Modelling Software-based Systems

Lecture 5 The access control problem in Event-B

Telecom Nancy (3A LE)

Dominique Méry
Telecom Nancy,Université de Lorraine

6 novembre 2024

dominique.mery@loria.fr

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

Current Summary

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

$$\begin{array}{l} < event\_name > \ \widehat{=} \\ \quad \textbf{WHEN} \\ \quad\quad < condition > \\ \quad \textbf{THEN} \\ \quad\quad < action > \\ \quad \textbf{END} \end{array}$$

where
- $< event\_name >$ is an identifier
- $< condition >$ is the firing condition of the event
- $< action >$ is a generalized substitution (parallel "assignment")

Non-deterministic Form of an Event

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

$$
\begin{array}{l}
< event\_name > \ \widehat{=} \\
\quad \textbf{ANY} \ < variable > \ \textbf{WHERE} \\
\qquad < condition > \\
\quad \textbf{THEN} \\
\qquad < action > \\
\quad \textbf{END}
\end{array}
$$

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 : $\begin{array}{c} T \\ \cdots \\ U \end{array}$

Invariant Preservation Verification (0)

$$\text{INVARIANT} \quad \wedge \quad \text{GUARD}$$
$$\Longrightarrow$$
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 :

$$I(x) \;\wedge\; G(x) \;\Longrightarrow\; I(E(x))$$

Invariant Preservation Verification (2)

- Given an event of the simple form :

EVENT e $\widehat{=}$
  WHEN
    $G(x)$
  THEN
    $x :| P(x, x')$
  END

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

$$I(x) \quad \wedge \quad G(x) \quad \wedge \quad P(x, x') \quad \Longrightarrow \quad I(x')$$

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 :

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

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 :

$$I(x) \;\wedge\; G(x,v) \;\Longrightarrow\; I(E(x,v))$$

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 $\widehat{=}$ **WHEN** $G(x)$ **THEN** $x := E(x)$ **END**	**EVENT** ce $\widehat{=}$ **WHEN** $H(y)$ **THEN** $y := F(y)$ **END**

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

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

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 :

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

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

Correct Refinement Verification (2)

- Given an abstract and a corresponding concrete event

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

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

$$I(x) \quad \wedge \quad J(x,y) \quad \wedge \quad H(y,w)$$
$$\Longrightarrow$$
$$\exists v \cdot (\, G(x,v) \quad \wedge \quad J(E(x,v), F(y,w))\,)$$

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

Correct Refinement Verification (3)

- Given a NEW event

```
EVENT ne ≙
  WHEN
    H(y)
  THEN
    y := F(y)
  END
```

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

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

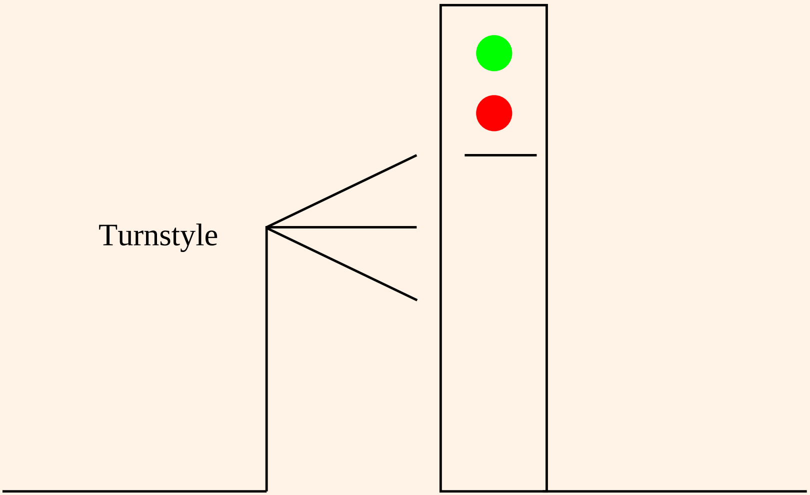
- $BA(ne)(y,y') \ \widehat{=} \ H(y) \wedge y' = E(y)$

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)


Turnstyle

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

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

Authorizations

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

Correct scenario

p1	l4
p2	l4
p3	l4

$\rightarrow$

p1	l2
p2	l4
p3	l4

$\rightarrow$

p1	l2
p2	l1
p3	l4

$\rightarrow$

p1	l4
p2	l1
p3	l4

$\rightarrow$

p1	l4
p2	l1
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

$$A \in P \leftrightarrow B$$

Model (2)

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

$$outside \in \mathrm{B}$$

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

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

$$\mathrm{c} \in \mathrm{P} \rightarrow \mathrm{B}$$

Invariant : situations compatible with auth. (prop. P5)
The function c is included in the relation A

$$\mathrm{c} \subseteq \mathrm{A}$$

A magic event which can be observed

- GUARD :
  - Given some person p and location l
  - p is authorized to be in l : $p, l \in \mathrm{A}$
  - p is not currently in l : $\mathrm{C}(p) \neq l$
- ACTION : - p jumps into l

EVENT observation1 $\widehat{=}$
  ANY p, l **WHERE**
    $p \in \mathrm{P} \quad \wedge$
    $l \in \mathrm{B} \quad \wedge$
    $p \mapsto l \in \mathrm{A} \quad \wedge$
    $\mathrm{C}(p) \neq l$
  THEN
    $\mathrm{C}(p) := l$
  END

Relation overriding

Given two relations a and b
Overriding a by b yields a new relation $a \mathbin{<\!\!+} b$

$$a \mathbin{<\!\!+} b \quad \widehat{=} \quad (\mathrm{dom}\,(b) \mathbin{\lhd\!\!\!-} a) \; \cup \; b$$

Abbreviation

$$f(x) := y \quad \widehat{=} \quad f := f \mathbin{<\!\!+} \{x \mapsto y\}$$

Invariant Preservation Proof

INVARIANT $\wedge$ GUARD

$\Longrightarrow$

ACTION establishes INVARIANT

$$
\begin{array}{l}
\mathrm{c} \subseteq \mathrm{A} \quad \wedge \\
p \in \mathrm{P} \quad \wedge \\
l \in \mathrm{B} \quad \wedge \\
p \mapsto l \ \in \ \mathrm{A} \\
\Longrightarrow \\
(\{p\} \lhd \mathrm{c}) \cup \{p \mapsto l\} \subseteq \mathrm{A}
\end{array}
$$

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

Refined Model

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

$$\textsc{structure} \in \mathrm{B} \leftrightarrow \mathrm{B}$$

$$\textsc{structure} \cap \mathrm{id}(\mathrm{B}) = \varnothing$$

Correct Refinement Verification (reminder)

Concrete events do not block more often than abstract ones

$I(x) \;\wedge\; J(x,y) \;\wedge$
disjunction of abstract guards
$\Longrightarrow$
disjunction of concrete guards

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

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

Event (prop. P8)
The guard is strengthened
The current location of p and the new location l must communicate

EVENT observation1 $\widehat{=}$
  ANY p, l **WHERE**
    $p \in \mathrm{P} \quad \wedge$
    $l \in \mathrm{B} \quad \wedge$
    $p \mapsto l \ \in \ \mathrm{A} \quad \wedge$
    $\mathrm{C}(p) \neq l$
  THEN
    $\mathrm{C}(p) \ := \ l$
  END

EVENT observation2 $\widehat{=}$
  REFINES $observation1$
  ANY p, l **WHERE**
    $p \in \mathrm{P} \quad \wedge$
    $l \in \mathrm{B} \quad \wedge$
    $p \mapsto l \ \in \ \mathrm{A} \quad \wedge$
    $\mathrm{C}(p) \mapsto l \ \in \ \mathrm{STRUCTURE}$
  THEN
    $\mathrm{C}(p) \ := \ l$
  END

Proofs

Invariant preservation : Success
Guard strengthening : Success

$$\exists (p,l) \cdot (p \mapsto l \in \mathrm{A} \wedge \mathrm{C}(p) \mapsto l \in \mathrm{STRUCTURE})$$
$$\Rightarrow$$
$$\exists (p,l) \cdot (p \mapsto l \in \mathrm{A} \wedge \mathrm{C}(p) \neq l)$$

Deadlockfreeness : Failure

$$\exists (p,l) \cdot (p \mapsto l \in \mathrm{A} \wedge \mathrm{C}(p) \neq l)$$
$$\Rightarrow$$
$$\exists (p,l) \cdot (p \mapsto l \in \mathrm{A} \wedge \mathrm{C}(p) \mapsto l \in \mathrm{STRUCTURE})$$

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.

$$\mathrm{A} \subseteq \mathrm{A} \,;\, \textsc{structure}^{-1}$$

$$p \mapsto l \in \mathrm{A} \implies \exists m \cdot (p \mapsto m \in \mathrm{A} \;\wedge\; l \mapsto m \in \textsc{structure})$$

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

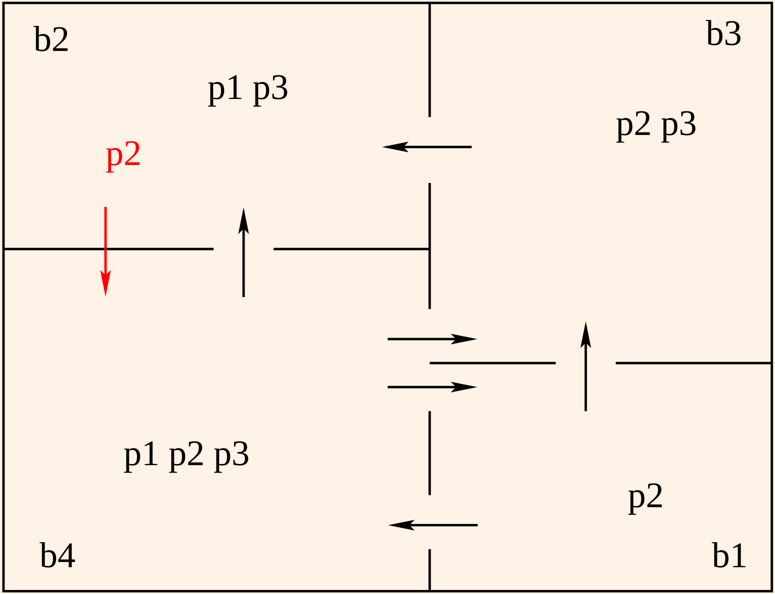
l1	l4
l2	l3
l2	l4
l3	l1
l3	l4
l4	l1

STRUCTURE^{-1}

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

A; STRUCTU

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


b2
p1 p3
b3
p2 p3
p2
p1 p2 p3
p2
b4
b1

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^{-1}

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

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

$$exit \subseteq com$$

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

$$\forall x \cdot (x \in s \implies \exists y \cdot (\, y \in s \ \wedge\ (x, y) \in exit\,)\,)$$
$$\implies$$
$$s = \varnothing$$

$exit$ 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.

$$\mathrm{A} \rhd \{outside\} \ \subseteq \ \mathrm{A} \, ; exit^{-1}$$

$$\begin{array}{l} \quad p \mapsto l \in \mathrm{A} \ \wedge \\ \quad l \neq outside \\ \Longrightarrow \\ \quad p \mapsto exit(l) \in \mathrm{A} \end{array}$$

For the experts

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

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

$$\begin{array}{l} t \subseteq \mathrm{B} \\ outside \in t \\ \forall (x,y) \cdot ((x \mapsto y) \in exit \ \wedge \ y \in t \implies x \in t) \end{array}$$
$$\implies$$
$$t = \mathrm{B}$$

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)

$$\mathrm{ORG} \in \textsc{doors} \rightarrow \mathrm{B}$$
$$\mathrm{DST} \in \textsc{doors} \rightarrow \mathrm{B}$$
$$\textsc{structure} = (\mathrm{ORG}^{-1} \,;\, \mathrm{DST})$$

Extending the Model (2)

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

$$\begin{array}{l}
\text{DAP} \in P \rightarrowtail\!\!\!\!\!+\!\!\!\rightarrow \text{DOORS} \\
(\text{DAP} \,;\, ORG) \subseteq \text{C} \\
(\text{DAP} \,;\, DST) \subseteq A
\end{array}$$

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)

$$\mathrm{GREEN} \;\widehat{=}\; \mathsf{ran}\,(\mathrm{DAP})$$

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)

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

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

$$
\begin{array}{l}
\text{admitted}\,(p, d) \quad \widehat{=} \\
\quad \mathrm{ORG}(d) = \mathrm{C}(p) \quad \wedge \\
\quad p \mapsto \mathrm{DST}(d) \in \mathrm{A} \quad \wedge \\
\quad p \notin \text{dom}\,(dap)
\end{array}
$$

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 $\widehat{=}$
  ANY p, d **WHERE**
    $p \in \mathrm{P} \wedge$
    $d \in \textsc{doors} \wedge$
    $d \notin \textsc{green} \cup \mathit{red} \wedge$
    admitted (p, d)
  THEN
    $\textsc{dap}(p) := d$
  END

A New Event (2)

Refusing a person p

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

EVENT refuse $\widehat{=}$
  ANY p, d **WHERE**
    $p \in \mathrm{P} \quad \wedge$
    $d \in \mathrm{DOORS} \quad \wedge$
    $d \notin \mathrm{GREEN} \cup red \quad \wedge$
    $\neg$ admitted (p, d)
  THEN
    $red := red \cup \{d\}$
  END

Refining Event OBSERVATION2

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

EVENT observation3 $\widehat{=}$
REFINES observation2
ANY d **WHERE**
$d \in \mathrm{GREEN}$
THEN
$\mathrm{C}(\mathrm{DAP}^{-1}(d)) := \mathrm{DST}(d)$
$\mathrm{DAP} := \mathrm{DAP} \rhd \{d\}$
END

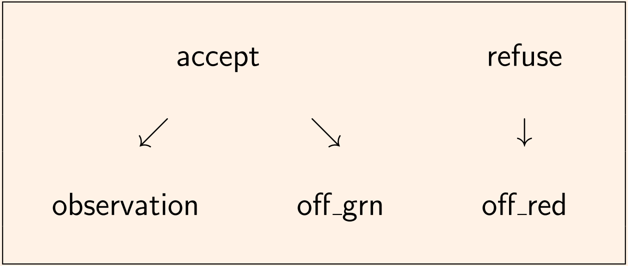
New Event (3)

Turning lights off

EVENT off_grn $\widehat{=}$
  ANY d **WHERE**
    $d \in \text{GREEN}$
  THEN
    $\text{DAP} := \text{DAP} \trianglerighteq \{d\}$
  END

EVENT off_red $\widehat{=}$
  ANY d **WHERE**
    $d \in red$
  THEN
    $red := red - \{d\}$
  END

Synchronization



Proofs

- 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

Permanent Obstruction of Card Readers

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

Final Decision

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¨.

Third Refinement : Introducing Card Readers

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

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

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

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

Third Refinement : Invariant

dom $(mCard)$, GREEN, red, $mAckn$ partition BLR

$$\mathsf{dom}\,(mCard) \cup \mathrm{GREEN} \cup red \cup mAckn = BLR$$

$$\mathsf{dom}\,(mCard) \cap (\mathrm{GREEN} \cup red \cup mAckn) = \varnothing$$

$$mAckn \cap (\mathrm{GREEN} \cup red) = \varnothing$$

Events (1)

EVENT CARD $\widehat{=}$
ANY p, d
WHERE
$p \in \mathrm{P}$
$d \in \textsc{doors} - BLR$
THEN
$BLR := BLR \cup \{d\}$
$mCard := mCard \cup \{d \mapsto p\}$
END

Events (2)

EVENT accept3 $\widehat{=}$
  ANY p, d
  WHERE
    $p \in \mathrm{P}$
    $d \in \textsc{doors}$
    $d \notin \textsc{green} \cup red$
    admitted (p, d)
  THEN
    $\textsc{dap}(p) := d$
  END

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

Events (3)

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

Events (4)

EVENT observation4 $\widehat{=}$
  REFINES observation3
  ANY d
  WHERE
    $d \in \text{GREEN}$
  THEN
    $\text{C}(\text{DAP}^{-1}(d)) := \text{DST}(d)$
    $\text{DAP} := \text{DAP} \rhd\!\!\!> \{d\}$
    $mAckn := mAckn \cup \{d\}$
  END

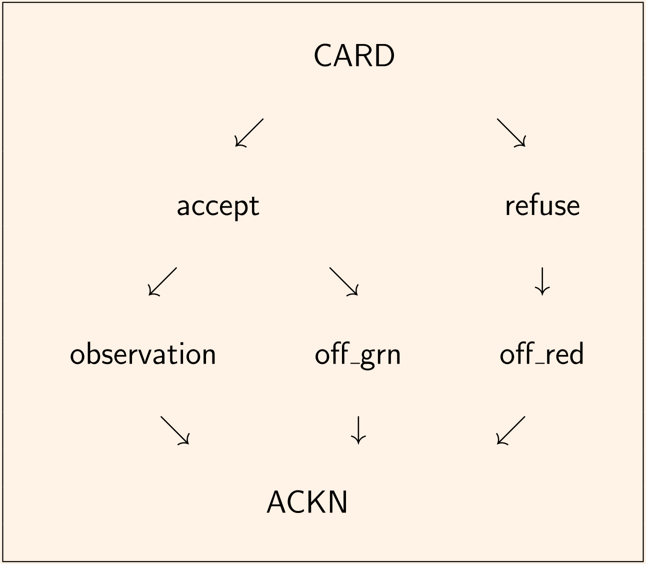
Events (5)

EVENT off_grn $\widehat{=}$
  ANY d **WHERE**
    $d \in \text{GREEN}$
  THEN
    $\text{DAP} := \text{DAP} \rhd \{d\}$
    $mAckn := mAckn \cup \{d\}$
  END

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

Events (6)

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


CARD
accept
refuse
observation
off_grn
off_red
ACKN

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 GREEN

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

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

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

$$mPass \cap mOff\_grn = \varnothing$$

Extending the Model : the Red Chain (1)

The set $mRefuse$ of messages (to doors)
The set RED of phyical 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

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

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

$$RED \cap mOff\_red = \varnothing$$

Events (1)

EVENT accept $\widehat{=}$
  ANY p, d **WHERE**
    $d \mapsto p \in mCard \quad \wedge$
    admitted (p, d)
  THEN
    $\text{DAP}(p) \ := \ d$
    $mCard \ := \ mCard - \{d \mapsto p\}$
    $mAccept \ := \ mAccept \cup \{d\}$
  END

Events (2)

EVENT ACCEPT $\widehat{=}$
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 $\widehat{=}$
REFINES observation4 **ANY** d **WHERE**
$d \in mPass$
THEN
$\textsc{c}(\textsc{dap}^{-1}(d)) := \mathrm{DST}(d)$
$\textsc{dap} := \textsc{dap} \rhd\!\!\!\!- \{d\}$
$mAckn := mAckn \cup \{d\}$
$mPass := mPass - \{d\}$
END

Events (5)

EVENT OFF_GRN $\widehat{=}$
  ANY d **WHERE**
    $d \in GRN$
  THEN
    $GRN := GRN - \{d\}$
    $mOff\_grn := mOff\_grn \cup \{d\}$
  END

Events (6)

EVENT off_grn $\widehat{=}$
  ANY d **WHERE**
    $d \in mOff\_grn$
  THEN
    $\mathrm{DAP} := \mathrm{DAP} \rhd \{d\}$
    $mAckn := mAckn \cup \{d\}$
    $mOff\_grn := mOff\_grn - \{d\}$
  END

Events (7)

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

Events (8)

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

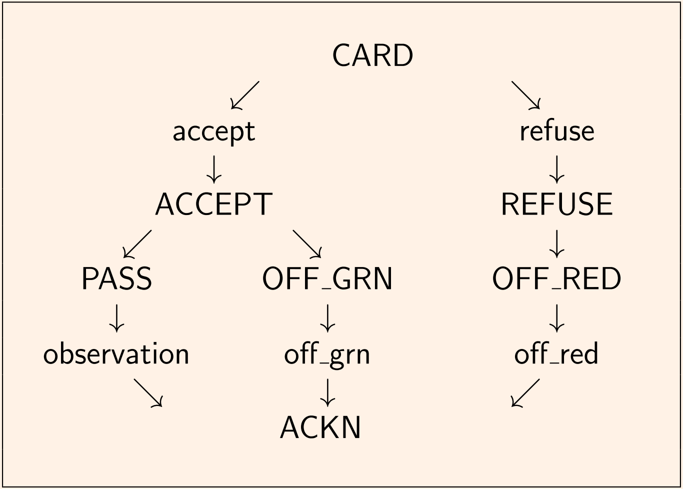
Events (9)

EVENT OFF_RED $\widehat{=}$
  ANY d **WHERE**
    $d \in RED$
  THEN
    $RED := RED - \{d\}$
    $mOff\_red := mOff\_red \cup \{d\}$
  END

Events (10)

EVENT off_red $\widehat{=}$
  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	$\rightarrow$	$mCard$	$\rightarrow$	$\begin{cases} \text{accept (1)} \\ \text{refuse (2)} \end{cases}$
ACCEPT	$\leftarrow$	$mAccept$	$\leftarrow$	(1)
PASS	$\rightarrow$	$mPass$	$\rightarrow$	observation (3)
OFF_GRN	$\rightarrow$	$mOff\_grn$	$\rightarrow$	off_grn (3)
REFUSE	$\leftarrow$	$mRefuse$	$\leftarrow$	(2)
OFF_RED	$\rightarrow$	$mOff\_red$	$\rightarrow$	off_red (3)
ACKN	$\leftarrow$	$mAckn$	$\leftarrow$	(3)

Decomposition (1)

Software Data

$$
\begin{array}{l}
aut \in \mathrm{P} \leftrightarrow \mathrm{B} \\
\mathrm{ORG} \in \textsc{doors} \rightarrow \mathrm{B} \\
\mathrm{DST} \in \textsc{doors} \rightarrow \mathrm{B} \\
\mathrm{A} \subseteq \mathrm{A}; \mathrm{DST}^{-1}; \mathrm{ORG} \\
\mathrm{C} \in \mathrm{P} \rightarrow \mathrm{B} \\
\\
dap \in \mathrm{P} \rightarrowtail\!\!\!\!\!\!\!\!\leftrightarrow \textsc{doors} \\
red \subseteq \textsc{doors}
\end{array}
$$

Decomposition (2)

Network data

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

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

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

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

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

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

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

Decomposition (3)

"Physical" Data

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

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

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

Software Operations

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

$(p, d) \longleftarrow$ CARD_HARD

ACCEPT_HARD(d)

REFUSE_HARD(d)

$d \longleftarrow$ PASS_HARD

$d \longleftarrow$ OFF_GRN_HARD

$d \longleftarrow$ OFF_RED_HARD

ACKN_HARD(d)

Network Software Operations

$(p, d) \longleftarrow$ read_card

write_accept(d)

write_refuse(d)

$d \longleftarrow$ read_pass

$d \longleftarrow$ read_off_grn

$d \longleftarrow$ read_off_red

write_ackn(d)

Network Physical Operations

SEND_CARD(p, d)

d $\longleftarrow$ RCV_ACCEPT

d $\longleftarrow$ RCV_REFUSE

SEND_PASS(d)

SEND_OFF_GRN(d)

SEND_OFF_RED(d)

d $\longleftarrow$ 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
```

Conclusion

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

Generalization of the Access Control Problem

- A is a variable which can be modified by events mdeolling 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.