Specifying & Refining a Bridge Controller

MEB: Chapter 2

EECS3342 E: System Specification and Refinement
Fall 2025

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***

Recall: 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 ↑ ⇒ (State Space ↑ ∧ Verification Difficulty ↑)
- A model's ***complexity*** should be guided by those properties intended to be verified against that model.

  ⇒ ***Infeasible*** to prove **all** desired properties on a model.

  ⇒ ***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.

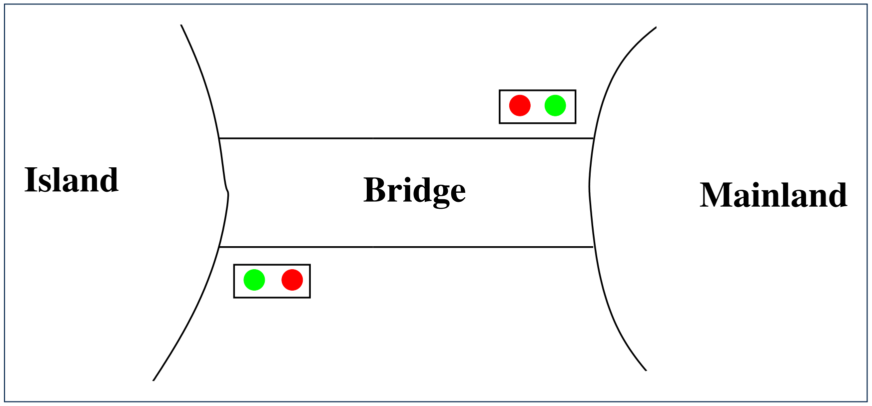
ENV1	The system is equipped with two traffic lights with two colors: green and red.
ENV2	The traffic lights control the entrance to the bridge at both ends of it.
ENV3	Cars are not supposed to pass on a red traffic light, only on a green one.
ENV4	The system is equipped with four sensors with two states: 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* :

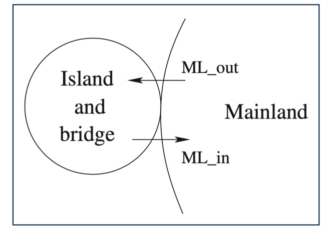
  From each ***model*** to its ***refinement***, only a manageable amount of details are added, making it ***feasible*** to conduct **analysis** and **proofs**.

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                ML_in
  begin                 begin
    n := n + 1            n := n − 1
  end                   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.

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

- 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. $\vdash G \equiv true \vdash G$ [Why not $\vdash G \equiv false \vdash G$?]

PO of Invariant Preservation: Sketch

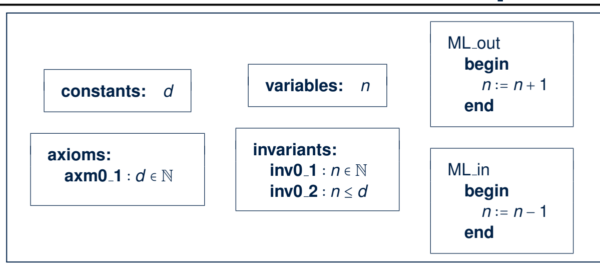
- Here is a sketch of the PO/VC rule for *invariant preservation*:

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

- 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*** $\langle$**axm0_1**$\rangle$
- v and v': list of ***variables*** in ***pre***- and ***post***-states $v \widehat{=} \langle n \rangle$, $v' \widehat{=} \langle n' \rangle$
- $I(c, v)$: 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* $\widehat{=} \langle \textbf{true} \rangle$, $G(\langle d \rangle, \langle n \rangle)$ of *ML_in* $\widehat{=} \langle \textbf{true} \rangle$
- $E(c, v)$: effect of the ***event***'s actions i.t.o. what variable values ***become***

  $E(\langle d \rangle, \langle n \rangle)$ of *ML_out* $\widehat{=} \langle n + 1 \rangle$, $E(\langle d \rangle, \langle n \rangle)$ of *ML_out* $\widehat{=} \langle n - 1 \rangle$
- $v' = E(c, v)$: ***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:

$$\frac{\begin{array}{l}d \in \mathbb{N}\\ n \in \mathbb{N}\\ n \le d\\ \vdash\\ n + 1 \in \mathbb{N}\end{array}}{}\ \text{ML\_out/\textbf{inv0\_1}/INV} \qquad \frac{\begin{array}{l}d \in \mathbb{N}\\ n \in \mathbb{N}\\ n \le d\\ \vdash\\ n + 1 \le d\end{array}}{}\ \text{ML\_out/\textbf{inv0\_2}/INV}$$

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***):

$$\begin{array}{l} d \in \mathbb{N} \\ n \in \mathbb{N} \\ n \le d \\ \vdash \\ n+1 \in \mathbb{N} \end{array} \quad \underline{\text{ML\_out/}\textbf{inv0\_1}\text{/INV}}$$

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**:

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

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.

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

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

Example Inference Rules (2)

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

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

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

$n-1$ 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.

$$\frac{H, P \vdash R \qquad H, Q \vdash R}{H, P \lor Q \vdash R} \ \textbf{OR\_L}$$

<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.

$$\frac{H \vdash P}{H \vdash P \lor Q} \ \textbf{OR\_R1}$$

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

$$\frac{H \vdash Q}{H \vdash P \lor Q} \ \textbf{OR\_R2}$$

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:

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

∴ ***ML_out/inv0_1/INV*** succeeds in being discharged.

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

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

∴ ***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*:

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

∴ ***ML_in/inv0_1/INV*** fails to be discharged.

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

$$\begin{array}{|l|}\hline d \in \mathbb{N} \\ n \in \mathbb{N} \\ n \le d \\ \vdash \\ n-1 \le d \\ \hline \end{array} \quad \textbf{MON} \quad \begin{array}{|l|}\hline n \le d \\ \vdash \\ n-1 < d \lor n-1 = d \\ \hline \end{array} \quad \textbf{OR\_1} \quad \begin{array}{|l|}\hline n \le d \\ \vdash \\ n-1 < d \\ \hline \end{array} \quad \textbf{DEC}$$

∴ ***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                ML_in
  when                  when
    n < d                 n > 0
  then                  then
    n := n + 1            n := n - 1
  end                   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*:

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

∴ ***ML_out/inv0_1/INV*** still succeeds in being discharged!

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

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

∴ ***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*:

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

∴ ***ML_in/inv0_1/INV*** now succeeds in being discharged!

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

$d \in \mathbb{N}$ $n \in \mathbb{N}$ $n \leq d$ $n > 0$ $\vdash$ $n - 1 \leq d$	**MON**	$n \leq d$ $\vdash$ $n - 1 < d \lor n - 1 = d$	**OR_1**	$n \leq d$ $\vdash$ $n - 1 < d$	**DEC**

∴ ***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 ≈ ***establishing*** inv.]
  - $P(n) \Rightarrow P(n+1)$ [***inductive*** cases ≈ ***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*.

∴ 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***: $n' = 0$

PO of Invariant Establishment

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

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

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

∵ 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 establishment*** is formulated as a ***sequent***:

Axioms $\vdash$ *Invariants* Satisfied at *Post-State*	INV	$A(c)$ $\vdash$ $I_i(c, \mathbf{K(c)})$	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:

$d \in \mathbb{N}$ $\vdash$ $0 \in \mathbb{N}$	init/inv0_1/INV	$d \in \mathbb{N}$ $\vdash$ $0 \leq d$	init/inv0_2/INV

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

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

∴ ***init/inv0_1/INV*** succeeds in being discharged.

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

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

∴ ***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***.
    ⇒ 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 $\mathbf{v} \mathrel{\hat=} \langle n \rangle$, $\mathbf{v'} \mathrel{\hat=} \langle n' \rangle$
  - $I(c, v)$: 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* $\mathrel{\hat=} \langle n < d \rangle$, $G(\langle d \rangle, \langle n \rangle)$ of *ML_in* $\mathrel{\hat=} \langle n > 0 \rangle$
- A system is ***deadlock-free*** if at least one of its ***events*** is ***enabled***:

Axioms *Invariants* Satisfied at *Pre-State* $\vdash$ Disjunction of the guards satisfied at *Pre-State*	DLF	$A(c)$ $I(c, \mathbf{v})$ $\vdash$ $G_1(c, \mathbf{v}) \vee \cdots \vee G_m(c, \mathbf{v})$	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.
  ⇒ 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:

$A(c)$ $I(c, \mathbf{v})$ $\vdash$ $G_1(c, \mathbf{v}) \vee \cdots \vee G_m(c, \mathbf{v})$	DLF	$d \in \mathbb{N}$ $n \in \mathbb{N}$ $n \le d$ $\vdash$ $n < d \vee n > 0$	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)

$\overline{H, P \vdash P}$ **HYP**	A goal is proved if it can be assumed.
$\overline{\bot \vdash P}$ **FALSE_L**	Assuming ***false*** ($\bot$), anything can be proved.
$\overline{P \vdash \top}$ **TRUE_R**	***true*** ($\top$) is proved, regardless of the assumption.
$\overline{P \vdash E = E}$ **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

$A(c)$
$I(c, v)$
$\vdash$
$G_1(c, v) \vee \cdots \vee G_m(c, v)$

DLF

$d \in \mathbb{N}$
$n \in \mathbb{N}$
$n \leq d$
$\vdash$
$n < d \vee n > 0$

??

Discharging PO of DLF: First Attempt

$d \in \mathbb{N}$, $n \in \mathbb{N}$, $n \leq d$ $\vdash$ $n < d \vee n > 0$

$\equiv$

$d \in \mathbb{N}$, $n \in \mathbb{N}$, $n < d \vee n = d$ $\vdash$ $n < d \vee n > 0$ **MON** $n < d \vee n = d$ $\vdash$ $n < d \vee n > 0$ **OR_L**

- $n < d$ $\vdash$ $n < d \vee n > 0$ **OR_R1** $n < d$ $\vdash$ $n < d$ **HYP**
- $n = d$ $\vdash$ $n < d \vee n > 0$ **EQ_LR, MON** $\vdash$ $d < d \vee d > 0$ **OR_R2** $\vdash$ $d > 0$ **?**

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:

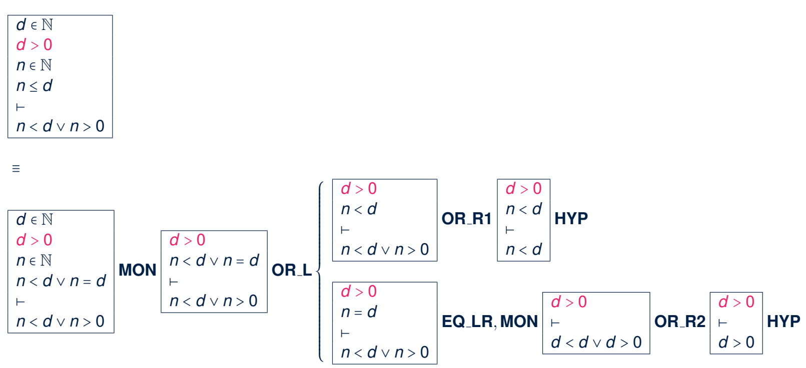
> **axioms:**
> **axm0_2** : $d > 0$

- 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



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 : $n \leq d$

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

Index (1)

Index (2)

Index (3)

Index (4)