



# General Summary

- 1 Refinement of models
- 2 Summary on Event-B
- 3 Case Study The Access Control (J.-R. Abrial)
- 4 Conclusion

# Summary

- 1 Refinement of models
- 2 Summary on Event-B
- 3 Case Study The Access Control (J.-R. Abrial)
- 4 Conclusion

- 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 Case Study The Access Control (J.-R. Abrial)
- 4 Conclusion

# Events as Relations over Variables Values

- Each variable  $V$  has a current value  $v$ , a next value  $v'$
- Each event  $e$  over variables  $V$  is defined by a relation over  $v$  and  $v'$  denoted  $BA(e)(v, v')$ .
- An event  $e$  has local parameters, variables, guards and actions.
- Events *observe* changes over state variables and changes can be related to code execution or to physical phenomena.

## Simple Form of an Event

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

```

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

```

where

- $\langle event\_name \rangle$  is an identifier
- $\langle condition \rangle$  is the firing condition of the event
- $\langle action \rangle$  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

- $\langle event\_name \rangle$  is an identifier
- $\langle variable \rangle$  is a (list of) variable(s)
- $\langle condition \rangle$  is the firing condition of the event
- $\langle action \rangle$  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 :  $\begin{matrix} T \\ \dots \\ U \end{matrix}$

$$\text{INVARIANT} \wedge \text{GUARD} \implies \text{ACTION establishes INVARIANT}$$

# 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  $\hat{=}$ 
  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**  $e \triangleq$   
**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  $\hat{=}$   
  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 (2)

# 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 (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 :

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

- $BA(ae)(x, x') \hat{=} G(x) \wedge x' = E(x)$
- $BA(ce)(y, y') \hat{=} H(y) \wedge y' = F(y)$

# Correct Refinement Verification (2)

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

**EVENT**  $ae \triangleq$   
**ANY**  $v$  **WHERE**  
 $G(x, v)$   
**THEN**  
 $x := E(x, v)$   
**END**

**EVENT**  $ce \triangleq$   
**ANY**  $w$  **WHERE**  
 $H(y, w)$   
**THEN**  
 $y := F(y, w)$   
**END**

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

- $BA(ae)(x, x') \triangleq \exists v. G(x, v) \wedge x' = E(x)$
- $BA(ce)(y, y') \triangleq \exists w. H(y, w) \wedge y' = F(y)$

# Correct Refinement Verification (3)

- Given a NEW event

**EVENT**  $ne \hat{=}$   
**WHEN**  
     $H(y)$   
**THEN**  
     $y := F(y)$   
**END**

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

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

- $BA(ne)(y, y') \hat{=} H(y) \wedge y' = F(y)$

## Current Summary

- 1 Refinement of models
- 2 Summary on Event-B
- 3 Case Study The Access Control (J.-R. Abrial)
- 4 Conclusion

# A Case Study by J.-R. Abrial

- 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**)

A diagram of a turnstile. On the left, a vertical line represents the turnstile body, with a horizontal line at the bottom. To its right, a vertical rectangle represents the control panel. Inside this panel, there are two circles: a green one on top and a red one below it. A horizontal line is positioned below the red circle. Three lines radiate from the right side of the turnstile body towards the control panel. The word "Turnstyle" is written to the left of the turnstile body.





## 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

persons = { p1, p2, p3 }  
locations = { l1, l2, l3, l4 }

## Authorizations

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

## Correct scenario

p1	l4	→	p1	l2	→	p1	l2	→	p1	l4	→	p1	l4
p2	l4		p2	l4		p2	l1		p2	l1		p2	l1
p3	l4		p3	l4		p3	l4		p3	l4		p3	l3

# Model (1)

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

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

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

$$A \in P \leftrightarrow B$$

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

$outside \in \mathbb{B}$

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



# 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 :  $\text{- } p \text{ jumps into } l$

```
EVENT observation1  $\hat{=}$   
  ANY  $p, l$  WHERE  
     $p \in P \wedge$   
     $l \in B \wedge$   
     $p \mapsto l \in A \wedge$   
     $c(p) \neq l$   
  THEN  
     $c(p) := l$   
  END
```



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

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



**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**

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

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

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

Event (prop. P8)

The guard is **strengthened**

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

```
EVENT observation1  $\hat{=}$   
  ANY  $p, l$  WHERE  
     $p \in P \wedge$   
     $l \in B \wedge$   
     $p \mapsto l \in A \wedge$   
     $c(p) \neq l$   
  THEN  
     $c(p) := l$   
  END
```

```
EVENT observation2  $\hat{=}$   
  REFINES observation1  
  ANY  $p, l$  WHERE  
     $p \in P \wedge$   
     $l \in B \wedge$   
     $p \mapsto l \in A \wedge$   
     $c(p) \mapsto l \in \text{STRUCTURE}$   
  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**

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

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

A

l1	l3
l1	l4
l3	l2
l4	l1
l4	l2
l4	l3

STRUCTURE

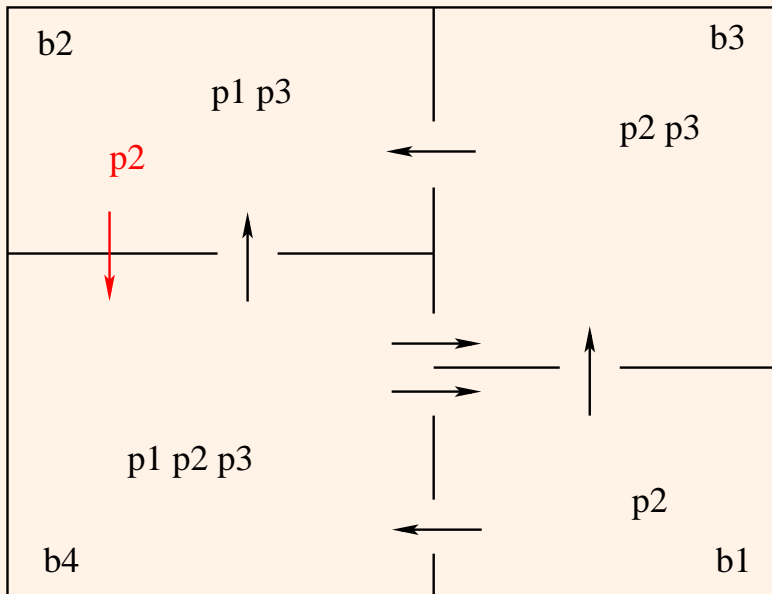
l1	l4
l2	l3
l2	l4
l3	l1
l3	l4
l4	l1

STRUCTURE<sup>-1</sup>

p1	l1	p
p1	l3	p
p1	l4	p
p2	l1	p

A; STRUCTURE

- Opening a door between l2 and l4
- Authorizing p2 to go to l2



# Solution

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

A

l1	l3
l1	l4
l2	l4
l3	l2
l4	l1
l4	l2
l4	l3

STRUCTURE

l1	l4
l2	l3
l2	l4
l3	l1
l3	l4
l4	l1
l4	l2

STRUCTURE<sup>-1</sup>

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

A; STRUCTURE

## 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**.

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

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

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)

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



## Extending the Model (2)

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

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

## 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)

$$\text{GREEN} \hat{=} \text{ran}(\text{DAP})$$

## 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)

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

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

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

# A New Event (1)

Accepting a person  $p$  - GUARD :

- $\left\{ \begin{array}{l} - \text{Given some person } p \text{ and door } d \\ - d \text{ is neither green nor red} \\ - p \text{ is admissible through } d \end{array} \right.$
- ACTION : - make  $p$  authorized to pass  $d$

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

# A New Event (2)

Refusing a person  $p$

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

```
EVENT refuse  $\hat{=}$   
  ANY  $p, d$  WHERE  
     $p \in P \wedge$   
     $d \in \text{DOORS} \wedge$   
     $d \notin \text{GREEN} \cup \text{red} \wedge$   
     $\neg \text{admitted}(p, d)$   
  THEN  
     $\text{red} := \text{red} \cup \{d\}$   
  END
```

# Refining Event OBSERVATION2

```
EVENT observation2  $\hat{=}$   
  ANY  $p, l$  WHERE  
     $p \in P$   
     $l \in B$   
     $p, l \in A$   
     $C(p) \mapsto l \in \text{STRUCTURE}$   
  THEN  
     $C(p) := l$   
  END
```

```
EVENT observation3  $\hat{=}$   
  REFINES observation2  
  ANY  $d$  WHERE  
     $d \in \text{GREEN}$   
  THEN  
     $C(\text{DAP}^{-1}(d)) := \text{DST}(d)$   
     $\text{DAP} := \text{DAP} \triangleright \{d\}$   
  END
```

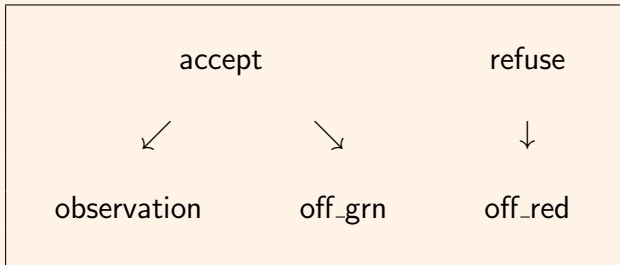


## Turning lights off

```
EVENT off_grn  $\hat{=}$   
  ANY  $d$  WHERE  
     $d \in \text{GREEN}$   
  THEN  
    DAP := DAP  $\triangleright$  { $d$ }  
  END
```

```
EVENT off_red  $\hat{=}$   
  ANY  $d$  WHERE  
     $d \in \text{red}$   
  THEN  
     $\text{red} := \text{red} - \{d\}$   
  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

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

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

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

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



# Events (1)

```
EVENT CARD  $\hat{=}$   
  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  $\hat{=}$   
  ANY  $p, d$   
  WHERE  
     $p \in P$   
     $d \in \text{DOORS}$   
     $d \notin \text{GREEN} \cup \text{red}$   
    admitted( $p, d$ )  
  THEN  
    DAP( $p$ ) :=  $d$   
  END
```

```
EVENT accept4  $\hat{=}$   
  REFINES accept3  
  ANY  $p, d$   
  WHERE  
     $d \mapsto p \in mCard$   
    admitted( $p, d$ )  
  THEN  
    DAP( $p$ ) :=  $d$   
     $mCard := mCard - \{d \mapsto p\}$   
  END
```

# Events (3)

```
EVENT refuse4  $\hat{=}$   
  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  $\hat{=}$   
  REFINES observation3  
  ANY  $d$   
  WHERE  
     $d \in \text{GREEN}$   
  THEN  
     $C(\text{DAP}^{-1}(d)) := \text{DST}(d)$   
     $\text{DAP} := \text{DAP} \triangleright \{d\}$   
     $mAckn := mAckn \cup \{d\}$   
  END
```

# Events (5)

```
EVENT off_grn  $\hat{=}$   
  ANY  $d$  WHERE  
     $d \in \text{GREEN}$   
  THEN  
     $\text{DAP} := \text{DAP} \triangleright \{d\}$   
     $mAckn := mAckn \cup \{d\}$   
  END
```

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

## Events (6)

```

EVENT ACKN  $\triangleq$ 
  ANY  $d$  WHERE
     $d \in mAckn$ 
  THEN
     $BLR := BLR - \{d\}$ 
     $mAckn := mAckn - \{d\}$ 
  END

```

# Synchronization







# 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** GREEN

$$mAccept \cup GRN \cup mPass \cup mOff\_grn = \text{GREEN}$$

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

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

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

# 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  $\hat{=}$   
  ANY  $p, d$  WHERE  
     $d \mapsto p \in mCard \wedge$   
    admitted( $p, d$ )  
  THEN  
    DAP( $p$ ) :=  $d$   
     $mCard := mCard - \{d \mapsto p\}$   
     $mAccept := mAccept \cup \{d\}$   
  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  $\hat{=}$   
  ANY  $d$  WHERE  
     $d \in GRN$   
  THEN  
     $GRN := GRN - \{d\}$   
     $mPass := mPass \cup \{d\}$   
  END
```

# Events (4)

```
EVENT observation5  $\hat{=}$   
  REFINES observation4 ANY  $d$  WHERE  
     $d \in mPass$   
  THEN  
     $C(DAP^{-1}(d)) := DST(d)$   
     $DAP := DAP \triangleright \{d\}$   
     $mAckn := mAckn \cup \{d\}$   
     $mPass := mPass - \{d\}$   
  END
```



# Events (5)

```
EVENT OFF_GRN  $\hat{=}$   
  ANY  $d$  WHERE  
     $d \in GRN$   
  THEN  
     $GRN := GRN - \{d\}$   
     $mOff-grn := mOff-grn \cup \{d\}$   
  END
```

# Events (6)

```
EVENT off_grn  $\hat{=}$   
  ANY  $d$  WHERE  
     $d \in mOff\_grn$   
  THEN  
     $DAP := DAP \triangleright \{d\}$   
     $mAckn := mAckn \cup \{d\}$   
     $mOff\_grn := mOff\_grn - \{d\}$   
  END
```

# Events (7)

```
EVENT refuse  $\hat{=}$   
  ANY  $p, d$  WHERE  
     $d \mapsto p \in mCard \wedge$   
     $\neg \text{admitted}(p, d)$   
  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  $\hat{=}$   
  ANY  $d$  WHERE  
     $d \in RED$   
  THEN  
     $RED := RED - \{d\}$   
     $mOff\_red := mOff\_red \cup \{d\}$   
  END
```

# Events (10)

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

# Synchronization



Hardware	Network			Software
CARD	→	<i>mCard</i>	→	{ accept (1) refuse (2)
ACCEPT	←	<i>mAccept</i>	←	(1)
PASS	→	<i>mPass</i>	→	observation (3)
OFF_GRN	→	<i>mOff_grn</i>	→	off_grn (3)
REFUSE	←	<i>mRefuse</i>	←	(2)
OFF_RED	→	<i>mOff_red</i>	→	off_red (3)
ACKN	←	<i>mAckn</i>	←	(3)



## Software Data

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

# Decomposition (2)

## Network data

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

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

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

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

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

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

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

## “Physical” Data

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

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

$RED \subseteq \text{DOORS}$

**EVENT** test\_soft( $p, d$ )

**EVENT**  $\text{accept\_soft}(p, d)$

**EVENT**  $\text{refuse\_soft}(d)$

**EVENT**  $\text{pass\_soft}(d)$

**EVENT**  $\text{off\_grn\_soft}(d)$

**EVENT**  $\text{off\_red\_soft}(d)$

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

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

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

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

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

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

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

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

```
write_accept( $d$ )
```

```
write_refuse( $d$ )
```

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

```
write_ackn( $d$ )
```

# Network Physical Operations

$$\text{SEND\_CARD}(p, d)$$
$$d \leftarrow \text{RCV\_ACCEPT}$$

$d \leftarrow \text{RCV\_REFUSE}$

$$\text{SEND\_PASS}(d)$$
$$\text{SEND\_OFF\_GRN}(d)$$
$$\text{SEND\_OFF\_RED}(d)$$
$$d \leftarrow \text{RCV\_ACKN}$$

```
EVENT CARD  $\hat{=}$   
  VAR  $p, d$  IN  
     $(p, d) \leftarrow \text{READ\_HARD};$   
    SEND_CARD( $p, d$ )  
  END
```

```
EVENT accept_refuse  $\hat{=}$   
  VAR  $p, d, b$  IN  
     $(p, d) \leftarrow \text{read\_card};$   
     $b \leftarrow \text{EVENT test\_soft}(p, d);$   
    IF  $b = \text{true}$  THEN EVENT accept_soft( $p, d$ ); write_accept( $d$ )  
    ELSE EVENT refuse_soft( $d$ ); write_refuse( $d$ ) END  
  END
```

```
EVENT ACCEPT  $\hat{=}$   
  VAR  $d$  IN  
     $d \leftarrow \text{RCV\_ACCEPT};$   
    ACCEPT_HARD( $d$ )  
  END
```

```
EVENT REFUSE  $\hat{=}$   
  VAR  $d$  IN  
     $d \leftarrow \text{RCV\_REFUSE};$   
    REFUSE_HARD( $q$ )  
  END
```



```

EVENT PASS  $\hat{=}$ 
  VAR  $d$  IN
     $d \leftarrow$  PASS_HARD;
    SEND_PASS( $d$ )
  END

```

```

EVENT OFF_GRN  $\hat{=}$ 
  VAR  $d$  IN
     $d \leftarrow$  OFF_GRN_HARD;
    SEND_OFF_GRN( $d$ )
  END

```

```

EVENT OFF_RED  $\hat{=}$ 
  VAR  $d$  IN
     $d \leftarrow$  OFF_RED_HARD;
    SEND_OFF_RED( $d$ )
  END

```

```

EVENT observation  $\hat{=}$ 
  VAR  $d$  IN
     $d \leftarrow$  read_pass;
    EVENT pass_soft( $d$ );
    write_ackn( $d$ )
  END

```

```

EVENT off_grn  $\hat{=}$ 
  VAR  $d$  IN
     $d \leftarrow$  read_off_grn;
    EVENT off_grn_soft( $d$ );
    write_ackn( $d$ )
  END

```

```

EVENT off_red  $\hat{=}$ 
  VAR  $d$  IN
     $d \leftarrow$  read_off_red;
    EVENT off_red_soft( $d$ );
    write_ackn( $d$ )
  END

```

```

EVENT ACKN  $\hat{=}$ 
  VAR  $d$  IN
     $d \leftarrow$  RCV_ACKN;
    ACKN_HARD( $d$ )
  END

```



# Current Summary

## Conclusion

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

## Generalization of the Access Control Problem

- $A$  is a variable which can be modified by events modelling the administration of the access control model :
  - ▶ adding authorizations to a set of persons
  - ▶ removing or deleting authorizations of a set of persons
- Generalizing to other problems :
  - ▶ a set of users  $U$  has access to a set of resources  $R$ .
  - ▶ a set of rooms  $R$  is managed by a set of keycards  $K$ .
  - ▶ a set of users  $U$  has access to a set of services  $S$ .