LMF Laboratoire Méthodes Formelles

CentraleSupélec | université PARIS-SACLAY

ATELIER/TUTO EVENT-B/RODIN

INTRODUCTION À LA MÉTHODE EVENT-B ET SES DIFFÉRENTS OUTILS

🎓 TAPAS-ANR meeting

🏛 Laboratoire Méthodes Formelles - LMF, Paris-Saclay, 19 November 2025

Idir AIT SADOUNE
idir.aitsadoune@centralesupelec.fr

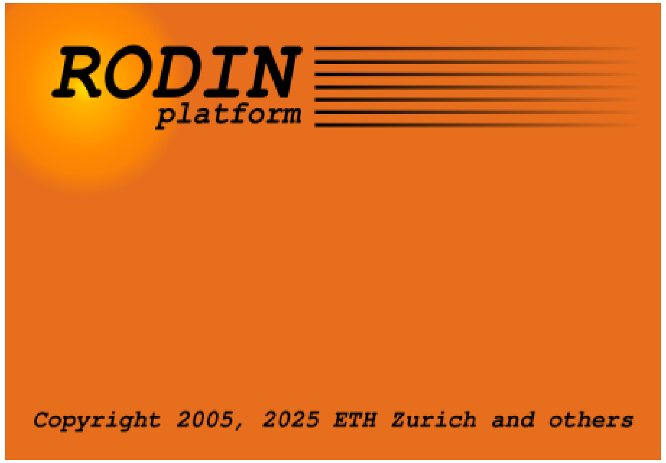
OUTLINE

- The Event-B method
- The Pro-B animator/model-checker
- The Theory plugin

Back to the outline - Back to the begin

THE RODIN PLATFORM

- The **Rodin** platform (an Eclipse-based IDE) is intended to support the construction and verification of **Event-B models**.
  - provides effective support for refinement and mathematical proof.
  - plugins for éditing models, generating proof obligations, proving, animating, medel-cheking, code generating ...



- **Rodin Platform and Plug-in Installation**:
  - Requires **Java JRE** (version 17 or later) → www.oracle.com/fr/java/.
  - Download the Core → sourceforge.net/projects/rodin-b-sharp/.

RODIN ON MACS

Procedure to run the Intel version of **Rodin** on **macs** with Apple Silicon processors:

1. download this JDK (it's a **Java 17** runtime for **Intel**)
2. install it by double-clicking it; the Java runtime is installed in `/Library/Java/JavaVirtualMachines/temurin-17.jre`
3. find the downloaded `Rodin.app` and modify the file `Rodin.app/Contents/Eclipse/rodin.ini`
   - add the next two lines just before the one with `-vmargs`

```
-vm
/Library/Java/JavaVirtualMachines/temurin-17.jre/Contents/Home/bin/java
```

4. as with all other **Rodin** releases for **mac**, one also needs to execute

```
$ xattr -rc Rodin.app
```

THE RODIN PLATFORM

Required plugins for this tutorial :

menu : `Help -> Install New Software ...`

- the **Atelier B Provers plugin** from the **Atelier B Provers** Update site.
  `https://www.atelierb.eu/update_site/atelierb_provers`
- the **ProB plugin** from the **ProB** Update site.
  `https://stups.hhu-hosting.de/rodin/prob1/release/`
- the **Theory plugin** from the **Rodin Plug-ins (archive)** Update site.
  `https://rodin-b-sharp.sourceforge.net/updates-archive`

OUTLINE

- The Event-B method
- The Pro-B animator/model-checker
- The Theory plugin

Back to the outline - Back to the begin

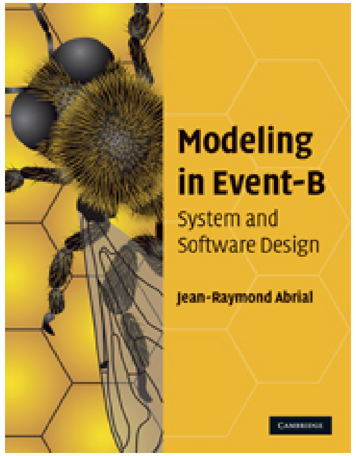
THE EVENT-B METHOD

- The **Event-B method** is an evolution of the classical **B method**.
  - modeling a system by a set of events instead of operations.

- The **Event-B method** is a formal method based on first-order logic and set theory.

- The **Event-B method** is based on :
  - the notions of pre-conditions and post-conditions (**Hoare**),
  - the weakest pre-condition (**Dijkstra**),
  - and the calculus of substitution (**Abrial**).

- The **Event-B method** is adapted to analyse discrete systems.
  - offers the possibility of modelling **discrete behaviors**.



THE EVENT-B METHOD

THE STATE OF A MODEL

- A discrete model is first made of a state
- The state is represented by some constants and variables
- Constants are linked by some properties
- Variables are linked by some invariants
- Properties and invariants are written using set-theoretic expressions

THE EVENT-B METHOD

THE EVENTS OF A MODEL (TRANSITIONS)

- A discrete model is also made of a number of events
- An event is made of a guard and an action
- The guard denotes the enabling condition of the event
- The action denotes the way the state is modified by the event
- Guards and actions are written using set-theoretic expressions

THE EVENT-B METHOD

A MODEL SCHEMATIC VIEW

```
CONTEXT $ctx_1$
EXTENDS $ctx_2$

SETS $s$
CONSTANTS $c$
AXIOMS
    $A(s, c)$
THEOREMS
    $T(s, c)$
END
```

```
MACHINE $mch_1$
REFINES $mch_2$
SEES $ctx_i$

VARIABLES $v$
INVARIANTS
    $I(s, c, v)$
THEOREMS
    $T(s, c, v)$
EVENTS
    $[events\_list]$
END
```

```
$event \ \widehat{=}$
    any $x$
    where
        $G(s, c, v, x)$
    then
        $BA(s, c, v, x, v')$
    end
```

Laboratoire
Méthodes
Formelles

THE EVENT-B METHOD

OPERATIONAL INTERPRETATION

```
Initialize;
while (some events have true guards) {
  Choose one such event;
  Modify the state accordingly
}
```

- An event execution is supposed to take no time
- Thus, no two events can occur simultaneously
- When all events have false guards, the discrete system stops
- When some events have true guards, one of them is chosen non-deterministically and its action modifies the state
- The previous phase is repeated (if possible)

THE EVENT-B METHOD

COMMENTS ON THE OPERATIONAL INTERPRETATION

- Stopping is not necessary: a discrete system may run for ever
- This interpretation is just given here for informal understanding
- The meaning of such a discrete system will be given by the proofs which can be performed on it

BUILDING LARGE COMPUTERIZED SYSTEMS

REFINEMENT

- Refinement allows us to build model gradually
- We shall build an ordered sequence of more precise models
- Each model is a refinement of the one preceding it
- A useful analogy: looking through a microscope
- Spatial as well as temporal extensions
- Data refinement

PURPOSE OF THIS LECTURE

- To present an **example of system development**
- Our approach $\longrightarrow$ a series of **more and more accurate models**
- This approach is called **refinement**
- The models formalize the view of an **external observer**
- With each refinement **observer** *“zooms in”* to see more details

PURPOSE OF THIS LECTURE

- Each model will be analyzed and **proved to be correct**
- The **aim** is to obtain a system that will be **correct by construction**
- The **correctness criteria** are formulated as **proof obligations**
- **Proofs** will be performed by using the **sequent calculus**
- **Inference rules** used in the sequent calculus will be **reviewed**

THE EVENT-B METHOD

MODELS AND PROOF OBLIGATIONS

CONTEXT ctx_1
EXTENDS ctx_2

SETS s
CONSTANTS c
AXIOMS
  $A(s, c)$
THEOREMS
  $T(s, c)$
END

MACHINE mch_1
REFINES mch_2
SEES ctx_i

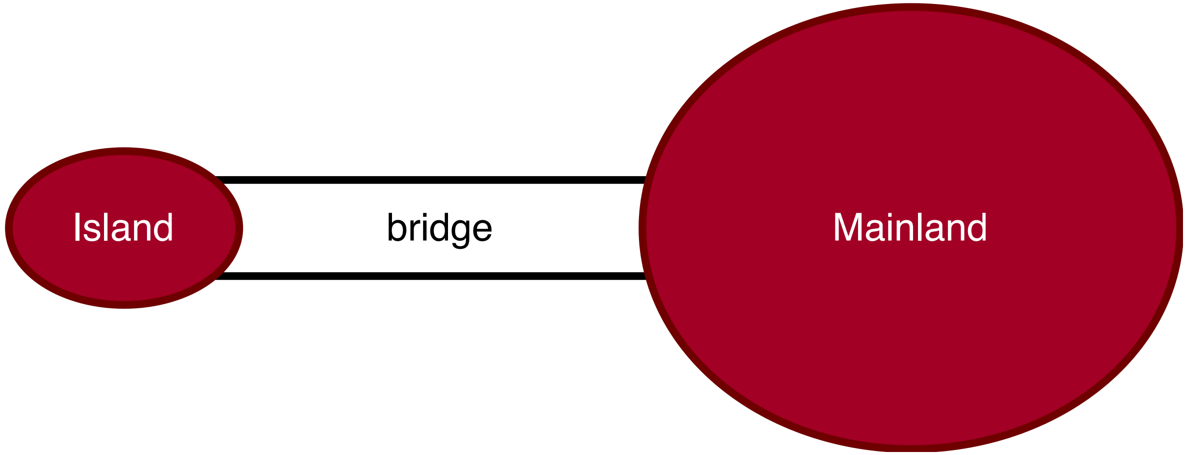
VARIABLES v
INVARIANTS
  $I(s, c, v)$
THEOREMS
  $T(s, c, v)$
EVENTS
  $[events\_list]$
END

$event \mathrel{\widehat{=}}$
  any x
  where
    $G(s, c, v, x)$
  then
    $BA(s, c, v, x, v')$
  end

$$
\begin{array}{lll}
A(s,c) & \vdash & T(s,c) \\
A(s,c) \land I(s,c,v) & \vdash & T(s,c,v) \\
A(s,c) \land I(s,c,v) \land G(s,c,v,x) & \vdash & \exists v'.\, BA(s,c,v,x,v') \\
A(s,c) \land I(s,c,v) \land G(s,c,v,x) \land BA(s,c,v,x,v') & \vdash & I(s,c,v') \\
\ldots & &
\end{array}
$$

A REQUIREMENTS DOCUMENT

- The function of this system is to **control cars** on a **narrow bridge**.
- This bridge is supposed to link the **mainland** to a small **island**.
- **FUN-1** $\longrightarrow$ controlling cars on a bridge between the mainland and an island.
- **FUN-2** $\longrightarrow$ the number of cars on the bridge and the island is limited.
- **FUN-3** $\longrightarrow$ the bridge is one way or the other, not both at the same time.



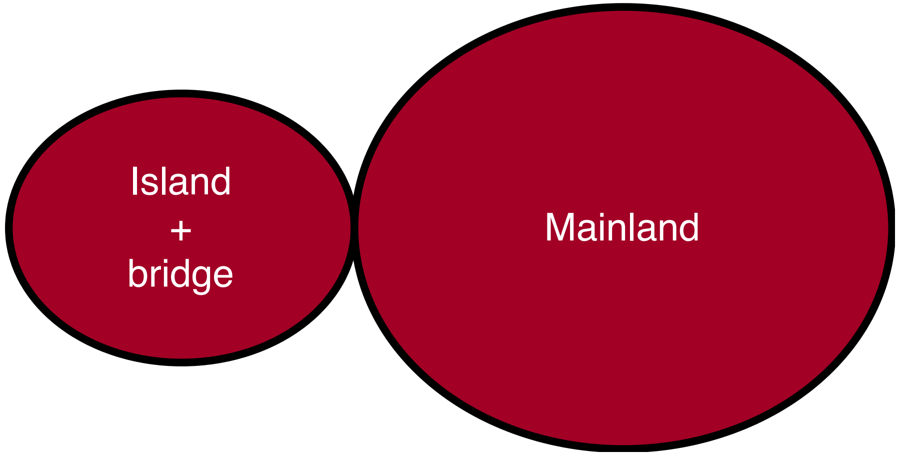
OUR REFINEMENT STRATEGY

- **Initial model** → Limiting the number of cars (**FUN-2**)
- **First refinement** → Introducing the one way bridge (**FUN-1**, **FUN-3**)

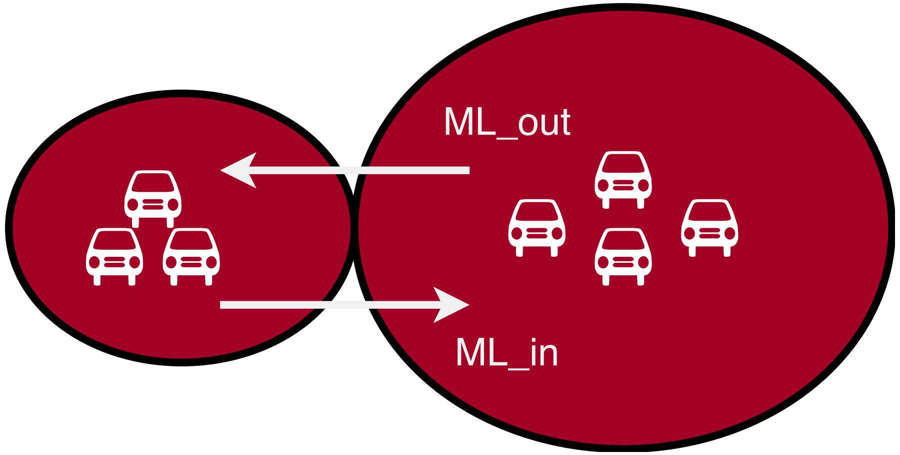
OUR REFINEMENT STRATEGY

- **Initial model** → Limiting the number of cars (**FUN-2**)
  - It is **very simple**
  - We do not even consider the bridge
  - We are just interested in the **pair “island-bridge”**
  - We are focusing **FUN-2** → limited number of cars on island-bridge
- **First refinement** → Introducing the one way bridge (**FUN-1**, **FUN-3**)

A SITUATION AS SEEN FROM THE SKY



TWO EVENTS THAT MAY BE OBSERVED



FORMALIZING THE STATE

- STATIC PART of the state → constant d with axiom axm0_1

```
CONSTANTS
    d
AXIOMS
    axm0_1:  d ∈ ℕ
```

- d is the maximum number of cars allowed on the Island-Bridge
- axm0_1 states that d is a natural number
- Constant d is a member of the set $\mathbb{N} = \{0, 1, 2, \ldots\}$

FORMALIZING THE STATE

- **DYNAMIC PART** of the state $\longrightarrow$ **variable** n with **invariants** `inv0_1` and `inv0_2`

VARIABLES

  n

INVARIANTS

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

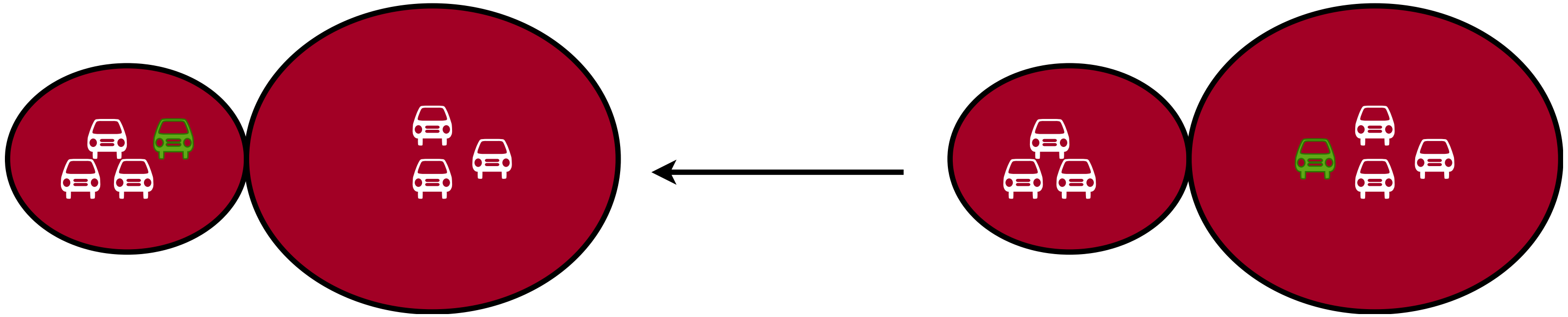
  inv0_2: $n \leq d$

- n is the **effective number of cars** on the Island-Bridge
- n is a natural number (`inv0_1`)
- n is always smaller than or equal to d (`inv0_2`) $\longrightarrow$ this is **FUN 2**

EVENT ML_out

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

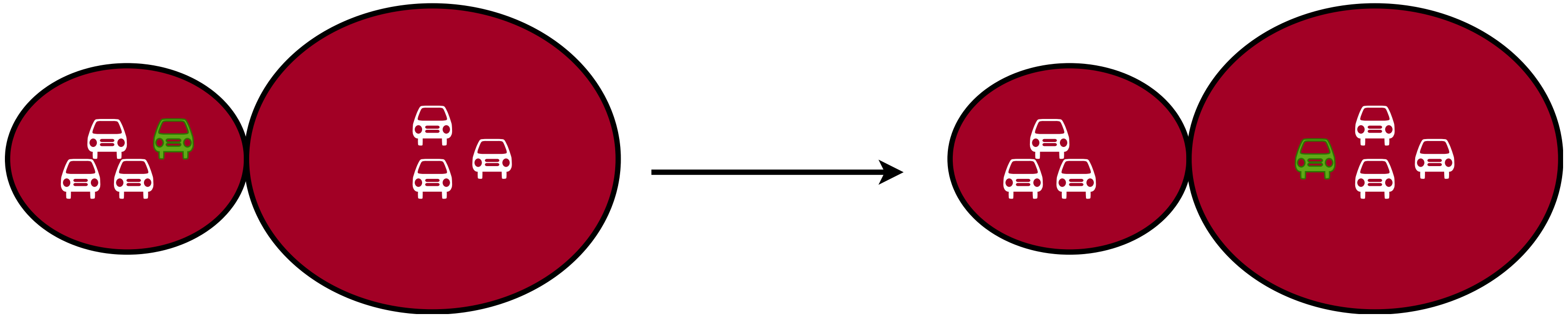
- A car is leaving the mainland and entering the Island-Bridge



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

EVENT `ML_in`

- We can also observe a **second transition** (or event)
- A car leaving the Island-Bridge and re-entering the mainland



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

Laboratoire
Méthodes
Formelles

FORMALIZING THE TWO EVENTS (APPROXIMATION)

- An event is denoted by its **name** and its **action** (an assignment)
- Event ML_out **increments** the number of cars

```
ML_out $\widehat{=}$
  then
    act0_1: $n := n + 1$
  end
```

- Event ML_in **decrements** the number of cars

```
ML_in $\widehat{=}$
  then
    act0_1: $n := n - 1$
  end
```

WHY AN APPROXIMATION?

- These events are approximations for **two reasons**:

  1. They might be **insufficient** at this stage because **not consistent with the invariant**
  2. They might be **refined** (made more precise) later

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

INVARIANTS

- An invariant is a **constraint** on the allowed values of the variables
- An invariant **must hold on all reachable states** of a model
- To verify that this holds we must show that
  1. the invariant holds for **initial states**, and
  2. the invariant is **preserved by all events**
- We will formalize these two statements as **proof obligations (POs)**
- We need a **rigorous proof** showing that these POs indeed hold

BEFORE-AFTER PREDICATES

- To each event can be associated a **before-after predicate**
- It describes the **relation** between the **values** of the variable(s) **just before** and **just after** the event occurrence
- The **before-value** is denoted by the **variable name**, say n
- The **after-value** is denoted by the **primed variable name**, say n'

Laboratoire
Méthodes
Formelles

BEFORE-AFTER PREDICATES

EXAMPLE

- The **events**

```
ML_out ≙
  then
    act0_1: $n := n + 1$
  end
```

```
ML_in ≙
  then
    act0_1: $n := n - 1$
  end
```

- The corresponding **before-after predicates**

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

These representations are equivalent.

Laboratoire
Méthodes
Formelles

ABOUT THE SHAPE
OF THE BEFORE-AFTER PREDICATES

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

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

- The after-value n' is defined as a **function** of the before-value n
- This is because the corresponding events are **deterministic**
- We shall also consider some **non-deterministic** events

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

Laboratoire
Méthodes
Formelles

INTUITION
ABOUT INVARIANT PRESERVATION

- Let us consider invariant `inv0_1`

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

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

$$n' = n + 1$$

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

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

BEING MORE PRECISE

- Under hypothesis $n \in \mathbb{N}$ the conclusion $n + 1 \in \mathbb{N}$ holds
- This can be written as follows

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

- This type of statement is called a **sequent**
- Sequent above → invariant preservation proof obligation for `inv0_1`

PROOF OBLIGATION

INVARIANT PRESERVATION

- We are given an event with before-after predicate $v' = E(c, v)$
- The following sequent expresses preservation of invariant $I_i(c, v)$

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

- It says → $I_i(c, E(c, v))$ provable under hypotheses $A(c)$ and $I(c, v)$
- We have given the name INV to this proof obligation

VERTICAL LAYOUT OF PROOF OBLIGATIONS

- The proof obligation

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

- can be re-written vertically as follows

Axioms	$A(c)$
Invariants	$I(c,v)$
$\vdash$	$\vdash$
Modified Invariant	$I_i(c, E(c,v))$

BACK TO OUR EXAMPLE

- We have two events

```
ML_out $\widehat{=}$
  then
    act0_1: $n := n + 1$
  end
```

```
ML_in $\widehat{=}$
  then
    act0_1: $n := n - 1$
  end
```

- ... and two invariants

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

inv0_2: $n \leq d$

- Thus, we need to prove four proof obligations

PROOF OBLIGATION FOR ML_out AND inv0_1

```
ML_out ≙
  then
    act0_1: $n := n + 1$ // $n' = n + 1$
  end
```

Axioms axm0_1	$d \in \mathbb{N}$
Invariant inv0_1	$n \in \mathbb{N}$
Invariant inv0_2	$n \leq d$
$\vdash$	$\vdash$
Modified Invariant inv0_1	$n + 1 \in \mathbb{N}$

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

Laboratoire
Méthodes
Formelles

PROOF OBLIGATION
FOR ML_out AND inv0_2

```
ML_out $\widehat{=}$
  then
    act0_1: $n := n + 1$ // $n' = n + 1$
  end
```

Axioms axm0_1	$d \in \mathbb{N}$
Invariant inv0_1	$n \in \mathbb{N}$
Invariant inv0_2	$n \leq d$
$\vdash$	$\vdash$
Modified Invariant inv0_2	$n + 1 \leq d$

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

PROOF OBLIGATION FOR ML_in AND inv0_1

```
ML_in $\widehat{=}$
  then
    act0_1: $n := n - 1$ // $n' = n - 1$
  end
```

Axioms axm0_1	$d \in \mathbb{N}$
Invariant inv0_1	$n \in \mathbb{N}$
Invariant inv0_2	$n \leq d$
$\vdash$	$\vdash$
Modified Invariant inv0_1	$n - 1 \in \mathbb{N}$

> This proof obligation is named **ML_in/inv0_1/INV**

PROOF OBLIGATION FOR ML_in AND inv0_2

```
ML_in $\widehat{=}$
  then
    act0_1: $n := n - 1$ // $n' = n - 1$
  end
```

Axioms axm0_1	$d \in \mathbb{N}$
Invariant inv0_1	$n \in \mathbb{N}$
Invariant inv0_2	$n \leq d$
$\vdash$	$\vdash$
Modified Invariant inv0_2	$n - 1 \leq d$

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

SUMMARY OF PROOF OBLIGATIONS

ML_out/inv0_1/INV

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

ML_out/inv0_2/INV

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

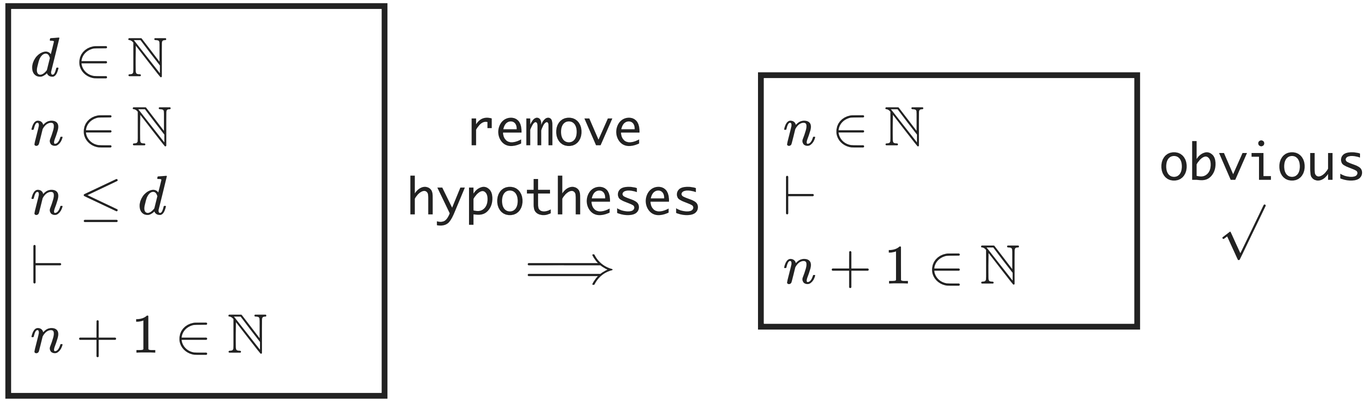
ML_in/inv0_1/INV

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

ML_in/inv0_2/INV

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

INFORMAL PROOF OF ML_out/inv0_1/INV



$$\boxed{\begin{array}{l} d \in \mathbb{N} \\ n \in \mathbb{N} \\ n \leq d \\ \vdash \\ n+1 \in \mathbb{N} \end{array}} \quad \overset{\text{remove hypotheses}}{\Longrightarrow} \quad \boxed{\begin{array}{l} n \in \mathbb{N} \\ \vdash \\ n+1 \in \mathbb{N} \end{array}} \quad \overset{\text{obvious}}{\surd}$$

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

INFERENCE RULES

MONOTONICITY OF HYPOTHESES

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

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

- It expresses the monotonicity of the hypotheses

SOME ARITHMETIC INFERENCE RULES

THE SECOND PEANO AXIOM

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

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

MORE ARITHMETIC INFERENCE RULES

AXIOMS ABOUT ORDERING RELATIONS ON THE INTEGERS

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

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

All inference rules implemented in Rodin are available here

Laboratoire
Méthodes
Formelles

PROOFS

- A **proof** is a tree of sequents with axioms at the leaves.
- The rules applied to the leaves are axioms.
- Each sequent is labeled with (name of) proof rule applied to it.
- The sequent at the root of the tree is called the root sequent.
- The purpose of a proof is to establish the truth of its root sequent.

A FORMAL PROOF OF ML_out/inv0_1/INV

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

Proof requires only application of two rules → MON and P2

A FAILED PROOF ATTEMPT

ML_out/inv0_2/INV

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

- We put a ? to indicate that we have no rule to apply
- **The proof fails** $\rightarrow$ we cannot conclude with rule INC ($n < d$ needed)

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

A FAILED PROOF ATTEMPT

ML_in/inv0_1/INV

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

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

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

A FORMAL PROOF OF `ML_in/inv0_2/INV`

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

$$\dfrac{}{n \leq m \quad \vdash \quad n - 1 \leq m}\ \mathrm{DEC}$$

REASONS FOR PROOF FAILURE

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

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

```
ML_out $\widehat{=}$
  then
    act0_1: $n := n + 1$
  end
```

```
ML_in $\widehat{=}$
  then
    act0_1: $n := n - 1$
  end
```

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

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

IMPROVING THE EVENTS

INTRODUCING GUARDS

```
ML_out $\widehat{=}$
  when
    grd0_1: $n < d$
  then
    act0_1: $n := n + 1$
  end
```

```
ML_in $\widehat{=}$
  when
    grd0_1: $0 < n$
  then
    act0_1: $n := n - 1$
  end
```

- We are adding guards to the events
- The guard is the necessary condition for an event to *occur*

PROOF OBLIGATION

GENERAL INVARIANT PRESERVATION

- Given c with axioms $A(c)$ and v with invariants $I(c,v)$
- Given an event with guard $G(c,v)$ and b-a predicate $v' = E(c,v)$
- We modify the **Invariant Preservation PO** as follows:

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

A FORMAL PROOF OF ML_out/inv0_1/INV

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

Adding new assumptions to a sequent does not affect its provability

A FORMAL PROOF OF `ML_out/inv0_2/INV`

$$\boxed{\begin{array}{l} d \in \mathbb{N} \\ n \in \mathbb{N} \\ n \leq d \\ \color{red}{n < d} \\ \vdash \\ n + 1 \leq d \end{array}} \quad \overset{\texttt{MON}}{\Longrightarrow} \quad \boxed{\begin{array}{l} n < d \\ \vdash \\ n + 1 \leq d \end{array}} \qquad \overset{\texttt{INC}}{\surd}$$

- Now we can conclude the proof using rule INC

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

A FORMAL PROOF OF ML_in/inv0_1/INV

$$\boxed{\begin{array}{l} d \in \mathbb{N} \\ n \in \mathbb{N} \\ n \leq d \\ 0 < n \\ \vdash \\ n - 1 \in \mathbb{N} \end{array}} \qquad \overset{\text{MON}}{\Longrightarrow} \qquad \boxed{\begin{array}{l} 0 < n \\ \vdash \\ n - 1 \in \mathbb{N} \end{array}} \qquad \begin{array}{c} \text{P2'} \\ \surd \end{array}$$

- Now we can conclude the proof using rule $\mathrm{P2'}$

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

A FORMAL PROOF OF ML_in/inv0_2/INV

$$\boxed{\begin{array}{l} d \in \mathbb{N} \\ n \in \mathbb{N} \\ n \leq d \\ 0 < n \\ \vdash \\ n - 1 \leq d \end{array}} \quad \overset{\text{MON}}{\Longrightarrow} \quad \boxed{\begin{array}{l} n \leq d \\ \vdash \\ n - 1 \leq d \end{array}} \quad \overset{\text{DEC}}{\surd}$$

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

RE-PROVING THE EVENTS
NO PROOFS FAIL

ML_out/inv0_1/INV

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

ML_out/inv0_2/INV

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

ML_in/inv0_1/INV

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

ML_in/inv0_2/INV

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

INITIALISATION

- Our system must be initialized (with no car in the island-bridge)
- The initialisation event is never guarded
- It does not mention any variable on the right hand side of $:=$
- Its before-after predicate is just an after predicate

```
init ≙
  begin
    init0_1: $n := 0$
  end
```

After predicate $\Longrightarrow$ $n' = 0$

PROOF OBLIGATION INVARIANT ESTABLISHMENT

- Given c with axioms $A(c)$ and v with invariants $I(c, v)$
- Given an init event with after predicate $v' = K(c)$
- The Invariant Establishment PO is the following:

$$\begin{array}{ll} \text{Axioms} & A(c) \\ \vdash & \vdash \\ \text{Modified Invariant} & I_i(c, K(c)) \end{array}$$

Laboratoire
Méthodes
Formelles

APPLYING THE INVARIANT ESTABLISHMENT PO

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

MORE ARITHMETIC INFERENCE RULES

- First Peano Axiom

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

- Third Peano Axiom (slightly modified)

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

PROOFS OF INVARIANT ESTABLISHMENT

$$\boxed{\begin{array}{l} d \in \mathbb{N} \\ \vdash \\ 0 \in \mathbb{N} \end{array}} \quad \overset{\text{MON}}{\Longrightarrow} \quad \boxed{\begin{array}{l} \vdash \\ 0 \in \mathbb{N} \end{array}} \qquad \begin{array}{c} \text{P1} \\ \surd \end{array}$$

$$\boxed{\begin{array}{l} d \in \mathbb{N} \\ \vdash \\ 0 \leq d \end{array}} \qquad \begin{array}{c} \text{P3} \\ \surd \end{array}$$

A MISSING REQUIREMENT

- It is possible for the system to be blocked if both guards are false
- We do not want this to happen
- We figure out that one important requirement was missing
- **FUN-4** ⟶ Once started, the system should work for ever (Deadlock Freedom)

PROOF OBLIGATION

THE THEOREM PO RULE

- Given c with axioms $A(c)$ and v with invariants $I(c, v)$
- Given the theorem $Th(c, v)$
- Given the guards $G_1(c, v), \ldots, G_m(c, v)$ of the events
- We have to prove the following:

$$\begin{array}{l} A(c) \\ I(c, v) \\ \vdash \\ Th(c, v) \end{array} \qquad\qquad \begin{array}{l} A(c) \\ I(c, v) \\ \vdash \\ G_1(c, v) \;\lor\; \ldots \;\lor\; G_m(c, v) \end{array}$$

Laboratoire
Méthodes
Formelles

APPLYING THE DEADLOCK FREEDOM PO

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

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

INFERENCE RULES FOR DISJUNCTION

- Proof by case analysis

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

- Choice for proving a disjunctive goal

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

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

LMF Laboratoire Méthodes Formelles

PROOF OF DEADLOCK FREEDOM

$$\boxed{\begin{array}{l} d \in \mathbb{N} \\ n \in \mathbb{N} \\ n \leq d \\ \vdash \\ n < d \ \lor \ 0 < n \end{array}} \quad \overset{\texttt{MON}}{\Longrightarrow} \quad \boxed{\begin{array}{l} n \leq d \\ \vdash \\ n < d \ \lor \ 0 < n \end{array}} \quad \overset{\texttt{OR\_L}}{\Longrightarrow} \quad \begin{array}{l} \boxed{\begin{array}{l} n < d \\ \vdash \\ n < d \ \lor \ 0 < n \end{array}} \\[1em] \boxed{\begin{array}{l} n = d \\ \vdash \\ n < d \ \lor \ 0 < n \end{array}} \end{array}$$

$$\boxed{\begin{array}{l} n < d \\ \vdash \\ n < d \ \lor \ 0 < n \end{array}} \quad \overset{\texttt{OR\_R1}}{\Longrightarrow} \quad \boxed{\begin{array}{l} n < d \\ \vdash \\ n < d \end{array}} \quad \overset{?}{\Longrightarrow}$$

seems to be obvious

$$\boxed{\begin{array}{l} n = d \\ \vdash \\ n < d \ \lor \ 0 < n \end{array}} \quad \overset{?}{\Longrightarrow}$$

can be (partially) solved
by applying the equality

MORE INFERENCE RULES

IDENTITY AND EQUALITY

- The identity axiom (conclusion holds by hypothesis)

$$\frac{}{P \ \vdash \ P}\ \text{HYP}$$

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

$$\frac{H(F), E = F \ \vdash \ P(F)}{H(E), E = F \ \vdash \ P(E)}\ \text{EQ\_LR}$$

$$\frac{}{\vdash \ E = E}\ \text{EQL}$$

Laboratoire
Méthodes
Formelles

PROOF OF DEADLOCK FREEDOM

$$\boxed{\begin{array}{l} n < d \\ \vdash \\ n < d \;\lor\; 0 < n \end{array}} \quad \overset{\texttt{OR\_R1}}{\Longrightarrow} \quad \boxed{\begin{array}{l} n < d \\ \vdash \\ n < d \end{array}} \quad \overset{\texttt{HYP}}{\surd}$$

$$\boxed{\begin{array}{l} n = d \\ \vdash \\ n < d \;\lor\; 0 < n \end{array}} \quad \overset{\texttt{EQ\_LR}}{\Longrightarrow} \quad \boxed{\begin{array}{l} \vdash \\ d < d \;\lor\; 0 < d \end{array}} \quad \overset{\texttt{OR\_R2}}{\Longrightarrow} \quad \boxed{\begin{array}{l} \vdash \\ 0 < d \;\; ? \end{array}}$$

- We still have a problem $\longrightarrow$ d must be positive!

ADDING THE FORGOTTEN AXIOM

- If $d = 0$, then no car can ever enter the Island-Bridge

```
CONSTANTS
    $d$
AXIOMS
    axm0_1:  $d \in \mathbb{N}$
    axm0_2:  $0 < d$
```

INITIAL MODEL

CONCLUSION

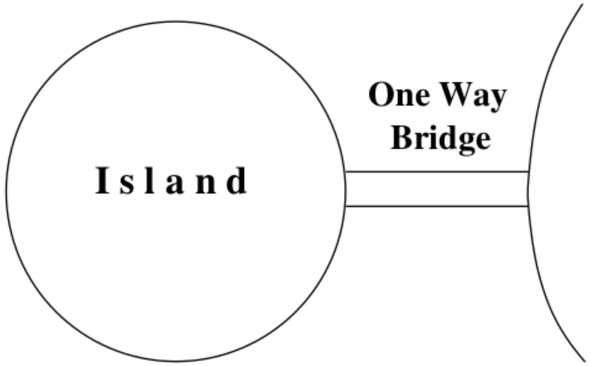
- Thanks to the proofs, we discovered 3 errors
- They were corrected by:
  - adding guards to both events
  - adding an axiom
- The interaction of modeling and proving is an essential element of Formal Methods with Proofs

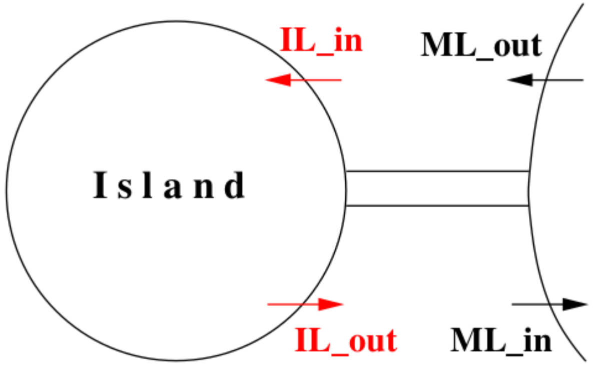
OUR REFINEMENT STRATEGY

- **Initial model** → Limiting the number of cars (**FUN-2**)
- **First refinement** → Introducing the one way bridge (**FUN-1, FUN-3**)
    - Our view of the system gets more accurate
    - We introduce the bridge and separate it from the island (**FUN-1**)
    - We refine the state and the events
    - We also add two new events → `IL_in` and `IL_out`
    - We are focusing on **FUN-3** → one-way bridge

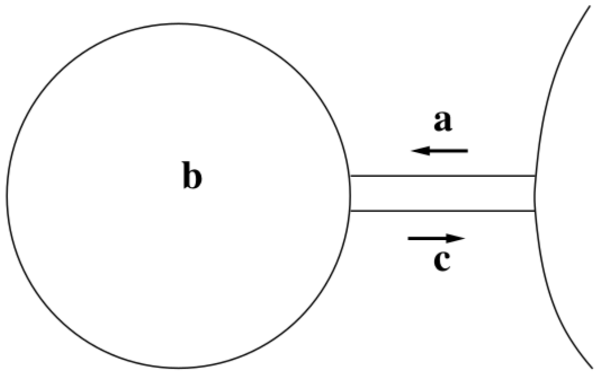
FIRST REFINEMENT

INTRODUCING A ONE-WAY BRIDGE





INTRODUCING THREE NEW VARIABLES



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

REFINING THE STATE

- Variables a, b, and c denote natural numbers

```
VARIABLES
  $a \ b \ c$
INVARIANTS
  inv1_1: $a \in \mathbb{N}$
  inv1_2: $b \in \mathbb{N}$
  inv1_3: $c \in \mathbb{N}$
```

REFINING THE STATE

INTRODUCING NEW INVARIANTS

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

  INVARIANTS

  …
  inv1_4: $a + b + c = n$

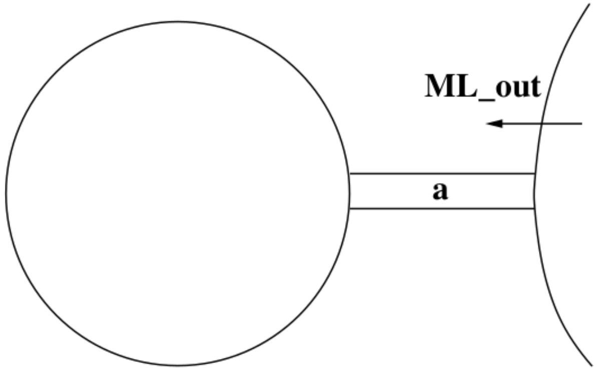
- Formalizing the new invariant → one way bridge (this is **FUN-3**)

  INVARIANTS

  …
  inv1_5: $a = 0 \vee c = 0$

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

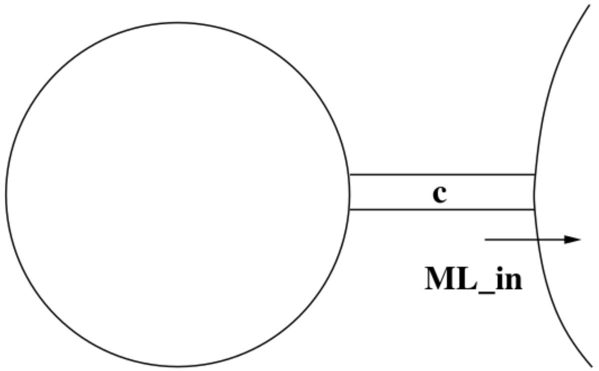
PROPOSAL FOR REFINING EVENT ML_out



```
ML_out  ≙
  when
    grd1_1:  $a + b < d$
    grd1_2:  $c = 0$
  then
    act1_1:  $a := a + 1$
  end
```

Laboratoire
Méthodes
Formelles

PROPOSAL FOR REFINING EVENT ML_in



```
ML_in ≙
  when
    grd1_1: $0 < c$
  then
    act1_1: $c := c - 1$
  end
```

Laboratoire
Méthodes
Formelles

BEFORE-AFTER PREDICATES

PRESERVED VARIABLES

```
ML_out ≙
  when
    grd1_1: $a + b < d$
    grd1_2: $c = 0$
  then
    act1_1: $a := a + 1$
  end
```

```
ML_in ≙
  when
    grd1_1: $0 < c$
  then
    act1_1: $c := c - 1$
  end
```

Before-after predicates showing the unmodified variables

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

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

Laboratoire
Méthodes
Formelles

INTUITION ABOUT REFINEMENT

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

- To show this we have to prove that
  1. every concrete event is simulated by its abstract counterpart
     (event refinement → following slides)
  2. to every concrete initial state corresponds an abstract one
     (initial state refinement → later)

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

INTUITION ABOUT REFINEMENT

```
ML_out $\widehat{=}$ //abstract
  when
    grd0_1: $n < d$
  then
    act0_1: $n := n + 1$
  end
```

```
ML_out $\widehat{=}$ //concrete
  when
    grd1_1: $a + b < d$
    grd1_2: $c = 0$
  then
    act1_1: $a := a + 1$
  end
```

- The concrete version is not contradictory with the abstract one
- When the concrete version is enabled then so is the abstract one
- Executions seem to be compatible

INTUITION ABOUT REFINEMENT

```
ML_in $\widehat{=}$ //abstract
  when
    grd0_1: $0 < n$
  then
    act0_1: $n := n - 1$
  end
```

```
ML_in $\widehat{=}$ //concrete
  when
    grd1_1: $0 < c$
  then
    act1_1: $c := c - 1$
  end
```

- Same remarks as in the previous slide
- But this has to be confirmed by well-defined proof obligations

PROOF OBLIGATIONS FOR REFINEMENT

- The concrete guard is stronger than the abstract one
- Each concrete action is compatible with its abstract counterpart

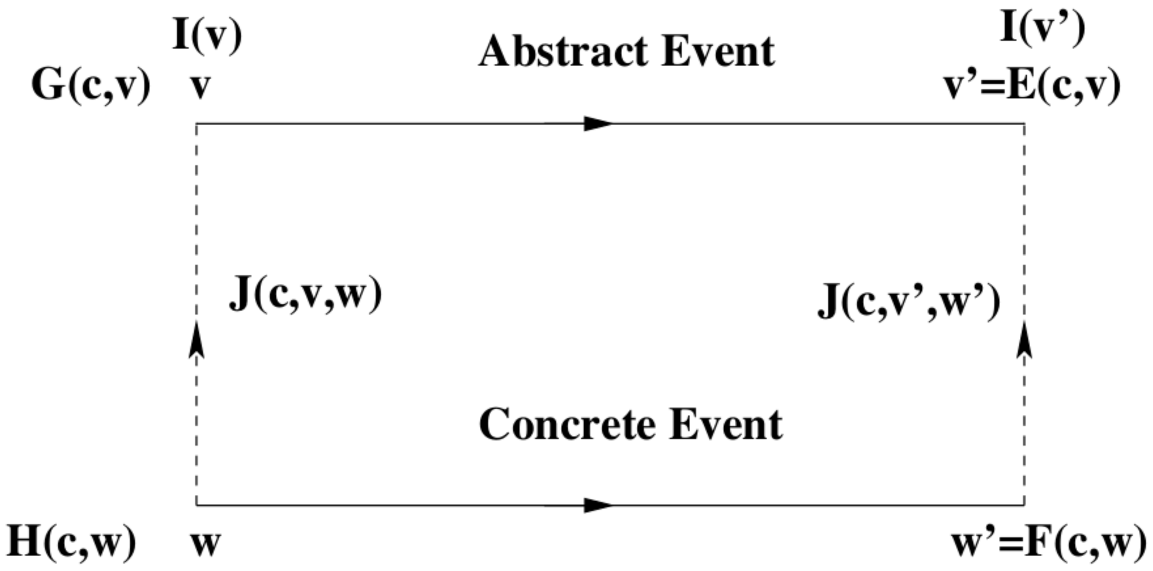
Laboratoire
Méthodes
Formelles

PROVING CORRECT REFINEMENT

THE SITUATION

- Constants c with axioms $A(c)$
- Abstract variables v with abstract invariant $I(c, v)$
- Concrete variables w with concrete invariant $J(c, v, w)$
- Abstract event with guards $G(c, v) \rightarrow G_1(c, v), G_2(c, v), \ldots$
- Abstract event with before-after predicate $v' = E(c, v)$
- Concrete event with guards $H(c, w)$ and b-a predicate $w' = F(c, w)$

PCORRECTNESS OF EVENT REFINEMENT



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

PROOF OBLIGATION

GUARD STRENGTHENING

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

GRD

APPLYING GUARD STRENGTHENING TO EVENT ML_out

PROOF OF ML_out/GRD

```
ML_out $\widehat{=}$ //abstract
  when
    grd0_1: $n < d$
  then
    act0_1: $n := n + 1$
  end
```

```
ML_out $\widehat{=}$ //concrete
  when
    grd1_1: $a + b < d$
    grd1_2: $c = 0$
  then
    act1_1: $a := a + 1$
  end
```

APPLYING GUARD STRENGTHENING TO EVENT ML_out

PROOF OF ML_out/GRD

$$\boxed{\begin{array}{l} d \in \mathbb{N} \\ 0 < d \\ n \in \mathbb{N} \\ n \leq d \\ a \in \mathbb{N} \\ b \in \mathbb{N} \\ c \in \mathbb{N} \\ a + b + c = n \\ a = 0 \lor c = 0 \\ a + b < d \\ c = 0 \\ \vdash \\ n < d \end{array}} \quad \xRightarrow{\texttt{MON}} \quad \boxed{\begin{array}{l} a + b + c = n \\ a + b < d \\ c = 0 \\ \vdash \\ n < d \end{array}} \quad \xRightarrow{\texttt{EQ\_LR}} \quad \boxed{\begin{array}{l} a + b + 0 = n \\ a + b < d \\ \vdash \\ n < d \end{array}} \quad \xRightarrow{\texttt{ARITH}} \quad \ldots$$

APPLYING GUARD STRENGTHENING TO EVENT ML_out

PROOF OF ML_out/GRD

$$\overset{\text{ARITH}}{\Longrightarrow} \ \ldots \ \boxed{\begin{array}{l} a+b=n \\ a+b<d \\ \vdash \\ n<d \end{array}} \ \overset{\text{EQ\_LR}}{\Longrightarrow} \ \boxed{\begin{array}{l} n<d \\ \vdash \\ n<d \end{array}} \ \overset{\text{HYP}}{\surd}$$

APPLYING GUARD STRENGTHENING TO EVENT ML_in

PROOF OF ML_in/GRD

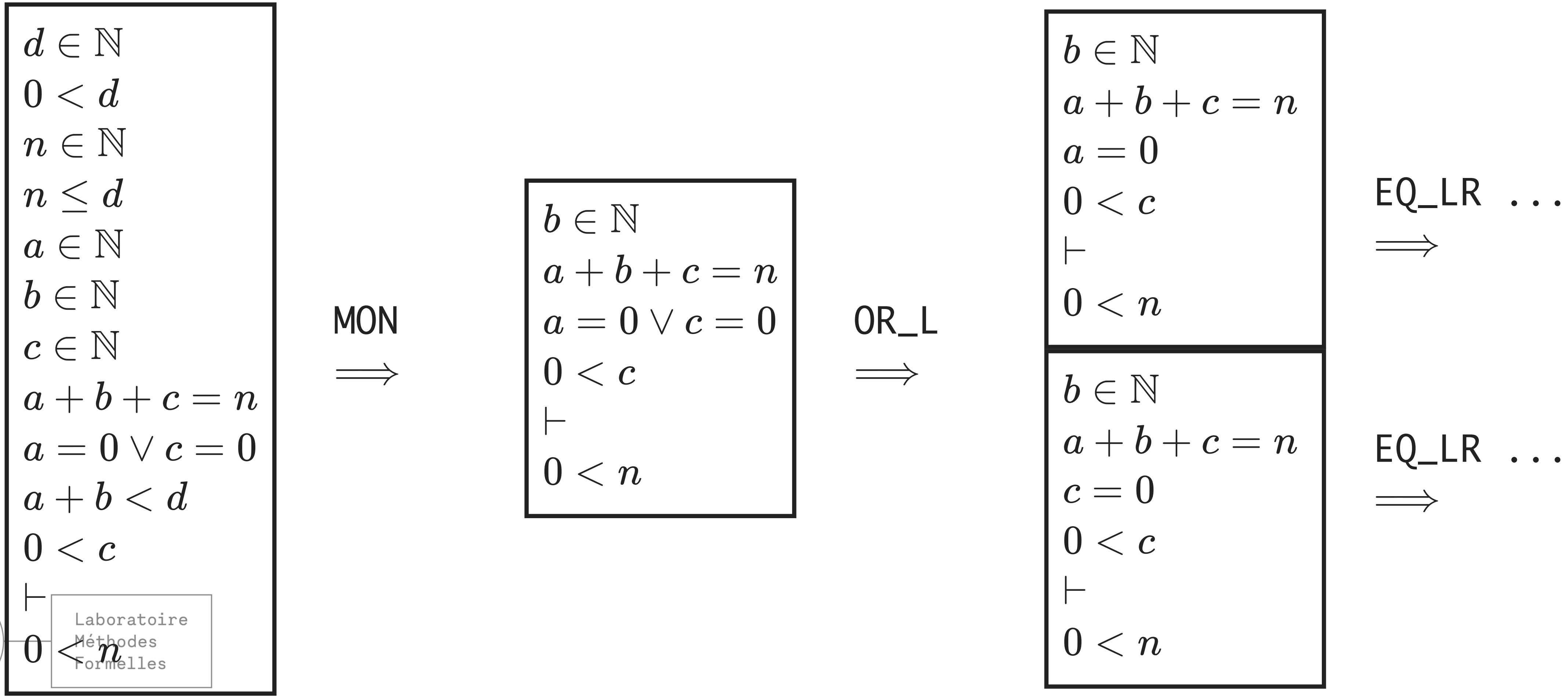
```
ML_in $\widehat{=}$ //abstract
  when
    grd0_1: $0 < n$
  then
    act0_1: $n := n - 1$
  end
```

```
ML_in $\widehat{=}$ //concrete
  when
    grd1_1: $0 < c$
  then
    act1_1: $c := c - 1$
  end
```

Laboratoire
Méthodes
Formelles

APPLYING GUARD STRENGTHENING TO EVENT ML_in

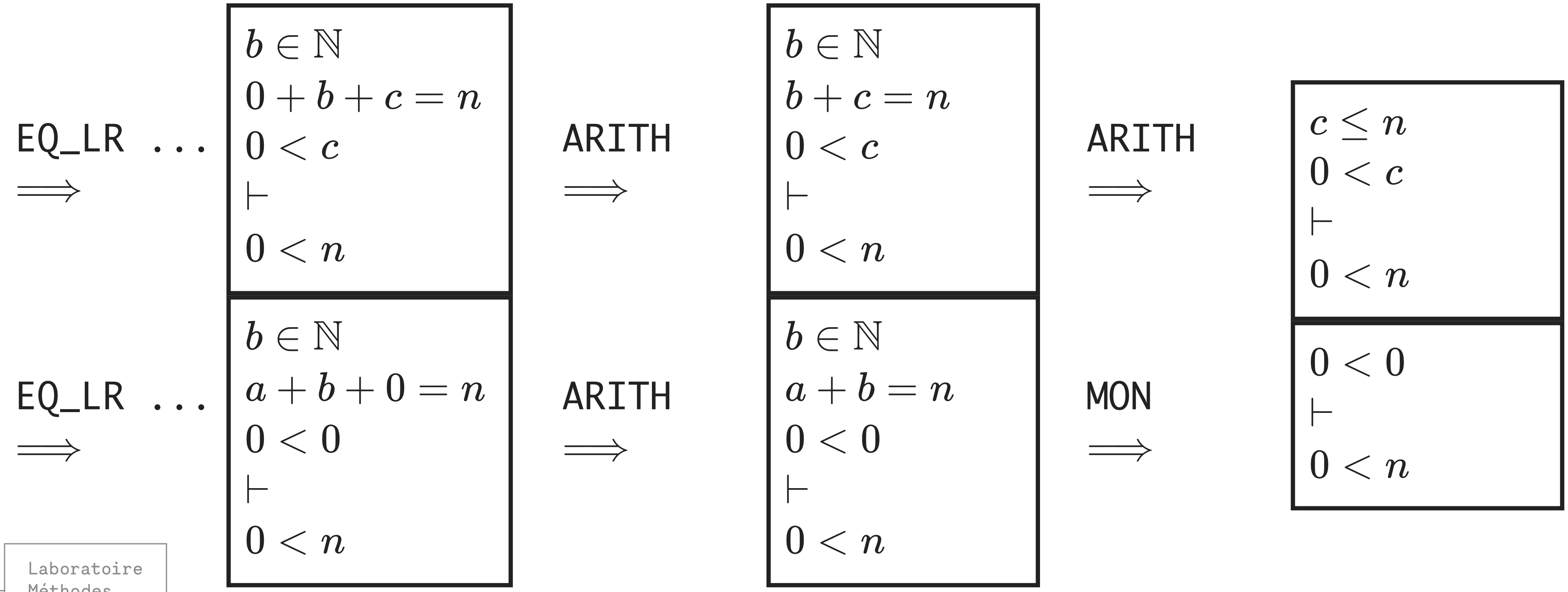
PROOF OF ML_in/GRD



$$\boxed{\begin{array}{l} d \in \mathbb{N} \\ 0 < d \\ n \in \mathbb{N} \\ n \leq d \\ a \in \mathbb{N} \\ b \in \mathbb{N} \\ c \in \mathbb{N} \\ a + b + c = n \\ a = 0 \lor c = 0 \\ a + b < d \\ 0 < c \\ \vdash \\ 0 < n \end{array}} \overset{\text{MON}}{\Longrightarrow} \boxed{\begin{array}{l} b \in \mathbb{N} \\ a + b + c = n \\ a = 0 \lor c = 0 \\ 0 < c \\ \vdash \\ 0 < n \end{array}} \overset{\text{OR\_L}}{\Longrightarrow} \begin{array}{l} \boxed{\begin{array}{l} b \in \mathbb{N} \\ a + b + c = n \\ a = 0 \\ 0 < c \\ \vdash \\ 0 < n \end{array}} \overset{\text{EQ\_LR} \ \ldots}{\Longrightarrow} \\ \boxed{\begin{array}{l} b \in \mathbb{N} \\ a + b + c = n \\ c = 0 \\ 0 < c \\ \vdash \\ 0 < n \end{array}} \overset{\text{EQ\_LR} \ \ldots}{\Longrightarrow} \end{array}$$

APPLYING GUARD STRENGTHENING TO EVENT ML_in

PROOF OF ML_in/GRD



EQ_LR ... $\Longrightarrow$

$b \in \mathbb{N}$

$0 + b + c = n$

$0 < c$

$\vdash$

$0 < n$

ARITH $\Longrightarrow$

$b \in \mathbb{N}$

$b + c = n$

$0 < c$

$\vdash$

$0 < n$

ARITH $\Longrightarrow$

$c \leq n$

$0 < c$

$\vdash$

$0 < n$

EQ_LR ... $\Longrightarrow$

$b \in \mathbb{N}$

$a + b + 0 = n$

$0 < 0$

$\vdash$

$0 < n$

ARITH $\Longrightarrow$

$b \in \mathbb{N}$

$a + b = n$

$0 < 0$

$\vdash$

$0 < n$

MON $\Longrightarrow$

$0 < 0$

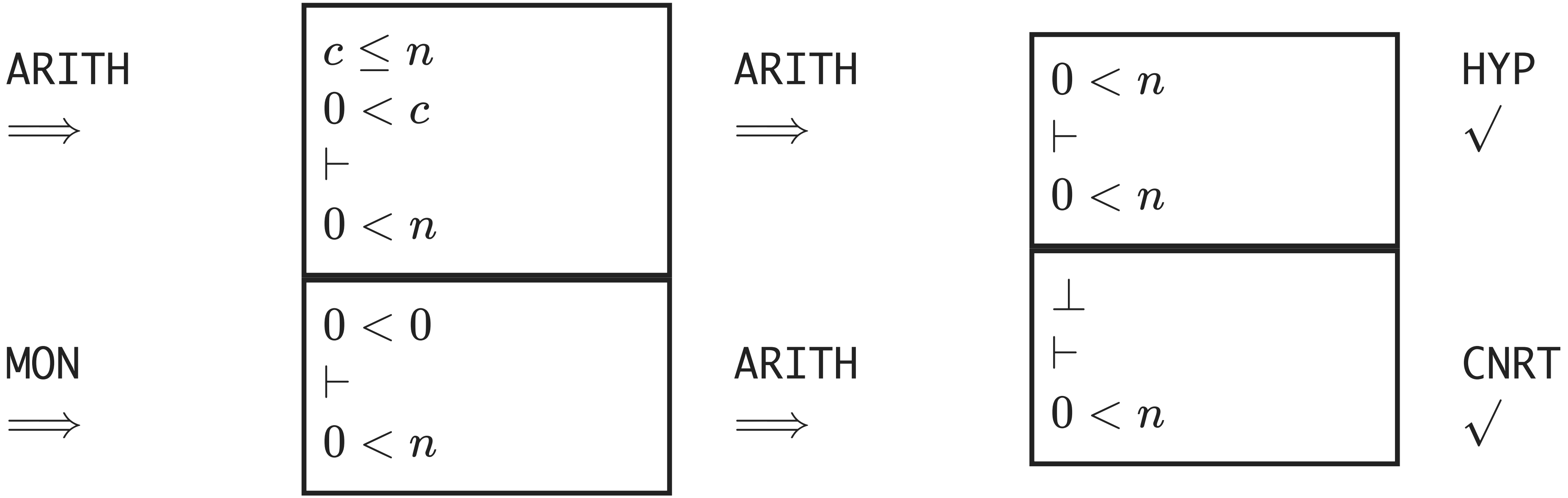
$\vdash$

$0 < n$

Laboratoire
Méthodes
Formelles

APPLYING GUARD STRENGTHENING TO EVENT ML_in

PROOF OF ML_in/GRD



ARITH $\Longrightarrow$

$c \leq n$
$0 < c$
$\vdash$
$0 < n$

ARITH $\Longrightarrow$

$0 < n$
$\vdash$
$0 < n$

HYP $\surd$

MON $\Longrightarrow$

$0 < 0$
$\vdash$
$0 < n$

ARITH $\Longrightarrow$

$\perp$
$\vdash$
$0 < n$

CNRT $\surd$

- In the previous proof, we have used and additional inference rule
- It says that a false hypothesis entails any goal $\quad \perp \vdash P \quad$ CNTR

OUTLINE

- The Event-B method
- **The Pro-B animator/model-checker**
- The Theory plugin

Back to the outline - Back to the begin

OUTLINE

- The Event-B method
- The Pro-B animator/model-checker
- **The Theory plugin**

Back to the outline - Back to the begin

THANK YOU

PDF version of the slides

Back to the begin - Back to the outline

LMF

Laboratoire
Méthodes
Formelles