il\_out\_10 \in \mathrm{BOOL}$

inv3_11 : $il\_in\_10 \in \mathrm{BOOL}$

Invariants (1)

When sensors are on, there are cars on them

inv3_12 : $IL\_IN\_SR = on \;\Rightarrow\; A > 0$

inv3_13 : $IL\_OUT\_SR = on \;\Rightarrow\; B > 0$

inv3_14 : $ML\_IN\_SR = on \;\Rightarrow\; C > 0$

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

Invariants (2)

Drivers obey the traffic lights

inv3_15 : $ml\_out\_10 = \mathrm{TRUE} \;\Rightarrow\; ml\_tl = green$

inv3_16 : $il\_out\_10 = \mathrm{TRUE} \;\Rightarrow\; il\_tl = green$

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

inv3_17 : $IL\_IN\_SR = on \;\Rightarrow\; il\_in\_10 = \mathrm{FALSE}$

inv3_18 : $IL\_OUT\_SR = on \;\Rightarrow\; il\_out\_10 = \mathrm{FALSE}$

inv3_19 : $ML\_IN\_SR = on \;\Rightarrow\; ml\_in\_10 = \mathrm{FALSE}$

inv3_20 : $ML\_OUT\_SR = on \;\Rightarrow\; ml\_out\_10 = \mathrm{FALSE}$

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)

inv3_21 : $il\_in\_10 = \mathrm{TRUE} \;\wedge\; ml\_out\_10 = \mathrm{TRUE} \;\Rightarrow\; A = a$

inv3_22 : $il\_in\_10 = \mathrm{FALSE} \;\wedge\; ml\_out\_10 = \mathrm{TRUE} \;\Rightarrow\; A = a + 1$

inv3_23 : $il\_in\_10 = \mathrm{TRUE} \;\wedge\; ml\_out\_10 = \mathrm{FALSE} \;\Rightarrow\; A = a - 1$

inv3_24 : $il\_in\_10 = \mathrm{FALSE} \;\wedge\; ml\_out\_10 = \mathrm{FALSE} \;\Rightarrow\; A = a$

Invariants (5)

Linking the physical and logical cars (2)

inv3_25 : $il\_in\_10 = \mathrm{TRUE} \;\wedge\; il\_out\_10 = \mathrm{TRUE} \;\Rightarrow\; B = b$

inv3_26 : $il\_in\_10 = \mathrm{TRUE} \;\wedge\; il\_out\_10 = \mathrm{FALSE} \;\Rightarrow\; B = b + 1$

inv3_27 : $il\_in\_10 = \mathrm{FALSE} \;\wedge\; il\_out\_10 = \mathrm{TRUE} \;\Rightarrow\; B = b - 1$

inv3_28 : $il\_in\_10 = \mathrm{FALSE} \;\wedge\; il\_out\_10 = \mathrm{FALSE} \;\Rightarrow\; B = b$

inv3_29 : $il\_out\_10 = \mathrm{TRUE} \;\wedge\; ml\_out\_10 = \mathrm{TRUE} \;\Rightarrow\; C = c$

inv3_30 : $il\_out\_10 = \mathrm{TRUE} \;\wedge\; ml\_out\_10 = \mathrm{FALSE} \;\Rightarrow\; C = c + 1$

inv3_31 : $il\_out\_10 = \mathrm{FALSE} \;\wedge\; ml\_out\_10 = \mathrm{TRUE} \;\Rightarrow\; C = c - 1$

inv3_32 : $il\_out\_10 = \mathrm{FALSE} \;\wedge\; ml\_out\_10 = \mathrm{FALSE} \;\Rightarrow\; C = c$

Invariants (6)

The basic properties hold for the physical cars

inv3_33 : $A = 0 \;\vee\; C = 0$

inv3_34 : $A + B + C \leq d$

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

Refining Abstract Events (1)

```
ML_out_1
  when
    ml_out_10 = TRUE
    a + b + 1 ≠ d
  then
    a := a + 1
    ml_pass := 1
    ml_out_10 := FALSE
  end
```

```
ML_out_2
  when
    ml_out_10 = TRUE
    a + b + 1 = d
  then
    a := a + 1
    ml_tl := red
    ml_pass := 1
    ml_out_10 := FALSE
  end
```

```
(abstract-)ML_out_1
  when
    ml_tl = green
    a + b + 1 ≠ d
  then
    a := a + 1
    ml_pass := 1
  end
```

```
(abstract-)ML_out_2
  when
    ml_tl = green
    a + b + 1 = d
  then
    a := a + 1
    ml_pass := 1
    ml_tl := red
  end
```


Refining Abstract Events (2)

```
IL_out_1
  when
    il_out_10 = TRUE
    b ≠ 1
  then
    b := b − 1
    c := c + 1
    il_pass := 1
    il_out_10 := FALSE
  end
```

```
IL_out_2
  when
    il_out_10 = TRUE
    b = 1
  then
    b := b − 1
    c := c + 1
    il_tl := red
    il_pass := 1
    il_out_10 := FALSE
  end
```

```
(abstract-)IL_out_1
  when
    il_tl = green
    b ≠ 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
    ml_in_10 = TRUE
    0 < c
  then
    c := c − 1
    ml_in_10 := FALSE
  end
```

```
IL_in
  when
    il_in_10 = TRUE
    0 < a
  then
    a := a − 1
    b := b + 1
    il_in_10 := 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
    il_out_10 = FALSE
  then
    ml_tl := green
    il_tl := red
    ml_pass := FALSE
  end
```

```
IL_tl_green
  when
    il_tl = red
    a = 0
    ml_pass = 1
    ml_out_10 = FALSE
  then
    il_tl := green
    ml_tl := red
    il_pass := 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
```

Adding New PHYSICAL Events (1)

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

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

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

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

Adding New PHYSICAL Events (2)

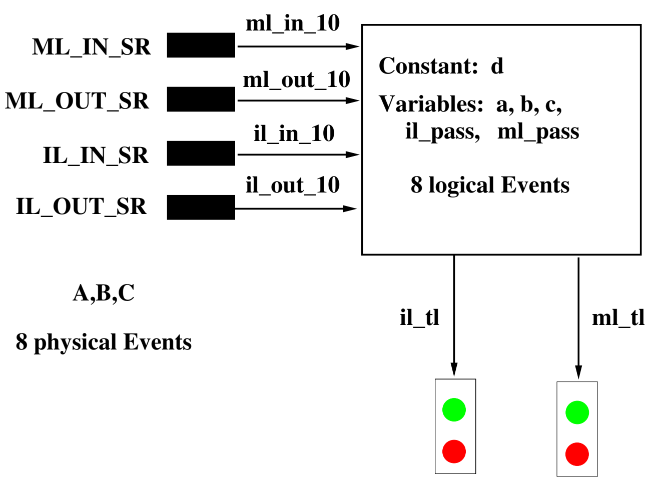
ML_out_dep
when
$ML\_OUT\_SR = on$
$ml\_tl = green$
then
$ML\_OUT\_SR := off$
$ml\_out\_10 := \mathrm{TRUE}$
end

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

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 (1)
  - 1st refinement: 27 (1)
  - 2nd refinement: 81 (1)
  - 3rd refinement: 138 (5)
- All proved automatically (except 8) by the Rodin Platform

Summary of Mathematical Notations (1)

$P \wedge Q$	conjunction
$P \vee Q$	disjunction
$P \Rightarrow Q$	implication
$\neg P$	negation
$x \in S$	set membership operator

Summary of Mathematical Notations (2)

$\mathbb{N}$	set of Natural Numbers: $\{0, 1, 2, 3, \ldots\}$
$\mathbb{Z}$	set of Integers: $\{0, 1, -1, 2, -2, \ldots\}$
$\{a, b, \ldots\}$	set defined in extension
$a + b$	addition of a and b
$a - b$	subtraction of a and b

Summary of Mathematical Notations (3)

$a * b$	product of a and b
$a = b$	equality relation
$a \leq b$	smaller than or equal relation
$a < b$	smaller than relation

Invariant Establishment Proof Rule

- For the init event in the initial model

Axioms of the constants $\Rightarrow$ Modified Invariants	INV

Invariant Preservation Proof Rule

- For other events in the initial model

Axioms of the constants Invariants Guard of the event $\Rightarrow$ Modified Invariants	INV

Deadlock Freeness Rule

- This rule is not mandatory

Axiom of the constant Invariants $\Rightarrow$ Disjunction of the guards	DLF

Refinement Rules (1): Guard Strengthening

- For old events only

Axioms of the constants Abstract invariants Concrete invariants Concrete guards $\Rightarrow$ Abstract guards	GRD

Refinement Rules (2): Invariant Establishment

- For init event only

Axioms of the constants $\Rightarrow$ Modified concrete invariants	INV

Refinement Rules (3): Invariant Preservation

- For all events (except init)

- New events refine an implicit non-guarded event with skip action

Axioms of the constants Abstract invariant Concrete invariant Concrete guard $\Rightarrow$ Modified concrete invariant	INV

Refinement Rules (4): Non-divergence of New Events

- For new events only

Axioms of the constants Abstract invariants Concrete invariants Concrete guard of a new event $\Rightarrow$ Variant $\in \mathbb{N}$	NAT

Refinement Rules (5): Non-divergence of New Events

- For new events only

Axioms of the constants Abstract invariants Concrete invariants Concrete guard of a new event $\Rightarrow$ Modiied variant < Variant	VAR

Refinement Rules (6): Relative Deadlock Freeness

- Global proof rule

Axioms of the constants Abstract invariants Concrete invariants Disjunction of abstract guards $\Rightarrow$ Disjunction of concrete guards	DLF

Refinement Rules (7)

- For old events (in case of superposition)

Axioms of constants Abstract invariants Concrete invariants Concrete guards $\Rightarrow$ Same actions on common variables	SIM