Specifying & Refining a Bridge Controller

MEB: Chapter 2

EECS3342 E: System Specification and Refinement
Fall 2026

CHEN-WEI WANG

Learning Outcomes

This module is designed to help you understand:

- What a ***Requirement Document*** (***RD***) is
- What a ***refinement*** is
- Writing ***formal specifications***
  - (Static) contexts: constants, axioms, theorems
  - (Dynamic) machines: variables, invariants, events, guards, actions
- ***Proof Obligations*** (***POs***) associated with proving:
  - ***refinements***
  - system ***properties***
- Applying ***inference rules*** of the ***sequent calculus***

Correct by Construction

- Directly reasoning about **source code** (written in a programming language) is too complicated to be feasible.
- Instead, given a ***requirements document***, prior to **implementation**, we develop ***models*** through a series of ***refinement*** steps:
  - Each model formalizes an ***external observer***'s perception of the system.
  - Models are "sorted" with ***increasing levels of accuracy*** w.r.t. the system.
  - The ***first model***, though the most ***abstract***, can already be proved satisfying some ***requirements***.
  - Starting from the ***second model***, each model is analyzed and proved ***correct*** relative to two criteria:
    1. Some ***requirements*** (i.e., R-descriptions)
    2. ***Proof Obligations*** (***POs***) related to the ***preceding model*** being ***refined by*** the ***current model*** (via "extra" ***state*** variables and ***events***).
  - The ***last model*** (which is *correct by construction*) should be **sufficiently close** to be transformed into a **working program** (e.g., in C).

State Space of a Model

- A model's *state space* is the set of **all** configurations:
  - Each ***configuration*** assigns values to **constants** & **variables**, subject to:
    - ***axiom*** (e.g., typing constraints, assumptions)
    - ***invariant*** properties/theorems
  - Say an initial model of a bank system with two constants and a variable:

    $c \in \mathbb{N}1 \land L \in \mathbb{N}1 \land accounts \in String \nrightarrow \mathbb{Z}$ /* typing constraint */

    $\forall id \bullet id \in \mathrm{dom}(accounts) \Rightarrow -c \leq accounts(id) \leq L$ /* desired property */

    Q. What is the ***state space*** of this initial model?

    A. All valid combinations of c, L, and $accounts$.
    - Configuration 1: $(c = 1,000, L = 500,000, b = \varnothing)$
    - Configuration 2: $(c = 2,375, L = 700,000, b = \{(''id1'', 500), (''id2'', 1,250)\})$

    ... [Challenge: ***Combinatorial Explosion***]
  - Model Concreteness $\uparrow \Rightarrow$ (State Space $\uparrow \land$ Verification Difficulty $\uparrow$)
- A model's ***complexity*** should be guided by those properties intended to be verified against that model.

  $\Rightarrow$ ***Infeasible*** to prove **all** desired properties on a model.

  $\Rightarrow$ ***Feasible*** to distribute desired properties over a list of ***refinements***.

Roadmap of this Module

- We will walk through the *development process* of constructing ***models*** of a control system regulating cars on a bridge.
  Such controllers exemplify a ***reactive system***.
  (with **sensors** and **actuators**)
- Always stay on top of the following roadmap:
  1. A ***Requirements Document*** (***RD***) of the bridge controller
  2. A brief overview of the ***refinement strategy***
  3. An initial, the most ***abstract*** model
  4. A subsequent ***model*** representing the ***1st refinement***
  5. A subsequent ***model*** representing the ***2nd refinement***
  6. A subsequent ***model*** representing the ***3rd refinement***

Requirements Document: Mainland, Island

Imagine you are asked to build a bridge (as an alternative to ferry) connecting the downtown and Toronto Island.

Page Source: https://soldbyshane.com/area/toronto-islands/

Requirements Document: E-Descriptions

Each ***E-Description*** is an **atomic** ***specification*** of a ***constraint*** or an ***assumption*** of the system's working environment.

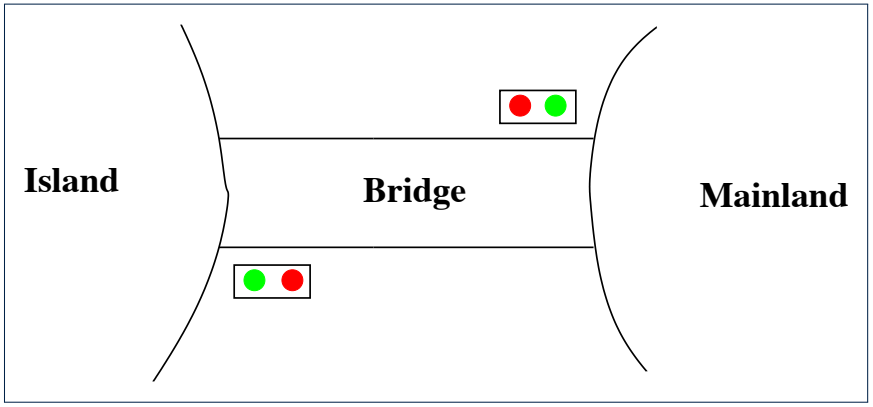
ENV1	The system is equipped with two traffic lights with two colors: green and red.
ENV2	The traffic lights control the entrance to the bridge at both ends of it.
ENV3	Cars are not supposed to pass on a red traffic light, only on a green one.
ENV4	The system is equipped with four sensors with two states: on or off.
ENV5	The sensors are used to detect the presence of a car entering or leaving the bridge: "on" means that a car is willing to enter the bridge or to leave it.

Requirements Document: R-Descriptions

Each ***R-Description*** is an **atomic** ***specification*** of an intended ***functionality*** or a desired ***property*** of the working system.

REQ1	The system is controlling cars on a bridge connecting the mainland to an island.
REQ2	The number of cars on bridge and island is limited.
REQ3	The bridge is one-way or the other, not both at the same time.

Requirements Document: Visual Summary of Equipment Pieces



Refinement Strategy

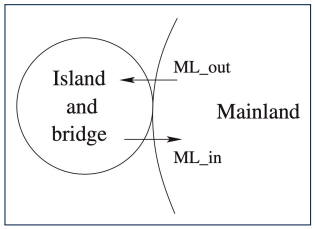
- Before diving into details of the ***models***, we first clarify the adopted ***design strategy of progressive refinements***.
  0. The ***initial model*** (m_0) will address the intended functionality of a limited number of cars on the island and bridge.
  [**REQ2**]
  1. A ***1st refinement*** (m_1 which ***refines*** m_0) will address the intended functionality of the *bridge being one-way*.
  [**REQ1**, **REQ3**]
  2. A ***2nd refinement*** (m_2 which ***refines*** m_1) will address the environment constraints imposed by *traffic lights*.
  [**ENV1, ENV2, ENV3**]
  3. A ***final, 3rd refinement*** (m_3 which ***refines*** m_2) will address the environment constraints imposed by *sensors* and the *architecture*: controller, environment, communication channels.
  [**ENV4, ENV5**]
- Recall *Correct by Construction*:
  From each ***model*** to its ***refinement***, only a manageable amount of details are added, making it ***feasible*** to conduct **analysis** and **proofs**.

Model m_0: Abstraction

- In this most ***abstract*** perception of the bridge controller, we do **not** even consider the bridge, traffic lights, and sensors!
- Instead, we focus on this single ***requirement***:

REQ2	The number of cars on bridge and island is limited.

- **Analogies**:
  - Observe the system from the sky: island and bridge appear only as a compound.



  - "***Zoom in***" on the system as ***refinements*** are introduced.

Model m_0: State Space

1. The ***static*** part is fixed and may be seen/imported.
A ***constant*** d denotes the maximum number of cars allowed to be on the ***island-bridge compound*** at any time.
(whereas cars on the mainland is unbounded)

constants: d

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

Remark. ***Axioms*** are assumed true and may be used to prove theorems.

2. The ***dynamic*** part changes as the system ***evolves***.
A ***variable*** n denotes the actual number of cars, at a given moment, in the ***island-bridge compound***.

variables: n

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

Remark. ***Invariants*** should be (subject to **proofs**):
- ***Established*** when the system is first initialized
- ***Preserved***/***Maintained*** after any ***enabled event***'s actions take effect

Model m_0: State Transitions via Events

- The system acts as an **ABSTRACT STATE MACHINE (ASM)**: it ***evolves*** as ***actions of enabled events*** change values of variables, subject to ***invariants***.
- At any given ***state*** (a valid ***configuration*** of constants/variables):
  - An event is said to be ***enabled*** if its guard evaluates to ***true***.
  - An event is said to be ***disabled*** if its guard evaluates to ***false***.
  - An ***enabled*** event makes a ***state transition*** if it occurs and its ***actions*** take effect.
- *1st* ***event***: A car exits mainland (and enters the island-bridge compound).

```
ML_out
  begin
    n := n + 1
  end
```

Correct Specification? Say $d = 2$.
Witness: ***Event Trace*** $\langle init, ML_out, ML_out, ML_out \rangle$

- *2nd* ***event***: A car enters mainland (and exits the island-bridge compound).

```
ML_in
  begin
    n := n − 1
  end
```

Correct Specification? Say $d = 2$.
Witness: ***Event Trace*** $\langle init, ML_in \rangle$

Model m_0: Actions vs. Before-After Predicates

- When an enabled event e occurs there are two notions of ***state***:
  - ***Before-/Pre-State***: Configuration just ***before*** e's actions take effect
  - ***After-/Post-State***: Configuration just ***after*** e's actions take effect

  Remark. When an enabled event occurs, its ***action(s)*** cause a **transition** from the ***pre-state*** to the ***post-state***.
- As examples, consider ***actions*** of m_0's two events:

Events	ML_out $n := n + 1$	ML_in $n := n - 1$
before–after predicates	$n' = n + 1$	$n' = n - 1$

  - An event ***action*** "$n := n + 1$" is not a variable assignment; instead, it is a ***specification***: "*n becomes n + 1 (when the state transition completes)*".
  - The ***before-after predicate*** (***BAP***) "$n' = n + 1$" expresses that n' (the ***post-state*** value of n) is one more than n (the ***pre-state*** value of n).
- When we express ***proof obligations*** (***POs***) associated with ***events***, we use ***BAP***.

Design of Events: Invariant Preservation

- Our design of the two events

```
ML_out
  begin
    n := n + 1
  end
```

```
ML_in
  begin
    n := n − 1
  end
```

  only specifies how the ***variable*** n should be updated.
- Remember, ***invariants*** are conditions that should never be ***violated***!

```
invariants:
  inv0_1 : n ∈ ℕ
  inv0_2 : n ≤ d
```

- By simulating the system as an ***ASM***, we discover ***witnesses*** (i.e., event traces) of the ***invariants*** not being preserved all the time.

$$\exists s \bullet s \in \textsc{State Space} \Rightarrow \neg \textit{invariants}(s)$$

- We formulate such a commitment to preserving ***invariants*** as a ***proof obligation (PO)*** rule (a.k.a. a ***verification condition (VC)*** rule).

Sequents: Syntax and Semantics

- We formulate each *PO/VC* rule as a (horizontal or vertical) ***sequent***:

$$H \vdash G \qquad \begin{array}{c} H \\ \vdash \\ G \end{array}$$

  - The symbol $\vdash$ is called the ***turnstile***.
  - H is a set of predicates forming the ***hypotheses/assumptions***.
    [assumed as ***true***]
  - G is a set of predicates forming the ***goal/conclusion***.
    [claimed to be ***provable*** from H]
- Informally:
  - $H \vdash G$ is ***true*** if G can be proved by assuming H.
    [i.e., We say "H ***entails*** G" or "H ***yields*** G"]
  - $H \vdash G$ is ***false*** if G cannot be proved by assuming H.
- Formally: $H \vdash G \iff (H \Rightarrow G)$

  Q. What does it mean when H is empty (i.e., no hypotheses)?

  A. $\vdash G \equiv \textit{true} \vdash G$ [Why not $\vdash G \equiv \textit{false} \vdash G$?]

PO of Invariant Preservation: Sketch

- Here is a sketch of the PO/VC rule for *invariant preservation*:

$$\frac{\begin{array}{l}\text{Axioms}\\ \textit{Invariants}\ \text{Satisfied at}\ \textit{Pre-State}\\ \text{Guards of the Event}\\ \vdash\\ \textit{Invariants}\ \text{Satisfied at}\ \textit{Post-State}\end{array}}{\text{INV}}$$

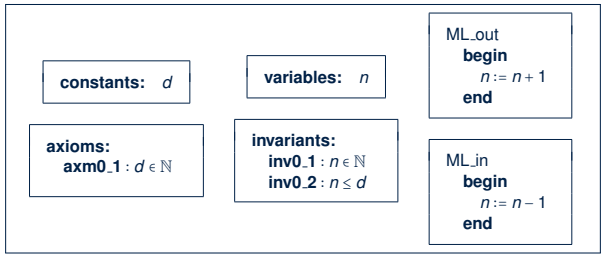
- Informally, this is what the above PO/VC *requires to prove*:

  Assuming **all** axioms, invariants, and the event's guards hold at the ***pre-state***,

  after the ***state transition*** is made by the event,

  all invariants hold at the ***post-state***.

PO of Invariant Preservation: Components



- c: list of ***constants*** $\langle d \rangle$
- $A(c)$: list of ***axioms*** $\langle \textbf{axm0_1} \rangle$
- v and v': list of ***variables*** in ***pre***- and ***post***-states $v \mathrel{\widehat{=}} \langle n \rangle$, $v' \mathrel{\widehat{=}} \langle n' \rangle$
- $I(c, v)$: list of ***invariants*** $\langle \textbf{inv0_1}, \textbf{inv0_2} \rangle$
- $G(c, v)$: the ***event***'s list of guards

  $G(\langle d \rangle, \langle n \rangle)$ of *ML_out* $\mathrel{\widehat{=}} \langle \textit{true} \rangle$, $G(\langle d \rangle, \langle n \rangle)$ of *ML_in* $\mathrel{\widehat{=}} \langle \textit{true} \rangle$
- $E(c, v)$: effect of the ***event***'s actions i.t.o. what variable values ***become***

  $E(\langle d \rangle, \langle n \rangle)$ of *ML_out* $\mathrel{\widehat{=}} \langle n + 1 \rangle$, $E(\langle d \rangle, \langle n \rangle)$ of *ML_out* $\mathrel{\widehat{=}} \langle n - 1 \rangle$
- $v' = E(c, v)$: ***before-after predicate*** formalizing E's actions

  BAP of *ML_out*: $\langle n' \rangle = \langle n + 1 \rangle$, BAP of *ML_in*: $\langle n' \rangle = \langle n - 1 \rangle$

Rule of Invariant Preservation: Sequents

- Based on the components $(c, A(c), v, I(c, v), E(c, v))$, we are able to formally state the ***PO/VC Rule of Invariant Preservation***:

$$\frac{\begin{array}{l}A(c)\\ I(c, v)\\ G(c, v)\\ \vdash\\ I_i(c, E(c, v))\end{array}}{\text{INV}}$$

  where I_i denotes a single invariant condition

  - Accordingly, how many ***sequents*** to be proved? [# events × # invariants]
  - We have two ***sequents*** generated for ***event*** *ML_out* of model m_0:

$$\frac{\begin{array}{l}d \in \mathbb{N}\\ n \in \mathbb{N}\\ n \le d\\ \vdash\\ n + 1 \in \mathbb{N}\end{array}}{\text{ML_out/}\textbf{inv0_1}\text{/INV}} \qquad \frac{\begin{array}{l}d \in \mathbb{N}\\ n \in \mathbb{N}\\ n \le d\\ \vdash\\ n + 1 \le d\end{array}}{\text{ML_out/}\textbf{inv0_2}\text{/INV}}$$

  Exercise. Write the ***POs of invariant preservation*** for event *ML_in*.

- Before claiming that a ***model*** is *correct*, outstanding ***sequents*** associated with all ***POs*** must be proved/discharged.

Inference Rules: Syntax and Semantics

- An *inference rule (IR)* has the following form:

$$\frac{A}{C}\ \textbf{L}$$

  Formally: $A \Rightarrow C$ is an axiom.

  Informally: To prove C, it is sufficient to prove A instead.

  Informally: C is the case, assuming that A is the case.

  - L is a name label for referencing the ***inference rule*** in proofs.
  - A is a **set** of sequents known as ***antecedents*** of rule L.
  - C is a **single** sequent known as ***consequent*** of rule L.

- Let's consider ***inference rules (IRs)*** with two different flavours:

$$\frac{H1 \vdash G}{H1, H2 \vdash G}\ \textbf{MON} \qquad \frac{}{n \in \mathbb{N} \vdash n + 1 \in \mathbb{N}}\ \textbf{P2}$$

  - IR **MON**: To prove $H1, H2 \vdash G$, it suffices to prove $H1 \vdash G$ instead.
  - IR **P2**: $n \in \mathbb{N} \vdash n + 1 \in \mathbb{N}$ is an ***axiom***.

    [proved automatically without further justifications]

Proof of Sequent: Steps and Structure

- To prove the following sequent (related to ***invariant preservation***):

$$\boxed{\begin{array}{l} d \in \mathbb{N} \\ n \in \mathbb{N} \\ n \leq d \\ \vdash \\ n+1 \in \mathbb{N} \end{array}} \quad \underline{\text{ML_out/}\textbf{inv0_1}\text{/INV}}$$

1. Apply a ***inference rule***, which ***transforms*** some "outstanding" **sequent** to <u>one</u> or <u>more</u> other **sequents** to be proved instead.
2. Keep applying ***inference rules*** until **<u>all</u>** ***transformed*** **sequents** are ***axioms*** that do **<u>not</u>** require any further justifications.

- Here is a *formal proof* of ML_out/**inv0_1**/INV, by applying IRs **MON** and **P2**:

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

Example Inference Rules (1)

$$\frac{}{\vdash\ 0 \in \mathbb{N}}\ \textbf{P1}$$

1st Peano axiom: 0 is a natural number.

$$\frac{}{n \in \mathbb{N}\ \vdash\ n+1 \in \mathbb{N}}\ \textbf{P2}$$

2nd Peano axiom: $n+1$ is a natural number, assuming that n is a natural number.

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

$n-1$ is a natural number, assuming that n is positive.

$$\frac{}{n \in \mathbb{N}\ \vdash\ 0 \leq n}\ \textbf{P3}$$

3rd Peano axiom: n is non-negative, assuming that n is a natural number.

Example Inference Rules (2)

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

$n+1$ is less than or equal to m, assuming that n is strictly less than m.

$$\frac{}{n \leq m\ \vdash\ n-1 < m}\ \textbf{DEC}$$

$n-1$ is strictly less than m, assuming that n is less than or equal to m.

Example Inference Rules (3)

$$\frac{H1\ \vdash\ G}{H1, H2\ \vdash\ G}\ \textbf{MON}$$

To prove a goal under certain hypotheses, it suffices to prove it under less hypotheses.

$$\frac{H, P\ \vdash\ R \qquad H, Q\ \vdash\ R}{H, P \vee Q\ \vdash\ R}\ \textbf{OR_L}$$

<u>Proof by Cases</u>:
To prove a goal under a disjunctive assumption, it suffices to prove **<u>independently</u>** the same goal, <u>twice</u>, under each disjunct.

$$\frac{H\ \vdash\ P}{H\ \vdash\ P \vee Q}\ \textbf{OR_R1}$$

To prove a disjunction, it suffices to prove the left disjunct.

$$\frac{H\ \vdash\ Q}{H\ \vdash\ P \vee Q}\ \textbf{OR_R2}$$

To prove a disjunction, it suffices to prove the right disjunct.

Revisiting Design of Events: *ML_out*

- Recall that we already proved ***PO*** ML_out/**inv0_1**/INV:

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

$\therefore$ ***ML_out/inv0_1/INV*** succeeds in being discharged.

- How about the other ***PO*** ML_out/**inv0_2**/INV for the same event?

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

$\therefore$ ***ML_out/inv0_2/INV*** fails to be discharged.

Revisiting Design of Events: *ML_in*

- How about the ***PO*** ML_in/**inv0_1**/INV for *ML_in*:

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

$\therefore$ ***ML_in/inv0_1/INV*** fails to be discharged.

- How about the other ***PO*** ML_in/**inv0_2**/INV for the same event?

$$\begin{array}{l} d \in \mathbb{N} \\ n \in \mathbb{N} \\ n \leq d \\ \vdash \\ n-1 \leq d \end{array} \quad \textbf{MON} \quad \begin{array}{l} n \leq d \\ \vdash \\ n-1 < d \lor n-1 = d \end{array} \quad \textbf{OR_1} \quad \begin{array}{l} n \leq d \\ \vdash \\ n-1 < d \end{array} \quad \textbf{DEC}$$

$\therefore$ ***ML_in/inv0_2/INV*** succeeds in being discharged.

Fixing the Design of Events

- Proofs of ***ML_out/inv0_2/INV*** and ***ML_in/inv0_1/INV*** fail due to the two events being *enabled when they should not*.
- Having this feedback, we add proper ***guards*** to *ML_out* and *ML_in*:

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

```
ML_in
  when
    n > 0
  then
    n := n − 1
  end
```

- Having changed both events, updated ***sequents*** will be generated for the PO/VC rule of ***invariant preservation***.
- All ***sequents*** $(\{ML_out,\ ML_in\} \times \{\textbf{inv0_1},\ \textbf{inv0_2}\})$ now ***provable***?

Revisiting Fixed Design of Events: *ML_out*

- How about the ***PO*** ML_out/**inv0_1**/INV for *ML_out*:

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

$\therefore$ ***ML_out/inv0_1/INV*** still succeeds in being discharged!

- How about the other ***PO*** ML_out/**inv0_2**/INV for the same event?

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

$\therefore$ ***ML_out/inv0_2/INV*** now succeeds in being discharged!

Revisiting Fixed Design of Events: *ML_in*

- How about the ***PO*** ML_in/**inv0_1**/INV for *ML_in*:

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

$\therefore$ ***ML_in/inv0_1/INV*** now succeeds in being discharged!

- How about the other ***PO*** ML_in/**inv0_2**/INV for the same event?

$$\boxed{\begin{array}{l} d \in \mathbb{N} \\ n \in \mathbb{N} \\ n \leq d \\ n > 0 \\ \vdash \\ n - 1 \leq d \end{array}} \quad \textbf{MON} \quad \boxed{\begin{array}{l} n \leq d \\ \vdash \\ n - 1 < d \lor n - 1 = d \end{array}} \quad \textbf{OR_1} \quad \boxed{\begin{array}{l} n \leq d \\ \vdash \\ n - 1 < d \end{array}} \quad \textbf{DEC}$$

$\therefore$ ***ML_in/inv0_2/INV*** still succeeds in being discharged!

Initializing the Abstract System m_0

- Discharging the four ***sequents*** proved that both ***invariant*** conditions are ***preserved*** between occurrences/interleavings of ***events*** `ML_out` and `ML_in`.
- But how are the ***invariants established*** in the first place?

  Analogy. Proving P via ***mathematical induction***, two cases to prove:
  - $P(1), P(2), \ldots$ [***base*** cases $\approx$ ***establishing*** inv.]
  - $P(n) \Rightarrow P(n+1)$ [***inductive*** cases $\approx$ ***preserving*** inv.]
- Therefore, we specify how the ***ASM***'s ***initial state*** looks like:

```
init
  begin
    n := 0
  end
```

✓ The IB compound, once ***initialized***, has no cars.

✓ Initialization always possible: guard is ***true***.

✓ There is no ***pre-state*** for *init*.

$\therefore$ The RHS of := must not involve variables.

$\therefore$ The RHS of := may only involve constants.

✓ There is only the ***post-state*** for *init*.

$\therefore$ Before-***After Predicate***: $n' = 0$

PO of Invariant Establishment

```
init
  begin
    n := 0
  end
```

✓ An ***reactive system***, once ***initialized***, should never terminate.

✓ Event *init* cannot "preserve" the ***invariants***.

$\because$ State before its occurrence (***pre-state***) does not exist.

✓ Event *init* only required to ***establish*** invariants for the first time

- A new formal component is needed:
  - $K(c)$: effect of ***init***'s actions i.t.o. what variable values ***become***
    e.g., $K(\langle d \rangle)$ of *init* $\widehat{=}$ $\langle 0 \rangle$
  - $v' = K(c)$: ***before-after predicate*** formalizing *init*'s actions
    e.g., BAP of *init*: $\langle n' \rangle = \langle 0 \rangle$
- Accordingly, PO of ***invariant establisment*** is formulated as a ***sequent***:

$$\boxed{\begin{array}{l} \text{Axioms} \\ \vdash \\ \textit{Invariants} \text{ Satisfied at } \textit{Post-State} \end{array}} \ \underline{\text{INV}} \qquad \boxed{\begin{array}{l} A(c) \\ \vdash \\ I_i(c, K(c)) \end{array}} \ \underline{\text{INV}}$$

Discharging PO of Invariant Establishment

- How many ***sequents*** to be proved? [# invariants]
- We have two ***sequents*** generated for ***event*** *init* of model m_0:

$$\boxed{\begin{array}{l} d \in \mathbb{N} \\ \vdash \\ 0 \in \mathbb{N} \end{array}} \ \underline{\text{init/}\textbf{inv0_1}\text{/INV}} \qquad \boxed{\begin{array}{l} d \in \mathbb{N} \\ \vdash \\ 0 \leq d \end{array}} \ \underline{\text{init/}\textbf{inv0_2}\text{/INV}}$$

- Can we discharge the ***PO*** init/**inv0_1**/INV?

$$\boxed{\begin{array}{l} d \in \mathbb{N} \\ \vdash \\ 0 \in \mathbb{N} \end{array}} \quad \textbf{MON} \quad \boxed{\begin{array}{l} \vdash \\ 0 \in \mathbb{N} \end{array}} \quad \textbf{P1}$$

$\therefore$ ***init/inv0_1/INV*** succeeds in being discharged.

- Can we discharge the ***PO*** init/**inv0_2**/INV?

$$\boxed{\begin{array}{l} d \in \mathbb{N} \\ \vdash \\ 0 \leq d \end{array}} \quad \textbf{P3}$$

$\therefore$ ***init/inv0_2/INV*** succeeds in being discharged.

System Property: Deadlock Freedom

- So far we have proved that our initial model m_0 is s.t. all ***invariant conditions*** are:
  - Established when system is first initialized via *init*
  - Preserved whenevner there is a ***state transition***
    (via an enabled event: *ML_out* or *ML_in*)
- However, whenever ***event occurrences*** are conditional (i.e., ***guards*** stronger than ***true***), there is a possibility of ***deadlock***:
  - A state where ***guards*** of all events evaluate to ***false***
  - When a ***deadlock*** happens, none of the ***events*** is ***enabled***.
    $\Rightarrow$ The system is blocked and not reactive anymore!
- We express this ***non-blocking*** property as a new requirement:

REQ4	Once started, the system should work for ever.

PO of Deadlock Freedom (1)

- Recall some of the formal components we discussed:
  - c: list of ***constants*** $\langle d \rangle$
  - $A(c)$: list of ***axioms*** $\langle$**axm0_1**$\rangle$
  - ***v*** and ***v'***: list of ***variables*** in ***pre***- and ***post***-states $v \mathrel{\widehat{=}} \langle n \rangle$, $v' \mathrel{\widehat{=}} \langle n' \rangle$
  - $I(c, v)$: list of ***invariants*** $\langle$**inv0_1, inv0_2**$\rangle$
  - $G(c, v)$: the event's list of ***guards***
    $G(\langle d \rangle, \langle n \rangle)$ of *ML_out* $\mathrel{\widehat{=}} \langle n < d \rangle$, $G(\langle d \rangle, \langle n \rangle)$ of *ML_in* $\mathrel{\widehat{=}} \langle n > 0 \rangle$
- A system is ***deadlock-free*** if at least one of its ***events*** is ***enabled***:

$$\frac{\begin{array}{l}\text{Axioms}\\ \textit{Invariants}\ \text{Satisfied at}\ \textit{Pre-State}\\ \vdash\\ \text{Disjunction of the guards satisfied at}\ \textit{Pre-State}\end{array}}{}\ \text{DLF} \qquad \frac{\begin{array}{l}A(c)\\ I(c, v)\\ \vdash\\ G_1(c, v) \lor \cdots \lor G_m(c, v)\end{array}}{}\ \text{DLF}$$

To prove about deadlock freedom
  - An event's effect of state transition is **not** relevant.
  - Instead, the evaluation of all events' ***guards*** at the ***pre-state*** is relevant.

PO of Deadlock Freedom (2)

- ***Deadlock freedom*** is not necessarily a desired property.
  $\Rightarrow$ When it is (like m_0), then the generated ***sequents*** must be discharged.
- Applying the PO of ***deadlock freedom*** to the initial model m_0:

$$\frac{\begin{array}{l}A(c)\\ I(c, v)\\ \vdash\\ G_1(c, v) \lor \cdots \lor G_m(c, v)\end{array}}{}\ \text{DLF} \qquad \frac{\begin{array}{l}d \in \mathbb{N}\\ n \in \mathbb{N}\\ n \leq d\\ \vdash\\ n < d \lor n > 0\end{array}}{}\ \text{DLF}$$

Our bridge controller being ***deadlock-free*** means that cars can ***always*** enter (via *ML_out*) or leave (via *ML_in*) the island-bridge compound.

- Can we formally discharge this ***PO*** for our ***initial model*** m_0?

Example Inference Rules (4)

$$\frac{}{H, P \vdash P}\ \textbf{HYP}$$

A goal is proved if it can be assumed.

$$\frac{}{\bot \vdash P}\ \textbf{FALSE_L}$$

Assuming ***false*** ($\bot$), anything can be proved.

$$\frac{}{P \vdash \top}\ \textbf{TRUE_R}$$

true ($\top$) is proved, regardless of the assumption.

$$\frac{}{P \vdash E = E}\ \textbf{EQ}$$

An expression being equal to itself is proved, regardless of the assumption.

Example Inference Rules (5)

$$\frac{H(F), E = F \vdash P(F)}{H(E), E = F \vdash P(E)} \quad \textbf{EQ_LR}$$

To prove a goal $P(E)$ assuming $H(E)$,
where both P and H depend on expression E,
it suffices to prove $P(F)$ assuming $H(F)$,
where both P and H depend on expresion F,
given that E is equal to F.

$$\frac{H(E), E = F \vdash P(E)}{H(F), E = F \vdash P(F)} \quad \textbf{EQ_RL}$$

To prove a goal $P(F)$ assuming $H(F)$,
where both P and H depend on expression F,
it suffices to prove $P(E)$ assuming $H(E)$,
where both P and H depend on expresion E,
given that E is equal to F.

Discharging PO of DLF: Exercise

$$\begin{array}{l} A(c) \\ I(c, v) \\ \vdash \\ G_1(c, v) \vee \cdots \vee G_m(c, v) \end{array} \quad \underline{\text{DLF}} \quad \begin{array}{l} d \in \mathbb{N} \\ n \in \mathbb{N} \\ n \leq d \\ \vdash \\ n < d \vee n > 0 \end{array} \quad ??$$

Discharging PO of DLF: First Attempt

$$\begin{array}{l} d \in \mathbb{N} \\ n \in \mathbb{N} \\ n \leq d \\ \vdash \\ n < d \vee n > 0 \end{array}$$

$\equiv$

$$\begin{array}{l} d \in \mathbb{N} \\ n \in \mathbb{N} \\ n < d \vee n = d \\ \vdash \\ n < d \vee n > 0 \end{array} \ \textbf{MON} \ \begin{array}{l} n < d \vee n = d \\ \vdash \\ n < d \vee n > 0 \end{array} \ \textbf{OR_L} \left\{ \begin{array}{l} \begin{array}{l} n < d \\ \vdash \\ n < d \vee n > 0 \end{array} \ \textbf{OR_R1} \ \begin{array}{l} n < d \\ \vdash \\ n < d \end{array} \ \textbf{HYP} \\ \begin{array}{l} n = d \\ \vdash \\ n < d \vee n > 0 \end{array} \ \textbf{EQ_LR, MON} \ \begin{array}{l} \vdash \\ d < d \vee d > 0 \end{array} \ \textbf{OR_R2} \ \begin{array}{l} \vdash \\ d > 0 \end{array} \ ? \end{array} \right.$$

Why Did the DLF PO Fail to Discharge?

- In our first attempt, proof of the 2nd case failed: $\vdash d > 0$
- This ***unprovable*** sequent gave us a good hint:
  - For the model under consideration (m_0) to be ***deadlock-free***,
    it is required that $d > 0$. [≥ 1 car allowed in the IB compound]
  - But current ***specification*** of m_0 ***not*** strong enough to entail this:
    - $\neg(d > 0) \equiv d \leq 0$ is possible for the current model
    - Given **axm0_1** : $d \in \mathbb{N}$

    $\Rightarrow d = 0$ is allowed by m_0 which causes a ***deadlock***.
- Recall the *init* event and the two ***guarded*** events:

```
init
  begin
    n := 0
  end
```

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

```
ML_in
  when
    n > 0
  then
    n := n − 1
  end
```

When $d = 0$, the disjunction of guards evaluates to ***false***: $0 < 0 \vee 0 > 0$
$\Rightarrow$ As soon as the system is initialized, it ***deadlocks immediately***
as no car can either enter or leave the IR compound!!

Fixing the Context of Initial Model

- Having understood the failed proof, we add a proper ***axiom*** to m_0:

axioms:
axm0_2 : $d > 0$

- We have effectively elaborated on **REQ2**:

REQ2	The number of cars on bridge and island is limited but positive.

- Having changed the context, an updated ***sequent*** will be generated for the PO/VC rule of ***deadlock freedom***.
- Is this new sequent now ***provable***?

Discharging PO of DLF: Second Attempt

$$\begin{array}{l} d \in \mathbb{N} \\ d > 0 \\ n \in \mathbb{N} \\ n \le d \\ \vdash \\ n < d \vee n > 0 \end{array}$$

$\equiv$

$$\begin{array}{l} d \in \mathbb{N} \\ d > 0 \\ n \in \mathbb{N} \\ n < d \vee n = d \\ \vdash \\ n < d \vee n > 0 \end{array} \;\textbf{MON}\; \begin{array}{l} d > 0 \\ n < d \vee n = d \\ \vdash \\ n < d \vee n > 0 \end{array} \;\textbf{OR_L}\; \begin{cases} \begin{array}{l} d > 0 \\ n < d \\ \vdash \\ n < d \vee n > 0 \end{array} \;\textbf{OR_R1}\; \begin{array}{l} d > 0 \\ n < d \\ \vdash \\ n < d \end{array} \;\textbf{HYP} \\ \begin{array}{l} d > 0 \\ n = d \\ \vdash \\ n < d \vee n > 0 \end{array} \;\textbf{EQ_LR, MON}\; \begin{array}{l} d > 0 \\ \vdash \\ d < d \vee d > 0 \end{array} \;\textbf{OR_R2}\; \begin{array}{l} d > 0 \\ \vdash \\ d > 0 \end{array} \;\textbf{HYP} \end{cases}$$

Initial Model: Summary

- The final version of our ***initial model*** m_0 is ***provably correct*** w.r.t.:
  - Establishment of ***Invariants***
  - Preservation of ***Invariants***
  - ***Deadlock*** Freedom
- Here is the final ***specification*** of m_0:

constants: d

axioms:
axm0_1 : $d \in \mathbb{N}$
axm0_2 : $d > 0$

variables: n

invariants:
inv0_1 : $n \in \mathbb{N}$
inv0_2 : $n \le d$

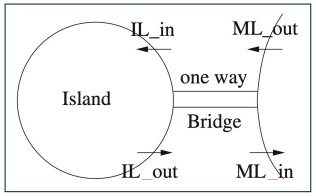
init
begin
$n := 0$
end

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

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

Model m_1: "More Concrete" Abstraction

- First ***refinement*** has a more ***concrete*** perception of the bridge controller:
  - We "**zoom in**" by observing the system from **closer to the ground**, so that the island-bridge compound is split into:
    - the island
    - the (one-way) bridge



  - Nonetheless, traffic lights and sensors remain ***abstracted*** away!
- That is, we focus on these two ***requirement***:

REQ1	The system is controlling cars on a bridge connecting the mainland to an island.
REQ3	The bridge is one-way or the other, not both at the same time.

- We are ***obliged to prove*** this ***added concreteness*** is ***consistent*** with m_0.

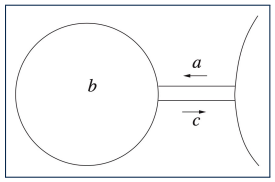
Model m_1: Refined State Space

1. The **static** part is the same as m_0's:

constants: d

axioms:
axm0_1 : $d \in \mathbb{N}$
axm0_2 : $d > 0$

2. The **dynamic** part of the ***concrete state*** consists of three ***variables***:



- ***a***: number of cars on the bridge, heading to the island
- ***b***: number of cars on the island
- ***c***: number of cars on the bridge, heading to the mainland

variables: a, b, c

invariants:
inv1_1 : $a \in \mathbb{N}$
inv1_2 : $b \in \mathbb{N}$
inv1_3 : $c \in \mathbb{N}$
inv1_4 : ??
inv1_5 : ??

✓ **inv1_1**, **inv1_2**, **inv1_3** are ***typing*** constraints.

✓ **inv1_4** ***links***/***glues*** the ***abstract*** and ***concrete*** states.

✓ **inv1_5** specifies that the bridge is one-way.

Model m_1: State Transitions via Events

- The system acts as an **ABSTRACT STATE MACHINE (ASM)**: it ***evolves*** as ***actions of enabled events*** change values of variables, subject to ***invariants***.
- We first consider the "old" ***events*** already existing in m_0.
- ***Concrete***/***Refined*** version of ***event*** *ML_out*:

```
ML_out
  when
    ??
  then
    a := a + 1
  end
```

  - Meaning of *ML_out* is ***refined***: a car exits mainland (getting on the bridge).
  - *ML_out* ***enabled*** only when:
    - the bridge's current traffic flows to the island
    - number of cars on both the bridge and the island is limited
- ***Concrete***/***Refined*** version of ***event*** *ML_in*:

```
ML_in
  when
    ??
  then
    c := c − 1
  end
```

  - Meaning of *ML_in* is ***refined***: a car enters mainland (getting off the bridge).
  - *ML_in* ***enabled*** only when: there is some car on the bridge heading to the mainland.

Model m_1: Actions vs. Before-After Predicates

- Consider the ***concrete***/***refined*** version of ***actions*** of m_0's two events:

Events

```
ML_in
  when
    0 < c
  then
    c := c − 1
  end
```

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

Before–after predicates

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

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

  - An event's ***actions*** are a **specification**: "c becomes c - 1 after the transition".
  - The ***before-after predicate*** (***BAP***) "$c' = c - 1$" expresses that c' (the ***post-state*** value of c) is one less than c (the ***pre-state*** value of c).
  - Given that the ***concrete state*** consists of three variables:
    - An event's ***actions*** only specify those changing from ***pre***-state to ***post***-state. [e.g., $c' = c - 1$]
    - Other unmentioned variables have their ***post***-state values remain unchanged. [e.g., $a' = a \wedge b' = b$]
- When we express ***proof obligations*** (***POs***) associated with ***events***, we use ***BAP***.

States & Invariants: Abstract vs. Concrete

- m_1 refines m_0 by introducing more ***variables***:
  - ***Abstract*** State (of m_0 being refined):

    variables: n
  - ***Concrete*** State (of the refinement model m_1):

    variables: a, b, c
- Accordingly, ***invariants*** may involve different ***states***:
  - ***Abstract*** Invariants (involving the ***abstract*** state only):

    invariants:
    inv0_1 : $n \in \mathbb{N}$
    inv0_2 : $n \leq d$
  - ***Concrete*** Invariants (involving at least the ***concrete*** state):

    invariants:
    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 : $a = 0 \vee c = 0$

Events: Abstract vs. Concrete

- When an ***event*** exists in both models m_0 and m_1, there are two versions of it:
  - The ***abstract*** version modifies the ***abstract*** state.

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

```
(abstract_)ML_in
  when
    n > 0
  then
    n := n − 1
  end
```

  - The ***concrete*** version modifies the ***concrete*** state.

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

```
(concrete_)ML_in
  when
    c > 0
  then
    c := c − 1
  end
```

- A ***new event*** may **only** exist in m_1 (the ***concrete*** model): we will deal with this kind of events later, separately from "redefined/overridden" events.

PO of Refinement: Components (1)

```
constants: d

axioms:
  axm0_1 : d ∈ ℕ
  axm0_2 : d > 0

variables: a, b, c

invariants:
  inv1_1 : a ∈ ℕ
  inv1_2 : b ∈ ℕ
  inv1_3 : c ∈ ℕ
  inv1_4 : a + b + c = n
  inv1_5 : a = 0 ∨ c = 0

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

ML_in
  when
    c > 0
  then
    c := c − 1
  end
```

- c: list of ***constants*** $\langle d \rangle$
- $A(c)$: list of ***axioms*** $\langle \mathbf{axm0_1} \rangle$
- v and v': ***abstract variables*** in pre- & post-states $v \mathrel{\widehat{=}} \langle n \rangle$, $v' \mathrel{\widehat{=}} \langle n \rangle$
- w and w': ***concrete variables*** in pre- & post-states $w \mathrel{\widehat{=}} \langle a, b, c \rangle$, $w' \mathrel{\widehat{=}} \langle a', b', c' \rangle$
- $I(c, v)$: list of ***abstract invariants*** $\langle \mathbf{inv0_1}, \mathbf{inv0_2} \rangle$
- $J(c, v, w)$: list of ***concrete invariants*** $\langle \mathbf{inv1_1}, \mathbf{inv1_2}, \mathbf{inv1_3}, \mathbf{inv1_4}, \mathbf{inv1_5} \rangle$

PO of Refinement: Components (2)

```
constants: d

axioms:
  axm0_1 : d ∈ ℕ
  axm0_2 : d > 0

variables: a, b, c

invariants:
  inv1_1 : a ∈ ℕ
  inv1_2 : b ∈ ℕ
  inv1_3 : c ∈ ℕ
  inv1_4 : a + b + c = n
  inv1_5 : a = 0 ∨ c = 0

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

ML_in
  when
    c > 0
  then
    c := c − 1
  end
```

- $G(c, v)$: list of guards of the ***abstract event***

$$G(\langle d \rangle, \langle n \rangle) \text{ of } ML_out \mathrel{\widehat{=}} \langle n < d \rangle,\ G(c, v) \text{ of } ML_in \mathrel{\widehat{=}} \langle n > 0 \rangle$$

- $H(c, w)$: list of guards of the ***concrete event***

$$H(\langle d \rangle, \langle a, b, c \rangle) \text{ of } ML_out \mathrel{\widehat{=}} \langle a + b < d, c = 0 \rangle,\ H(c, w) \text{ of } ML_in \mathrel{\widehat{=}} \langle c > 0 \rangle$$

PO of Refinement: Components (3)

```
constants: d

axioms:
  axm0_1 : d ∈ ℕ
  axm0_2 : d > 0

variables: a, b, c

invariants:
  inv1_1 : a ∈ ℕ
  inv1_2 : b ∈ ℕ
  inv1_3 : c ∈ ℕ
  inv1_4 : a + b + c = n
  inv1_5 : a = 0 ∨ c = 0

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

ML_in
  when
    c > 0
  then
    c := c − 1
  end
```

- $E(c, v)$: effect of the ***abstract event***'s actions i.t.o. what variable values **become**

$$E(\langle d \rangle, \langle n \rangle) \text{ of } ML_out \mathrel{\widehat{=}} \langle \mathbf{n} + 1 \rangle,\ E(\langle d \rangle, \langle n \rangle) \text{ of } ML_in \mathrel{\widehat{=}} \langle \mathbf{n} - 1 \rangle$$

- $F(c, w)$: effect of the ***concrete event***'s actions i.t.o. what variable values **become**

$$F(c, w) \text{ of } ML_out \mathrel{\widehat{=}} \langle \mathbf{a} + 1, b, c \rangle,\ F(c, w) \text{ of } ML_in \mathrel{\widehat{=}} \langle a, b, \mathbf{c} - 1 \rangle$$

Sketching PO of Refinement

The PO/VC rule for a *proper refinement* consists of two parts:

1. ***Guard Strengthening***

Axioms *Abstract Invariants* Satisfied at Pre-State *Concrete Invariants* Satisfied at Pre-State *Guards* of the *Concrete Event* $\vdash$ *Guards* of the *Abstract Event*	GRD

- A ***concrete*** transition always has an ***abstract*** counterpart.
- A ***concrete*** event is enabled *only if* ***abstract*** counterpart is enabled.

2. ***Invariant Preservation***

Axioms *Abstract Invariants* Satisfied at Pre-State *Concrete Invariants* Satisfied at Pre-State *Guards* of the *Concrete Event* $\vdash$ *Concrete Invariants* Satisfied at **Post**-State	INV

- A ***concrete*** event performs a ***transition*** on ***concrete*** states.
- This ***concrete*** state ***transition*** must be consistent with how its ***abstract*** counterpart performs a corresponding ***abstract transition***.

Note. ***Guard strengthening*** and ***invariant preservation*** are only applicable to events that might be ***enabled*** after the system is launched.

The special, non-guarded `init` event will be discussed separately later.

Refinement Rule: Guard Strengthening

- Based on the components, we are able to formally state the ***PO/VC Rule of Guard Strengthening for Refinement***:

$$\frac{\begin{array}{l} A(c) \\ I(c, v) \\ J(c, v, w) \\ H(c, w) \\ \vdash \\ G_i(c, v) \end{array}}{\text{GRD}}$$

where G_i denotes a single ***guard*** condition of the ***abstract*** event

  - How many ***sequents*** to be proved? [# *abstract* guards]
  - For *ML_out*, only one ***abstract*** guard, so one ***sequent*** is generated :

$$\frac{\begin{array}{lllll} d \in \mathbb{N} & d > 0 & & & \\ n \in \mathbb{N} & n \leq d & & & \\ a \in \mathbb{N} & b \in \mathbb{N} & c \in \mathbb{N} & a + b + c = n & a = 0 \vee c = 0 \\ a + b < d & c = 0 & & & \\ \vdash & & & & \\ n < d & & & & \end{array}}{\text{ML_out/GRD}}$$

- **Exercise**. Write *ML_in*'s ***PO of Guard Strengthening for Refinement***.

PO Rule: Guard Strengthening of *ML_out*

$$\frac{\begin{array}{rl} \textbf{axm0_1} & d \in \mathbb{N} \\ \textbf{axm0_2} & d > 0 \\ \textbf{inv0_1} & n \in \mathbb{N} \\ \textbf{inv0_2} & n \leq d \\ \textbf{inv1_1} & a \in \mathbb{N} \\ \textbf{inv1_2} & b \in \mathbb{N} \\ \textbf{inv1_3} & c \in \mathbb{N} \\ \textbf{inv1_4} & a + b + c = n \\ \textbf{inv1_5} & a = 0 \vee c = 0 \\ \textit{Concrete} \text{ guards of } ML_out & \begin{cases} a + b < d \\ c = 0 \end{cases} \\ & \vdash \\ \textit{Abstract} \text{ guards of } ML_out & n < d \end{array}}{\textbf{ML_out/GRD}}$$

PO Rule: Guard Strengthening of *ML_in*

$$\frac{\begin{array}{rl} \textbf{axm0_1} & d \in \mathbb{N} \\ \textbf{axm0_2} & d > 0 \\ \textbf{inv0_1} & n \in \mathbb{N} \\ \textbf{inv0_2} & n \leq d \\ \textbf{inv1_1} & a \in \mathbb{N} \\ \textbf{inv1_2} & b \in \mathbb{N} \\ \textbf{inv1_3} & c \in \mathbb{N} \\ \textbf{inv1_4} & a + b + c = n \\ \textbf{inv1_5} & a = 0 \vee c = 0 \\ \textit{Concrete} \text{ guards of } ML_in & c > 0 \\ & \vdash \\ \textit{Abstract} \text{ guards of } ML_in & n > 0 \end{array}}{\textbf{ML_in/GRD}}$$

Proving Refinement: ML_out/GRD

$$\begin{array}{l} d \in \mathbb{N} \\ d > 0 \\ n \in \mathbb{N} \\ n \le d \\ a \in \mathbb{N} \\ b \in \mathbb{N} \\ c \in \mathbb{N} \\ a + b + c = n \\ a = 0 \vee c = 0 \\ a + b < d \\ c = 0 \\ \vdash \\ n < d \end{array} \quad \textbf{MON} \quad \begin{array}{l} a + b + c = n \\ a + b < d \\ c = 0 \\ \vdash \\ n < d \end{array} \quad \textbf{EQ_LR, MON} \quad \begin{array}{l} a + b + 0 = n \\ a + b < d \\ \vdash \\ n < d \end{array} \quad \textbf{ARI} \quad \begin{array}{l} a + b = n \\ a + b < d \\ \vdash \\ n < d \end{array} \quad \textbf{EQ_LR, MON} \quad \begin{array}{l} n < d \\ \vdash \\ n < d \end{array} \quad \textbf{HYP}$$

Proving Refinement: ML_in/GRD

$$\begin{array}{l} d \in \mathbb{N} \\ d > 0 \\ n \in \mathbb{N} \\ n \le d \\ a \in \mathbb{N} \\ b \in \mathbb{N} \\ c \in \mathbb{N} \\ a + b + c = n \\ a = 0 \vee c = 0 \\ c > 0 \\ \vdash \\ n > 0 \end{array} \quad \textbf{MON} \quad \begin{array}{l} b \in \mathbb{N} \\ a + b + c = n \\ a = 0 \vee c = 0 \\ c > 0 \\ \vdash \\ n > 0 \end{array} \quad \textbf{OR_L} \quad \left\{ \begin{array}{l} \begin{array}{l} b \in \mathbb{N} \\ a + b + c = n \\ a = 0 \\ c > 0 \\ \vdash \\ n > 0 \end{array} \quad \textbf{EQ_LR, MON} \quad \begin{array}{l} b \in \mathbb{N} \\ 0 + b + c = n \\ c > 0 \\ \vdash \\ n > 0 \end{array} \quad \textbf{ARI} \quad \begin{array}{l} b \in \mathbb{N} \\ b + c = n \\ c > 0 \\ \vdash \\ n > 0 \end{array} \quad \textbf{ARI} \quad \begin{array}{l} c \le n \\ c > 0 \\ \vdash \\ n > 0 \end{array} \quad \textbf{ARI} \quad \begin{array}{l} n > 0 \\ \vdash \\ n > 0 \end{array} \quad \textbf{HYP} \\ \begin{array}{l} b \in \mathbb{N} \\ a + b + c = n \\ c = 0 \\ c > 0 \\ \vdash \\ n > 0 \end{array} \quad \textbf{EQ_LR} \quad \begin{array}{l} b \in \mathbb{N} \\ a + b + 0 = n \\ c = 0 \\ 0 > 0 \\ \vdash \\ n > 0 \end{array} \quad \textbf{MON} \quad \begin{array}{l} 0 > 0 \\ \vdash \\ n > 0 \end{array} \quad \textbf{ARI} \quad \begin{array}{l} \bot \\ \vdash \\ n > 0 \end{array} \quad \textbf{FALSE_L} \end{array} \right.$$

Refinement Rule: Invariant Preservation

- Based on the components, we are able to formally state the ***PO/VC Rule of Invariant Preservation for Refinement***:

$$\begin{array}{l} A(c) \\ I(c, v) \\ J(c, v, w) \\ H(c, w) \\ \vdash \\ J_i(c, E(c, v), F(c, w)) \end{array} \quad \underline{\text{INV}}$$

where J_i denotes a single ***concrete invariant***

  - # ***sequents*** to be proved? [# *concrete*, **old** evts × # *concrete* invariants]
  - Here are two (of the ten) ***sequents*** generated:

$$\begin{array}{l} d \in \mathbb{N} \\ d > 0 \\ n \in \mathbb{N} \\ n \le d \\ a \in \mathbb{N} \\ b \in \mathbb{N} \\ c \in \mathbb{N} \\ a + b + c = n \\ a = 0 \vee c = 0 \\ a + b < d \\ c = 0 \\ \vdash \\ (a + 1) + b + c = (n + 1) \end{array} \quad \underline{\text{ML_out/inv1_4/INV}}$$

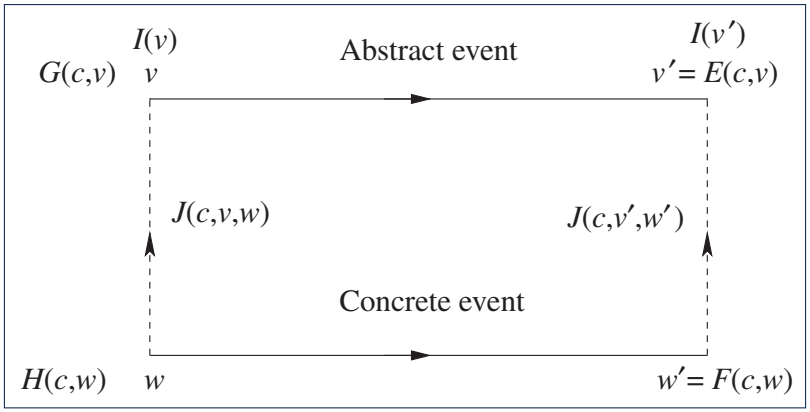
$$\begin{array}{l} d \in \mathbb{N} \\ d > 0 \\ n \in \mathbb{N} \\ n \le d \\ a \in \mathbb{N} \\ b \in \mathbb{N} \\ c \in \mathbb{N} \\ a + b + c = n \\ a = 0 \vee c = 0 \\ c > 0 \\ \vdash \\ a = 0 \vee (c - 1) = 0 \end{array} \quad \underline{\text{ML_in/inv1_5/INV}}$$

- **Exercises**. Specify and prove other **eight** ***POs of Invariant Preservation***.

Visualizing Inv. Preservation in Refinement

Each ***concrete*** event (w to w') is ***simulated by*** an ***abstract*** event (v to v'):

- ***abstract*** & ***concrete*** pre-states related by ***concrete*** invariants $J(c, v, w)$
- ***abstract*** & ***concrete*** post-states related by ***concrete*** invariants $J(c, v', w')$



INV PO of m_1: ML_out/inv1_4/INV

axm0_1	$d \in \mathbb{N}$
axm0_2	$d > 0$
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	$a = 0 \vee c = 0$
Concrete guards of *ML_out*	$a + b < d$ $c = 0$
	$\vdash$
Concrete invariant **inv1_4** with *ML_out*'s effect in the post-state	$(a + 1) + b + c = (n + 1)$

ML_out/inv1_4/INV

INV PO of m_1: ML_in/inv1_5/INV

axm0_1	$d \in \mathbb{N}$
axm0_2	$d > 0$
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	$a = 0 \vee c = 0$
Concrete guards of *ML_in*	$c > 0$
	$\vdash$
Concrete invariant **inv1_5** with *ML_in*'s effect in the post-state	$a = 0 \vee (c - 1) = 0$

ML_in/inv1_5/INV

Proving Refinement: ML_out/inv1_4/INV

$d \in \mathbb{N}$
$d > 0$
$n \in \mathbb{N}$
$n \leq d$
$a \in \mathbb{N}$
$b \in \mathbb{N}$
$c \in \mathbb{N}$
$a + b + c = n$
$a = 0 \vee c = 0$
$a + b < d$
$c = 0$
$\vdash$
$(a + 1) + b + c = (n + 1)$

MON

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

ARI

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

EQ_LR, MON

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

EQ

Proving Refinement: ML_in/inv1_5/INV

$d \in \mathbb{N}$
$d > 0$
$n \in \mathbb{N}$
$n \leq d$
$a \in \mathbb{N}$
$b \in \mathbb{N}$
$c \in \mathbb{N}$
$a + b + c = n$
$a = 0 \vee c = 0$
$c > 0$
$\vdash$
$a = 0 \vee (c - 1) = 0$

MON

$a = 0 \vee c = 0$
$c > 0$
$\vdash$
$a = 0 \vee (c - 1) = 0$

OR_L

- $a = 0$
  $c > 0$
  $\vdash$
  $a = 0 \vee (c - 1) = 0$
  OR_R1
  $a = 0$
  $c > 0$
  $\vdash$
  $a = 0$
  HYP
- $c = 0$
  $c > 0$
  $\vdash$
  $a = 0 \vee (c - 1) = 0$
  EQ_LR, MON
  $0 > 0$
  $\vdash$
  $a = 0 \vee (0 - 1) = 0$
  ARI
  $\bot$
  $\vdash$
  $a = 0 \vee -1 = 0$
  FALSE_L

Initializing the Refined System m_1

- Discharging the **twelve sequents** proved that:
  - ***concrete invariants*** preserved by `ML_out` & `ML_in`
  - ***concrete guards*** of *ML_out* & *ML_in* entail their ***abstract*** counterparts
- What's left is the specification of how the ***ASM***'s ***initial state*** looks like:

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

✓ No cars on bridge (heading either way) and island

✓ Initialization always possible: guard is ***true***.

✓ There is no ***pre-state*** for *init*.

∴ The RHS of := must not involve variables.

∴ The RHS of := may only involve constants.

✓ There is only the ***post-state*** for *init*.

∴ Before-***After Predicate***: $a' = 0 \wedge b' = 0 \wedge c' = 0$

PO of m_1 Concrete Invariant Establishment

- Some (new) formal components are needed:
  - $K(c)$: effect of ***abstract init***'s actions:
    e.g., $K(\langle d \rangle)$ of *init* $\widehat{=}$ $\langle 0 \rangle$
  - $v' = K(c)$: ***before-after predicate*** formalizing ***abstract*** *init*'s actions
    e.g., BAP of *init*: $\langle n' \rangle = \langle 0 \rangle$
  - $L(c)$: effect of ***concrete init***'s actions:
    e.g., $K(\langle d \rangle)$ of *init* $\widehat{=}$ $\langle 0, 0, 0 \rangle$
  - $w' = L(c)$: ***before-after predicate*** formalizing ***concrete*** *init*'s actions
    e.g., BAP of *init*: $\langle a', b', c' \rangle = \langle 0, 0, 0 \rangle$
- Accordingly, PO of ***invariant establisment*** is formulated as a **sequent**:

Axioms ⊢ *Concrete Invariants* Satisfied at Post-State	INV	$A(c)$ ⊢ $J_i(c, K(c), L(c))$	INV

Discharging PO of m_1 Concrete Invariant Establishment

- How many ***sequents*** to be proved? [# *concrete* invariants]
- Two (of the five) sequents generated for ***concrete*** *init* of m_1:

$d \in \mathbb{N}$ $d > 0$ ⊢ $0 + 0 + 0 = 0$	init/**inv1_4**/INV	$d \in \mathbb{N}$ $d > 0$ ⊢ $0 = 0 \vee 0 = 0$	init/**inv1_5**/INV

- Can we discharge the ***PO*** init/**inv1_4**/INV?

$d \in \mathbb{N}$ $d > 0$ ⊢ $0 + 0 + 0 = 0$	**ARI, MON**	⊢ ⊤	**TRUE_R**

∴ ***init/inv1_4/INV*** succeeds in being discharged.

- Can we discharge the ***PO*** init/**inv1_5**/INV?

$d \in \mathbb{N}$ $d > 0$ ⊢ $0 = 0 \vee 0 = 0$	**ARI, MON**	⊢ ⊤	**TRUE_R**

∴ ***init/inv1_5/INV*** succeeds in being discharged.

Model m_1: New, Concrete Events

- The system acts as an **ABSTRACT STATE MACHINE (ASM)**: it ***evolves*** as ***actions of enabled events*** change values of variables, subject to ***invariants***.
- Considered ***concrete/refined events*** already existing in m_0: *ML_out* & *ML_in*
- ***New event*** *IL_in*:

```
IL_in
  when
    ??
  then
    ??
  end
```

  - *IL_in* denotes a car entering the island (getting off the bridge).
  - *IL_in* ***enabled*** only when:
    - The bridge's current traffic flows to the island.
      Q. Limited number of cars on the bridge and the island?
      A. Ensured when the earlier *ML_out* (of same car) occurred
- ***New event*** *IL_out*:

```
IL_out
  when
    ??
  then
    ??
  end
```

  - *IL_out* denotes a car exiting the island (getting on the bridge).
  - *IL_out* ***enabled*** only when:
    - There is some car on the island.
    - The bridge's current traffic flows to the mainland.

Model m_1: BA Predicates of Multiple Actions

Consider ***actions*** of m_1's two ***new*** events:

```
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
```

- What is the ***BAP*** of *ML_in*'s ***actions***?

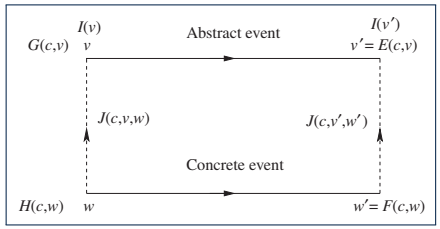
$$a' = a - 1 \wedge b' = b + 1 \wedge c' = c$$

- What is the ***BAP*** of *ML_in*'s ***actions***?

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

Visualizing Inv. Preservation in Refinement

- Recall how a ***concrete*** event is ***simulated*** by its ***abstract*** counterpart:



- For each ***new*** event:
  - Strictly speaking, it does **not** have an ***abstract*** counterpart.
  - It is ***simulated by*** a special ***abstract*** event (transforming v to v'):

```
skip
  begin

  end
```

- *skip* is a "dummy" event: non-guarded and does nothing
- **Q**. ***BAP*** of the skip event?

  A. $n' = n$

Refinement Rule: Invariant Preservation

- The new events *IL_in* and *IL_out* do not exist in $\mathbf{m_0}$, but:
  - They **exist** in $\mathbf{m_1}$ and may impact upon the ***concrete*** state space.
  - They ***preserve*** the ***concrete invariants***, just as *ML_out* & *ML_in* do.
- Recall the ***PO/VC Rule of Invariant Preservation for Refinement***:

$$\frac{\begin{array}{l} A(c) \\ I(c, v) \\ J(c, v, w) \\ H(c, w) \\ \vdash \\ J_i(c, E(c, v), F(c, w)) \end{array}}{\text{INV}}$$

where J_i denotes a single ***concrete invariant***

  - How many ***sequents*** to be proved? [# ***new*** evts × # ***concrete*** invariants]
  - Here are two (of the ten) ***sequents*** generated:

$$\frac{\begin{array}{l} d \in \mathbb{N} \\ d > 0 \\ n \in \mathbb{N} \\ n \leq d \\ a \in \mathbb{N} \\ b \in \mathbb{N} \\ c \in \mathbb{N} \\ a + b + c = n \\ a = 0 \vee c = 0 \\ a > 0 \\ \vdash \\ (a - 1) + (b + 1) + c = n \end{array}}{\text{IL_in/inv1_4/INV}} \qquad \frac{\begin{array}{l} d \in \mathbb{N} \\ d > 0 \\ n \in \mathbb{N} \\ n \leq d \\ a \in \mathbb{N} \\ b \in \mathbb{N} \\ c \in \mathbb{N} \\ a + b + c = n \\ a = 0 \vee c = 0 \\ a > 0 \\ \vdash \\ (a - 1) = 0 \vee c = 0 \end{array}}{\text{IL_in/inv1_5/INV}}$$

- **Exercises**. Specify and prove other **eight** ***POs of Invariant Preservation***.

INV PO of m_1: IL_in/inv1_4/INV

IL_in/inv1_4/INV

axm0_1	$d \in \mathbb{N}$
axm0_2	$d > 0$
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	$a = 0 \vee c = 0$
Guards of *IL_in*	$a > 0$
	$\vdash$
Concrete invariant **inv1_4** with *IL_in*'s effect in the post-state	$(a - 1) + (b + 1) + c = n$

INV PO of m_1: IL_in/inv1_5/INV

axm0_1	$d \in \mathbb{N}$
axm0_2	$d > 0$
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	$a = 0 \vee c = 0$
Guards of *IL_in*	$a > 0$
	$\vdash$
Concrete invariant **inv1_5** with *IL_in*'s effect in the post-state	$(a - 1) = 0 \vee c = 0$

IL_in/inv1_5/INV

Proving Refinement: IL_in/inv1_4/INV

$d \in \mathbb{N}$
$d > 0$
$n \in \mathbb{N}$
$n \leq d$
$a \in \mathbb{N}$
$b \in \mathbb{N}$
$c \in \mathbb{N}$
$a + b + c = n$
$a = 0 \vee c = 0$
$a > 0$
$\vdash$
$(a - 1) + (b + 1) + c = n$

MON

$a + b + c = n$
$\vdash$
$(a - 1) + (b + 1) + c = n$

ARI

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

HYP

Proving Refinement: IL_in/inv1_5/INV

$d \in \mathbb{N}$
$d > 0$
$n \in \mathbb{N}$
$n \leq d$
$a \in \mathbb{N}$
$b \in \mathbb{N}$
$c \in \mathbb{N}$
$a + b + c = n$
$a = 0 \vee c = 0$
$a > 0$
$\vdash$
$(a - 1) = 0 \vee c = 0$

MON

$a = 0 \vee c = 0$
$a > 0$
$\vdash$
$(a - 1) = 0 \vee c = 0$

OR_L

- $a = 0$, $a > 0$, $\vdash$ $(a - 1) = 0 \vee c = 0$ — **EQ_LR, MON**
  - $0 > 0$, $\vdash$ $(0 - 1) = 0 \vee c = 0$ — **ARI**
    - $\bot$, $\vdash$ $-1 = 0 \vee c = 0$ — **FALSE_L**
- $c = 0$, $a > 0$, $\vdash$ $(a - 1) = 0 \vee c = 0$ — **OR_R2**
  - $c = 0$, $a > 0$, $\vdash$ $c = 0$ — **HYP**

Livelock Caused by New Events Diverging

- An alternative m_1 (with **inv1_4**, **inv1_5**, and **guards** of new events removed):

```
constants: d
axioms:
  axm0_1 : d ∈ ℕ
  axm0_2 : d > 0
variables: a, b, c
invariants:
  inv1_1 : a ∈ ℤ
  inv1_2 : b ∈ ℤ
  inv1_3 : c ∈ ℤ

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
  begin
    a := a − 1
    b := b + 1
  end

IL_out
  begin
    b := b − 1
    c := c + 1
  end
```

Concrete invariants are **under-specified**: only typing constraints.

Exercises: Show that **Invariant Preservation** is provable, but **Guard Strengthening** is not.

- Say this alternative m_1 is implemented as is:
  IL_in and *IL_out* **always** ***enabled*** and may occur ***indefinitely***, preventing other "old" events (*ML_out* and *ML_in*) from ever happening:
  $\langle init, ML_out, IL_in, IL_out, IL_in, IL_out, \ldots \rangle$
  Q: What are the corresponding ***abstract*** transitions?
  A: $\langle init, ML_out, skip, skip, skip, skip, \ldots \rangle$ [≈ executing `while(true);`]
- We say that these two ***new*** events *diverge*, creating a *livelock*:
  - Different from a ***deadlock*** ∵ **always** an event occurring (*IL_in* or *IL_out*).
  - But their ***indefinite*** occurrences contribute **nothing** useful.

PO of Convergence of New Events

The PO/VC rule for *non-divergence/livelock freedom* consists of two parts:

- Interleaving of ***new*** events characterized as an integer expr.: ***variant***.
- A variant $V(c,w)$ may refer to constants and/or ***concrete*** variables.
- In the original m_1, let's try **variants** : $2 \cdot a + b$

1. ***Variant Stays Non-Negative***

$$\frac{\begin{array}{l} A(c) \\ I(c,v) \\ J(c,v,w) \\ H(c,w) \\ \vdash \\ V(c,w) \in \mathbb{N} \end{array}}{\text{NAT}}$$

- Variant $V(c, w)$ measures how many more times the ***new*** events can occur.
- If a ***new*** event is ***enabled***, then $V(c,w) > 0$.
- When $V(c,w)$ reaches 0, some "old" events must happen s.t. $V(c,w)$ goes back above 0.

2. ***A New Event Occurrence Decreases Variant***

$$\frac{\begin{array}{l} A(c) \\ I(c,v) \\ J(c,v,w) \\ H(c,w) \\ \vdash \\ V(c,F(c,w)) < V(c,w) \end{array}}{\text{VAR}}$$

- If a ***new*** event is ***enabled*** and occurs, the value of $V(c,w) \downarrow$.

PO of Convergence of New Events: NAT

- Recall: PO related to ***Variant Stays Non-Negative***:

$$\frac{\begin{array}{l} A(c) \\ I(c,v) \\ J(c,v,w) \\ H(c,w) \\ \vdash \\ V(c,w) \in \mathbb{N} \end{array}}{\text{NAT}}$$

How many ***sequents*** to be proved?

[# ***new*** events]

- For the ***new*** event *IL_in*:

$$\frac{\begin{array}{lll} d \in \mathbb{N} & d > 0 & \\ n \in \mathbb{N} & n \leq d & \\ a \in \mathbb{N} & b \in \mathbb{N} & c \in \mathbb{N} \\ a + b + c = n & a = 0 \vee c = 0 & \\ a > 0 & & \\ \vdash & & \\ 2 \cdot a + b \in \mathbb{N} & & \end{array}}{\text{IL_in/NAT}}$$

Exercises: Prove **IL_in/NAT** and Formulate/Prove **IL_out/NAT**.

PO of Convergence of New Events: VAR

- Recall: PO related to ***A New Event Occurrence Decreases Variant***

$$\frac{\begin{array}{l} A(c) \\ I(c,v) \\ J(c,v,w) \\ H(c,w) \\ \vdash \\ V(c,F(c,w)) < V(c,w) \end{array}}{\text{VAR}}$$

How many ***sequents*** to be proved?

[# ***new*** events]

- For the ***new*** event *IL_in*:

$$\frac{\begin{array}{lll} d \in \mathbb{N} & d > 0 & \\ n \in \mathbb{N} & n \leq d & \\ a \in \mathbb{N} & b \in \mathbb{N} & c \in \mathbb{N} \\ a + b + c = n & a = 0 \vee c = 0 & \\ a > 0 & & \\ \vdash & & \\ 2 \cdot (a-1) + (b+1) < 2 \cdot a + b & & \end{array}}{\text{IL_in/VAR}}$$

Exercises: Prove **IL_in/VAR** and Formulate/Prove **IL_out/VAR**.

Convergence of New Events: Exercise

Given the original $\mathbf{m_1}$, what if the following ***variant*** expression is used:

variants : $a + b$

Are the formulated sequents still ***provable***?

PO of Refinement: Deadlock Freedom

- Recall:
  - We proved that the initial model m_0 is deadlock free (see **DLF**).
  - We proved, according to ***guard strengthening***, that if a ***concrete*** event is enabled, then its ***abstract*** counterpart is enabled.
- PO of ***relative deadlock freedom*** for a ***refinement*** model:

$$\begin{array}{l} A(c) \\ I(c,v) \\ J(c,v,w) \\ G_1(c,v) \vee \cdots \vee G_m(c,v) \\ \vdash \\ H_1(c,w) \vee \cdots \vee H_n(c,w) \end{array} \quad \text{DLF}$$

If an ***abstract*** state does not *deadlock* (i.e., $G_1(c,v) \vee \cdots \vee G_m(c,v)$), then its ***concrete*** counterpart does not *deadlock* (i.e., $H_1(c,w) \vee \cdots \vee H_n(c,w)$).

- Another way to think of the above PO:
  The ***refinement*** does **not** introduce, in the ***concrete***, any "new" *deadlock* scenarios **not** existing in the ***abstract*** state.

PO Rule: Relative Deadlock Freedom m_1

$$\begin{array}{rl} \textbf{axm0_1} & \{\ d \in \mathbb{N} \\ \textbf{axm0_2} & \{\ d > 0 \\ \textbf{inv0_1} & \{\ n \in \mathbb{N} \\ \textbf{inv0_2} & \{\ n \leq d \\ \textbf{inv1_1} & \{\ a \in \mathbb{N} \\ \textbf{inv1_2} & \{\ b \in \mathbb{N} \\ \textbf{inv1_3} & \{\ c \in \mathbb{N} \\ \textbf{inv1_4} & \{\ a + b + c = n \\ \textbf{inv1_5} & \{\ a = 0 \vee c = 0 \\ \text{Disjunction of } \textit{abstract} \text{ guards} & \left\{ \begin{array}{rr} & n < d \\ \vee & n > 0 \end{array} \right\} \begin{array}{l} \textbf{guards of } \textit{ML_out} \textbf{ in } m_0 \\ \textbf{guards of } \textit{ML_in} \textbf{ in } m_0 \end{array} \\ \vdash & \\ \text{Disjunction of } \textit{concrete} \text{ guards} & \left\{ \begin{array}{rr} & a + b < d \wedge c = 0 \\ \vee & c > 0 \\ \vee & a > 0 \\ \vee & b > 0 \wedge a = 0 \end{array} \right\} \begin{array}{l} \textbf{guards of } \textit{ML_out} \textbf{ in } m_1 \\ \textbf{guards of } \textit{ML_in} \textbf{ in } m_1 \\ \textbf{guards of } \textit{IL_in} \textbf{ in } m_1 \\ \textbf{guards of } \textit{IL_out} \textbf{ in } m_1 \end{array} \end{array} \quad \textbf{DLF}$$

Example Inference Rules (6)

$$\frac{H, \neg P \vdash Q}{H \vdash P \vee Q} \ \textbf{OR_R}$$

To prove a ***disjunctive goal***, it suffices to prove one of the disjuncts, with the the negation of the the other disjunct serving as an additional hypothesis.

$$\frac{H, P, Q \vdash R}{H, P \wedge Q \vdash R} \ \textbf{AND_L}$$

To prove a goal with a ***conjunctive hypothesis***, it suffices to prove the same goal, with the the two conjuncts serving as two separate hypotheses.

$$\frac{H \vdash P \qquad H \vdash Q}{H \vdash P \wedge Q} \ \textbf{AND_R}$$

To prove a goal with a ***conjunctive goal***, it suffices to prove each conjunct as a separate goal.

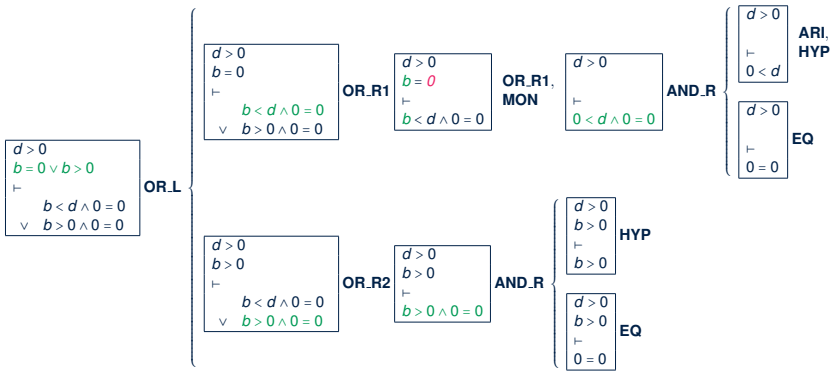
Proving Refinement: DLF of m_1

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

MON

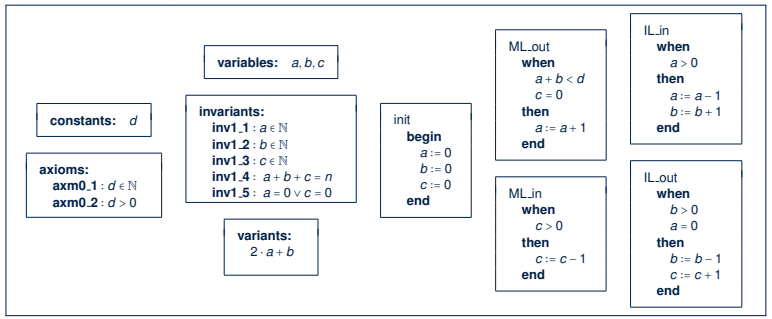
$$\begin{array}{l} d > 0 \\ a \in \mathbb{N} \\ b \in \mathbb{N} \\ c \in \mathbb{N} \\ \vdash \\ \quad a + b < d \wedge c = 0 \\ \vee \quad c > 0 \\ \vee \quad a > 0 \\ \vee \quad b > 0 \wedge a = 0 \end{array} \ \textbf{OR_R, ARI} \ \begin{array}{l} d > 0 \\ a \in \mathbb{N} \\ b \in \mathbb{N} \\ c = 0 \\ \vdash \\ \quad a + b < d \wedge c = 0 \\ \vee \quad c > 0 \\ \vee \quad a > 0 \\ \vee \quad b > 0 \wedge a = 0 \end{array} \ \textbf{EQ_LR, MON} \ \begin{array}{l} d > 0 \\ a \in \mathbb{N} \\ b \in \mathbb{N} \\ \vdash \\ \quad a + b < d \wedge 0 = 0 \\ \vee \quad 0 > 0 \\ \vee \quad a > 0 \\ \vee \quad b > 0 \wedge a = 0 \end{array} \ \textbf{OR_R, ARI} \ \begin{array}{l} d > 0 \\ a = 0 \\ b \in \mathbb{N} \\ \vdash \\ \quad a + b < d \wedge 0 = 0 \\ \vee \quad b > 0 \wedge a = 0 \end{array} \ \textbf{EQ_LR, MON} \ \begin{array}{l} d > 0 \\ b \in \mathbb{N} \\ \vdash \\ \quad 0 + b < d \wedge 0 = 0 \\ \vee \quad b > 0 \wedge 0 = 0 \end{array} \ \textbf{ARI} \ \begin{array}{l} d > 0 \\ b = 0 \vee b > 0 \\ \vdash \\ \quad b < d \wedge 0 = 0 \\ \vee \quad b > 0 \wedge 0 = 0 \end{array} \ \ldots$$

Proving Refinement: DLF of m_1 (continued)



First Refinement: Summary

- The final version of our ***first refinement*** m_1 is ***provably correct*** w.r.t.:
  - Establishment of ***Concrete Invariants*** [*init*]
  - Preservation of ***Concrete Invariants*** [old & new events]
  - Strengthening of ***guards*** [old events]
  - ***Convergence*** (a.k.a. livelock freedom, non-divergence) [new events]
  - Relative ***Deadlock*** Freedom
- Here is the final specification of m_1:



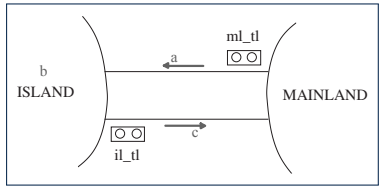
Model m_2: "More Concrete" Abstraction

- 2nd ***refinement*** has even more ***concrete*** perception of the bridge controller:
  - We "**zoom in**" by observing the system from **even closer to the ground**, so that the one-way traffic of the bridge is controlled via:

ml_tl: a traffic light for exiting the ML

il_tl: a traffic light for exiting the IL

abstract variables ***a***, ***b***, ***c*** from m_1 still used (instead of being replaced)



  - Nonetheless, sensors remain ***abstracted*** away!
- That is, we focus on these three ***environment constraints***:

ENV1	The system is equipped with two traffic lights with two colors: green and red.
ENV2	The traffic lights control the entrance to the bridge at both ends of it.
ENV3	Cars are not supposed to pass on a red traffic light, only on a green one.

- We are ***obliged to prove*** this ***added concreteness*** is ***consistent*** with m_1.

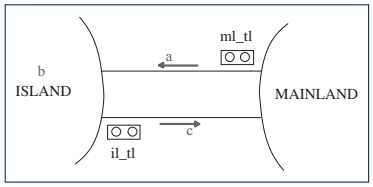
Model m_2: Refined, Concrete State Space

1. The **static** part introduces the notion of traffic light colours:

sets: *COLOR* | **constants:** *red*, *green*

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

2. The **dynamic** part shows the ***superposition refinement*** scheme:



- ***Abstract*** variables ***a***, ***b***, ***c*** from m_1 are still in use in ***m_2***.
- Two new, ***concrete*** variables are introduced: ***ml_tl*** and ***il_tl***
- **Constrast**: In m_1, ***abstract*** variable n is replaced by ***concrete*** variables a, b, c.

variables:
a, b, c
ml_tl
il_tl

invariants:
inv2_1 : $ml_tl \in COLOUR$
inv2_2 : $il_tl \in COLOUR$
inv2_3 : ??
inv2_4 : ??

- **inv2_1** & **inv2_2**: typing constraints
- **inv2_3**: being allowed to exit ML **means** cars within limit and no opposite traffic
- **inv2_4**: being allowed to exit IL **means** some car in IL and no opposite traffic

Model m_2: Refining Old, Abstract Events

- The system acts as an **ABSTRACT STATE MACHINE (ASM)**: it ***evolves*** as ***actions of enabled events*** change values of variables, subject to ***invariants***.
- ***Concrete/Refined*** version of ***event*** *ML_out*:

```
ML_out
  when
    ??
  then
    a := a + 1
  end
```

  - Recall the ***abstract*** guard of *ML_out* in m_1: $(c = 0) \wedge (a + b < d)$

    $\Rightarrow$ Unrealistic as drivers should **not** know about a, b, c!
  - *ML_out* is ***refined***: a car exits the ML (to the bridge) only when:
    - the traffic light *ml_tl* allows
- ***Concrete/Refined*** version of ***event*** *IL_out*:

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

  - Recall the ***abstract*** guard of *IL_out* in m_1: $(a = 0) \wedge (b > 0)$

    $\Rightarrow$ Unrealistic as drivers should **not** know about a, b, c!
  - *IL_out* is ***refined***: a car exits the IL (to the bridge) only when:
    - the traffic light *il_tl* allows

Q1. How about the other two "old" ***events*** *IL_in* and *ML_in*?
A1. No need to ***refine*** as already ***guarded*** by *ML_out* and *IL_out*.
Q2. What if the driver disobeys *ml_tl* or *il_tl*? [**A2**. **ENV3**]

Model m_2: New, Concrete Events

- The system acts as an **ABSTRACT STATE MACHINE (ASM)**: it ***evolves*** as ***actions of enabled events*** change values of variables, subject to ***invariants***.
- Considered ***events*** already existing in m_1:
  - *ML_out* & *IL_out* [REFINED]
  - *IL_in* & *ML_in* [UNCHANGED]
- ***New event*** *ML_tl_green*:

```
ML_tl_green
  when
    ??
  then
    ml_tl := green
  end
```

  - *ML_tl_green* denotes the traffic light *ml_tl* turning green.
  - *ML_tl_green* ***enabled*** only when:
    - the traffic light not already green
    - limited number of cars on the bridge and the island
    - No opposite traffic

    [$\Rightarrow$ *ML_out*'s ***abstract*** guard in m_1]
- ***New event*** *IL_tl_green*:

```
IL_tl_green
  when
    ??
  then
    il_tl := green
  end
```

  - *IL_tl_green* denotes the traffic light *il_tl* turning green.
  - *IL_tl_green* ***enabled*** only when:
    - the traffic light not already green
    - some cars on the island (i.e., island not empty)
    - No opposite traffic

    [$\Rightarrow$ *IL_out*'s ***abstract*** guard in m_1]

Invariant Preservation in Refinement m_2

```
constants: d
sets: COLOR
axioms:
  axm0_1 : d ∈ ℕ
  axm0_2 : d > 0
  axm2_1 : COLOR = {green, red}
  axm2_2 : green ≠ red

variables:
  a, b, c
  ml_tl
  il_tl
invariants:
  inv2_1 : ml_tl ∈ COLOUR
  inv2_2 : il_tl ∈ COLOUR
  inv2_3 : ml_tl = green ⇒ a + b < d ∧ c = 0
  inv2_4 : il_tl = green ⇒ b > 0 ∧ a = 0

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

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

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

IL_tl_green
  when
    il_tl = red
    b > 0
    a = 0
  then
    il_tl := green
  end

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

ML_in
  when
    c > 0
  then
    c := c − 1
  end
```

Recall the ***PO/VC Rule of Invariant Preservation for Refinement***:

$$\frac{\begin{array}{l} A(c) \\ I(c, v) \\ J(c, v, w) \\ H(c, w) \\ \vdash \\ J_i(c, E(c, v), F(c, w)) \end{array}}{\text{INV}}$$

where J_i denotes a single ***concrete invariant***

- How many ***sequents*** to be proved? [# ***concrete*** evts × # ***concrete*** invariants = 6 × 4]
- We discuss two sequents: *ML_out*/**inv2_4**/*INV* and *IL_out*/**inv2_3**/*INV*

Exercises. Specify and prove (some of) other twenty-two ***POs of Invariant Preservation***.

INV PO of m_2: ML_out/inv2_4/INV

axm0_1	$d \in \mathbb{N}$
axm0_2	$d > 0$
axm2_1	$COLOUR = \{green, red\}$
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	$ml_tl \in COLOUR$
inv2_2	$il_tl \in COLOUR$
inv2_3	$ml_tl = green \Rightarrow a + b < d \wedge c = 0$
inv2_4	$il_tl = green \Rightarrow b > 0 \wedge a = 0$
Concrete guards of *ML_out*	$ml_tl = green$
	$\vdash$
Concrete invariant **inv2_4** with *ML_out*'s effect in the post-state	$il_tl = green \Rightarrow b > 0 \wedge (a + 1) = 0$

ML_out/inv2_4/INV

INV PO of m_2: IL_out/inv2_3/INV

axm0_1	$d \in \mathbb{N}$
axm0_2	$d > 0$
axm2_1	$COLOUR = \{green, red\}$
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	$ml_tl \in COLOUR$
inv2_2	$il_tl \in COLOUR$
inv2_3	$ml_tl = green \Rightarrow a + b < d \wedge c = 0$
inv2_4	$il_tl = green \Rightarrow b > 0 \wedge a = 0$
Concrete guards of *IL_out*	$il_tl = green$
	$\vdash$
Concrete invariant **inv2_3** with *ML_out*'s effect in the post-state	$ml_tl = green \Rightarrow a + (b-1) < d \wedge (c+1) = 0$

IL_out/inv2_3/INV

Example Inference Rules (7)

$$\frac{H, P, Q \vdash R}{H, P, P \Rightarrow Q \vdash R}\ \textbf{IMP_L}$$

If a hypothesis ***P*** matches the assumption of another ***implicative hypothesis P ⇒ Q***, then the conclusion ***Q*** of the ***implicative hypothesis*** can be used as a new hypothesis for the sequent.

$$\frac{H, P \vdash Q}{H \vdash P \Rightarrow Q}\ \textbf{IMP_R}$$

To prove an ***implicative goal P ⇒ Q***, it suffices to prove its conclusion ***Q***, with its assumption ***P*** serving as a new hypotheses.

$$\frac{H, \neg Q \vdash P}{H, \neg P \vdash Q}\ \textbf{NOT_L}$$

To prove a goal ***Q*** with a ***negative hypothesis ¬ P***, it suffices to prove the negated hypothesis $\neg(\neg P) \equiv P$ with the negated original goal ***¬ Q*** serving as a new hypothesis.

Proving ML_out/inv2_4/INV: First Attempt

$d \in \mathbb{N}$
$d > 0$
$COLOUR = \{green, red\}$
$green \neq red$
$n \in \mathbb{N}$
$n \leq d$
$a \in \mathbb{N}$
$b \in \mathbb{N}$
$c \in \mathbb{N}$
$a + b + c = n$
$a = 0 \vee c = 0$
$ml_tl \in COLOUR$
$il_tl \in COLOUR$
$ml_tl = green \Rightarrow a + b < d \wedge c = 0$
$il_tl = green \Rightarrow b > 0 \wedge a = 0$
$ml_tl = green$
$\vdash$
$il_tl = green \Rightarrow b > 0 \wedge (a+1) = 0$

MON

$green \neq red$
$il_tl = green \Rightarrow b > 0 \wedge a = 0$
$ml_tl = green$
$il_tl = green$
$\vdash$
$il_tl = green \Rightarrow b > 0 \wedge (a+1) = 0$

IMP_R

$green \neq red$
$il_tl = green \Rightarrow b > 0 \wedge a = 0$
$ml_tl = green$
$il_tl = green$
$\vdash$
$b > 0 \wedge (a+1) = 0$

IMP_L

$green \neq red$
$b > 0 \wedge a = 0$
$ml_tl = green$
$il_tl = green$
$\vdash$
$b > 0 \wedge (a+1) = 0$

AND_L

$green \neq red$
$b > 0$
$a = 0$
$ml_tl = green$
$il_tl = green$
$\vdash$
$b > 0 \wedge (a+1) = 0$

AND_R

- $green \neq red$, $b > 0$, $a = 0$, $ml_tl = green$, $il_tl = green$, $\vdash$, $b > 0$ — **HYP**
- $green \neq red$, $b > 0$, $a = 0$, $ml_tl = green$, $il_tl = green$, $\vdash$, $(a+1) = 0$ — **EQ_LR, MON**
  - $green \neq red$, $ml_tl = green$, $il_tl = green$, $\vdash$, $(0+1) = 0$ — **ARI**
    - $green \neq red$, $ml_tl = green$, $il_tl = green$, $\vdash$, $1 = 0$ — **??**

Proving IL_out/inv2_3/INV: First Attempt

$d \in \mathbb{N}$
$d > 0$
$COLOUR = \{green, red\}$
$green \neq red$
$n \in \mathbb{N}$
$n \leq d$
$a \in \mathbb{N}$
$b \in \mathbb{N}$
$c \in \mathbb{N}$
$a + b + c = n$
$a = 0 \vee c = 0$
$ml_tl \in COLOUR$
$il_tl \in COLOUR$
$ml_tl = green \Rightarrow a + b < d \wedge c = 0$
$il_tl = green \Rightarrow b > 0 \wedge a = 0$
$il_tl = green$
$\vdash$
$ml_tl = green \Rightarrow a + (b-1) < d \wedge (c+1) = 0$

MON

$green \neq red$
$ml_tl = green \Rightarrow a + b < d \wedge c = 0$
$il_tl = green$
$\vdash$
$ml_tl = green \Rightarrow a + (b-1) < d \wedge (c+1) = 0$

IMP_R

$green \neq red$
$ml_tl = green \Rightarrow a + b < d \wedge c = 0$
$il_tl = green$
$ml_tl = green$
$\vdash$
$a + (b-1) < d \wedge (c+1) = 0$

IMP_L

$green \neq red$
$a + b < d \wedge c = 0$
$il_tl = green$
$ml_tl = green$
$\vdash$
$a + (b-1) < d \wedge (c+1) = 0$

AND_L

$green \neq red$
$a + b < d$
$c = 0$
$il_tl = green$
$ml_tl = green$
$\vdash$
$a + (b-1) < d \wedge (c+1) = 0$

AND_R

- $green \neq red$, $a + b < d$, $c = 0$, $il_tl = green$, $ml_tl = green$, $\vdash$, $a + (b-1) < d$ — **MON**
  - $a + b < d$, $\vdash$, $a + (b-1) < d$ — **ARI**
- $green \neq red$, $a + b < d$, $c = 0$, $il_tl = green$, $ml_tl = green$, $\vdash$, $(c+1) = 0$ — **EQ_LR, MON**
  - $green \neq red$, $il_tl = green$, $ml_tl = green$, $\vdash$, $(0+1) = 0$ — **ARI**
    - $green \neq red$, $il_tl = green$, $ml_tl = green$, $\vdash$, $1 = 0$ — **??**

Failed: ML_out/inv2_4/INV, IL_out/inv2_3/INV

- Our first attempts of proving ***ML_out/inv2_4/INV*** and ***IL_out/inv2_3/INV*** both failed the 2nd case (resulted from applying IR **AND_R**):

$$\boxed{green \neq red \wedge il_tl = green \wedge ml_tl = green \ \vdash \ 1 = 0}$$

- This ***unprovable*** sequent gave us a good hint:
  - Goal $\boxed{1 = 0 \equiv \textbf{false}}$ suggests that the ***safety requirements*** $a = 0$ (for **inv2_4**) and $c = 0$ (for **inv2_3**) ***contradict*** with the current m_2.
  - Hyp. $\boxed{il_tl = green = ml_tl}$ suggests a ***possible, dangerous state*** of m_2, where two cars heading different directions are on the one-way bridge:

$\langle$	$\underbrace{init}$	,	$\underbrace{ML_tl_green}$	,	$\underbrace{ML_out}$	,	$\underbrace{IL_in}$	,	$\underbrace{IL_tl_green}$	,	$\underbrace{IL_out}$	,	$\underbrace{ML_out}$	$\rangle$
	$d = 2$		$d = 2$		$d = 2$		$d = 2$		$d = 2$		$d = 2$		$d = 2$	
	$a' = 0$		$a' = 0$		**a' = 1**		**a' = 0**		$a' = 0$		$a' = 0$		**a' = 1**	
	$b' = 0$		$b' = 0$		$b' = 0$		**b' = 1**		$b' = 1$		**b' = 0**		$b' = 0$	
	$c' = 0$		$c' = 0$		$c' = 0$		$c' = 0$		$c' = 0$		**c' = 1**		$c' = 1$	
	$ml_tl' = red$		**ml_tl' = green**		$ml_tl' = green$		$ml_tl' = green$		$ml_tl' = green$		$ml_tl' = green$		$ml_tl' = green$	
	$il_tl' = red$		$il_tl' = red$		$il_tl' = red$		$il_tl' = red$		**il_tl' = green**		$il_tl' = green$		$il_tl' = green$	

Fixing m_2: Adding an Invariant

- Having understood the failed proofs, we add a proper ***invariant*** to m_2:

invariants:
. . .
inv2_5 : $ml_tl = red \vee il_tl = red$

- We have effectively resulted in an improved m_2 more faithful w.r.t. **REQ3**:

REQ3	The bridge is one-way or the other, not both at the same time.

- Having added this new invariant ***inv2_5***:
  - Original 6×4 generated sequents to be updated: **inv2_5** a new hypothesis e.g., Are ***ML_out/inv2_4/INV*** and ***IL_out/inv2_3/INV*** now ***provable***?
  - Additional 6×1 sequents to be generated due to this new invariant e.g., Are ***ML_tl_green/inv2_5/INV*** and ***IL_tl_green/inv2_5/INV*** ***provable***?

INV PO of m_2: ML_out/inv2_4/INV – Updated

axm0_1	$d \in \mathbb{N}$
axm0_2	$d > 0$
axm2_1	$COLOUR = \{green, red\}$
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	$ml_tl \in COLOUR$
inv2_2	$il_tl \in COLOUR$
inv2_3	$ml_tl = green \Rightarrow a + b < d \wedge c = 0$
inv2_4	$il_tl = green \Rightarrow b > 0 \wedge a = 0$
inv2_5	$ml_tl = red \vee il_tl = red$
Concrete guards of *ML_out*	$ml_tl = green$
	$\vdash$
Concrete invariant **inv2_4** with *ML_out*'s effect in the post-state	$il_tl = green \Rightarrow b > 0 \wedge (a + 1) = 0$

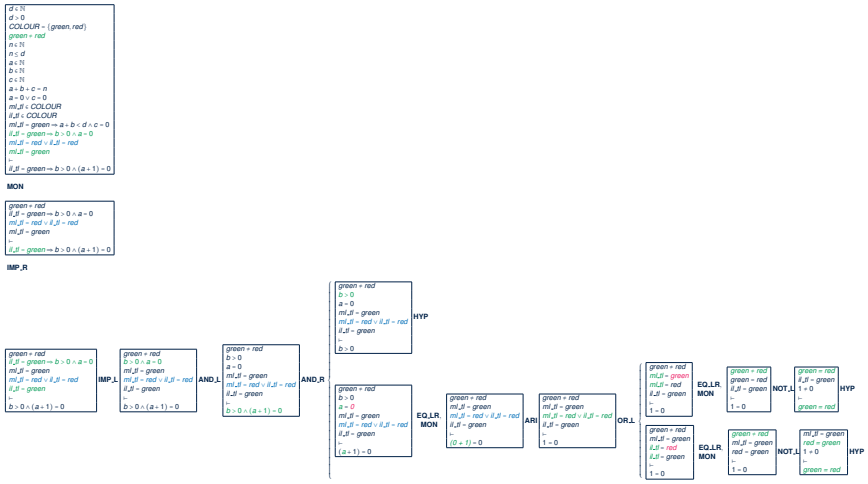
ML_out/inv2_4/INV

INV PO of m_2: IL_out/inv2_3/INV – Updated

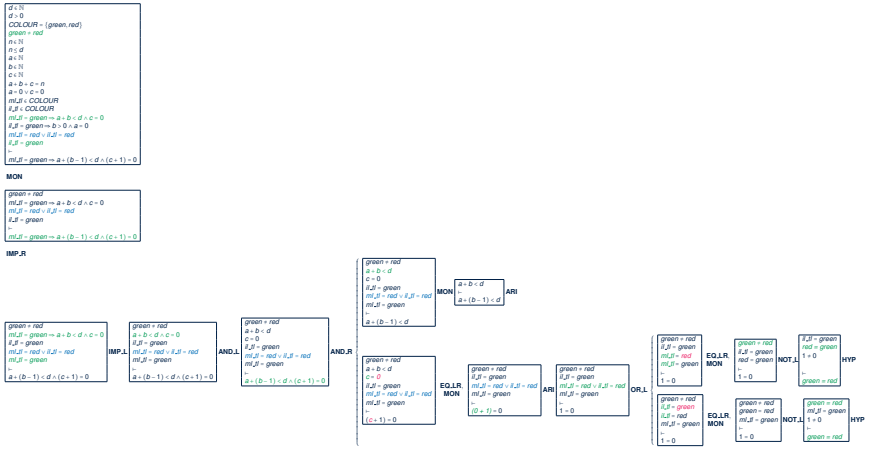
axm0_1	$d \in \mathbb{N}$
axm0_2	$d > 0$
axm2_1	$COLOUR = \{green, red\}$
axm2_2	$green \neq red$
inv0_1	$n \in \mathbb{N}$
inv0_2	$n \leq d$
inv1_1	$a \in \mathbb{N}$
inv1_2	$b \in \mathbb{N}$
inv1_3	$c \in \mathbb{N}$
inv1_4	$a + b + c = n$
inv1_5	$a = 0 \vee c = 0$
inv2_1	$ml_tl \in COLOUR$
inv2_2	$il_tl \in COLOUR$
inv2_3	$ml_tl = green \Rightarrow a + b < d \wedge c = 0$
inv2_4	$il_tl = green \Rightarrow b > 0 \wedge a = 0$
inv2_5	$ml_tl = red \vee il_tl = red$
Concrete guards of *IL_out*	$il_tl = green$
	$\vdash$
Concrete invariant **inv2_3** with *ML_out*'s effect in the post-state	$ml_tl = green \Rightarrow a + (b - 1) < d \wedge (c + 1) = 0$

IL_out/inv2_3/INV

Proving ML_out/inv2_4/INV: Second Attempt



Proving IL_out/inv2_3/INV: Second Attempt



Fixing m_2: Adding Actions

- Recall that an ***invariant*** was added to m_2:

invariants:
inv2_5 : $ml_tl = red \lor il_tl = red$

- Additional 6 × 1 sequents to be generated due to this new invariant:
  - e.g., ***ML_tl_green*/inv2_5/INV** [for ***ML_tl_green*** to preserve **inv2_5**]
  - e.g., ***IL_tl_green*/inv2_5/INV** [for ***IL_tl_green*** to preserve **inv2_5**]
- For the above ***sequents*** to be ***provable***, we need to revise the two events:

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

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

Exercise: Specify and prove ***ML_tl_green*/inv2_5/INV** & ***IL_tl_green*/inv2_5/INV**.

INV PO of m_2: ML_out/inv2_3/INV

ML_out/inv2_3/INV

axm0_1	$d \in \mathbb{N}$
axm0_2	$d > 0$
axm2_1	$COLOUR = \{green, red\}$
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	$a = 0 \lor c = 0$
inv2_1	$ml_tl \in COLOUR$
inv2_2	$il_tl \in COLOUR$
inv2_3	$ml_tl = green \Rightarrow a + b < d \land c = 0$
inv2_4	$il_tl = green \Rightarrow b > 0 \land a = 0$
inv2_5	$ml_tl = red \lor il_tl = red$
Concrete guards of *ML_out*	$ml_tl = green$
	$\vdash$
Concrete invariant **inv2_3** with *ML_out*'s effect in the post-state	$ml_tl = green \Rightarrow (a + 1) + b < d \land c = 0$

Proving ML_out/inv2_3/INV: First Attempt

$d \in \mathbb{N}$
$d > 0$
$COLOUR = \{green, red\}$
$green \neq red$
$n \in \mathbb{N}$
$n \leq d$
$a \in \mathbb{N}$
$b \in \mathbb{N}$
$c \in \mathbb{N}$
$a + b + c = n$
$a = 0 \lor c = 0$
$ml_tl \in COLOUR$
$il_tl \in COLOUR$
$ml_tl = green \Rightarrow a + b < d \land c = 0$
$il_tl = green \Rightarrow b > 0 \land a = 0$
$ml_tl = red \lor il_tl = red$
$ml_tl = green$
$\vdash$
$ml_tl = green \Rightarrow (a + 1) + b < d \land c = 0$

MON

$ml_tl = green \Rightarrow a + b < d \land c = 0$
$\vdash$
$ml_tl = green \Rightarrow (a + 1) + b < d \land c = 0$

IMP_R

$ml_tl = green \Rightarrow a + b < d \land c = 0$
$ml_tl = green$
$\vdash$
$(a + 1) + b < d \land c = 0$

IMP_R

$a + b < d \land c = 0$
$ml_tl = green$
$\vdash$
$(a + 1) + b < d \land c = 0$

AND_L

$a + b < d$
$c = 0$
$ml_tl = green$
$\vdash$
$(a + 1) + b < d \land c = 0$

AND_R

- $a + b < d$, $c = 0$, $ml_tl = green$, $\vdash$ $(a + 1) + b < d$ — **??**
- $a + b < d$, $c = 0$, $ml_tl = green$, $\vdash$ $c = 0$ — **HYP**

Failed: ML_out/inv2_3/INV

- Our first attempt of proving ***ML_out/inv2_3/INV*** failed the 1st case (resulted from applying IR **AND_R**):

$$a + b < d \land c = 0 \land ml_tl = green \vdash (a + 1) + b < d$$

- This ***unprovable*** sequent gave us a good hint:
  - Goal $\underbrace{(a + 1)}_{a'} + \underbrace{b}_{b'} < d$ specifies the ***capacity requirement***.
  - Hypothesis $c = 0 \land ml_tl = green$ assumes that it's safe to exit the ML.
  - Hypothesis $a + b < d$ is **not** strong enough to entail $(a + 1) + b < d$.
    - e.g., $d = 3, b = 0, a = 0$ [$(a + 1) + b < d$ evaluates to ***true***]
    - e.g., $d = 3, b = 1, a = 0$ [$(a + 1) + b < d$ evaluates to ***true***]
    - e.g., $d = 3, b = 0, a = 1$ [$(a + 1) + b < d$ evaluates to ***true***]
    - e.g., $d = 3, b = 0, a = 2$ [$(a + 1) + b < d$ evaluates to ***false***]
    - e.g., $d = 3, b = 1, a = 1$ [$(a + 1) + b < d$ evaluates to ***false***]
    - e.g., $d = 3, b = 2, a = 0$ [$(a + 1) + b < d$ evaluates to ***false***]
  - Therefore, $a + b < d$ (allowing one more car to exit ML) should be split:
    - $a + b + 1 \neq d$ [more later cars may exit ML, ***ml_tl*** remains ***green***]
    - $a + b + 1 = d$ [no more later cars may exit ML, ***ml_tl*** turns ***red***]

Fixing m_2: Splitting *ML_out* and *IL_out*

- Recall that ***ML_out/inv2_3/INV*** failed ∵ two cases not handled separately:
  - $a + b + 1 \neq d$ [more later cars may exit ML, ***ml_tl*** remains ***green***]
  - $a + b + 1 = d$ [no more later cars may exit ML, ***ml_tl*** turns ***red***]
- Similarly, ***IL_out/inv2_4/INV*** would fail ∵ two cases not handled separately:
  - $b - 1 \neq 0$ [more later cars may exit IL, ***il_tl*** remains ***green***]
  - $b - 1 = 0$ [no more later cars may exit IL, ***il_tl*** turns ***red***]
- Accordingly, we split ***ML_out*** and ***IL_out*** into two with corresponding guards.

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

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

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

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

Exercise: Given the latest m_2, how many sequents to prove for ***invariant preservation***?
Exercise: Specify and prove ***ML_out_i*/inv2_3/INV** & ***IL_out_i*/inv2_4/INV** (where $i \in 1 \,..\, 2$).
Exercise: Each split event (e.g., *ML_out_1*) refines its ***abstract*** counterpart (e.g., *ML_out*)?

m_2 Livelocks: New Events Diverging

- Recall that a system may ***livelock*** if the new events diverge.
- Current m_2's two new events ***ML_tl_green*** and ***IL_tl_green*** may *diverge* :

```
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
    b > 0
    a = 0
  then
    il_tl := green
    ml_tl := red
  end
```

- *ML_tl_green* and *IL_tl_green* both ***enabled*** and may occur ***indefinitely***, preventing other "old" events (e.g., *ML_out*) from ever happening:

$\langle$ *init*	*ML_tl_green*	*ML_out_1*	*IL_in*	*IL_tl_green*	*ML_tl_green*	*IL_tl_green* $, \ldots \rangle$
$d = 2$	$d = 2$	$d = 2$	$d = 2$	$d = 2$	$d = 2$	$d = 2$
$a' = 0$	$a' = 0$	$a' = 1$	$a' = 0$	$a' = 0$	$a' = 0$	$a' = 0$
$b' = 0$	$b' = 0$	$b' = 0$	$b' = 1$	$b' = 1$	$b' = 1$	$b' = 1$
$c' = 0$	$c' = 0$	$c' = 0$	$c' = 0$	$c' = 0$	$c' = 0$	$c' = 0$
$ml_tl = red$	$ml_tl' = green$	$ml_tl' = green$	$ml_tl' = green$	$ml_tl' = red$	$ml_tl' = green$	$ml_tl' = red$
$il_tl = red$	$il_tl' = red$	$il_tl' = red$	$il_tl' = red$	$il_tl' = green$	$il_tl' = red$	$il_tl' = green$

$\Rightarrow$ Two traffic lights keep changing colors so rapidly that **no** drivers can ever pass!

- **Solution**: Allow color changes between traffic lights in a disciplined way.

Fixing m_2: Regulating Traffic Light Changes

We introduce two variables/flags for regulating traffic light changes:

- ***ml_pass*** is ***1*** **if**, since ***ml_tl*** was last turned ***green***, at least one car exited the ML onto the bridge. Otherwise, ***ml_pass*** is ***0***.
- ***il_pass*** is ***1*** **if**, since ***il_tl*** was last turned ***green***, at least one car exited the IL onto the bridge. Otherwise, ***il_pass*** is ***0***.

variables: ml_pass, il_pass

invariants:
- **inv2_6** : $ml_pass \in \{0, 1\}$
- **inv2_7** : $il_pass \in \{0, 1\}$
- **inv2_8** : $ml_tl = red \Rightarrow ml_pass = 1$
- **inv2_9** : $il_tl = red \Rightarrow il_pass = 1$

ML_out_1
when
$ml_tl = green$
$a + b + 1 \neq d$
then
$a := a + 1$
$ml_pass := 1$
end

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

IL_out_1
when
$il_tl = green$
$b \neq 1$
then
$b := b - 1$
$c := c + 1$
$il_pass := 1$
end

IL_out_2
when
$il_tl = green$
$b = 1$
then
$b := b - 1$
$c := c + 1$
$il_tl := red$
$il_pass := 1$
end

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

IL_tl_green
when
$il_tl = red$
$b > 0$
$a = 0$
$ml_pass = 1$
then
$il_tl := green$
$ml_tl := red$
$il_pass := 0$
end

Fixing m_2: Measuring Traffic Light Changes

- Recall:
  - Interleaving of ***new*** events charactered as an integer expression: ***variant***.
  - A variant $V(c, w)$ may refer to constants and/or ***concrete*** variables.
  - In the latest m_2, let's try **variants** : $ml_pass + il_pass$
- Accordingly, for the ***new*** event *ML_tl_green*:

$d \in \mathbb{N}$	$d > 0$	
$COLOUR = \{green, red\}$	$green \neq red$	
$n \in \mathbb{N}$	$n \leq d$	
$a \in \mathbb{N}$	$b \in \mathbb{N}$	$c \in \mathbb{N}$
$a + b + c = n$	$a = 0 \vee c = 0$	
$ml_tl \in COLOUR$	$il_tl \in COLOUR$	
$ml_tl = green \Rightarrow a + b < d \wedge c = 0$	$il_tl = green \Rightarrow b > 0 \wedge a = 0$	
$ml_tl = red \vee il_tl = red$		
$ml_pass \in \{0, 1\}$	$il_pass \in \{0, 1\}$	
$ml_tl = red \Rightarrow ml_pass = 1$	$il_tl = red \Rightarrow il_pass = 1$	
$ml_tl = red$	$a + b < d$	$c = 0$
$il_pass = 1$		
$\vdash$		
$0 + il_pass < ml_pass + il_pass$		

ML_tl_green/VAR

Exercises: Prove **ML_tl_green/VAR** and Formulate/Prove **IL_tl_green/NAT**.

PO Rule: Relative Deadlock Freedom of m_2

axm0_1	$d \in \mathbb{N}$	
axm0_2	$d > 0$	
axm2_1	$COLOUR = \{green, red\}$	
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	$ml_tl \in COLOUR$	
inv2_2	$il_tl \in COLOUR$	
inv2_3	$ml_tl = green \Rightarrow a + b < d \wedge c = 0$	
inv2_4	$il_tl = green \Rightarrow b > 0 \wedge a = 0$	
inv2_5	$ml_tl = red \vee il_tl = red$	
inv2_6	$ml_pass \in \{0, 1\}$	
inv2_7	$il_pass \in \{0, 1\}$	
inv2_8	$ml_tl = red \Rightarrow ml_pass = 1$	
inv2_9	$il_tl = red \Rightarrow il_pass = 1$	
Disjunction of ***abstract*** guards	$a + b < d \wedge c = 0$	**guards of** *ML_out* **in** m_1
	$\vee\ c > 0$	**guards of** *ML_in* **in** m_1
	$\vee\ a > 0$	**guards of** *IL_in* **in** m_1
	$\vee\ b > 0 \wedge a = 0$	**guards of** *IL_out* **in** m_1
	$\vdash$	
Disjunction of ***concrete*** guards	$ml_tl = red \wedge a + b < d \wedge c = 0 \wedge il_pass = 1$	**guards of** *ML_tl_green* **in** m_2
	$\vee\ il_tl = red \wedge b > 0 \wedge a = 0 \wedge ml_pass = 1$	**guards of** *IL_tl_green* **in** m_2
	$\vee\ ml_tl = green \wedge a + b + 1 \neq d$	**guards of** *ML_out_1* **in** m_2
	$\vee\ ml_tl = green \wedge a + b + 1 = d$	**guards of** *ML_out_2* **in** m_2
	$\vee\ il_tl = green \wedge b \neq 1$	**guards of** *IL_out_1* **in** m_2
	$\vee\ il_tl = green \wedge b = 1$	**guards of** *IL_out_2* **in** m_2
	$\vee\ c > 0$	**guards of** *ML_in* **in** m_2
	$\vee\ a > 0$	**guards of** *IL_in* **in** m_2

DLF

Proving Refinement: DLF of m_2

$d \in \mathbb{N}$
$d > 0$
$COLOUR = \{green, red\}$
$green \neq red$
$n \in \mathbb{N}$
$n \leq d$
$a \in \mathbb{N}$
$b \in \mathbb{N}$
$c \in \mathbb{N}$
$a + b + c = n$
$a = 0 \vee c = 0$
$ml_tl \in COLOUR$
$il_tl \in COLOUR$
$ml_tl = green \Rightarrow a + b < d \wedge c = 0$
$il_tl = green \Rightarrow b > 0 \wedge a = 0$
$ml_tl = red \vee il_tl = red$
$ml_pass \in \{0, 1\}$
$il_pass \in \{0, 1\}$
$ml_tl = red \Rightarrow ml_pass = 1$
$il_tl = red \Rightarrow il_pass = 1$
$a + b < d \wedge c = 0$
$\vee\ c > 0$
$\vee\ a > 0$
$\vee\ b > 0 \wedge a = 0$
$\vdash$
$ml_tl = red \wedge a + b < d \wedge c = 0 \wedge il_pass = 1$
$\vee\ il_tl = red \wedge b > 0 \wedge a = 0 \wedge ml_pass = 1$
$\vee\ ml_tl = green$
$\vee\ il_tl = green$
$\vee\ a > 0$
$\vee\ c > 0$

⋮

$d \in \mathbb{N}$, $d > 0$, $b \in \mathbb{N}$, $ml_tl = red$, $il_tl = red$, $ml_tl = red \Rightarrow ml_pass = 1$, $il_tl = red \Rightarrow il_pass = 1$, $\vdash$, $b < d \wedge ml_pass = 1 \wedge il_pass = 1$, $\vee\ b > 0 \wedge ml_pass = 1 \wedge il_pass = 1$ ⋯

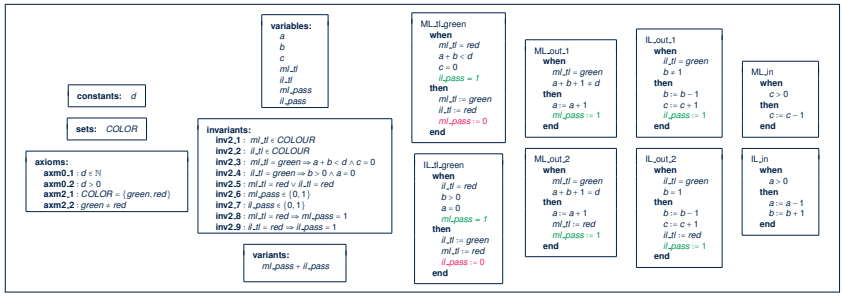
$d \in \mathbb{N}$, $d > 0$, $b \in \mathbb{N}$, $ml_tl = red$, $il_tl = red$, $ml_pass = 1$, $il_pass = 1$, $\vdash$, $b < d \wedge ml_pass = 1 \wedge il_pass = 1$, $\vee\ b > 0 \wedge ml_pass = 1 \wedge il_pass = 1$ ⋯

$d > 0$, $b \in \mathbb{N}$, $\vdash$, $b < d \vee b > 0$ — **ARI** → $d > 0$, $b > 0 \vee b = 0$, $\vdash$, $b < d \vee b > 0$ — **OR_L**

- $d > 0$, $b > 0$, $\vdash$, $b < d \vee b > 0$ — **OR_R2** → $d > 0$, $b > 0$, $\vdash$, $b > 0$ — **HYP**
- $d > 0$, $b = 0$, $\vdash$, $b < d \vee b > 0$ — **EQ_LR, MON** → $d > 0$, $\vdash$, $0 < d \vee 0 > 0$ — **OR_R1** → $d > 0$, $\vdash$, $0 < d$ — **HYP**

Second Refinement: Summary

- The final version of our ***second refinement*** m_2 is ***provably correct*** w.r.t.:
  - Establishment of ***Concrete Invariants*** [*init*]
  - Preservation of ***Concrete Invariants*** [old & new events]
  - Strengthening of ***guards*** [old events]
  - ***Convergence*** (a.k.a. livelock freedom, non-divergence) [new events]
  - Relative ***Deadlock*** Freedom
- Here is the final specification of m_2:



Index (1)

Learning Outcomes
Correct by Construction
State Space of a Model
Roadmap of this Module
Requirements Document: Mainland, Island
Requirements Document: E-Descriptions
Requirements Document: R-Descriptions
Requirements Document: Visual Summary of Equipment Pieces
Refinement Strategy
Model m_0: Abstraction

Index (2)

Model m_0: State Space
Model m_0: State Transitions via Events
Model m_0: Actions vs. Before-After Predicates
Design of Events: Invariant Preservation
Sequents: Syntax and Semantics
PO of Invariant Preservation: Sketch
PO of Invariant Preservation: Components
Rule of Invariant Preservation: Sequents
Inference Rules: Syntax and Semantics
Proof of Sequent: Steps and Structure
Example Inference Rules (1)

Index (3)

Example Inference Rules (2)
Example Inference Rules (3)
Revisiting Design of Events: *ML_out*
Revisiting Design of Events: *ML_in*
Fixing the Design of Events
Revisiting Fixed Design of Events: *ML_out*
Revisiting Fixed Design of Events: *ML_in*
Initializing the Abstract System m_0
PO of Invariant Establishment
Discharging PO of Invariant Establishment
System Property: Deadlock Freedom

Index (4)

PO of Deadlock Freedom (1)
PO of Deadlock Freedom (2)
Example Inference Rules (4)
Example Inference Rules (5)
Discharging PO of DLF: Exercise
Discharging PO of DLF: First Attempt
Why Did the DLF PO Fail to Discharge?
Fixing the Context of Initial Model
Discharging PO of DLF: Second Attempt
Initial Model: Summary
Model m_1: "More Concrete" Abstraction

Index (5)

Model m_1: Refined State Space
Model m_1: State Transitions via Events
Model m_1: Actions vs. Before-After Predicates
States & Invariants: Abstract vs. Concrete
Events: Abstract vs. Concrete
PO of Refinement: Components (1)
PO of Refinement: Components (2)
PO of Refinement: Components (3)
Sketching PO of Refinement
Refinement Rule: Guard Strengthening
PO Rule: Guard Strengthening of *ML_out*

Index (6)

PO Rule: Guard Strengthening of *ML_in*
Proving Refinement: ML_out/GRD
Proving Refinement: ML_in/GRD
Refinement Rule: Invariant Preservation
Visualizing Inv. Preservation in Refinement
INV PO of m_1: ML_out/inv1_4/INV
INV PO of m_1: ML_in/inv1_5/INV
Proving Refinement: ML_out/inv1_4/INV
Proving Refinement: ML_in/inv1_5/INV
Initializing the Refined System m_1
PO of m_1 Concrete Invariant Establishment

Index (7)

Discharging PO of m_1 Concrete Invariant Establishment
Model m_1: New, Concrete Events
Model m_1: BA Predicates of Multiple Actions
Visualizing Inv. Preservation in Refinement
Refinement Rule: Invariant Preservation
INV PO of m_1: IL_in/inv1_4/INV
INV PO of m_1: IL_in/inv1_5/INV
Proving Refinement: IL_in/inv1_4/INV
Proving Refinement: IL_in/inv1_5/INV
Livelock Caused by New Events Diverging

Index (8)

PO of Convergence of New Events

PO of Convergence of New Events: NAT

PO of Convergence of New Events: VAR

Convergence of New Events: Exercise

PO of Refinement: Deadlock Freedom

PO Rule: Relative Deadlock Freedom of m_1

Example Inference Rules (6)

Proving Refinement: DLF of m_1

Proving Refinement: DLF of m_1 (continued)

First Refinement: Summary

Model m_2: "More Concrete" Abstraction

Index (9)

Model m_2: Refined, Concrete State Space

Model m_2: Refining Old, Abstract Events

Model m_2: New, Concrete Events

Invariant Preservation in Refinement m_2

INV PO of m_2: ML_out/inv2_4/INV

INV PO of m_2: IL_out/inv2_3/INV

Example Inference Rules (7)

Proving ML_out/inv2_4/INV: First Attempt

Proving IL_out/inv2_3/INV: First Attempt

Failed: ML_out/inv2_4/INV, IL_out/inv2_3/INV

Fixing m_2: Adding an Invariant

Index (10)

INV PO of m_2: ML_out/inv2_4/INV – Updated

INV PO of m_2: IL_out/inv2_3/INV – Updated

Proving ML_out/inv2_4/INV: Second Attempt

Proving IL_out/inv2_3/INV: Second Attempt

Fixing m_2: Adding Actions

INV PO of m_2: ML_out/inv2_3/INV

Proving ML_out/inv2_3/INV: First Attempt

Failed: ML_out/inv2_3/INV

Fixing m_2: Splitting *ML_out* and *IL_out*

m_2 Livelocks: New Events Diverging

Fixing m_2: Regulating Traffic Light Changes

Index (11)

Fixing m_2: Measuring Traffic Light Changes

PO Rule: Relative Deadlock Freedom of m_2

Proving Refinement: DLF of m_2

Second Refinement: Summary