

# Modelling Software-based Systems

## Lecture 4

### System Engineering using Refinement-based Methodology

#### Master Informatique

Dominique Méry  
Telecom Nancy, Université de Lorraine

12 février 2026  
[dominique.mery@loria.fr](mailto:dominique.mery@loria.fr)

# General Summary

① Refinement of models

② Summary on Event-B

③ The Access Control

④ Conclusion

# Current Summary

① Refinement of models

② Summary on Event-B

③ The Access Control

④ Conclusion

# Summing up...

- Refinement relates Event-B models
- Problem for starting a refinement-based development
- Problem for finding the best abstract model
- Problem for discharging unproved proof obligations generated for each refinement step
- The Access Control Problem

# Current Summary

① Refinement of models

② Summary on Event-B

③ The Access Control

④ Conclusion

# Simple Form of an Event

- An event of the **simple form** is denoted by :

```
< event_name > ≡  
WHEN  
    < condition >  
THEN  
    < action >  
END
```

where

- $\langle \text{event\_name} \rangle$  is an identifier
- $\langle \text{condition} \rangle$  is the firing condition of the event
- $\langle \text{action} \rangle$  is a generalized substitution (**parallel assignment**)

# Non-deterministic Form of an Event

- An event of the **non-deterministic** form is denoted by :

```
< event_name > ≡  
ANY < variable > WHERE  
    < condition >  
THEN  
    < action >  
END
```

where

- *< event\_name >* is an identifier
- *< variable >* is a (list of) variable(s)
- *< condition >* is the firing condition of the event
- *< action >* is a generalized substitution (**parallel assignment**)

# Shape of a Generalized Substitution

A generalized substitution can be

- Simple assignment :  $x := E$
- Generalized assignment :  $x : P(x, x')$
- Set assignment :  $x : \in S$
- Parallel composition :
  - $T$
  - $\dots$
  - $U$

,

# Invariant Preservation Verification (0)

INVARIANT  $\wedge$  GUARD  
 $\implies$   
ACTION **establishes** INVARIANT

# Invariant Preservation Verification (1)

- Given an event of the simple form :

```
EVENT e  ≡  
  WHEN  
    G(x)  
  THEN  
    x := E(x)  
  END
```

and invariant  $I(x)$  to be preserved, the statement to prove is :

$$I(x) \wedge G(x) \implies I(E(x))$$

## Invariant Preservation Verification (2)

- Given an event of the simple form :

```
EVENT e  ≡  
WHEN  
    G(x)  
THEN  
    x : |P(x, x')  
END
```

and invariant  $I(x)$  to be preserved, the statement to prove is :

$$I(x) \wedge G(x) \wedge P(x, x') \implies I(x')$$

# Invariant Preservation Verification (3)

- Given an event of the simple form :

```
EVENT e  ≡  
  WHEN  
    G(x)  
  THEN  
    x :∈ S(x)  
END
```

and invariant  $I(x)$  to be preserved, the statement to prove is :

$$I(x) \wedge G(x) \wedge x' \in S(x) \implies I(x')$$

# Invariant Preservation Verification (4)

- Given an event of the non-deterministic form :

```
EVENT e ≡  
ANY v WHERE  
    G(x, v)  
THEN  
    x := E(x, v)  
END
```

and invariant  $I(x)$  to be preserved, the statement to prove is :

$$I(x) \wedge G(x, v) \implies I(E(x, v))$$

# Refinement Technique (1)

- Abstract models works with variables  $x$ , and concrete one with  $y$
- A **gluing invariant**  $J(x, y)$  links both sets of vrbls
- Each **abstract event** is refined by **concrete one** (see below)

## Refinement Technique (2)

- Some new events may appear : they refine “skip”
- Concrete events must not block more often than the abstract ones
- The set of new event alone must always block eventually

# Correct Refinement Verification (1)

- Given an **abstract** and a corresponding **concrete** event

```
EVENT ae  ≡  
WHEN  
  G(x)  
THEN  
  x := E(x)  
END
```

```
EVENT ce  ≡  
WHEN  
  H(y)  
THEN  
  y := F(y)  
END
```

and invariants  $I(x)$  and  $J(x, y)$ , the statement to prove is :

$$I(x) \wedge J(x, y) \wedge H(y) \implies G(x) \wedge J(E(x), F(y))$$

# Correct Refinement Verification (2)

- Given an **abstract** and a corresponding **concrete** event

```
EVENT ae  $\hat{=}$ 
ANY v WHERE
  G(x, v)
THEN
  x := E(x, v)
END
```

```
EVENT ce  $\hat{=}$ 
ANY w WHERE
  H(y, w)
THEN
  y := F(y, w)
END
```

$$\begin{aligned} I(x) &\wedge J(x, y) \wedge H(y, w) \\ \implies \exists v \cdot (G(x, v) \wedge J(E(x, v), F(y, w))) \end{aligned}$$

# Correct Refinement Verification (3)

- Given a NEW event

```
EVENT ce  ≡  
  WHEN  
    H(y)  
  THEN  
    y := F(y)  
  END
```

and invariants  $I(x)$  and  $J(x, y)$ , the statement to prove is :

$$I(x) \wedge J(x, y) \wedge H(y) \implies J(x, F(y))$$

# Current Summary

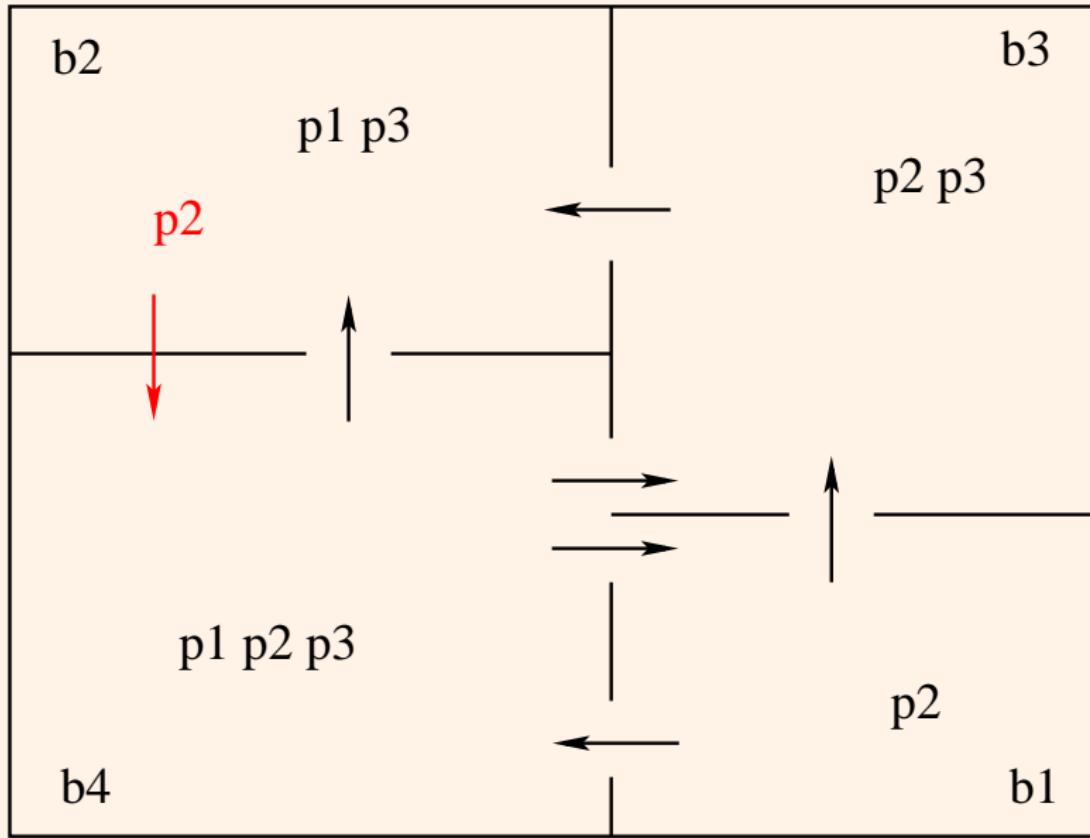
① Refinement of models

② Summary on Event-B

③ The Access Control

④ Conclusion

- To control **accesses** into locations.
- People are assigned certain **authorizations**
- Each person is given a **magnetic card**
- Doors are “one way” **turnstyles**
- Each turnstyle is equipped with :
  - a **card reader**
  - two **lights** (one **green**, the other **red**)



# Access Protocol (after introducing card in reader)

- If access **permitted** {
  - green light **turned on**
  - turnstyle **unblocked** for 30 sec
- Passing, or 30 sec elapsed {
  - green light **turned off**
  - turnstyle **blocked** again
- If access **refused** {
  - red light **turned on** for 2 sec
  - turnstyle **stays blocked**

# Goal of System Study

- Sharing between **Control and Equipment**
- For this : constructing a **closed model**
- Defining the **physical environment**
- Possible **generalization** of problem
- Studying **safety** questions
- Studying **synchronisation** questions
- Studying **marginal** behaviour

# Basic System Properties

- P1 : The model concerns **people** and **locations**
- P2 : A person is authorized to be in **some locations**
- P3 : A person can only be in **one location at a time**
- D1 : **Outside** is a location where everybody can be
- P4 : A person is **always in some location**
- P5 : **A person is always authorized to be in his location**

# Example

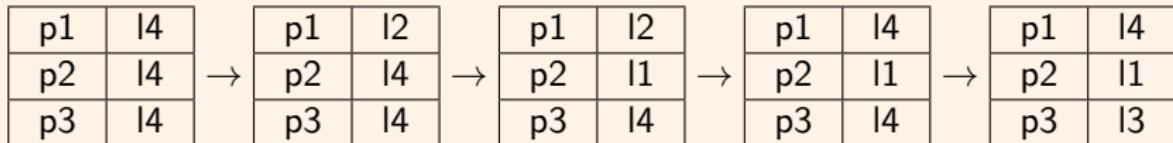
## Sets

$$\begin{array}{lcl} \text{persons} & = & \{ p1, p2, p3 \} \\ \text{locations} & = & \{ l1, l2, l3, l4 \} \end{array}$$

## Authorizations

p1	l2, l4
p2	l1, l3, l4
p3	l2, l3, l4

## Correct scenario



# Model (1)

**Basic sets** : persons P and locations B (prop. P1)

**Constant** : authorizations A (prop. P2)

A is a **binary relation** between P and B

$$A \in P \leftrightarrow B$$

## Model (2)

Constant : *outside* is a location where everybody is authorized to be (decision D1)

$$\text{outside} \in B$$

$$P \times \{\text{outside}\} \subseteq A$$

## Model (3)

**Variable** : situations C (prop. P3 and P4)

C is a **total function** between P and B

A total function is a **special case** of a binary relation

$$c \in P \rightarrow B$$

**Invariant** : situations **compatible** with auth. (prop. P5)

The function C is **included** in the relation A

$$C \subseteq A$$

# A magic event which can be observed

- GUARD :  $\left\{ \begin{array}{l} \text{- Given some person } p \text{ and location } l \\ \text{- } p \text{ is authorized to be in } l : p, l \in A \\ \text{- } p \text{ is not currently in } l : c(p) \neq l \end{array} \right.$
- ACTION : -  $p$  jumps into  $l$

```
EVENT observation1  ≡  
ANY p, l WHERE  
  p ∈ P  ∧  
  l ∈ B  ∧  
  p ↦ l ∈ A  ∧  
  c(p) ≠ l  
THEN  
  c(p) := l  
END
```

# Relation overriding

Given two relations  $a$  and  $b$

Overriding  $a$  by  $b$  yields a new relation  $a \triangleleft b$

$$a \triangleleft b \quad \hat{=} \quad (\text{dom}(b) \triangleleft a) \cup b$$

Abbreviation

$$f(x) := y \quad \hat{=} \quad f := f \triangleleft \{x \mapsto y\}$$

# Invariant Preservation Proof

INVARIANT  $\wedge$  GUARD  
 $\implies$   
ACTION establishes INVARIANT

$$\begin{aligned} C \subseteq A &\quad \wedge \\ p \in P &\quad \wedge \\ l \in B &\quad \wedge \\ p \mapsto l \in A & \\ \implies & \\ (\{p\} \triangleleft C) \cup \{p \mapsto l\} &\subseteq A \end{aligned}$$

# First Refinement : Introducing Geometry

- P6 : The geometry define how locations communicate
- P7 : A location does not communicate with itself
- P8 : Persons move between communicating locations

**Constant** : communication STRUCTURE (prop. P6 and P7)

STRUCTURE is a binary relation between B

The intersection of STRUCTURE with the **identity relation** on B is empty

$$\text{STRUCTURE} \in B \leftrightarrow B$$

$$\text{STRUCTURE} \cap \text{id}(B) = \emptyset$$

# Correct Refinement Verification (reminder)

Concrete events **do not block more often than abstract ones**

$$\begin{aligned} I(x) \wedge J(x, y) \wedge \\ \text{disjunction of abstract guards} \\ \implies \\ \text{disjunction of concrete guards} \end{aligned}$$

New events block eventually (decreasing the same quantity  $V(y)$ )

$$I(x) \wedge J(x, y) \wedge H(y) \wedge V(y) = n \implies V(F(y)) < n$$

# Refined Event

Event (prop. P8)

The guard is **strengthened**

The current location of  $p$  and the new location  $l$  **must communicate**

```
EVENT observation1  ≡  
ANY p, l WHERE  
  p ∈ P  ∧  
  l ∈ B  ∧  
  p ↪ l ∈ A  ∧  
  c(p) ≠ l  
THEN  
  c(p) := l  
END
```

```
EVENT observation2  ≡  
REFINES observation1  
ANY p, l WHERE  
  p ∈ P  ∧  
  l ∈ B  ∧  
  p ↪ l ∈ A  ∧  
  c(p) ↪ l ∈ STRUCTURE  
THEN  
  c(p) := l  
END
```

Invariant preservation : Success

Guard strengthening : Success

$$\begin{aligned} & \exists (p, l) \cdot (p \mapsto l \in A \wedge c(p) \mapsto l \in \text{STRUCTURE}) \\ \Rightarrow & \exists (p, l) \cdot (p \mapsto l \in A \wedge c(p) \neq l) \end{aligned}$$

Deadlockfreeness : Failure

$$\begin{aligned} & \exists (p, l) \cdot (p \mapsto l \in A \wedge c(p) \neq l) \\ \Rightarrow & \exists (p, l) \cdot (p \mapsto l \in A \wedge c(p) \mapsto l \in \text{STRUCTURE}) \end{aligned}$$

# Safety Problem

P9 : No person must remain blocked in a location.

## Solution

P10 : Any person authorized to be in a location must also be authorized to go in another location which communicates with the first one.

$$A \subseteq A ; \text{STRUCTURE}^{-1}$$

$$p \mapsto l \in A \implies \exists m \cdot ( p \mapsto m \in A \wedge l \mapsto m \in \text{STRUCTURE} )$$

# Example

p1	I2	p2	I4
p1	I4	p3	I2
p2	I1	p3	I3
p2	I3	p3	I4

A

I1	I3
I1	I4
I3	I2
I4	I1
I4	I2
I4	I3

STRUCTURE

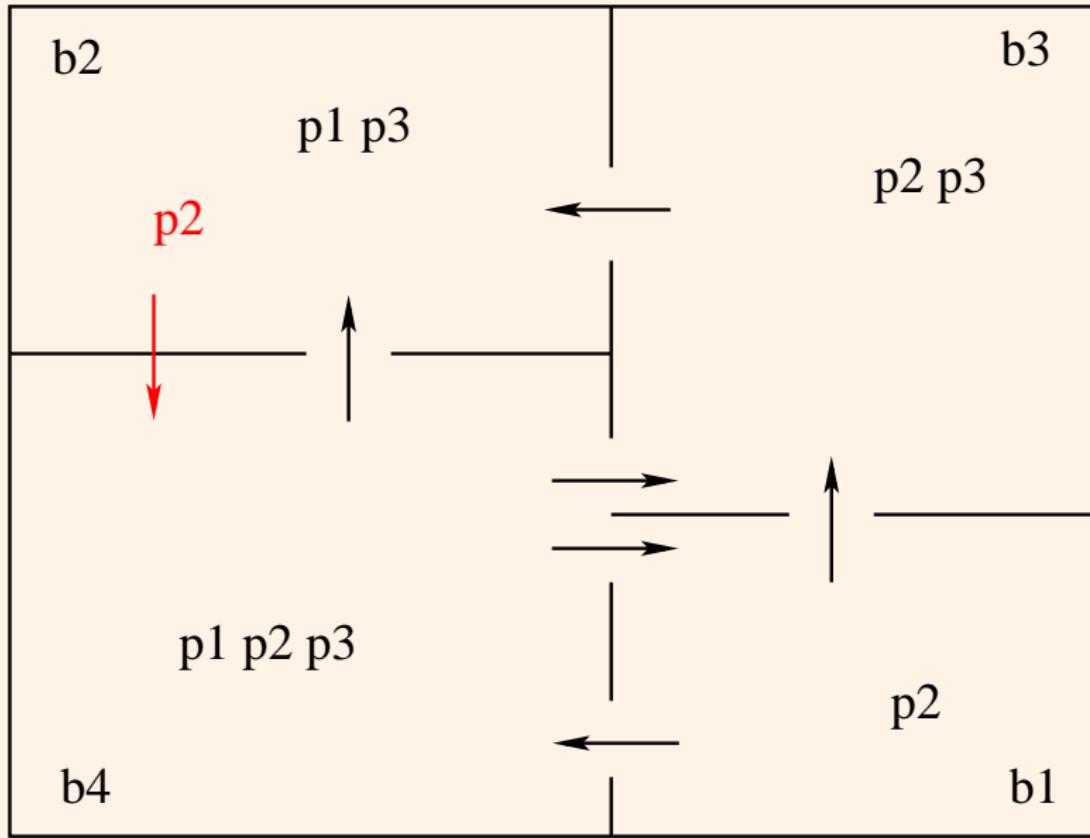
I1	I4
I2	I3
I2	I4
I3	I1
I3	I4
I4	I1

STRUCTURE<sup>-1</sup>

p1	I1
p1	I3
p1	I4
p2	I1

A; STRU

- Opening a door between I2 and I4
- Authorizing p2 to go to I2



# Solution

p1	I2	p2	I4
p1	I4	p3	I2
p2	I1	p3	I3
p2	I2	p3	I4
p2	I3		

A

I1	I3
I1	I4
I2	I4
I3	I2
I4	I1
I4	I2
I4	I3

STRUCTURE

I1	I4
I2	I3
I2	I4
I3	I1
I3	I4
I4	I1
I4	I2

STRUCTURE<sup>-1</sup>

p1	I1		
p1	I2		
p1	I3		
p1	I4		
p2	I1		
p2	I2		
p2	I3		
p2	I4		

A; STRUCTU

## Decision

D2 : The system that we are going to construct does not guarantee that people can move “outside”.

# A better solution (1)

Constante :  $exit$  is a function, included in  $com$ , with no cycle

$$exit \in B - \{outside\} \rightarrow B$$

$$exit \subseteq com$$

$$\forall s \cdot (s \subseteq B \implies (s \subseteq exit^{-1}[s] \implies s = \emptyset))$$

$$\begin{aligned} & \forall x \cdot (x \in s \implies \exists y \cdot (y \in s \wedge (x, y) \in exit)) \\ \implies & \\ & s = \emptyset \end{aligned}$$

$exit$  is a tree **spanning** the graph represented by  $com$

## A better solution (2)

P10' : Every person authorized to be in a location (which is not “outside”) must also be authorized to be in another location communicating with the former and **leading towards the exit**.

$$A \triangleright \{outside\} \subseteq A ; exit^{-1}$$

$$\begin{aligned} p \mapsto l &\in A \wedge \\ l &\neq outside \\ \implies p \mapsto &exit(l) \in A \end{aligned}$$

## For the experts

Show that no cycle implies the possibility to prove property by induction and vice-versa

$$\forall s \cdot (s \subseteq B \wedge s \subseteq \text{exit}^{-1}[s] \implies s = \emptyset)$$

$\Leftrightarrow$

$$\forall t \cdot (t \subseteq B \wedge \text{outside} \in t \wedge \text{exit}^{-1}[t] \subseteq t \implies t = B)$$

$$t \subseteq B$$

$$\text{outside} \in t$$

$$\forall (x, y) \cdot ((x \mapsto y) \in \text{exit} \wedge y \in t \implies x \in t)$$

$\implies$

$$t = B$$

## Second Refinement : Introducing Doors

P11 : Locations communicate via one-way doors.

P12 : A person get through a door only if accepted.

P13 : A door is acceptable by at most one person at a time.

P14 : A person is accepted for at most one door only.

P15 : A person is accepted if at the origin of the door.

P16 : A person is accepted if authorized at destination.

# Extending the Model (1)

**Set** : the set DOORS of doors

**Constants** : The origin ORG and destination DST of a door  
(prop. P11)

$$\begin{aligned} \text{ORG} &\in \text{DOORS} \rightarrow \mathbb{B} \\ \text{DST} &\in \text{DOORS} \rightarrow \mathbb{B} \\ \text{STRUCTURE} &= (\text{ORG}^{-1}; \text{DST}) \end{aligned}$$

## Extending the Model (2)

**Variable** : the rel. DAP between persons and doors (prop. P12 to P16)

$$\begin{aligned} \text{DAP} &\in P \leftrightarrow \text{DOORS} \\ (\text{DAP} ; \text{ORG}) &\subseteq C \\ (\text{DAP} ; \text{DST}) &\subseteq A \end{aligned}$$

## Second Refinement : More Properties

- P17 : Green light of a door is lit **when access is accepted.**
- P18 : When a person has got through, the **door blocks**.
- P19 : After 30 seconds, the **door blocks automatically**.
- P20 : Red light is lit for 2 sec.**when access is refused.**
- P21 : Red and green lights are **not lit simultaneously**.

## Extending the Model (3)

**Definition** : GREEN is exactly the range of DAP (prop. P17 to P19)

$$\text{GREEN} \quad \widehat{=} \quad \text{ran}(\text{DAP})$$

## Extending the Model (4)

**Variable** : The set *red* of red doors (prop. P20)

$$red \subseteq \text{DOORS}$$

**Invariant** : GREEN and *red* are incompatible (prop. P21)

$$\text{GREEN} \cap red = \emptyset$$

# Condition for Admission

P22 : Person  $p$  is accepted through door  $d$  if

- $p$  is situated within the origin of  $d$
- $p$  is authorized to move to the dest. of  $d$
- $p$  is not engaged with another door

$$\begin{aligned} \text{admitted}(p, d) &\equiv \\ \text{ORG}(d) = c(p) &\wedge \\ p \mapsto \text{DST}(d) \in A &\wedge \\ p \notin \text{dom}(dap) \end{aligned}$$

# A New Event (1)

Accepting a person  $p$  - GUARD :

- {
  - Given **some** person  $p$  and door  $d$
  - $d$  is neither green nor red
  - $p$  is admissible through  $d$
- ACTION : - make  $p$  authorized to pass  $d$

```
EVENT accept  $\triangleq$ 
ANY  $p, d$  WHERE
   $p \in P \wedge$ 
   $d \in DOORS \wedge$ 
   $d \notin GREEN \cup red \wedge$ 
  admitted( $p, d$ )
THEN
  DAP( $p$ ) :=  $d$ 
END
```

# A New Event (2)

Refusing a person  $p$

- GUARD :  $\begin{cases} \text{- Given some person } p \text{ and door } d \\ \text{- } d \text{ is neither green nor red} \\ \text{- } p \text{ is not admissible through } d \end{cases}$
- ACTION : - lit the red light

```
EVENT refuse  ≡  
ANY p, d WHERE  
  p ∈ P  ∧  
  d ∈ DOORS  ∧  
  d ∉ GREEN ∪ red  ∧  
  ¬admitted(p, d)  
THEN  
  red := red ∪ {d}  
END
```

# Refining Event OBSERVATION2

```
EVENT observation2  ≡  
  ANY p, l WHERE  
    p ∈ P  
    l ∈ B  
    p, l ∈ A  
    c(p) ↦ l ∈ STRUCTURE  
  THEN  
    c(p) := l  
  END
```

```
EVENT observation3  ≡  
  REFINES observation2  
  ANY d WHERE  
    d ∈ GREEN  
  THEN  
    C(DAP-1(d)) := DST(d)  
    DAP := DAP ▷ {d}  
  END
```

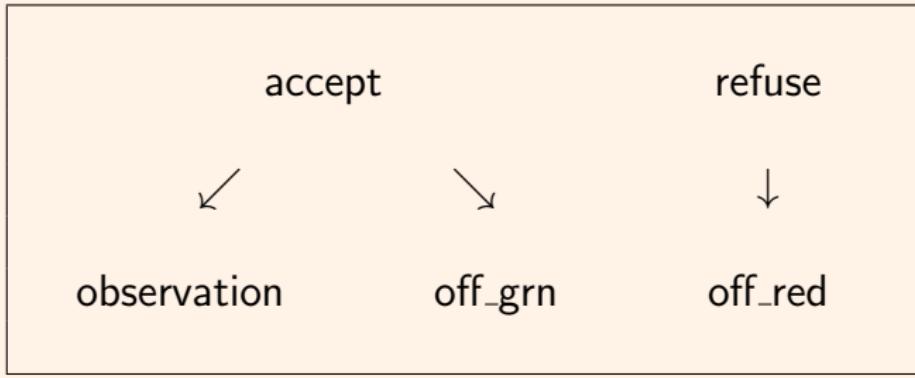
# New Event (3)

Turning lights off

```
EVENT off_grn  ≡  
  ANY d WHERE  
    d ∈ GREEN  
  THEN  
    DAP := DAP ▷ {d}  
  END
```

```
EVENT off_red  ≡  
  ANY d WHERE  
    d ∈ red  
  THEN  
    red := red - {d}  
  END
```

## Synchronization



- Event observation is a **correct refinement** : OK
- Other events **refine skip** : OK
- Event observation **does not deadlock more** : OK
- New events **do not take control indefinitely** : FAILURE

## DANGER

- People without the required authorizations try indefinitely to enter some locations.
- Other people with the authorization always change mind at the last moment.

## SOLUTIONS

- Make such practice impossible ???
- Card Readers can “swallow” a card

D3 : The system we are going to construct will not prevent people from **blocking doors indefinitely** :

- either by trying indefinitely to enter places into which they are **not authorized to enter**,
- or by indefinitely abandoning “on the way” their intention to enter the places in which they are in fact **authorized to enter**.

A decision

D4 : Each card reader is supposed **to stay blocked**

between :

- the **sending** of a card to the system
- the **reception** of an acknowledgement.

## Third Refinement : Model Extension

The set  $BLR$  of blocked Card Readers

The set  $mCard$  of messages sent by Card Readers

The set  $mAckn$  of acknowledgment messages

$$BLR \subseteq \text{DOORS}$$

$$mCard \in \text{DOORS} \rightarrow P$$

$$mAckn \subseteq \text{DOORS}$$

## Third Refinement : Invariant

$\text{dom}(mCard)$ , **GREEN**, *red*,  $mAckn$  **partition** *BLR*

$\text{dom}(mCard) \cup \text{GREEN} \cup \text{red} \cup mAckn = \text{BLR}$

$\text{dom}(mCard) \cap (\text{GREEN} \cup \text{red} \cup mAckn) = \emptyset$

$mAckn \cap (\text{GREEN} \cup \text{red}) = \emptyset$

# Events (1)

```
EVENT CARD  ≡  
ANY  $p, d$   
WHERE  
     $p \in P$   
     $d \in \text{DOORS} - BLR$   
THEN  
     $BLR := BLR \cup \{d\}$   
     $mCard := mCard \cup \{d \mapsto p\}$   
END
```

# Events (2)

```
EVENT accept3  ≡  
  ANY p, d  
  WHERE  
    p ∈ P  
    d ∈ DOORS  
    d ∉ GREEN ∪ red  
    admitted (p, d)  
  THEN  
    DAP(p) := d  
  END
```

```
EVENT accept4  ≡  
  REFINES accept3  
  ANY p, d  
  WHERE  
    d ↦ p ∈ mCard  
    admitted (p, d)  
  THEN  
    DAP(p) := d  
    mCard := mCard - {d ↦ p}  
  END
```

# Events (3)

```
EVENT refuse4  ≡  
REFINES  refuse3  
ANY p, d  
WHERE  
     $d \mapsto p \in mCard$   
     $\neg \text{admitted}(p, d)$   
THEN  
     $red := red \cup \{d\}$   
     $mCard := mCard - \{d \mapsto p\}$   
END
```

# Events (4)

```
EVENT observation4 ≡  
REFINES observation3  
ANY d  
WHERE  
    d ∈ GREEN  
THEN  
    C(DAP-1(d)) := DST(d)  
    DAP := DAP ▷ {d}  
    mAckn := mAckn ∪ {d}  
END
```

# Events (5)

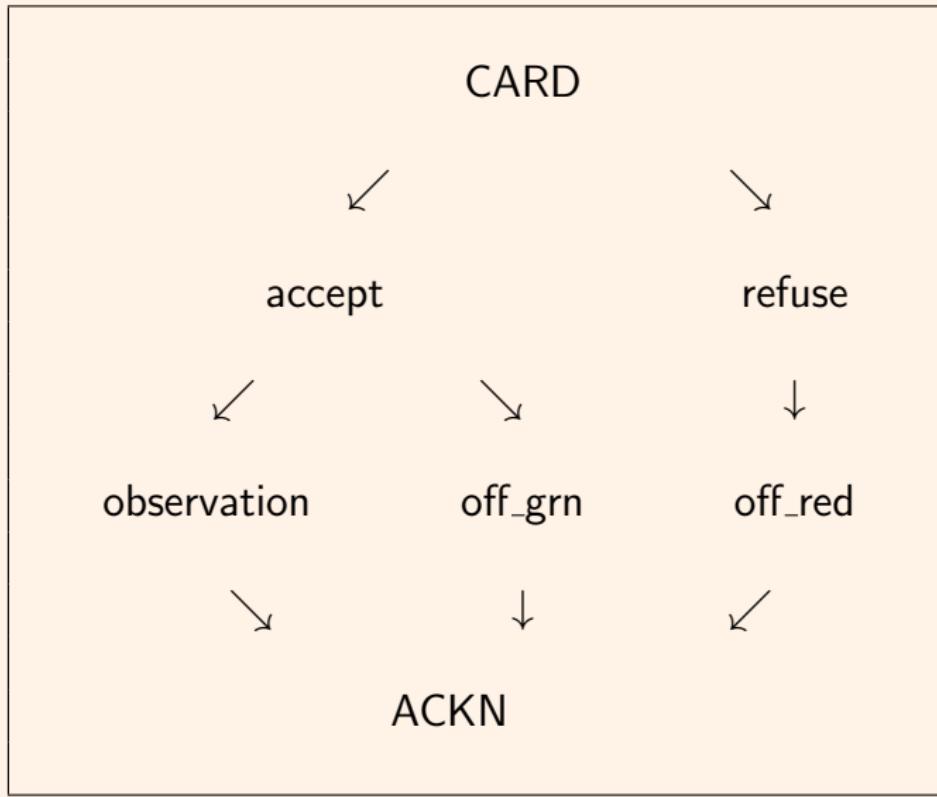
```
EVENT off_grn  ≡  
  ANY d WHERE  
    d ∈ GREEN  
  THEN  
    DAP := DAP ▷ {d}  
    mAckn := mAckn ∪ {d}  
  END
```

```
EVENT off_red  ≡  
  ANY d WHERE  
    d ∈ red  
  THEN  
    red := red - {d}  
    mAckn := mAckn ∪ {d}  
  END
```

# Events (6)

```
EVENT ACKN  ≡  
  ANY d WHERE  
    d ∈ mAckn  
  THEN  
    BLR := BLR - {d}  
    mAckn := mAckn - {d}  
  END
```

# Synchronization



# Fourth Refinement : Physical Doors and Lights

## Decisions

D5 : When a door has been cleared, it **blocks itself** automatically

without any intervention from the control system.

D6 : Each door incorporates a **local clock** for

- the extinction of the **green light after 30 sec.**
- the extinction of the **red light after 2 sec.**

## Extending the Model : the **Green Chain** (1)

The set  $mAccept$  of acceptance messages (to doors)

The set  $GRN$  of physical green doors

The set  $mPass$  of passing messages (from doors)

The set  $mOff\_grn$  of messages (from doors)

$$mAccept \subseteq \text{DOORS}$$

$$GRN \subseteq \text{DOORS}$$

$$mPass \subseteq \text{DOORS}$$

$$mOff\_grn \subseteq \text{DOORS}$$

## Extending the Model : the **Green Chain** (2)

$mAccept, GRN, mPass, mOff\_grn$  **partition**  $grn$

$$mAccept \cup GRN \cup mPass \cup mOff\_grn = grn$$

$$mAccept \cap (GRN \cup mPass \cup mOff\_grn) = \emptyset$$

$$GRN \cap (mPass \cup mOff\_grn) = \emptyset$$

$$mPass \cap mOff\_grn = \emptyset$$

# Extending the Model : the Red Chain (1)

The set  $mRefuse$  of messages (to doors)

The set  $RED$  of physical red doors

The set  $mOff\_red$  of messages (from doors)

$$mRefuse \subseteq \text{DOORS}$$

$$RED \subseteq \text{DOORS}$$

$$mOff\_red \subseteq \text{DOORS}$$

## Extending the Model : the Red Chain (2)

$mRefuse, RED, mOff\_red$  partition  $red$

$$mRefuse \cup RED \cup mOff\_red = red$$

$$mRefuse \cap (RED \cup mOff\_red) = \emptyset$$

$$RED \cap mOff\_red = \emptyset$$

# Events (1)

```
EVENT accept  ≡  
ANY p, d WHERE  
d, p ∈ mCard  ∧  
admitted(p, d)  
THEN  
    DAP(p) := d  
    mCard := mCard - {d ↦ p}  
    mAcept := mAcept ∪ {d}  
END
```

## Events (2)

```
EVENT ACCEPT  $\hat{=}$ 
ANY  $d$  WHERE
 $d \in mAccept$ 
THEN
 $GRN := GRN \cup \{d\}$ 
 $mAccept := mAccept - \{d\}$ 
END
```

# Events (3)

```
EVENT PASS  ≡  
ANY d WHERE  
d ∈ GRN  
THEN  
GRN := GRN – {d}  
mPass := mPass ∪ {d}  
END
```

# Events (4)

```
EVENT observation5  ≡  
REFINES  observation4  ANY d WHERE  
d ∈ mPass  
THEN  
    C(DAP-1(d)) := DST(d)  
    DAP := DAP ▷ {d}  
    mAckn := mAckn ∪ {d}  
    mPass := mPass - {d}  
END
```

# Events (5)

```
EVENT OFF_GRN  ≡  
  ANY d WHERE  
    d ∈ GRN  
  THEN  
    GRN := GRN - {d}  
    mOff_grn := mOff_grn ∪ {d}  
  END
```

# Events (6)

```
EVENT off_grn  ≡  
  ANY d WHERE  
    d ∈ mOff_grn  
  THEN  
    DAP := DAP ▷ {d}  
    mAckn := mAckn ∪ {d}  
    mOff_grn := mOff_grn - {d}  
  END
```

# Events (7)

```
EVENT refuse  ≡  
  ANY  $p, d$  WHERE  
     $d, p \in mCard \wedge$   
     $\neg \text{admitted}(p, d)$   
  THEN  
     $red := red \cup \{d\}$   
     $mCard := mCard - \{d \mapsto p\}$   
     $mRefuse := mRefuse \cup \{d\}$   
  END
```

# Events (8)

```
EVENT REFUSE  $\hat{=}$ 
ANY  $d$  WHERE
 $d \in mRefuse$ 
THEN
 $RED := RED \cup \{d\}$ 
 $mRefuse := mRefuse - \{d\}$ 
END
```

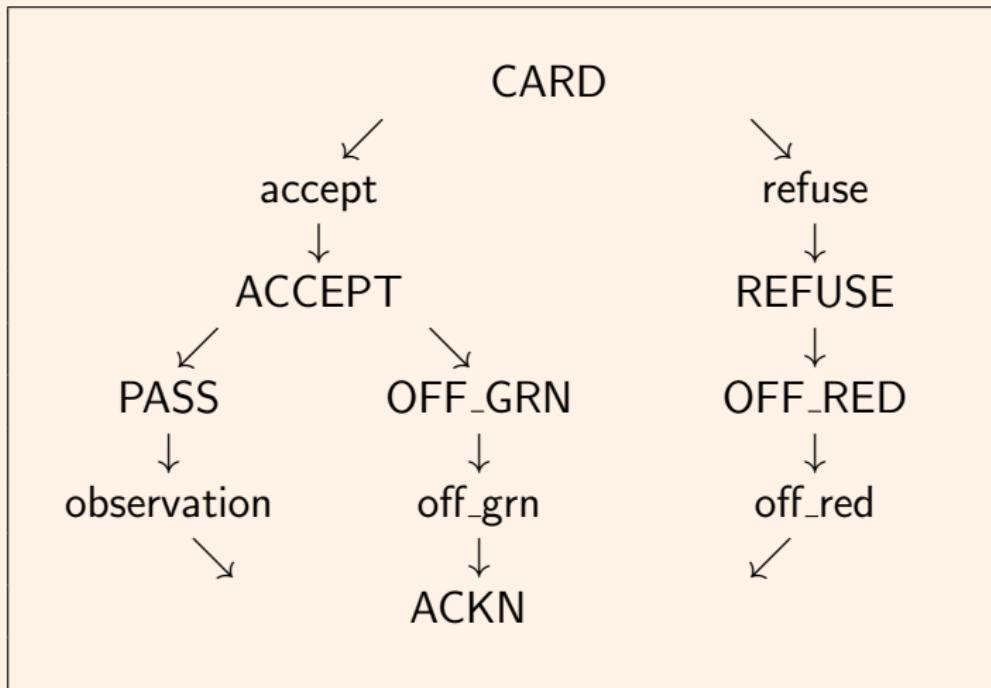
# Events (9)

```
EVENT OFF_RED  ≡  
  ANY d WHERE  
    d ∈ RED  
  THEN  
    RED := RED - {d}  
    mOff.red := mOff.red ∪ {d}  
  END
```

# Events (10)

```
EVENT off_red  ≡  
  ANY d WHERE  
     $d \in mOff\_red$   
  THEN  
     $red := red - \{d\}$   
     $mAckn := mAckn \cup \{d\}$   
     $mOff\_red := mOff\_red - \{d\}$   
  END
```

# Synchronization



# Communications

Hardware	Network			Software
CARD	→	$mCard$	→	$\begin{cases} \text{accept (1)} \\ \text{refuse (2)} \end{cases}$
ACCEPT	←	$mAccept$	←	(1)
PASS	→	$mPass$	→	observation (3)
OFF_GRN	→	$mOff\_grn$	→	off_grn (3)
REFUSE	←	$mRefuse$	←	(2)
OFF_RED	→	$mOff\_red$	→	off_red (3)
ACKN	←	$mAckn$	←	(3)

# Decomposition (1)

## Software Data

$$\begin{aligned} aut &\in P \leftrightarrow B \\ ORG &\in DOORS \rightarrow B \\ DST &\in DOORS \rightarrow B \\ A &\subseteq A; DST^{-1}; ORG \\ C &\in P \rightarrow B \end{aligned}$$
$$\begin{aligned} dap &\in P \rightsquigarrow DOORS \\ red &\subseteq DOORS \end{aligned}$$

# Decomposition (2)

## Network data

$mCard \in \text{DOORS} \leftrightarrow P$

$mAckn \subseteq \text{DOORS}$

$mAccept \subseteq \text{DOORS}$

$mPass \subseteq \text{DOORS}$

$mOff\_grn \subseteq \text{DOORS}$

$mRefuse \subseteq \text{DOORS}$

$mOff\_red \subseteq \text{DOORS}$

# Decomposition (3)

“Physical” Data

$$BLR \subseteq \text{DOORS}$$

$$GRN \subseteq \text{DOORS}$$

$$RED \subseteq \text{DOORS}$$

EVENT `test_soft( $p, d$ )`

EVENT `accept_soft( $p, d$ )`

EVENT `refuse_soft( $d$ )`

EVENT `pass_soft( $d$ )`

EVENT `off_grn_soft( $d$ )`

EVENT `off_red_soft( $d$ )`

# Physical Operations

$(p, d) \leftarrow \text{CARD\_HARD}$

$\text{ACCEPT\_HARD}(d)$

$\text{REFUSE\_HARD}(d)$

$d \leftarrow \text{PASS\_HARD}$

$d \leftarrow \text{OFF\_GRN\_HARD}$

$d \leftarrow \text{OFF\_RED\_HARD}$

$\text{ACKN\_HARD}(d)$

# Network Software Operations

$(p, d) \leftarrow \text{read\_card}$

$\text{write\_accept}(d)$

$\text{write\_refuse}(d)$

$d \leftarrow \text{read\_pass}$

$d \leftarrow \text{read\_off\_grn}$

$d \leftarrow \text{read\_off\_red}$

$\text{write\_ackn}(d)$

# Network Physical Operations

```
SEND_CARD( $p, d$ )  
 $d \leftarrow RCV\_ACCEPT$   
 $d \leftarrow RCV\_REFUSE$   
SEND_PASS( $d$ )  
SEND_OFF_GRN( $d$ )  
SEND_OFF_RED( $d$ )  
 $d \leftarrow RCV\_ACKN$ 
```

```

EVENT CARD  ≡
VAR p, d IN
(p, d) ← READ_HARD;
SEND_CARD(p, d)
END

```

```

EVENT accept_refuse  ≡
VAR p, d, b IN
(p, d) ← read_card;
b ← EVENT test_soft(p, d);
IF b = true THEN EVENT accept_soft(p, d); write_accept(d)
ELSE EVENT refuse_soft(d); write_refuse(d) END
END

```

```

EVENT ACCEPT  ≡
VAR d IN
d ← RCV_ACCEPT;
ACCEPT_HARD(d)
END

```

```

EVENT REFUSE  ≡
VAR d IN
d ← RCV_REFUSE;
REFUSE_HARD(q)
END

```

```
EVENT PASS  ≡  
VAR d IN  
d ← PASS_HARD;  
SEND_PASS(d)  
END
```

```
EVENT OFF_GRN  ≡  
VAR d IN  
d ← OFF_GRN_HARD;  
SEND_OFF_GRN(d)  
END
```

```
EVENT OFF_RED  ≡  
VAR d IN  
d ← OFF_RED_HARD;  
SEND_OFF_RED(d)  
END
```

```
EVENT observation  ≡  
VAR d IN  
d ← read_pass;  
EVENT pass_soft(d);  
write_ackn(d)  
END
```

```
EVENT off_grn  ≡  
VAR d IN  
d ← read_off_grn;  
EVENT off_grn_soft(d);  
write_ackn(d)  
END
```

```
EVENT off_red  ≡  
VAR d IN  
d ← read_off_red;  
EVENT off_red_soft(d);  
write_ackn(d)  
END
```

```
EVENT ACKN  ≡  
VAR d IN  
d ← RCV_ACKN;  
ACKN_HARD(d)  
END
```

## 22 Properties et 6 “System” Decisions - One Problem Generalization

- Access between locations
- One Negative Choice :
- Possible Card Readers Obstructions
- Three Physical Decisions
- Automatic Blocking of Doors
- Automatic Blocking of Card Readers
- Setting up of Clocks on Doors
- The overall development required 183 proofs
- 147 automatic (80%)
- 36 interactive

# Current Summary

- ① Refinement of models
- ② Summary on Event-B
- ③ The Access Control
- ④ Conclusion

# Conclusion

- Identify an abstract model
- Identify constants and states
- Identify components
- Plan the refinement
- Start as long as the model is not well defined !